Firm Order Estimation

Combining Artificial Intelligence and Firm Data to enable Enhanced Forecasting and Key Performance Indicator Estimation

Degree project report in Computer Engineering

Johan Birgersson & Erik Hillestad Andreasson

Department of Computer Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023
www.chalmers.se

Degree project report 2023

An Experimental Study of the Performance of a RNN Based Model versus a KNN Model on Estimating Firms Order Prices

-

Johan Birgersson, Erik Hillestad Andreasson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

An Experimental Study of the Performance of a RNN Based Model versus a KNN Model on Estimating Firms Order Prices
-
Johan Birgersson, Erik Hillestad Andreasson

© Johan Birgersson, Erik Hillestad Andreasson, 2023.

Supervisor: David Lidell, Computing Science
Examiner: Jonas Duregård,Computing Science

Degree project report 2023
Department of Computer Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

An Experimental Study of the Performance of a RNN Based Model versus a KNN Model on Estimating Firms Order Prices

-

Johan Birgersson, Erik Hillestad Andreasson
Department of Computer Engineering
Chalmers University of Technology

Abstract

This study investigated the performance of K-Nearest Neighbors (KNN) and Long Short-Term Memory (LSTM), a Recurrent Neural Network variant, in estimating missing data in both synthetic and real firm databases. Our objective was to identify the most effective model for imputation, considering estimation accuracy and robustness to data-specific characteristics, such as outliers and noise. Although KNN showed superior performance based on Mean Absolute Error and Mean Squared Error metrics, high Mean Absolute Percentage Error values observed for both models suggest potential issues, such as overfitting and the influence of extreme values. Neither model demonstrated significant promise for missing data estimation in this context, emphasizing the need for careful data preprocessing, model selection, and parameter tuning. Consequently, future research should consider alternative preprocessing techniques and machine learning models, underlining the importance of a nuanced understanding of the data and careful model and parameter selection for robust and accurate prediction outcomes.

Keywords: RNN, LTSM, KNN, Neural Network, Performance, Firm Data.

Acknowledgements

We would like to acknowledge our supervisor, Mr. David Lidell, for his guidance in our approach to the academic documentation. His last-minute support was key to the completion of this project. We also wish to thank our friends and family for their constant encouragement during this process.

Johan Birgersson & Erik Hillestad Andreasson, Gothenburg, June 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

KPI	Key Performance Indicator
RNN	Recurrent Neural Network
LSTM	Long-Short-Term Memory
KNN	K-Nearest Neighbors
MAE	Mean Absolute Error
MSE	Mean Squared Error
MAPE	Mean Absolute Percentage Error

Contents

List of Acronyms **ix**

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Background . . . 1
1.2 Purpose . . . 1
1.3 Goals . . . 2
1.4 Research Questions . . . 2
1.5 Limitations / Demarcations . . . 2

2 Theory **5**
2.1 Missing data estimation . . . 5
2.1.1 RNN . . . 6
2.1.2 KNN . . . 6
2.1.3 Comparison of RNN and KNN . . . 6
2.2 Synthetic Data in Missing Data estimation . . . 7
2.2.1 Justification for Using Synthetic Data . . . 7
2.2.2 Generating Realistic Synthetic Data . . . 7
2.2.3 Evaluation and Generalizability . . . 7
2.2.4 Comparisons with Real-World Data . . . 8
2.2.5 Transparency in Synthetic Data Generation . . . 8

3 Methods **9**
3.1 Study Design . . . 9
3.2 Data Collection . . . 9
3.2.1 Synthetic data: Northwind . . . 9
3.2.2 Real data: Digikala . . . 10
3.3 Data Processing . . . 10
3.4 Estimation Methods . . . 11
3.4.1 Recurrent Neural Networks (RNN) . . . 11
3.4.2 K-Nearest Neighbors (KNN) . . . 13
3.5 Data Analysis . . . 14
3.6 Evaluation Measures . . . 14
3.7 Factors Influencing Performance . . . 14

4 Results **15**
4.1 Basic Data Descriptions 15
4.2 Explanatory Data Analysis 17
4.2.1 Real data 17
4.2.2 Synthetic data 20
4.3 Empirical Results and Discussion 22
4.3.1 Real Data 23
4.3.2 Synthetic Data 24

5 Conclusion **27**

Bibliography **29**

A Appendix 1 **I**
A.1 Experimental Findings: Real Data I
A.2 Experimental Findings: Synthetic Data II
A.3 Winsorizing III

List of Figures

4.1 Order Price Over Time for Real Order Data (Iranian Rial) 16
4.2 Order Price Over Time for Synthetic Order Data (U.S. Dollar) 16
4.3 Spearman test on real data 18
4.4 Data distribution target variable real data 19
4.5 Spearman test on Synthetic data 20
4.6 Data distribution target variable synthetic data 22

A.1 KNN, PowerTransformer I
A.2 KNN, RobustScalar I
A.3 RNN, PowerTransformer 5 days timewindow I
A.4 RNN, RobustScalar 5 days timewindow I
A.5 KNN, PowerTransformer II
A.6 KNN, RobustScalar II
A.7 RNN, PowerTransformer 5 days timewindow II
A.8 RNN, RobustScalar 5 days timewindow II
A.9 Unwinsorized Data Distribution III
A.10 Winsorized Data Distribution III

List of Tables

3.1 Model Tuning: RNN 12

4.1 Descriptive Statistics of Real Order Data 15
4.2 Descriptive Statistics of Synthetic Order Data * Manually Created variable . . . 15
4.3 Model Comparison: Real data 23
4.4 Model Comparison: Synthetic data 24

1

Introduction

1.1 Background

As businesses continue to operate in an increasingly competitive and fast-paced market, the ability to make accurate and timely decisions is more important than ever. One way to achieve this is by utilizing data and artificial intelligence (AI) to gain insights and make predictions about future performance. One specific area where businesses often struggle is with the process of forecasting key performance indicators (KPIs) such as order flow. This is because the data they have may be incomplete or contain gaps, making it difficult to get an accurate picture of future performance.

Given this challenge, there is a clear need for a solution that can effectively process and analyze large amounts of data in real-time, and use AI to predict and fill in gaps in order to estimate future KPIs. A database-connected dashboard that can be used as a database look-up and incorporates real-time financial market adjustments would be an ideal solution to address these issues.

This project aims to develop such an application that can help businesses stay competitive by providing them with real-time insights and predictions based on financial market data. It would enable them to make more informed decisions and improve their forecasting capabilities, ultimately leading to better performance and increased profitability.

1.2 Purpose

The purpose of our study is the investigation and evaluation of two distinct methodologies for missing data estimation, specifically Long Short-Term Memory networks (a type of Recurrent Neural Networks) and K-Nearest Neighbors, both in the context of synthetic and real firm databases. We aim to explore and compare these methods using two different types of data, each simulating different aspects of real-world scenarios. Our ultimate goal is to provide insights into these methods' performance and suitability, leading to well-informed recommendations for choosing the most efficient method for handling missing data issues. This study, we hope, will lay the groundwork for the future practical applications in this field.

1.3 Goals

The primary goals of this study are:

1. To implement and apply RNN and KNN methods for missing data estimation in a the context of a firm's database.

2. To evaluate and compare the performance of the RNN and KNN methods in terms of estimation accuracy, computational efficiency, and robustness to outliers and noise.

3. To identify the factors influencing the performance of each method, such as data characteristics, missing data mechanisms, and model parameters.

4. To provide answer to the following research questions addressing the suitability and effectiveness of RNN and KNN in handling missing data within the specific context of the synthetic firm's database.

5. To provide recommendations for the best-fitted method for missing data estimation in the given context, considering the trade-offs between model complexity, computational efficiency, and estimation performance.

1.4 Research Questions

The following research questions have been formulated to address the goals of this study and evaluate the suitability and effectiveness of RNN and KNN methods in handling missing data within the specific context of the firm's database:

1. RQ1: How do the RNN and KNN methods perform in terms of estimation accuracy?

2. RQ2: How robust are the RNN and KNN methods to outliers and noisy data?

3. RQ3: What factors influence the performance of the RNN and KNN methods in the context of the firm's database, such as data characteristics, missing data mechanisms, and model parameters?

1.5 Limitations / Demarcations

This study is subject to the following limitations and demarcations:

1. The focus is on RNN and KNN methods for missing data estimation, and other estimation methods are not considered in this study.

2. The evaluation and comparison of the methods are conducted within the specific context of the synthetic firm's database.

3. The performance of the methods may be influenced by the choice of model parameters.

4. The study is limited to assessing the performance of the methods without exploring their impact on downstream applications or forecasting tasks.

2

Theory

2.1 Missing data estimation

Missing data is a prevalent issue in data analysis, occurring for various reasons such as data entry errors, technical problems, or user omissions [5]. Incomplete data can lead to biased or inaccurate results, impacting the quality of decision-making based on that data. Numerous methods are available for missing data estimation, each with its own strengths and weaknesses. Some of the most common methods include mean estimation, regression estimation, and multiple estimation [5]. More recently, machine learning algorithms such as Recurrent Neural Networks (RNN) and K-Nearest Neighbors (KNN) have been applied to missing data estimation, demonstrating promising results [7, 3]. In this section we explore different methods used for missing data estimation, focusing on two specific approaches: RNN [2] and KNN [1].

The comparison between RNN and KNN in the context of missing data estimation is particularly relevant due to their unique strengths and differences in handling various data types and scenarios. Both methods have demonstrated promising results in recent papers [7, 3], making them suitable candidates for comparison. RNN has gained considerable attention in the scientific community due to its ability to model temporal dependencies in sequential data [2]. This capability has made RNN especially effective in handling time series data with missing values, as it can predict missing values based on past observations [7]. RNNs have been successfully applied in various domains, such as healthcare, finance, and transportation [6, 2]. Furthermore, the use of advanced RNN variants like Long Short-Term Memory (LSTM) [9] and Gated Recurrent Units (GRU) [8] has improved the performance of RNNs in capturing long-term dependencies, addressing the vanishing gradient problem in traditional RNNs. On the other hand, KNN is a well-established non-parametric method that has been widely used for classification and regression tasks [1]. Its simplicity and ease of implementation have made it popular in various applications, including missing data estimation [3]. KNN can handle both numerical and categorical data and is effective in estimating missing values based on the similarity between data points [4]. Additionally, KNN is considered a lazy learning method, as it does not require an explicit training phase, which can be advantageous in certain situations.

Given the unique characteristics and strengths of RNN and KNN, comparing these two methods can provide valuable insights into their performance and applicability in different data scenarios. Their comparison allows for a more comprehensive understanding of the trade-offs between computational complexity, training data requirements, and robustness to outliers and noise, which can inform researchers and practitioners in selecting the most appropriate method for missing data estimation tasks.

2.1.1 RNN

RNN is a type of neural network architecture commonly used for sequence modeling and prediction [2]. In missing data estimation, RNN can model temporal dependencies between variables and predict missing values based on past observations [7]. Moreover, the model has been shown to perform well in predicting missing values in time series data and has been applied in various domains, such as healthcare, finance, and transportation [6, 2].

A significant advantage of RNNs is their ability to capture and leverage the temporal information inherent in time series data [10]. Traditional methods for time series data estimation often do not account for the temporal relations among observations and treat time series as typical structured data, resulting in the loss of information from time data [10]. In contrast, deep learning models, such as the RNN, have gained attention for their effectiveness in handling time series data, as they can capture the time information from data and improve the performance of estimation tasks [10]. RNNs can be further enhanced by incorporating more sophisticated architectures, such as Long Short-Term Memory (LSTM) networks or Gated Recurrent Units (GRUs), which address the vanishing gradient problem and improve the model's ability to capture long-range dependencies [9, 2].

2.1.2 KNN

KNN is a non-parametric method commonly used for classification and regression tasks [1]. In missing data estimation, KNN can impute missing values based on the similarity between the missing data point and its nearest neighbors [3]. KNN has been shown to be effective in estimating missing values in both numerical and categorical data and can handle high-dimensional data sets [4]. One of the main advantages of KNN is its simplicity and ease of implementation, making it a popular choice for various missing data estimation tasks. Additionally, KNN can be further improved by incorporating various distance metrics or weighting schemes, allowing the method to be tailored to the specific characteristics of the data set [3].

2.1.3 Comparison of RNN and KNN

Both RNN and KNN possess strengths and weaknesses in missing data estimation. RNN is particularly effective in handling time series data, where missing values often correlate with past observations [7]. However, RNN requires a large amount of

training data to perform well and can be computationally expensive [2]. In contrast, KNN is a simpler method that can handle missing values in both numerical and categorical data without requiring a large amount of training data [4]. Nevertheless, KNN can be sensitive to outliers and noisy data, and may not be suitable for high-dimensional data sets [3]. The choice between RNN and KNN for missing data estimation will depend on the specific requirements of the data set and the desired trade-offs between model complexity, computational efficiency, and estimation performance.

In conclusion, missing data estimation is a critical issue in data analysis, with numerous methods available to address it. RNN and KNN are two promising approaches that have been shown to be effective in different contexts [7, 3]. Depending on the data's nature, one method may be more suitable than the other. Further research is needed to explore these methods' performance in various domains and identify the best practices for missing data estimation.

2.2 Synthetic Data in Missing Data estimation

Synthetic data has gained significant interest in recent years as a valuable alternative to real-world data, particularly in situations where obtaining real-world data is challenging or when researchers want to explore specific trends or patterns in a controlled setting. This section presents a literature review on the use of synthetic data in missing data estimation and its implications.

2.2.1 Justification for Using Synthetic Data

Several studies have highlighted the benefits of using synthetic data in research projects. One of the primary advantages is the ability to generate data with specific characteristics, which allows researchers to explore various aspects of the problem in a more controlled way [18]. In addition, synthetic data can help overcome privacy concerns, as it does not contain information about real individuals, making it a suitable option for research involving sensitive data [12].

2.2.2 Generating Realistic Synthetic Data

To ensure the validity of research findings based on synthetic data, it is crucial to generate data that closely mimics the characteristics of real-world data. Several methods have been proposed for generating synthetic data, including parametric and non-parametric methods [17, 16]. In addition, methods based on deep learning, such as Generative Adversarial Networks (GANs), have shown promising results in generating realistic synthetic data [13].

2.2.3 Evaluation and Generalizability

When using synthetic data, it is essential to consider the limitations of evaluation and the generalizability of the findings to real-world scenarios. One approach to

addressing this issue is to use multiple evaluation metrics that capture different aspects of the estimation performance [14]. It is important to also carefully discuss the potential impact of using synthetic data on the generalizability of their findings to real-world applications [18].

2.2.4 Comparisons with Real-World Data

Comparing the performance of models on both synthetic and real-world data can strengthen the validity of the research findings. This approach has been employed in various studies, showing that models developed and evaluated on synthetic data can perform well on real-world data [11, 15]. As such, this study makes compares the model performance on both real and synthetic data. Note however, that the synthetic data in our case has a few different characteristics as compared to the real data. These differences are describes further in the data section of the methodology.

2.2.5 Transparency in Synthetic Data Generation

Transparency in synthetic data generation is crucial for understanding and replicating research findings. Researchers should clearly document the data generation process, including any assumptions made, methods used, and the parameters or distributions involved [12]. This transparency allows others to evaluate the quality of the synthetic data and assess the robustness of the research findings. In conclusion, synthetic data can be an acceptable and valuable alternative to real-world data in missing data estimation research, provided that researchers carefully justify its use, generate realistic data, consider the limitations of evaluation and generalizability, compare their findings with real-world data when possible, and maintain transparency in the data generation process.

3

Methods

This chapter outlines the research methodology employed in this study. The objective is to provide a detailed description of the study design, data collection, data analysis, and evaluation measures used to investigate and evaluate the effectiveness of two different approaches to missing data estimation: Recurrent Neural Networks (RNN) and K-Nearest Neighbors (KNN). Our study will test these models on two distinct datasets. One which is synthetic, and one which contains real firm data.

3.1 Study Design

This study will use a quantitative approach and follow an experimental design. Two different methods (RNN and KNN) will be used to estimate missing data on a synthetic firm's database as well as the real firm data. The performance of each method will then be evaluated and compared.

3.2 Data Collection

We employ two methods of data collection synthetic and realistic data. The reason for this choice stems from the initial difficulty in finding real-world data on firm orders. Initial screening suggested that we would be able to test our estimation on synthetically created data. Albeit this task proved to be challenging considering the degree of complexity one would need to mimic, and the rigor in documenting these characteristics one would need to employ in order to be able to draw conclusions from it. Instead, we found that there exists testing data for firm databases, as such we obtained this data and proceeded to model on this data. As uncovered in the Exploratory data analysis the data is found to be synthetic without the intended aim to be realistic in terms of exhibiting realistic customer behaviors. Therefore, we proceeded to look for real data on orders with the aim to remedy these drawbacks. This lead us to the real data on the firm Digikala described below.

3.2.1 Synthetic data: Northwind

The Northwind database from YugabyteDB will be utilized to simulate real-world data. The database is created with the intetion of providing developers with a testing database for development purposes. The dataset contains no missing values

by design. As such, to test the data with missing values we split the data into a test and a train set per a 80 and 20 percent respective split. We use this method since we assume that there will be no pattern in the missing data and that the observations missing will be missing at random. The synthetic comes in the form of a database containing various tables representing a synthetic firm's operations, including customers, orders, products, and sales. The data within the Northwind database will be queried to extract a dataset of order data, which mimics common data characteristics and missing data mechanisms found in business settings. Some limitations of this data are covered in the Exploratory Data analysis.

3.2.2 Real data: Digikala

The Digikala data is a database obtained through Kaggle, which is a platform for companies to pose firm-specific challenges for Data Scientists. We utilize data on orders from 2013 to 2018 to test our model on real-world firm data. Similarly to the synthetic data there are no missing values in the dataset. For our experiments we have assumed that the data we estimate is missing at random, and as such we perform a regular 80/20 percent train/test split on the data. The real data has a difference in the features contained in the dataset as compared to the synthetic. Namely, the Digikala dataset contains features on Order ID, Customer ID, Item ID, Order Price, and Purchase Quantity.

3.3 Data Processing

We perform some necessary preprocessing on both datasets, that is conversion into the necessary types for the models. Noteworthy is that the data we utilize in this study is intended to be in the authentic format of a database registry of firm orders. As such, this report does not make attempts at generating additional features than those provided from the datasets, the only exception to this is in the feature engineering of the target variable of the synthetic dataset where we combine unitprice and quanity as well as add freight cost and remove discount of each order to make it comparable to the real world data set.. This is done in order to better be able to compare the models performance on the two datasets.

For the real world data we find that there are distinct outliers that have the potential to impact the model accuracy greatly see appendix Figure 9. Therefore we implement the option to winsor the data, flattening the top and bottom five percents of the data distribution to minimize disproportionate learning from these outliers. This carries the risk that our model will be estimating values lower than otherwise but since we aim to implement a distance based metric in our KNN model we add this option in order to test the effect of reducing outliers. Additionally this gives us an additional way, apart from feature scaling to test the robustness of our findings.

3.4 Estimation Methods

3.4.1 Recurrent Neural Networks (RNN)

The logic for utilizing a recurrent neural network based model is found in the sequential aspect of the model. We hypothesise that with seasonal, cyclical business effects present in the orders of firms we may be able to better capture the effect of time using such a model. RNN models will be designed and trained to handle sequential data with missing values. The architecture, hyperparameters, and training techniques will be carefully selected to optimize the model's performance.

Model architecture

LSTM Layers: The number of LSTM layers and the number of units within each layer are hyperparameters. The code allows for the flexibility of choosing between 1 and 3 LSTM layers (for i in range(hp.Int('num_layers', 1, 3))). The number of units in each LSTM layer is set within the range of 32 to 256 with a step size of 32 (units=hp.Int(f'num_units_i', 32, 256, step=32)). This flexibility in the number of layers and units allows for exploring different model complexities.

Dropout Layers: Dropout layers are used for regularization to prevent overfitting. The decision of whether to include dropout layers or not is a hyperparameter (if hp.Choice('use_dropout', [True, False])). If chosen, the dropout rate is also a hyperparameter (rate=hp.Float('dropout_rate', 0.1, 0.5, step=0.1)). The range of dropout rates from 0.1 to 0.5 with a step size of 0.1 provides a reasonable range of values to explore.

Time-Distributed Dense Layer: The output layer of the model is a time-distributed dense layer with a linear activation function. This layer is responsible for predicting the missing values. The use of a time-distributed dense layer allows the model to generate a prediction for each time step in the sequence.

The model is compiled with the Mean Squared Error (MSE) loss function and the Adam optimizer. The learning rate is also a hyperparameter, and it is defined within the range of 1e-4 to 1e-2 (optimizer=Adam(lr=hp.Float('learning_rate', 1e-4, 1e-2, sampling='log'))). The choice of the Adam optimizer and the MSE loss function is commonly used for regression tasks and has shown good performance in neural network models.

Hyperparameter Tuning

Parameter	Range (Stepsize)
num_layers	1-3
num_units_i	32-256
use_dropout	Yes/No
dropout_rate	0.1-0.5 (0.1)
learning_rate	1e-4 to 1e-2

Table 3.1: Model Tuning: RNN

The hyperparameter search space is defined using the Keras Tuner's RandomSearch method. The key hyperparameters that are tuned include:

Number of LSTM Layers (num_layers): This hyperparameter allows for exploring the number of LSTM layers in the model. The range of 1 to 3 provides flexibility in model complexity.

Number of Units in LSTM Layers (num_units_i): The number of units within each LSTM layer is a hyperparameter that controls the capacity and expressiveness of the model. The range of 32 to 256 with a step size of 32 allows for exploring different granularities of units.

Use of Dropout Layers (use_dropout): Dropout layers are a regularization technique. This hyperparameter allows for the exploration of whether to include dropout layers in the model or not.

Dropout Rate (dropout_rate): If the use of dropout layers is chosen, the dropout rate becomes a hyperparameter. The range of 0.1 to 0.5 with a step size of 0.1 covers a reasonable range of dropout rates.

Learning Rate (learning_rate): The learning rate is a crucial hyperparameter that controls the step size during optimization. The range of 1e-4 to 1e-2 with a logarithmic sampling ('sampling='log') provides a diverse range of learning rates to explore.

The motivation behind these choices is to allow for flexibility and exploration of different model architectures and hyperparameters. By tuning the number of LSTM layers and units, the model can adapt to different complexities of the data and capture temporal dependencies. The inclusion of dropout layers helps prevent overfitting and improves the generalization ability of the model. The choice of the Adam optimizer and MSE loss function is based on their effectiveness in regression tasks.

The hyperparameter search space is defined to cover a reasonable range of values for each hyperparameter. The range of values for the number of LSTM layers, units, and dropout rates provides a balance between model complexity and simplicity. The logarithmic sampling for the learning rate allows for a diverse exploration of learning rates across different orders of magnitude.

The RandomSearch method with a limited number of trials and executions per trial helps strike a balance between computational resources and the exploration of the hyperparameter search space. The validation split and early stopping based on validation loss ensure that the model's performance is assessed on unseen data and helps prevent overfitting.

Overall, the model architecture and the defined hyperparameter search space aim to find an optimal RNN model configuration that can effectively impute missing values within the Digikala and Northwind databases while balancing model complexity, generalization ability, and computational efficiency.

3.4.2 K-Nearest Neighbors (KNN)

As a canvas for comparison we use Knearest neighbor as a proxy for the mean imputation often used in statistical methods of imputing missing data. As such, a KNN algorithm is implemented to estimate missing values based on similarity measures. This technique assumes that similar data points are near to each other in the feature space. The parameters, like the number of neighbors (k) and the type of distance metric, will be decided based on the nature of the data and the performance requirements. For the continuous variables, Euclidean distance may be used, while for categorical variables, Hamming distance could be a better choice. The model's effectiveness will be improved by applying feature scaling and dimensionality reduction techniques before implementing KNN.

Model architecture

The KNN model is defined using the KNeighborsRegressor class from scikit-learn. The default hyperparameters of the KNN model are used in this implementation.

Hyperparameter Tuning

Hyperparameter Search: To determine the optimal number of neighbors (K) for the KNN model, a grid search is performed using the GridSearchCV class from scikit-learn. The grid of hyperparameters to search over is defined by the param_grid dictionary, with the n_neighbors parameter ranging from 1 to 30. The grid search utilizes a 5-fold cross-validation (cv=5) for evaluating different hyperparameter combinations.

Grid Search Fitting: The grid search is fit to the flattened training data (X_train_flat, y_train_flat) using the fit() method. This process evaluates the model's performance with different hyperparameter combinations and identifies the best model based on the specified scoring metric (default is mean squared error).

Best Model Selection: The best model obtained from the grid search is extracted using the best_estimator_ attribute of the GridSearchCV object. The best KNN model is stored in the variable best_knn.

The motivation behind this approach is to explore the optimal number of neighbors

for the KNN model, which is a critical hyperparameter. By performing a grid search with cross-validation, we evaluate various KNN models and select the one that yields the best performance based on the chosen scoring metric.

The grid search allows for an efficient exploration of the hyperparameter space, in this case, the range of n_neighbors. By evaluating multiple combinations, we can identify the KNN model that provides the most accurate estimation of missing values within the Northwind database.

Overall, this methodology aims to leverage the KNN algorithm's simplicity and effectiveness in imputing missing data by determining the optimal number of neighbors through a grid search. This ensures that the KNN model utilized for missing data estimation within the Northwind database is fine-tuned and capable of producing accurate estimations.

3.5 Data Analysis

The data analysis will involve the application of the RNN and KNN methods to the synthetic and real firm's data. The estimated data will then be analyzed to assess the performance of each method.

3.6 Evaluation Measures

The evaluation of the estimation methods will be based on three criteria: estimation accuracy, computational efficiency, and robustness to outliers and noise.

estimation accuracy will be measured using Mean Absolute Error (Mean Absolute Error) and Mean Absolute Percentage Error (MAPE). Lower values indicate better accuracy.

Computational efficiency will be assessed by measuring the runtime of each estimation method, with a shorter runtime indicating better efficiency.

Robustness to outliers and noise will be evaluated by maintaining and then reducing outliers using feature and target processing techniques. Namley, Robust Scaler and PowerTransformer and noise to the data and comparing the performance of the two methods.

3.7 Factors Influencing Performance

An in-depth analysis will be conducted to identify factors that influence the performance of each method. These factors include data characteristics (e.g., types of variables, distribution of data), and model parameters (e.g., number of layers in RNN, number of neighbors in KNN).

4

Results

4.1 Basic Data Descriptions

Table 4.1 presents the descriptive statistics of the real dataset prior to any preprocessing, and feature scaling. Here we can see that there are 2 000 observations, the average price of an order is about 1.5 million Iranian Rial, with an enormous standard deviation of 4.1 million. The 50th percentile (median) reveals that the data is heavily skewed to the left with a median value of 350 000 Iranian Rial.

	Order ID	Customer ID	Item ID	Order Price	Purchase Quantity
count	2,000	2,000	2,000	2,000	2,000
mean	9,926,576	2,926,650	373,264	1,485,139	1
Std	6,460,968	1,925,163	387,018	4,128,435	2
min	1,025,391	466,211	113	1	1
25%	5,078,834	1,010,518	93,309	130,000	1
50%	8,635,904	2,975,940	236,270	350,000	1
75%	12,451,809	4,320,636	507,420	1,121,078	1
max	24,837,058	7,267,099	1,953,312	70,082,569	40

Table 4.1: Descriptive Statistics of Real Order Data

Table 4.2 presents the descriptive statistics for the synthetic data, similar to table 4.1. Here we can see that the synthetic dataset contains 2155 observations and 8 features, plus one manually created feature (Order Price).

	Order ID	Employee ID	Shipping Method	Shipping Cost	Product ID	Unit Price	Purchase Quantity	Price Discount	Order Price*
count	2,155	2,155	2,155	2,155	2,155	2,155	2,155	2,155	2,155
mean	10,659	4	2	96	41	26	24	0	634
std	241	2	1	133	22	30	19	0	1,041
min	10,248	1	1	0	1	2	1	0	5
25%	10,451	2	1	20	22	12	10	0	156
50%	10,657	4	2	54	41	18	20	0	361
75%	10,862	7	3	121	60	32	30	0	734
max	11,077	9	3	1,008	77	264	130	0	15,827

Table 4.2: Descriptive Statistics of Synthetic Order Data * Manually Created variable

Figures 4.1 and 4.2 show the 7-day rolling mean of order prices for the real and synthetic datasets, respectively. Both figures reveal an incredibly large gap throughout the observed periods between the smaller order values and larger order values.

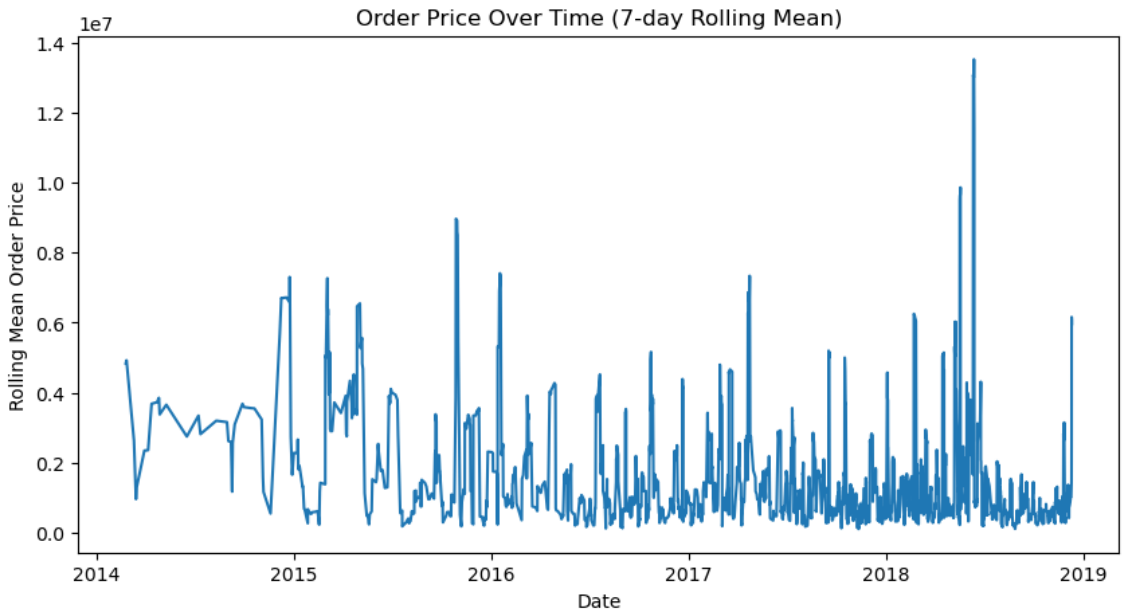

Figure 4.1: Order Price Over Time for Real Order Data (Iranian Rial)

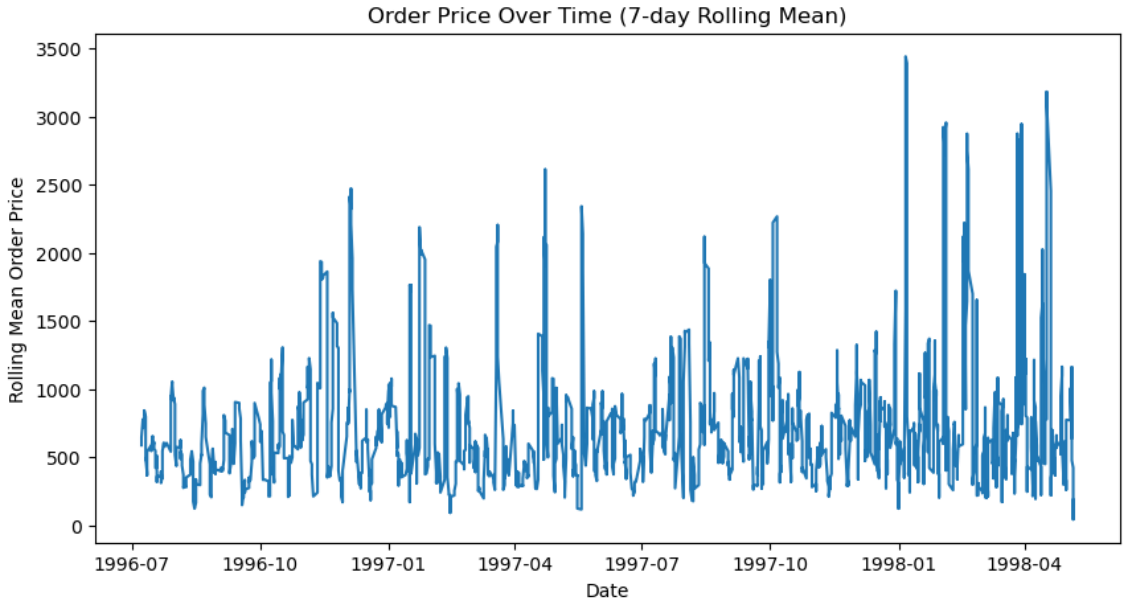

Figure 4.2: Order Price Over Time for Synthetic Order Data (U.S. Dollar)

4.2 Explanatory Data Analysis

This section explores the underlying characteristics of the two different data sets employed in this paper. Thus, allowing us to understand the requirements for preprocessing and adaptations needed for modeling. The explanatory data analysis is, in many cases, the most important step as it will allow us to process the data appropriately and interpret the subsequent results.

Henceforward, the explanatory analysis is divided into two parts. The first part focuses on the "real" data set, based on the Iranian Amazon.com-like company Digikala. The second part looks at the synthetic data set retrieved from Kaggle. Jointly, the two sections provide a detailed overview of the two data sets employed, allowing for an exhaustive comparative analysis in later sections.

4.2.1 Real data

From the correlation matrix derived using Spearman's Rank correlation, we can find several insights which would be valuable in constructing and interpreting our predictive models - LSTM and KNN. Figure 4.3 presents Spearman's rank correlation matrix on the real dataset. Here we can see the relationship between the different variables. The matrix reveals a few things right away. Unlike Pearson correlation, Spearman's rank correlation does not presuppose a linear connection between the variables or that they are assessed on an interval scale. Instead, it is a non-parametric test to measure the strength and direction of association between two ranked variables. As a result, the Spearman test is more robust to outliers and non-linear relationships compared to the Pearson correlation.

The primary observation from the correlation matrix is the strong positive correlation among 'Order ID', 'Customer ID', and 'Item ID'. Although these are unique identifier variables, the high correlation implies a potential structure in the way orders are placed and fulfilled. This pattern could be exploited by our LSTM model, which is adept at recognizing temporal dependencies and patterns in sequential data. For instance, it might identify a relationship between the order in which customers place orders or the sequence of items being purchased.

Meanwhile, the low correlation between 'Order Price' and the rest of the features might present a challenge, especially for the KNN algorithm. KNN, as an instance-based learning method, relies heavily on the similarity of feature vectors to make predictions. A low correlation suggests that the "distance" in feature space might not be strongly related to the differences in the target variable, which is what KNN implicitly assumes. This could potentially impact the predictive accuracy of our KNN model. To address this, we will be experimenting with different distance metrics, namely Euclidean and Manhattan, to find an appropriate measure of similarity for our dataset.

On the contrary, for LSTM, this low correlation might not be an issue since LSTM models can potentially capture more complex, non-linear relationships between pre-

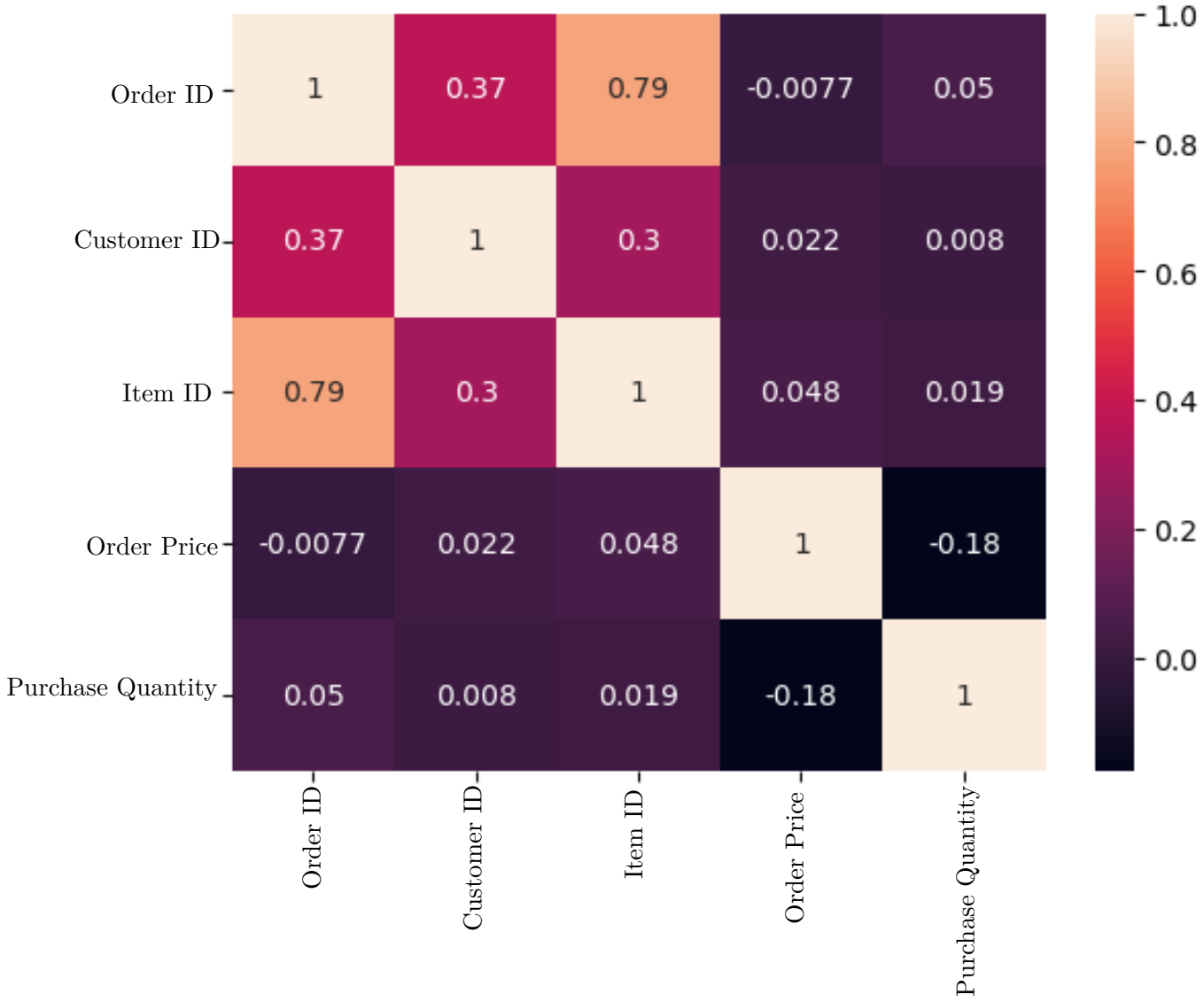

Figure 4.3: Spearman test on real data

dictors and the target variable. The sequence in which data is presented could play a significant role in such cases, and LSTM models are specifically designed to work well with this kind of data.

Lastly, since the correlation coefficients offer valuable insights into the structure and relationships in the dataset, the predictive models we're using (LSTM and KNN) have the ability to capture even more complex relationships and patterns. However, While the models may uncover the named additional complexity beyond the Spearman correlations, it is still important to understand the correlations as they act as a foundation for interpreting the results in the later analysis.

Figure 4.4 presents both the histogram and Q-Q plot of the 'Order Price' variable. The histogram reveals a left-skewed distribution. This means that most orders are of relatively lower value, while high-value orders are much more scarce. The order

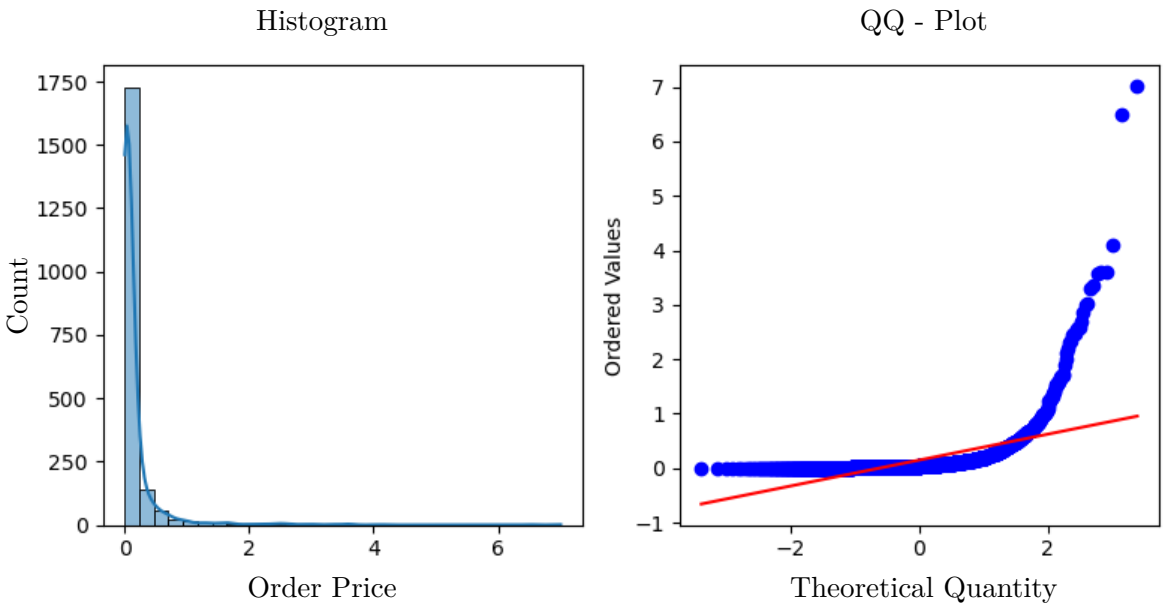

Figure 4.4: Data distribution target variable real data

values follow a common pattern in e-commerce, where a large number of customers make small purchases and a small number of customers make large purchases. However, these large purchases can significantly contribute to the total revenue.

The skewness is an important aspect to consider in regards to our models. While, KNN and LSTM models do not make any assumptions about the distribution of the target variable, extreme values have a greater impact on the KNN model due to its method of using distance between samples to make predictions. Hence, data transformations that mitigate the skewness have the possibility to improve model performance. Additionally, a salient feature of the LSTM is its ability to process sequential data. In our data, high-value orders might represent specific moments in time (e.g., holidays, promotions) that could be better understood when looking at the sequence of orders.

The Q-Q plot indicates that the distribution of 'Order Price' deviates from the normal distribution. This is specifically apparent in the upper quantiles of the distribution, where the plot shows a steep upward curve. This signifies a long tail on the right side of the distribution, a characteristic of positive or right skewness.

In relation to our models, this deviation from normality as indicated by the Q-Q plot again underscores a potential need for transformations to mitigate this skewness and outlier distributions in our data. For the KNN model, closer proximity of data points after such a transformation might lead to more accurate predictions. For the LSTM, having the data in a more normal distribution could help with the stability and efficiency of the learning process and thus improve estimation accuracy and robustness.

To summarize, both the histogram and the Q-Q plot underscore the importance of

exploring data transformations as a preprocessing step before modeling. While our chosen models, LSTM and KNN, do not explicitly assume a specific distribution of the target variable, achieving a distribution closer to normal could potentially aid our result in regards to improved model performance in two aspects, estimation accuracy, and robustness.

4.2.2 Synthetic data

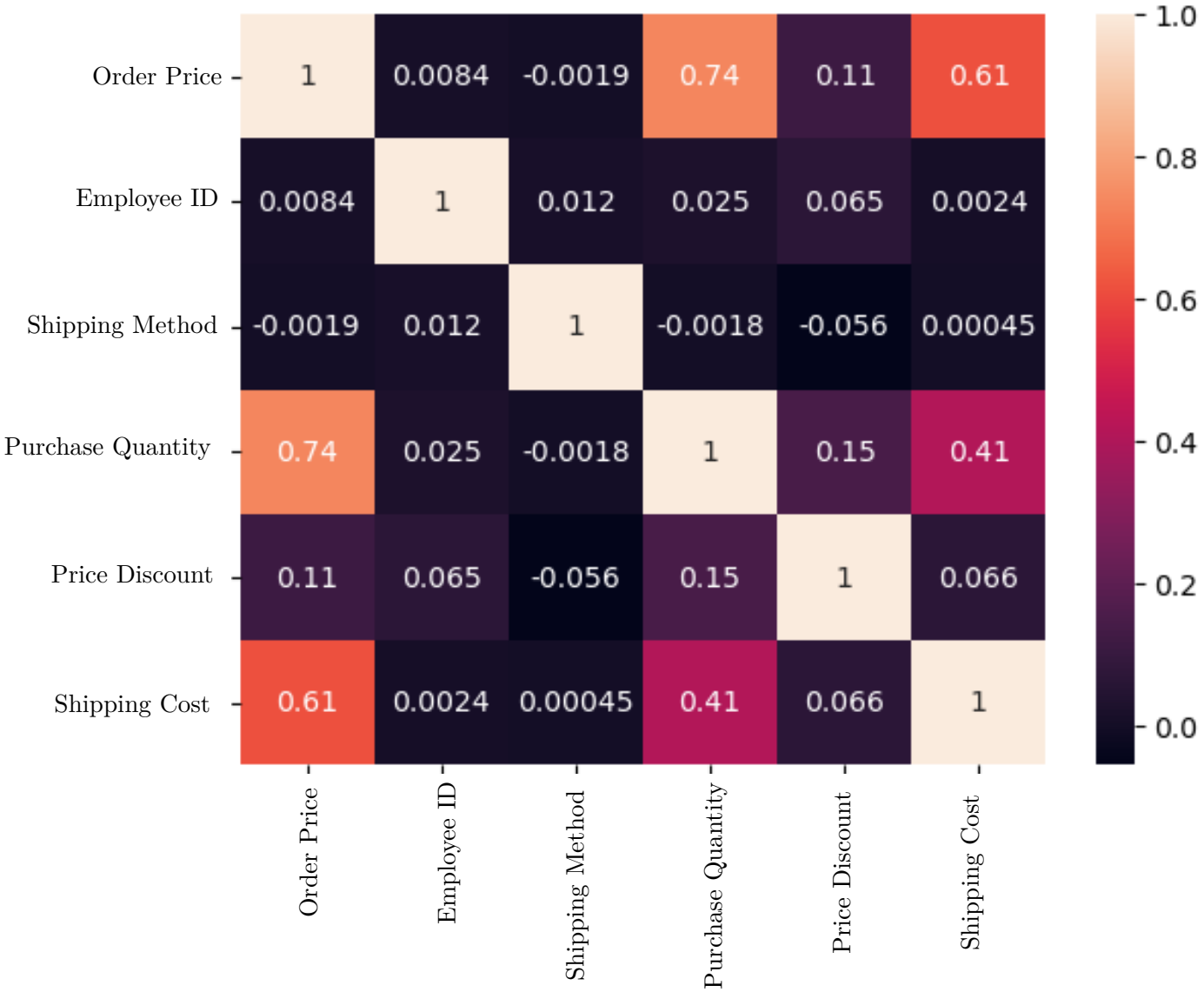

Figure 4.5: Spearman test on Synthetic data

As a part of uncovering patterns and understanding the data, this study presents the Spearmans's rank correlations for our six variables: 'Order Price', 'Employee ID', 'Shipping Method', 'Purchase Quantity', 'Price Discount', and 'Shipping Cost'. The correlations of which provide insight into the relationships between variables. This information can then be leveraged in our upcoming LSTM and KNN modeling.

A notable observation from Figure 4.5 is a strong positive correlation between 'Order Price' and 'Purchase Quantity', at 0.74, suggesting that larger orders tend to have a greater number of items. This is fairly intuitive as more items ordered would most often contribute to a higher gross order amount. Similarly, 'Order Price' shows a moderately strong correlation with 'Shipping Cost', at 0.61, implying that higher order values may be associated with more expensive freight. Moreover, the correlations between 'Order Price' and other variables ('Employee ID', 'Shipping Method', 'Price Discount') are quite low. As they suggests that these variables may not have a direct, strong relationship with 'Order Price'. Albeit, it doesn't rule out their potential impact on the models. For instance, different employees might have different effects on the sales, or discounts might indirectly influence the order amount through their impact on the quantity ordered.

Similarly to the real dataset, the correlation values can be indicative of which features might contribute more to the prediction of 'Order Price' for the KNN models. At the same time, our LSTM models, able to handle sequential data, have the ability to leverage the correlations in a more complex way. Perhaps leading to better relative results as it learns how the variables evolve over time.

When analyzing the synthetic data, we need to consider that it might be designed to represent real-world business scenarios. Moreso, we anticipate that it is. That being said, as the LSTM and KNN models do not, via their nature, demand a specific distribution or correlation between the features, understanding these relationships can, as with the real data, be instrumental in feature selection and engineering, as well as interpreting the model results.

Finally, with the absence of high correlation between features and the target variable it suggests that any multicollinearity will not be a significant issue for our models in this dataset. Thus, hopefully allowing each feature to contribute to the predictive power of our models independently. Still, the significant correlations present will be important to consider when interpreting the empirical results of the models in later sections.

Figure 4.6 reveals the histogram of the order value variable 'Order Price' for the synthetic dataset. Interestingly, the distribution shown exhibits very similar trends to the real dataset counterpart. In short, it shows a heavily left-skewed distribution, indicating that the majority of the orders carry a relatively lower value. This again, follows the rationale that e-commerce, retail or sales-related industries often are characterized by a high volume of lower-priced products and a lower volume of more expensive products and orders.

However, compared to the real dataset, it is worth noting that the concentration of orders in the synthetic data appears to be less pronounced towards the first bar of the histogram. The second and subsequent bars display higher counts, which might indicate a broader spread of order values in the synthetic dataset. While lower-value orders are still dominating, the synthetic dataset seems to contain a more even distribution of order values. This pattern observed in the synthetic data may impact how our LSTM and KNN models learn from the data. Both models

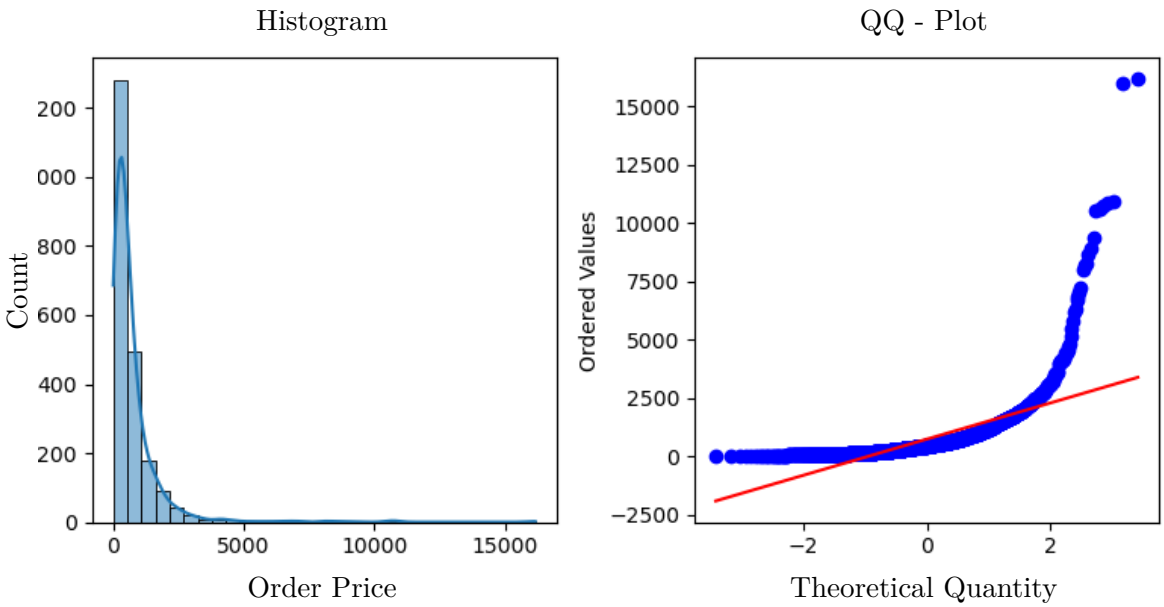

Figure 4.6: Data distribution target variable synthetic data

may need to be robust to the wide range of order values and not overly sensitive to the more frequent lower-value orders.

The second plot in Figure 4.6 displays the Q-Q plot of the synthetic dataset. Similarly to the real data, the Q-Q plot indicates a convex pattern, suggesting that the 'Order Price' variable also deviates significantly from normal distribution in the synthetic dataset. Similar to our approach with the real data, applying transformations such as power transformation could be considered to normalize the distribution and potentially improve the performance of our predictive models. The distribution of the synthetic data is the product of the trends that the manufacturer of data decides to implement, and so because the synthetic data attempts to replicate a real business scenario, it is not surprising that we find similar characteristics for all of the plots.

In conclusion, the exploratory data analysis of the synthetic dataset provides useful initial insights of the underlying data, which will help in defining the tuning for the LSTM and KNN models. The distinct features of the synthetic dataset, particularly the distribution of 'Order Price', provide valuable context for model selection, data preprocessing, and interpretation of subsequent model results.

4.3 Empirical Results and Discussion

In this section it is important to note that we can only infer comparisons between models and methods using the MAPE, as the other metrics can only be used to compare different renditions of the same model setup and preprocessing.

For instance, the feature scaling of the power transformer transforms the variables to mimic a standard normal distribution, as such the values will be in the form of

standard deviations. While the robust transformer has similar characteristics it also flattens outliers.

4.3.1 Real Data

Model	Scaler	Best Parameters	MAE	MSE	MAPE
KNN	Power Transformer	Neighbors: 1, Manhattan distance, uniform	1.0342	1.5119	3.9604
KNN	Robust Scalar	Neighbors: 2, Manhattan distance, uniform	1.2678	18.9399	21357469592348.895
RNN	Power Transformer	5 days timewindow	0.1878	0.2039	431611839993856.26
RNN	Robust Scalar	5 days timewindow	0.4383	5.0089	645015631395553.8

Table 4.3: Model Comparison: Real data

The empirical investigation into our real dataset, sourced from the Iranian company Digikala, reveals several noteworthy patterns and observations.

Regarding our first research question on estimation accuracy, the KNN model appears more precise when considering MAE and MSE measures. However, its accuracy in the context of MAPE diverges sharply when using the Robust Scaler. The contrast between the MAPEs of the KNN model under different scalers is substantial, raising the question of how scaling affects the model's performance with real-world data.

Addressing our second research question on robustness to outliers and noisy data, the results indicate that both the KNN and RNN models appear sensitive to noise in the data, reflected in the remarkably high MAPE values, particularly when utilizing the Robust Scaler.

In terms of factors influencing model performance (Research Question 3), the optimal parameters for the KNN model involved a surprisingly low number of neighbors (1 and 2 for Power Transformer and Robust Scalar respectively).

However, given the astonishingly high MAPE values for all models, we should approach these results with a degree of caution. The models may not have generalized well to the data due to its complex, non-linear, and possibly non-stationary nature. The data's heavy tilt towards lower-order values, as indicated in our exploratory analysis, might also have influenced these results.

In sum, while the KNN model seems to generally outperform the RNN model in terms of MAE and MSE, both models face substantial challenges in terms of MAPE. This raises questions about the suitability of these models for real-world, non-linear, and noisy datasets like the one from Digikala. It also stresses the importance of understanding the nature of the data and aligning it with suitable model architectures and parameters.

4.3.2 Synthetic Data

Model	Scaler	Best Parameters	MAE	MSE	MAPE
KNN	Power Transformer	Neighbors: 30, Euclidean distance, uniform	0.5584	0.4781	1.8557
KNN	Robust Scalar	Neighbors: 30, Manhattan distance, uniform	0.5193	1.7008	1.2461
RNN	Power Transformer	5 days timewindow	0.2477	0.3031	42433729332133.93
RNN	Robust Scalar	5 days timewindow	0.2944	0.8315	137463707516919.83

Table 4.4: Model Comparison: Synthetic data

Analyzing the results from our empirical investigation on the synthetic dataset, we derive several general observations related to our research questions.

In terms of estimation accuracy (Research Question 1), the KNN model, in both the Power Transformer and Robust Scaler instances, found substantially lower, albeit not good Mean Absolute Percentage Error (MAPE) values compared to the RNN model. This suggests that the KNN model, irrespective of the scaling method used, provided more accurate estimations than the RNN model for this specific synthetic dataset. It is worth noting that the MAE and MSE values for the RNN model were lower than those for the KNN model, but the extremely high MAPE values for RNN suggest some issues with percentage errors in estimation. This finding is particularly intriguing given the dataset's skewness towards lower order values, as revealed by the histogram in the exploratory data analysis. The KNN's ability to effectively handle a larger volume of lower-value data points might have contributed to its superior performance in terms of MAPE.

Investigating the robustness to outliers and noisy data (Research Question 2), the models' performance seems to hint towards the KNN model's relative resilience. The Robust Scaler, renowned for its robustness to outliers, does surprisingly not contribute to a better MAPE performance for the RNN model, despite the substantial outliers in our dataset. Generally a cause for this could be due to feature scaling being performed after the sequencing of the data, which has the risk of distrupting the scaling or removing important cross section information from the data. However, this study's preprocessing step performed the feature scaling prior to sequencing the data. As such, we can rule out this possibility as a potential cause in this case. Conversely, the KNN model using the Robust Scaler seems to handle the transformed data better, thereby aligning with our earlier observations from the Q-Q plot analysis, where the synthetic dataset showed deviations from the normal distribution.

Addressing the factors influencing the models' performance (Research Question 3), we can conjecture that the choice of parameters, data characteristics, and model parameters might play an instrumental role. In particular, the KNN model's optimal configuration with 30 neighbors and Manhattan distance indicates a promising avenue for further exploration. Moreover, the choice between Euclidean and Manhattan distance metrics, in relation to the effect of each on the model's sensitivity to changes in different dimensions of the data, could warrant additional investigation. A potential venue for future research could be to expand the initial features for the

model to learn from, as we saw in the exploratory data analysis there are few cases where the features are correlated with the 'Order Price', a potential venue could for instance be to feature engineer customer categories based on purchasing behaviour. However, for the purposes of keeping the data as close to the database format as possible, we did not attempt these measures.

Nonetheless, the synthetic nature of the dataset demands caution when intepreting these results. Synthetic data often presents regularity and less noise than real-world data, potentially influencing the model's performance and can generally risk creating biased models. Furthermore, the substantial MAPE values for the RNN models imply a possible sensitivity to extreme values in the prediction, suggesting a need for more granular investigation.

To conclude, our empirical analysis of the synthetic dataset indicates a generally superior performance of the KNN model in terms of accuracy and robustness, influenced by an interplay of various factors, including the choice of scaling method and model parameters. It encourages further studies with real-world datasets to validate these findings and evaluate their general applicability.

5

Conclusion

This study has explored the performance of two models, KNN and a LSTM-RNN, and their ability to estimate missing data in a real and synthetic firm database. With the primary focus to determine a model which fits best for missing data imputations. The study considers factors such as estimation accuracy, robustness to outliers and noisy data, and other data-specific characteristics influencing model performance.

Within the exploratory data analysis, the study found data trends and characteristics in both the synthetic and real datasets. Noteworthy, the study identified a strong skewness toward lower values in both datasets. Additionally, data inspection highlighted non-linear relationships in the data. These factors combined provide a valuable perspective on the potential challenges that the subsequent machine learning models could face and later provided vital information needed to interpret the results of each model. Following this, both models were evaluated based on the Mean Absolute Error, Mean Squared Error, and Mean Absolute Percentage Error metrics. The MAPE being the centered metric for the basis of model comparison due to the scale-independent nature of the metric. The study observed contrasting results between each respective model configuration: While KNN generally provided more accurate results in terms of MAE and MSE, the MAPE values were extraordinarily high for both KNN and RNN models. Particularly, the models' performances diverged significantly when different scaling methods were used, which underscored the importance of preprocessing steps in dealing with real-world data.

Regarding the high MAPE values; Firstly, they could indicate an overfitting problem, where the models are too finely tuned to the training data, and hence perform poorly on unseen data. Another plausible explanation could be the influence of outliers and extreme values, as well as the heavy tilt towards lower order values in the data. Lastly, it could be the use of the MAPE as a measure of the accuracy, with the stronng prevalence of zero centered values in the target variable there is a risk of extreme output values. As such, this study advises future researchers to consider these potential issues while designing and implementing similar prediction models. In relation to previous research and studies using RNN and KNN models to estimate missing data, we do not find promising results regarding the performance of these models. The reason for this is likely found in the aforementioned causes to the high MAPE values.

Finally the three research questions posed; we cannot from this research conclude

that any of the two analyzed models provide any potential in the task of estimating missing data in the context of a firm database. However, we find that their performance can be significantly influenced by the nature of the data and the choice of preprocessing techniques. Therefore, understanding the data at hand and carefully considering appropriate transformations and parameters is of great importance.

Both KNN and RNN models show indications of being sensitive to outliers and noisy data. However, the degree of sensitivity varied depending on the scaler used. Finally, the most influential factors in determining the performance of these methods were the scaler used, and the underlying distribution of the data.

In short, this study highlights challenges and considerations to be aware of when applying KNN and RNN models to business data. Our findings show that a nuanced understanding of the data and careful consideration of model and parameter choices is vital. Future work could explore different preprocessing techniques, other machine learning models, or a wider range of model parameters and features to enhance prediction accuracy and robustness.

Bibliography

[1] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3), 175-185.

[2] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bengio, Y., & Courville, A. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[3] García-Laencina, P. J., Sancho-Gómez, J. L., & Figueiras-Vidal, A. R. (2010). Pattern classification with missing data: A review. *Neural Computing and Applications*, 19(2), 263-282.

[4] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

[5] Little, R. J., & Rubin, D. B. (2002). *Statistical Analysis with Missing Data* (2nd ed.). John Wiley & Sons.

[6] Lipton, Z. C., Kale, D. C., Elkan, C., & Wetzel, R. (2016). Learning to diagnose with LSTM recurrent neural networks. *arXiv preprint arXiv:1511.03677*.

[7] Yoon, J., Zame, W. R., & van der Schaar, M. (2018). Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 65(5), 980-990.

[8] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[9] Hochreiter, S., & Schmidhuber,J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

[10] Fang, C., & Wang, C. (2020). Time Series Data Imputation: A Survey on Deep Learning Approaches. *arXiv*. doi: 10.48550/arXiv.2011.11347.

[11] Buczak, A. L., Babin, S. M., Moniz, L. J. (2016). Data-driven approach for creating synthetic electronic medical records. *BMC Medical Informatics and Decision Making*, 16(1), 1-12.

[12] Drechsler, J. (2011). *Synthetic Datasets for Statistical Disclosure Control: Theory and Implementation*. Springer Science Business Media.

[13] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2672-2680.

[14] Moritz, S., Sardá, A., Bartz-Beielstein, T., Zaefferer, M., Stork, J. (2017). Comparison of different methods for univariate time series imputation in R. *arXiv preprint arXiv:1510.03924*.

[15] Nowok, B., Raab, G. M., Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, 74(11), 1-26.

[16] Raab, G. M., Nowok, B., Dibben, C. (2017). Practical Data Synthesis for Large Samples. *Journal of Privacy and Confidentiality*, 7(3), 67-97.

[17] Rubin, D. B. (1981). The Bayesian bootstrap. *The Annals of Statistics*, 9(1), 130-134.

[18] Snoke, J., Raab, G. M., Nowok, B., Dibben, C., Slavkovic, A. (2018). General and specific utility measures for synthetic data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 181(3), 663-688.

A

Appendix 1

A.1 Experimental Findings: Real Data

```
Best parameters:  {'n_neighbors': 1, 'p': 1, 'weights': 'uniform'}
Mean Absolute Error on test data:  1.0342229778506373
Mean Squared Error on test data:  1.5119308334859625
Mean Absolute Percentage Error on test data:  3.9604098643893964
```

Figure A.1: KNN, PowerTransformer

```
Best parameters:  {'n_neighbors': 2, 'p': 1, 'weights': 'uniform'}
Mean Absolute Error on test data:  1.267788801688666
Mean Squared Error on test data:  18.93994071694983
Mean Absolute Percentage Error on test data:  21357469592348.895
```

Figure A.2: KNN, RobustScalar

```
2/2 [==============================] - 1s 13ms/step
Mean Absolute Error on test data:  0.1878197035691235
Mean Squared Error on test data:  0.20391972183145 54
Mean Absolute Percentage Error on test data:  43161183993856.26
```

Figure A.3: RNN, PowerTransformer 5 days timewindow

```
2/2 [==============================] - 1s 16ms/step
Mean Absolute Error on test data:  0.43828005598582875
Mean Squared Error on test data:  5.008918493792763
Mean Absolute Percentage Error on test data:  645015631395553.8
```

Figure A.4: RNN, RobustScalar 5 days timewindow

A.2 Experimental Findings: Synthetic Data

```
Best parameters:  {'n_neighbors': 30, 'p': 2, 'weights': 'uniform'}
Mean Absolute Error on test data:  0.558439533267503
Mean Squared Error on test data:  0.47809027045747826
Mean Absolute Percentage Error on test data:  1.8557199353685123
```

Figure A.5: KNN, PowerTransformer

```
Best parameters:  {'n_neighbors': 30, 'p': 1, 'weights': 'uniform'}
Mean Absolute Error on test data:  0.5193333468777276
Mean Squared Error on test data:  1.700798403473969
Mean Absolute Percentage Error on test data:  1.2460889337734542
```

Figure A.6: KNN, RobustScalar

```
5/5 [==============================] - 1s 44ms/step
Mean Absolute Error on test data:  0.24774039126332212
Mean Squared Error on test data:  0.30312156418053593
Mean Absolute Percentage Error on test data:  42433729332133.93
```

Figure A.7: RNN, PowerTransformer 5 days timewindow

```
5/5 [==============================] - 1s 21ms/step
Mean Absolute Error on test data:  0.29444754556124614
Mean Squared Error on test data:  0.8315275924712682
Mean Absolute Percentage Error on test data:  1374637075169 19.83
```

Figure A.8: RNN, RobustScalar 5 days timewindow

A.3 Winsorizing

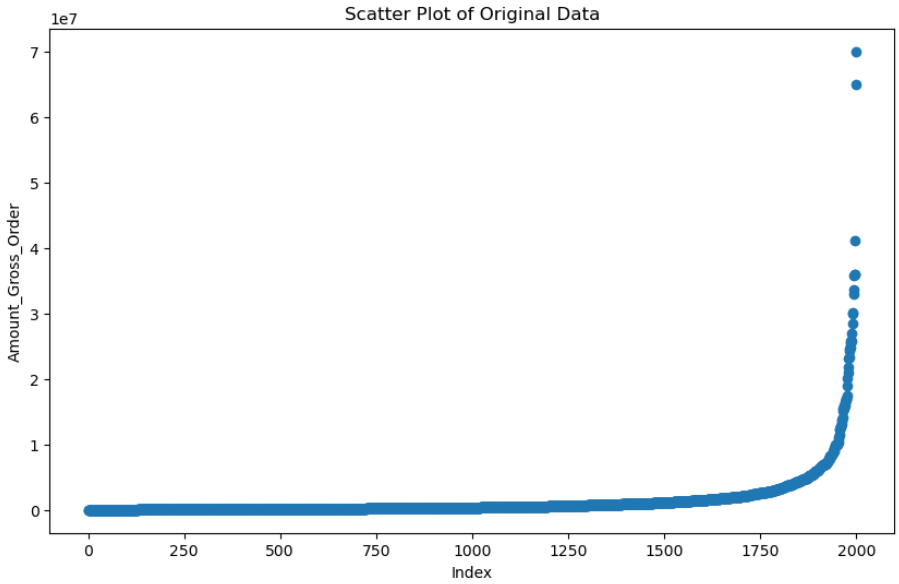

Figure A.9: Unwinsorized Data Distribution

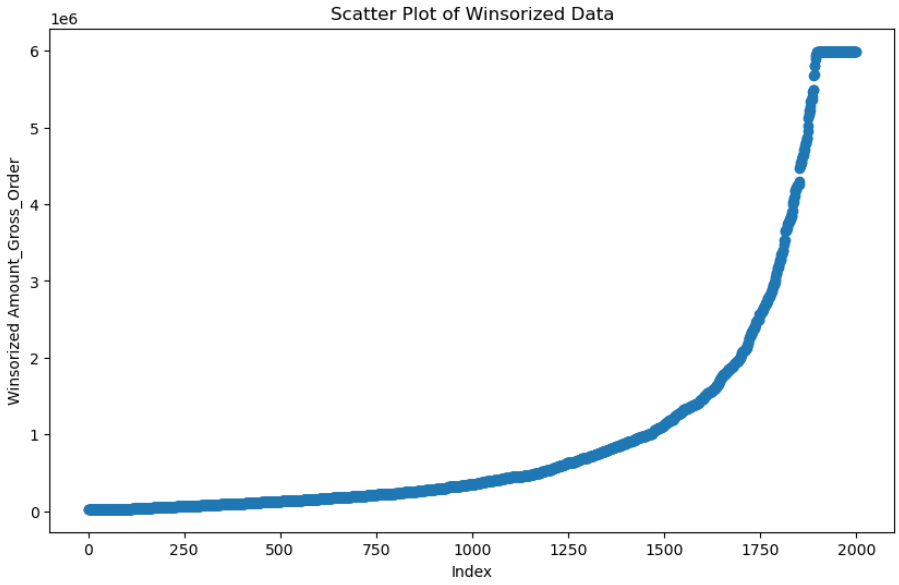

Figure A.10: Winsorized Data Distribution

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY