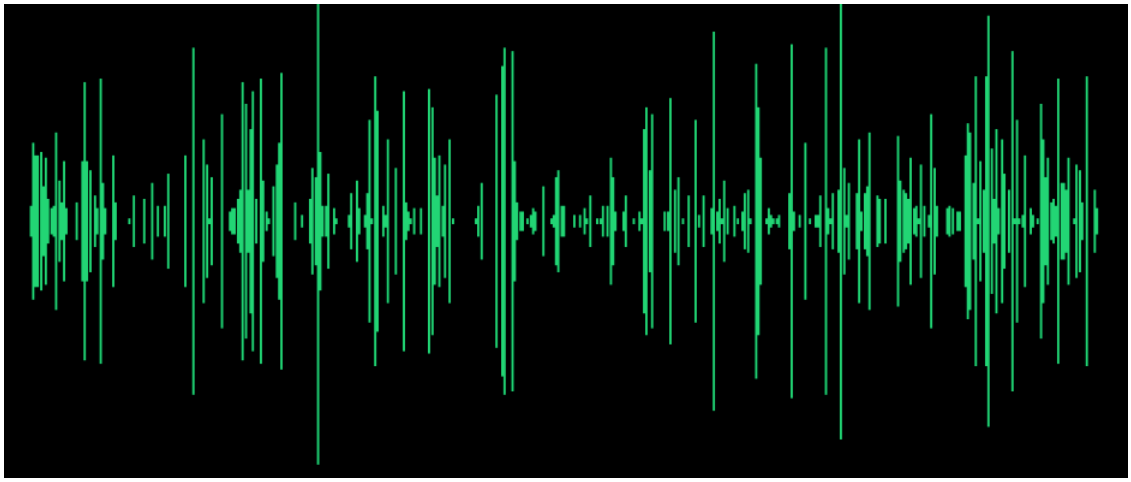

Predictive Caching

Predict the content a user is most likely to download using mobile data so as to pre-fetch it when the user's device is connected to a high bandwidth network

Master's thesis in Computer science and engineering

Nickey Lizbat Lawrence
Renjith Sebastian

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

Predictive Caching

Predict the content a user is most likely to download using mobile data so as to pre-fetch it when the user's device is connected to a high bandwidth network

Nickey Lizbat Lawrence
Renjith Sebastian

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

Predictive Caching
Predict the content a user is most likely to download using mobile data so as to pre-fetch it when the user's device is connected to a high bandwidth network

Supervisor: Moa Johansson, Department of Computer Science and Engineering
Advisors: Erik Carlsson, Emil Pedersen, Pierre Wessman, Company
Examiner: Carl-Johan Seger, Department of Computer Science and Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Generic music track image

Typeset in LaTeX
Gothenburg, Sweden 2019

Predictive Caching
Predict the content a user is most likely to download using mobile data so as to pre-fetch it when the user's device is connected to a high bandwidth network

Nickey Lizbat Lawrence
Renjith Sebastian
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Though digitization has revolutionized the entertainment industry, streaming services like Netflix, Spotify, etc. are the ones who made the content available to the users through hand-held devices. These services require an active internet connection to deliver the requested content to the user device, consuming the expensive mobile data subscriptions of the user. The aim of the thesis project is to optimize the mobile data usage by predicting the content a user is most likely to download so that it can be pre-fetched when the user's device is connected to high bandwidth, less-expensive network. Different use cases were considered to identify the potential candidates that a user is most likely to download through mobile data subscription. First, users are highly probable to download the personalized content recommended by these services. Hence, the user behavior on personalized content was modeled using a Logistic Regression algorithm as a generic baseline approach. Second, the users tend to use multiple devices to stream content and it is very likely that they play the same content from different devices. This has a strong pre-cache potential in the context that contents viewed/listened to in one device could be used to predict the possible streaming behavior in the user's other devices. Third, the users prefer to play contents from different playlists provided by streaming services. The third use case exploited the user behavior on playlists to predict the contents a user is likely to download in future. We employed a Gradient Boosting algorithm to model the device sync and playlist use cases. The results were evaluated using a generic evaluation metric defined solely for the purpose, and different use cases were compared. The device sync model predicted 15% of the potential savings that were identified through data analysis, whereas the playlist model predicted 30%.

Keywords: Computer science, engineering, thesis, machine learning, predictive caching, user behavior, xgboost, logistic regression.

Acknowledgements

We sincerely thank each of our company advisors, Erik Carlsson, Emil Pedersen, and Pierre Wessman, for supporting, guiding and encouraging us to explore independently. We would like to thank and express our gratitude to our supervisor Moa Johansson at the Chalmers University of Technology, for the proper guidance and help. Furthermore, we would like to thank the entire team at the company we worked with, who have always been great to us.

Nickey Lizbat Lawrence, Renjith Sebastian, Gothenburg, June 2019

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Context . . . 2
1.2 Goals and Challenges . . . 2
1.3 Literature Survey . . . 3

2 Background **5**
2.1 Machine Learning . . . 5
2.1.1 Supervised Learning . . . 5
2.1.2 Feature engineering . . . 5
2.2 Feature Selection . . . 6
2.2.1 Univariate Analysis . . . 6
2.2.2 Bivariate Analysis . . . 6
2.2.3 Recursive Feature Elimination and Cross Validation (RFECV) 6
2.2.4 Feature Scaling . . . 7
2.3 Logistic Regression . . . 7
2.4 Ensembles . . . 8
2.4.1 Gradient Boosting . . . 8
2.4.2 Feature Importance Scoring . . . 8
2.5 Grid Search Algorithm . . . 8
2.6 Regularization . . . 9
2.7 Evaluation Metrics . . . 9
2.7.1 Accuracy, Precision and Recall . . . 9
2.7.2 Receiver Operating Characteristic (ROC) curve . . . 11
2.7.2.1 Area Under the Curve (AUC) . . . 11

3 Exploratory Data Analysis **12**
3.1 Device Sync . . . 12
3.2 Playlists . . . 14
3.2.1 User Consistency . . . 17

4 Methods **18**
4.1 Generic Modeling Approach . . . 18
4.2 Baseline Target . . . 19

4.2.1 Rule-Based Heuristic . . . 19
4.2.2 Machine Learning Model . . . 20
4.3 Use Cases . . . 21
4.4 Device Sync . . . 22
4.4.1 Dataset Preparation and Modeling . . . 22
4.4.2 Feature Engineering . . . 24
4.5 Playlists . . . 25
4.5.1 Approaches . . . 25
4.5.2 Stage 1 : Playlist Prediction . . . 25
4.5.2.1 Dataset Preparation and Modeling . . . 26
4.5.2.2 Feature Engineering . . . 27
4.5.3 Stage 2 : Content Prediction . . . 27
4.5.3.1 Dataset Preparation and Modeling . . . 27
4.5.3.2 Feature Engineering . . . 28

5 Evaluation and Results **29**
5.1 True Byte Rate (TBR) . . . 29
5.2 Baseline Model . . . 29
5.2.1 Rule-based Heuristic . . . 29
5.2.2 Machine Learning Model . . . 30
5.2.3 Threshold calibration . . . 31
5.3 Device Sync Model . . . 32
5.3.1 Approach 1: All devices . . . 32
5.3.1.1 Threshold calibration . . . 33
5.3.1.2 Observations . . . 35
5.3.2 Approach 2: Fixed Devices . . . 35
5.3.2.1 Threshold calibration . . . 36
5.3.2.2 Observations . . . 37
5.3.3 Summary . . . 37
5.4 Playlist Model . . . 37
5.4.1 Stage 1 : Playlist Prediction . . . 38
5.4.2 Stage 2 : Content Prediction . . . 39
5.4.2.1 Threshold calibration . . . 40
5.4.2.2 Observations . . . 41
5.4.3 Summary . . . 42

6 Related Work **43**

7 Conclusion **45**
7.1 Conclusion . . . 45
7.2 Future Work . . . 45

Bibliography **47**

List of Figures

2.1 Sigmoid function 7
2.2 Accuracy, Precision and Recall 10
2.3 Confusion Matrix 10
2.4 ROC curve 11

3.1 Potential savings using Sync Download compared to Total Download 13
3.2 Percentage saved using Sync Download compared to Total Download (Weekly Analysis) 13
3.3 Potential savings using Sync Download compared to Total Download 14
3.4 Percentage saved using Sync Download compared to Total Download 14
3.5 Absolute carrier downloads for top five playlists 15
3.6 Percentage of carrier downloads for top five playlists 15
3.7 Potential savings for top five playlists 16
3.8 Percentage of potential savings for top five playlists 17

4.1 Steps in Generic Modeling Approach 18
4.2 The average trend value for a period of four weeks was computed and summed up with total number of plays in week 4 to compute the total number of plays in week 5 20
4.3 Dataset Preparation for Baseline Machine Learning Model 20
4.4 Dataset Preparation - Weekly Target 22
4.5 Dataset Preparation - Monthly Target 23
4.6 Two-stage Prediction Model for Playlists Use Case 25
4.7 Dataset Preparation for Stage 1: Playlist Prediction 26

5.1 True Byte Rate 30
5.2 Bytes saved Vs Bytes wasted 30
5.3 Percentage of users who actually listened to the pre-fetched tracks . . 31
5.4 Total MB saved at TBR = 0.5 32
5.5 Approach 1 : True Byte Rate for Weekly Target 33
5.6 Approach 1 : True Byte Rate for Monthly Target 33
5.7 Approach 1 : Weekly Target - Precision Recall Curve for MB saved . 34
5.8 Approach 1 : Monthly Target - Precision Recall Curve for MB saved 34
5.9 Approach 2 : True Byte Rate for Weekly Target 35
5.10 Approach 2 : True Byte Rate for Monthly Target 36
5.11 Approach 2 : Weekly Target - Precision Recall Curve for MB saved . 36
5.12 Approach 2 : Monthly Target - Precision Recall Curve for MB saved 37

5.13 Stage 1 : TBR for a prediction period of one week 38
5.14 Stage 1 : TBR for a prediction period of one month 39
5.15 Stage 2 : TBR for a prediction period of one week 39
5.16 Stage 2 : TBR for a prediction period of one month 40
5.17 Stage 2 : Weekly Prediction - Precision Recall Curve for MB saved . 41
5.18 Stage 2 : Monthly Prediction - Precision Recall Curve for MB saved . 41

List of Tables

4.1 Features used in baseline machine learning models 21
4.2 Hyper-parameters for baseline LightGBM model 21
4.3 Hyper-parameters for baseline Logistic Regression model 21
4.4 Hyper-parameters for XGBoost model used for Device Sync use case . 24
4.5 Hyper-parameters for XGBoost model used for Playlists use case - Stage 1 Playlist Prediction 27
4.6 Hyper-Parameters for XGBoost model used for Playlists use case : Stage 2 Content Prediction 28

1
Introduction

The wave of Artificial Intelligence (AI) and Machine Learning (ML) have empowered market leaders in diverse domains with innovative ways to attract and retain their customers. Audio/video streaming services give access to millions of audio/video content from artists all over the world. The "on-demand" download technique adopted in these services enable users to view/listen to their favorites on their mobile device online upon request. However, it requires an active internet connection to download the content to the mobile device which consumes the expensive mobile data subscriptions of the customer. As per the Ericsson Quarter 4 Mobility Report [1] in 2018, the number of mobile subscriptions grew at 2 percent year-on-year and is nearly 7.9 billion. Additionally, the number of mobile broadband subscriptions grew at 15 percent year-on-year and is nearly 5.7 billion. These figures clearly indicate the magnitude of mobile subscribers and their potential usage of streaming services through mobile data.

Machine learning models could be used to optimize the mobile data usage by predicting what a user will play in order to pre-fetch it on the device before he/she actually hits play. For example, a user who has been consistently playing personalized content is highly likely to play more from these playlists. By exploiting the historical user behavior on the personalized content, we can formulate a prediction problem to maximize the model's likelihood of predicting what the user actually ended up playing. This thesis project used machine learning techniques to predict the content a user is most likely to download, based on historical user behavior. These predictions could then be used to pre-fetch the content when the user's device is connected to high bandwidth and less expensive network. This will, in turn, optimize mobile data usage and provide faster download, thus ensuring customer satisfaction.

In Chapter 2, the theoretical background of the underlying machine learning concepts that have been used throughout this project is presented. Exploratory Data Analysis(EDA) that was performed for different use case scenarios to identify potential pre-fetch candidates with maximum byte savings, with primary focus on use case complexity versus potential impact is detailed in Chapter 3. In Chapter 4, the different implementation steps performed are described in detail. Further, the evaluation strategy and results obtained for the different use cases and approaches are presented in Chapter 5. Moreover, previous work relevant to this field of study

[1] https://www.ericsson.com/assets/local/mobility-report/documents/2019/emr-q4-update-2018.pdf

is detailed in Chapter 6. We conclude by reflecting on the results obtained and relevant discussion on future work through Chapter 7.

1.1 Context

User Modeling and User-Adapted Interaction[2] is an interesting research field, related to human-machine interaction, for customizing systems to the user-specific needs. With the advent of machine learning, this has gained widespread attention leading to the development of several sophisticated systems providing excellent customer experience. These systems are personalizing our experience, telling us which products to buy (Amazon) [4], which movies to watch (Netflix) [1], which songs to listen to (Spotify) [3] etc.

The popular audio/video streaming services have effective recommender systems already in place, with suggestions tailored to the user's interests. Though it is likely that a user will select some content from the recommender system's prediction, it is quite unlikely that every user always does so. A naive approach would be to pre-fetch the top-rated content from the existing recommender system's output. This would consume the device memory for users who are not actively using the recommender system's predictions. In this project, we attempted to develop machine learning models by analyzing potential use cases. The hypothesis we put forth was that the predictions made by our model can perform better than the naive approach. It will add value to the business in the context that they could be used to optimize mobile data usage and memory for the customers.

1.2 Goals and Challenges

The goal was to save at least 10% of the total mobile data downloads of a subset of users by pre-fetching the model's predictions. Empirically, we compared the prediction results with the ground truth data and computed the total bytes from the carrier a user would have saved by pre-fetch. We defined a generic evaluation metric called 'True Byte Rate' (TBR) as the mega bytes (MB) saved out of the total MB pre-fetched from the model predictions, which was used to compare the different models.

The major challenges were as follows:

1. It was very unlikely that the model developed would have 100% precision and was expected to predict a reasonable number of false positives. Under the assumption that the model predictions would be pre-fetched to the user's device without any heuristics, the bytes wastage induced by false positives would consume the device storage unnecessarily. On the contrary, a minimum number of false positives would be required as eviction candidates when different pre-fetch models are used simultaneously, so as to account for the reactive caching policy, Least Recently Used (LRU) [20], used by these services.

[2] http://www.umuai.org

2. Stochastic user behavior, meaning (a) not all users were using the personalized content, and (b) not all users were consistently using the streaming service, affected the generalization of the prediction model. This imposed the challenge of collecting a good quality dataset. After collecting data, the appropriate feature pre-processing steps were performed depending on the architecture of the specific machine learning algorithm used.
3. The fact that one needs to consider ethics [17, 18] and risk questions [16, 19] contributes greatly to the difficulty of creating machine learning models. While the system recommending a content the user might dislike would not produce any devastating consequences, that's not the case while pre-fetching that content on to the user device. The risk and ethical questions that were considered are as follows.
   (a) Pre-fetching data on to the user's device without his/her consent.
   (b) Clogging the user's device memory with downloaded content.
   (c) Deciding a threshold on the error in predictions that is acceptable by the business for this scenario.
   (d) Influence of socially sensitive data (gender, age etc.) on the model's predictions.
4. The idea is to pre-cache contents when the user's device is connected to a high-bandwidth network. The term high-bandwidth could be deceiving from a streaming service's point of view in that, a user might be connected to a mobile data hotspot even though the system interprets it as WiFi.
5. Seasonal data trends could affect the prediction power of the model. For instance, users tend to listen to festive songs during Christmas.

1.3 Literature Survey

Creating highly personalised experience for each user is crucial to the success of all audio/video streaming services. Most of the successful automated recommendation systems[1, 3, 4] focus on both the creators and listeners, through the collaborative filtering approach, to capture information based on the daily user-item activity data, so as to recommend fresh content to the users. In 2018, the music-streaming company, Spotify, hosted the RecSys Challenge on Automatic Playlist Continuation[21], together with researchers from JKU Linz and UMass Amherst, to predict content that would complete a given playlist, which was again similar to recommendations. Researches around personalizing Spotify's home page, is ongoing so as to ensure each user gets a targeted list of playlists. On the contrary, this project aimed to identify content that a user will download through mobile data rather than recommend content that a user might like, and hence the primary focus was on pre-fetching or caching the content that the user is most likely to download using mobile data. To the best of our knowledge, no similar work that primarily focused on predictive caching of audio/video content has been attempted earlier, even though there have been studies related to webpage pre-caching and pre-fetching mechanisms for I/O Storage Systems. Predictive pre-fetching mechanisms for the latter scenario predict the application's future I/O data accesses by building a history of the I/O

requests wherein the goal is to make the data available in memory even before the request. This approach intends to boost the performance of read/write operations by saving the instructions in faster cache memory compared to slower main memory. Several predictive pre-fetching models and algorithms including machine learning approaches, mostly neural networks, have been studied extensively. On the contrary, this projects aims to optimise the mobile data consumption rather than the application performance.

2
Background

This chapter serves as a theoretical background for the chosen methods that have proven to be suitable for the project under consideration. The various aspects of system development based on machine learning include: getting the data set, cleaning the raw data, processing and selecting the right features, selecting and tuning a machine learning model and then finally, evaluating the model's performance. We will present these topics briefly in the coming sections.

2.1 Machine Learning

Machine learning is a sub-field of artificial intelligence that enables a computer to identify patterns in observed data(the training data) and use them to make predictions on unseen data(the test data) without being explicitly programmed. It provides the machine with an ability to learn from historical data.

There are two main categories of machine learning depending on the type and amount of supervision involved - supervised learning and unsupervised learning. In supervised learning, the system is trained with labeled data or data with the ground truth whereas in unsupervised, the system tries to find an underlying structure in the data without labels. Now we will briefly introduce supervised learning and its types.

2.1.1 Supervised Learning

A majority of the practical machine learning problems rely on supervised learning, which can be thought of as a teacher supervising the learning process. Given a set of inputs and outputs, the goal of the supervised learning algorithm is to generalize or learn a mapping from the inputs to the outputs, such that it can predict the output for new unseen data. The output could be a class label (in classification) or a real number (in regression), depending on the problem being solved. In the former scenario, we need at least two classes for the classification task, in which case it is called binary classification. Tasks with more than two output classes fall under multi-class classification.

2.1.2 Feature engineering

A feature is an attribute, property or a measurable unit that acts as input to a machine learning algorithm. Typically, there are two broad types of features - raw

features and derived features. Features that are obtained directly from the raw data set without any manipulation are termed raw features. Derived features are the ones extracted or derived from the raw data set.

The process of using domain knowledge to extract meaningful information or features from a raw data set is called feature engineering and is crucial to the success of any machine learning model. More the features, better the chances of a good model. Though it is highly important to create a lot of features, choosing the relevant features is an art in applied machine learning. Even if some of the chosen features are irrelevant, one cannot afford to miss the relevant ones as they are highly influential in the prediction results. Various feature selection techniques could then be employed to filter out useful features with respect to the problem domain.

2.2 Feature Selection

Feature selection is the process of finding right features, essentially aimed at improving the prediction power of the model. In addition to developing an accurate prediction model, feature selection plays a crucial role in reducing model complexity and preventing over fitting of the model. Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may decrease the accuracy of the model. Several feature selection techniques are prevalent in industry, where three techniques used in the project are discussed in the below section.

2.2.1 Univariate Analysis

Univariate analysis is the simplest statistical test for analyzing data, used to select the features that have the strongest relationship with the output target variable. It deals with one variable, hence the name, and does not consider relationships between variables.

2.2.2 Bivariate Analysis

Bivariate Analysis deals with two variables simultaneously unlike univariate analysis. The purpose is to analyze the relationship between two variables, whether there exists an association and the strength of this association, or whether there are differences and the importance of these differences.

2.2.3 Recursive Feature Elimination and Cross Validation (RFECV)

Recursive Feature Elimination (RFE) is a feature selection technique by which features are removed recursively by fitting a machine learning model. First, the model is trained on the initial set of features and the importance of each feature is obtained. Then, the least important features are pruned from current set of features based on step size configured. This is recursively repeated on the remaining set until the

optimal number of features to select is eventually reached. By recursively eliminating features in each loop, RFE attempts to eliminate dependencies and collinearity existing in the model, if any. Moreover, a technique called cross-validation is used to make a fixed number of data partitions and RFE is performed on each partition to find the optimal number of features.

2.2.4 Feature Scaling

Feature Scaling is a data pre-processing technique which is generally used to improve the performance of machine learning models. Different feature scaling techniques available in scikit-learn are StandardScaler, MinMaxScaler, RobustScaler and Normalizer. In this project, MinMaxScalar was used which transforms the given features by scaling each feature value to a specified range, for example, between zero and one. It works by subtracting the minimum value of the given feature and dividing the same by the range, which is the difference between the maximum and minimum values.

2.3 Logistic Regression

Linear classifiers select the outputs based on a scoring function, given by,

$$score = w * x,$$

where w is the weight vector and x is the input feature. It is difficult to interpret the confidence of such classifier models directly. Logistic Regression is a linear classifier model for binary classification task that provides a probabilistic output. It closely aligns with linear regression but differs in the sense that here the response variable is categorical.

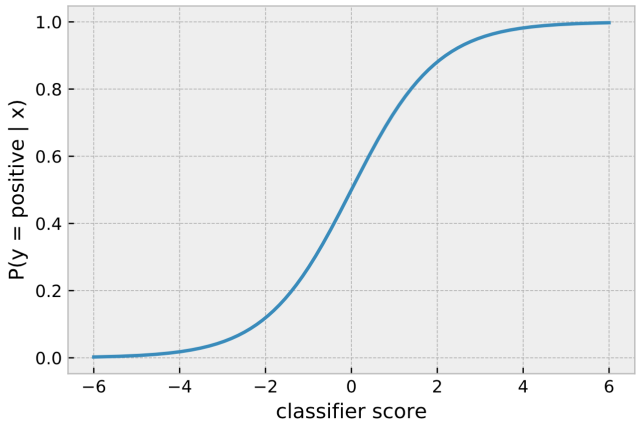

Figure 2.1: Sigmoid function

Logistic Regression predicts the probability of a class using the logistic or sigmoid function, and hence the name. The sigmoid function is an S-shaped curve that can

take any real-valued number and map it to a value between 0 and 1. If the curve goes to positive infinity, the prediction value is 1, and if the curve goes to negative infinity, the prediction will be 0, as could be seen in Figure 2.1.

Mathematically, the sigmoid function is given by,

$$P(x) = \frac{1}{1 + e^{-x}}$$

2.4 Ensembles

An ensemble is a collection of models that give a single final prediction. The intuition behind using these models is that multiple predictors might give a better result than a single predictor trying to predict the target.

Ensembles are of two types - bagging and boosting. Bagging is the technique where many independent predictors are combined using model averaging techniques. Boosting, on the other hand, is a technique where the models are combined sequentially and the new predictors learn from the mistakes of the previous predictors.

2.4.1 Gradient Boosting

Gradient Boosting is an example of a boosting algorithm that produces a single prediction model in the form of an ensemble of weak prediction models, typically decision trees. The algorithm makes use of a concept called residual, which is the difference between the current approximation and the known ground truth value. A weak model is then trained to map the features to the residuals considering the fact that adding such a residual on top of an existing model's prediction will take it closer to the ground truth value. The overall prediction is thus improved upon by adding more such residuals from different sequential predictors. For our purpose, LightGBM[23] and XGBoost[22] implementations were used.

2.4.2 Feature Importance Scoring

Feature importance provides a score indicating how useful each feature was in creating the specific model. Ensembles like gradient boosting can provide feature importance estimates from a trained model. These scores, obtained from the 'feature_importances_' variable of the trained model, are used for feature selection.

2.5 Grid Search Algorithm

A parameter whose value is set before the training process begins is termed a hyperparameter. The training algorithm learns the parameters from the data, given these hyperparameters. Hence, a hyperparameter is considered an external characteristic of the model whereas a parameter is its internal characteristic which can be estimated from the dataset. Upon training multiple machine learning models on

the training dataset, the one with the best performance needs to be selected. Performance evaluation and model selection are crucial to building successful machine learning models. Evaluating the performance of a model is affected by several factors, with the choice of hyperparameters being the most important one. Considering the fact that one algorithm might perform better than the other with different sets of hyperparameters, it is not wise to randomly select their values. The Grid Search algorithm could be used to automatically select the optimal hyperparameters for a machine learning algorithm which results in the most accurate predictions. Using its scoring parameter, the metric to evaluate the model on, for example recall, could be specified. The Grid Search algorithm generates a model for every possible combination of the specified hyperparameters and evaluates each model based on the scoring parameter provided.

2.6 Regularization

When a machine learning model learns the noise in the training data, it is said to overfit the data and loses its generalization power. Regularization is a technique used to solve this overfitting problem in machine learning models, whereby we penalize the loss function by adding a multiple of Lasso (L1) or Ridge (L2) norm to the weight vector. On adding this penalty term, called the regularization term, the model is simplified so that it doesn't fit to the noise in the training data. L1 and L2 are the most common regularization techniques. The general cost function now takes the below form,

$$\text{Cost function} = \text{Loss} + \text{Regularization term}$$

LASSO (Least Absolute Shrinkage and Selection Operator) or L1 is a regularization technique which could also be considered as a powerful feature selection technique. This is because, in L1, we penalize the absolute value of the weights and these weights may be reduced to zero. L2 regularization is also known as ridge regression(Tikhonov regularization), which adds the sum of the squares of all the feature weights as the penalty term. L2 is also known as weight decay since it drives the weight values towards zero.

2.7 Evaluation Metrics

Choosing a meaningful evaluation protocol relevant to the problem domain is crucial in machine learning projects since we need to measure the quality of our model's predictions and compare various alternatives. We present the generally applicable evaluation metrics that are relevant for this thesis project.

2.7.1 Accuracy, Precision and Recall

In a binary classification task, we get four different outcomes, as listed below.

1. True Positive
   Data points predicted as positive that are actually positive
2. False Positive
   Data points predicted as positive that are actually negative
3. True Negative
   Data points predicted as negative that are actually negative
4. False Negative
   Data points predicted as negative that are actually positive

The most common evaluation metric, accuracy given in Figure 2.2, is defined as the percentage of total data points that are classified correctly. In a skewed data set where the positives are greatly outnumbered by the negatives, accuracy is not a good measure for evaluating the model's performance. For example, consider a model trained on a data set with 95 negatives and 5 positives. This model will have a tendency to predict all the test case samples as negative, which is a terrible but highly accurate model.

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

Figure 2.2: Accuracy, Precision and Recall

A confusion matrix, as represented in Figure 2.3, tells us the actual and predicted labels from a classification problem, whose cell values could be used to compute further meaningful metrics. Two of these that are relevant to our application are Recall and Precision, listed in Figure 2.2. While Recall is the ability to find all the relevant data points in a data set, Precision is the ability to identify only the relevant data points.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 2.3: Confusion Matrix

2.7.2 Receiver Operating Characteristic (ROC) curve

ROC curve is a visualization technique which plots the True Positive Rate(TPR) or Recall on the y-axis against the False Positive Rate(FPR) on the x-axis, as a function of the model's threshold above which the data point is classified as positive. FPR is the number of data points incorrectly classified as positive out of the total true negatives.

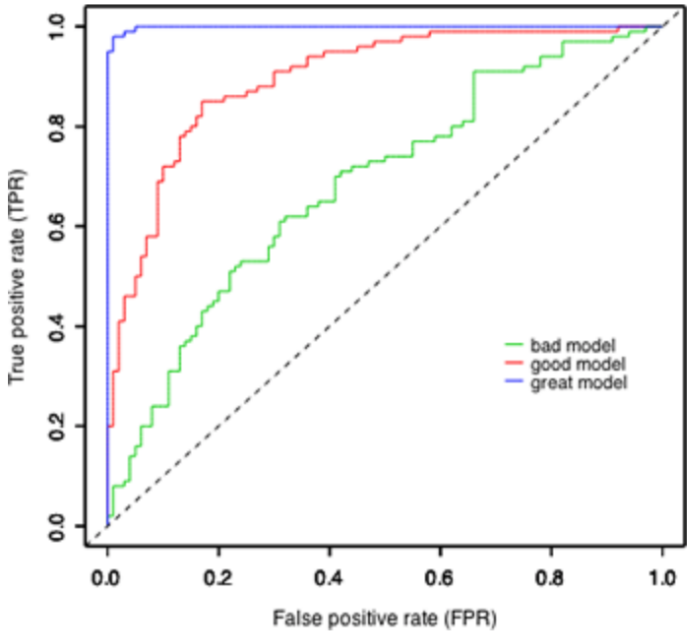

Figure 2.4: ROC curve

A threshold of 1.0 would point to the lower left of the graph shown in Figure 2.4[1], as there are no data points identified as positive at that threshold value. As the threshold is decreased, true positives and false positives increase. At a threshold of 0.0, we have all the data points identified as positive and refers to the upper right corner of the curve.

2.7.2.1 Area Under the Curve (AUC)

In order to quantify a model's ROC curve, a new metric was introduced. It is the total Area Under the Curve (AUC) with a value between 0 and 1. A random classifier will have an AUC score of 0.5. Higher the AUC score, better the classification performance. In the graph in Figure 2.4, the blue model with the highest AUC score is better compared to the rest.

[1]https://www.sakai.unc.edu

3

Exploratory Data Analysis

Exploratory Data Analysis (EDA) was conducted on different use cases to identify the potential scenarios that could bring maximum byte savings if modeled. The study was focused on identifying the use case complexity versus potential impact in order to prioritize the available opportunities.

User behavior on personalized playlists, the effect of cross-device syncing of contents for users with multiple devices, and the impact on predictive caching for users following any particular artist were some of the use cases that were considered for the purpose. The upper bound on potential mobile data savings were computed during the data analysis and was used to rank the use cases accordingly.

3.1 Device Sync

It is very common for a user to possess and use multiple devices to stream audio/video contents provided by services like YouTube, Spotify, etc. Cross-device syncing is an important signal for a predictive caching model since it is highly likely that the same content will be played on multiple devices of a user. The device sync analysis was conducted to figure out the magnitude of mega bytes that could be saved by taking this approach. The potential mobile data savings was evaluated by computing the content size played in multiple devices. This analysis gave an insight on the upper bound on the mobile data savings through this use case.

The contents played by a sample of users for four consecutive weeks were used for the analysis. The total mobile data consumption and the mobile data consumption for the content synced by the same sample of users in the upcoming week were computed. From the Figure 3.1, it was observed that the device sync use case could save a reasonable proportion of total download using mobile data. This approach had a potential of approximately 13%, as shown in Figure 3.2 on a consistent basis if modelled.

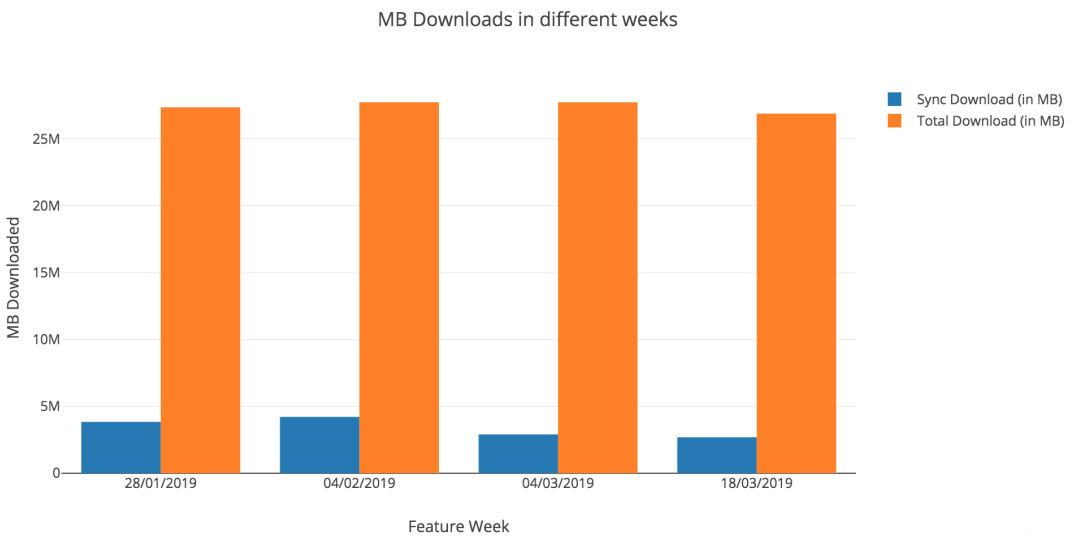

Figure 3.1: Potential savings using Sync Download compared to Total Download

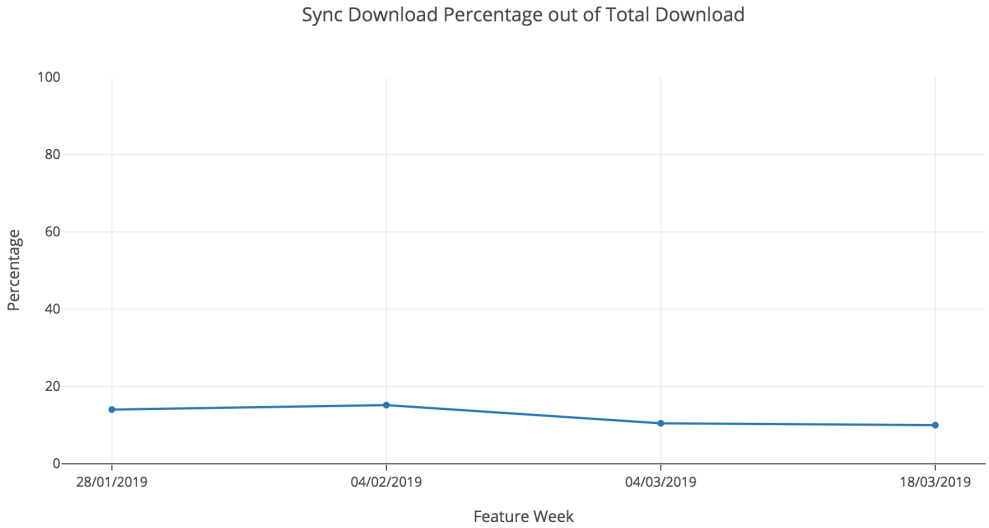

Figure 3.2: Percentage saved using Sync Download compared to Total Download (Weekly Analysis)

We extrapolated the analysis to a larger prediction period to analyze the long-term impact of predictive caching using device sync. In the Figure 3.3, the prediction period increased to one month and the analysis were conducted by computing the total consumption using mobile data and the mobile data consumption for the content synced for a period of one month. This approach also had a potential of approximately 13%, as shown in Figure 3.4 on a consistent basis if modelled.

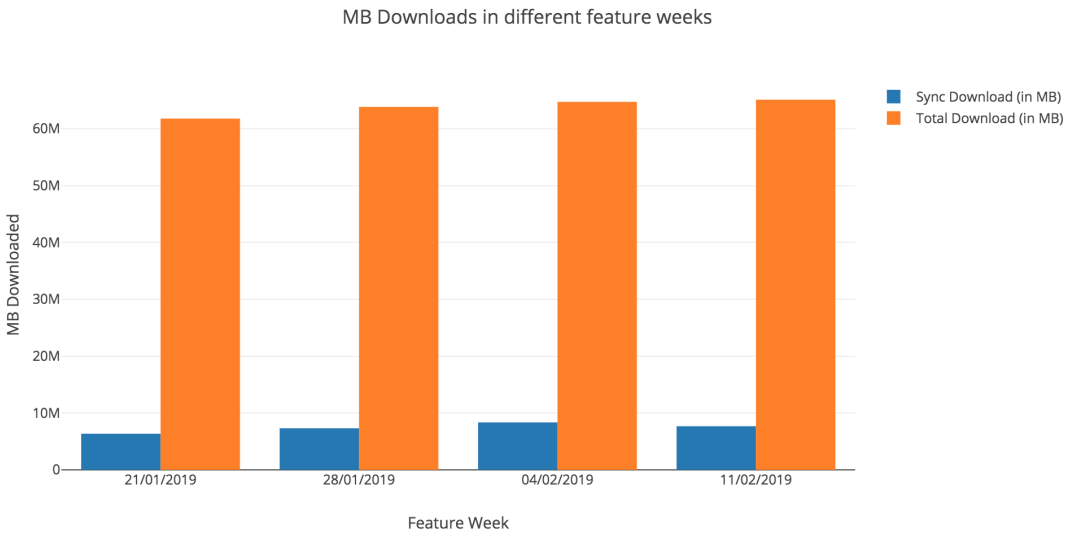

Figure 3.3: Potential savings using Sync Download compared to Total Download

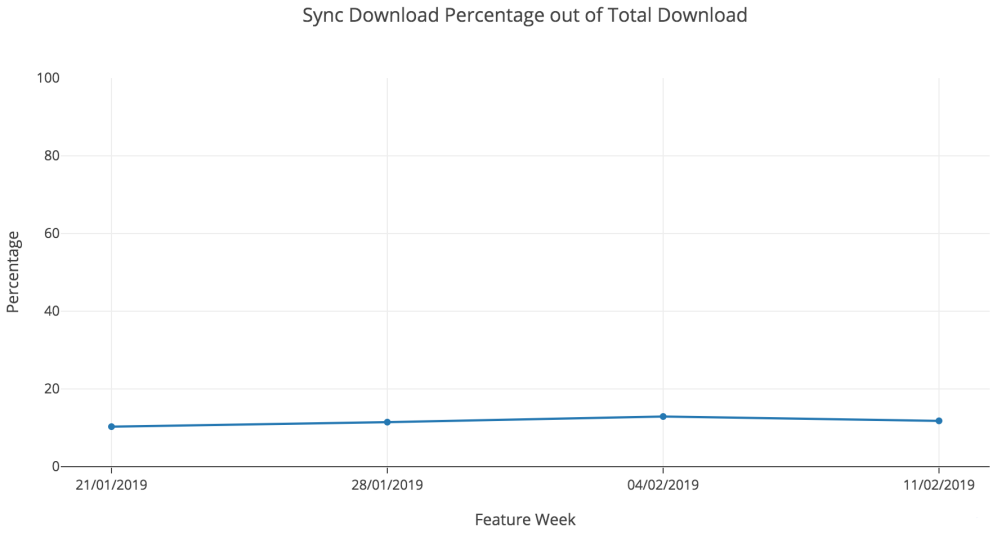

Figure 3.4: Percentage saved using Sync Download compared to Total Download

3.2 Playlists

Majority of the users prefer to play contents from any playlist provided by streaming services. Predictive caching focused on the playlist was an attempt to exploit this user behavior to predict the contents a user is likely to download in the future. The term playlist used here is generic since it comprises of both user created playlist, rec-

ommended contents collection and user specific as well as non user specific playlist generated by streaming services. The users are subjected to play any number of playlists and a standardization on the number of playlists for all users was essential for the effective modelling of playlist use case.

The first step in the analysis process was to rank top five playlist for a user based on the magnitude of mobile data consumed for a period of one month. On analysis, it was observed that approximately 35% of total carrier downloads are from the top 5 playlists of a user and approximately 82% of total playlist carrier downloads are from the top 5 playlists of a user, details of which are shown in Figure 3.5. Since top five playlists consume majority of the mobile data, it was reasonable to consider only those playlists of a user for modelling.

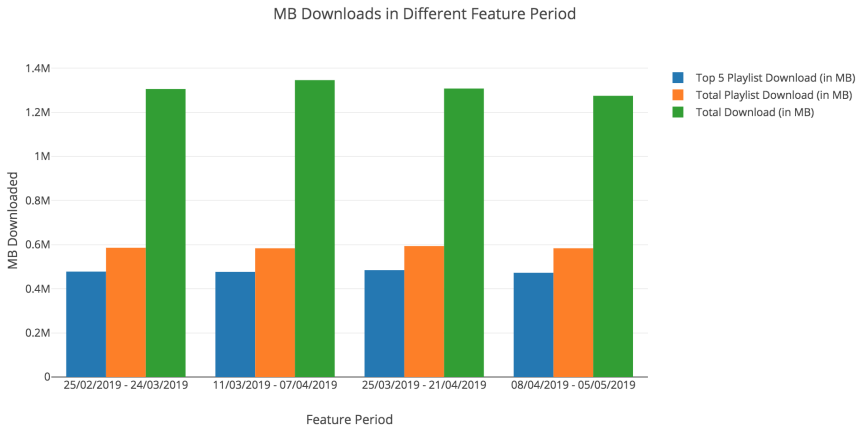

Figure 3.5: Absolute carrier downloads for top five playlists

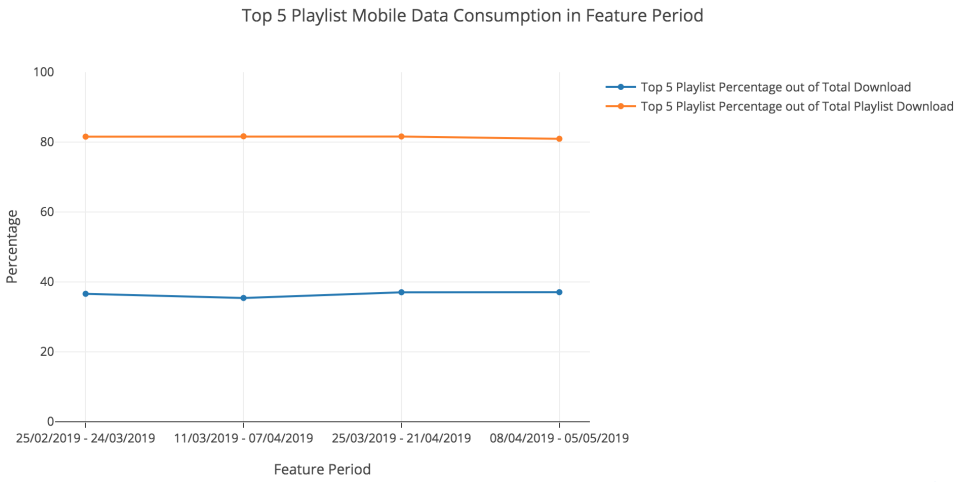

Figure 3.6: Percentage of carrier downloads for top five playlists

The potential saving in the target week using only top five playlists of a user was computed to figure out the upper bound on mobile data savings through this use case. On analysing the target week savings, it was observed that approximately 30% of carrier downloads for playlists are from the top 5 playlists. Moreover, approximately 13% of the total carrier downloads considering all the tracks played via search, playlist or tracks suggested from following artists or albums etc. are from the top 5 playlists. These figures are represented in Figure 3.7. The analysis showed a consistent savings of approximately 13% across different target weeks as shown in Figure 3.8.

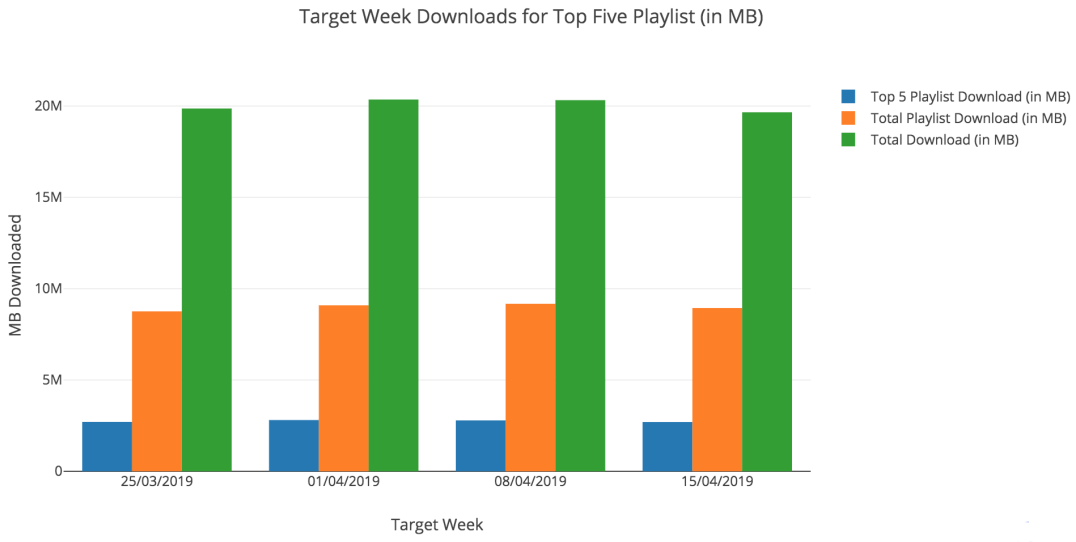

Figure 3.7: Potential savings for top five playlists

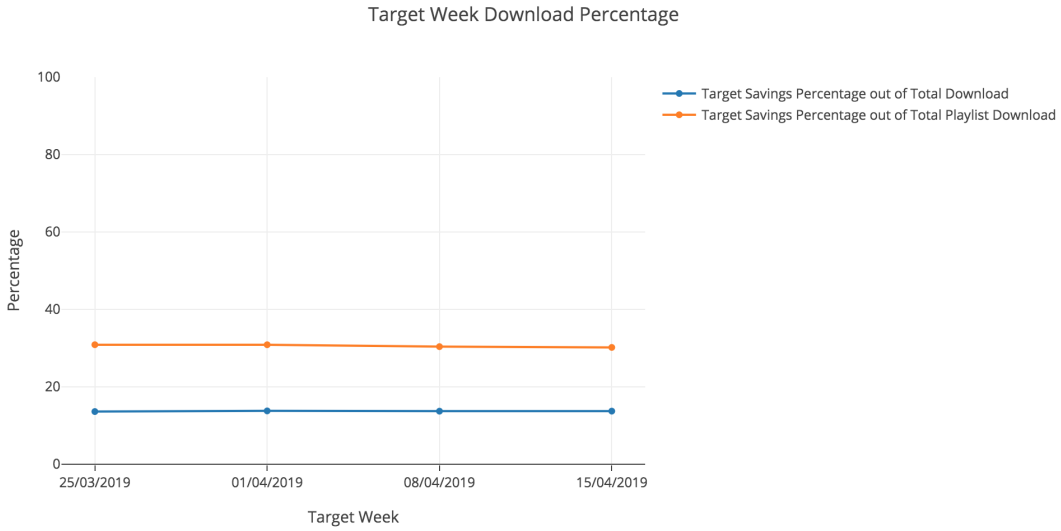

Figure 3.8: Percentage of potential savings for top five playlists

3.2.1 User Consistency

On analysing the user consistency on top five playlists for the selected subset of users, the following were observed, which further motivated the reason to model the playlists for predictive caching.

1. Users with carrier downloads for same playlist for 2 weeks or more - 37%
2. Users with carrier downloads for same playlist for 3 weeks or more - 12.3%
3. Users with carrier downloads for same playlist for exactly 4 weeks - 4%

4
Methods

This chapter details the different implementation steps undertaken during the thesis project. An initial target based on the usage of the personalized content, provided by the streaming services, was defined as the baseline and two approaches were used to model the specific problem at hand. The generic approach used for modeling the different machine learning approaches is presented before proceeding on to the different use cases and models.

4.1 Generic Modeling Approach

Five steps as shown in Figure 4.1 were performed across the different use cases, each of which are detailed below.

Figure 4.1: Steps in Generic Modeling Approach

1. Data Collection
   Appropriate data sets, based on opinions from domain experts and intuitive hypothesis for each use case under consideration, were generated, as detailed in upcoming sections.
2. Feature Engineering
   For each data point in the generated data set, a set of raw features were identified based on several hypothesis. Further, a set of derived features were computed using these raw features. In order to identify the features that could have an impact on the predictions, univariate and bivariate analysis as described in Sections 2.2.1 and 2.2.2 were performed. Feature cross validation as mentioned in Section 2.2.3 was conducted so as to eliminate collinear features and appropriate feature scaling(Section 2.2.4) was done for the remaining subset of features. Finally, the feature importance scores(Section 2.4.2) were computed using the appropriate prediction model being used for the specific use case.

3. Model Training
   Different machine learning models were trained on the training data sets generated for the specific use cases, details of which are described in later sections.
4. Hypreparameter Tuning
   Once the model was trained, hyper-parameters were tuned using the Grid Search algorithm(Section 2.5) on validation set.
5. Evaluation on Test set
   As the final step, evaluation was performed using the test set, details of which are presented in Chapter 5.

4.2 Baseline Target

Considering the fact that users are more likely to use the personalized playlist, an initial target was formulated as a binary classification problem to predict whether a user will download it using mobile data or not. In the below two sections, we present the rule-based heuristic and the machine learning models that were developed.

4.2.1 Rule-Based Heuristic

The rule-based heuristic was a naive attempt to mathematically model the trend in the usage of personalized content of a user without the aid of any machine learning techniques. In the rule-based heuristic, a subset of active users in the system who had played the personalized playlist for a period of four weeks was chosen at random. The trend across weeks was mathematically modeled to make predictions on whether the user will play the same personalized content in the fifth week or not. The trend value for each week is computed using the formula,

$$\text{Trend value} = \text{Count of plays in current week} - \text{Count of plays in previous week}$$

The trend value thus computed for each user was compared to a predefined threshold value. The threshold was defined as half the total size of the personalized playlist so as to have a probability of at least 50% for the content from the playlist to be played. Users with a high positive trend value were tagged as positive and users with a high negative trend value as negative. The remaining users in the selected subset were categorized as positive if the total number of plays in the fourth week was greater than the predefined threshold, otherwise negative.

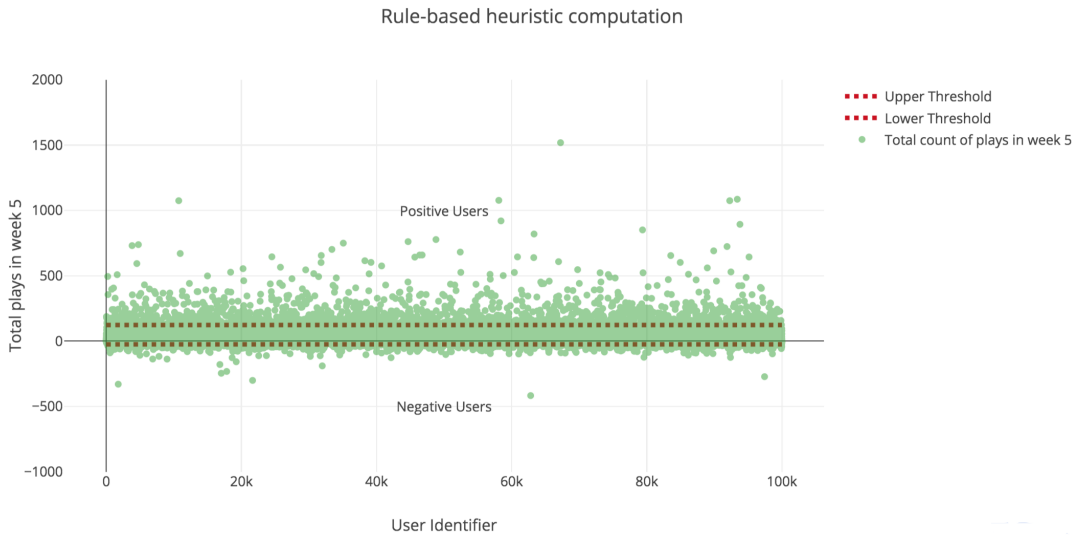

Figure 4.2: The average trend value for a period of four weeks was computed and summed up with total number of plays in week 4 to compute the total number of plays in week 5

4.2.2 Machine Learning Model

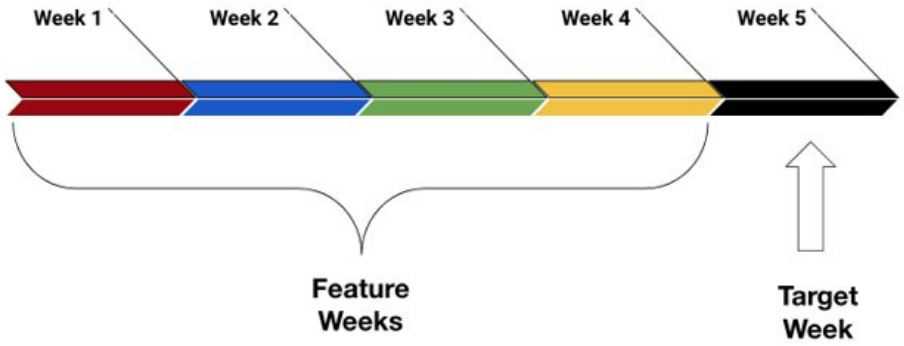

Figure 4.3: Dataset Preparation for Baseline Machine Learning Model

We formulated the binary classification problem as a supervised learning approach where the target was defined as positive if the user did download any track in the personalized playlist using mobile data, and negative, otherwise. Similar to the rule-based heuristic, a subset of active users in the system who had played the

personalized playlist for a period of four consecutive weeks were chosen at random. While preparing the data set for the model to be trained on, we considered four sets of weeks to compute the usage of the personalized playlist as features(listed in Table 4.1) and the fifth week as the target week, as shown in Figure 4.3. A balanced data set was used for training and a validation set comprising 30% of the data was used to evaluate the generalization power of the model and to prevent overfitting. The model outputs the probability of the content to be considered as positive. The supervised learning models, Logistic Regression and LightGBM [23], were trained on the training dataset, and the results are presented in Chapter 5. The hyper parameters of these models[1][2] were tuned using the Grid Search method to find the optimal values which are listed in Tables 4.2 and 4.3 respectively.

Table 4.1: Features used in baseline machine learning models

Primary Key	**Feature Names**
User identifier	Total count in Week 1
	Total count in Week 2
	Total count in Week 3
	Total count in Week 4

Table 4.2: Hyper-parameters for baseline LightGBM model

Parameter Name	**Parameter Value**
Number of predictors	100
Boosting Type	Gradient Boosting Decision Tree
Objective	Binary
Number of leaves	64
Learning Rate	0.03
Colsample by Tree	0.8
Metric	Binary logloss

Table 4.3: Hyper-parameters for baseline Logistic Regression model

Parameter Name	**Parameter Value**
Number of epochs	100
Regularization	L2

4.3 Use Cases

In the subsequent sections, we will describe the methods adopted for modeling the different use cases discussed in Chapter 3 and the feature engineering techniques applied to find the relevant and appropriate features to output best predictions.

[1]https://xgboost.readthedocs.io/en/latest/parameter.html
[2]http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

4.4 Device Sync

In this section, we describe the modeling approaches considered for device sync use case.

4.4.1 Dataset Preparation and Modeling

For the purpose of an unambiguous explanation, we will be using the terms primary device and secondary device throughout this section. The devices used by the user to listen to some content in week one were termed as the primary device and the ones that would be used by the user to download the same content in the upcoming week were termed as secondary devices. Each data point for a device sync dataset can be uniquely identified by the user identifier, the primary device used to play the content in week one, the content identifier and the secondary device. For each data point, we defined a feature week and target week, as shown in Figure 4.4, where the feature week was used to select the user devices (both primary and secondary devices) and the content for which we wanted to make predictions for. The target was defined as positive (class 1) if the user did download the content listened to in the primary device in the secondary device the upcoming week, called target week, using mobile data, otherwise negative (class 0). All the devices used by the user in the feature week are considered as primary device whereas only those devices with mobile data downloads in the feature week are considered as a secondary device for the purpose of predictive caching. Moreover, data points irrelevant for the device sync modeling, for example, when the user had used both primary and secondary devices to stream content in the feature week itself, were carefully removed to ensure data consistency.

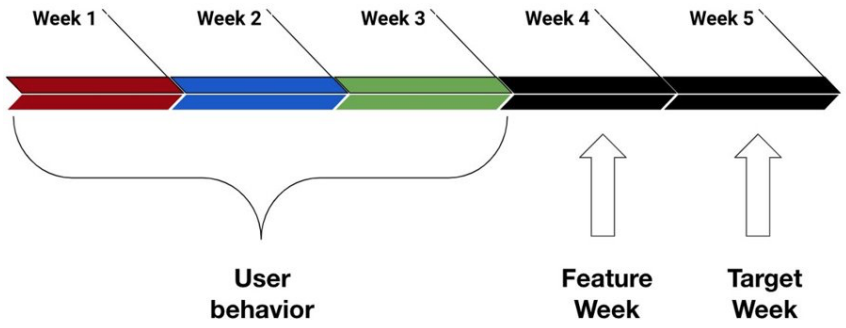

Figure 4.4: Dataset Preparation - Weekly Target

In order for the model to give better predictions, user behavior in the past weeks had to be taken into account. For example, the proportion of content, played by the user in week one (refer to Figure 4.4) in the primary device, downloaded in the secondary device in the upcoming week could be indicative of how actively the user uses the secondary device for downloading content through mobile data. User

behavior of the past three weeks was used to measure how consistent are they in using multiple devices as well as to infer on cross-device syncing patterns exhibited by that specific user.

While modeling the data, we implemented two approaches based on how the primary device-secondary device combination was chosen.

1. All devices - All the different combinations of the user devices from the feature week were considered, provided the primary and the secondary devices are not the same and the latter had a mobile data download for some content.
2. Fixed devices - For fixing the devices, we took into account all the primary-secondary device combinations for each user over a period of one month. Each such combination was ranked based on how frequently content sync has happened between these devices. In order to fix the devices for modeling, for each user, we used the top two from the ranked list, provided the secondary had a mobile data download for some content.

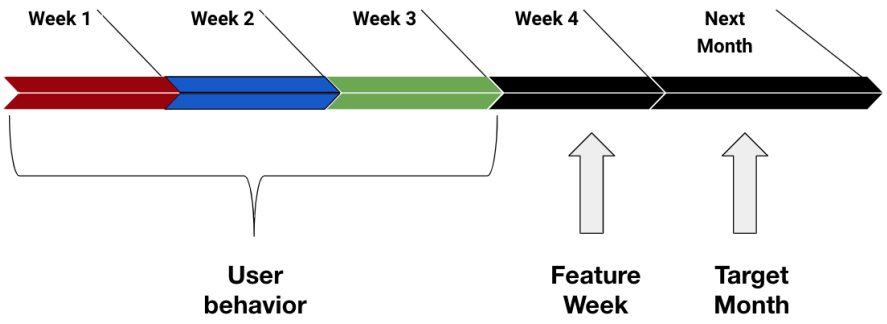

Figure 4.5: Dataset Preparation - Monthly Target

In order to compare the results of a weekly target with a longer target period and verify if one week is too short a prediction window, we also modeled another dataset with a monthly target, as shown in Figure 4.5.

The data for training, validation, and test, was collected for an active set of users for different sets of weeks in December, January, and February. The data set thus generated was highly imbalanced in terms of the number of negative and positive samples. Hence, an unbalanced data set was used for training since it represents the real data accurately. In addition, 30% of the training data was used as a validation set. A Gradient Boosting algorithm called XGBoost classifier was used to train the data set as ensemble techniques are known to perform better with skewed data set. The classifier outputs a probability on whether a user has downloaded the content on a secondary device or not. The results are presented in Chapter 5. The hyper parameters of the XGBoost models were tuned using the Grid Search method to find the optimal values which are listed in 4.4.

Table 4.4: Hyper-parameters for XGBoost model used for Device Sync use case

Parameter Name	**Parameter Values**	
	All Devices	Fixed Devices
number of predictors	100	100
booster	Gradient Boosting Decision Tree	Gradient Boosting Decision Tree
objective	Binary	Binary
learning_rate	0.2	0.1
colsample_bytree	0.8	0.8
scale_pos_weight	53	27
max_depth	3	3
min_child_weight	5	5
subsample	0.8	0.8
reg_alpha (L1 regularization)	0.8	0.7
reg_lambda (L2 regularization)	0.8	0.7

4.4.2 Feature Engineering

For each data point, we identified sets of raw features and derived features from the available data set based on hypothesis, as listed below.

1. Raw features
   (a) User level
      i. Countries where mobile connection is cheap, one might download more from carrier
      ii. Long-term users are highly likely to show a consistent behavior in combination with other features
   (b) Content level
      i. Popular content are more prone to be played in multiple devices
      ii. Content in multiple playlists are more prone to be played
      iii. Newly released content has high pre-cache potential
   (c) Device level
      i. New devices have high pre-cache potential
      ii. Device type, for example, if it is a desktop or mobile
2. Derived features
   (a) User-Content level
      i. User's past content preferences
   (b) User-Device level
      i. User consistency in using multiple devices
      ii. Time of the day when the user used the devices
   (c) Content-Device level
      i. Content in personalised playlist are more likely to be played in multiple devices
      ii. Device similarity of the content

4.5 Playlists

In this section, we describe the modeling approaches considered for the playlist use case.

4.5.1 Approaches

The two approaches that were considered are as follows.

1. Playlist Prediction - The sub-problem was formulated as a binary classification problem to predict whether the user will download the content from a playlist or not. Here, the idea was to download all the contents within the positively predicted playlist.
2. Content Prediction - The sub-problem was designed to restrict the number of contents that would be downloaded from a playlist. Here, the problem was modeled as a two-stage prediction process as shown in Figure 4.6, with (i) Stage 1 predicting the playlist and (ii) Stage 2 predicting the particular content within the positively predicted playlist. This could be seen as a second content prediction model appended to the initial playlist prediction model, both of which are targeted as binary classification sub-problems.

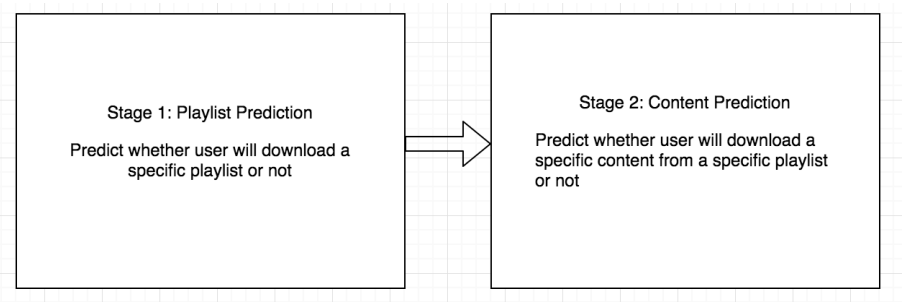

Figure 4.6: Two-stage Prediction Model for Playlists Use Case

4.5.2 Stage 1 : Playlist Prediction

This section details the dataset preparation steps, various modeling approaches and the feature engineering performed for stage 1 predicting the playlist.

4.5.2.1 Dataset Preparation and Modeling

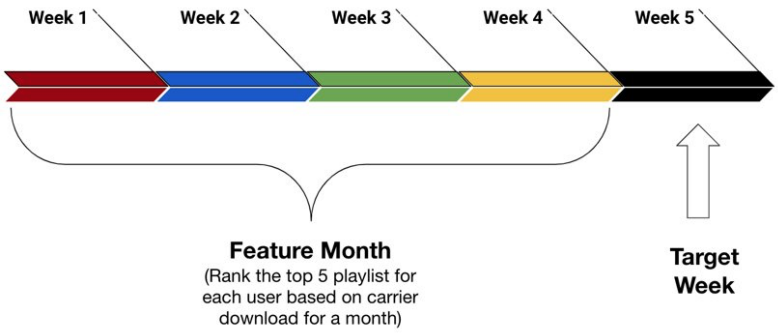

Figure 4.7: Dataset Preparation for Stage 1: Playlist Prediction

Each data point for the playlist prediction model was uniquely identified by the user identifier and the playlist identifier. For each data point, we defined a feature period of one month and a target week, as shown in Figure 4.7. The number of playlists played by each user was randomly varying which motivated the need to fix the same. As mentioned in Section 3.2, the top five playlists of user accounts to 82% of the downloads through a mobile carrier and hence we decided to fix the number of playlists for any user to be at most five. The playlists were ranked based on the total count of the playlist contents that were downloaded using mobile carrier by that user for a month. For each such data point, the target was defined as positive (class 1), if the user did download at least one track from the playlist in the next two weeks, otherwise negative (class 0).

As explained in the device sync model in Section 4.4.1, it was important to capture the user behavior in past weeks in order for the model to give better predictions. For example, the number of weeks with carrier downloads in the feature period for each playlist would give a measure of consistency over time, and is indicative of how actively the playlist is hit by a download request. User behavior of past four weeks from the target week was used to measure this user consistency patterns for each data point.

The data for training, validation and test was collected for an active set of users for different sets of weeks in January, February and March. The data set thus generated was highly imbalanced in terms of the number of negative and positive samples. But for the purpose of training, a balanced data set was carefully chosen. Both Logistic Regression and XGBoost classifier were used separately to train the data set and the results are presented in Chapter 5. The hyper-parameters for the XGBoost model were tuned using the Grid Search Algorithm and are listed in Table 4.5.

Table 4.5: Hyper-parameters for XGBoost model used for Playlists use case - Stage 1 Playlist Prediction

Parameter Name	Parameter Value
epochs	100
booster	Gradient Boosting Decision Tree
objective	Binary
learning_rate	0.09
colsample_bytree	0.8
scale_pos_weight	6
max_depth	3
subsample	0.9
min_child_weight	5
reg_lambda (L2 regularization)	1

4.5.2.2 Feature Engineering

For each data point in the stage 1 prediction model, we identified sets of raw features and derived features from the available data set based on hypothesis, as listed below.

1. Raw features
   (a) User level
      i. Age group of the user
   (b) Playlist level
      i. Number of days since the playlist was last updated
      ii. Playlist type, user-created or personalised
      iii. Playlist age
      iv. Popularity of the playlist
2. Derived features
   (a) User Playlist level
      i. Rank of the playlist
      ii. Weeks with bytes from carrier is a measure of consistency over time
      iii. Count of carrier downloads in previous weeks could indicate the user consistency for the ranked playlists
      iv. Sum of bytes from carrier for each week in the feature time

4.5.3 Stage 2 : Content Prediction

This section details the dataset prepration steps, various modeling approaches and the feature engineering performed for stage 2.

4.5.3.1 Dataset Preparation and Modeling

Each data point for the content prediction model can be uniquely identified by the user identifier, playlist identifier and the content identifier. In order to select these data points, all the positively predicted playlists from stage 1 and all the current contents within those playlists were considered. For this purpose, the latest playlist

data available as of the day before the target period were considered. The target was defined as positive (class 1) if the user did download that content in the next two weeks, irrespective of the playlist associated with it, otherwise negative (class 0).

The data for training, validation and test were created by running the stage 1 playlist prediction model for an active set of users for different weeks in February, March and April. The data set thus generated was highly imbalanced in terms of the number of negative and positive samples. Hence, an unbalanced data set was used for training since it represents the real data accurately. In addition, 30% of the training data was used as a validation set. As in the device sync model, an XGBoost classifier was used to train the data set as ensemble techniques are known to perform better with skewed data set. The results are presented in Chapter 5. The hyper-parameters for the XGBoost model were tuned using the Grid Search Algorithm and are listed in Table 4.6.

Table 4.6: Hyper-Parameters for XGBoost model used for Playlists use case : Stage 2 Content Prediction

Parameter Name	**Parameter Value**
epochs	100
booster	Gradient Boosting Decision Tree
objective	Binary
learning_rate	0.1
colsample_bytree	0.6
scale_pos_weight	26
max_depth	1
min_child_weight	5
subsample	0.4

4.5.3.2 Feature Engineering

For each data point in the stage 2 prediction model, we identified sets of raw features and derived features from the available data set based on hypothesis, as listed below.

1. Raw features
   (a) User level
      i. Age group of the user
   (b) Track level
      i. Days since the track got added to the playlist
      ii. Position of the track in the playlist
      iii. Popularity of the track
      iv. Number of followers for the track
2. Derived features
   (a) User Track level
      i. Whether this track is already cached by the user or not decides if it should be pre-cached

5

Evaluation and Results

In this chapter, we present the evaluation strategy and the results obtained for the different approaches described in 4.

5.1 True Byte Rate (TBR)

In order to evaluate the machine learning models being developed for predictive caching and to determine the quality of their predictions, we devised a generic evaluation metric called 'True Byte Rate' (TBR). It is used to compute the fraction of mega bytes (MB) saved out of the MB pre-fetched from the model predictions. TBR was effective in evaluating the business impact of using the model since it reflects on how many MB could have been saved by pre-fetching the predicted content on to the user's device. This was performed by comparing the prediction results with the ground truth data and computing the (i) bytes saved and (ii) bytes wasted. Predicting a user as positive when he did not play the personalized content is worse than predicting a user as negative when he did play it because we are unnecessarily consuming the user device's memory for false positives. Evaluation metrics like Precision, Recall, and Area Under the Curve(AUC) score could be used to evaluate a classification model, but they are not sufficient to assess the business value of the model's predictions. This necessitated the need for a generic evaluation metric for predictive caching.

5.2 Baseline Model

5.2.1 Rule-based Heuristic

The rule-based heuristic calculated the user's personalized content usage in the fifth week based on their historical behavior for the past four weeks. It was evaluated on a subset of one million users taken at random for a period of 29^{th} October 2018 to 28^{th} November 2018. The rule-based heuristic identified about 1% of users as positive to download the personalized content using mobile data while the ground truth was around 9%. Though the heuristic method was not sufficient enough to predict the user behavior on the fifth week, it strongly indicated the significance of considering historical user behavior for predicting the download pattern of a user and hence was crucial for the later machine learning models.

5.2.2 Machine Learning Model

In this section, we present the results obtained using the baseline model described in Section 4.2.2.

We plotted the percentage of bytes saved (in MB) out of the total that was pre-fetched, the True Byte Rate(TBR), on different thresholds as shown in Figure 5.1. The threshold here defines the probability for a user to download the personalized content given by the model predictions. For a threshold of 0.70, at least 50% MB got saved out of the pre-fetched. The percentage of bytes getting saved was getting decreased with the decrease in the threshold as seen from the figures. So, the bytes saved vs bytes wasted was plotted as shown in Figure 5.2, and it was seen that the bytes getting wasted were increasing with the decrease in threshold. This explained the behavior of decrease in the percentage of bytes getting saved with the decrease in threshold.

Figure 5.1: True Byte Rate

Figure 5.2: Bytes saved Vs Bytes wasted

Further, on analyzing the usage distribution of all users on the personalized content, it could be seen that the number of users playing the first few tracks in the playlist is very high compared to the number of users playing the 30^{th} track. The percentage of users who actually listened to the pre-fetched tracks is shown in Figure 5.3. For the first few tracks, there was a steady increase in the number of users and it got flattened out towards the end as expected.

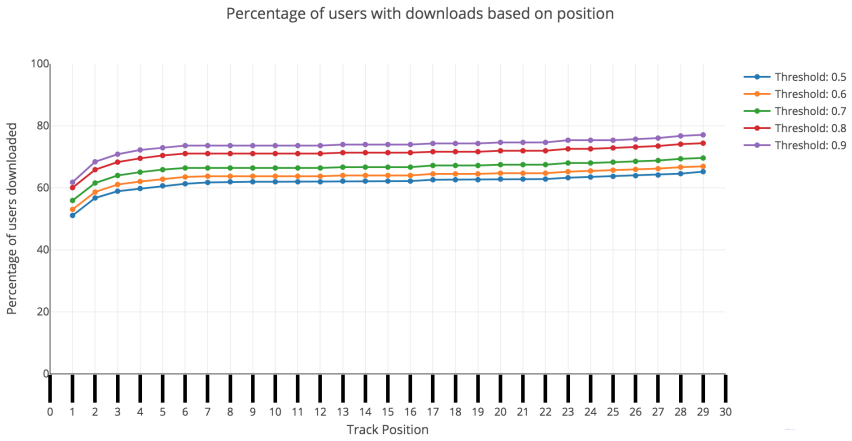

Figure 5.3: Percentage of users who actually listened to the pre-fetched tracks

5.2.3 Threshold calibration

Threshold calibration was an important step in model evaluation whereby an optimal threshold to be used for the model was decided upon. It focused on how beneficial the model would be in the sense that, what use of the model would potentially save the most data. For this purpose, a minimum byte savings of 50% out of total pre-fetched is fixed regardless of the threshold by setting the TBR as 0.5. It could be seen from Figure 5.1 that the thresholds 0.5, 0.6 and 0.7 did attain a TBR of 0.5 if we pre-fetched 3, 15 and 27 contents respectively from the personalized playlist under consideration. The total bytes saved on these specific points were computed to identify the threshold with maximum savings. It could be seen from Figure 5.4 that by downloading approximately 30 tracks at threshold 0.7 the model attained 50% savings, which is possible either when half of the users downloaded all the 30 tracks or when all the users downloaded 15 tracks.

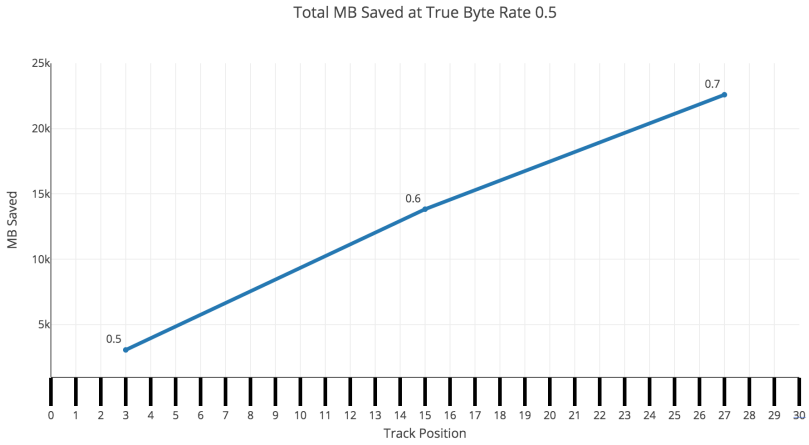

Figure 5.4: Total MB saved at TBR = 0.5

5.3 Device Sync Model

From the initial EDA, it was observed that the device sync use case could save a maximum of 13% mega bytes from mobile data out of the total device sync downloads for a period of one month. As described in Section 4.4.1, two different approaches were considered for device sync - one by selecting all the primary-secondary device combinations and another by fixing a set of specific primary-secondary device combinations for a user. The threshold calibration for device sync was performed to find an optimal threshold that would provide maximum byte savings with minimum byte wastage. This could be determined by plotting the Precision-Recall curve on mobile data saved during the process. Here, we present the results obtained for these two approaches using weekly and monthly targets in the below subsections.

5.3.1 Approach 1: All devices

This approach was tested on a data set containing 2000 users taken at random. The TBR evaluation was conducted for weekly and monthly targets as could be seen from Figures 5.5 and 5.6 respectively. It was observed that the byte savings increased with the increase in threshold value until it attained a maximum and then dropped. The drop in byte savings at a high threshold value was due to the fact that the ratio of false positives to true positives was substantially high, that is, effectively the byte wastage was high. For a weekly target objective, the TBR ranged from 8% to 27% whereas, for a monthly target objective, the TBR ranged from 22% to 62%. This was as expected, since the probability for a user to sync the contents to a secondary device in the week immediately following the feature week was less compared to the probability to sync in the month immediately after the feature week. The longer the prediction period, the more the chances of device sync by a user. Though, a target period of one month was an over-estimate from the business point of view.

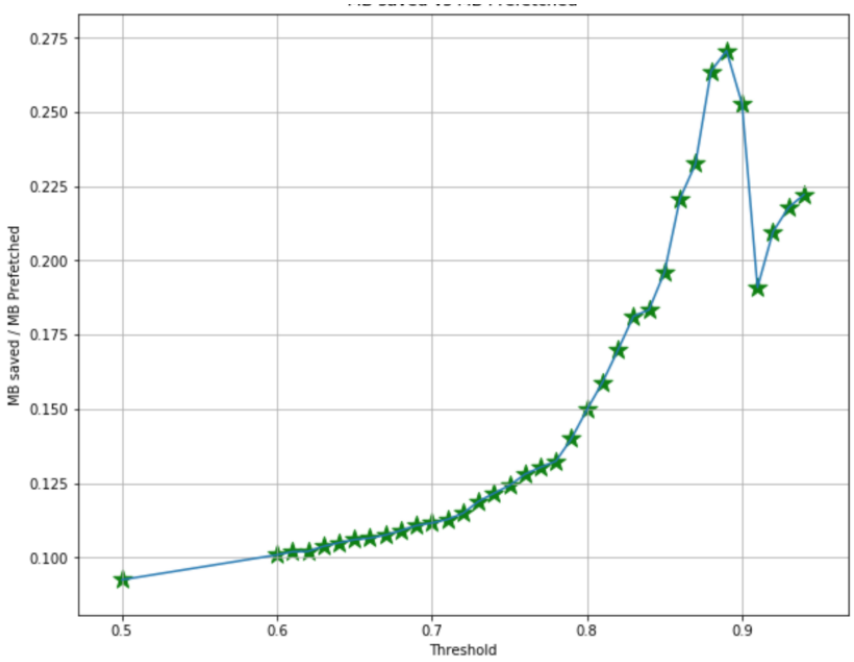

Figure 5.5: Approach 1 : True Byte Rate for Weekly Target

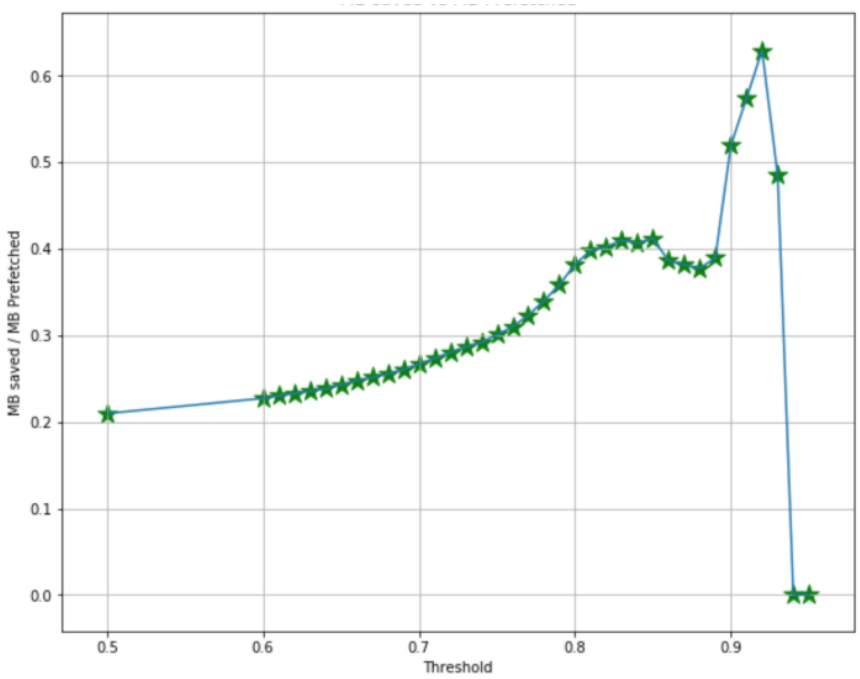

Figure 5.6: Approach 1 : True Byte Rate for Monthly Target

5.3.1.1 Threshold calibration

In order to find an optimal threshold with maximum savings and minimum wastage for weekly and monthly targets, we computed the Precision and Recall values with

respect to the total mega bytes (MB) saved from model predictions, as shown in Figures 5.7 and 5.8 respectively. On this front, Precision is the total MB saved out of the total pre-fetched, which is exactly the TBR that has been defined for the evaluation metric. Recall is the total MB from carrier saved out of the total MB that was actually downloaded using mobile data, in the context of device sync.

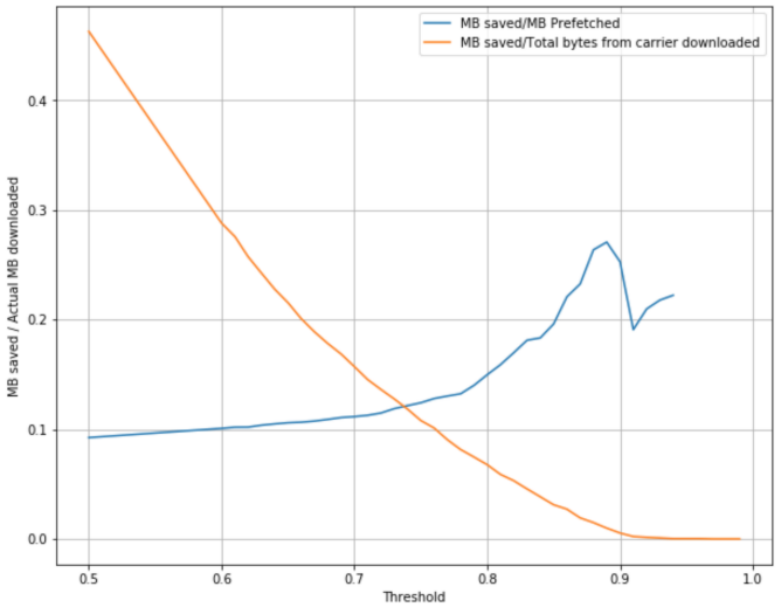

Figure 5.7: Approach 1 : Weekly Target - Precision Recall Curve for MB saved

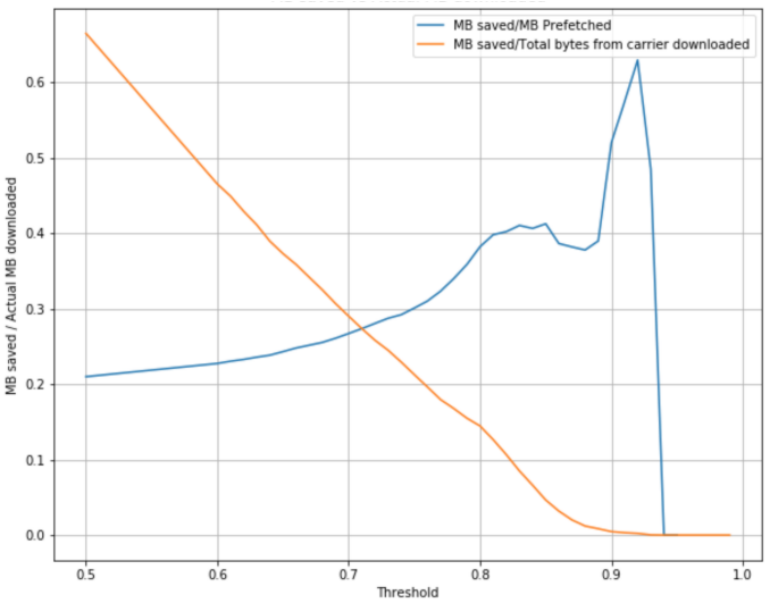

Figure 5.8: Approach 1 : Monthly Target - Precision Recall Curve for MB saved

The observations are consolidated in the section below.

5.3.1.2 Observations

1. Weekly Target
   (a) True Byte Rate (TBR) ranged from 8 to 50%
   (b) Optimal Threshold ranged between 0.7 & 0.75 with,
      i. 11-12% TBR
      ii. 380-530 users, out of which 140-210 actually downloaded
      iii. True Positives : False Positives ratio was 1 : 6
2. Monthly Target
   (a) True Byte Rate (TBR) ranged from 22 to 62%
   (b) Optimal Threshold ranged between 0.7 & 0.75 with,
      i. 28-30% TBR
      ii. 430-630 users, out of which 280-420 actually downloaded
      iii. True Positives : False Positives ratio was 1 : 2

5.3.2 Approach 2: Fixed Devices

This approach was tested on a data set containing 1000 users taken at random. The TBR evaluation was conducted for weekly and monthly targets as could be seen from Figures 5.9 and 5.10 respectively. On comparison with All-Devices approach described in Section 5.3.1, the TBR values were high in Fixed-Devices approach for both weekly and monthly targets. The TBR value ranged from 13% to 38% for the weekly target objective and 28% to 48% for the monthly target. This was clearly higher compared to the values obtained in Approach 1.

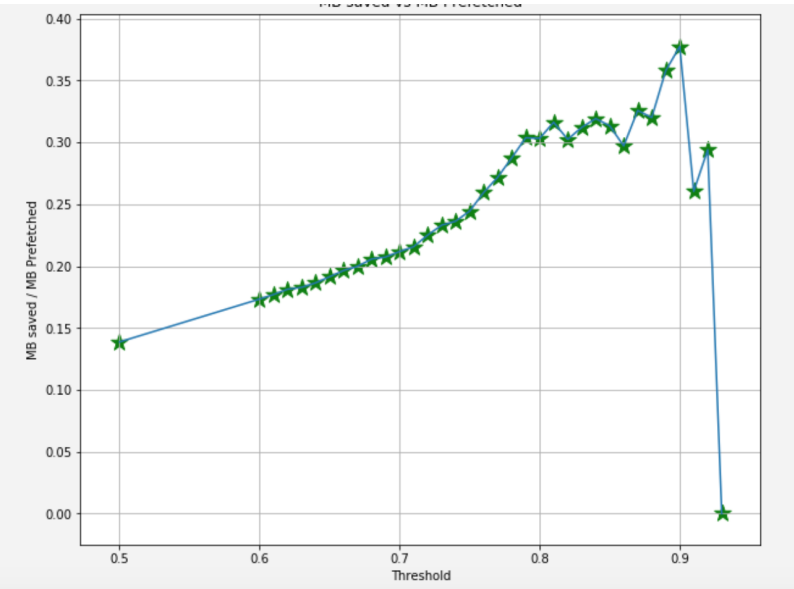

Figure 5.9: Approach 2 : True Byte Rate for Weekly Target

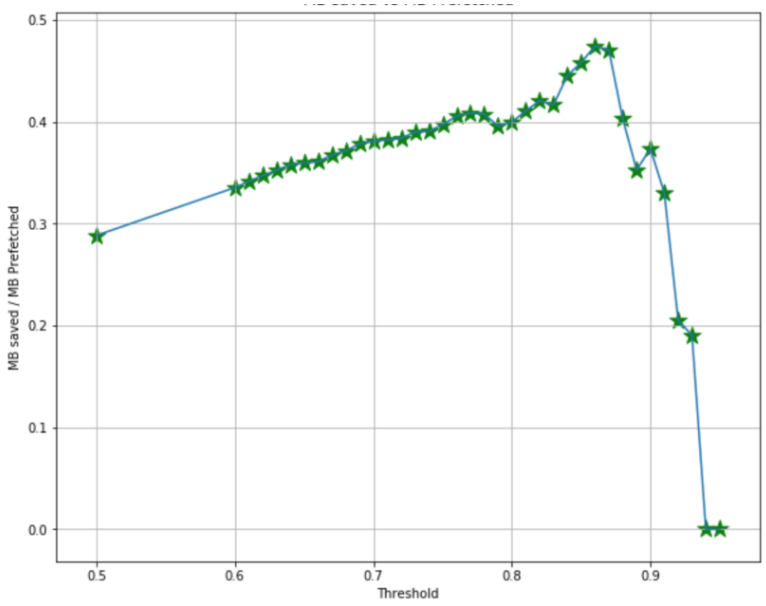

Figure 5.10: Approach 2 : True Byte Rate for Monthly Target

5.3.2.1 Threshold calibration

The same approach as described in Section 5.3.1.1 was used, and the results for weekly and monthly targets are plotted in Figures 5.11 and 5.12 respectively. The observations are consolidated in the section below.

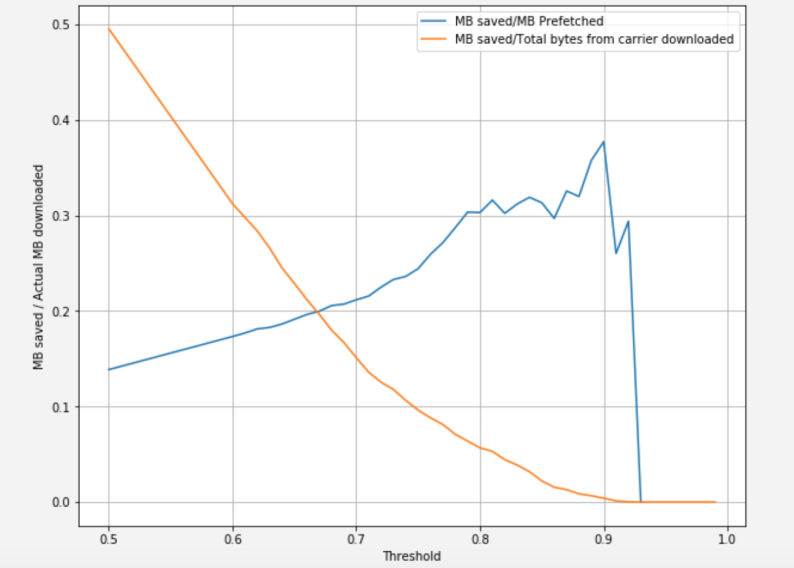

Figure 5.11: Approach 2 : Weekly Target - Precision Recall Curve for MB saved

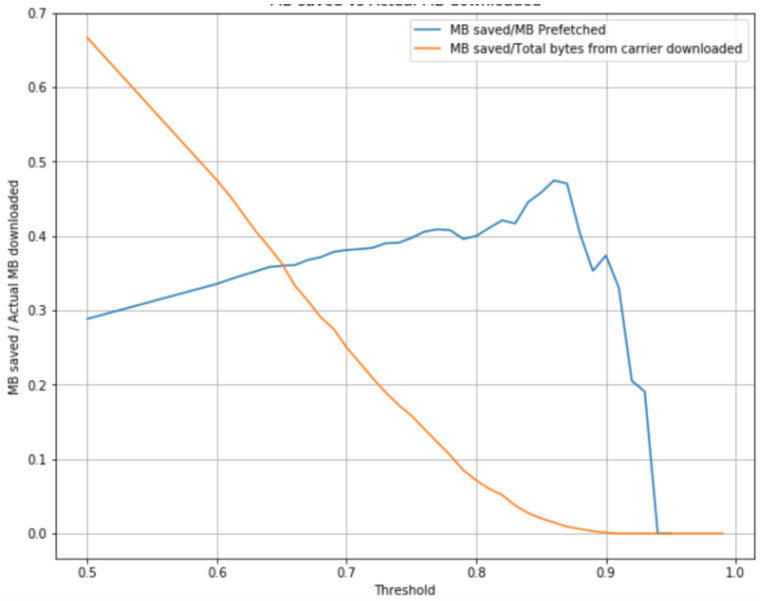

Figure 5.12: Approach 2 : Monthly Target - Precision Recall Curve for MB saved

5.3.2.2 Observations

1. Weekly Target
   (a) True Byte Rate (TBR) ranged from 13-38%
   (b) Optimal Threshold ranged between 0.65 & 0.70 with,
      i. 19-21% TBR
      ii. 370-500 users, out of which 180-280 actually downloaded
      iii. True Positives : False Positives ratio was 1 : 4
2. Monthly Target
   (a) True Byte Rate (TBR) ranged from 28 to 48%
   (b) Optimal Threshold ranged between 0.6 & 0.7 with,
      i. 33-38% TBR
      ii. 470-670 users, out of which 350-530 actually downloaded
      iii. True Positives : False Positives ratio was 1 : 1.5

5.3.3 Summary

It was observed that the device sync model with fixed-device approach for monthly target could save approximately 2% out of the potential 13% savings discovered during the Exploratory Data Analysis.

5.4 Playlist Model

From the initial EDA, it was observed that 82% of the total playlist carrier downloads in a month were from the top five playlists of a user and 13% of the total carrier downloads were from the top 5 playlists in a week. As described in 4.5.1,

two different approaches were considered for playlists use case - one by predicting whether the user will download the whole playlist or not, and another by appending a content prediction stage, which determines the exact content to be pre-fetched. The TBR evaluation was conducted for weekly and monthly targets on single stage playlist prediction model as well as two-stage content prediction model.

5.4.1 Stage 1 : Playlist Prediction

This approach was tested on a data set containing 2500 users taken at random. The TBR evaluation was conducted for weekly and monthly targets as could be seen from Figures 5.13 and 5.14 respectively.

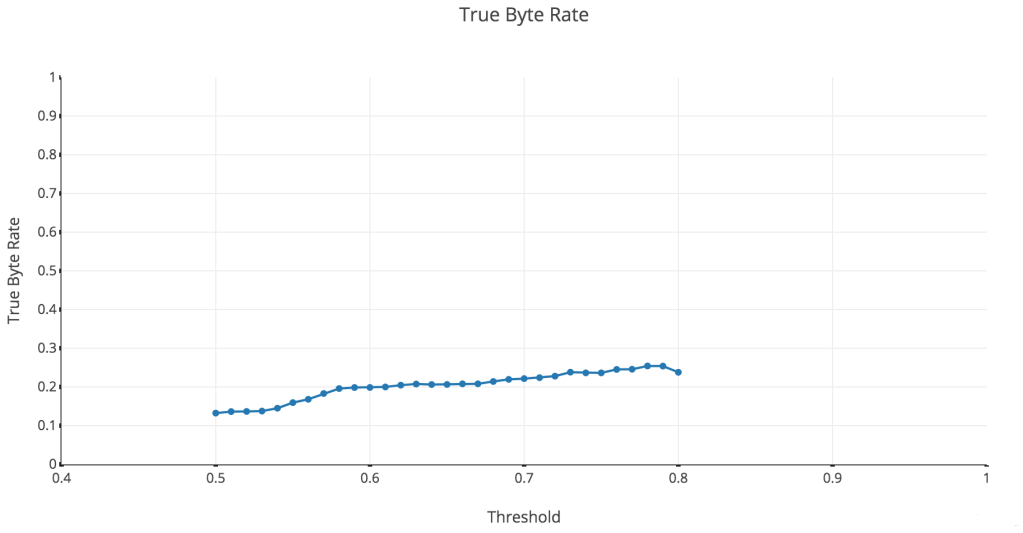

Figure 5.13: Stage 1 : TBR for a prediction period of one week

In weekly prediction, the TBR value ranged from 13% to 25%. It was observed to have 20% savings out of the total pre-fetched, on an average, and approximately 80% attributed to the total wastage. This motivated the need to expand the prediction window to a period of one month. For monthly prediction, the TBR value ranged from 25% to 49%. Compared to weekly prediction, the TBR value increased from approximately 25% to 49% at the highest possible model threshold of 0.8. Though an increase in the prediction period to one month effectively doubled the TBR range compared to the weekly prediction, the byte wastage was significantly high on an average. Moreover, the approach to pre-fetch all the tracks in the positively predicted playlist seemed to unnecessarily use up the device storage. Thus, threshold calibration was not performed for stage 1.

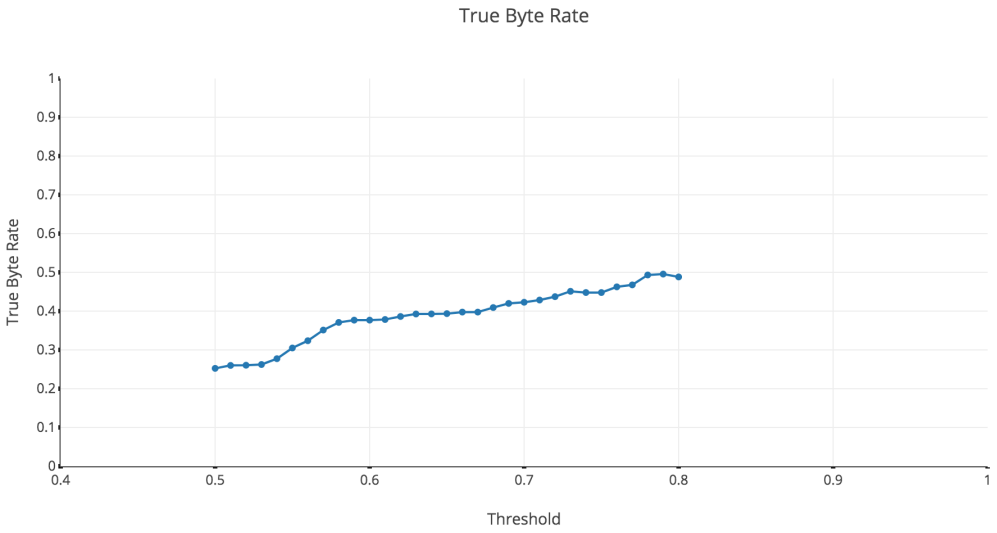

Figure 5.14: Stage 1 : TBR for a prediction period of one month

5.4.2 Stage 2 : Content Prediction

This approach was tested on a data set containing 2500 users taken at random. The TBR evaluation was conducted for weekly and monthly targets as could be seen from Figures 5.15 and 5.16 respectively.

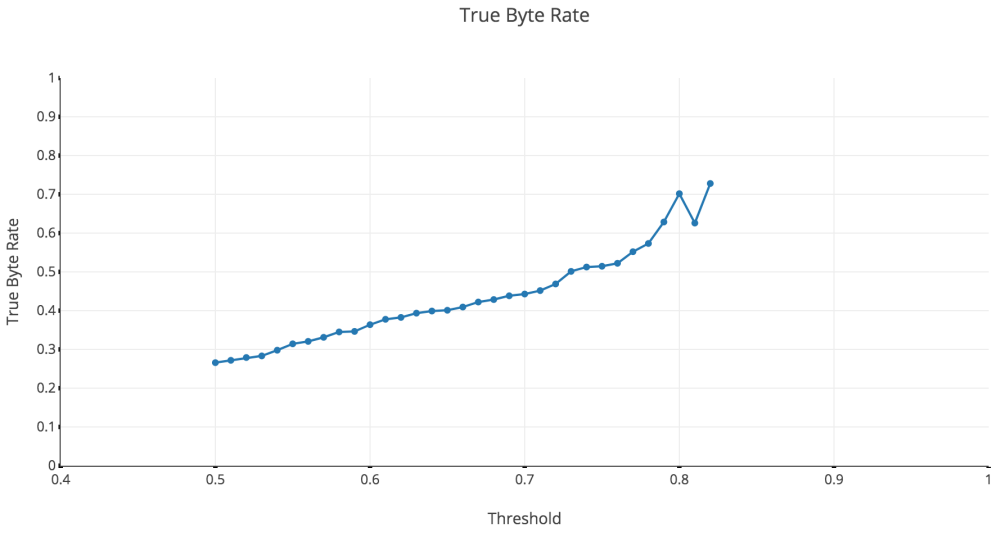

Figure 5.15: Stage 2 : TBR for a prediction period of one week

In weekly prediction, the TBR value ranged from 26% to 70%. In comparison with the stage 1 model for weekly prediction, the stage 2 model attained a TBR of 26%

at a threshold of 0.5 whereas the stage 1 had a TBR of 26% only at threshold 0.8. It was observed to have 50% savings out of the total pre-fetched, on an average, and approximately 50% attributed to the total wastage. For completeness to compare with the stage 1 model, TBR evaluation was performed for monthly prediction window as well and the value ranged from 47% to 89%. Compared to weekly prediction, the TBR value increased from approximately 70% to 89% at the highest possible model threshold of 0.8. It was observed that the TBR value showed a steady increase with the increase in threshold for both weekly and monthly targets.

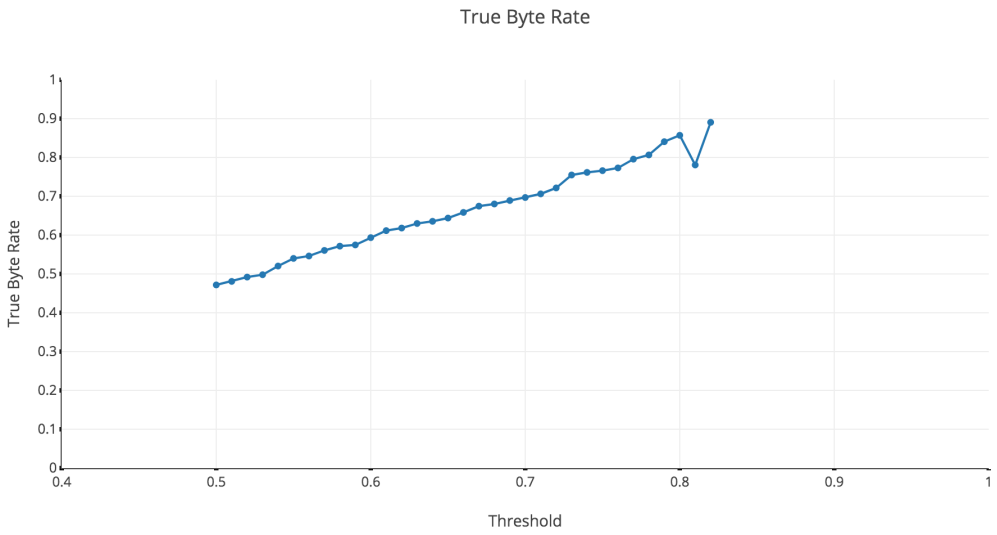

Figure 5.16: Stage 2 : TBR for a prediction period of one month

5.4.2.1 Threshold calibration

The same approach as described in Section 5.3.1.1 was used, and the results for weekly and monthly targets are plotted in Figures 5.17 and 5.18 respectively. The observations are consolidated in the section below.

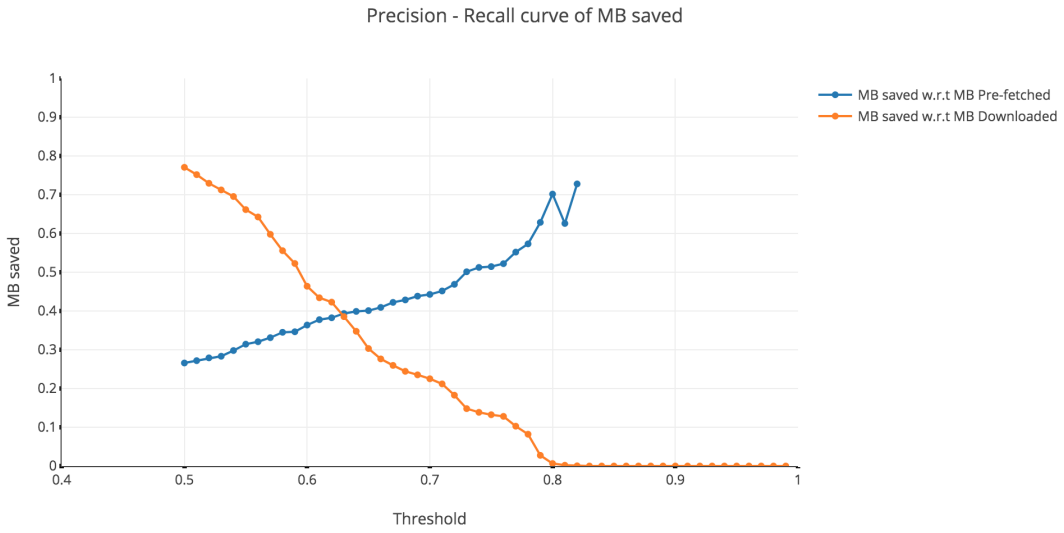

Figure 5.17: Stage 2 : Weekly Prediction - Precision Recall Curve for MB saved

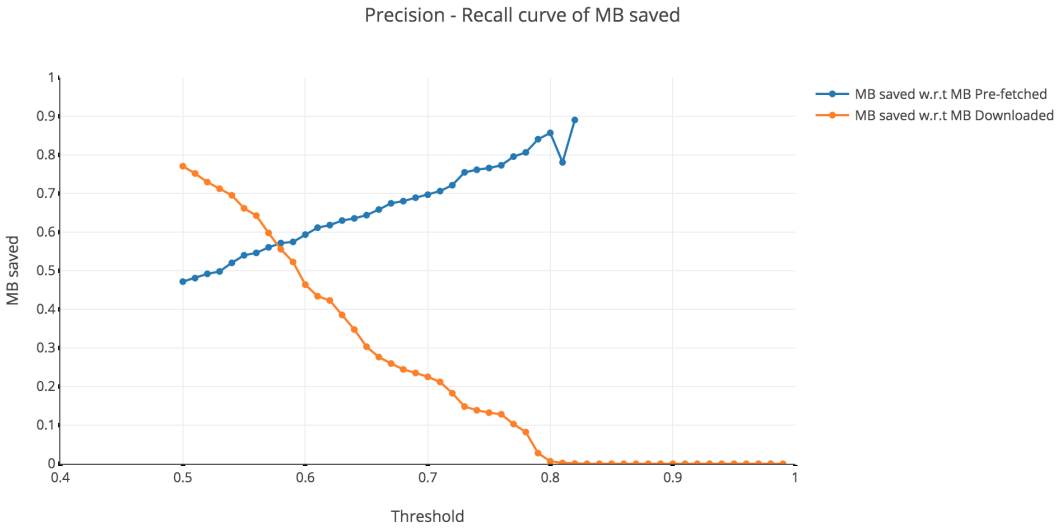

Figure 5.18: Stage 2 : Monthly Prediction - Precision Recall Curve for MB saved

5.4.2.2 Observations

1. Evaluation on Weekly Prediction
   (a) True Byte Rate (TBR) ranged from 26% to 70%
   (b) Optimal Threshold ranged between 0.60 & 0.65 with,
      i. 36%-40% TBR
      ii. 1720-2070 users, out of which 1100-1300 actually downloaded

iii. True Positives : False Positives ratio was 1 : 1.5

2. Evaluation on Monthly Prediction
   (a) True Byte Rate (TBR) ranged from 47% to 89%
   (b) Optimal Threshold ranged between 0.55 & 0.60 with,
      i. 53%-59% TBR
      ii. 2070-2347 users, out of which 1636-1867 actually downloaded
      iii. True Positives : False Positives ratio was 1 : 0.8

5.4.3 Summary

It was observed that the stage 2 model for weekly target could save approximately 4% out of the potential 13% savings discovered during the Exploratory Data Analysis.

6
Related Work

In this section, we present the previous research which are relevant in the area of predictive caching. However, to the best of our knowledge, no work related to predicting the content a user is most likely to download, based on their streaming pattern, using mobile data has been attempted earlier.

Manxing Du Maria Kihl et. al. considered request patterns for TV programs from a popular Swedish TV service provider over 11 weeks to propose a pre-fetching scheme at the user end to pre-load videos before user requests [11]. The idea was to explore the potential of reducing the start-up latency of streaming media services. They demonstrated that the cache hit ratio was significantly improved by the new scheme and customizing it for different video categories could further improve the performance. The proposed system had two components - prefetching engine to foresee what users would watch next before they request the content and the prediction engine to store the video prefix to the local cache. They proposed that in order to implement pre-fetching, the prediction engine needs to determine what content should be pre-loaded based on a user's viewing history of the same TV series. The scheme suggested by the authors examined the user's request patterns of episodes in the series to pre-fetch N adjacent episodes for each viewed episode because they claimed that a TV series consists of a series of episodes which will be released regularly and have high consistency and similarity in content. They considered two extreme cases as baselines for evaluating the pre-fetching performance - pre-fetch all the available episodes in a series to the end user as long as a user watched an episode in that series, or passively cache all user demands without pre-loading prior to user requests. They concluded that 69% of all the requested videos can be correctly predicted using the proposed pre-fetching strategy.

Koch et. al. proposed a proactive caching strategy MIRA(Music Video Information Retrieval for cAching) for music video content [12], which leverages music content features and popularity characteristics. It had two main modules : (i) Music Feature Extractor and (ii) Popularity Predictor. In the first module, a Convolutional Neural Network (ConvNet/CNN) was used to derive music features so as to train a genre and a mood classifier. This classification was then used as an input for two proactive caching policies used to identify content that is likely to be popular next. The second module estimated the future popularity of music videos based on the number of past views. On evaluating, the genre proactive caching policy showed the highest gain measured by the cache hit rate (CHR) with 4.5%. S. Dernbach et. al. explored an approach to determine whether or not a Video Service Provider (VSP) should

use a cache filling policy based solely upon global popularity or consider regional tastes as well [15]. They proposed three metrics for VSPs which could be used to quantify the extent to which regional tastes are present. Their model captured the overlap between inter-regional and intra-regional preferences of movie content.

Koch et. al. presented the significance of new methods to decrease the expensive mobile network challenges [8]. Their work focused on the potential of proactive caching content from the YouTube music categories. A set of user behavior and music specific features were identified from over 4 million music video user sessions for designing new music specific in-network caching policies and a new evaluation strategy for evaluating these new policies were proposed. Different cache size configurations were used for the evaluation purpose. As per their findings, proactive caching is beneficial for large-sized caches with many users and smaller caches. On the other hand, medium-sized caches did not show a positive effect with proactive caching. The paper also investigated the cache performance with varying frequency of proactive caching and amount of input data.

Koch et. al. considered the performance of existing caching strategies with respect to the cache hit rate [9]. They designed a content category-aware caching strategy termed ACDC (Adaptive Content-Aware Designed Cache) where the workload consisted of different categories such as news, comedy and music. They demonstrated that ACDC increased the cache hit rate up to 18.39% and decreased transmission latency up to 12%.

In [13], S. Dutta et. al. proposed a Video-on-Demand(VoD) system, for wireless mobile devices, based on a new caching algorithm called Intelligent Network Caching Algorithm (INCA) which made use of analytics-driven look ahead scheme for both pre-fetch and replacement policies. Hu et. al. formulated the content pre-fetching problem as a Markov Decision Process (MDP), which considered both the server load and users' behavior on TV series watching patterns using real-traces into account, to solve the problem in the online manner [14]. Their proposed reinforcement learning-based algorithm achieved a better precision and hit ratio (e.g., 80%) with about 70% (resp. 50%) cost saving compared to the random (resp. heuristic) algorithm. M. Sun et. al. presented the Predict-then-Prefetch caching strategy in 5G networks whereby the base station capacity is partitioned into a proactive cache and a reactive cache [10]. The former is used to pre-fetch popular content for a sum total maximum of popularity, and the latter to cache content which is unpopular. This strategy improved the hit ratio by 30% and reduced latency by 50% in the architecture of 200M small base stations, thus enhancing the quality of experience.

7
Conclusion

This chapter reflects on the results obtained, discusses the relevance and suggests future course of action.

7.1 Conclusion

This thesis project focused on using machine learning techniques to optimize the mobile data usage by predicting what a user would download in order to pre-fetch it on the device before he actually hits play. The goal was to save 10% of the mobile data downloads for a subset of active users. Through Exploratory Data Analysis (EDA), different use cases were considered to identify the potential candidates that a user is most likely to download through mobile data subscription, and a range of different machine learning models were developed for the potential use cases in this setting. The predictions made by the different models were evaluated and shown to perform better than pre-fetching the top-rated content from the existing recommender system's output, which was the initial hypothesis.

Out of the four scenarios considered for device sync, fixed-devices approach with monthly target was observed to save approximately 2% mobile data downloads out of a total of 13% for the subset of users considered. From the study, the initial use case does seem to be of significant advantage from the business perspective, with the caveat that the target duration of one month must be restricted further, possibly to two weeks. Further, the playlist use case was evaluated for a target of one week and was shown to save approximately 4% mobile data downloads out of a total of 13% for the subset of users considered. These findings show that predictive caching by analyzing user behavior on multiple use cases has the potential of optimizing mobile data usage considerably.

7.2 Future Work

In a wider perspective, predictive caching is an interesting problem where more potential use cases could be targeted through EDA as a possible future work. One such scenario that was identified during the EDA was the user behavior on the contents from the artists they follow. For the use cases that were targeted for this project, we restricted the feature period to a maximum of one month. It would be interesting to use a larger period to train the model on. Further, all these use cases

exhibited a temporal characteristic which could be modeled as a time-series problem using recurrent neural networks.

Bibliography

[1] Carlos A. Gomez-Uribe and Neil Hunt. The Netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), Volume 6 Issue 4, ACM New York, NY, USA, January 2016.

[2] Jin Huang and Charles X. Ling. Using AUC and accuracy in evaluating learning algorithms. IEEE Trans. on Knowl. and Data Eng., 17(3):299– 310, March 2005.

[3] Kurt Jacobson, Vidhya Murali, Edward Newett, Brian Whitman, and Ro- main Yon. Music personalization at Spotify. RecSys '16 Proceedings of the 10th ACM Conference on Recommender Systems, ACM New York, NY, USA, September 2016.

[4] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, Volume 7 Issue 1, 76-80, IEEE Educational Activities Department Piscataway, NJ, USA, January 2003.

[5] Qi Meng. LightGBM: A highly efficient gradient boosting decision tree. 04 2018.

[6] Joanne Peng, Kuk Lida Lee, and Gary M. Ingersoll. An introduction to Logistic Regression Analysis and Reporting. Journal of Educational Research - J EDUC RES, 96:3–14, 09 2002.

[7] David Powers and Ailab . Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness correlation. volume 2, pages 2229–3981. Journal of Machine Learning Technologies, January 2011.

[8] Christian Koch, Ganna Krupii and David Hausheer. Proactive Caching of Music Videos based on Audio Features, Mood, and Genre. In Proceedings of ACM Multimedia Systems (MMSys), 2017

[9] Christian Koch, Johannes Pfannmüller, Amr Rizk, David Hausheer, Ralf Steinmetz, "Category-aware Hierarchical Caching for Video-on-Demand Content on YouTube", In Proc. of ACM Multimedia Systems (MMSys), 2018, pages 89-100

[10] M. Sun, H. Chen and B. Shu, "Predict-then-Prefetch Caching Strategy to Enhance QoE in 5G Networks," 2018 IEEE World Congress on Services (SERVICES), San Francisco, CA, 2018, pp. 67-68.

[11] Manxing Du Maria Kihl, Åke Arvidsson, Christina Lagerstedt, Anders Gavler, "Analysis of Prefetching Schemes for TV-on-Demand Service", 2015 10th International Conference on Digital Telecommunications, Department of Electrical and Information Technology, ELLIIT: the Linköping-Lund initiative on IT and mobile communication, 2015

[12] C. Koch, S. Werner, A. Rizk and R. Steinmetz, "MIRA: Proactive Music Video Caching Using ConvNet-Based Classification and Multivariate Popularity Pre-

diction," 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Milwaukee, WI, 2018, pp. 109-115.

[13] S. Dutta, A. Narang, S. Bhattacherjee, A. S. Das and D. Krishnaswamy, "Predictive Caching Framework for Mobile Wireless Networks," 2015 16th IEEE International Conference on Mobile Data Management, Pittsburgh, PA, 2015, pp. 179-184.

[14] Hu, Wen et al. "Towards Wi-Fi AP-Assisted Content Prefetching for On-Demand TV Series: A Reinforcement Learning Approach." CoRR abs/1703.03530 (2017): n. pag.

[15] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot and A. Ashkan, "Cache content-selection policies for streaming video services," IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, 2016, pp. 1-9.

[16] Aziz, S. and M. Dowling (2019). "Machine Learning and AI for Risk Management", in T. Lynn, G. Mooney, P. Rosati, and M. Cummins (eds.), Disrupting Finance: FinTech and Strategy in the 21st Century, Palgrave, pp 33-50.

[17] Yapo, Adrienne and Joseph Weiss. "Ethical Implications of Bias in Machine Learning." HICSS (2018).

[18] Malhotra, Charm Kotwaf, Vinod Dalai, Surahhi. (2018). Ethical Framework for Machine Learning 1-8. 10.23919/ITU-WT.2018.8597767.

[19] Kalayci, Sacide et al. "Credit risk analysis using machine learning algorithms." 2018 26th Signal Processing and Communications Applications Conference (SIU) (2018): 1-4.

[20] Elizabeth J. O'Neil, Patrick E. O'Neil, and Gerhard Weikum. 1993. The LRU-K page replacement algorithm for database disk buffering. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data (SIGMOD '93), Peter Buneman and Sushil Jajodia (Eds.). ACM, New York, NY, USA, 297-306.

[21] Schedl, Markus Zamani, Hamed Chen, Ching-Wei Deldjoo, Yashar Elahi, Mehdi. (2017). RecSys Challenge 2018: Automatic Playlist Continuation. The International Journal of Multimedia Information Retrieval (IJMIR).

[22] Chen, Tianqi Guestrin, Carlos. (2016). XGBoost: A Scalable Tree Boosting System. 785-794. 10.1145/2939672.2939785.

[23] Ke, Guolin et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." NIPS (2017).