CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Evaluation of models for forecasting traffic speed and classifying traffic delay in European cities

Exploring the ability of classification models to make predictions on unseen city data

Master's thesis in Computer science and engineering

ARMAND GHAFFARPOUR, OSCAR HANSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Evaluation of models for forecasting traffic speed and classifying traffic delay in European cities

Exploring the ability of classification models to make predictions on unseen city data

ARMAND GHAFFARPOUR, OSCAR HANSSON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Evaluation of models for forecasting traffic speed and classifying traffic delay in European cities
Exploring the ability of classification models to make predictions on unseen city data
ARMAND GHAFFARPOUR, OSCAR HANSSON

Supervisor: Vilhelm Verendel, Computer Science and Engineering
Examiner: Daniel Johansson, Space, Earth and Environment

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Evaluation of models for forecasting traffic speed and classifying traffic delay in European cities Exploring the ability of classification models to make predictions on unseen city data

ARMAND GHAFFARPOUR, OSCAR HANSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Congestion and traffic delays are big challenges that cities face today. In this thesis, we use traffic speed data from 2018 for forecasting traffic speeds and classifying traffic delays. The data consists of 15-minute time-intervals and covers 15 European cities. As a starting point we investigate if machine learning can be used to forecast traffic speeds, specifically if more advanced forecasting models improve upon simple statistical models. It is also investigated if weather variables such as precipitation and temperature improve the forecasting models. Forecasting models require continuous traffic speed input data in order to make predictions about the future. To create more generic models capable of making predictions regardless of the current traffic speed, binary classification models are used with the goal of classifying if there is a delay or not at some point in time. The traffic speed data is transformed into a binary value, 1 for when there is a delay and 0 for when there is no delay. This is used as the target variable of the models. For the classification models, we use temporal features, weather features and graph-related centrality features that describe the road location. We also take into account the infrastructure along a road as well as the area around a road that might have an impact on traffic congestion. A larger portion of this work is focused on the binary classification with regards to traffic delay. The classification is done to answer three questions. What are the most important features for creating classification models? How well do the models perform on untrained city data? Do the models improve when increasing the number of training cities?

Regardless if weather data was used in conjunction with traffic speed data as input to the model, more advanced forecasting models didn't improve the performance significantly. With regards to the classification results, the most important features were found to be related to public transportation, where bus stops were the most prominent feature followed by schools. Moreover, it was shown that generalizing the trained models to new city data was indeed possible and that it performed better than a random classifier (a classifier that guesses the class for an input). Finally the results showed an overall increase in performance of the classifiers when increasing the number of training cities but more work is needed for optimizing the models in the future.

Keywords: computer science, thesis, traffic data, machine learning, classification, forecasting, congestion, time series.

Acknowledgements

We would like to thank our supervisor Vilhelm Verendel for his advise and insights throughout the course of this thesis. His guidance was vital for its success. We would also like to thank our examiner Daniel Johansson for his time and expertise regarding climate-related issues.

Armand Ghaffarpour & Oscar Hansson, Gothenburg, August 2021

Contents

List of Figures

List of Tables

1

Introduction

Congestion and traffic delays are still one of the largest challenges that big cities face today. In 2004 transportation accounted for 33% of the total CO_2 emissions in USA and 80% of these were coming from cars[12]. Furthermore, the estimated total delay caused by traffic congestion was 6 billion hours each year in the United States in 1994[13]. The societal costs of congestion are high and besides lost wages and inconvenience costs in disrupting economic activity, there are also costs of extra fuel, air pollution and many others. This makes analyzing traffic speeds very important for prediction and prevention of traffic congestion.

The area of traffic congestion prediction has been explored before. Kong, et al. looked at the use of floating car trajectory data for estimation and prediction of traffic congestion[16]. Another study conducted at the University of Texas investigated the use of deep neural networks to track traffic congestion and predict short-term traffic congestion where the congestion states were calculated using neighboring measuring stations found in California[18].

The amount of traffic delay is closely correlated to traffic speed since a low traffic speed can be translated to some measure of delay or congestion. One aim is therefore to evaluate different forecasting models, both statistical and machine learning models, in terms of predicting traffic speed. Forecasting models require continuous input data in order to make predictions about the future. This is not ideal if the goal is to use a trained model on new cities that don't have any traffic data. In that context, a classification model is more suitable. The original traffic speed data will be transformed into a measurement of delay and used as the output vector during training and evaluation of the classification models. The input vector will instead consist of features such as time of day, week day, weather etc. Focus will therefore be on developing binary classification models for determining if there is a delay at a road at a given time.

The traffic speed data used in this project was collected from 50 cities during all of 2018. Each city contains traffic data for a number of roads covered in that city. The traffic speed data for each road was gathered in 15-minute time intervals. For this thesis, 6 months of the traffic data was used from 15 European cities. The data was acquired using the HERE framework[42].

1.1 Thesis Goal

There are two main goals of this thesis work. One is to explore different statistical and machine learning models for forecasting traffic speeds. Are there any benefits to using more advanced models when forecasting traffic speeds? How will weather affect our predictions? These might be important factors to consider when building models with extreme weather in the future based on climate change. Today's world is ever-changing, as are weather conditions, temperatures and other areas related to climate change. As we move into the future this is an important factor to consider. Transportation is one of many areas in our society that will be affected. Koetse et al. presents empirical findings on the effect on transportation world-wide due to climate change[15]. Unforeseen precipitation such as heavy rain- or snowfall proved to increase the risk of fatal accidents by up to 75% according to the overview study.

Another important goal is to develop classification models for classifying traffic delay. This includes the following steps. The translation of traffic speed data into a measurement of delay. Retrieval of relevant features used for training classification models on input-output data, where the input consists of several retrieved features and the output consists of a binary class (1 if there is a delay and 0 if there is no delay). By exploring different classification models and different ways of training the models the aim is thereafter to answer the following questions:

- What features have the best predictive capabilities?
- How well does generalization of the models between cities really work? Does it only work when the cities share a lot of features such as structure, culture and technology?
- Will training on more cities improve the performance of our models on new data? If so, how much?

1.2 Limitations

Due to changes in data collection for the first six months of 2018 and the last six months we chose to exclude the later part of 2018 and only work on the first six months of data. Furthermore, only a subset of the original cities were used.

2
Theory

In this chapter the theory used during analysis, forecasting and classification will be explained. This will be done in chronological order, starting with the theory for analysis.

2.1 Evaluation strategy

This section deals with a typical evaluation strategy for model evaluation where the data is split in training and test data.

2.1.1 Baseline Model

Often when comparing models' performance some kind of model is used as a baseline for achieving a good comparison. This baseline model is often a naive implementation which often means that it wont be good enough for real-world use.

2.2 Missing Data

When working with large data sets acquired over a period of time there is often the case that some values are missing for one reason or the other. In machine learning and time series forecasting missing data are not allowed. There are different ways of handling missing data but many focus on filling in (imputing) the data in some way. A few strategies for filling in missing data are presented here.

2.2.1 Mean Imputation

A mean imputation strategy takes the mean value of the data set and uses that to replace the missing values.

2.2.2 Train and Test split

A train and test split is a way of evaluating a model and is often used in machine learning but can also be used for evaluating statistical models such as AR. The technique involves splitting a data set that will be used for modelling into a train and test subset of the original data. The train subset will be used to fit or "train" the model and the test data will be used to evaluate its performance. Usually one

allocates more data to the train subset in order to better fit the data to a given model.

2.3 Forecasting Models

This chapter explains different models commonly used for time series forecasting. The models explained are AR and Multi layer Perceptron (MLP). An error measurement for comparing and evaluating models is also explained. Before proceeding with an explanation of AR, the concepts Time series, Autocorrelation and Forecasting model will be explained. After that, the concepts of stationary time series, differencing, unit root test and white noise time series will be explained before proceeding with the rest of the forecasting models.

2.3.1 Time Series

A Time series is a representation of the data as a sequence of data-points ordered by time.

2.3.2 Autocorrelation

The auto-correlation of a variable in a time series is how that variable correlates to a lagged version of itself. A higher value indicating a higher correlation. The lag is usually expressed as a difference in time. To determine auto-correlation the following auto-correlation function $\rho = \frac{cov(X,X')}{{\sigma_X}^2}$ is used[45].

2.3.3 Forecasting Model

A forecasting model is a model applied to time series data in an attempt to predict future values of one or several variables of that time series from previously observed values.

2.3.4 Auto- Regressive Model(AR)

The most simple statistical model used in our work will be the Auto-Regressive Model. An Auto Regressive model is, as the name entails, a model where regression is performed on the same variable evaluating that variable as both input and output. A time series can be modeled by a certain number of regressed variables and an error term or white noise term ϵ. More formally it can be defined as a value y at time t, that is linearly dependent on some number of previous values $y_{t-1}, y_{t-2}, ..., y_{t-k}$. Where k is the number of previous values that y_t depends on. To this we also add a stochastic term denoted by ϵ that represents the random nature of that value. The equation for an AR(k) process is:

$$y_t = c + \sum_{i=1}^{k} \phi_i * y_{t-i} + \epsilon_t$$

Where c is a constant[38]. Note that the amount of previous values a.k.a. the lag needs to be determined. Auto-correlation is used to calculate the lag.

2.3.5 Stationary time series

A stationary time series is a series that exhibit the same properties as a stationary process. In this context a stationary process is a time series where the properties of that time series remain unchanged after shifting the series a certain number of steps t in any direction[31]. Because of this a stationary time series have constant mean, variance and auto-correlation. A stationary time series is a requirement for some statistical regression models such as AR. Wold's theorem states that any stationary time series can be approximated using a linear representation of the data. Since AR is a linear representation of the data, it needs to be stationary in order to achieve a good approximation[32].

2.3.6 Differencing

The process of converting non-stationary time series data to stationary time series data by calculating the difference between each consecutive data point and using these calculated data points instead of the previous time series.

2.3.7 Unit Root tests

A unit root test can be used to determine if a time series is stationary which otherwise might be hard to determine if the trend or other properties of the time series don't remain constant over time[39]. The Augmented Dickey-Fuller unit root test is one such test which is often used[22]. To illustrate how the test works we can take an arbitrary time series represented by an AR process with lag 1 (AR(1)):

$$y_t = \rho y_{t-1} + \epsilon_t$$

The ρ is a coefficient, t is the time index and ϵ the error term. If $|\rho| = 1$ a unit root exists and the time series is considered non-stationary, otherwise it is stationary.
Lets assume a unit root exists, with $\rho = 1$, it is possible to expand the equation and get

$$y_t = y_0 + \sum_{j=1}^{t} \epsilon_j$$

This works because ϵ is just the error term which is assumed to have a 0 mean and constant variance σ^2. Using the above equation and knowledge the expectation value (mean) and variance becomes:

$$\mathbb{E}(y_t) = y_0$$
$$\mathbb{VAR}(y_t) = \sigma^2(\rho^0 + \rho^2 + ... + \rho^{2(t-1)})$$

Since ρ is a constant 1 this can be further simplified to:

$$\mathbb{VAR}(y_t) = \sigma^2 * t$$

In this case the AR(1) process violates the constant variance requirement of a stationary process and thus is not stationary. To make it stationary different transformations can be used. One such transformation is to take the difference of all consecutive values in the time series.

2.3.8 White noise time series

A white noise time series is a series with a constant mean equal to zero and a constant variance and where the auto-correlation is close to zero. White noise time series cannot be used for meaningful prediction. It is therefore important to first verify that the data intended for evaluation is not a form of white noise[37].

2.4 Graph centrality

In graph theory there are many different centrality metrics, all of which aim to calculate the importance of each vertex in a graph in some way. In this section a sub-set of these centrality metrics are presented. These include degree centrality, in-degree centrality, out-degree centrality and closeness centrality.

2.4.1 Degree centrality

Degree centrality is a measure of the number of adjacent neighbors to a given vertex in a graph[58].

$$C_d(v) = deg(v)$$

where $deg(v)$ is the number of adjacent vertices to v and $C_d(v)$ is the degree centrality of vertex v. Oftentimes the centrality is normalized by dividing by the maximum possible degrees $n - 1$ in a graph of size n.

$$C_d(v) = \frac{deg(v)}{n-1}$$

If the graph is multi-directional the maximum possible degrees can be more than $n - 1$ due to loops and thus the centrality value can be more than 1.

2.4.2 In-degree centrality

In a directed graph in-degree centrality is calculated as the number of incoming edges e_{in} to a vertex v.

2.4.3 Out-degree centrality

In a directed graph out-degree centrality is calculated as the number of outgoing edges e_{out} from a vertex v.

2.4.4 Closeness centrality

The closeness measures how close, on average, a vertex is to all other vertices in a graph[59]. It is defined as the inverse of the average distance to all other vertices:

$$C_c(v) = \frac{n-1}{\sum_{i=0}^{n-1} dist(v, vertex(i)}$$

where $dist(v, vertex(i))$ is the distance between a vertex v and vertex number i in a graph. $n-1$ is the number of all other vertices in the graph.

2.5 Artificial Neural Network

An artificial neural network model aims to use features of the human brain to mathematically model complex problems[20]. The artificial neural network model is one kind of machine learning model often applied in data science. In this section the aim is to describe the different parts of a neural network to understand MLP.

The overall aim for a neural network model is to learn some behavior based on training data. A neural network has the possibility to model both linearly separable data as well as data which is not linearly separable where ordinary statistical models such as linear regression or AR might fail. As the name entails a neural network is a network of connected neurons[40]. The neurons in the network are organized in layers that serve different purposes. In its simplest form the network has an input layer and an output layer. The inputs correspond to the predictor variables and the output to the explanatory variable(s). In this case the network is equivalent to a linear regression. Figure 2.1 shows all components included in this simple neural network. The input layer consists of two input neurons and the output layer consists of one neuron.

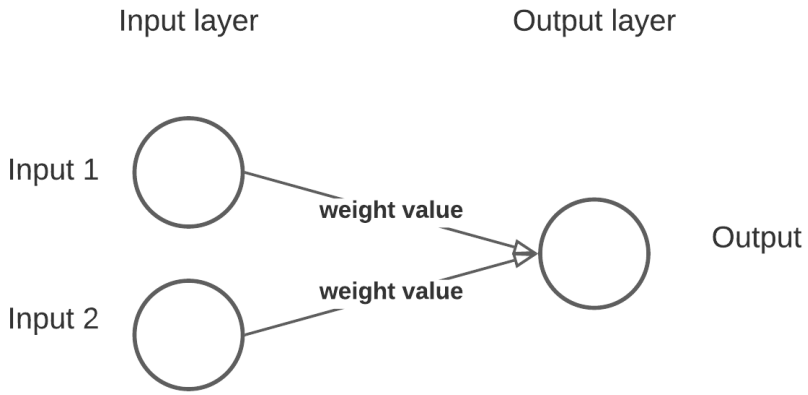


Figure 2.1: A simple neural network architecture comprised of the input layer containing two neurons and the output layer containing one neuron.

Each predictor variable has an attached coefficient called the "weight" as seen in the connected edges from the input layer to the output layer in 2.1. This weight value is estimated when training the model. In this example the predictory variables x_1, x_2 that corresponds to "Input 1" and "Input 2" will both have their own weight value, denoted w_1 and w_2. Combined, the equation becomes $y = x_1 * w_1 + x_2 * w_2$ where y is the output value.

The weight values are estimated using a learning algorithm such as back-propagation. In essence back-propagation will make adjustments to the weight values after every training sample. The details of back-propagation will be explained further down.

More advanced neural networks also include a number of so-called "hidden layers" that are used when modeling non-linearly separable problems. A network with a lot of hidden layers is often referred to as a "Deep neural network". A network that contains one or more hidden layers is also called a multi-layered neural network. Since the input is propagated through the layers such a network is usually referred to as a multi-layered feed-forward neural network. This network architecture is presented in Figure 2.2 containing one input layer, one hidden layer and one output layer.

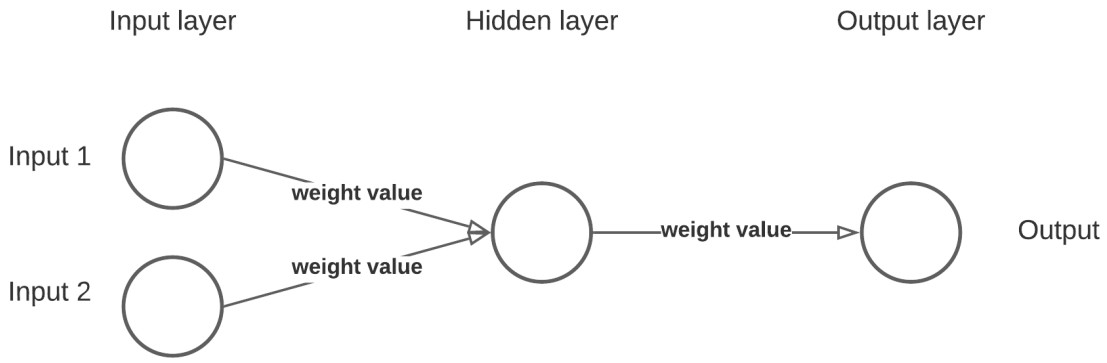


Figure 2.2: A neural network architecture comprised of the input layer containing two neurons, the hidden layer containing one neuron and the output layer containing one neuron.

2.5.1 Neuron

The neurons in an artificial network are present in all layers of of the network architecture. The value of the neurons in the input layer is determined by the numerical input value for that neuron. The value of the connected neuron in the next layer of the network is a linear combination of the input values/neurons and the weights connected to the next neuron in the next layer. In this example the value for a next neuron j can be represented by the following equation:

$$z_j = b_j + \sum_{i=1}^{2} w_{ij} * x_i$$

where w_{ij} is the weight value between the input neuron i to the hidden layer neuron

j and x_i is the value of the input neuron i. b_j is the bias value for hidden layer neuron j. This is then fed through an activation function[40].

2.5.2 Activation function

The activation function is the final step, after summing the linear combination of weights and inputs and adding the bias, to get an output value. There exists several different activation functions for different purposes and can be categorized into linear activation function and nonlinear activation functions. One example of a linear activation functions is the binary threshold activation function suggested by Mculloch-Pitts[20]. The binary threshold activation is defined as:

$$y = 1 \text{ if } z \geq \Theta$$

$$y = 0 \text{ if } z < \Theta$$

where $\Theta = -b$

An example of a nonlinear activation function is a sigmoid function. A sigmoid function has the characteristics of an "s-shaped" function and one such function is the logistic function:

$$f(z) = \frac{1}{1 + e^{-z}}$$

Cybenko, G. showed that a neural network with only one hidden layer and a nonlinear activation function such as a sigmoid function can approximate any continuous function f[47].
Rectifier Linear Unit (relu) is another commonly used activation function and is often the preferred choice for performance and accuracy for deep learning architectures[48].

$$f(z) = max(0, z)$$

2.5.3 Supervised Training

Supervised training is done by providing training samples where the output of those training samples are known. The network is "supervised" in the sense that after each run-through of the layers with a set of input values the output generated by the network is checked against the correct output. A cost function is used to calculate the cost using the difference between the generated output and the correct output. An example of a cost function is the Mean Squared Error (MSE). The cost is then used to update the weight values using a learning algorithm such as back-propagation.

2.5.4 Weights

Each connection between one neuron in a layer to a neuron in another layer has a weight value. This weight value is continuously modified after each training round in order to minimize the loss function.

2.5.5 Layers

In the most simple case a neural network only has an input layer and an output layer. For modelling more complex problems a neural network architecture is often comprised of at least one hidden layer as well. All layers contain a pre-determined amount of neurons. All neurons in the input layer is connected to every neuron in the next layer and the same applies to the next layer except for the output layer.

2.5.6 Loss Function

A loss function maps the output of the neural network to some value representing the cost of that value. The goal is often to minimize the loss function and thus minimize the cost of the outputs of the network. An example of a loss function is the mean squared error (MSE).

2.5.7 Gradient descent methods/Optimization strategies

Gradient methods are used to iteratively change the weight values of a network in order to minimize the loss function. This is done by calculating the gradient of the loss function in respect to the weight.

2.5.7.1 Gradient Descent

Gradient Descent finds a local minimum in an iterative manner. Observe the function below:

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \eta_j \mathbf{d}_j \tag{2.1}$$

where η_j is the step length and $\mathbf{d}_j$ is called the search direction at $\mathbf{x}_j$. The idea is to get a little closer to the local minimum after each iterative step. Now observe the Taylor expansion of the first order of $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_0$, where f is the loss function.

$$f(\vec{\mathbf{x}}) \approx f(\vec{\mathbf{x}_0}) + \nabla f(\vec{\mathbf{x}_0})^T (\vec{\mathbf{x}} - \vec{\mathbf{x}_0})$$

As (Wahde, 2008, p.19) explains, the most effective way to traverse to the local minimum from $\mathbf{x}_0$ such that $f(\mathbf{x}) < f(\mathbf{x}_0)$, is by setting the search direction to the negative gradient at $\mathbf{x}_0$[4].

$$\mathbf{d}_0 = -\nabla f(\mathbf{x}_0)$$

With this in mind the eqn (2.1) now takes the form:

$$\mathbf{x}_{j+1} = x_j - \eta_j \nabla f(\mathbf{x}_j)$$

This iterative step is repeated until the gradient vanishes(convergence).

2.5.7.2 Stochastic Gradient Descent

Stochastic gradient descent(SGD) is one of the most used optimization strategies in machine learning. SGD is in principle the same as gradient descent, the difference is that SGD selects a sample in a stochastic manner from the whole dataset

and performs gradient descent on that sample to decrease the running time of the algorithm[5].

2.5.7.3 ADAM

ADAM is one variant of gradient descent that is common for training forecasting methods. It uses gradient of the first order. It also performs moving average on the portentially noisy data and gets a smoother moving average function as illustrated below:

$$v_t = \beta * v_{t-1} + (1 - \beta) * g_t^2$$

where β is set to a value between $[0, 1)$ and g_t^2 is the element-wise square of all gradients at time t. $g_t = \nabla f_\theta(\theta)$ where θ are the parameters.
With this in mind the algorithm for ADAM is illustrated below:

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
    $m_0 \leftarrow 0$ (Initialize 1st moment vector)
    $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
    $t \leftarrow 0$ (Initialize timestep)
    while θ_t not converged do
        $t \leftarrow t + 1$
        $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
        $m_t \leftarrow \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$ (Update biased first moment estimate)
        $v_t \leftarrow \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$ (Update biased second raw moment estimate)
        $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias−corrected first moment estimate)
        $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias−corrected second moment estimate)
        $\theta_t \leftarrow \theta_{t-1} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
    end while
    return θ_t (Resulting parameters)

The algorithm above is taken from the paper *"Adam: A Method for Stochastic Optimization"*[3]. Where it is covered in more detail.

2.5.8 Back-propagation

A neural network with only an input and output layer has no internal representation (the hidden layers containing neurons) that has to be learned and can be solved using the perceptron convergence procedure detailed in Minsky, M. et al[50]. For

multi-layered networks back-propagation has been shown to be an efficient way of computing the gradients of the loss function in respect to the weights of the network. The Back-propagation is used in conjunction with some kind of gradient method where a typical method is the stochastic gradient descent method[49].
Back-propagation works by iterating backwards, starting from the output neuron, and calculating the updates to the each weight value by:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} [49]$$

where i is the neuron in the previous layer and j is the neuron in the current layer, η is the learning rate of the network which is often very small and $\frac{\partial E}{\partial w_{ij}}$ is the the derivative of the error term with respect to the current weight value. The error term here is equivalent to the cost function. All of the weights in the network is updated according to this rule.

2.5.9 Multi Layer Perceptron (MLP)

MLP is a type of feedforward neural network. This means that the neurons in each layer is only connected to the next layer in a list of consecutive layers. The last layer is the output layer [1].

2.5.10 Error measurement

An error measurement is often used to measure the error of a model's predictions in the case of time series analysis and forecasting. This is usually done by some kind of difference between the data predicted at a certain time step by the model and the observed correct value at that time step. As explained in the previous section about splitting the data in a train and test subset, the test set usually acts as the observed data that is compared to the model predictions. A commonly used error measurement is the Root Mean Squared Error (RMSE). This can be formally defined as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{t=0}^{N} e_t^2}$$

where the error e_t is defined as:

$$e_t = p_t - o_t$$

for a time t for the predicted value p_t and the observed value o_t.

2.6 Classification models

Classification models classify data points into some category based on the features of that data point. This section will explain two common classification models as well as cross validation as an evaluation strategy to mitigate the risk of overfitting the training data to a model. The models include a logistic regression classifier and a random forest classifier. Accuracy, precision, recall and reciever operating characteristic (ROC) as measurement scores will also be introduced and explained.

2.6.1 Logistic Regression Classifier

The Logistic Regression Classifier(LRC) analyzes data with arbitrary many parameters in order to classify data to either true or false based on a probability that has been evaluated. It is labeled as a supervised machine learning method.

2.6.1.1 Input & Output Data

The input data for the LRC comes in form of vectors. Where x_i is the i'th vector of a total of n vectors. Each vector also has a number of k parameters. So for instance, x_{ij} would represent the j'th parameter of the i'th x-vector in an input matrix X. The output data Y is a $n \times 1$ vector where each feature of that vector is evaluated to either $1 = TRUE$ or $0 = FALSE$ in accordance to the the respective vector in the matrix X.
The input data X and the output data Y can be illustrated as the following:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,k} \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Say we have access to some arbitrary input X' and output Y'.
The posterior probability $P(Y = Y'|X')$ defines how likely it is to receive that output based on the input. The probability p ranges from [0-1].

2.6.1.2 Decision function

However a probability is another word for uncertainty, since the classifier need to decide whether to classify each data point as true or false the uncertain posterior probability needs to be converted to a differentiable *decision function*. The decision function, also known as the *sigmoid function* has the general form $sig(t) = \frac{1}{1+e^{-t}}$ and is shown in Figure 2.3. It has an upper bound of 1 and lower bound of 0.

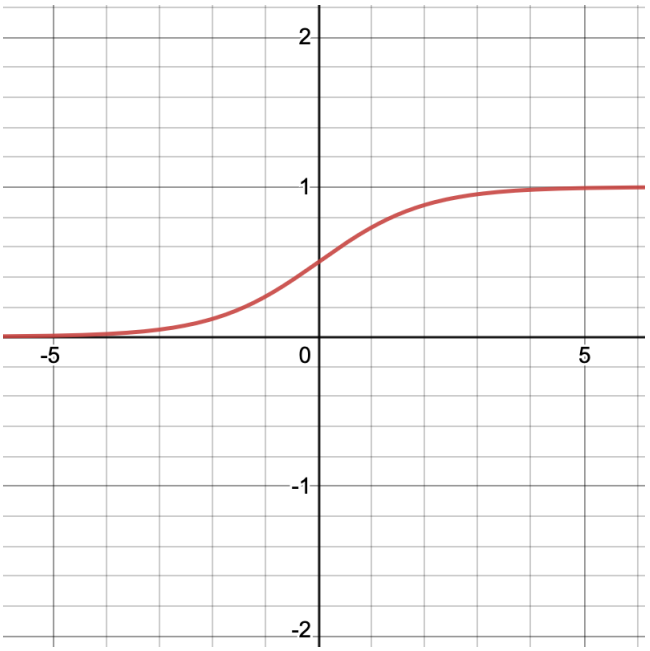


Figure 2.3: A plot of the generic sigmoid function $sig(t) = \frac{1}{1+e^{-t}}$

In our case we can re-write the posterior as

$$P(Y|X) = \frac{1}{1 + e^{-f(x)}}$$

, where $f(x)$ consists of the features with their corresponding weights in linear form:

$$f(x) = x_0 + x_1\beta_1 + x_2\beta_2 + x_k\beta_k + \epsilon$$

where $x, \beta, f(x) \in R^k$, ϵ is the random measuring error estimation. The last two equations can be combined and re-written to form the **log of odds ratio**:

$$log\left[\frac{P(Y|X)}{1 - P(Y|X)}\right] = x_0 + x_1\beta_1 + x_2\beta_2 + x_k\beta_k + \epsilon = f(x)$$

It is necessary to adjust the weights ($\beta's$) in order to reach the maximum-likelihood estimation(MLE). Meaning that the estimation of p should be as close to 1 as possible for samples labeled as 1 and as close to 0 as possible when the samples are labeled as 0[10][11].

2.6.1.3 Classification step

Given that we have reached MLE let's observe the $log_{10}(x)$ function. This is shown in figure 2.4.

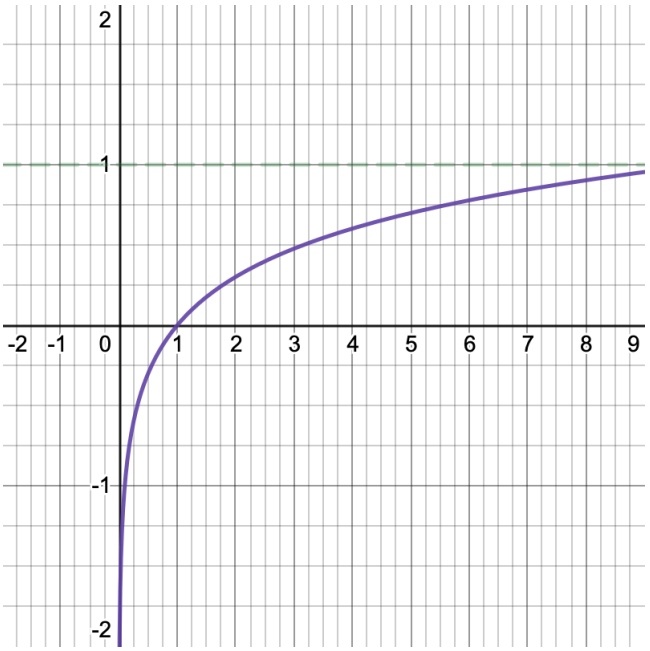


Figure 2.4: A plot of the log_{10} function, $y = log_{10}(x)$

Upon passing x = 1 where y = 0 we can classify according to this ratio

$$log\left[\frac{P(TRUE)}{P(FALSE)}\right] = log\left[\frac{P(Y|X)}{1 - P(Y|X)}\right]$$

The ratio will evaluate as a positive value if $P(TRUE) > P(FALSE)$ and negative if $P(FALSE) > P(TRUE)$ leading us to classify as $TRUE$ if positive and $FALSE$ if negative[9].

2.6.2 Decision Tree

A decision tree is created from mappings of input-output data. The input of a decision tree is a 2-dimensional vector $(X_1, X_2, ...X_N)$ where each X_i is a feature vector representing the values for that feature. The output is a 1-dimensional vector $Y = (y_1, y_2, ..., y_N)$ where each y_i is a class label. A decision tree is a directed tree and consists of a Root Node corresponding to the start of the decision tree, several internal nodes at each point where the tree splits left t_l or right t_r creating two outgoing edges. The nodes that have no outgoing edges are referred to as leaf nodes and contains the best guess for that path[29]. A tree is generated from top to down with some splitting criterion at each new node. This splitting criterion checks all feature values and determines the best feture value to split based on the lowest impurity measure calculated for each split variation[27]. There are several impurity metrics used for determining the splits. Gini impurity is one of them and will be explained below.

The feature value to split on is based on the maximum expected decrease of some impurity measure when considering all kinds of splits possible at some point in the

tree[27]. The splitting rule of a decision tree is defined as:

$$\Delta I = I_{metric}(X) - I_{metric}(t_l) * p(t_l) - I_{metric}(t_r) * p(t_r)$$

where ΔI is the decrease in impurity, $I_{metric}(X)$ is the impurity score with set X, $p(t_l)$ is the proportion of the set split to the left and $p(t_r)$ is the proportion of the set split the right.

2.6.2.1 Gini Impurity/Mean decrease impurity (MDI)

The Gini Impurity is defined as:

$$I_{gini}(X) = \sum_{i=0}^{k} p_i(1 - p_i)$$

where X is a set of input data and p_i is the proportion of input data belonging to a certain class k.

2.6.3 Random Forest Classifier

A Random Forest Classifier uses an ensemble of regular decision trees in order to improve the general performance of the classifier. The number of trees used to build the forest will differ depending on the purpose but generally one uses around 100 trees or more. The trees are built using a bootstrapped data set from the original data set X and the number of features used in X is selected at random. Thus, a bootstrapped data set originally containing $X = (X_1, X_2, X_3, X_4)$ might result in a subset of features $S = (X_2, X_4)$ where two of the original four features were picked at random. A decision tree is built using the data for these two inner vectors. This is done until the predetermined number of decision trees have been created. When predicting a class label on new data, the data is propagated through each sub-tree and the majority class from all of the trees is selected as the guess of the forest[24].

2.6.4 Cross validation

Cross validation aims to estimate the general performance of a statistical- or machine learning- model. Instead of only evaluating a model on one train-test split the model is evaluated several times in order to gauge general performance. There are several different cross validation methods and in this section K-fold cross validation and Grouped Shuffled cross validation will be explained.

2.6.4.1 K-fold cross validation

K-fold cross validation is one of the most used cross validation methods. This method shuffles the data and split it into k equally sized sub-samples containing train and test data. During each evaluation it is split differently, meaning that different parts of the data will end up in the train and test set during each fold[44]. To better illustrate this, take an example where the data set contains 9 values $d = [1, 2, 3, 4, 5, 6, 7, 8, 9]$ and k is chosen to be 3. The data set is shuffled around

and a third of the data set is used as test data during each fold. Shuffled data $d = [5, 2, 6, 1, 3, 4, 8, 9, 7]$ where the three folds are:

Original data set. $d = [1, 2, 3, 4, 5, 6, 7, 8, 9]$
Shuffled data set. $d = [5, 2, 6, 1, 3, 4, 8, 9, 7]$

Fold 1. $Train = [1, 3, 4, 8, 9, 7]$ $Test = [5, 2, 6]$
Fold 2. $Train = [5, 2, 6, 8, 9, 7]$ $Test = [1, 3, 4]$
Fold 3. $Train = [5, 2, 6, 1, 3, 4]$ $Test = [8, 9, 7]$

By training and evaluating on different parts of the data set one can get an estimate of the general performance by averaging the scores acquired from each fold.

2.6.4.2 Grouped Shuffled Cross Validation

Oftentimes it is interesting to evaluate the data with specified fractions of train and test data. This can be done by creating several randomly shuffled data sets from the original data set and then picking a fraction to be train and another fraction to be test. Unlike K-fold cross validation this doesn't guarantee that all data will be used both for training and testing. The data can also be grouped on domain-specific features of the data such as the year the data was gathered for example. In this way the same year won't be included both in the train and test set simultaneously but one year is restricted to either the training set or test set[43].

2.6.5 Accuracy Measurement

Typical measurements for classifier evaluation include accuracy, precision, recall scores. Receiver Operations Characteristics (ROC) are also very common. These terms will be explained in this section.

2.6.5.1 Accuracy

This is a measurement of the proportion of inputs that got correctly classified.

$$A = \frac{CC}{N}$$

where A is the accuracy, CC is the amount of correctly classified inputs (positive and negative) and N is the number of inputs.

2.6.5.2 Precision

Precision, also known as positive predictive value, is a measurement of the proportion of positively classified entries (1) that are true positives[56].

$$P = \frac{TP}{N+}$$

where P is the precision, TP is the true positive amount and N^+ the amount of inputs classified as positive.

2.6.5.3 Recall

Recall, also known as true positive rate, is the number of correctly classified positives out of all positive classes (1)[2].

$$R = \frac{TP}{TP + FN}$$

where R is the recall, TP is the number of true positives and FN the amount of false negatives.

2.6.5.4 ROC-curve and Area under curve (AUC)

ROC-curve can be used to evaluate and visualize the performance of classifiers. In the binary case the curve is defined by the true positive rate and the false negative rate of the positve and negative classes (1 and 0). The vertical axis of the graph represents the true positive rate (TPR) i.e recall and the horizontal axis represents the false positive rate $FPR = \frac{FP}{FP+TN}$ where FP is the amount of false positives and TN the amount of true negatives[57]. Some classifiers returns the probability scores for its instances and uses a threshold to decide which class an instance should be classified as. The ROC-curve can be used to measure the performance classifiers at different thresholds. This can be very valuable in order to create a more "conservative" classifier i.e one that aim to reduce the number of false positives, or a more "liberal" classifier that aims to increase the number of true positives while not caring about also classifying false positives. Furthermore, the area under the curve (AUC) can be used to measure the overall performance of a classifier and is useful when comparing different types of classifiers[7]. An example ROC-curve is presented in Figure 2.5.

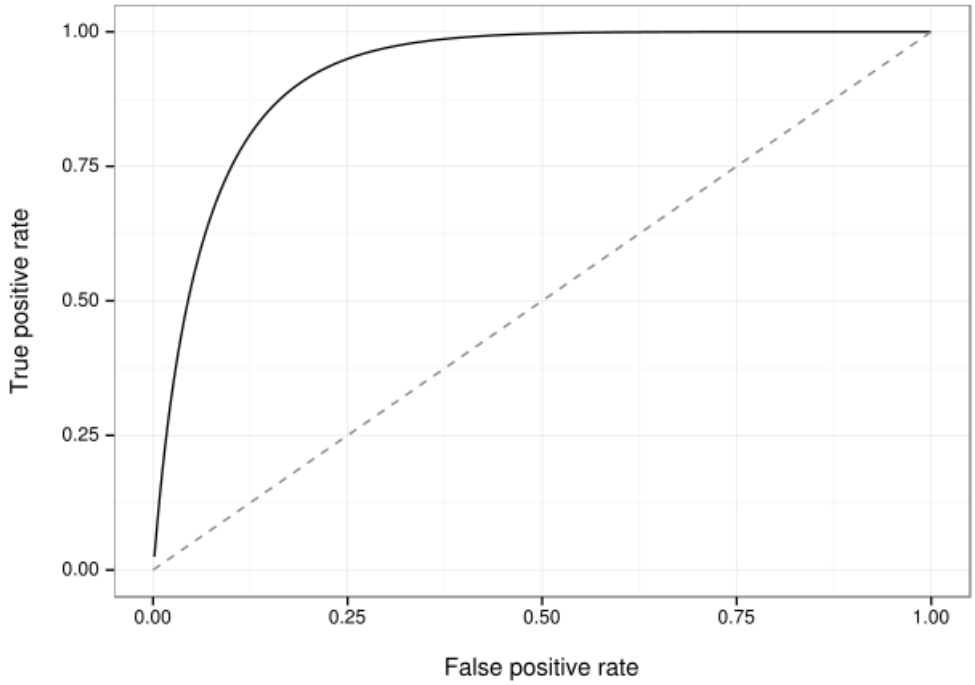


Figure 2.5: An example of a ROC-curve. The dotted line represents the case where a model cannot discern two classes and essentially guesses the class. The black line represents a good model where the true positive rate is high and the false positive rate is low. Figure source: [21].

This is commonly used in machine learning to measure the performance of a model. A line that passes through the top-left corner value of 1 tells us that there is a threshold where the model perfectly distinguishes the classes. A classifier that follows the dashed line is as good as a classifier that chooses the class at random. Oftentimes the area under the curve (AUC) is used to measure the performance of a model if no plot is generated. A high AUC is good, an AUC of 0.5 is a model with no decision-capability (bad) and 0 is consistently wrong decision.

2.6.5.5 Precision-Recall Curve

The precision-recall curve is another important tool to measure the effectiveness of a classifier. It is valuable in situations when the data set has an imbalanced proportion of classes[7]. The precision is often worse on imbalanced data set compared to balanced data sets. Since the ROC-curve doesn't display precision, any reduction in performance usually goes unnoticed and that's where the Precision-Recall-curve is a good complement.

In order to determine if the classifier performs better than a random classifier a baseline threshold is established[2]. This baseline is calculated differently compared to the ROC-curve baseline. It is the proportion of positive classes among the data:

$$B = \frac{P}{P + N}$$

where P is the number of positive classes in the data set and N is the number of negative classes in the data set. Two examples of precision-recall-curves, including their baselines, are shown below for a balanced and imbalanced data set. Figure 2.6 shows the precision-recall curve for a balanced data set to the left and the curve for an imbalanced data set to the right.

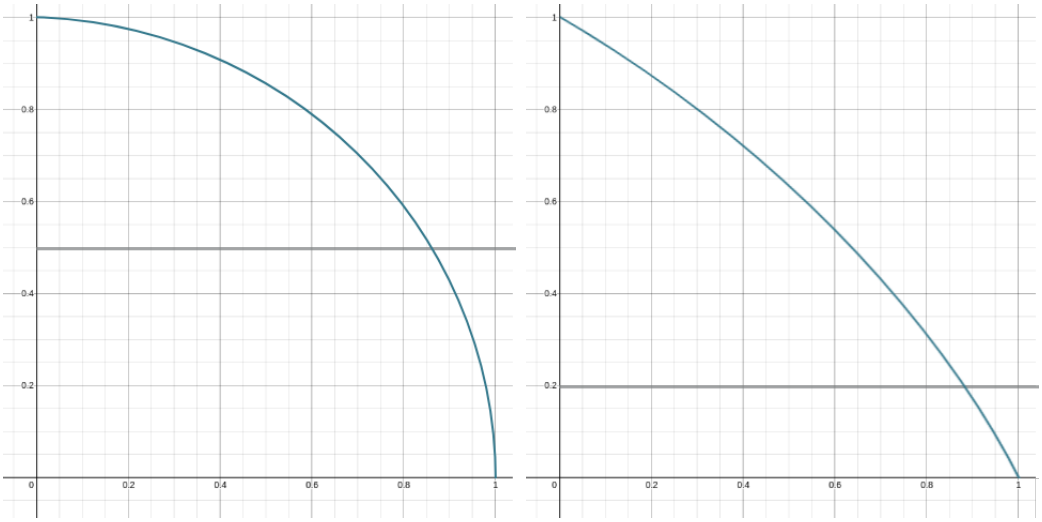


Figure 2.6: The left precision-recall-curve is for a balanced data set and the right shows an imbalanced data set. y-axis represents the precision and the x-axis represents the recall.

For the balanced data set, observe the threshold at $y = 0.5$ for when the classifier is considered random. In almost all cases the curve is above this line which is indicative of a good classifier. The imbalanced data set contains 5 times more negative classes compared to positives as is reflected by the threshold at $y = 0.2$. The imbalanced data set shows a slightly steeper curve but since the threshold is lower it is still considered a good classifier.

3
Data

In this chapter the different types of data used in this project will be explained in detail. These include OpenStreetMap data, HERE traffic data and weather data.

3.1 OpenStreetMap

OpenStreetMap started as a crowd-sourced project where users could map out buildings, landmarks and other points of interest (POI) on a publicly available world map[46]. Today the users are in the millions and OpenStreetMap provide valuable information about buildings, roads, landmarks and other POI around the world for everyone to access for free. Because of the easy accessibility and the amount of data available, OpenStreetMap is chosen for gathering extra features for forecasting and classification.

3.2 HERE Traffic Data

The data used in this project was gathered using the location data provider HERE[42]. This data was gathered 50 cities during the entire year of 2018 and analyzed by Verendel et al.[19]. In this paper, a subset of 15 European cities were selected for further analysis and forecasting. These are presented in the table further below. A subset of 6 cities from the 15 cities were selected for classification due to time-constraints and hardware-limitations. In this section the different types of data will be explained. These include real-time measured traffic speed data, average traffic speed data, coordinates representing a road and road length. The data from HERE covers a subset of all roads that exist in each city.

3.2.1 Speed data

Speed measurements were collected approximately every 5 minutes for each road in every city. Due to potential time differences or network/power outages the data collected during a 15-minute time interval was averaged to one value. This results in 96 time intervals per day. Because of a change of how the data was collected by HERE midway through 2018, only the data from the first half of 2018 was selected, resulting in 17280 time intervals of data. The traffic data is shifted to match the local time zone of each city.

3.2.1.1 Real-time traffic speed data

Real-time traffic speed data is the measured speed at every time interval in km/h for each road.

3.2.1.2 Average free-flow speed data

Average free-flow speed data is the calculated average by HERE in km/h.

3.2.2 Road coordinate representation

Each road is represented as edges by a list of consecutive WGS84/GPS coordinates in latitude, longitude format.

3.2.3 Road length

Each road has its own road length measured in km.

3.2.4 Cities

The selected subset of 15 cities is visualized in Figure 3.1. The figure shows Europe and the colored dots correspond to the selected cities. The legend shows the city names.

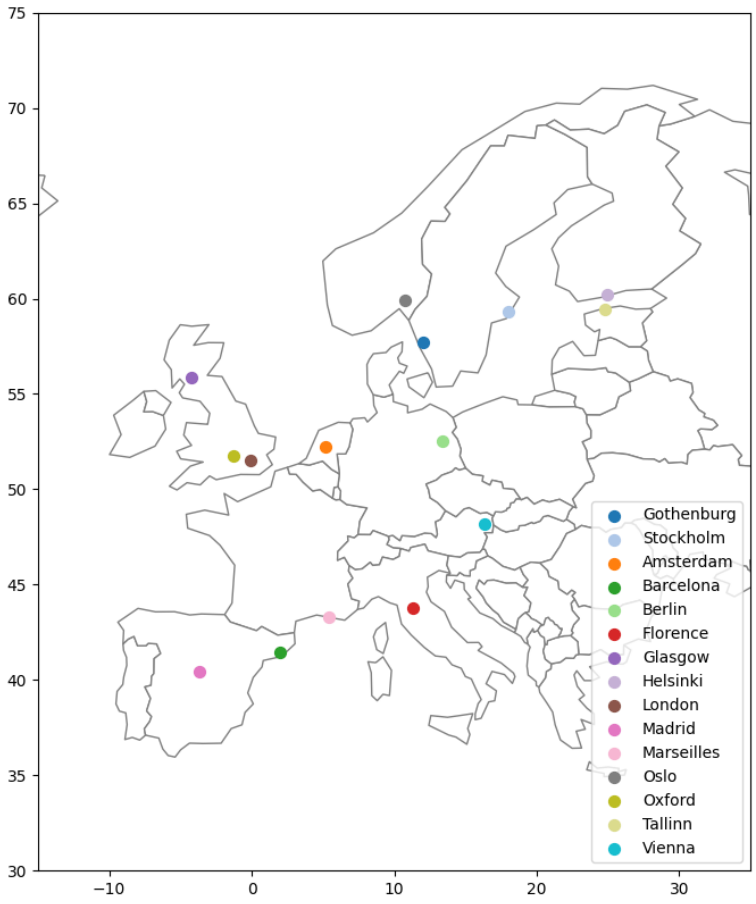


Figure 3.1: Map of Europe showing the selected cities as colored dots.

3.2.5 Road Coverage per city

The road coverage is a proportion of roads covered by the HERE data. An OpenStreetMap graph is generated by taking the northen-most latitude, sorthern-most latitude, eastern-most longitude and western-most longitude coordinates from the roads of a city. The proportion of roads covered by the HERE data in the generated graph is then considered the coverage. The coverage is presented in Table 3.1. Most cities achieve a coverage of at least 10% and some cities have a coverage of over 30% which is good. In the table, each road covered refers to an edge in the generated graph. To exemplify, the first row shows the city Gothenburg. In Gothenburg the HERE data covers 16115 roads out of the 62917 generated by the graph. This works out to a coverage of 25.61%.

City	**Roads covered**	**Roads total**	**Coverage**
Gothenburg	16115	62917	25.61%
Stockholm	22149	113079	19.59%
Amsterdam	70024	728645	9.61%
Barcelona	75075	228369	32.87%
Berlin	36206	161867	22.37%
Florence	16181	73348	22.06%
Glasgow	9601	112620	8.53%
Helsinki	15981	34329	46.55%
London	102869	596486	17.25%
Madrid	36553	114370	31.96%
Marseilles	5307	35570	14.92%
Oslo	10015	29306	34.17%
Oxford	1680	13933	12.06%
Tallinn	7080	24845	28.50%
Vienna	28153	75279	37.40%

Table 3.1: Table showing road coverage per city. Each row includes the exact amount of edges covered, total number of edges and the coverage.

3.2.6 Road Coverage in a city

To exemplify how coverage may look like in a city we present Figure 3.2 showing the road coverage in Gothenburg, Sweden. The colored edges represent roads covered and gray edges roads not covered by the data.

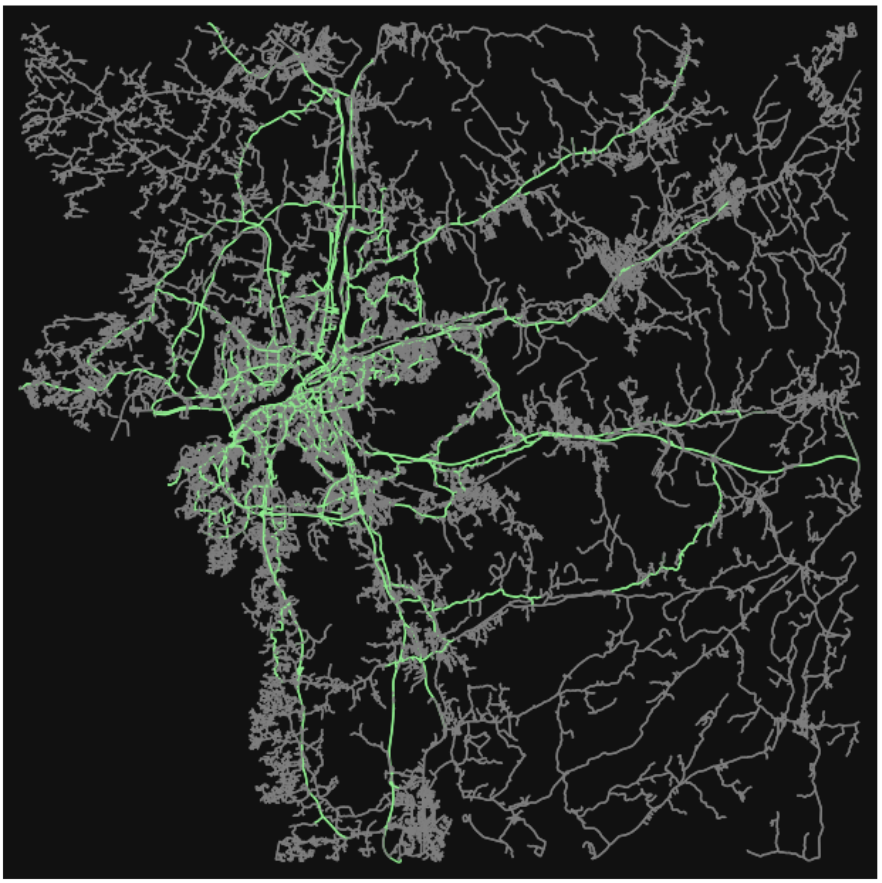


Figure 3.2: Map showing road coverage in Gothenburg, Sweden.

The figure shows that the coverage is greater towards the center of the city where the green roads are more closely packed. Moving outside of the center the covered roads are more sparse. Generally there are more roads in a city center compared to the outside of a city which makes the coverage representative of that.

3.3 Weather data

Copernicus Climate Data Store's ERA5-LAND hourly data set is used to get accurate measurements for a given area in Europe for precipitation and temperature[36].

4
Methods

In this chapter we present our methods. These include analysis of the data and the evaluation of both the forecasting and the classification methods.

4.1 Analysis

Analysis of the data is a crucial first step for determining the next steps in our work. The analysis will be performed on the traffic speed data gathered through the HERE API. The traffic speed time series data will be checked for stationarity to guarantee good performance of the AR model in the forecasting part.

4.1.1 Traffic speed data

The traffic speed data is assumed to not be white noise based on a non-zero auto-correlation and a non-zero mean. This is observed when plotting the mean speed values of each road and city in Figure 4.1 and the significant auto-correlations of each road and city in Figure 4.2.

Looking at 4.1 the mean is always above zero which is a good indicator that the traffic speed data is not white noise. The median represented by the green line shows a speed of around 30 km/h. The lower limit, represented by the lowest horizontal black line, shows the lowest mean speed value of around 5 km/h and the upper limit(upper horizontal line) shows the highest mean speed of around 80 km/h. Outliers are represented by the black circles creating a vertical line on top of the box-plot.

Looking at 4.2 the auto-correlation is never zero which is a good indicator that the traffic speed data is not white noise. The upper limit is 40 since that was the limit chosen when calculating the significant number of auto-correlations. The lower limit is slightly below 10 and the outliers are never zero.

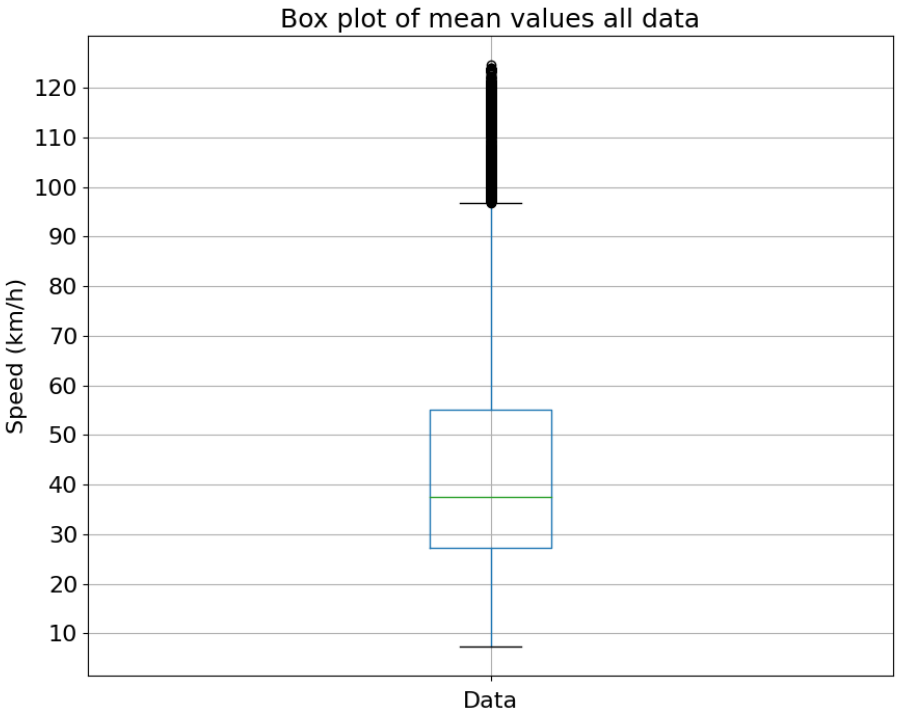


Figure 4.1: Box plot showing the mean value observed on each road.

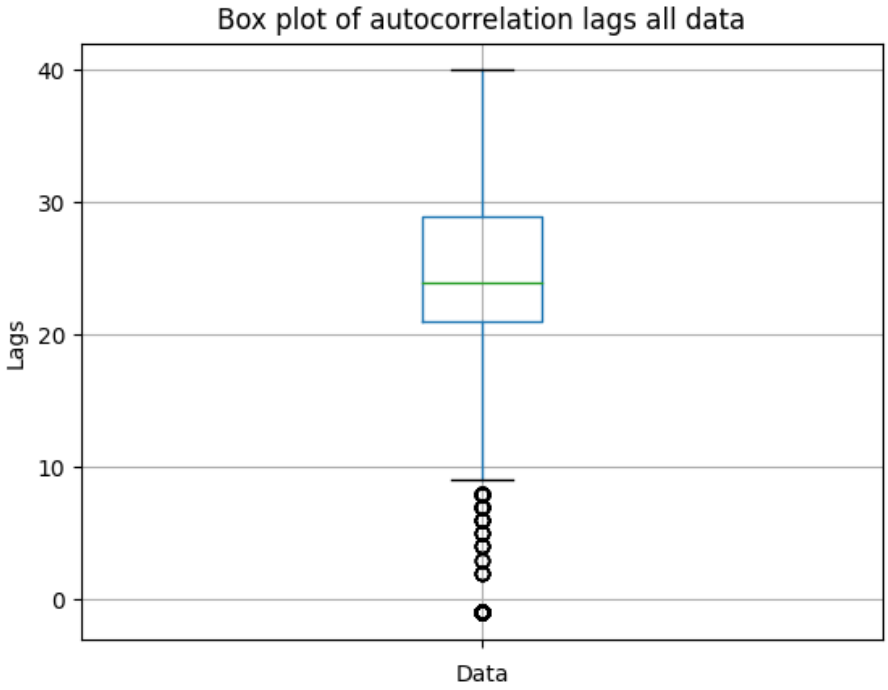


Figure 4.2: Box plot showing the auto-correlation lags for each road.

4.1.2 Stationary or non-stationary and transformations

To use the data for fitting an AR model stationarity needs to be checked. This will be done using the Augmented Dickey-Fuller test. If the data is non-stationary a transformation by difference will be applied before fitting any statistical models. The Augmented Dickey-Fuller test is performed with a confidence interval set to 95%.

4.2 Weather

The ERA5-Land data-set was chosen for gathering weather data [36]. This data set was chosen because of its large data availability and the resolution of the data on the hour. Since the HERE API records traffic data at a 15-minute time-interval, the weather data is post-processed by copying the hour value 4 times in order to match the HERE data. Since the weather data comes in Universal Time Standard (UTC) format a city's data is shifted to its local time zone.

An Australian study (Keay et al, 2005) looked at the impact on different weather variables and traffic flow and found that the strongest correlation was found with rainfall and that it had the greatest impact in spring and winter[35]. The attributes of interest are therefore temperature and precipitation. The reason being that a combination of these attributes can affect the flow of traffic. Snowfall, rain or lack there of may affect the traffic in different circumstances. The weather data is acquired from an area defined by the northern-most latitude, southern-most latitude, eastern-most longitude and western-most longitude of the coordinates from all roads of a city.

4.3 Forecasting

This section will present the different models used for forecasting traffic speed and our evaluation strategy. The models include AR, MLP and a Naive Baseline model. The overall aim of forecasting is to answer the following questions:

- Are there any benefits to using more advanced models when forecasting traffic speeds?
- How will weather affect our predictions?

4.3.1 Preprocessing

Before evaluation the data will be preprocessed. The preprocessing include filtering roads based on missing values.

4.3.1.1 Missing Values

The original HERE traffic speed data includes missing values at times where real-time measurements weren't available. A subset of roads are chosen based on the

percentage of missing values. The threshold is set to 20% so that all roads that have more than 20% missing values are excluded from the models. This is to get sufficient quality training samples from each road.

4.3.2 Baseline model

To properly evaluate the models a Baseline model will be established. If a model performs worse than the baseline model it either needs tuning or is not worth using for predicting traffic speeds. The baseline model in the forecasting case will use the previous value in the time series to predict the next value. It is essentially a lagged version of the original time series[6].

4.3.3 AR

The order of the AR model will be determined by the mean of the number of significant partial autocorrelation lags of the preprocessed roads[8]. Thus, the order is set to 7.

4.3.4 MLP

The MLP model consists of one input layer, one hidden layer and one output layer. There are 25 neurons in both the input layer and the hidden layer. This number stems from the mean of the significant autocorrelation lags. The output layer has 1 neuron, since we are only looking at the one-step forecasting ability.Relu will be the choice of activation function in the hidden layer. Stochastic gradient descent method used is the Adaptive Moment Estimation (Adam) optimizer with a learning rate set to 0.01.

4.3.5 Evaluation

This section will detail how the data is split before training a model, how missing values are imputed, what error measurement is used and how the model results will be compared.

4.3.5.1 Train-test split

The traffic speed data for each road will be split in a train and test set with 80% train data and 20% test data.

4.3.5.2 Imputation

Mean imputation will be performed on the traffic speed data set for each road. The train data set is imputed separately from the test data set. This is done to avoid data leakage that might occur if data from the test set is used to infer data in the train set.

4.3.5.3 Error measurement

A RMSE score is calculated based on the difference in predicted values compared to the real values.

4.3.5.4 Evaluation strategy

The aim is to evaluate the one-step forecasting ability which is equivalent to a 15-minute prediction into the future. This will be done for a uni-variate case that only considers the traffic speed and also a multi-variate case that considers weather data as well. The multi-variate case will only be evaluated using the MLP model since the other models are uni-variate-based.

The models will be evaluated on a subset of the first 10 preprocessed roads from each city. A model is built and evaluated for each road in each city. The RMSE scores are then averaged over all roads and all the cities used. Since the purpose is to observe any improvements in forecasting when switching to more advanced models this should suffice.

4.3.5.5 Input vector Univariate

Table 4.1 shows the univariate input vector that only consists of the speed in km/h.

Speed (km/h)

Table 4.1: Table showing Univariate Input Vector, speed is the only feature.

4.3.5.6 Input vector Multivariate

Table 4.2 shows the multivariate input vector that consists of the speed, precipitation and temperature.

Speed (km/h)	**Precipitation(m)**	**Temperature (K)**

Table 4.2: Table showing the Multivariate Input Vector. It includes speed measured in km/h, precipitation measured in meters and temperature measured in Kelvin.

4.4 Feature Extraction

In order to improve the performance of our models certain relevant features need to be compiled. These features include a set of geographical points of interest (POI) in proximity to all roads and different measures of centrality. These will be used for the classification models.

4.4.1 POI

The POI will be gathered from OpenStreetMap for each road. These will be split into two groups, one in close proximity along a road and another larger area surrounding the road. These groups will be referred to as corridor features and area features respectively.

A study conducted in Beijing, China looked at the impact of differences in infrastructure on traffic congestion[34]. In this study they performed a detailed survey to gather information about the most popular POI in terms of travel purposes of individuals. They found that "“work,” “school,” “shopping,” “leisure,” and “return home” were the primary traveling purposes of Beijing residents, accounting for about 85% of the total travel". We have chosen our POI based on this knowledge and our own intuition of what places might attract more people and thus influence traffic flow. For example one can assume that a trash bin wont affect the traffic situation as much as a bus stop or a library. The following categories were chosen as POI:

Main Category	Sub Categories
Amenity	School, Hospital and Bus station
Railway	Station and Platform
Highway	Bus stop

These POI will be gathered both as corridor features and area features yielding 12 features in total.

4.4.1.1 Corridor features

Corridor features will be gathered within a 50 meter radius of a road. These are shown in Table 4.3.

School(50m)	Hospital(50m)	Bus Station(50m)	Station(50m)	Platform(50m)	Bus stop(50m)

Table 4.3: Table showing the 6 corridor features

4.4.1.2 Area features

Area features will be gathered within a 500 meter area extending the road area. These are shown in Table 4.4.

School(500m)	Hospital(500m)	Bus Station(500m)	Station(500m)	Platform(500m)	Bus stop(500m)

Table 4.4: Table showing the 6 area features

4.4.2 Centrality

Three different features regarding centrality are taken into account. Closeness, in-degree and out-degree. The centrality value is calculated differently for each. Centrality is calculated on each node of a graph generated as in Section 3.2.6. The roads

are then matched to the nodes of this graph as explained in Section 3.2.6. Each road will then be represented by a list of nodes. In order to associate one centrality value to each road, the sum of all centrality values of the nodes along a road are averaged. The centrality features are shown in Table 4.5.

In-degree centrality	Out-degree centrality	Closeness centrality

Table 4.5: Table showing the three centrality metrics used

4.5 Classification

The classification will be performed using binary classifiers, that given an input at a 15-minute time interval, classifies it as either 0 (no delay) or 1 (delay). In order to classify delays the original traffic speed data will be transformed to a measurement of delay at each time. The models include a **Naive Majority-Based Classifier** which will act as the baseline, a **Logistic Regression Classifier** and a **Random Forest Classifier**.

To perform classification we present the necessary transformations applied to the traffic speed data in order to get the delay classes. Furthermore, we describe the final feature vector containing our 20 features used for training and evaluation of the classifiers. The evaluation strategy and evaluation metrics of choice are presented. Finally, we present the three scenarios used for evaluating the generalization performance of the models. The scenarios are evaluated on a subset of the cities of size 6. These include Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid. The overall aim of the classification-step is to answer the following questions:

- Will training on more cities improve the performance of our models on new data? If so, how much?
- What features have the best predictive capabilities?
- How well does generalization of the models between cities really work? Does it only work when the cities share a lot of features such as structure, culture and technology?

4.5.1 Transforming the traffic speed into delays

In order to classify delays, the traffic speed data for each road at each 15-minute time-interval have to be transformed into a measurement of delay. The delays will then be compared to calculated delay thresholds in minutes in order to generate the classes (1 or 0) for the classification models. This will be the output vector used during training and testing of the models. This transformation will be performed on each city's traffic speed data set. The following formula will be used to transform a traffic speed (km/h) at time t into a delay (minutes):

$$delay(t) = 60 * rl \left(\frac{1}{ts(t)} - \frac{1}{ats(t)} \right)$$

where $ts(t)$ is the traffic on a road at time t, ats is the average traffic speed on a road at time t and rl is the road length of the given road.

In order to determine the threshold we convert the average traffic speed measured on a road into the average time in minutes it takes to travel the road. The following formula is used:

$$average\_travel\_time(t) = 60 * rl \left(\frac{1}{ats(t)} \right)$$

The delay threshold value is calculated by increasing the $average\_travel\_time(t)$ by 50% and then subtract it by the $average\_travel\_time(t)$:

$$threshold(t) = (average\_travel\_time(t) * 1.5) - average\_travel\_time(t)$$

The delay at a 15-minute time interval is then compared to this threshold in order to determine the class for that time interval t:

$$delay(t) > threshold(t) \rightarrow 1$$
$$delay(t) < threshold(t) \rightarrow 0$$

4.5.2 Naive Majority-Based Classifier

As the name suggest the naive majority-based classifier will take the majority class observed in the training set and use that for predictions in the test set during each train-test fold. This model will act as a baseline for comparison against both the Logistic Regression- and Random Forest-Classifier.

4.5.3 Logistic Regression Classifier

In this section we explain how the ROC curve is generated for the Logistic Regression Classifier. The class probabilities used to generate the ROC-curve is calculated using the decision function of the Logistic Regression Classifier.

4.5.4 Random Forest Classifier

The Random Forest Classifier is constructed by 100 smaller decision trees. In this section we explain how the ROC-curve is generated for this classifier. The class probabilities used to generate the ROC-curve is calculated by taking the mean of the predicted class probabilities of the individual trees in the forest. In a single decision tree the class probability is calculated as the fraction of samples of that class in the leaves.

4.5.4.1 Feature Importance

The feature importance is calculated using the Gini Impurity. It is defined as the normalized total reduction of the Gini Impurity caused by the introduction of the feature[25]. This will be a percentage for each feature showing an estimate of that

feature's influence for the model predictions. The score is between 0 and 1 for each feature importance and all feature importances will sum to 1.

4.5.5 Feature vector

Classification models are trained using input data containing several features as mentioned before in Section 4.4. The input data is constructed as a 2-dimensional feature-vector where each column corresponds to a feature. This feature-vector will contain *weekday*, *time of day*, *corridor features*, *area features* and *centrality features* in that order.
As for the number of rows in the feature vector it is defined as:

$$\begin{aligned} rows = number\_of\_cities* \\ (time\_intervals\_half\_year* \\ number\_of\_sampled\_roads\_per\_city) \end{aligned} \tag{4.1}$$

where $time\_intervals\_half\_year$ is the number of 15-minute time intervals in the span of 6 months and will result in 17280 intervals per road. We sample 100 roads from each city which will be $number\_of\_sampled\_roads\_per\_city$.

Weekday: This includes the days of a week encoded as a range [0,6] where *0* is *Monday* and *1* is *Tuesday* etc. To account for each 15-minute time interval each day is copied 96 times creating a weeks worth of values. This is then repeated for $\frac{180}{7}$ weeks since we are dealing with half a year of values.

Time of day: This includes the time of day as represented by the current 15-minute time interval. The values range from *0* to *95* where *0* is at time *00:00* and *1* is at *00:15* etc. This is also repeated for 180 days.

Temperature The third column is the **temperature** measured in *Kelvin* acquired from the ERA-5-LAND data set.

Precipitation the fourth is the amount of **precipitation** measured in *meters* acquired from the ERA-5-LAND data set.

Corridor: This column includes all 6 corridor features as shown in Table 4.3. They are copied 17280 times in order to match the row length of the other columns.

Area: This column includes all 6 area features as shown in Table 4.4. They are copied 17280 times in order to match the row length of the other columns.

Centrality: This includes the in-degree centrality, out-degree centrality and closeness centrality. All of these three are copied 17280 times in order to match the row length of the other columns.

All features can be compiled into one feature vector where an example of such a feature vector is shown in Table 4.6. **WD** refers to **Weekday**, **TS** refers to **Time of day**, **Temp** refers to **Temperature**, **Corridor** refers to Table 4.3, **Area** refers to Table 4.4 and **Centrality** refers to Table 4.5.

WD	TS	Temp	Precipitation	Corridor	Area	Centrality
0	0	293.15	0.0001	...	...	...
0	1	293.15	0.0001	...	...	...
0	2	293.15	0.0001	...	...	...
0	3	293.15	0.0001	...	...	...
0	4	295.15	0.0000	...	...	...
0	5	295.15	0.0000	...	...	...
0	6	295.15	0.0000	...	...	...
0	7	295.15	0.0000	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.6: Example of a Feature Vector. Each column in the table represents one feature.

4.5.6 Selecting subset of cities

The subset of cities were chosen based on comparable centrality distributions and road length distributions. The cities are European-based and include three non-capitals and three capitals.

Non-capitals	Capitals
Gothenburg	Stockholm
Florence	Berlin
Barcelona	Madrid

The centrality distribution of each city is presented in Figure 4.3 and the road length distribution of each city is presented in Figure 4.4.

Looking at 4.3, all cities follow a normal distribution and are quite comparable which is good. Some exhibit more skewness than others such as Berlin.

Looking at 4.4, all cities have a different number of roads but the overall shape seems to be quite similar between the cities. The majority of roads are 100 meters or less in length which is to be expected in cities.

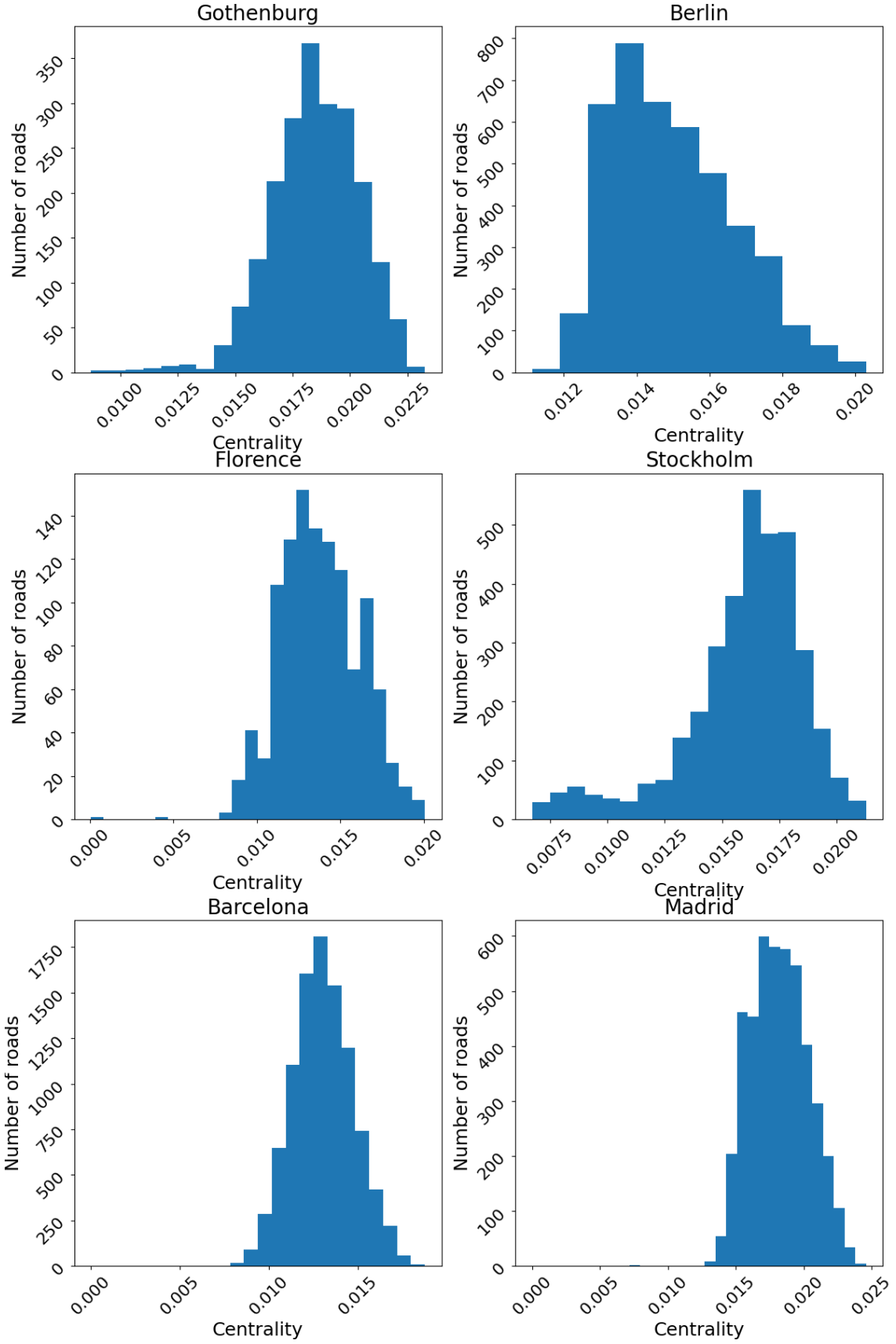


Figure 4.3: Plots showing the centrality distribution in each city used in the evaluation of the classifiers.

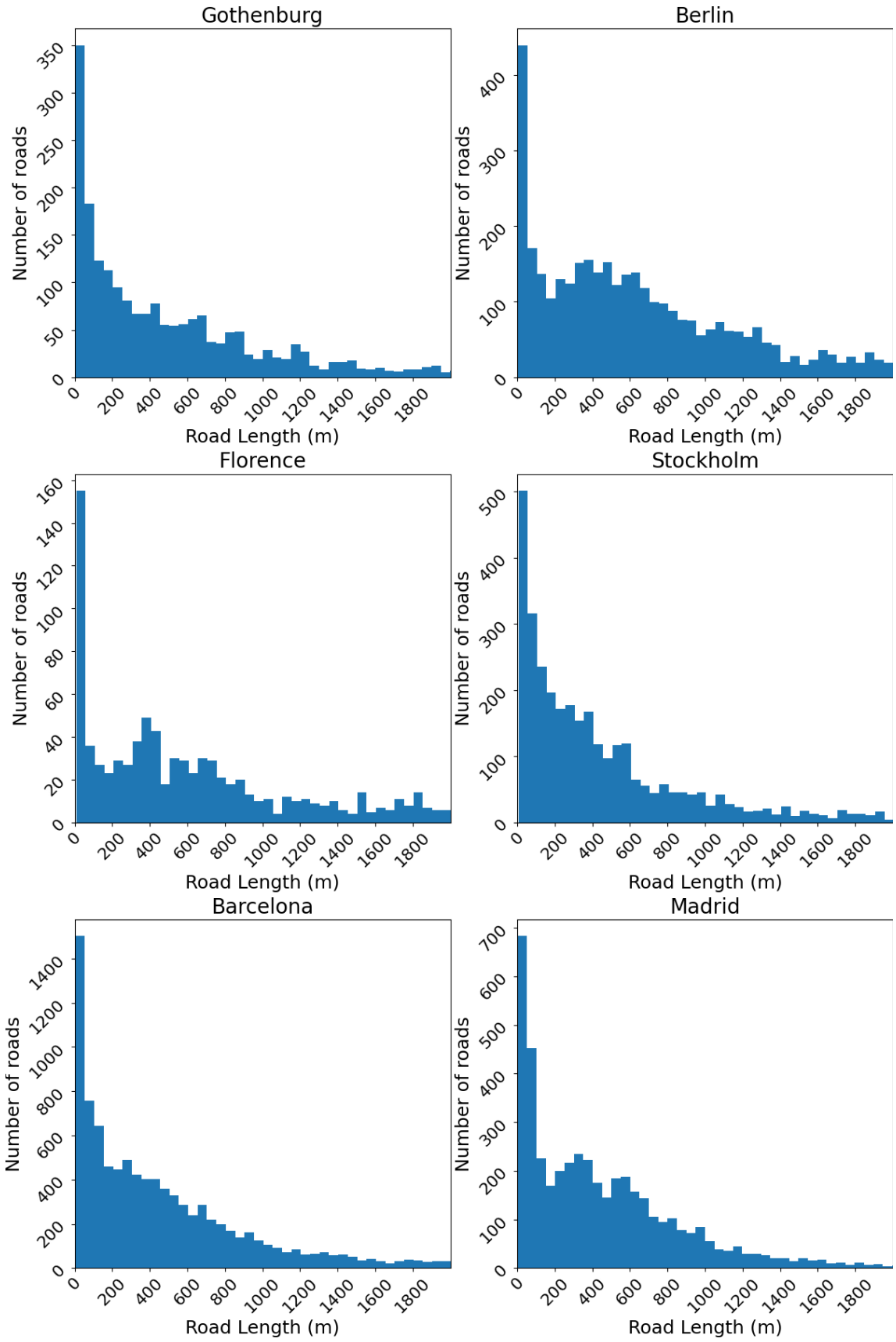


Figure 4.4: Plots showing the road length distribution in each city used in the evaluation of the classifiers. All cities have a large portion of roads longer than 100 meters.

The reason behind only selecting six out of all cities is due to our limited time and hardware capabilities. Furthermore only European cities were selected, this is due to the weather data only covering Europe. The selected cities are evenly distributed between non-capitals and capitals. The rationale is that there are possible differences between capitals and non-capitals that may affect the outcome of our models. For example, capitals are in general larger and more populated and because of this the HERE data may have better coverage in the capitals compared to non-capitals.

4.5.7 Preprocessing

Since the number of roads from each city is often in the thousands we will perform a number of pre-processing steps to select a subset of the roads. The pre-processing steps include filtering roads based on missing values, filtering on road length and sampling 100 roads per city. Since the number of available roads differ between cities, this guarantees equal contribution from each city when training the models. It also makes training the models feasible within the time-frame of this project.

4.5.7.1 Missing values

The original HERE traffic speed data includes missing values at times where real-time measurements weren't available. A subset of roads are chosen based on the percentage of missing values. The threshold is set to 20% so that all roads that have more than 20% missing values are filtered out from the models. This is to get sufficient quality training samples from each road.

4.5.7.2 Imputation

The remaining roads after filtering will be imputed using a mean-based imputation strategy.

4.5.7.3 Filtering on road length

The roads are filtered on road length so that all roads less than 100 meters are excluded.

4.5.7.4 Sampling roads

The roads are randomly sampled from the set of roads acquired after accounting for missing values and road length. The sampling is done using the closeness centrality metric. Each road has an associated closeness value which gives useful information in what part of the city that road is situated. A higher closeness means that a lot of other roads are close to the current road. This probably indicates a road more towards the center of the city. A medium closeness probably indicates a road that is a little further from the center but still in the city. A low closeness probably indicates a road in the outskirts of the city. The roads will be split into these three groups based on the distribution of their closeness values. Plotting the distribution of the closeness values in each city we can see normal distributions in every case.

We define the mean of the centrality values of a city as $\mu_{centrality}(city)$ and the standard deviation as $\sigma_{centrality}(city)$. Using this and the fact that the centralities are normally distributed the three groups can be defined as:

Low density roads: All centrality values below the first quantile. This can be defined as the range $[min_{centrality}(city), \mu_{centrality}(city) - \sigma_{centrality}(city)]$. This range will contain 15.9% of the roads.
Medium density roads: Between first and third quantile. This can be defined as the range $[\mu_{centrality}(city) - \sigma_{centrality}(city), \mu_{centrality}(city) + \sigma_{centrality}(city)]$. This range will contain 68.2% of the roads.
High density roads: All centrality values above the third quantile. This can be defined as the range $[\mu_{centrality}(city) + \sigma_{centrality}(city), max_{centrality}(city)]$. This range will contain 15.9% of the roads

This can be further illustrated by a normal distribution plot. Figure 4.5 shows a normal distribution with a mean of 0 and standard deviation of 1. The purple area shows 68.2% of the values which will correspond to the medium density roads. The black areas to the left and right of the purple will correspond to the low density roads and high density roads respectively.

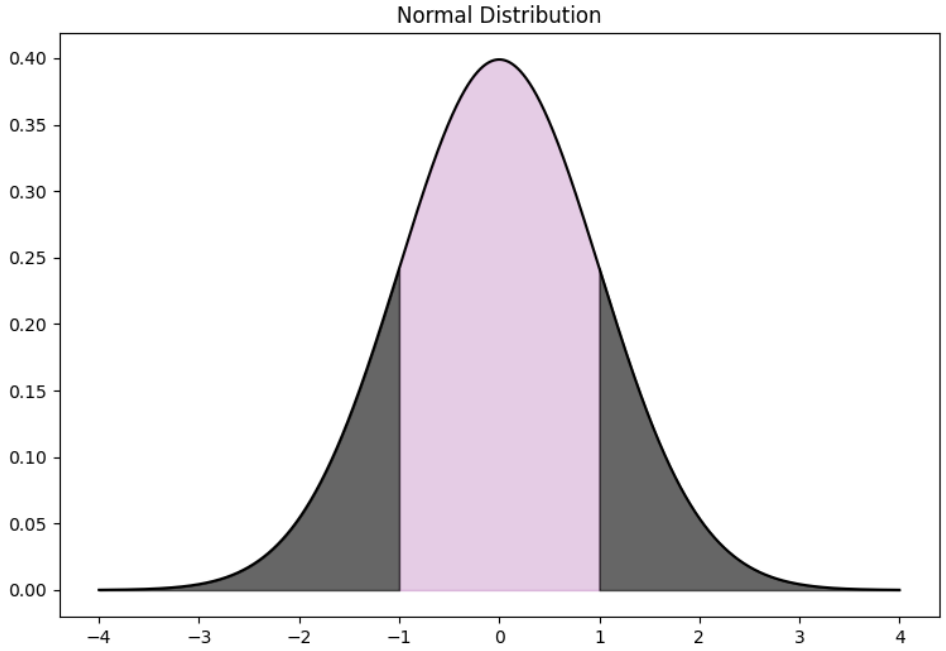


Figure 4.5: Normal distribution plot showing the range of medium density roads as the area in the middle, the range of low density roads to the left and high density roads to the right.

4.5.8 Evaluation

This part will go over the evaluation metrics , the cross validation strategy and the different evaluation scenarios used in this thesis. We propose three different evaluation scenarios in order to answer the questions presented in the beginning of this chapter.

The metrics used for evaluation will be accuracy, precision, recall and ROC/AUC.

5-fold Group Shuffle split-cross validation will be used. The data will be grouped based on the road id's generated from the concatenation of the city name and the current road index in the list of roads. This is to ensure that the same road doesn't end up in both the train and test set in a cross validation fold. The train-test data will be split in 80% train data and 20% test data for each fold.

We define **three evaluation scenarios** in order to test out the generalization performance of the models. Scenario I aims to test the generalization performance between a smaller group of non-capitals and capitals separately, Scenario II aims to test the generalization performance for a mixed smaller group of non-capitals and capitals and Scenario III aims to test the performance when the group of Scenario II is increased in size. All three models will be evaluated in scenario I while the other scenarios will be evaluated using the best performing model from scenario I.

In all scenarios one city will be reserved for testing. The remaining cities are used to construct the input features from sampled and filtered roads. Accuracy, precision, recall and ROC/AUC are calculated in each fold for performance evaluation. A plot containing the 5 folds' ROC-curves and an averaged ROC-curve across all folds is generated and saved. Furthermore, a plot containing the precision-recall curves for each fold will be generated and saved for both the cross validation set and test city validation set. For the Random Forest Classifier the importance of every feature is also saved and presented for all Scenarios. This will yield a value for each feature that represents the importance of that feature in proportion to all other features. The value is between 0 and 1 where a higher score corresponds to a higher importance.

Scenario I
In Scenario I, non-capitals and capitals are treated and evaluated as two separate groups. Scenario I will produce 6 classification models for each type of classification model yielding a total of 18 models. The performance will be assessed according to the evaluation metrics and a few examples of ROC-curves and precision-recall-curves. The examples will include one configuration from the non-capital group (first entry in Table 4.7) and one from the capital group (first entry in Table 4.8). The different configurations of train- and test-sets for the non-capitals are shown in Table 4.7. For the capitals they are shown in Table 4.8.

Table 4.7: Evaluation group of non-capitals

Train	Test
Gothenburg, Florence	Barcelona
Gothenburg, Barcelona	Florence
Florence, Barcelona	Gothenburg

Table 4.8: Evaluation group of capitals

Train	Test
Stockholm, Madrid	Berlin
Stockholm, Berlin	Madrid
Madrid, Berlin	Stockholm

Each evaluation group contains three configurations of cities. Looking at the configurations in the non-capital group we can take the first row as an example. Here the training set consists of Gothenburg and Florence. Data will be gathered from 100 roads sampled from Gothenburg and 100 roads sampled from Florence and will be the input vector to the model. The test set is Barcelona and will be used to test the fitted models created during cross validation. Since we are using 5-fold cross validation, 5 models will be created. These will be used to predict the classes in the data of Barcelona. The different metrics are calculated based on the predicted classes during each fold. The returned metric values of all folds are averaged and saved as a final result.

Scenario II
Scenario II incorporates both non-capitals and capitals in the same configurations in order to observe any performance differences when they are mixed. In order to make the comparison fair the size of each configuration is still kept to 3 cities, with 2 training cities and 1 test city. Every city will be used as a test city, giving a total of 6 test cities, for better comparability to Scenario I. In order to mitigate the effect of certain city configurations performing better we will randomly sample the set of train cities in order to create 10 configurations for each test city. Accounting for the 6 test cities this will yield a total of 60 models.

Each test city will have 10 values per metric. The mean of every metric value can be compared to the corresponding metric value from scenario I in order to measure the effect on performance when mixing capitals and non-capitals.

Scenario III
In Scenario III we want to further measure how the performance is affected when increasing the number of cities used for training. Scenario III is constructed in the same way as scenario II by randomly sampling the training cities for each test city. The following configurations will be used during scenario III:

- Scenario III: Configuration 1

  - Scenario II 3 training cities, 1 test city. 60 model configurations in total.
- Scenario III: Configuration 2
  - Scenario II 4 training cities, 1 test city. 60 model configurations in total.
- Scenario III: Configuration 3
  - Scenario II 5 training cities, 1 test city. 60 model configurations in total.

This will in total create 180 models all of which will be compared in the same way as scenario II using box plots of the acquired metric scores. In order to compare the different scenario III configurations the scores of the 60 models constructed during each variation of scenario II will be aggregated into one box plot for each metric and scenario III-configuration.

5
Results

In this chapter the analysis results are presented in Section 5.1, in Section 5.2 the results from forecasting are presented and in Section 5.3 the results from classification are presented.

5.1 Analysis

This section will present the results from the Augmented Dickey-Fuller tests that were performed in order to check if the data was stationary or not.

5.1.1 Checking for stationarity

Augmented Dickey-Fuller test was performed on all roads with less than 20% missing values. Using a 95% confidence interval it was concluded that the data is stationary. Thus, no transformations were needed in order to perform forecasting using AR.

5.2 Forecasting

In this section the results from the forecasting are presented. The aim was to see if more advanced models achieved a better result by looking at the RMSE scores of the individual models. The scores are rounded to 2 decimal places.

A line plot showing the scores for each model is presented in Figure 5.1. The Baseline model achieved an average score of 7.50. The AR model achieved a lower score of 6.98 and the MLP univariate model achieved a marginally lower score of 6.89 compared to the AR model. Interestingly, adding precipitation and temperature data of a city to the MLP model resulted in a worse score of 6.93 compared to the univariate case. The difference is marginal when looking at all models except for the Baseline case.

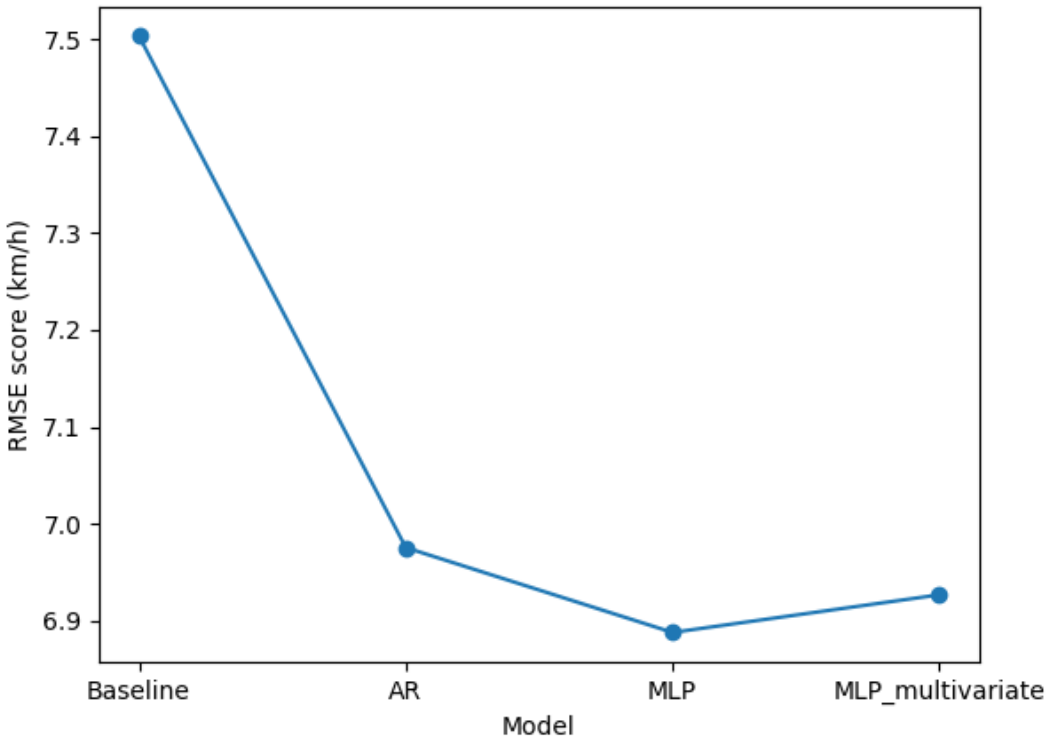


Figure 5.1: A line plot showing the average RMSE score for the Baseline, AR, univariate MLP and multivariate MLP model.

Going by these results a small improvement can be seen when using more advanced models although quite insignificant in this case. The use of weather variables in the form of precipitation and temperature doesn't improve the model either.

5.3 Classification

In this section, an example of sampled roads are presented in Section 5.3.1. In Section 5.3.2 the results from the Naive Majority Classifier, Logistic Regression Classifier and Random Forest Classifier are presented. These are presented in the form of metric plots covering the scores of each metric for both Cross Validation and for the evaluation city. The feature importances are also presented using a color-coded plot for each feature. Lastly, the resulting ROC-curves and precision-recall curves for two configurations are presented.

5.3.1 Sampling roads

In order to get a good picture of how the sampling works an example of sampled roads is presented in Figure 5.2. The left map of 5.2 shows the sampled roads highlighted in cyan and the right map shows all the roads in green. The city used for this example is Gothenburg, Sweden.

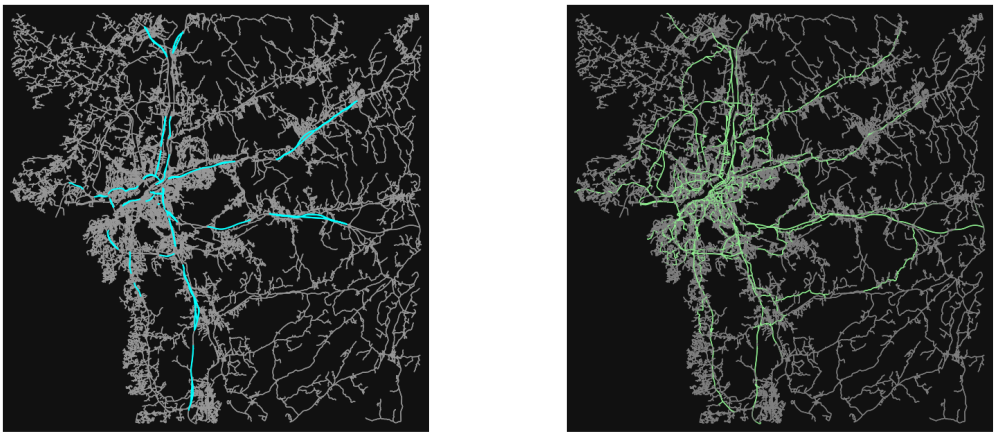


Figure 5.2: The maps show the roads in the city of Gothenburg, Sweden. The gray roads represent all roads of the city whereas the highlighted roads are the ones corresponding to the HERE data.

The left map shows quite a good distribution of the kind of roads that were picked in the sampled case. There are more roads picked that are towards the center of the city as expected by the design of the sampling method described in Section 4.5.7.4.

5.3.2 Scenario I

In this section the results from Scenario I are presented. In Section 5.3.2.1 the feature importance plot for both capitals and non-capitals is presented. In the following sections the different classifiers' results are presented in the form of ROC-curves, precision-recall-curves and other summary statistics plots. For each classifier one non-capital configuration and one capital configuration is used to illustrate the results. The two configurations are shown below:

Non-capital configuration: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.

Capital configuration: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

For all the configurations see Appendix A.

5.3.2.1 Feature Importances

In this part, the feature importances are presented in Figure 5.3 for capitals and in Figure 5.4 for non-capitals. These are calculated on the fitted Random Forest Classifiers.

Capitals

The feature importances for the capital configurations are presented below in 5.3. Each pair of train cities for the capitals is presented below on the x axis. The three configurations of train cities include Berlin & Madrid, Berlin & Stockholm and Stockholm & Madrid. The corresponding test cities are Stockholm, Madrid and Berlin in that order.

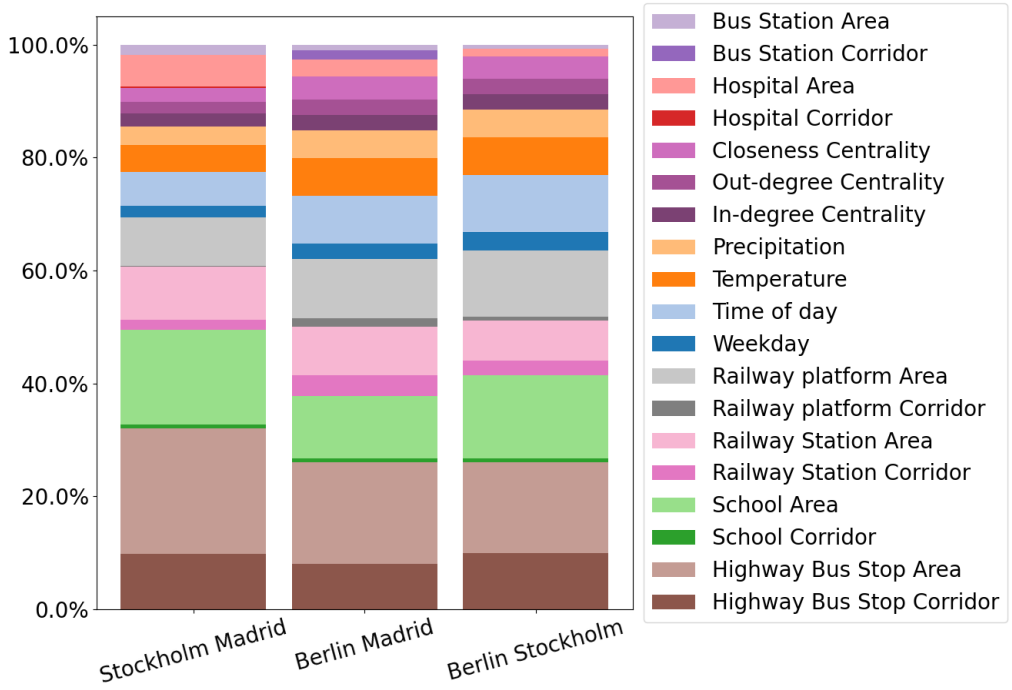


Figure 5.3: A stacked bar chart showing the importances of every feature for each fitted pair of capital cities. The x axis shows the pairs of cities that were used when fitting the classifier and the y axis shows the percentage of contribution of the features. Each feature is color-coded separately but similar features are coded in different shades of the same color. One example is temperature and precipitation where both are of a different shade of orange. The feature importances are sorted on the grouped similar features. One such group is *Highway Bus Stop Area & Corridor*.

The bar chart for the capital pairs shows that the most important features out of the 19 features for predicting traffic delays are bus stops. The Highway Bus Stop Area & Corridor accounts for over 20% of the feature importance in all three pairs and over 25% in Stockholm & Madrid. *Hospital Area*, *Hospital Corridor*, *Bus Station Area* and *Bus Station Corridor* are consistently the least important features for the capital cities. All other groups contribute roughly around 10% to the importance more or less. Interestingly, the Area seems to be more important the Corridor. The different kinds of centrality metrics don't contribute that much to the predictability, only accounting for a couple of percentages each.

Overall, Highway Bus Stop Area and School Area are the top features in the capital configurations.

Non-capitals

The feature importances for the non-capital configurations are presented below in 5.4.

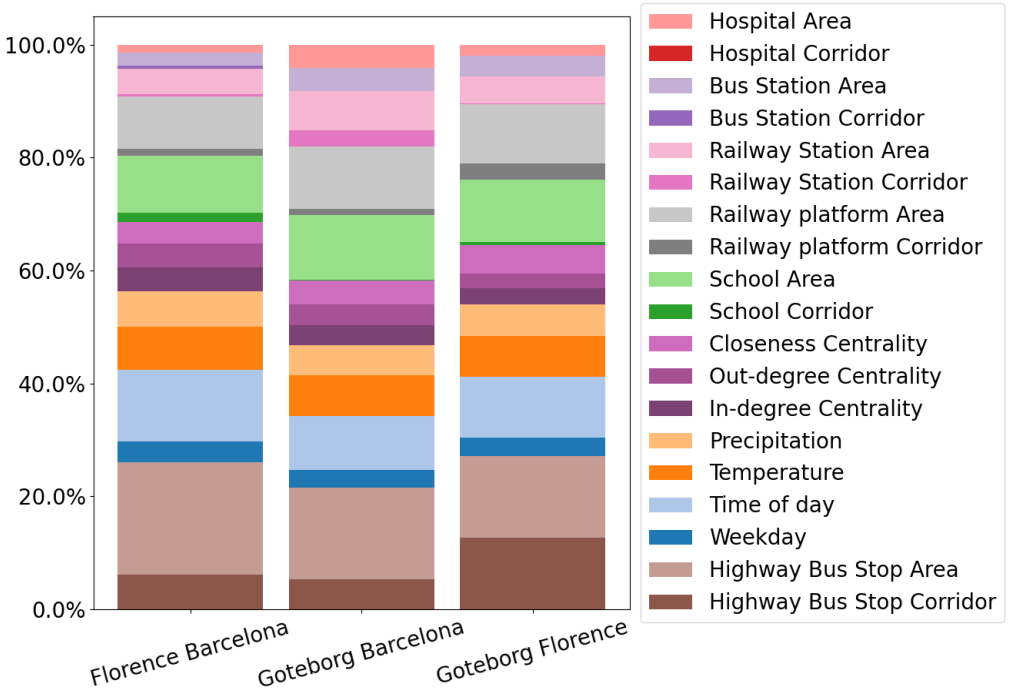


Figure 5.4: A stacked bar chart showing the importances of every feature for each fitted pair of non-capital cities. The colors of each feature is the same as the capitals bar chart for easier comparison. Since there can be differences in the feature importances compared to the capitals the sort order might differ.

Compared to the capitals group, the non-capitals group has a different overall order of the feature importances. *Highway Bus Stop Area & Corridor* is still considered the best performing group of features accounting for over 20% of the importance. *School Area* accounts for a smaller percentage of importance compared to the capitals group but is still one of the top contributors together with *time of day* and *Railway Platform Area*.

Despite some small differences, the importances seem to follow quite closely between the two groups.

5.3.2.2 Naive Majority Classifier

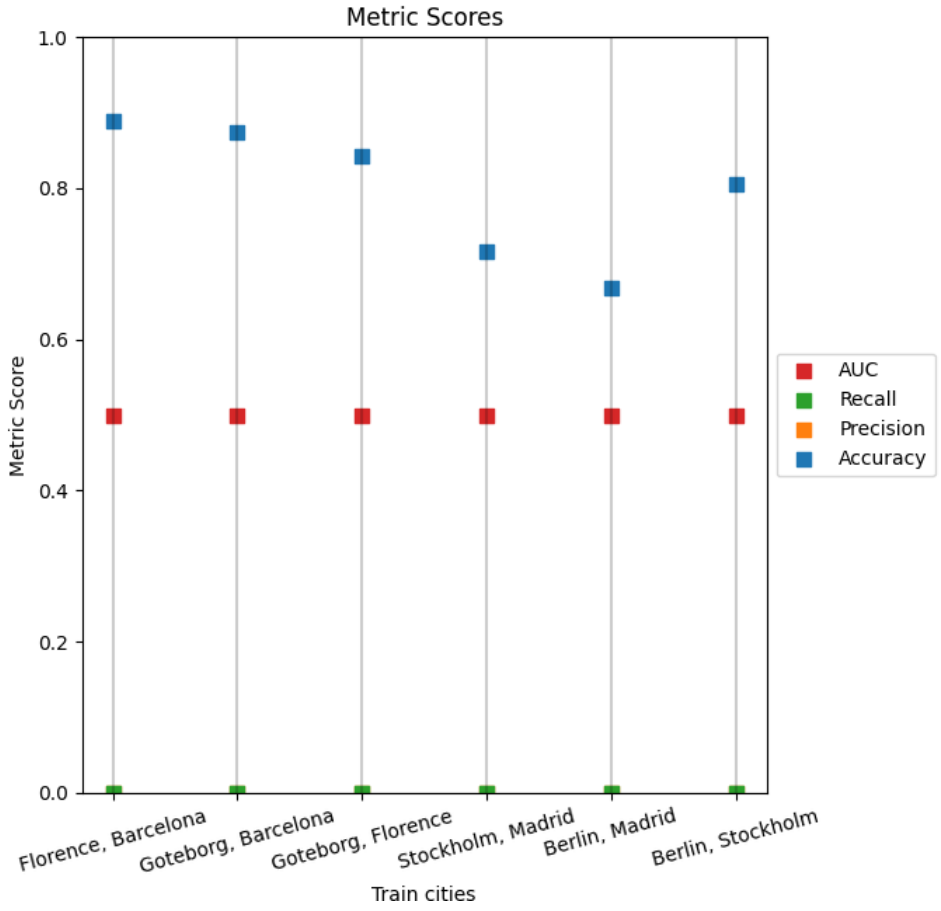
In this section the results of the Naive Majority Classifier are presented. Summary statistics showing the metric scores for different configurations are presented first, followed by ROC-curves and precision-recall-curves.

Summary statistics

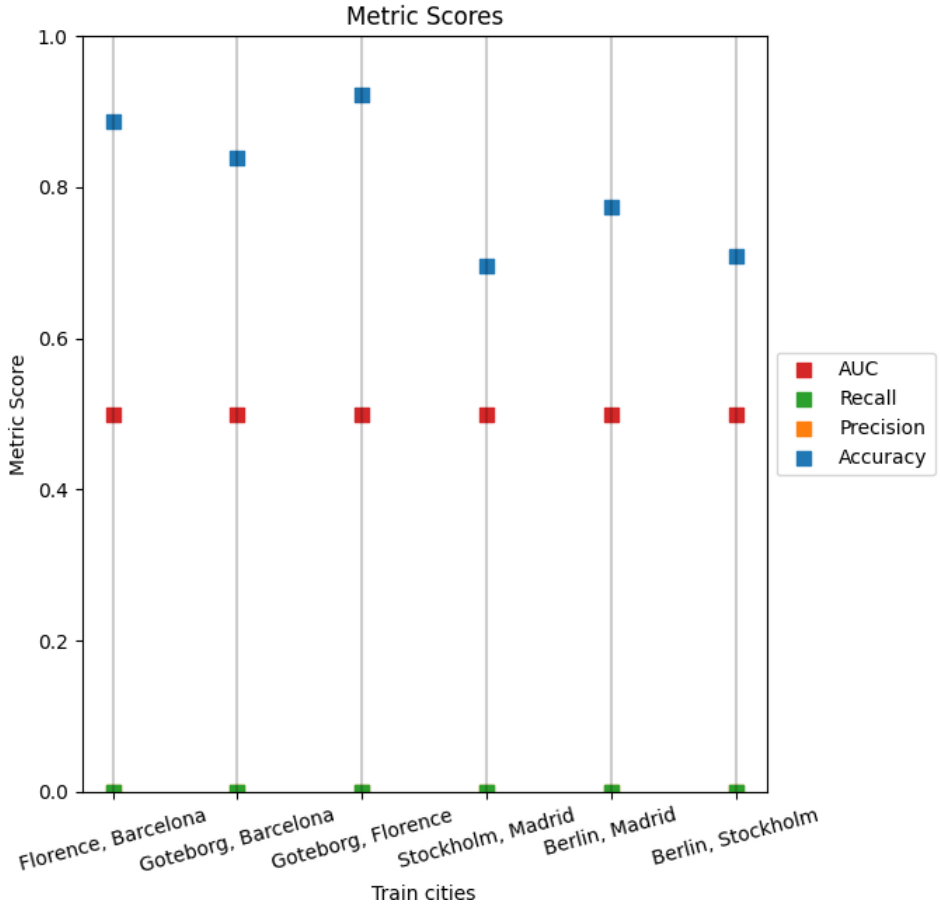
The metric scores for each configuration are presented below. Cross Validation results are shown in Figure 5.5a and Validation results are shown in Figure 5.5b.

Figure 5.5: Metric Scores for each configuration. For each train configuration the Accuracy, Precision, Recall and AUC is plotted on a straight line for easy comparison between configurations. The top plot corresponds to the average test score acquired from the cross validation folds. The test cities (used for calculating the validation scores for the plot in the bottom) are Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid in that order.
Note: **Each plot** consists of both **non-capital** and **capital configurations**. The first three entries from the left corresponds to the non-capital configurations and the remaining three entries corresponds to the capital configurations.

(a) Cross Validation



(b) Validation i.e test city



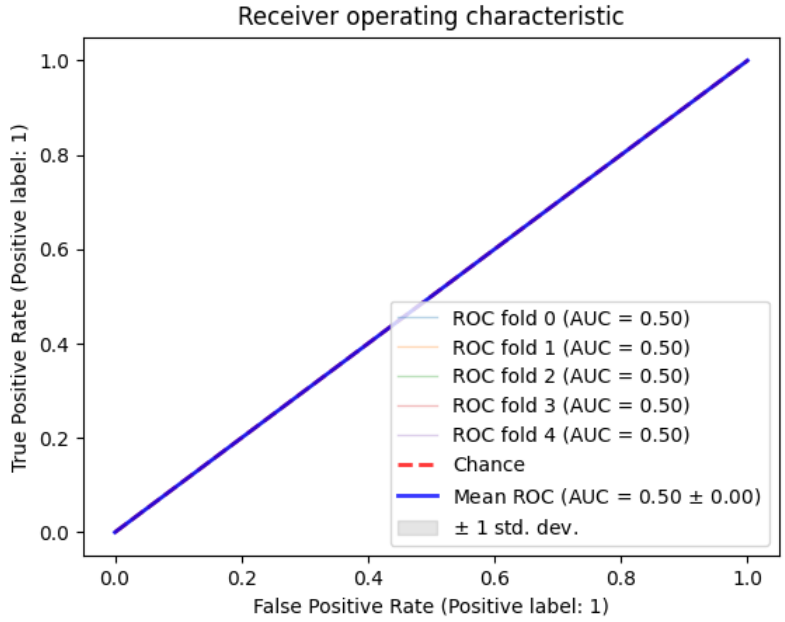
The summary statistics show for both the cross validation and test validation scores a high accuracy of over 75% in all configurations. As expected by the Naive Majority Classifier, both precision and recall is 0 showing that the classifier cannot correctly separate the classes. This is supported by the AUC score which is representative of a classifier that picks the classes at random (random classifier). In both plots the accuracy is lower for the capital configurations compared to the non-capital configurations. This shows that there are more delays in the capital configurations compared to the non-capital ones.

ROC-curves

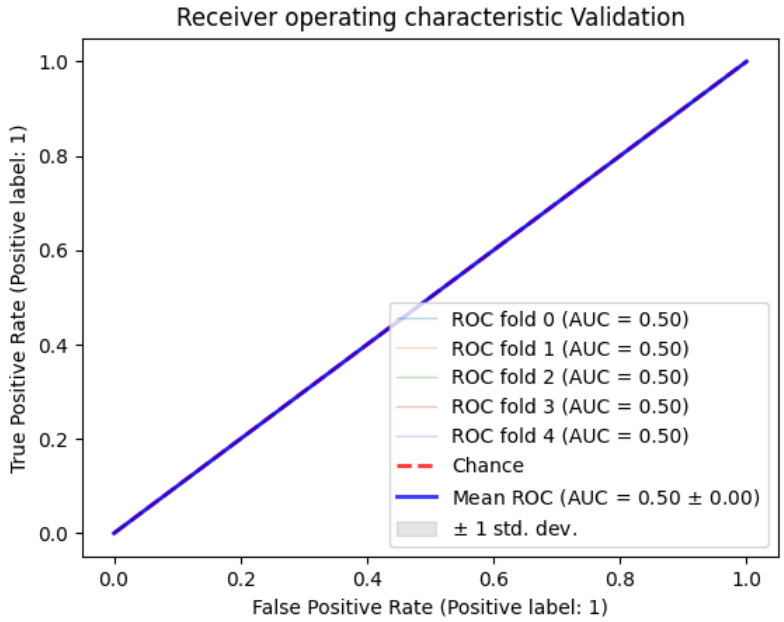
In this section the ROC-curves for one configuration are presented in Figure 5.6. The configuration includes Gothenburg and Florence in the train set and Barcelona in the test set used for validation. The curves for cross validation is presented in Figure 5.6a. For the validation set the results are shown in Figure 5.6b.

Figure 5.6: Train set: {Gothenburg, Florence} Test set: {Barcelona}.

(a) Cross Validation



(b) Validation i.e test city



Both figures are identical showing an AUC score of 0.50 for the classifiers in each fold. This results in the blue mean curve that can be observed in the figures. The curve covers the dashed red curve representing a random classifier which means

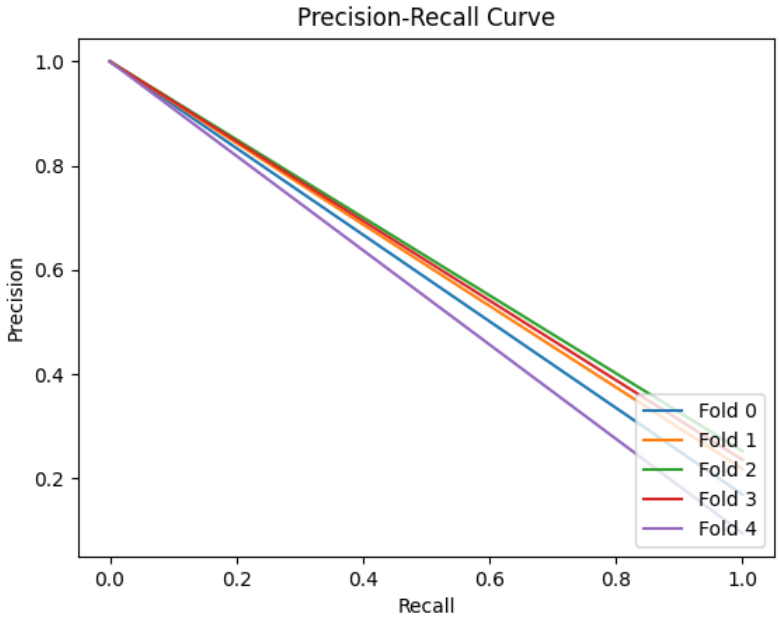
that the classifier is as good as a random classifier and has no ability to separate the classes.

Precision-recall-curves

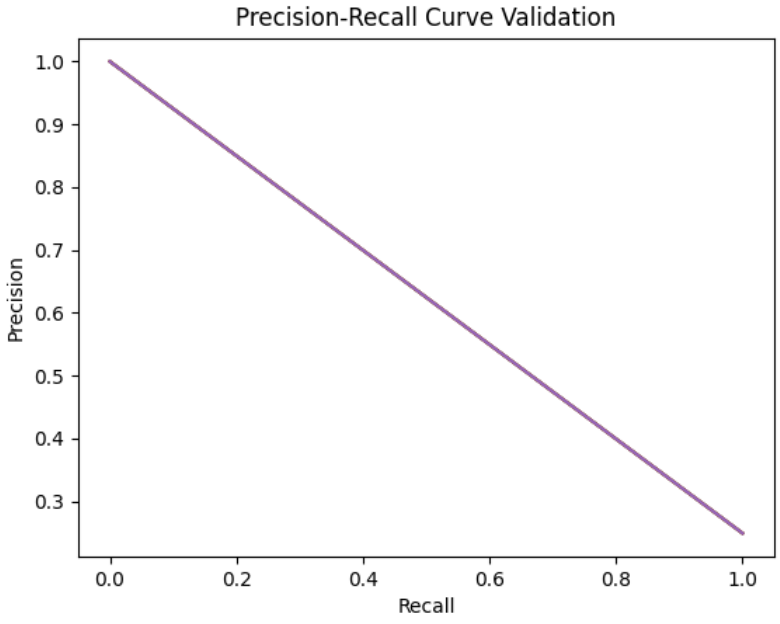
In this section the precision-recall-curves for the same configuration as before are presented in Figure 5.7. The curves for cross validation is presented in Figure 5.7a. For the validation set the results are shown in Figure 5.7b.

Figure 5.7: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.

(a) Cross Validation



(b) Validation i.e test city



Both figures show similar results with slight variations in the cross validation figure. This is expected in the cross validation case since during each fold the data is split differently, creating different imbalances in the resulting splits. The test set for Fold 0 in 5.7a for example has around 20% delay classes and 80% non-delay classes which can be observed by looking at the precision value for the rightmost recall value. This is also known as the baseline of the classifier which shows if the classifier is random or not. Since the majority classifier only guesses the majority class an adjustment of the threshold won't change the precision and recall values. Moreover, the extreme values in the top left of each figure and bottom right of each figure are always plotted by the design of the plotting function. Since the values of recall and precision don't change with the threshold these will be the only two resulting points in this case, creating the diagonal lines in the figures. This might be misleading since it appears to be above the baseline. Because of this, precision-recall curves are not suitable for evaluating this type of classifier.

5.3.2.3 Logistic Regression Classifier

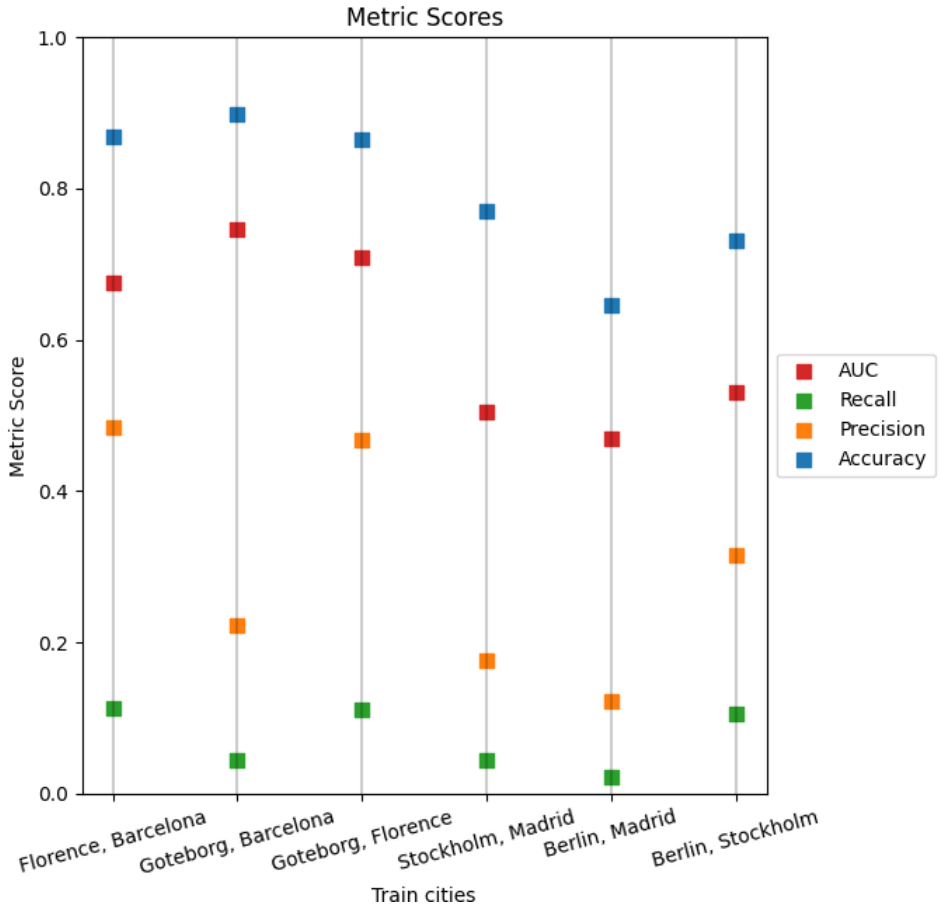
In this section the results of the Logistic Regression Classifier are presented. Summary statistics showing the metric scores for different configurations are presented first, followed by ROC-curves and precision-recall-curves.

Summary statistics

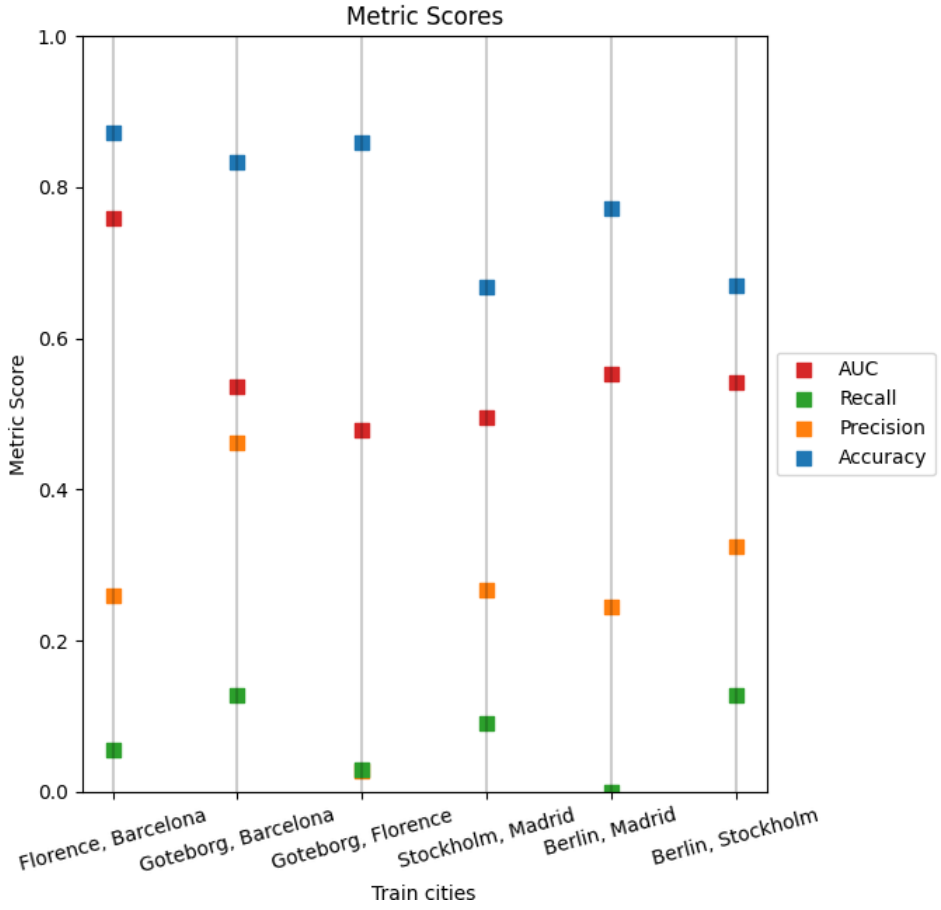
The metric scores for each configuration are presented below. Cross Validation results are shown in Figure 5.8a and Validation results are shown in Figure 5.8b.

Figure 5.8: Metric Scores for each configuration. The cross validation scores are on the plot in the top. The test cities (used for calculating the test validation scores for the bottom plot) are Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid in that order.

(a) Cross Validation



(b) Validation i.e test cities



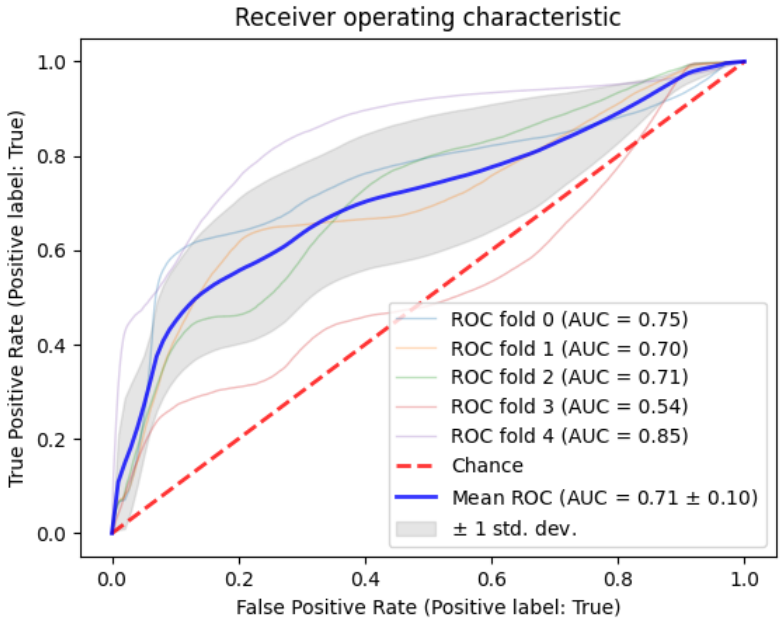
The accuracy scores are high for the Logistic Regression Classifier and around the same as the Naive Classifier. For both the cross validation and validation case a higher AUC can be observed in all configurations compared to the Naive Classifier which shows that the classifier is also better at separating the classes and performs better than a random classifier. For the test validation set, the mean AUC score is around 0.5 for most of the configurations which is not very good. This indicates performance of a random classifier when testing the models on new cities.

ROC-curves

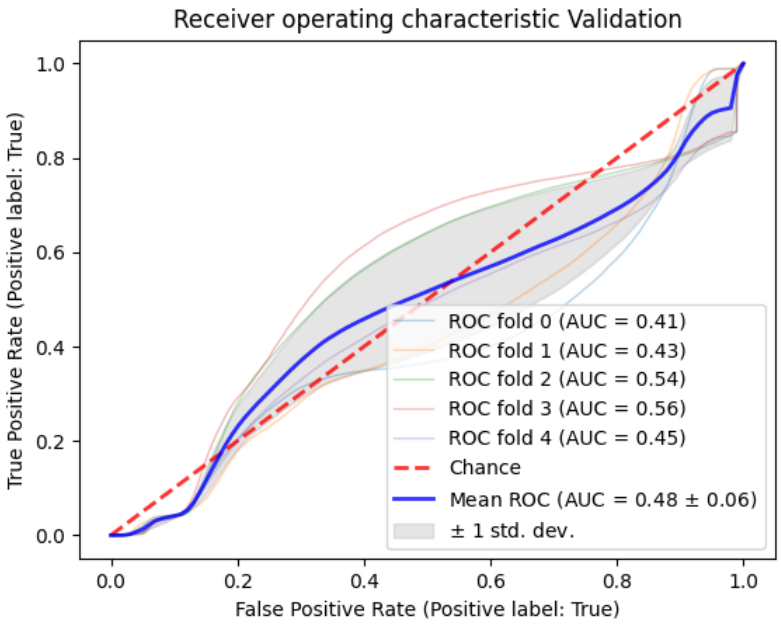
In this section the ROC-curves for a non-capital configuration are presented in Figure 5.9 and for a capital configuration in Figure 5.10. The non-capital configuration is the same as for the Naive Majority Classifier and includes Gothenburg and Florence in the train set and Barcelona in the test set used for validation. For the non-capital configuration the curves for cross validation is presented in Figure 5.9a and for the validation set the results are shown in Figure 5.9b. The capital configuration includes Stockholm and Madrid in the train set and Berlin in the test set used for validation. For the capital configuration, cross validation and validation results are presented in Figure 5.10a and Figure 5.10b respectively.

Figure 5.9: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.

(a) Cross Validation



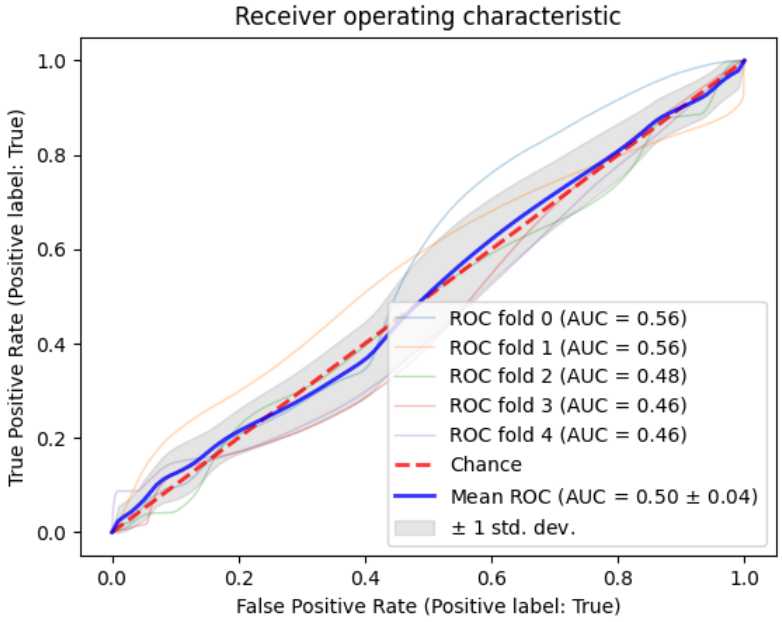
(b) Validation i.e test city



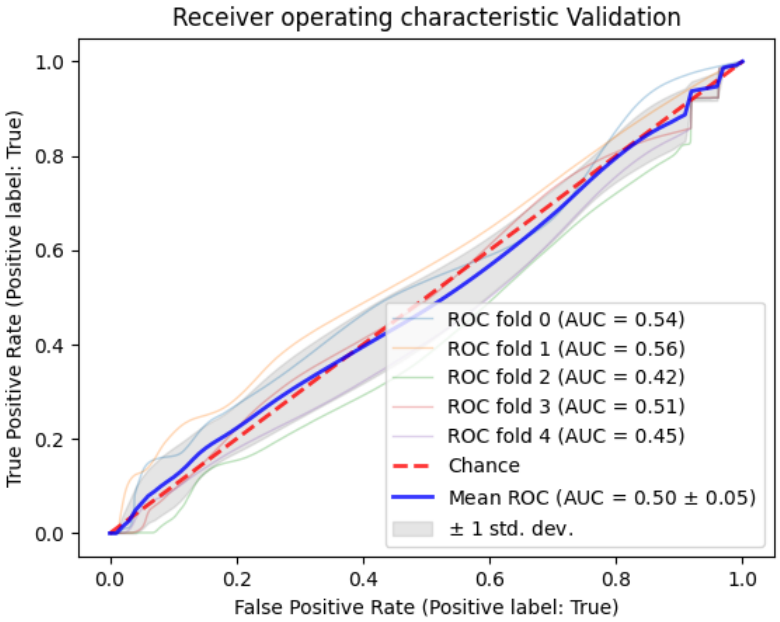
The ROC-curves for the cross validation case show a classifier that performs better than a random classifier in the majority of folds except for fold 3 where it falls below the 50% line in some cases. As for the test validation case it performs conceivably worse where all folds fall below the 50% line at different points. The mean AUC score for the cross validation case is 0.71 but for the test validation case is only 0.48.

Figure 5.10: Train set: {Stockholm, Madrid}. Test set: {Berlin}

(a) Cross Validation



(b) Validation i.e test city



The performance is considerably worse for the capital configuration in the cross validation case. In both the cross-validation case and test validation case the curves for all folds are consistently close to the 50% line. Both curves have a mean AUC

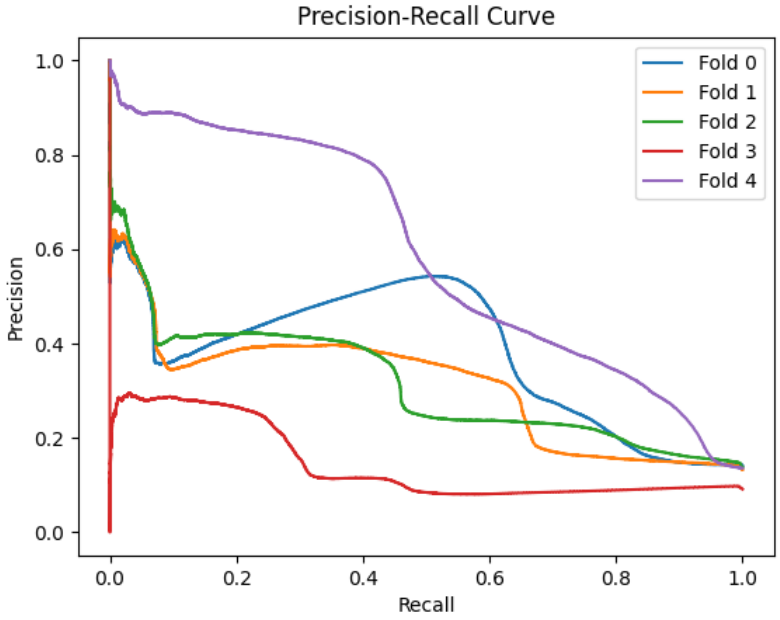
score at 0.5 making this model no better than a random classifier.

Precision-recall-curves

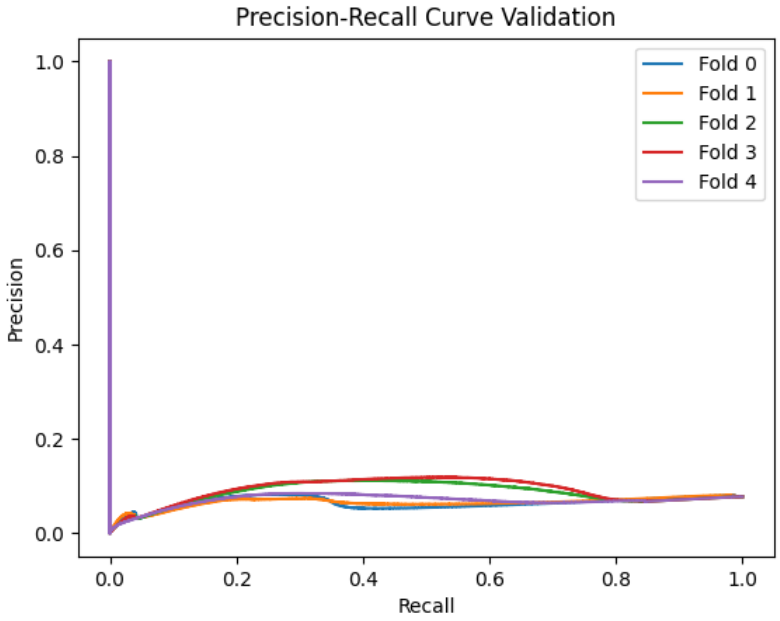
In this section the precision-recall-curves for the non-capital configuration and capital configuration are presented in Figure 5.11 and Figure 5.12 respectively. The cross validation and validation results for the non-capital configuration are presented in Figure 5.11a and Figure 5.11b. The corresponding results for the capital configuration are presented in Figure 5.12a and Figure 5.12b.

Figure 5.11: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.

(a) Cross Validation



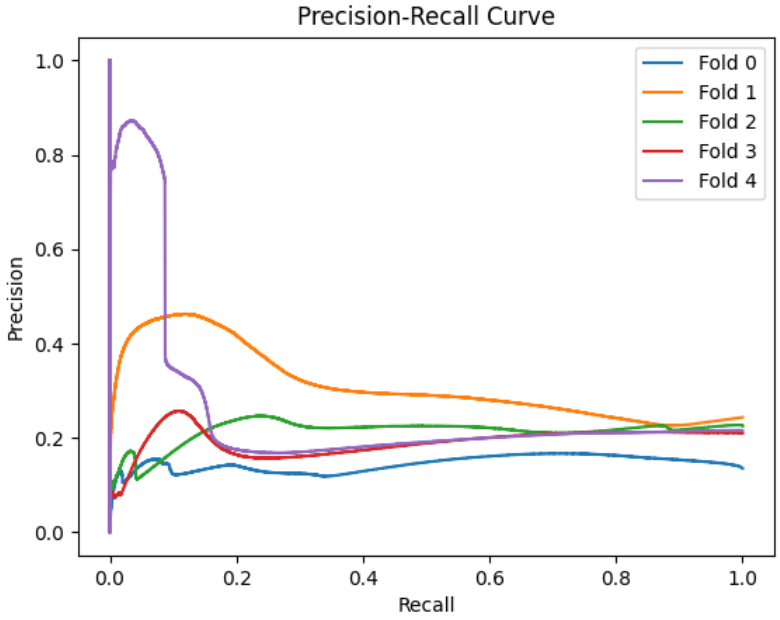
(b) Validation i.e test city



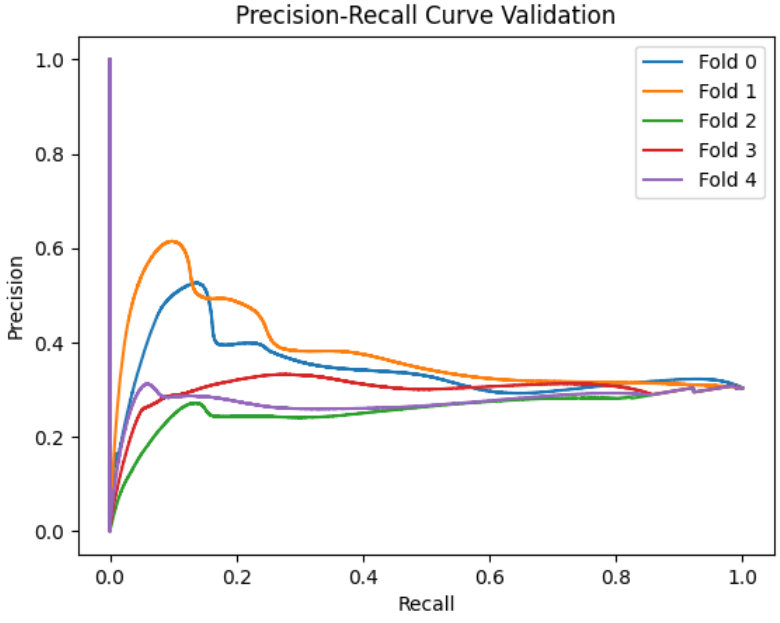
The precision-recall curves for the cross-validation case show that the classifier overall performs better than a random classifier. The majority of folds manages to stay above the baseline thresholds (the rightmost y values for the fold curves), with the exception of fold 3. This can be observed in the ROC-curve (5.9a) in the corresponding TPR range of [0.4,0.8] and FPR range of [0.5, 0.8]. In the test validation set all folds are quite flat and also around the baseline thresholds. The classifier can't be considered better than a random classifier for the test city of Barcelona.

Figure 5.12: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

(a) Cross Validation



(b) Validation i.e test city



Looking at the precision-recall curves once again show sub-par performance for the capital configuration, where several folds fall below the baseline threshold at various recall values in both the cross-validation case and test validation case. The classifier can't really be considered better than a random classifier based on these results.

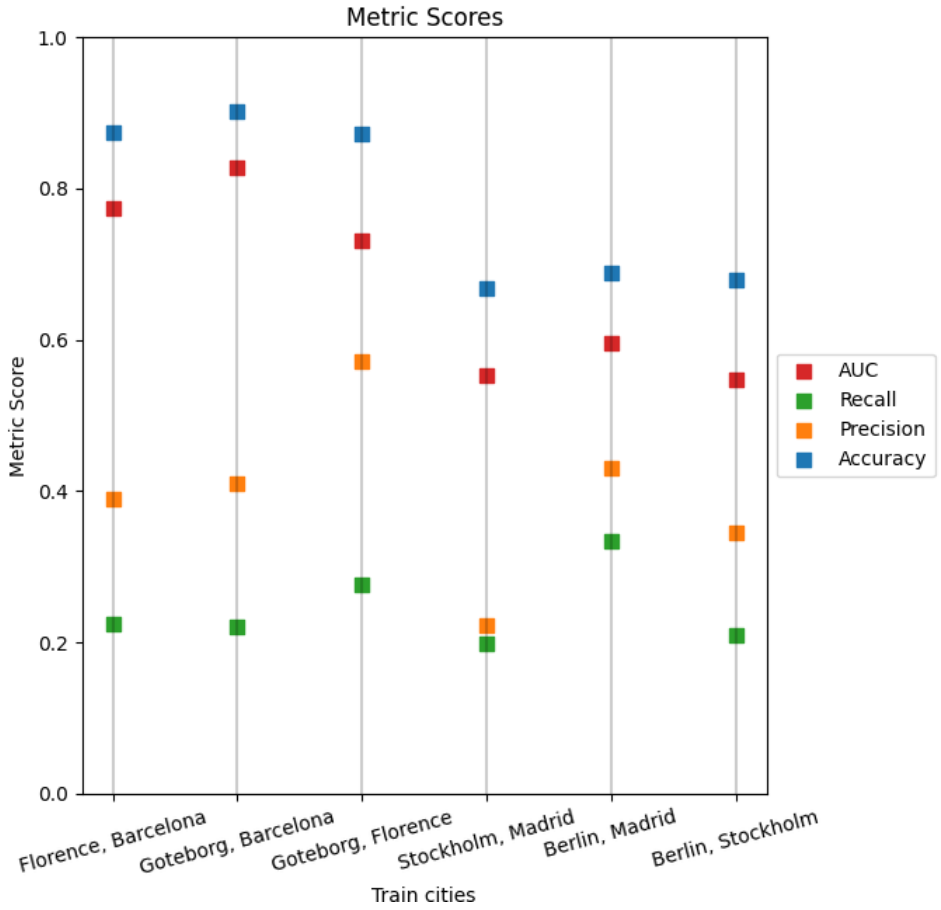
5.3.2.4 Random Forest Classifier

Summary statistics

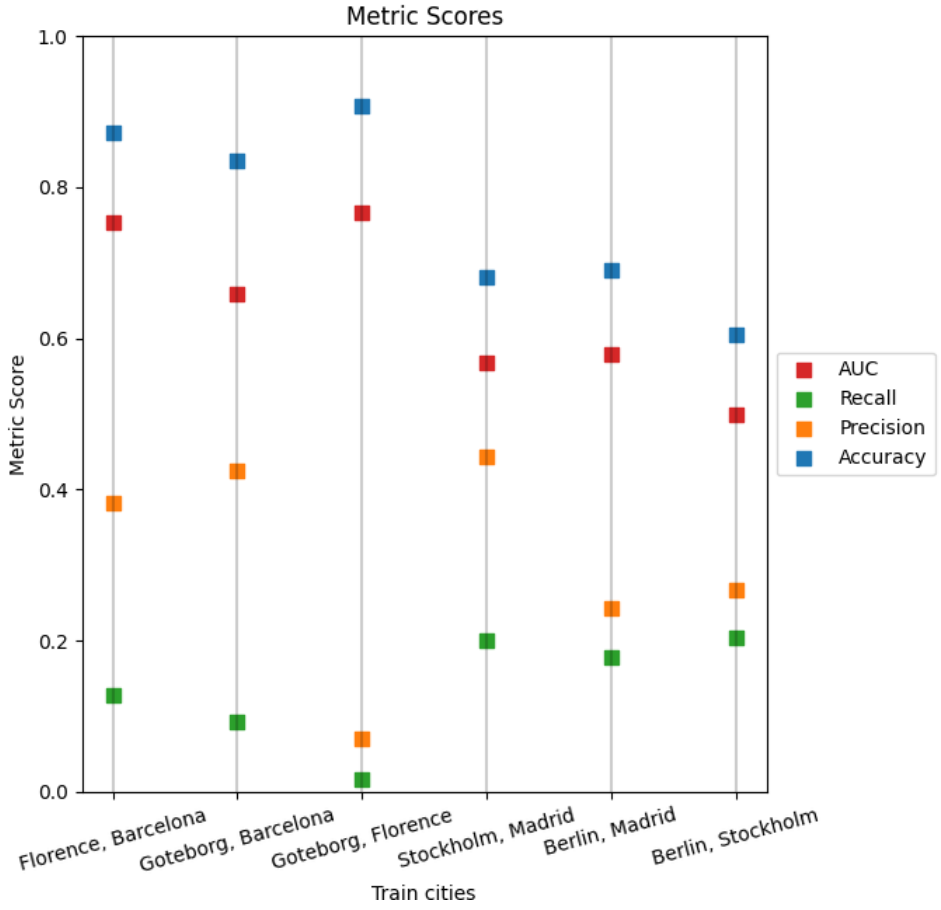
The metric scores for each configuration are presented below. Cross Validation results are shown in Figure 5.13a and Validation results are shown in Figure 5.13b.

Figure 5.13: Metric Scores for each configuration. The test cities (used for calculating the validation scores for the bottom plot) are Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid in that order.

(a) Cross Validation



(b) Validation i.e test cities



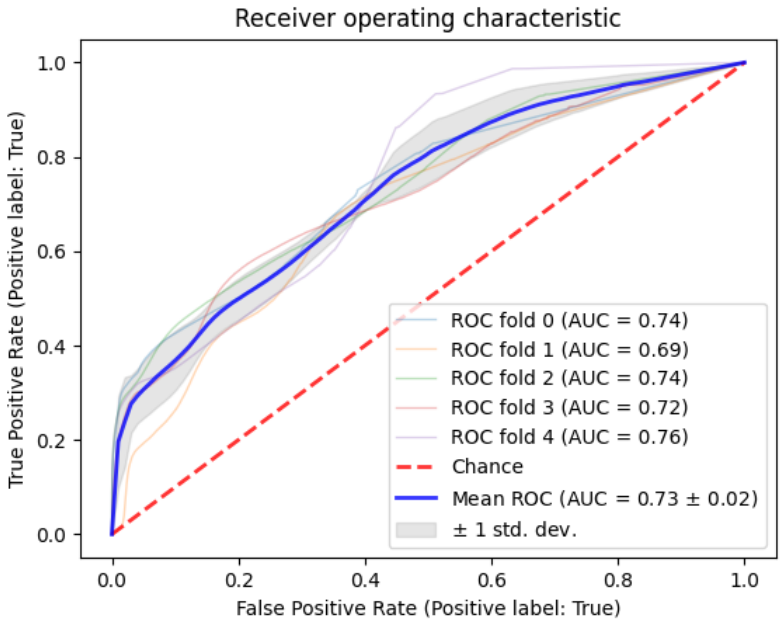
Looking at the cross validation results in 5.13a, it is apparent that the accuracy and the AUC are consistently scoring higher for non-capitals compared to the capitals. The precision also seems to be consistently higher compared to the recall values for each configuration. For the validation set (5.13b) the accuracy and AUC are still higher for non-capital test cities compared to capital test cities. Generally the scores seem to follow the same pattern for both the cross validation and validation results. Overall, the scores for the Random Forest Classifier are higher compared to the scores for the Logistic Regression Classifier in Figure 5.8.

ROC-curves

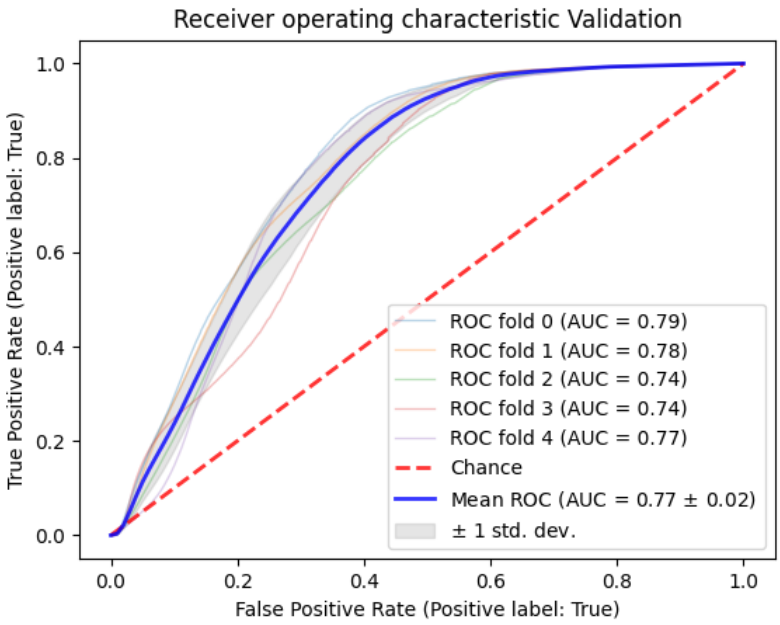
In this section the ROC-curves for one non-capital configuration are presented in Figure 5.14 and for one capital configuration in Figure 5.15. For the non-capital configuration the curves for cross validation is presented in Figure 5.14a and for the validation set the results are shown in Figure 5.14b. For the capital configuration cross validation and validation results are presented in Figure 5.15a and Figure 5.15b respectively.

Figure 5.14: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.

(a) Cross Validation



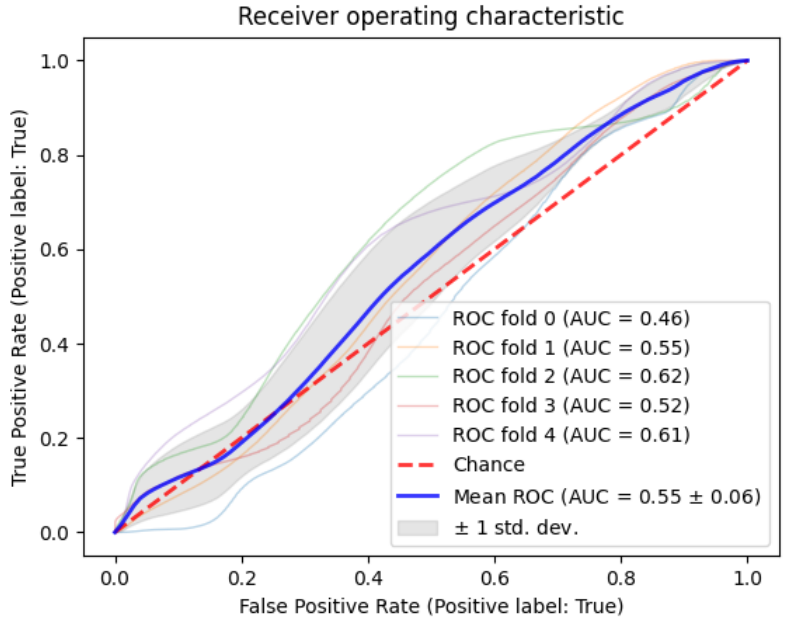
(b) Validation i.e test city



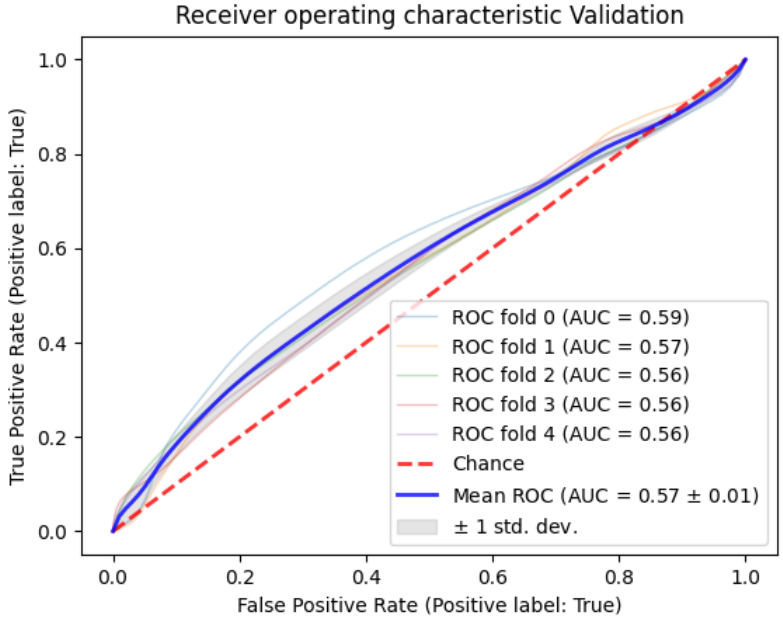
As can be seen in both the cross validation curves and test validation curves the classifier trained on this configuration manages to separate the classes quite well, achieving an AUC score well-above a random classifier. Interestingly, the model is slightly better when looking at the validation set compared to what it was trained on, achieving a mean AUC score of 0.77 compared to 0.73 in the Cross Validation-case.

Figure 5.15: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

(a) Cross Validation



(b) Validation i.e test city



The ROC-curves for the capital configuration show a model with considerably worse performance compared to the non-capital configuration. The test validation curves have a mean AUC score of 0.57 compared to 0.55 in the cross validation case. In the cross validation case the standard deviation is considerably larger at 0.06 compared to the test validation case of 0.01. The cross validation curves fall below the

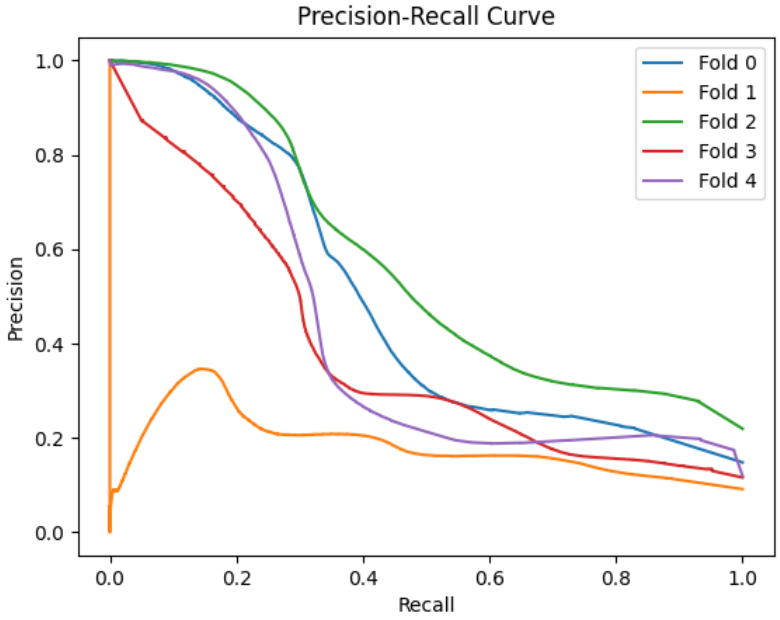
performance of a random classifier for some thresholds.

Precision-recall-curves

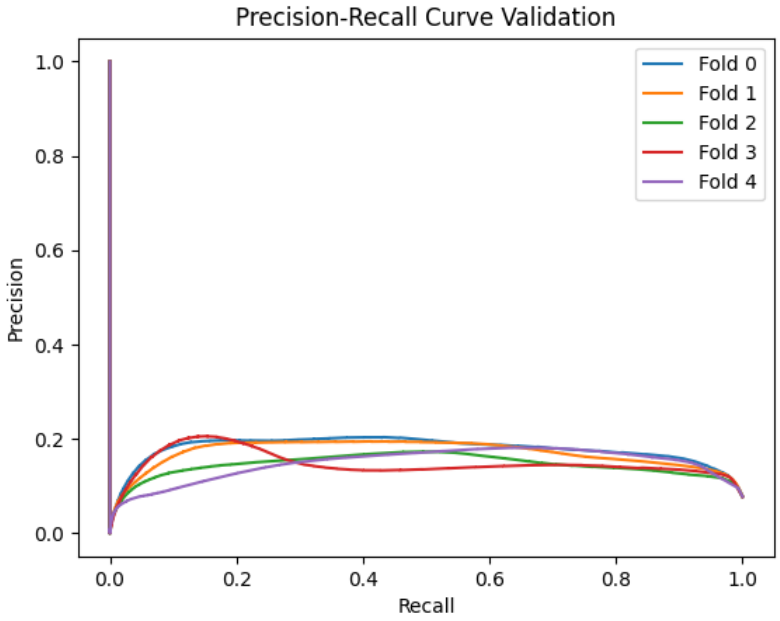
In this section the precision-recall-curves for the non-capital configuration and capital configuration are presented in Figure 5.16 and Figure 5.17 respectively. The cross validation and validation results for the non-capital configuration are presented in Figure 5.16a and Figure 5.16b. The corresponding results for the capital configuration are presented in Figure 5.17a and Figure 5.17b.

Figure 5.16: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.

(a) Cross Validation



(b) Validation i.e test city

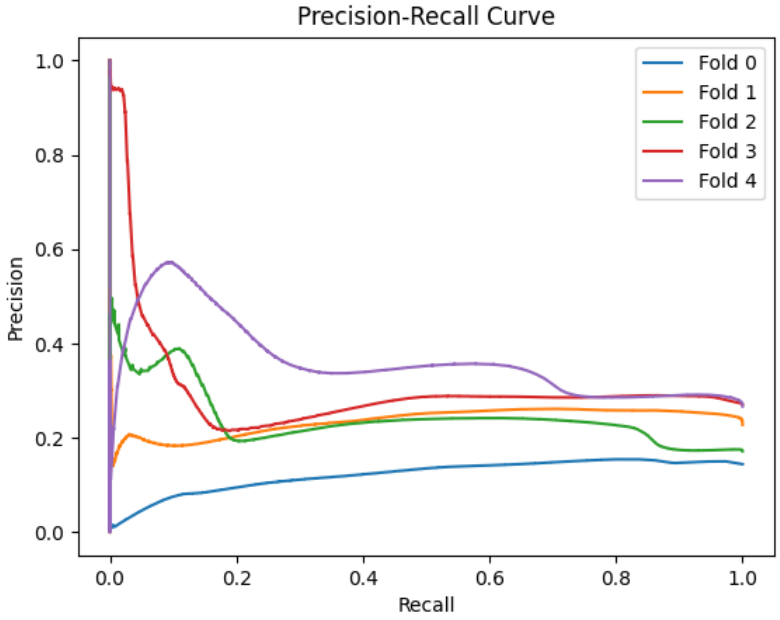


The precision-recall-curves differ between the cross validation and validation case. The test validation set is more stable in the precision range $[0.15, 0.2]$ for the most part whereas the cross validation case has a range of roughly $[0.2, 0.8]$. This is quite expected due to the difference in the amount of positive classes in each cross validation fold. Since all folds stay above the thresholds for the cross validation figure it can be considered better than a random classifier.

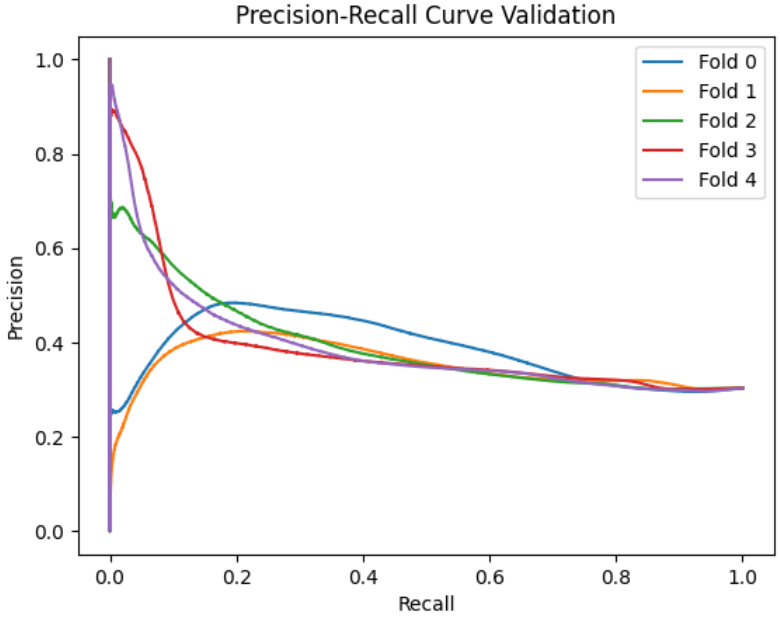
In the test validation case the amount of positive classes doesn't change between the evaluation of the folds. This makes the precision-recall curves more stable. They also follow the same pattern across folds. The curves are lower on the precision-axis but still manages to perform above the threshold, showing that it is better than a random classifier.

Figure 5.17: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

(a) Cross Validation



(b) Validation i.e test city



In the capital configuration the cross validation set appears to be more stable, indicating less variety in the amount of positive classes in each fold. Several folds in the cross validation case fall below the threshold value for lower recall values, which once again shows that for some folds and thresholds it performs worse than a random classifier.

The validation set shows the precision around the threshold value for all the recall values indicating barely better performance compared to a random classifier. For all capital city configurations, results can be seen in Appendix A.3. For the capital configurations the performance is consistently worse compared to the non-capital cities.

5.3.3 Scenario II

For Scenario II the Random Forest Classifier was chosen. The results of the Random Forest Classifier and Logistic Regression Classifier from Scenario I show that the Random Forest Classifier performed better in general. Furthermore, the Random Forest Classifier has the ability to calculate feature importance and is faster at training the model compared to Logistic Regression. This makes it the best choice for Scenario II and Scenario III where the training data increases. In this section the feature importances from Scenario II are presented as the average feature importance for each sub-sample of the 10 samples per test city in Section 5.3.3.1. For convenience, the main results of Scenario II are also summarized below in a metric plot of the averaged scores per sample set in Section 5.3.3.2.

5.3.3.1 Feature Importance

In this part, the feature importances are presented in Figure 5.18 containing each test city with 10 sample configurations.

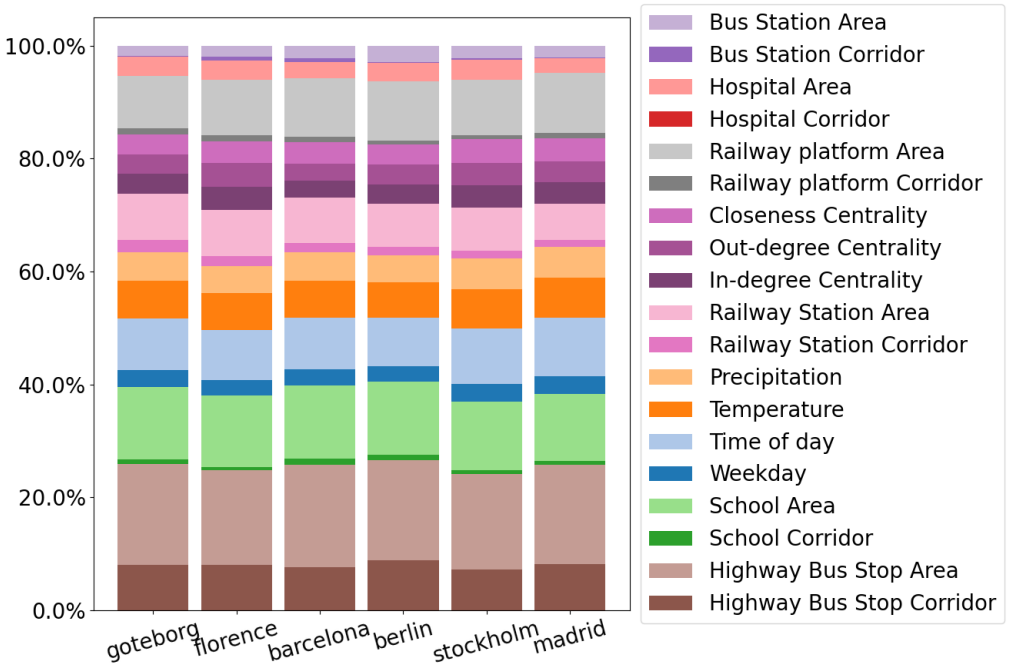


Figure 5.18: A stacked bar chart showing the importances of every feature for averaged samples of 10 configurations for each test city. One sample contains 10 configurations of city pairs used for training the model. Each configuration will have a feature importance list. Since there are 10 configurations for each test city there will be 10 such lists in total so these are averaged in order to get the above feature importance plot.

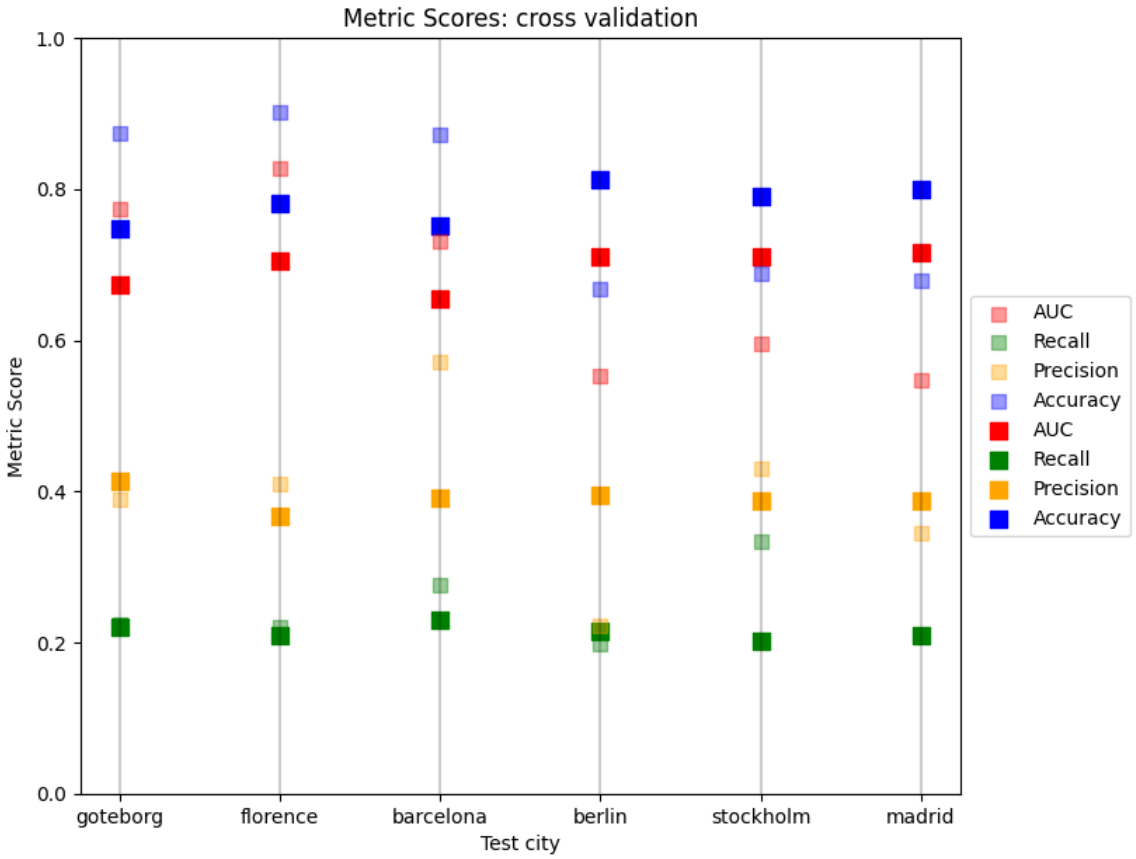
As observed in the figure, the feature importances are similar across all averages of samples. This is to be expected since for each set of samples there are various combinations of training cities that can be similar or the same across sample sets. For example, both test cities Gothenburg and Florence may share the combination {Stockholm, Madrid}. As the feature importances are averaged for each set of samples, this similarity is reflected in that all test cities share the same order of feature importances. Once again, *Highway Bus Stop* is the most important feature. This is followed by *School* and then *Time of Day & Weekday*. The rest of the features contribute roughly the same amount, except for the last features *Hospital* and *Bus Station*, accounting for a few percentages each.

5.3.3.2 Summary statistics

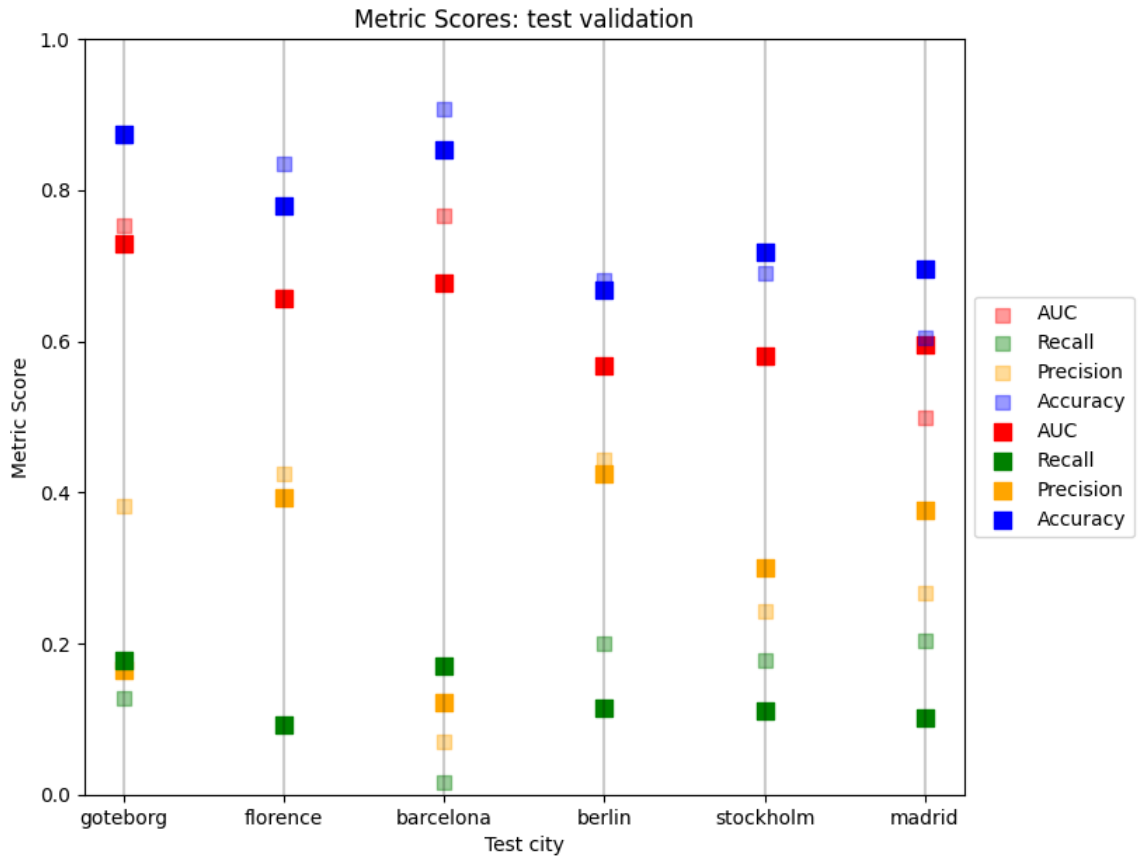
The metric scores for each test city in the sampled configurations are presented below. Cross Validation results are shown in Figure 5.19a and Validation results are shown in Figure 5.19b.

Figure 5.19: Averaged metric scores of sampled configurations compared to Scenario I. The scores from Scenario I have decreased size and are transparent in order to distinguish them from Scenario II. The upper plot corresponds to the scores of the Cross Validation case and the lower plot corresponds to the scores of the Validation case. The test city for each set of samples is shown on the x-axis.

(a) Cross Validation



(b) Validation i.e test cities



Despite the mix of non-capitals and capitals there are no major changes in the scores. The accuracy and AUC remains around the same scores as before for both the cross validation and test validation case. Looking at the precision and recall values these fluctuated a bit between Scenario I and Scenario II which can be observed in both figures. Generally, the scores are worse when the capitals are used as test cities compared to when non-capitals are used, this can be observed in the validation figure. In the cross validation figure all metrics are more stable. Since for each test city in Scenario II there are 10 models these can be similar or even the same between different cities. One such example could be the test cities Gothenburg and Florence that both might contain the configuration {Barcelona, Berlin} since all configurations are randomly sampled to acquire the 10 samples for each test city. This is a reason for the similar scores between the test cities in the cross validation plot.

5.3.4 Scenario III

In this section the results from Scenario III are presented for each metric. The results are presented using tables where each column corresponds to the number of

used training cities and the rows correspond to the test cities that were used for evaluation. The color coded values for the 5 training cities show whether the score increased or decreased compared to 2 cities.

Accuracy

Test City	2 cities	3 cities	4 cities	5 cities
Gothenburg	0.87	0.88	0.87	0.89
Florence	0.78	0.79	0.80	0.77
Barcelona	0.85	0.84	0.83	0.84
Berlin	0.67	0.69	0.68	0.76
Stockholm	0.72	0.73	0.72	0.79
Madrid	0.70	0.69	0.71	0.65

Looking at the accuracy metric it improved for 3 cities but also decreased for 3 cities. Overall the increases are larger than the decreases but not by much comparing 5 cities to 2 cities.

AUC

Test City	2 cities	3 cities	4 cities	5 cities
Gothenburg	0.73	0.74	0.75	0.80
Florence	0.66	0.67	0.64	0.72
Barcelona	0.68	0.65	0.66	0.63
Berlin	0.57	0.60	0.60	0.70
Stockholm	0.58	0.61	0.60	0.68
Madrid	0.60	0.59	0.58	0.58

Area under the curve shows an improvement for 4 out of 6 cities. The improvements are more prominent here compared to the accuracy and the improvements are larger compared to the decreases.

Precision

Test City	2 cities	3 cities	4 cities	5 cities
Gothenburg	0.17	0.22	0.25	0.27
Florence	0.39	0.36	0.40	0.57
Barcelona	0.12	0.11	0.12	0.20
Berlin	0.43	0.41	0.40	0.36
Stockholm	0.30	0.34	0.30	0.41
Madrid	0.38	0.34	0.35	0.37

Precision also improves in 4 out 6 cases and for some cities by quite a lot. Florence increased with 18 percentage points which is really good. Overall, the increases are more than the decreases.

Recall

Test City	2 cities	3 cities	4 cities	5 cities
Gothenburg	0.18	0.16	0.23	0.17
Florence	0.09	0.08	0.06	0.08
Barcelona	0.17	0.16	0.29	0.16
Berlin	0.12	0.09	0.10	0.07
Stockholm	0.11	0.10	0.11	0.26
Madrid	0.10	0.07	0.07	0.03

Recall was the only metric that decreased for the majority of the cities when comparing 5 cities to 2 cities. The only case where it increased was for Stockholm with 15 percentage points.

6
Conclusion

6.1 Discussion

In this section we will discuss the forecasting and classification results. The aim is to answer the questions posed in the thesis goal:

- Forecasting
    1. Are there any benefits to using more advanced models when forecasting traffic speeds?
    2. How will weather affect our predictions?
- Classification
    1. What features have the best predictive capabilities?
    2. How well does generalization of the models between cities really work? Does it only work when the cities share a lot of features such as structure, culture and technology?
    3. Will training on more cities improve the performance of our models on new data? If so, how much?

6.1.1 Forecasting

By the results in Section 5.2 there doesn't seem to be any strong evidence that forecasting performs better for the MLP model compared to the more simple AR model. There could be a number of reasons why this is the case. One might argue that the size of the training data was too small for the models to perform better. Furthermore, an increase in the number of time steps in the input vector in the MLP case, might improve the performance further compared to the other models. A more thorough investigation, including more features, is needed in order to conclude if more advanced models really improve the forecasting ability.

In this paper, no improvement was seen when including the features *precipitation* and *temperature* in the MLP model. Since the same weather data is provided for every road in a city at any given time, it could explain its incapability to provide useful information to the model. Another reason could be, once again, the size of the training data. Yearly training data might be needed for the model to make more informative decisions on how the different seasons might affect the predictions.

6.1.2 Classification

This section will focus on the discussion of the classification results. These include a discussion of the metric score plots, the best features and how well the models generalize to the test city data.
To further analyze the classification results, lets first review the metric formulas one more time:

$$Accuracy = All\ correctly\ classified\ data\ /\ All\ data$$

ROC-curve = TPR plotted on the y-axis against the FPR on the x-axis.

AUC = The area below the ROC-curve.

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$

Metric Scores

Generally, the metric plots 5.8 and 5.13 show a very high accuracy for both the Logistic Regression Classifier and Random Forest Classifier. This is to be expected and is not a good indication of the actual ability of the classifier to classify delays. Since the data is imbalanced (usually 1:10 ratio in favor of non-delays) the classifier can have an accuracy of 90% but might only predict the majority class. Looking at the *Precision* and *Recall* we can see *Precision* being larger than *Recall* for the most part. This indicates that the amount of FN is larger than FP. This means that the models are worse at identifying positives (delays) as opposed to negatives (non-delays). This is understandable, since our data is very imbalanced with far more negatives compared to positives. It is however unfortunate since we want to build models that are good at predicting delays i.e positive classes. One potential way to mitigate the effect of the imbalance is to introduce an upsampling technique in order to equalize the amount of positive and negative classes in the training samples.

That being said, looking at the AUC score in Figure 5.8b, we cannot say from our results that the Logistic Regression Classifier is better than a random classifier. Looking at the AUC score in Figure 5.13b we are able to produce Random Forest classifiers that have a higher AUC score than 0.5 in general. The TPR is higher compared to the FPR, meaning that these models correctly classifies more positives than they incorrectly classify positives, which is encouraging. This also shows that our Random Forest classifiers are better than a random classifier.

6.1.2.1 Comparing the classification models

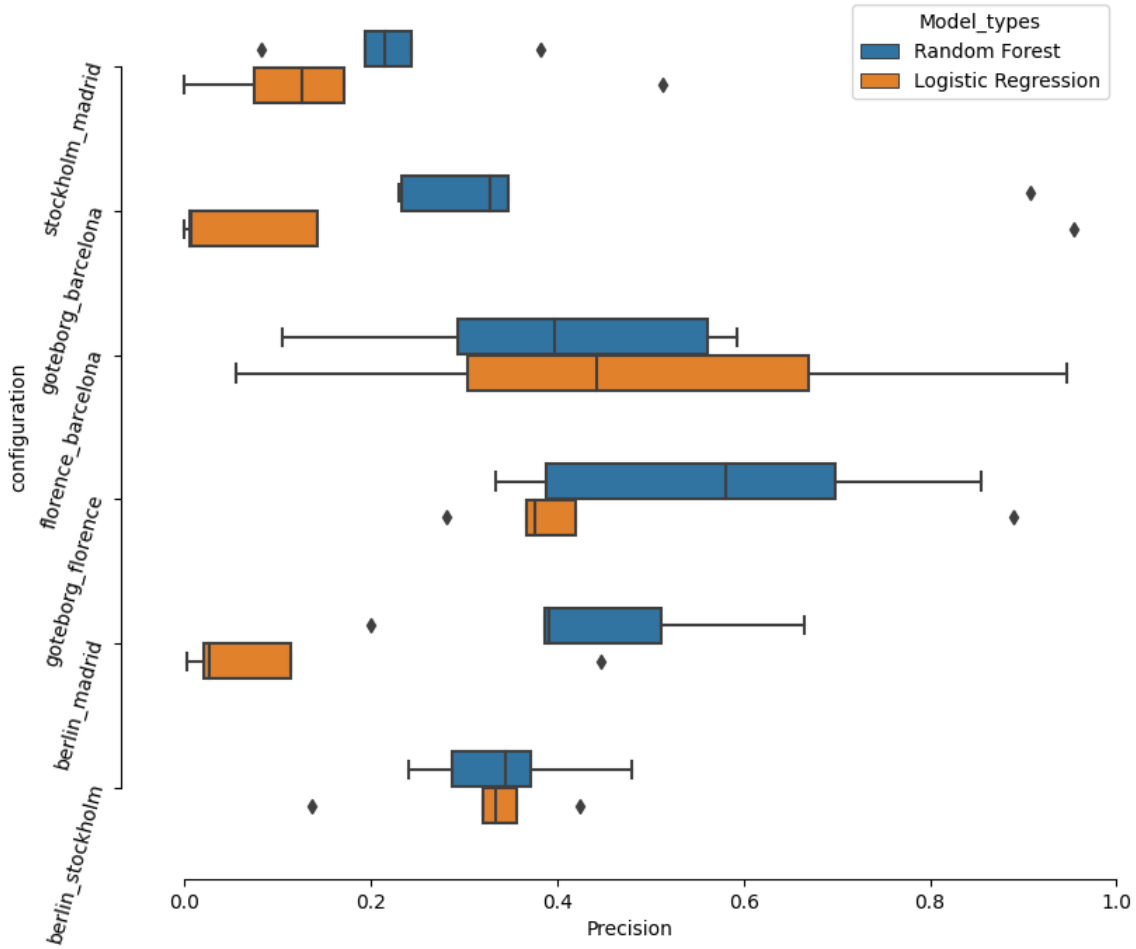
In this section we will discuss the differences between the classification models and how to choose the best model for classifying delays. In this paper we compared a Naive Majority Classifier, a Logistic Regression Classifier and a Random Forest

Classifier. To measure the applicability of each model to a real-world scenario we didn't just use accuracy as a measure of performance, but also ROC AUC, Precision and Recall. Together, these provide a much better picture of the actual performance of a classifier. In a real-world scenario the importance of classifying the delays correctly is of great importance since that can have potentially large consequences depending on how such a system might be used. It is therefore essential to reduce False Negatives as much as possible. In order to measure the model's ability to do this we use *Precision* and *Recall*. As mentioned before, Recall is the proportion of true positives that a model managed to classify. The Precision is the proportion of correctly classified positives. Since it is of great importance to know if a delay prediction is correct, both of these have been taken into account. An optimal model would have a high Precision and Recall.

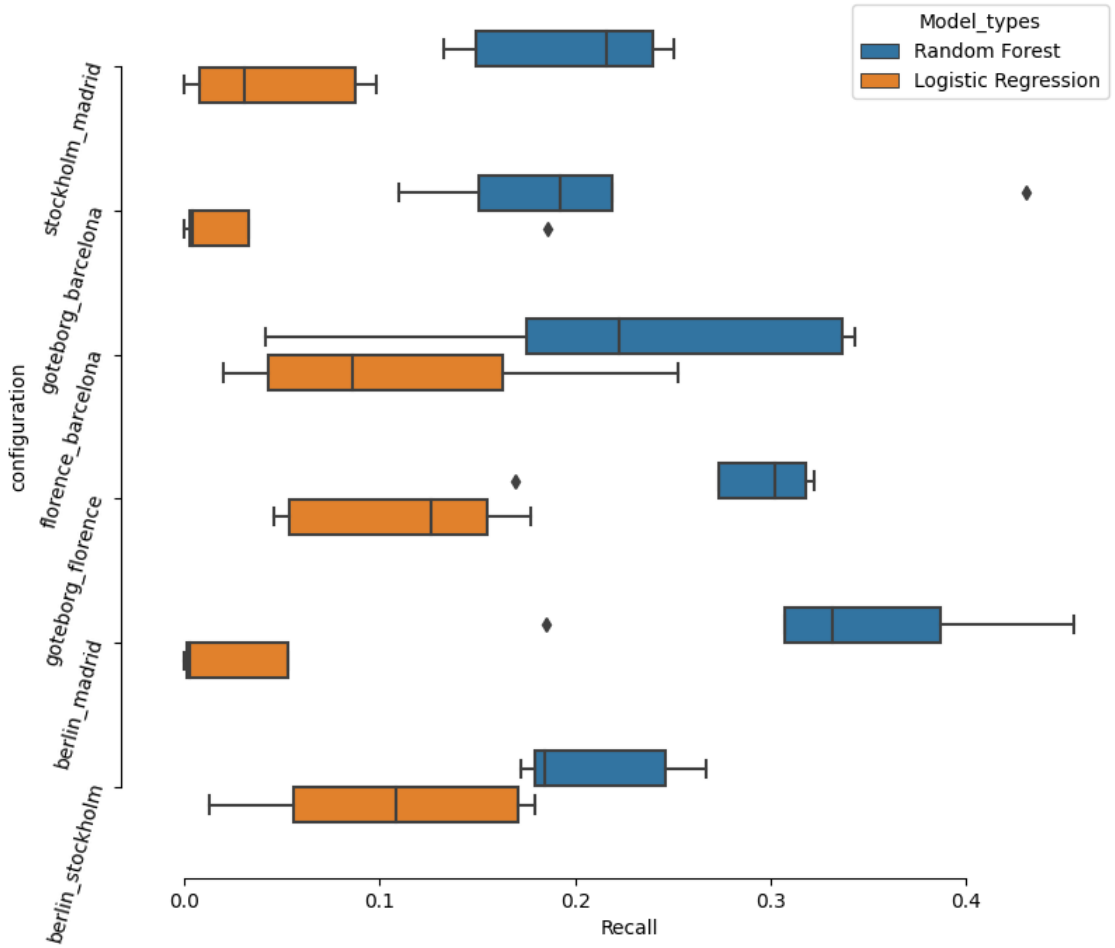
The Naive Majority Classifier had a Precision and Recall of 0 and is therefore not a good model to use in practice. The Logistic Regression Classifier and Random Forest Classifier both had a precision and recall above 0 except for one occasion for the Logistic Regression Classifier. This was when Berlin and Madrid were the cities included in the training configuration. The mean AUC value averaged over the 6 different configurations of Scenario I is 0.56 for Logistic Regression and 0.64 for the Random Forest Classifier rounded to two decimal places. This translates to roughly an increase of eight percentage points of AUC score for the Random Forest Classifier. This, paired with the overall average higher standard deviation of the Logistic Regression Classifier makes the Random Forest Classifier appear better. The differences in Precision and Recall between the Logistic Regression Classifier and Random Forest Classifier are significant. In order to confirm this, one box-plot containing the Precision values for each configuration and fold is presented in Figure 6.1a and one box-plot containing the Recall values is presented in 6.1b.

Figure 6.1: The figures are two box plots illustrating the difference in performance between Random Forest and the Logistic Regression for the cross validation case. The top figure shows a comparison in precision while the bottom figure shows the comparison in Recall.

(a) Precision Box plots



(b) Recall Box plots



Looking at 6.1a we see that the precision for Random Forest almost always outperforms the Logistic Regression. The only difference is in the configuration: *Train set: {Florence, Barcelona} Test set: {Gothenburg}* and even then the difference is not very large. In 6.1b we observe the Recall, it is shown that the Random Forest outperforms the Logistic Regression for every configuration. For both 6.1a and 6.1b the median of the box-plots is always higher with the exception of the configuration previously mentioned. It is now clear that for our models, the Random Forest Classifier is the obvious choice.

Since the goal is to create a model that is good at classifying delays i.e. a model with high Recall, analyzing the results presented in the Precision-Recall curves are necessary. This can be illustrated by taking the example configuration with train cities {Gothenburg, Florence} and test city {Barcelona}. If the requirement is to cover 80% of the delays by the model the Recall needs to be 0.8. When this con-

figuration achieves a recall of 0.8 we can observe the corresponding Precision value is roughly below 0.2 in the Precision-Recall Figure 5.16 to the right (the test city). Since we allow for more false positives this is an okay performance of the model. In our opinion all of the scores should be higher in order to apply the model in practice. In this example configuration the scores were on an okay level but looking at the other configuration used in the results, Figure 5.17, we can observe considerably worse precision when choosing the same Recall value of 0.8. Because of this we think that the classification results of this thesis can advantageously be used as a good base for further research. By selecting a subset of the most important features found in this thesis and using the same method of building the input vector and evaluating the results, one can further improve the model result.

Selecting the best features

Going by the feature importance bar chart from Scenario II, the most important features for predictability are *Highway Bus Stops* and *Schools*, as well as other forms of public transportation. This is understandable since many people commute by public transportation which usually is more available in the central parts of the city. Moreover, places like schools usually have restrictions in terms of traffic speed which might lend themselves towards a higher probability of congestion. Since the features hospital and bus station didn't provide much useful information according to the generated feature importances, these could be removed in favor of other features in the future. One potential flaw of the Gini Impurity as the method for assessing the feature importances is that features with high cardinality have a tendency to get a higher importance. A high cardinality refers to the uniqueness of values. This essentially means that if a feature has a lot of unique values it might rank higher in the feature importance list[28].

In order to further improve upon the classifiers it is essential to explore different feature sets and assess their importance. Some examples of such features are discussed in Section 6.2.2.

Generalizability

The results from the different Scenarios show that it is possible to train a model on some cities which is generalizable to other cities. The degree of success varies between the different configurations of train cities and test cities but overall the models perform better than random classifiers on the test city (which is what measures the actual real-world generalizability). In some cases the performance of the model was barely able to perform better than a random classifier as can be seen in Figure 5.15. This can probably be attributed to fundamental differences between the training cities and the test city, which in this case was Berlin. Looking at other configurations where Berlin is the test city we can observe the same pattern of worse performance on the validation set i.e Berlin compared to the Cross Validation folds (where Berlin is not present). This is supported by other configurations where Berlin acted as the test city. For examples see Figure A.36, Figure A.37 and Figure A.39. This is interesting because this might mean that Berlin has some fundamental dif-

ferences in how the city is structured that contributes to the performance decrease when the city is included in models. Looking at the distribution of centrality in Figure 4.3 we can also see that Berlin has a right-skewed distribution whereas the other cities have considerably less skewness. This might invalidate the sampling technique since the right-skewed distribution of Berlin doesn't reflect that of the others. Since Berlin contains more roads with less centrality the model might need training on more roads that are in the higher range in order to more accurately reflect the other cities. Another way of looking at it is that Berlin doesn't have a lower density of roads but rather that the center point of the city is not clearly defined. Other metrics could be introduced in order to sample the roads on more than just the centrality metric. One final suggestion is to introduce some adjustments for the skewness.

Increasing the number of training cities

Going by the results from Scenario III we generally see an improvement in the metric scores when increasing the number of training cities with recall being the only exception. One reason for this could be that it has to do with the random elements involved during training of the models and more specifically the cross validation splits. We believe that recall can be improved by adjusting the classifiers probabilistic threshold which could vary between the number of training cities used. We also believe that the reason for a decrease in the score for some cities could be due to some cities not performing well together as we have stated previously. Since the cities that had a skewed normal distribution performed worse in some cases, one example being Berlin, it would be interesting to choose cities that share a similar closeness distribution.

6.2 Future considerations

In this section we explore in what ways our work can be further developed.

6.2.1 Refined feature selection

The features selected for our models were selected roughly by intuition and by influence from the study conducted in Beijing (see Section 4.4.1). It is reasonable to expect that roads that have close proximity to schools or junctions such as bus stops where overcrowding occurs are more prone to traffic delays. Despite this rather rough selection of features, the models do improve over the Naive Classifier. It is therefore interesting to explore the improvements that can be made if the feature selection were to be more refined. To refine the feature selection one need to have a clear understanding of where people are gathering at specific times. One way of doing this is to gather location data on the movement of people in order to determine where overcrowding occurs.

6.2.2 Static and non-static features

In this work we only accounted for mostly different kinds of static features that are more or less consistent over longer periods of time. It would also be interesting to analyze features that are of non-static nature, which may lead to non-reoccurring traffic patterns. Examples of such features are road maintenance and accidents that could negatively affect the traffic flow. By expectation, the current classification models will perform poorly in these circumstances.

6.2.3 Climate change and extreme weather

Because of the ongoing climate change and the extreme weather that may follow as a consequence, it would be interesting to see how extreme weather conditions affect the traffic. This is however very hard to do due to the few instances of extreme weather across European cities. Furthermore what classifies as extreme weather? People residing in different geographical areas may have different opinions on that definition based on the weather that they are accustomed to.

Let us, for the sake of simplicity, classify extreme weather as weather that affect infrastructure. For example, the weather might prevent people from driving through roads that have been destroyed or are now deemed too dangerous to use due to the current weather conditions. In order for the models to take extreme weather into account the models must be trained in cities where extreme weather is more prevalent.

6.2.4 Final words

As we move into the future where more people will likely live in great urban areas, the planning and flow of a city is crucial for reducing congestion and the pollution that follows. In this thesis we investigated different ways of working with traffic speed data in the form of forecasting and classification. In our thesis we didn't see an improvement when using more advanced models for forecasting traffic speed. We didn't see any improvements when adding weather variables either but rather a decrease in performance. In our case the RMSE scores were averaged over all cities and roads which makes it hard to know if weather variables had a positive effect on prediction in some cities. An interesting follow-up on this would be to evaluate each city on its own and see if some cities actually benefit from the use of weather variables or not. A major focus was put on the classification of traffic delays which could be very interesting for determining when and where congestion occur in different cities. This could be used as more knowledge when planning new cities or improving the cities that we have in order to reduce congestion and CO_2 emissions. In this paper we presented three different Scenarios in order to determine if it was even possible to classify traffic delays in cities where no traffic data is available. Our results suggest that this is possible but more work is needed in order to improve the performance of the models for practical applications. We also saw that increasing the number of training cities overall improved the metric scores. The most important features in our case proved to be the area features

bus station and school. Suggestions of future work include refined sampling and feature selection, including non-static features and looking at larger sets of cities. All things considered, the potential for future improvements are promising and it will be interesting to see how models built using traffic speed data as well as other data can be used to improve the infrastructure of the future.

Bibliography

[1] Gardner, M. W., Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment, 32(14-15), 2627-2636.

[2] Saito, T., Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PloS one, 10(3), e0118432.

[3] Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[4] Wahde, M. (2008). Biologically inspired optimization methods: an introduction. WIT press, 17-21

[5] Ketkar, N. (2017). Stochastic gradient descent. In Deep learning with Python (pp. 113-132). Apress, Berkeley, CA.

[6] Deo, R., Samui, P., Roy, S. S. (Eds.). (2020). ch. 9. Predictive Modelling for Energy Management and Power Systems Engineering. Elsevier.

[7] Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874.

[8] PennState Eberly College of Science. 14.1- Autoregressive Models. `https://online.stat.psu.edu/stat501/lesson/14/14.1`. (accessed on < 2021-05-28 >).

[9] Subasi, C., (2019, Mars 4). Logistic Regression Classifier. towards data science. `https://towardsdatascience.com/logistic-regression-classifier-8583e0c3cf9`. (accessed on < 2021-05-11 >.

[10] Norton, E. C., Dowd, B. E. (2018). Log odds and the interpretation of logit models. Health services research, 53(2), 859-878.

[11] Myung, I. J. (2003). Tutorial on maximum likelihood estimation. Journal of mathematical Psychology, 47(1), 90-100.

[12] Barth, M., Boriboonsomsin, K. (2008). Real-world carbon dioxide impacts of traffic congestion. Transportation Research Record, 2058(1), 163-171.

[13] Arnott, R., Small, K. (1994). The economics of traffic congestion. American scientist, 82(5), 446-455.

[14] Colon, C., Hallegatte, S., Rozenberg, J. (2021). Criticality analysis of a country's transport network via an agent-based supply chain model. Nature Sustainability, 4(3), 209-215.

[15] Koetse, M. J., Rietveld, P. (2009). The impact of climate change and weather on transport: An overview of empirical findings. Transportation Research Part D: Transport and Environment, 14(3), 205-221.

[16] Kong, X., Xu, Z., Shen, G., Wang, J., Yang, Q., Zhang, B. (2016). Urban traffic congestion estimation and prediction based on floating car trajectory data. Future Generation Computer Systems, 61, 97-107.

[17] Polson, N. G., Sokolov, V. O. (2017). Deep learning for short-term traffic flow prediction. Transportation Research Part C: Emerging Technologies, 79, 1-17.

[18] Fouladgar, M., Parchami, M., Elmasri, R., Ghaderi, A. (2017, May). Scalable deep traffic flow neural networks for urban traffic congestion prediction. In 2017 International Joint Conference on Neural Networks (IJCNN) (pp. 2251-2258). IEEE.

[19] Verendel, V., Yeh, S. (2019). Measuring traffic in cities through a large-scale online platform. Journal of Big Data Analytics in Transportation, 1(2), 161-173.

[20] McCulloch, W. S., Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.

[21] Wikimedia. 2021. `https://commons.wikimedia.org/wiki/File:Threshold_roc.wikipedia_edit.svg` (accessed on < 2021-06-03 >)

[22] Dickey, D. A., Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. Journal of the American statistical association, 74(366a), 427-431.

[23] Görtler, J., Kehlbeck, R., Deussen, O. (2019). A visual exploration of gaussian processes. Distill, 4(4), e17.

[24] Louppe, G., Wehenkel, L., Sutera, A., Geurts, P. (2013). Understanding variable importances in forests of randomized trees. Advances in neural information processing systems 26.

[25] Scikit-learn: Machine Learning in Python. Pedregosa, F., Varoquaux, G., Gramfort, A. et al. 2011. Journal of Machine Learning Research, 12, 2825–2830. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.feature_importances_`. (accessed on < 2021-05-31 >).

[26] Ahmed, N. K., Atiya, A. F., Gayar, N. E., El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. Econometric Reviews, 29(5-6), 594-621.

[27] Xia, F., Zhang, W., Li, F., Yang, Y. (2008). Ranking with decision tree. Knowledge and information systems, 17(3), 381-395.

[28] Breiman, L., Friedman, J., Stone, C. J., Olshen, R. A. (1984). Classification and regression trees. CRC press.

[29] Rokach, L., Maimon, O. (2005). Top-down induction of decision trees classifiers-a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 35(4), 476-487.

[30] Melikov, P., Kho, J. A., Fighiera, V., Alhasoun, F., Audiffred, J., Mateos, J. L., González, M. C. (2021). Characterizing Urban Mobility Patterns: A Case Study of Mexico City. Urban Informatics, 153.

[31] Park, K. I., Park. (2018). Fundamentals of Probability and Stochastic Processes with Applications to Communications. Springer International Publishing.

[32] Wold, H. (1938). A study in the analysis of stationary time series (Doctoral dissertation, Almqvist Wiksell).

[33] Chow, A. H., Santacreu, A., Tsapakis, I., Tanasaranond, G., Cheng, T. (2014). Empirical assessment of urban traffic congestion. Journal of advanced transportation, 48(8), 1000-1016.

[34] Zhang, T., Sun, L., Yao, L., Rong, J. (2017). Impact analysis of land use on traffic congestion using real-time traffic and POI. Journal of Advanced Transportation, 2017.

[35] Keay, K., Simmonds, I. (2005). The association of rainfall and other weather variables with road traffic volume in Melbourne, Australia. Accident analysis prevention, 37(1), 109-124.

[36] Muñoz Sabater, J., (2019). ERA5-Land hourly data from 1981 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS). (Accessed on < 2021-04-19 >), 10.24381/cds.e2161bac Copernicus.

[37] Hyndman, R.J., Athanasopoulos, G. (2021). Forecasting: principles and practices 3rd edition. ch. 2.9. OTexts: Melbourne, Australia. `OTexts.com/fpp3`. (accessed on < 2021-05-06 >.

[38] Hyndman, R.J., Athanasopoulos, G. (2021). Forecasting: principles and practices 3rd edition. ch. 8.3. OTexts: Melbourne, Australia. `OTexts.com/fpp3`. (accessed on < 2021-05-06 >.

[39] Hyndman, R.J., Athanasopoulos, G. (2021). Forecasting: principles and practices 3rd edition. ch. 8.1. OTexts: Melbourne, Australia. `OTexts.com/fpp3`. (accessed on < 2021-05-06 >.

[40] Hyndman, R.J., Athanasopoulos, G. (2021). Forecasting: principles and practices 3rd edition. ch. 12.4. OTexts: Melbourne, Australia. `OTexts.com/fpp3`. (accessed on < 2021-05-07 >.

[41] Copernicus. (2021). Copernicus API Reference. `https://cds.climate.copernicus.eu/toolbox/doc/api.html`. Accessed on < 2021-05-06 >.

[42] HERE. (2021). HERE API Reference. `https://developer.here.com/`. Accessed on < 2021-05-05 >.

[43] Group Shuffle Split Cross Validation. (2021). SkLearn Reference Document. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ShuffleSplit.html`. Accessed on < 2021-05-24 >.

[44] Hastie, T., Tibshirani, R., Friedman, J. (2001). The elements of statistical learning, Data Mining, Inference, and Prediction (2nd ed.). Springer.

[45] Weisstein, Eric W. Statistical Correlation. MathWorld: A Wolfram Web Resource. `https://mathworld.wolfram.com/StatisticalCorrelation.html`. Accessed < 2021-05-06 >

[46] OpenStreetMap. (2017). `https://www.openstreetmap.org`

[47] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4), 303-314.

[48] Glorot, X., Bordes, A., Bengio, Y. (2011, June). Deep sparse rectifier neural networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics (pp. 315-323). JMLR Workshop and Conference Proceedings.

[49] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1985). Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

[50] Minsky, M., Papert, S. A. (2017). Perceptrons: An introduction to computational geometry. MIT press.
[51] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8), 2554-2558.
[52] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.
[53] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
[54] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107-116.
[55] Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE.
[56] Trevethan, R. (2017). Sensitivity, specificity, and predictive values: foundations, pliabilities, and pitfalls in research and practice. Frontiers in public health, 5, 307.
[57] Brown, C. D., Davis, H. T. (2006). Receiver operating characteristics curves and related decision measures: A tutorial. Chemometrics and Intelligent Laboratory Systems, 80(1), 24-38.
[58] Freeman, L. C. (1978). Centrality in social networks conceptual clarification. Social networks, 1(3), 215-239.
[59] Sabidussi, G. (1966). The centrality index of a graph. Psychometrika, 31(4), 581-603.

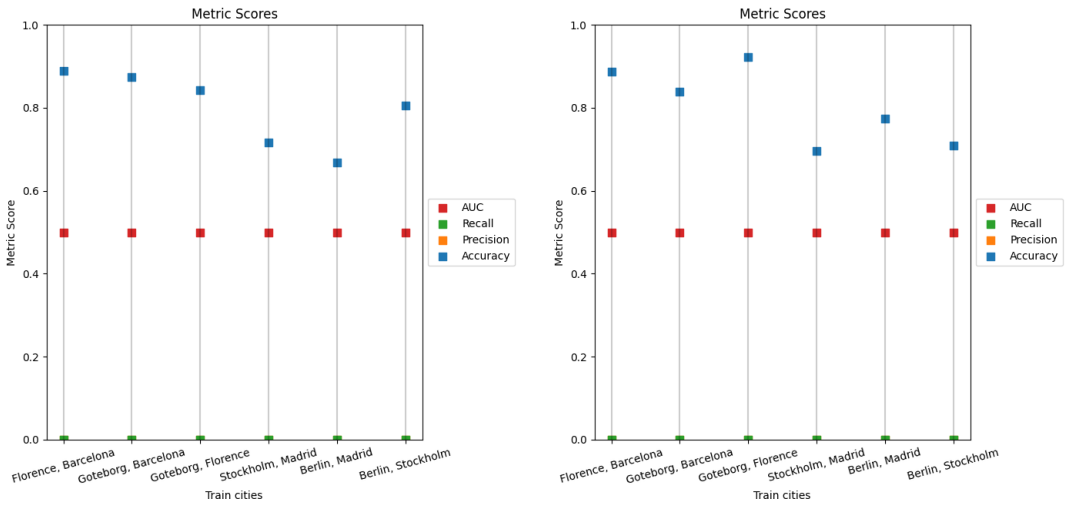
A

Appendix 1

A.1 Scenario I: Naive Majority Classifier

A.1.1 Metric Scores

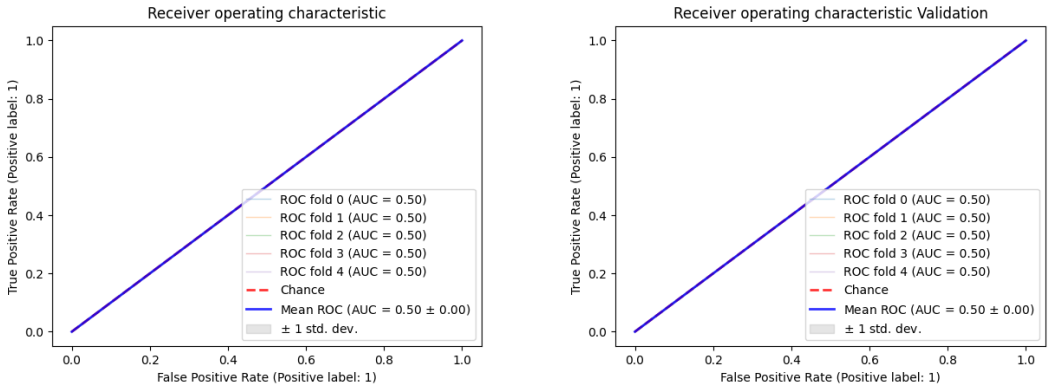
Figure A.1: Metric Scores for each configuration. The test cities (used for calculating the validation scores for the plot to the right) are Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid in that order.



A.1.2 ROC-curves

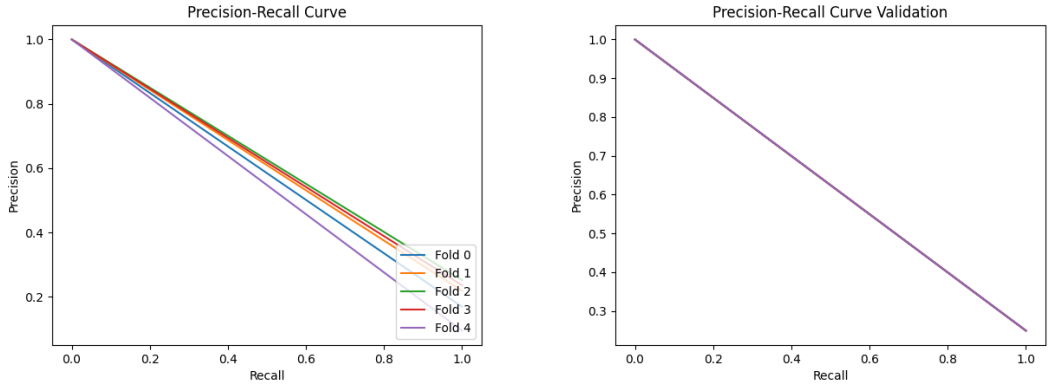
The Naive Classifier achieves no precision nor recall so just one example will be shown here since every plot is essentially the same.

Figure A.2: Train set: {Berlin, Madrid}. Test set: {Stockholm}.



A.1.3 Precision-recall-curves

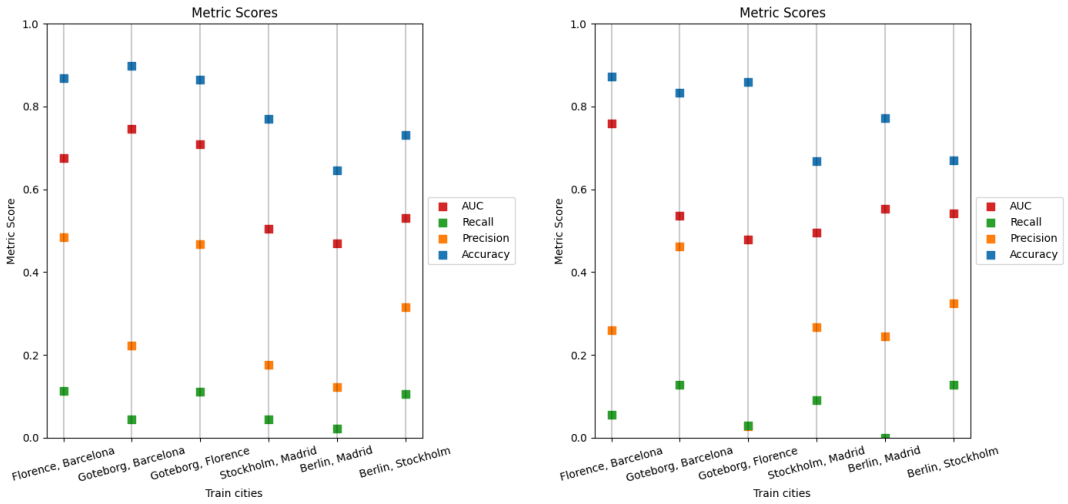
Figure A.3: Train set: {Berlin, Madrid}. Test set: {Stockholm}.



A.2 Scenario I: Logistic Regression Classifier

A.2.1 Metric Scores

Figure A.4: Metric Scores for each configuration. The test cities (used for calculating the validation scores for the plot to the right) are Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid in that order.



A.2.1.1 ROC-curves

Non-Capitals

Figure A.5: Train set: {Florence, Barcelona}. Test set: {Gothenburg}.

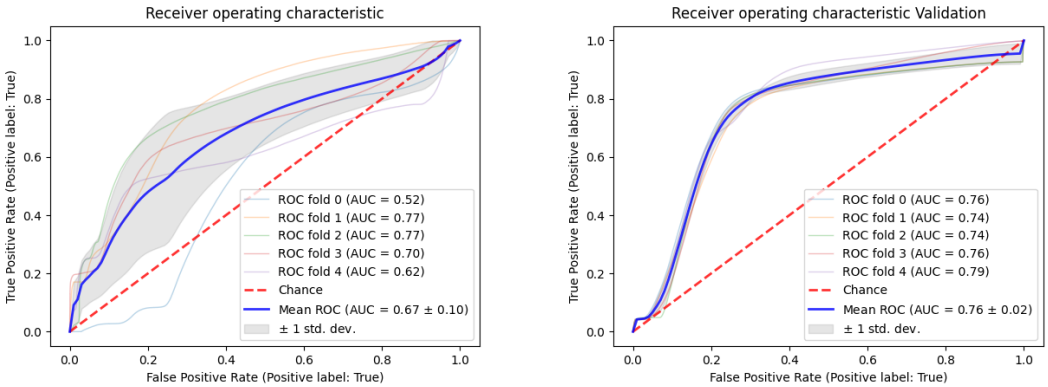


Figure A.6: Train set: {Gothenburg, Barcelona}. Test set: {Florence}.

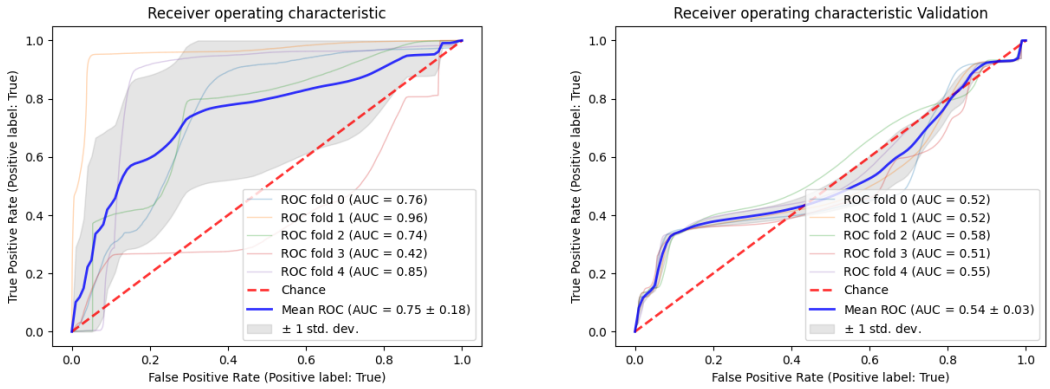
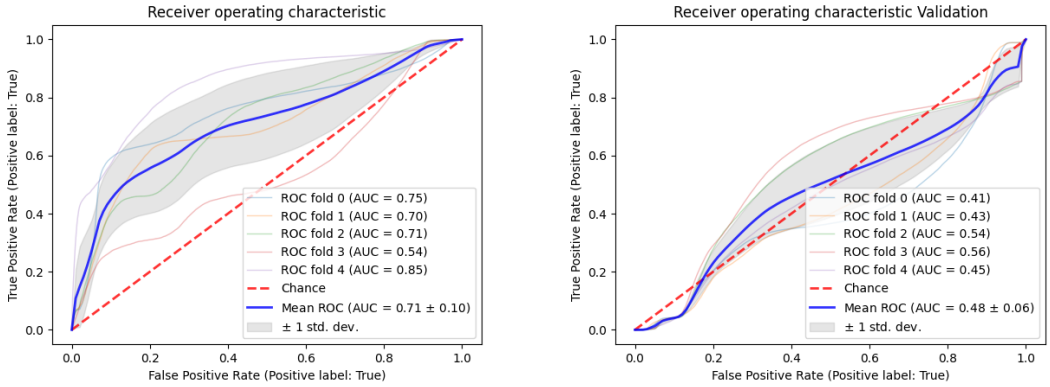


Figure A.7: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.



Capitals

Figure A.8: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

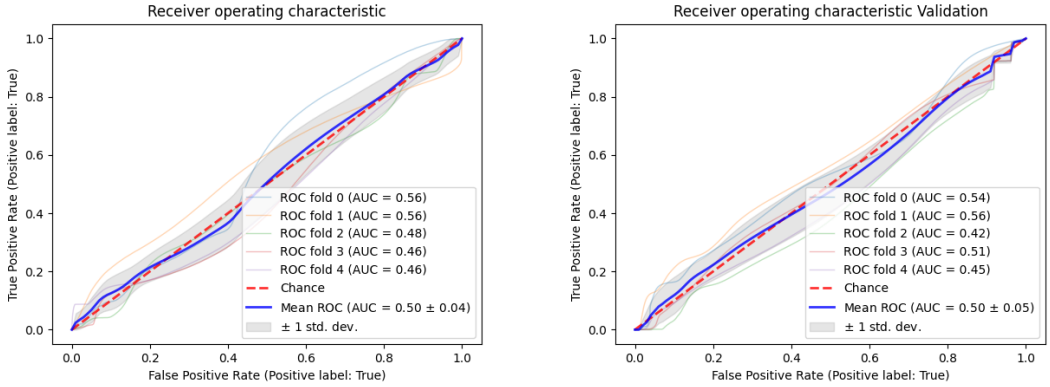


Figure A.9: Train set: {Berlin, Madrid}. Test set: {Stockholm}.

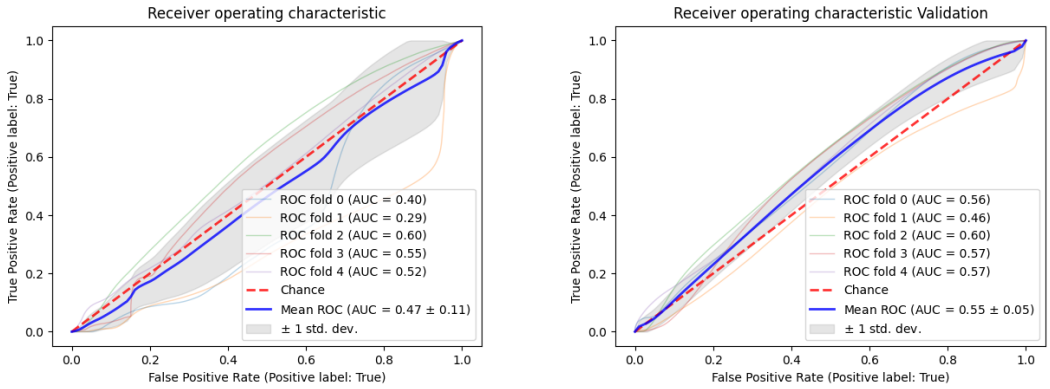
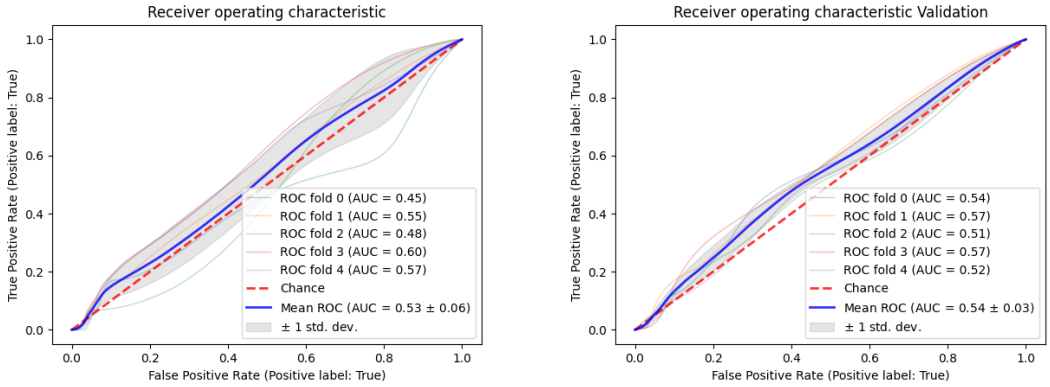


Figure A.10: Train set: {Berlin, Stockholm}. Test set: {Madrid}.



A.2.1.2 Precision-recall-curves

Non-Capitals

Figure A.11: Train set: {Florence, Barcelona}. Test set: {Gothenburg}.

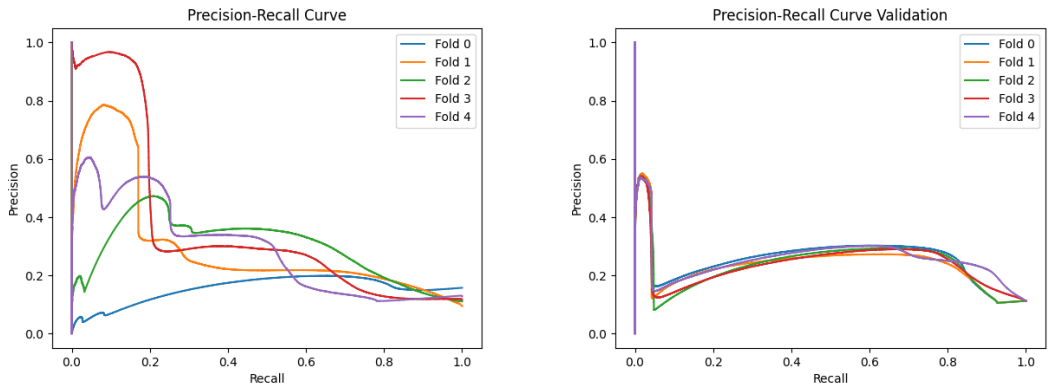


Figure A.12: Train set: {Gothenburg, Barcelona}. Test set: {Florence}.

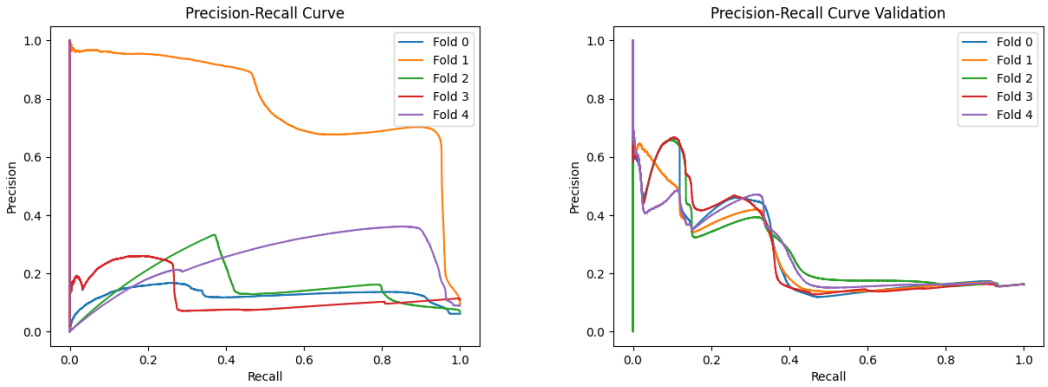
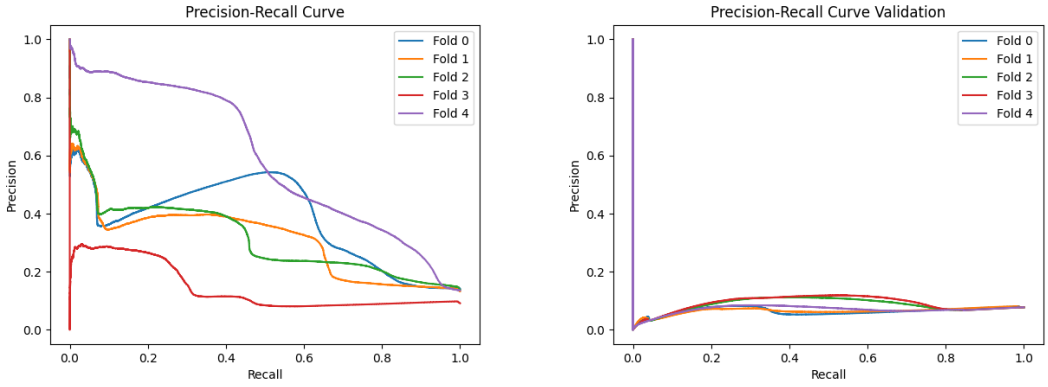


Figure A.13: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.



Capitals

Figure A.14: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

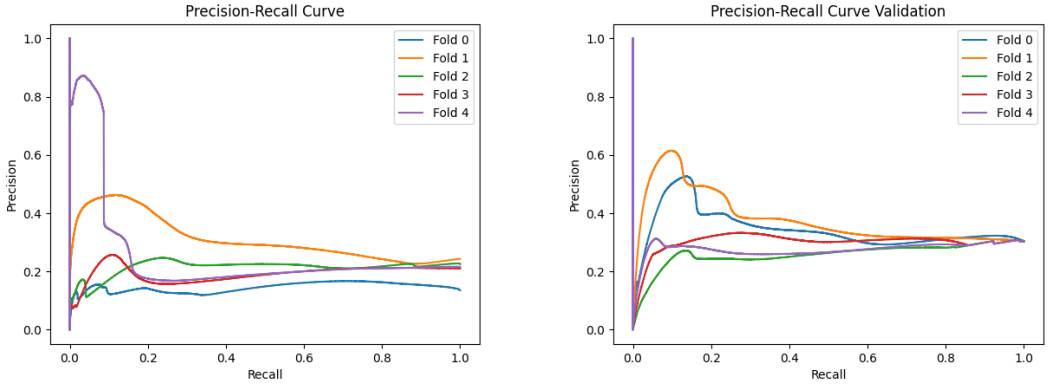


Figure A.15: Train set: {Berlin, Madrid}. Test set: {Stockholm}.

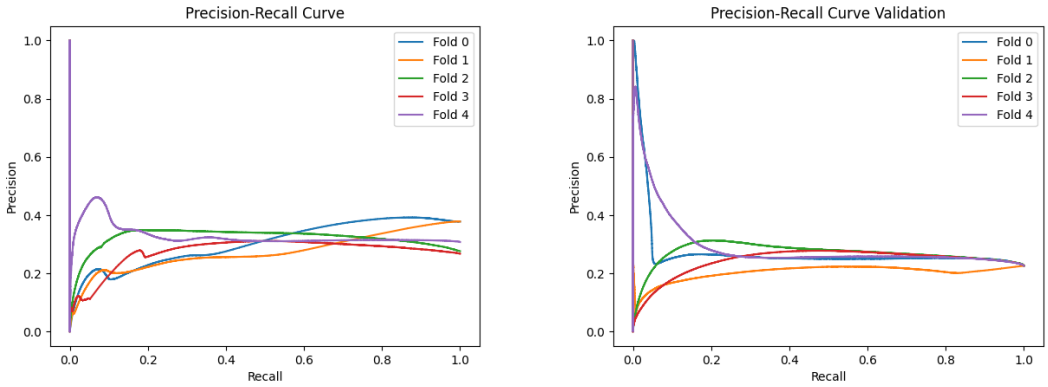
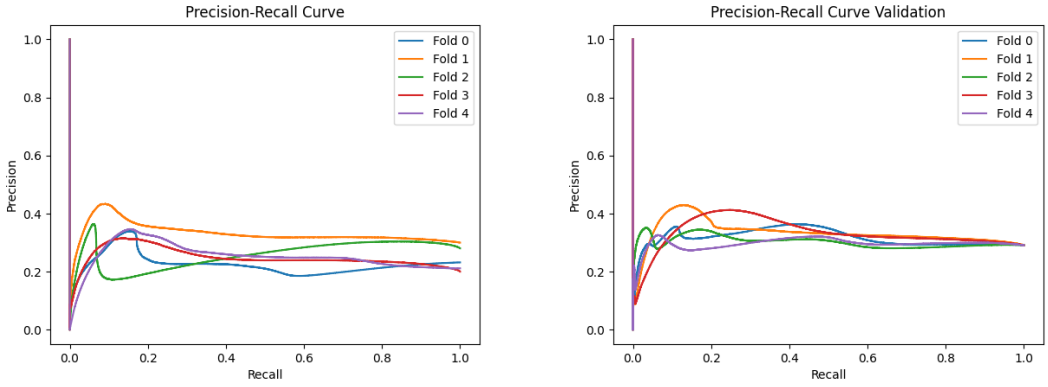


Figure A.16: Train set: {Berlin, Stockholm}. Test set: {Madrid}.

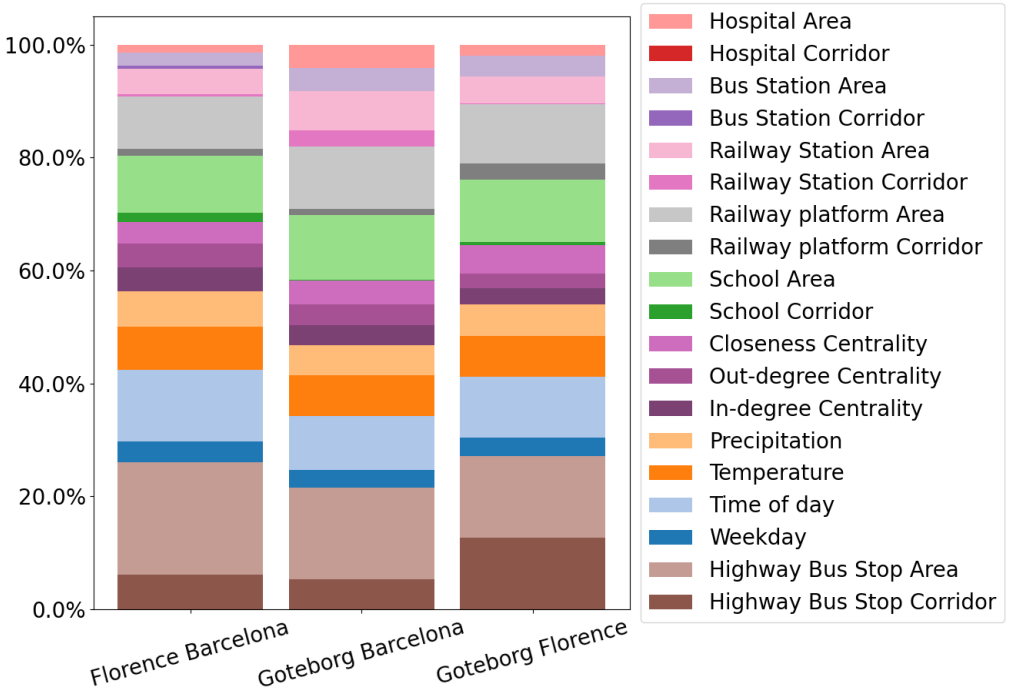


A.3 Scenario I: Random Forest Classifier

A.3.1 Feature Importance

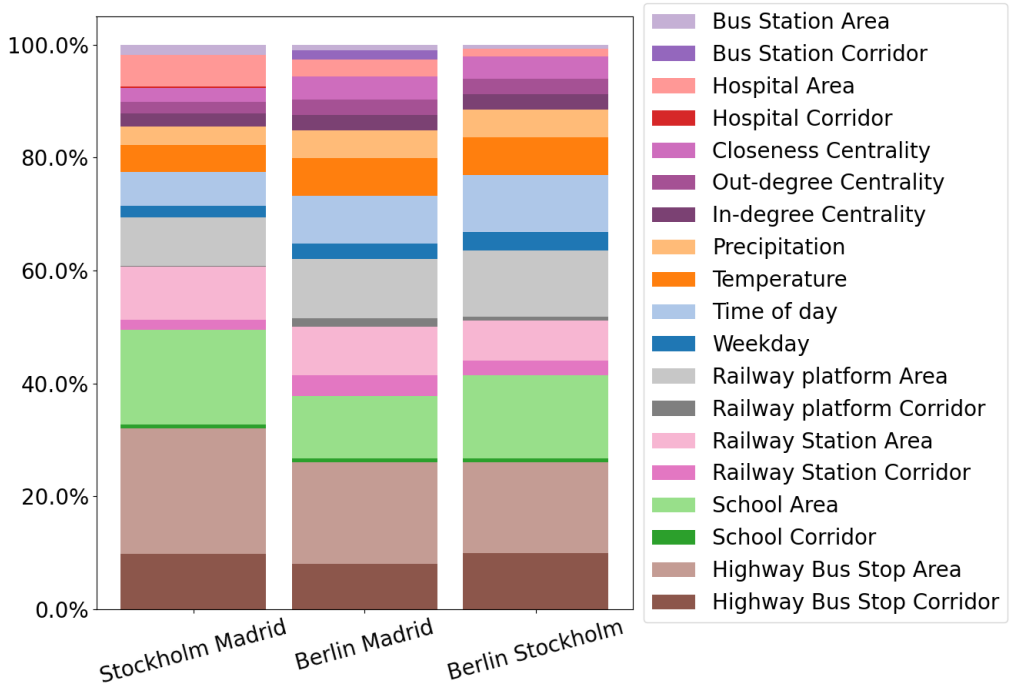
Non-Capitals

Figure A.17: Showing feature importances in non-capital-configurations.



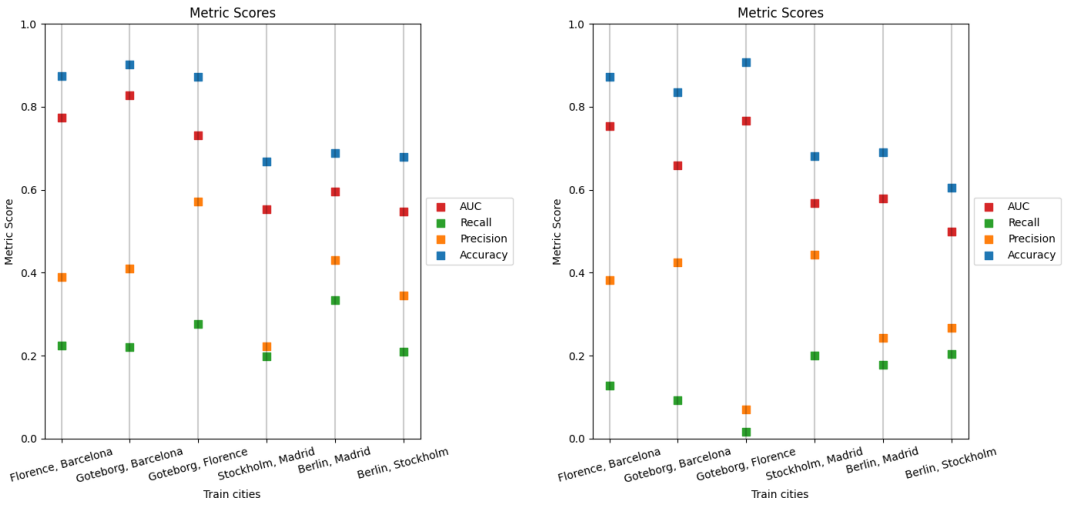
Capitals

Figure A.18: Showing feature importances in capital-configurations.



A.3.2 Metric Scores

Figure A.19: Metric Scores for each configuration. The test cities (used for calculating the validation scores for the plot to the right) are Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid in that order.



A.3.2.1 ROC-curves

Non-Capitals

Figure A.20: Train set: {Florence, Barcelona}. Test set: {Gothenburg}.

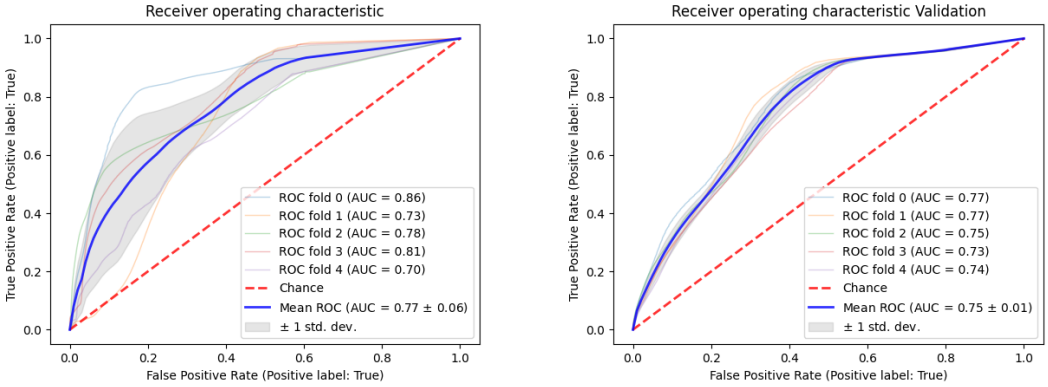


Figure A.21: Train set: {Gothenburg, Barcelona}. Test set: {Florence}.

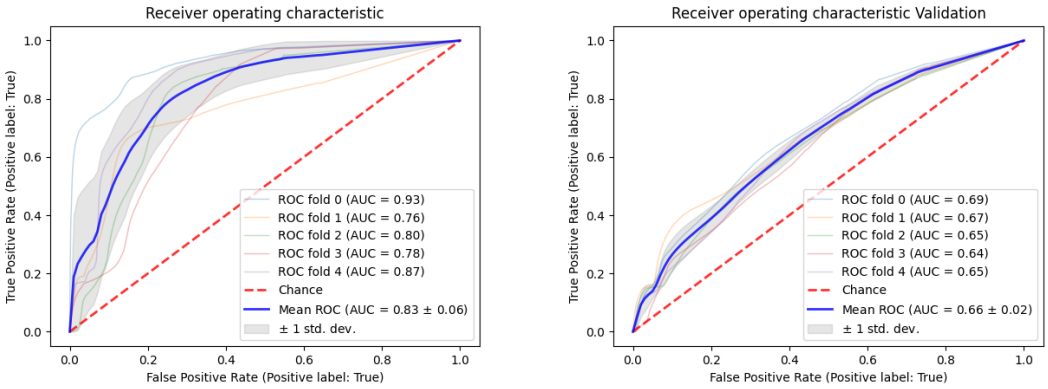
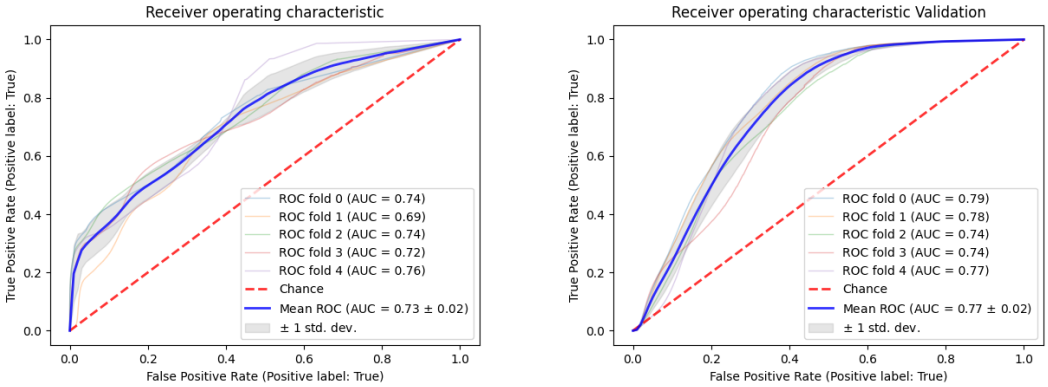


Figure A.22: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.



Capitals

X

Figure A.23: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

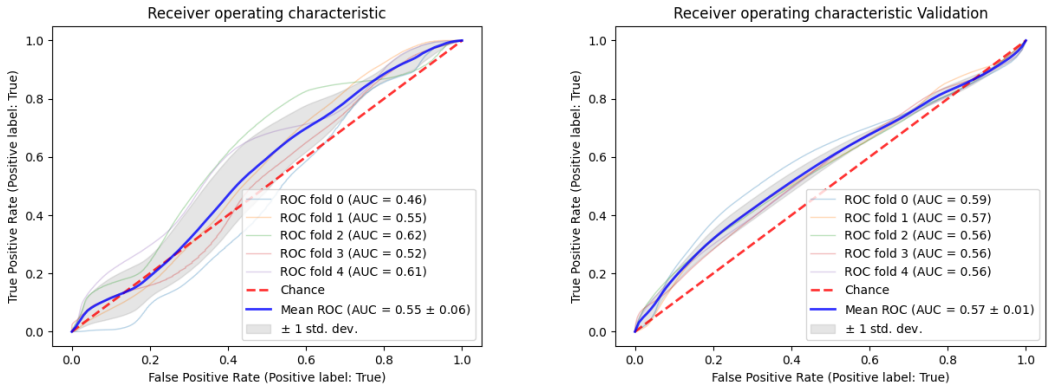


Figure A.24: Train set: {Berlin, Madrid}. Test set: {Stockholm}.

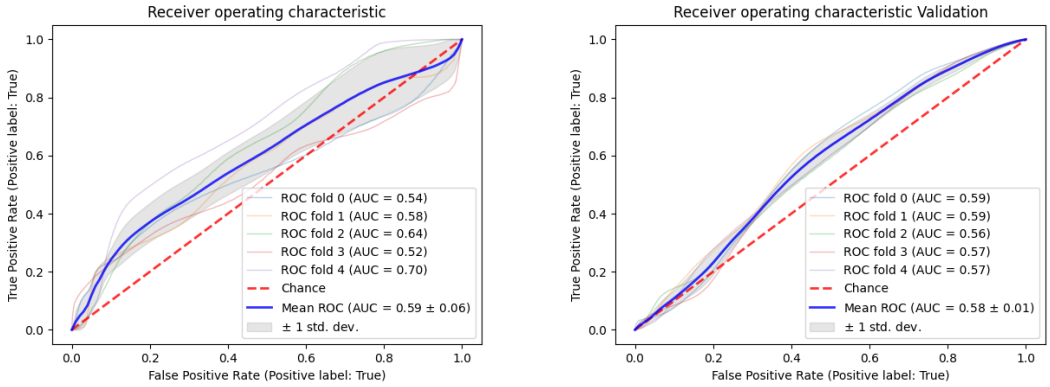
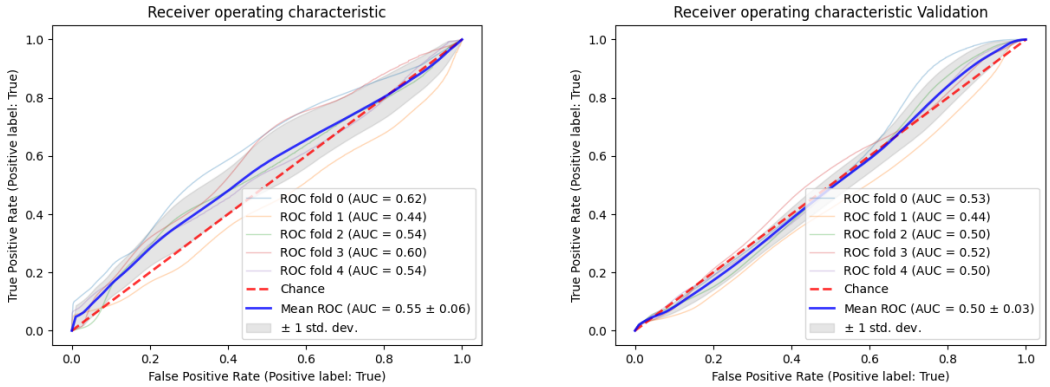


Figure A.25: Train set: {Berlin, Stockholm}. Test set: {Madrid}.



A.3.2.2 Precision-recall-curves

Non-Capitals

Figure A.26: Train set: {Florence, Barcelona}. Test set: {Gothenburg}.

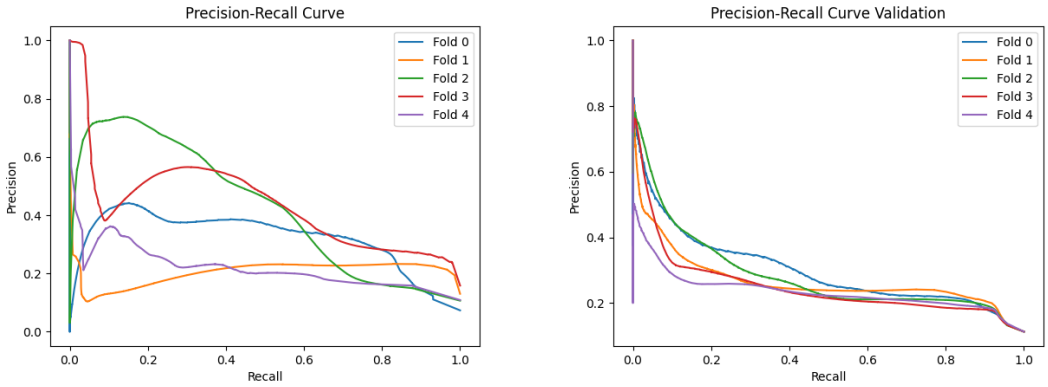


Figure A.27: Train set: {Gothenburg, Barcelona}. Test set: {Florence}.

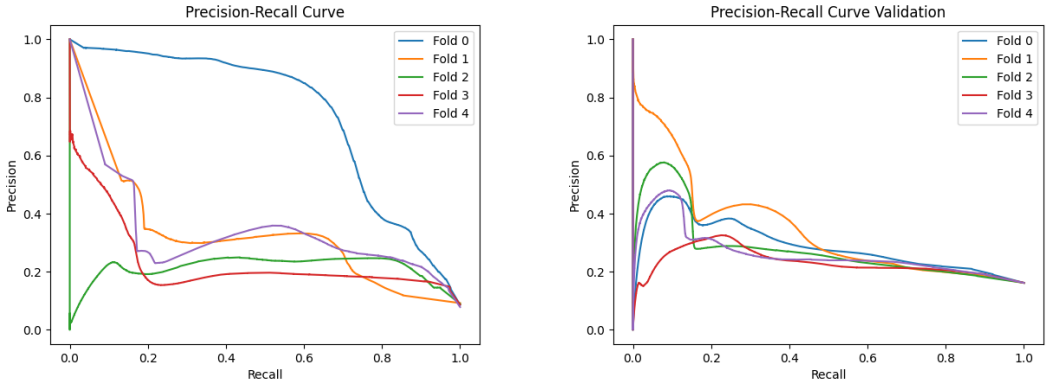
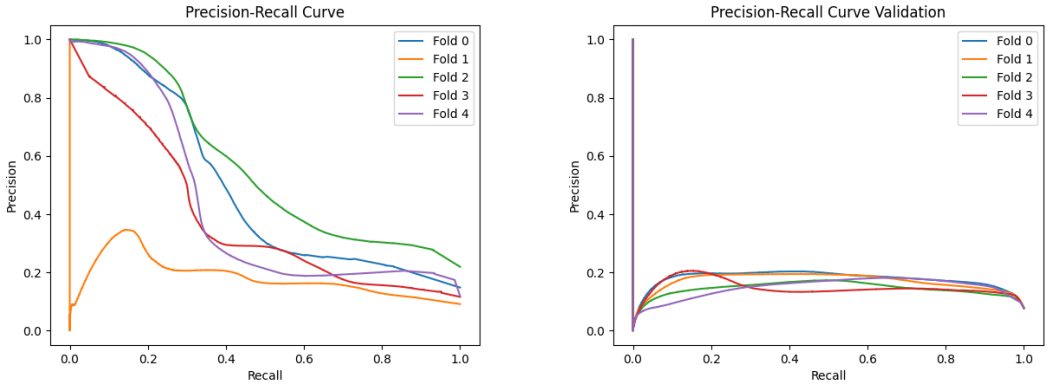


Figure A.28: Train set: {Gothenburg, Florence}. Test set: {Barcelona}.



Capitals

Figure A.29: Train set: {Stockholm, Madrid}. Test set: {Berlin}.

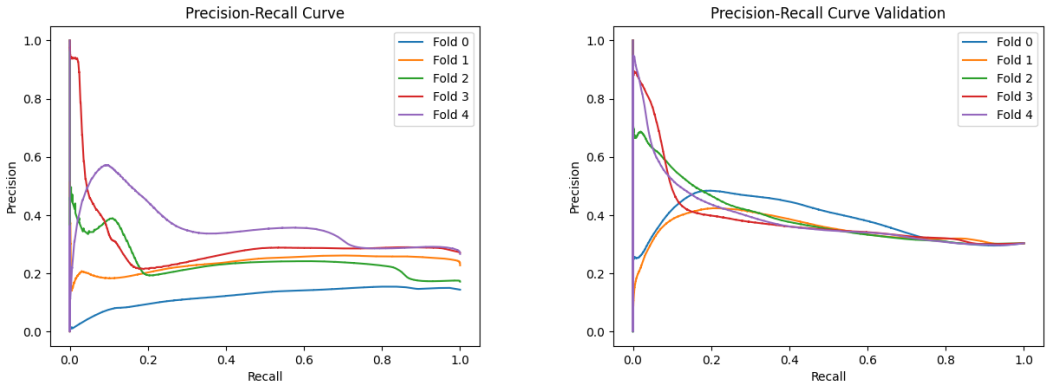


Figure A.30: Train set: {Berlin, Madrid}. Test set: {Stockholm}.

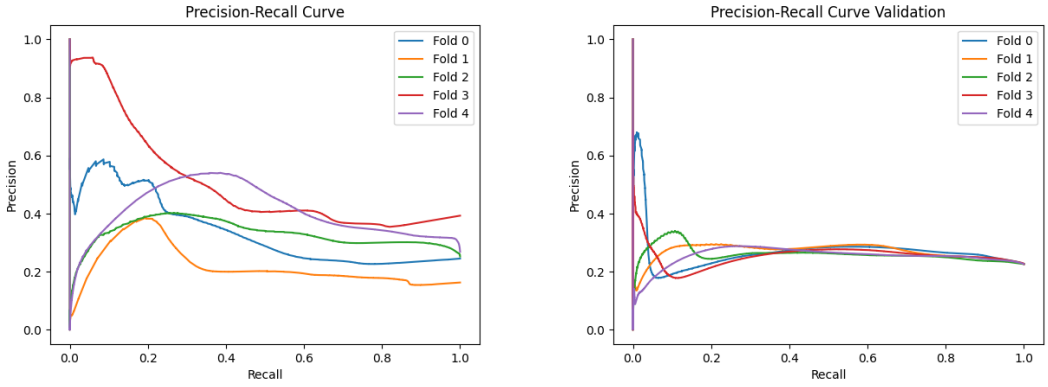
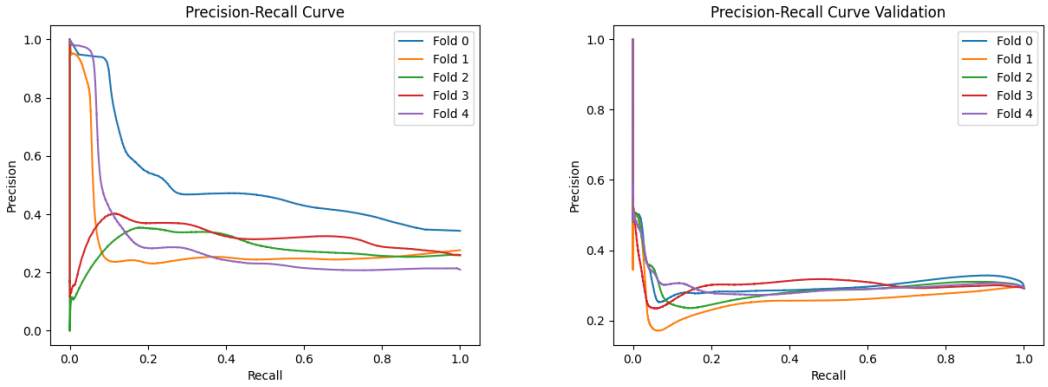


Figure A.31: Train set: {Berlin, Stockholm}. Test set: {Madrid}.



A.4 Scenario I: Comparison

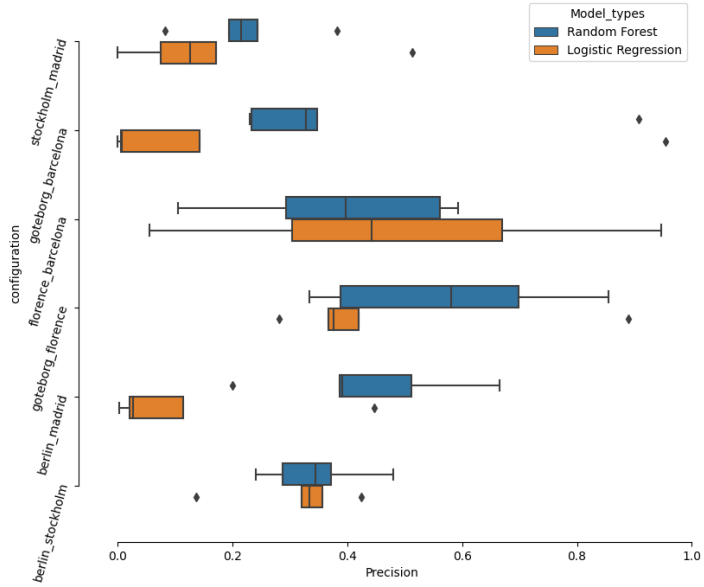


Figure A.32: Box-plot showing precision scores for Logistic Regression Classifier and Random Forest Classifier side-by-side.

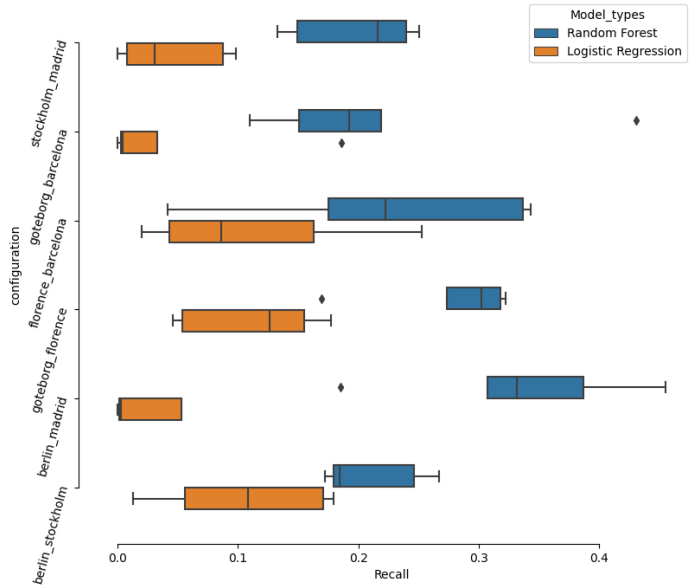


Figure A.33: Box-plot showing recall scores for Logistic Regression Classifier and Random Forest Classifierside-by-side.

A.5 Scenario II: Random Forest Classifier

A.5.1 Feature Importance

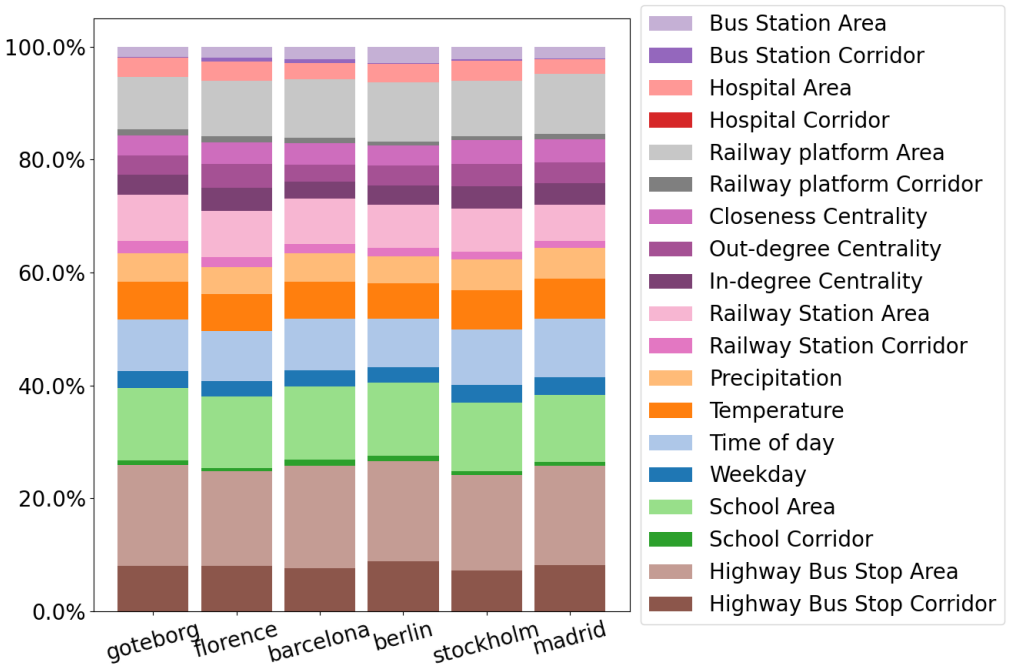
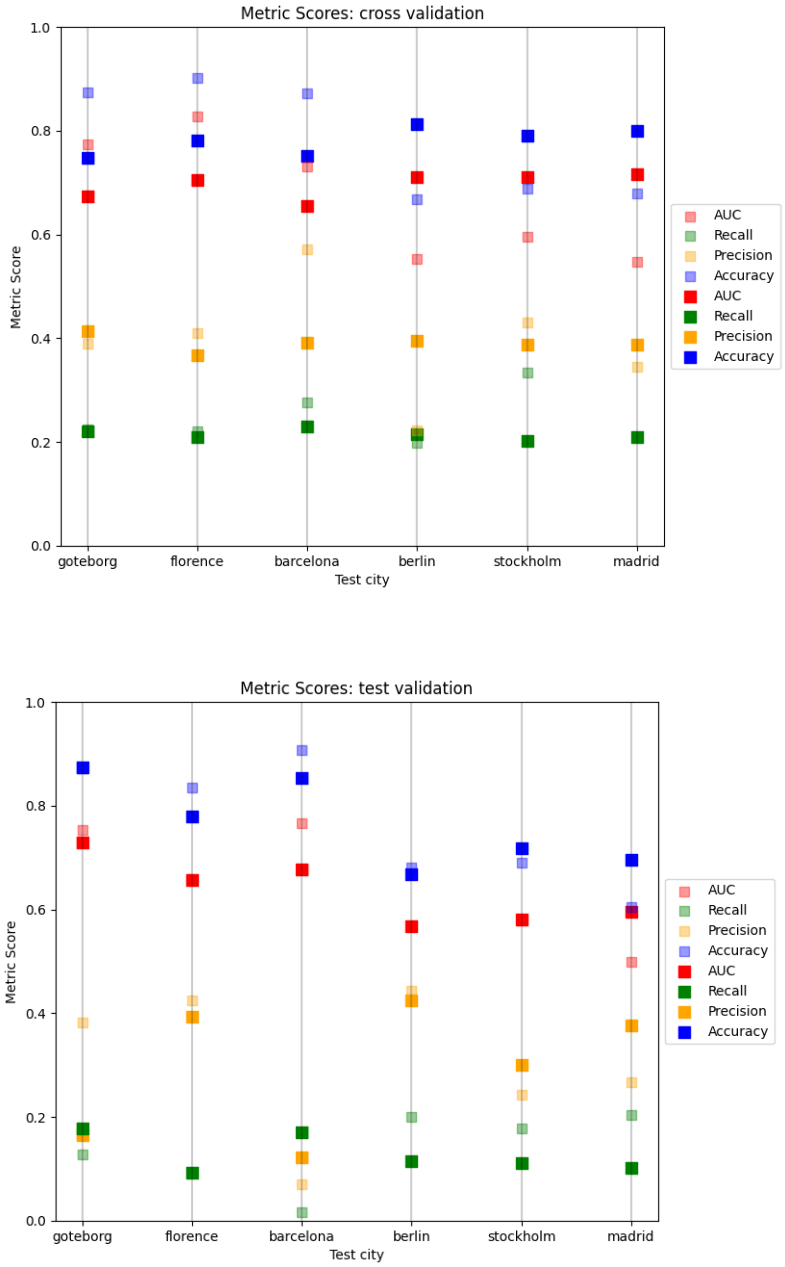


Figure A.34: A stacked bar chart showing the importances of every feature for averaged samples of 10 configurations for each test city.

A.5.2 Metric Scores

Figure A.35: Averaged metric scores of sampled configurations. The test cities for the average score for each set of 10 samples (used for calculating the validation scores for the last plot) are Gothenburg, Florence, Barcelona, Berlin, Stockholm and Madrid in that order.



A.5.2.1 ROC-curves

Figure A.36: Train set: {Barcelona, Florence}. Test set: {Berlin}.

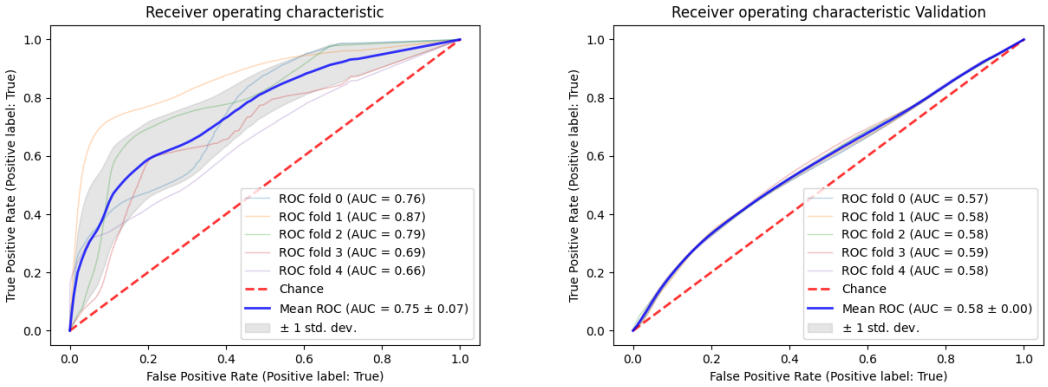


Figure A.37: Train set: {Barcelona, Gothenburg}. Test set: {Berlin}.

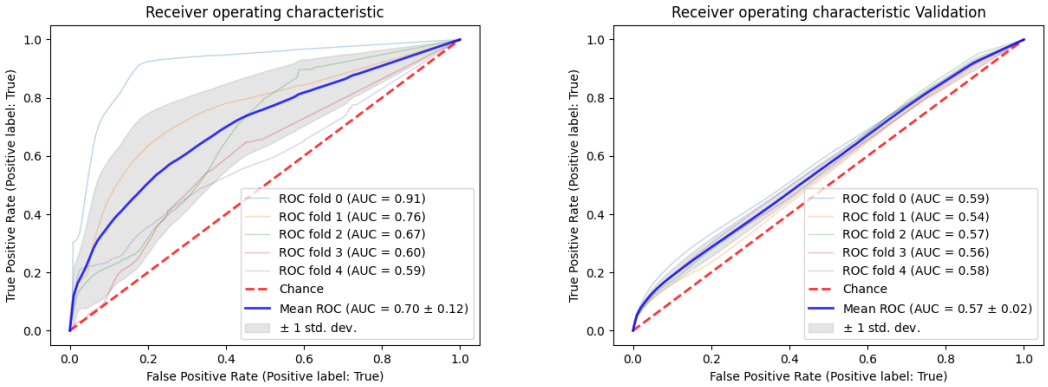


Figure A.38: Train set: {Florence, Madrid}. Test set: {Berlin}.

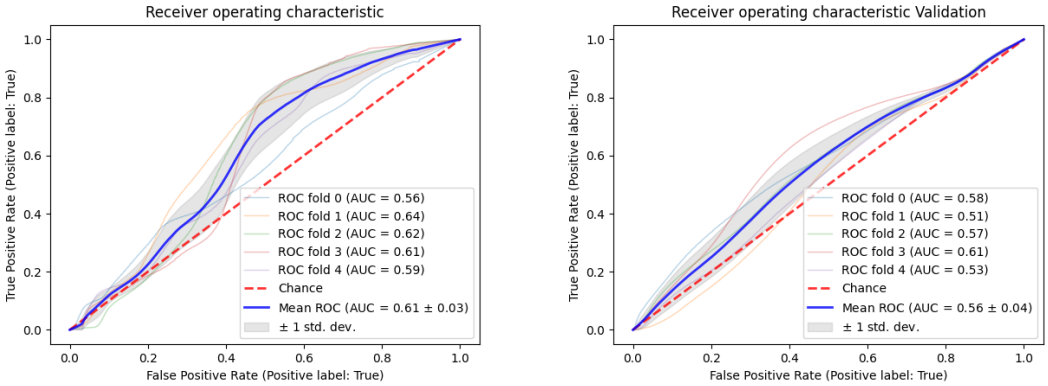


Figure A.39: Train set: {Stockholm, Barcelona}. Test set: {Berlin}.