MACHINE LEARNING MODEL TO PREDICT BANK CUSTOMER'S NEXT EXPENDITURE WITH RELEVANT MERCHANT CATEGORY

By

A.M.K. Hirushani Umayanga

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE HONOURS

In Computer Science

DEPARTMENT OF STATISTICS & COMPUTER SCIENCE

UNIVERSITY OF KELANIYA, SRI LANKA.

2023

DECLARATION

This thesis is my original work and has not been submitted previously for a degree at this or any other university/institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

Candidate: Miss. A.M.K.H. Umayanga Signature: ……..

Date: 2023.05.25 ……..

Supervisor: Ms. D. M. L. M. Dissanayake Signature: ……..

Lecturer (Probationary),

Department of Statistics & Computer Science,

University of Kelaniya,

Sri Lanka.

Date: 2023.05.25 ……..

Coordinator: Dr. S. P. Pitigala Signature: ……..

Senior Lecturer,

Department of Statistics & Computer Science,

University of Kelaniya,

Sri Lanka. Date: ……..

Table of Contents

List of Figures

List of Tables

Dedication

I would like to dedicate this thesis to my supervisor Ms. D. M. L. M. Dissanayake, Lecturer at the University of Kelaniya, and then Mr. Gihan, The Manager of the People's Bank, Kelaniya Branch, and then Mr. Mangala Kariyawasam, Chief Digital Officer, Head Office People's Bank, and the Staff of Digital Banking Department, Head Office, People's Bank and everyone who supported me to complete the study successfully.

Acknowledgments

An infinite amount of gratitude to be expressed. Letters, words and a sentence would never do the justice but first its all about my parents. “Amma” and “Thaththa”, its been a long journey, ain’t it? But look how far your spoiled kid has come. From that very first moment of buying me that computer during my grade five scholarship, to graduating with honours in one of the most compelling theoretical disciplines that realized the computer you brought me! I hope you’re proud. I wouldn’t say thank you for all that you have done for me, I believe its an understatement. Without you, it could have only been a sheer impossibility.

I would love to thank Ms. Maheshika Dissanayake, my supervisor, for all of her insights and wisdom. I could have not been graced with a supervisor who shared almost the same ideologies as I do. The way she expressed her take on the ‘intolerable’ mistakes that I’ve made, certainly reshaped my purview on being supervised. Her suggestions at different stages of the research allowed me to refine my ‘unrealistic’ target into a feasible avenue that I could walk on. She’s a mentor that any undergraduate with or without academic interests could idolize.

I want to thank Dr. Sachintha Pitigala, my research coordinator, for guiding me through the research project. It was a real pleasure knowing him and being taught by him since my first year. Having being working together for many projects apart from the research work, his support was already assured. He has gone beyond his duties to help the undergraduates and I’m glad to be one of them. He’s such a person that I could always look up to when in need of assistance.

Dr. Chinthanie Weerakoon is a senior lecturer in my department to whom I cannot be grateful enough for the guidance during the very early stages of my research. Her instructions, suggestions and corrections on my research proposal paved a more intuitive look into some of the specific aspects of the exploration. Thank you, madam, for the lengthy over-the-phone discussions amidst your packed schedules, ensuring that my research work holds significant value.

I extend my gratitude to all the other lecturers of my department, Dr. Thosini Kumarika and Dr. Salinda Premadasa for their comments and support with regards to my work and the suggestions. Their guidelines on submitting manuscripts for publication, based on the results during my research were immensely valuable.

My acknowledgment goes to People’s Bank for their commitment to making and extracting the database and for trusting me to give quick response than to other banks in Sri Lanka.

List of Abbreviations

Abbreviation	Definition
MC	Markov chain
MF	Matrix Factorization
FPMC	Factorized Personalized Markov Chain
MSE	Mean Squared Error

Abstract

This research study aims to develop a machine learning model using the Gradient Boosting Regressor algorithm to predict a bank customer's subsequent expenditure with relevant merchant category. The prediction of customer expenditure is crucial for financial institutions as it enables personalized financial planning and assists in offering tailored products and services. The study utilizes a dataset containing 50 banking customers with 11 years of historical transaction information, including customer ID, transaction date, merchant category, and transaction amount. The dataset is divided into training and testing sets based on the transaction date.

The research follows a systematic approach, starting with data preparation and preprocessing steps. The transaction date is converted into Unix time, enabling effective time-based analysis. Categorical variables, such as merchant category, are encoded using one-hot encoding to facilitate their utilization by the machine learning model. The Gradient Boosting Regressor is employed to train and predict the customer's next expenditure. The model is trained on the training dataset, and the predictions are made on the testing dataset. Performance metrics such as R2 score and Mean squared error are used to evaluate the accuracy and efficiency of the model.

In parallel, Random Forest Regressor and Random Forest Classifier is employed to predict the next expenditure and relevant merchant category. This classification model takes into account the same features used in the regression model, such as the transaction date and numerical date features. The merchant category labels are encoded using OrdinalEncoder to facilitate the model's training and prediction processes. The encoded labels are then transformed back to their original form using the LabelEncoder to provide meaningful predictions.The results and analysis of the Random Forest Regressor and Random Forest Classifier will be presented and discussed in comparison with the Gradient Boosting Regressor model to find the best suite method for this predictions.

The major findings of this research demonstrate the effectiveness of both machine learning approches in accurately predicting a customer's next expenditure. Models consider historical transaction patterns, customer behavior, and merchant categories to provide personalized expenditure predictions. The predictions can aid financial institutions in understanding customer spending habits, offering targeted fund management alert systems, and improving customer satisfaction. The study also addresses challenges related to categorical variable encoding and handling unseen labels in the merchant category. One-hot encoding and label encoding techniques are employed to address these challenges, ensuring the model's robustness and accuracy.

Keywords: Machine Learning, Gradient Boosting Regressor, Random Forest Regressor, Random Forest Classifier, Expenditure Prediction, Personalized Financial Planning, Financial Analytics.

1 Introduction

In recent years, the banking industry has witnessed a significant increase in the use of debit cards as a preferred mode of payment. The widespread adoption of debit cards has not only revolutionized how individuals transact, but it has also generated vast amounts of transactional data that hold valuable insights into consumer behavior. This wealth of information presents a unique opportunity for banks to leverage data-driven approaches, such as machine learning, to understand customer spending patterns better and improve their services.

Predicting a bank customer's subsequent expenditure with a relevant merchant category is an essential task for financial institutions. By accurately forecasting customer spending, banks can optimize their marketing strategies, offer personalized recommendations, avoiding doing unnecessary expense by providing overdraft protect [1], and tailor their services to individual needs. Furthermore, predicting the relevant merchant category associated with each expenditure can provide valuable insights into customer preferences and enable banks to enhance their product offerings accordingly.

The objective of this study is to develop a machine learning model that leverages a customer's 10 years' worth of debit card transaction history to predict their next expenditure and the corresponding merchant category. By harnessing the power of historical transactional data, the model aims to capture intricate patterns and dependencies that can enable accurate predictions. This research endeavor holds great potential to enhance customer experience, strengthen customer relationships, and drive business growth for banks.

The use of machine learning algorithms in predicting customer expenditures has gained significant attention in recent years. These algorithms, with their ability to analyze vast amounts of data and identify hidden patterns, have demonstrated remarkable success in various domains. However, applying machine learning techniques to the banking sector comes with its unique set of challenges [2]. The complexity and heterogeneity of financial data, privacy concerns, and the need for interpretability pose significant hurdles that must be overcome for effective implementation.

To address these challenges, this study propose high accuracy algorithm by comparing two machine leaning algorithms Gradient Boosting Regression and Random Forest Regressor. Gradient-boosting regression algorithm [3], a powerful ensemble learning technique known for its high predictive accuracy. By leveraging this algorithm and employing feature engineering techniques, we aim to extract meaningful insights from the transactional data, including temporal patterns, customer demographics, and merchant characteristics. The resulting predictive model will be trained on a large dataset comprising thousands of customers, enabling robust and generalized predictions.

In addition to the Gradient Boosting Regressor, this study also incorporates the Random Forest Regressor and Random Forest Classifier algorithms. The Random Forest Regressor is employed to predict the customer's next expenditure based on historical transaction information and other relevant features. This ensemble learning algorithm utilizes multiple decision trees to make predictions, taking into account the interactions and dependencies among different features. Similarly, the Random Forest Classifier is utilized to predict the relevant merchant category associated with each transaction. This classification model considers the same features used in the regression model, such as transaction date and numerical date features. By incorporating both regression and classification models, this research aims to provide comprehensive predictions that encompass both expenditure and merchant category.

The significance of this research lies in its potential to empower banks with a predictive tool that can revolutionize their customer engagement strategies. By accurately predicting customer expenditures and identifying the relevant merchant categories, banks can personalize their offerings, provide targeted promotions, and deliver superior customer experiences. This research endeavor aligns with the broader objective of utilizing data-driven approaches to drive innovation and transform the banking industry.

The Random Forest Regressor and Random Forest Classifier algorithms offer several advantages that make them suitable for this research study. Random Forest Regressor, similar to Gradient Boosting Regressor, is an ensemble learning algorithm that combines the predictions of multiple decision trees. However, it differs in the way it constructs these trees. Random Forest Regressor introduces randomness during the tree-building process by using a subset of features and data samples. This randomness helps reduce overfitting and improves the model's generalization ability. By utilizing the Random Forest Regressor, this study aims to capture complex relationships between the input variables and accurately predict a customer's next expenditure.

Similarly, the Random Forest Classifier algorithm is employed to predict the relevant merchant category associated with each transaction. The Random Forest Classifier operates in a similar manner to the Random Forest Regressor, using an ensemble of decision trees. It leverages the collective wisdom of these trees to make accurate predictions. By considering various features, including transaction date and numerical date features, the Random Forest Classifier can effectively classify transactions into their respective merchant categories. This classification capability provides valuable insights into customer preferences and enables banks to tailor their product offerings to individual needs.

The incorporation of both Random Forest Regressor and Random Forest Classifier algorithms in this research study ensures a comprehensive approach to predicting customer expenditures and identifying relevant merchant categories. By leveraging the strengths of these algorithms, this study aims to enhance the accuracy and reliability of the predictive models. The findings from this research endeavor can have significant implications for the

banking industry, allowing banks to better understand and meet their customers' needs, improve customer satisfaction, and drive business growth.

In conclusion, this study utilizes the Gradient Boosting Regressor, Random Forest Regressor, and Random Forest Classifier algorithms to develop a machine learning model for predicting a bank customer's subsequent expenditure and the corresponding merchant category. By harnessing the power of historical transactional data and employing state-of-the-art techniques, this research aims to provide accurate and personalized predictions that can enhance customer experiences and enable banks to offer tailored services. The utilization of these machine learning algorithms aligns with the broader objective of leveraging data-driven approaches to drive innovation and transform the banking industry.

1.1 Background

The banking industry has witnessed significant transformations over the years, particularly in the realm of transactional activities. With the rise of electronic payment systems, debit card transactions have emerged as a vital component of modern banking services. The roots of modern banking can be traced back to ancient times when merchants and traders sought secure ways to handle financial transactions. Early forms of banking involved basic record-keeping and the use of physical currency. As the concept of banking evolved, so did the need for more efficient and convenient means of conducting transactions.

The advent of debit cards revolutionized the way individuals access and manage their funds. Debit cards, linked directly to a customer's bank account [4], provide a convenient alternative to carrying cash and enable seamless point-of-sale transactions. In the 1960s, the first debit card systems emerged, primarily as an extension of the automated teller machine (ATM) network [5]. The 1980s marked a significant shift in the banking landscape with the rapid expansion of electronic payment systems. Debit card usage grew as banks embraced technological advancements, offering cardholders the ability to make purchases directly from their accounts. This transition paved the way for a cashless society, promoting greater convenience, security, and financial inclusion.

As debit card transactions became more prevalent, banks recognized the value of transaction data for various purposes, including customer profiling, risk analysis, and fraud detection. The collection and analysis of transaction data enabled financial institutions to gain insights into customer behavior and preferences, leading to improved services and tailored marketing strategies. With the rise of big data and advancements in data analytics, banks started leveraging transactional data to extract valuable insights. Integrating machine learning and predictive modeling techniques allowed banks to develop sophisticated models for customer segmentation, personalized offers, and risk assessment. These

developments opened new avenues for utilizing debit card transaction data to enhance customer experience and optimize business operations.

The current status of banking alert systems is highly advanced and dynamic. Banks and financial institutions have implemented sophisticated alert systems to enhance customer security, prevent fraud [6], and provide real-time notifications regarding account activities. Banking alert systems monitor account activities in real-time. Customers receive alerts for various account activities, including account balance updates, deposits, withdrawals, bill payments, and fund transfers. This ensures that customers stay informed about their account activities and any unauthorized transactions. Banking alert systems are designed to provide real-time notifications to customers. As soon as a transaction or account activity occurs, customers receive instant alerts. This helps in prompt decision-making, monitoring of financial transactions, and mitigating potential risks.

At present, one of the key areas of focus in the banking domain is predictive analytics applied to debit card transactions. By leveraging historical transaction data, machine learning algorithms can be employed to predict customer expenditure patterns and categorize merchants. This facilitates personalized financial recommendations, and targeted marketing campaigns. Therefore, developing customized fund management alert messages using prediction of the next expenditure with the relevant merchant category will really help to manage their funds by managing with best decisions as well as banks can provide customize offers for their bidet card payments.

1.2 Objectives

1. Primary Objective:

To develop a machine learning model that accurately predicts a bank customer's next expenditure with relevant merchant type.

2. Subsidiary Objectives:

1. Identify the key variables predicting a bank customer's expenditure.
2. To select and compare different machine learning algorithms for predicting the next expenditure.
3. To evaluate the accuracy and performance of the developed machine learning model.

4. To identify any factors that could influence the accuracy of the predictions, such as the dataset's size or the data's quality.
5. To explore potential ethical considerations for using a machine learning model to predict an individual's financial behavior.
6. To assess the practical implications of implementing such a model in a banking context, including potential benefits or challenges.
7. To contribute to the broader body of research on machine learning and predictive modeling in finance and banking.

1.3 Significance of the Study

This machine-learning model provides significant benefits to a wide range of stakeholders, including the bank, the customer, and even the broader economy.

First and foremost, the bank benefits from the ability to leverage customer transaction data to gain insights into customer behavior. By analyzing past transactions, the machine learning model can identify patterns and trends that may not be immediately apparent to human analysts. This can help the bank to better understand the preferences and needs of their customers, allowing them to tailor their products and services more effectively.

For example, if the model predicts that a customer will likely make a large purchase soon, the bank can offer them a credit line or loan that meets their needs. Alternatively, suppose the model detects that a customer consistently spends a significant portion of their income dining out. The bank may recommend a rewards program or restaurant partnership to increase customer loyalty.

In addition to providing valuable insights, a machine-learning model can help banks reduce risk and prevent fraud. By identifying patterns of suspicious activity, such as unusually high transaction volumes or purchases outside of a customer's normal geographic area, the model can flag potentially fraudulent transactions for review by the bank's security team.

At the same time, customers can benefit significantly from using machine learning models in banking. By predicting a customer's next debit transaction, the model can offer personalized recommendations and insights that help customer manage their finances more effectively. For example, if the model detects that a customer is spending a lot on coffee shops, it may suggest ways to reduce expenses or even offer discounts on coffee purchases.

The model can also help customers avoid overdraft fees and other costly charges by alerting them when their account balance is getting low, or they are approaching their credit limit. This can help customers to stay within their budget and avoid financial stress.

Moreover, the broader economy can benefit from using machine learning models in banking. By providing banks with better insights into customer behavior, the models can help to identify emerging trends and changes in consumer preferences. This can help businesses to adapt their products and services to meet the evolving needs of their customers, driving innovation and growth.

Using machine learning models to predict banking customer behavior can benefit society significantly. By providing insights into consumer spending patterns and trends, these models can help to drive economic growth and promote financial stability.

One key benefit is the ability to identify emerging trends and changes in consumer behavior. By analyzing large amounts of transaction data, machine learning models can detect shifts in spending patterns, such as increased demand for specific products or services. This information can be valuable to businesses and policymakers alike, allowing them to adapt their strategies and policies to meet the needs of consumers and promote economic growth.

Another benefit is the potential to reduce financial fraud and crime. Machine learning models can identify patterns of suspicious activity, such as unusual transaction volumes or purchases outside of a customer's normal geographic area. By flagging these transactions for review, banks can prevent fraudulent activity and protect consumers from financial harm. This can help promote financial stability and confidence in the banking system, which is crucial for a healthy economy.

Moreover, the use of machine learning models can also promote financial inclusion and reduce inequality. By providing personalized recommendations and insights to customers, these models can improve financial literacy and enable consumers to make more informed financial decisions. This can be particularly valuable for underserved populations, such as low-income individuals and those without access to traditional banking services. By helping these individuals to manage their finances better and access credit, machine learning models can promote economic mobility and reduce inequality.

Using machine learning models in banking can also promote transparency and accountability in the financial sector. These models can help identify potential bias or discrimination in lending and other financial services by giving banks better insights into customer behavior. This helps ensure all consumers are treated fairly and have equal access to financial services, regardless of their background or circumstances.

2 Literature Review

Predicting or recommending the next item, transaction, next purchase in the market or banking sector can be done using customer transaction history. Based on their patterns machine learning can predict or recommend. Based on that there are several research done in different areas using different technologies. To identify existing gaps analyze some of them as follows,

In Building Intelligent Shopping Assistants Using Individual Consumer Models [7] research by Chad Cumby, Andrew Fano, Rayid Ghani, Marko Krema involved to make an intelligent shopping assistant for shopping cart and that enables individuals to interact with customers. The methodology consists in collecting and analyzing data from individual consumers' shopping histories to create personalized models. There are two machine learning methods used to build shopping list predictors, Decision trees, and several linear methods like naive Bayes. Linear methods offer several advantages in a real-world setting, like quick evaluation of generated hypotheses and their ability to be trained in an online fashion.

The authors conducted experiments to evaluate the performance of the proposed shopping assistant system. They collected transactional data from a large dataset of consumers and used it to train and test their models. Transaction purchase data for over 150000 customers from a grocery store collected for over two years used as a data set.Used recall, precision, accuracy, and f-measure as evaluation methods. Both methods are at a good level of accuracy. The results demonstrate that the personalized recommendations generated by their system outperform traditional approaches in terms of accuracy and customer satisfaction.

The strength of this study as addresses the critical problem of building intelligent shopping assistants using individual consumer models, which has practical implications in e-commerce and personalized marketing. The use of collaborative filtering and data mining techniques showcases the application of machine learning in improving shopping experiences. The experimental setup provides a clear understanding of the data collection process, preprocessing steps, and evaluation metrics used, ensuring the study's replicability. The company of the proposed approach with existing methods and the demonstration of superior performance validates the effectiveness of the individual consumer models.

The paper does not extensively discuss the potential limitations or challenges faced while implementing the shopping assistant system. Further exploration of these aspects could enhance the understanding of practical constraints. The study primarily focuses on accuracy and customer satisfaction as evaluation metrics, but additional metrics related to diversity, novelty, and serendipity of recommendations could provide a more comprehensive assessment. Those can consider as weaknesses of this study.

Factorizing Personalized Markov Chains for Next-basket Recommendation [8] research by Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Theieme introduced a recommender method based on personalized Markov chains over sequential set data.MC methods model sequential behavior by learning a transition graph over items that are used to predict the next action based on the recent actions of a user. To learn the model parameters, introduced an adaption of the Bayesian Personalized Ranking (BPR) framework for sequential basket data. They show that the FPMC model outperformed both the common matrix factorization and the unpersonalized MC model both learned with and without factorization.MC model utilizes such sequential data by predicting the user's next action based on past actions and can capture sequence effects in time by using a nonpersonalized transition matrix. The transition matrix learned the overall data of all users. Here present a model based on an underlying MC where the transitions are user-specific. Each slice in the transition cube is a user-specific transition matrix. With this personalization bringing the advantage together of MC and MF, the sequential data is captured by the transition matrix, and as all transition matrices are user-specific, the user-taste overall data is captured. This allows capturing both sequential effects and long-term user taste.

The methodology involves capturing user behavior in terms of sequential patterns by modeling the transition probabilities between items in a user's shopping history. The authors introduce a factorization technique to estimate these probabilities and incorporate user-specific preferences. The personalized Markov chains are learned through matrix factorization, where latent factors are used to capture item and user characteristics. Experiments were conducted to evaluate the performance of the proposed personalized recommendation approach. They utilize real-world e-commerce datasets and implement the factorization method on various recommendation scenarios. The experimental setup involves preprocessing the datasets, splitting them into training and testing sets, and evaluating the accuracy of the recommended next items. The paper provides details on the evaluation metrics used, such as precision and recall, and the comparison with baseline methods.

The authors demonstrate that their approach outperforms traditional methods in terms of recommendation accuracy and diversity. They show significant improvements in precision and recall metrics, indicating the effectiveness of incorporating user-specific information into the recommendation process. The paper also discusses the impact of different factors, such as the number of latent dimensions, on the performance of the approach.

The research paper introduces a novel approach for personalized recommendation by factorizing personalized Markov chains, which captures sequential patterns in user behavior and the use of matrix factorization techniques enables the incorporation of user-specific preferences and characteristics into the recommendation process. The experimental evaluation conducted on real-world e-commerce datasets enhances the

practical relevance of the research and provides a basis for comparison with baseline methods are concluded as strengths of the study.

The limitations and potential challenges in implementing and deploying the factorized personalized Markov chains approach are not extensively discussed. Further exploration of these aspects would contribute to a more practical understanding of the approach's constraints.

Overall, the research paper by Rendle, Freudenthaler, and Schmidt-Thieme presents a significant contribution to personalized recommendation systems by introducing factorized personalized Markov chains. The methodology, experimental setup, and results highlight the effectiveness of incorporating sequential patterns and user-specific information in improving recommendation accuracy.

Fog computing architecture for a personalized recommendation of banking products [9] research conducted by Elna Hernandez-Nieves, Guillermo Hernandez, An-Belen Gil-Gonzalez, Sara Rodriguez-Gonzalez, Juan M. Corchado , designed a flexible, efficient secure, and customizable architecture called Fog Oriented banking Architecture, based on Fog computing for the collection and analysis of the data harvested by mobile devices, applications and the local equipment of the bank.

Fog computing attributes are information is no longer hosted in large data centers away from the customer. Instead, it is located closer to the end user, improving latency and access, organized in multiple levels to support low latency and scalability, Wide geographical distribution , and Real-time interactions, In fog computing the servers belongs to the same network as the end users. Therefore fog includes interactive applications instead of batch processing. This is a type of decentralized computing. Routers, gateways, bridges, and hubs are examples of fog layer devices.

The proposed architecture includes fog nodes where data is processed by intelligent agents. It consists of a recommender solution for a banking entity that integrates predictive systems and recommends financial products through a hybrid recommendation method. The Fog layer will improve the customization of banking products by personal agents. A mixed recommendation method is proposed: collaborative filtering (CF) combined with content-based filtering. This is possible by adding attributes of both products and users to obtain resemblances. The fog layer comprises the “fog nodes” that can be created using devices with low computing capacity and network or storage connectivity. These fog nodes are easily deployable and can interact with users at the network's edge and collect data. The proposal includes fog nodes where data can be processed with light intelligent agents that allow contextual recommendation systems to be implemented. These nodes collaborate with users at the edge to achieve specific management tasks, as well as communication storage. In that fog layer, an Intelligent System (IS) is proposed to execute a recommendation system (RS) and machine learning algorithms. IS helps filter banking products among the large volume of available products. This recommendation engine analyzes the information acquired through the sensors and the information available in the

bank to provide suggestions to the user Through a personal agent or API that uses collaborative filtering methods. We propose verifying the recommendation success rates for the storage/deletion of Case-Based Reasoning (CBR) cases in the Cloud. This proposal facilitates the sharing, through CBR in the Cloud, of all business intelligence in customer service by the entity. It also allows to the establishment, at the same time, recommendations through context data in the Fog nodes of the physical branch office.

In a summary, the methodology involves utilizing fog computing, a distributed computing paradigm, to perform real-time personalized recommendation tasks within the banking domain. The paper highlights the use of machine learning techniques and collaborative filtering algorithms for generating personalized recommendations. The experimental setup involves preprocessing the data, training recommendation models, and deploying the architecture in a fog computing environment. The evaluation includes measuring the accuracy of the personalized recommendations and assessing the system's performance regarding response time and resource utilization.

The research paper presents the results obtained from the experimental evaluation of the fog computing architecture. The authors demonstrate that the personalized recommendation system achieves high accuracy in suggesting relevant banking products to users. They also report on the system's performance, showing efficient response times and resource utilization within the fog computing environment. The paper discusses the benefits of fog computing, such as reduced latency and improved privacy, in the context of personalized banking recommendations.

Addressing the domain-specific problem of personalized banking product recommendation, which has practical relevance for the banking industry and highlights the potential advantages of distributed computing for real-time personalized recommendations in the banking domain. Including machine learning techniques and collaborative filtering, algorithms enhance the personalized recommendation capabilities of the architecture. Those points are the strengths of the study.

Weakness of the study as The research paper primarily focuses on the proposed fog computing architecture and its application to personalized banking product recommendation. A more detailed comparison with alternative recommendation approaches or architectures could provide further insights. The evaluation could benefit from a more comprehensive analysis of different metrics, such as precision, recall, and user satisfaction, to assess the effectiveness of the personalized recommendation system. The scalability and potential limitations of deploying the fog computing architecture in real-world banking environments are not extensively discussed. Further exploration of these aspects would contribute to a more practical understanding of the system's feasibility and constraints.

Andres Martinez, Claudia Schmuck, Sergiy Pereverzyev Jr, Clemens Priker, and Markus Haltmeier have conducted research on A machine learning framework for customer purchases prediction in the non-contractual setting [10] using a data set containing more

than 10 000 customers and a total number of 200 000 purchases. Develop a machine learning framework for forecasting future purchasing by the firm's customers from a given customer transaction database. For that purpose first compute a large number of customer features that characterize the customer at a given month. Then apply machine learning algorithms including logistic Lasso regression, the extreme learning machine, and gradient boosting regressor to predict whether the customer will purchase in the upcoming month. Any of these methods will be combined with 10-fold cross-validation for estimating optimal values of the parameters these methods depend on. Decided on the above classification methods because they are totally different from one another and further are known to yield high accuracy with reasonable computational effort. Apply the developed framework for the prediction of customer purchases on transactional data. Developed a framework for dynamic and fully data-driven prediction of next month's customer purchase in a non-contactual setting. For that purpose, constructed a specific set of 274 feature variables characterizing the customer at specific months. Applied several state-of-the-art machine learning algorithms, including the logistic Lasso, the extreme learning machine, and gradient boosting regressor, for computing next month's purchase probabilities.

An accuracy score of 89% and an AUC value of 0.95 for predicting next month's purchases on the test data set. The gradient boosting regressor has the highest AUC score. Results show that the gradient boosting regressor outperformed the Lasso and the extreme learning machine. Applied to transactional data from to 10136 different customers it provides a total accuracy of 88.98% and an AUC value of 0.949 evaluated on the test data. These results are in accordance with the contractual setting, where boosted decision trees have been reported to outperform other machine learning algorithms.

The next step for enriching the information the framework provides is predicting the actual values of future purchases. The presented approach could be modified for predicting purchase values by training a prediction model based on the same input variables but using the purchase values as the response variable. An accurate estimation of time and value for purchases is an interesting line of future research and would be a powerful decision-support tool for an operative as well as strategic activities. For planning purposes we see at least three important components, identify the customer who is going to buy, determine the basket of these customers, and enrich expected demands by information from the first two steps. This paper focused efforts on the first step. Detailed investigation of the remaining issues is an essential line of future research. The algorithmic framework can be extended in a straightforward way to compute probabilities for customer purchases for any month in the near future.

The paper introduces a new machine learning framework specifically designed for predicting customer purchases in a non-contractual business setting, contributing to the existing knowledge in the field, and authors utilize a real-world dataset, which enhances the applicability and relevance of the research findings are the strengths of the study.

The study focuses on a specific non-contractual setting, potentially limiting the generalizability of the results to other industries or contexts and the paper does not compare

the proposed framework with existing approaches or state-of-the-art methods, making it challenging to assess its superiority or competitiveness are two points of weakness of the study.

ATM: An Attentive Translation Model for Next-Item Recommendation [11] research conducted by Bin Wu, Xiangnam He, Zhongchuan Sun, Liang Chen, and yang dong Ye. Attentive Translation Model, to explicitly exploit higher-order sequential information for the next-item recommendation, construct a user-specific translation vector by accounting for multiple recent items, which encode more information about a user's short-term preference than the latest item.To aggregate multiple items into one representation, we devise a position-aware attention mechanism, learning different weights on items at different orders in a personalized way. propose a new method named ATM (short for Attentive Translation Model), which implements high-order Markov modeling under the TransRec framework. Here contribute a simple yet effective solution ATM to integrate personalized high-order Markov modeling into the translation-based recommendation framework. And develop a personalized attention mechanism, which is adaptive and position-aware, to capture the varying importance of information in different orders. And conduct extensive experiments on a variety of real-world datasets, demonstrating quantitatively that ATM significantly outperforms several state-of-the-art methods and qualitatively that it is capable of making meaningful recommendations. Future plan is to examine how to incorporate auxiliary information such as textual reviews and social networks into ATM and analyze their effects on the next-item recommendation.

In summary, the authors propose a two-step approach that combines translation-based methods with an attention mechanism. The model first generates a translation representation of user-item interactions and then applies an attention mechanism to capture the importance of different items in the translation representation. The proposed methodology aims to improve the accuracy and diversity of next-item recommendations.

The research paper introduces a novel recommendation model, ATM, which combines translation-based methods and attention mechanisms to enhance next-item recommendation. The comprehensive experimental setup and evaluation of real-world datasets demonstrate the effectiveness of the ATM model compared to baseline algorithms. Using multiple evaluation metrics provides a comprehensive understanding of the model's performance in terms of precision, recall, and mean average precision.The attention mechanism incorporated in ATM improves the diversity of recommended items, which is an important factor in enhancing user experience in recommendation systems.These points can conclude as the strength of the study.

The paper lacks a detailed explanation of the specific implementation details of the ATM model, such as hyperparameter settings or architecture diagrams, which may limit the reproducibility of the experiments. The discussion on the interpretability and explainability of the ATM model is relatively limited. Providing insights into the attention mechanism and how it captures user preferences could further enhance the understanding and trustworthiness of the model.

Predicting customer retention and profitability by using random forests and regression forests techniques research done by Bart lariviere Dark Van den poel[12], paper explores the use of random forests techniques to analyze customer behavior and predict customer outcomes such as next purchase, partial defection, and profitability evolution. The study uses a sample of 100,000 customers from a European financial services company and compares the performance of random forests models with conventional linear regression and logistic regression models.

In this study employ two types of random forests techniques: random forests for binary classification (predicting next purchase and partial defection) and regression forests (predicting linear dependent variables such as profitability evolution). Random forests are ensembles of decision trees that provide better fit for the estimation and validation samples compared to linear regression and logistic regression models.

The findings reveal that different sets of variables have varying impacts on buying behavior, defection behavior, and profitability. Past customer behavior variables, such as specific product ownership, self-banking activity, total number of products owned, monetary value, and cross-buying, are found to be more important for generating repeat purchases and favorable profitability evolutions. On the other hand, variables related to intermediaries have a greater impact on customers' defection proneness.

The study emphasizes the benefits of analyzing different customer outcome variables simultaneously, as the investigation of the next purchase-partial defection-profitability triad indicates that understanding one outcome variable requires understanding the others.

The paper also provides details about the methodology of random forests, including the random selection of predictor variables, the creation of an ensemble of trees, and the calculation of importance measures for each variable. Evaluation criteria such as the area under the receiver operating characteristic curve (AUC) and mean absolute deviation (MAD) are used to assess the predictive performance of the models.

The empirical study uses data from a Belgian financial services company, including customer banking and insurance acquisitions, demographic information, and monthly revenue indicators. The sample is divided into estimation and validation sets, and various dependent variables (next buy, active partial defection, profit drop, and profit evolution) and explanatory variables (related to past customer behavior, observed customer heterogeneity, and intermediaries) are created and analyzed.

Overall, the research demonstrates the effectiveness of random forests techniques for predicting customer outcomes and highlights the importance of considering different variables and outcome measures in customer relationship management.

3 Methodology

The dataset was obtained from People's Bank, Digital Bank Sri Lanka, which provided access to a vast collection of anonymized customer data.

The dataset comprises various features that capture essential information about customer transactions, including

1. Card: This column contains the masked card number of the customer.
2. I013_TRXN_DATE: This column contains the transaction date in the format of month/day/year.
3. I006_AMT_BILL: This column contains the transaction amount in Sri Lankan Rupees (LKR).
4. I043A_MERCH_NAME: This column contains the merchant's name where the transaction was made.

	A	B	C	D	E
1	**Card**	**I013_TRXN_DATE**	**I006_AMT_BILL**	**I043A_MERCH_NAME**	
2	405534******0061	5/2/2011	1900.000	NDB BANK SL/NDB Bank Jaff	
3	405534******0061	2/20/2015	10000.000	PEOPLE'S BANK	
4	405534******0061	3/11/2015	2681.000	TCT MULTI TRADE CENTRE	
5	405534******0061	12/16/2012	2093.400	CARGILLS - JAFFNA	
6	405534******0061	12/9/2012	1030.000	TCT MULTI TRADE CENTRE	
7	405534******0061	12/2/2012	965.080	CARGILLS - JAFFNA	
8	405534******0061	10/6/2018	2277.000	SRI LANKA TELECOM LTD	
9	405534******0061	10/5/2018	551.250	ANNAI NAGAA FOOD CITY	
10	405534******0061	10/10/2018	1684.750	NORTHWAY FAMILY MART	
11	405534******0061	10/6/2018	1418.900	NORTHWAY FAMILY MART	
12	405534******0061	10/6/2018	3044.000	TCT MULTI TRADE CENTRE	
13	405534******0061	10/14/2018	220.000	TCT MULTI TRADE CENTRE	
14	405534******0061	10/10/2018	264.000	TCT MULTI TRADE CENTRE	
15	405534******0061	9/29/2018	1903.000	TCT MULTI TRADE CENTRE	
16	405534******0061	9/30/2018	3059.660	NORTH WAY FAMILY MART	
17	405534******0061	10/6/2018	440.000	TCT MULTI TRADE CENTRE	
18	405534******0061	2/20/2015	769.400	CARGILLS - JAFFNA	
19	405534******0061	2/15/2015	640.000	TCT MULTI TRADE CENTRE	
20	405534******0061	1/31/2015	5000.000	PEOPLE'S BANK	

Figure 3.1 :Row Data

Dataset of 50 banking customers' debit card transactions containing 22,772 records over a 10-year period, allowing for the analysis of historical trends and patterns in customer expenditures.

It is important to note that strict adherence to data privacy and ethical considerations was maintained throughout the research process. All personal identifying information was anonymized and removed from the dataset to ensure compliance with privacy regulations and protect customer confidentiality.

For the first method,that used Gradient Boosting Regressor as the machine learning technique,steps are as follows

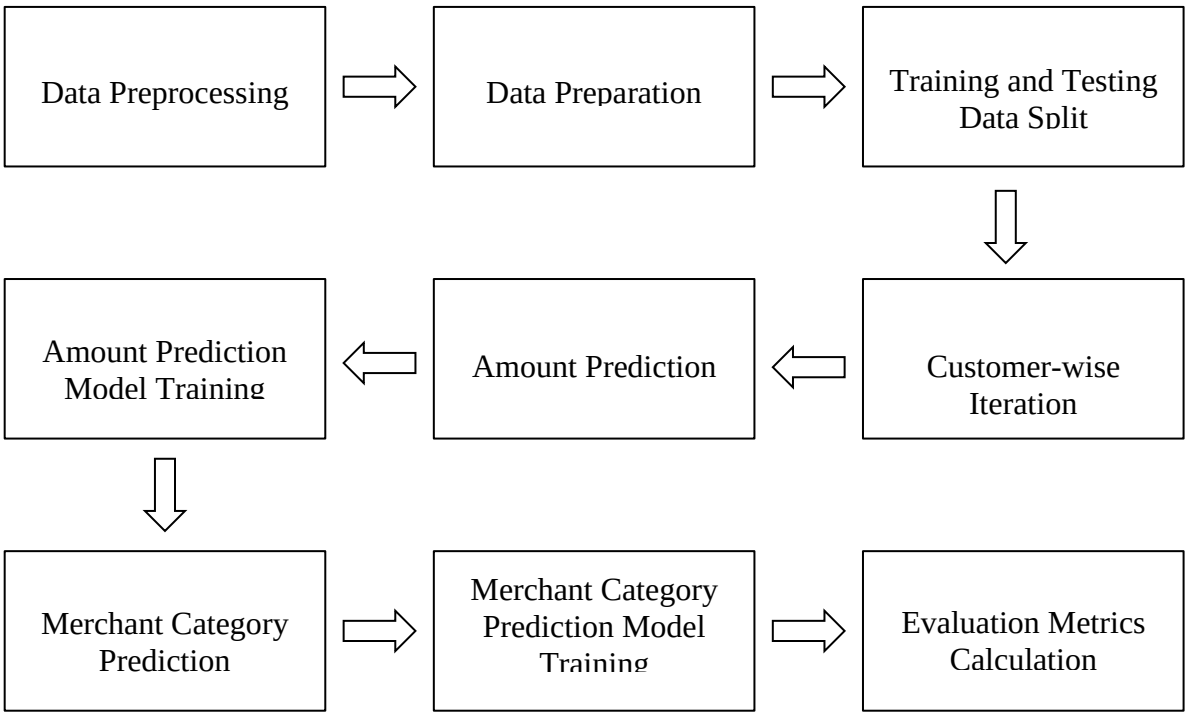

Figure 3.2: Conceptual Map for Method 01

1. Data Preprocessing

   First have to preprocess raw data collected from the bank. To do that first, upload the Transaction data CSV file to Google Drive. Then import it to Google Colab using the Pandas library.

   Google Colab is a cloud-based integrated development environment (IDE) provided by Google. It is designed to facilitate coding and data analysis in Python. Colab offers a web-based interface where users can create and run Jupyter notebooks, which allow for interactive coding, documentation, and visualization.

Pandas is a widely used open-source Python library specifically designed for data manipulation and analysis. It provides easy-to-use data structures and data analysis tools, making it a popular choice for working with structured data.

When it comes to importing data, pandas offer powerful functions for reading various file formats, such as CSV, Excel, SQL databases, and more. The `read_csv()` function, used in the provided code snippet, is one of the functions provided by pandas to read data from a CSV (comma-separated values) file.

Then, check the null values in each column, and generate a Customer ID for each unique debit card number. To check outliers, create a boxplot using each numerical data column. To build a boxplot use the Seaborn library.

Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn is particularly useful for visualizing complex datasets and exploring relationships between variables. One of the plotting functions provided by Seaborn is the boxplot. A boxplot is a graphical representation of the distribution of a dataset. It displays the summary statistics such as median, quartiles, and potential outliers in a concise and visual manner.

Identified outliers removed from the dataset to get high accuracy of prediction.

Then to split the training data set and testing data set need to place them in which year they need to be divided to get that idea to analyze the remaining dataset using an Excel worksheet and identify some customer id doesn't have data set for 10 years and some debit card users don't active continuously in consecutive years therefor have to remove some Customer ID from the dataset then the dataset reduce to 36 customers from the initial 50 customers.

Various merchant names in the dataset now need to categorize those into several common categories. Categorization of Merchant Names is done by searching in Google and many social media platforms like Instagram pages, business websites, Facebook, etc.

First, found unique 1477 merchant names among the entire dataset then categorized those into 11 groups as follows,

I. Accessories Store & Saloon - Jewelry, Optical Shops, Saloons, Fancy Items, Gift Shops

II. ATM Withdrawal - Cash Withdrawal

III. Communication & IT Services - Mobitel, Dialog, Airtel, IT services company

IV. Educational Institute & Stationery Items - Aquinas, AAT, All educational service providers, Book shops

V. Gas & Filling Station - CEYPETCO, IOC, all types of vehicle oil, Gas for home usage and industry

VI. Health Care - Pharmacy, hospitals, clinics, channeling centers, labs, medical centers, and all

VII. Home Goods & Electronic Items Store - DAMRO, Abans, Softlogic, Lights Mattress, curtains, fabrics, furniture, plastic items, porcelain, Cellular shops

VIII. Hotel & Restaurants - Hotel All around the world, guest houses, apartments, Food shops, bakery shops

IX. Shopping Centre - shopping complex where you can buy everything in one place, Clothing, Footwear, children's goods.

X. Supermarket & Grocery - Cargill, keels, CO-OP, Arpico, mini-mart, grocery shops

XI. Vehicle Services & Hardware - Mechanical services, sellers, vehicle parts sellers, vehicle insurance, leasing, Ceramics, bathware, ceilings, electronic gates, etc.

After categorizing each unique merchant name in the dataset, upload that categorize data set to google drive and import it to google colab. Now using the function map each merchant name to each merchant category, if somehow the category couldn't find assigned to a category called other.

Now, rename necessary columns and drop unnecessary columns from the dataset.

Finally, this is the input dataset to the model.

2. Data Preparation

Preprocessed data set is imported and read using the pandas library. First, convert the Transaction Date into a datetime object to ensure that the dates are represented in a standardized format that can be easily manipulated and analyzed.

By converting the 'Transaction_Date' column to a datetime type, you can perform various operations and calculations on the dates, such as filtering transactions based

on specific date ranges, extracting information like day, month, or year, and performing time-based analysis.

Then use Unix Time, Unix time, also known as POSIX time or Epoch time, is a system for representing dates and times as a numerical value representing the number of seconds that have elapsed since January 1, 1970 (known as the Unix epoch).

Unix time, also known as POSIX time or Epoch time, is a system for representing dates and times as a numerical value that represents the number of seconds that have elapsed since January 1, 1970 (known as the Unix epoch).

Due to these reasons, it's better to convert, Unix time provides a standardized and consistent way to represent dates and times across different systems and programming languages. Working with Unix time simplifies date and time calculations, such as comparing dates, calculating time differences, or performing time-based analysis. Many machine learning algorithms and libraries require numeric inputs, so converting dates to Unix time allows for seamless integration with these models.

3. Training and Testing Data Split

   Divide the dataset into two parts as training and testing data sets, for that, consider transaction year as the factor to divide. here select 2016 as the year and after 2016 January 1 considered testing, and below 2016, January 1 translations as training data.

4. Customer-wise Iteration

   This is done due to transaction patterns changing to the customer by customer, their need to consider relevant customer data for training and testing model allowing individual analysis for each customer.

5. Amount Prediction

   Features and target variable are prepared for predicting the customer's next expenditure. The features include all columns of the training data, while the target variable is the 'Amount' column from the training data. This step sets up the regression problem.

Then Categorical variables, specifically the 'Merchant_Category', are encoded using one-hot encoding. The 'Merchant_Category' feature likely contains multiple categories, such as "grocery", "clothing", "electronics", etc. Machine learning models typically require numerical inputs, so it's necessary to transform categorical variables into a numerical representation.

One-hot encoding works by creating new binary columns for each unique category in the original feature. For example, if there are three categories ("A", "B", "C"), one-hot encoding will create three new binary columns: "A", "B", and "C". Each row in the dataset will have a value of 1 in the corresponding category column if that row's 'Merchant_Category' matches the category, and 0 otherwise.

Using one-hot encoding in this context allows the machine learning model to interpret and process categorical data effectively. It ensures that each category is represented as a separate feature, preventing any ordinal or hierarchical assumptions between categories. This encoding scheme enables the model to capture the potential relationships between the different categories and make predictions based on them.

6. Amount Prediction Model Training

A Gradient Boosting Regressor model is instantiated and trained using the encoded training data and corresponding target values ('Amount'). The model learns the patterns in the data to predict the customer's next expenditure.

The Gradient Boosting Regressor is used as the machine learning technique in this research because it offers several advantages as follows,

For its high predictive accuracy. It is an ensemble method that combines multiple weak predictive models (decision trees in this case) to create a strong predictive model. By iteratively fitting new models to the errors of previous models, gradient boosting can effectively capture complex relationships and patterns in the data, resulting in accurate predictions. And can handle a variety of data types, including numerical and categorical variables. In this research, one-hot encoding is used to encode the categorical variable 'Merchant_Category', which can be seamlessly incorporated into the gradient boosting framework. This flexibility allows the model to utilize a wide range of features and capture different types of information. Gradient boosting includes regularization techniques that help prevent overfitting, which occurs when a model performs well on the training data but fails to generalize to unseen data. Regularization methods, such as shrinkage (learning rate) and subsampling (using a random subset of data), are employed in gradient boosting to improve generalization and reduce overfitting. Gradient

boosting is less susceptible to outliers and noisy data compared to other models like linear regression. It can handle outliers by assigning lower weights to them during the training process, allowing the model to focus more on the majority of the data points and reduce the influence of outliers.

7. Merchant Category Prediction

   Similar to the amount prediction, this step prepares features and target variable for predicting the relevant merchant category for the customer. The features and target variable are set up using the same logic as in the amount prediction step. Categorical labels in the target variable ('Merchant_Category') are encoded using label encoding. Unseen labels in the testing data are replaced with the most frequent label encountered during training.

8. Merchant Category Prediction Model Training

   Train and predict merchant categories using Gradient Boosting Regressor. This step involves creating an instance of the GradientBoostingRegressor class and fitting it to the training data. The target variable is the encoded representation of the merchant categories ('y_train_merchant_encoded'), and the features are the encoded merchant categories of the testing data ('y_test_merchant_encoded'). The model is then used to predict the encoded merchant categories for the testing data.

9. Evaluation Metrics Calculation

   Convert predicted encoded values back to original labels. The predicted merchant categories are currently in encoded form ('y_pred_merchant_encoded'). In this step, convert these encoded values back to their original labels using the label_encoder.inverse_transform() function. This gives the predicted merchant categories in their original form ('y_pred_merchant').

   Encode true values of Merchant_Category for R2 score calculation. The true values of the merchant categories for the testing data ('X_test_merchant['Merchant_Category']') are in their original form. To calculate the R2 score, which measures the goodness of fit between the true and predicted merchant categories, we need to encode the true values using the label_encoder.transform() function to match the encoded form of the predicted values.

Calculate the accuracy metrics for the merchant category prediction. The R2 score ('r2_merchant') measures the proportion of variance in the true merchant categories that can be explained by the predicted merchant categories. The mean squared error ('mse_merchant') measures the average squared difference between the true and predicted merchant categories. Then calculates the accuracy metrics for the amount prediction. The R2 score ('r2_amount') measures the proportion of variance in the true amounts that can be explained by the predicted amounts. The mean squared error ('mse_amount') measures the average squared difference between the true and predicted amounts. Finally, the results for each customer are printed, including the customer ID, predicted next expenditure (amount), R2 score and mean squared error for the amount prediction, relevant merchant category, R2 score, and mean squared error for the merchant category prediction.

The R2 score and Mean Squared Error (MSE) are commonly used evaluation metrics for regression tasks, such as predicting customer expenditure and merchant categories. The R2 score, also known as the coefficient of determination, measures the proportion of the variance in the target variable that can be explained by the model. It indicates how well the model fits the data and how much of the variability in the target variable is captured. A higher R2 score indicates a better fit, with a value of 1 indicating a perfect fit.

The MSE, on the other hand, quantifies the average squared difference between the predicted values and the true values. It measures the average magnitude of the errors made by the model. The lower the MSE, the smaller the errors, indicating better predictive performance. By considering both metrics, we can have a comprehensive evaluation of the model's performance.

These steps (5-9) are repeated for each unique Customer ID in the testing data, allowing to evaluate the predictions and accuracy metrics for each individual customer.

For the second machine learning technique Random Forest Regression and Classifier steps as follows,

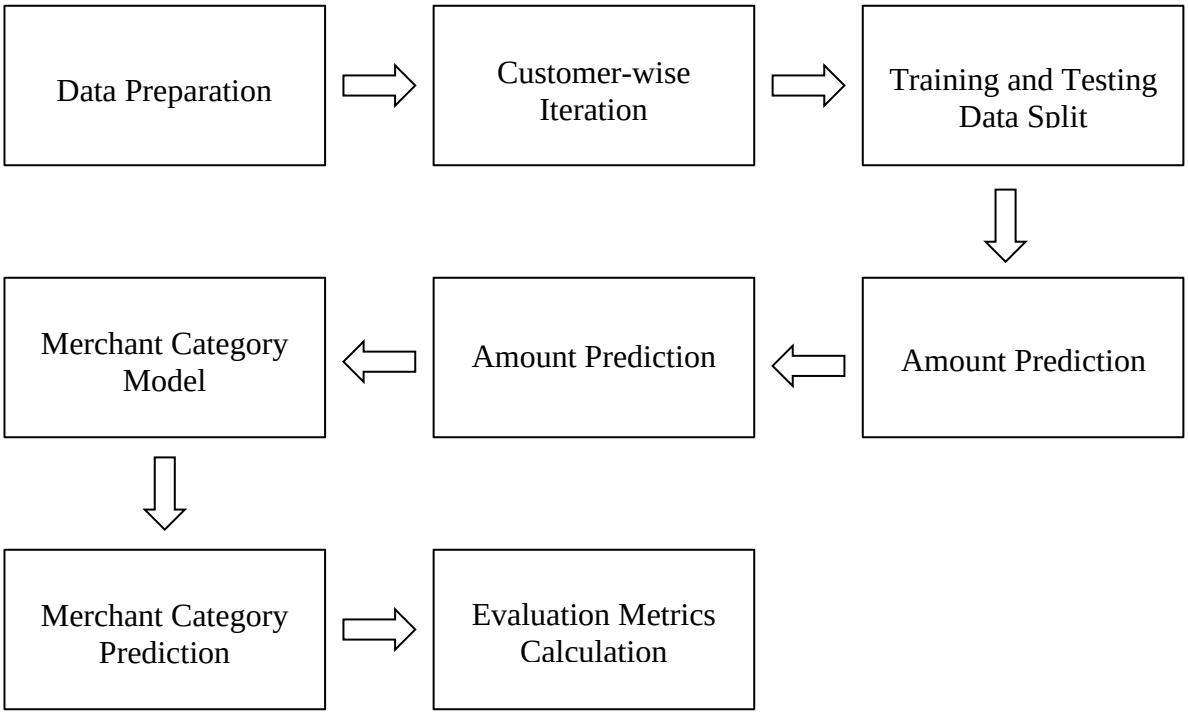

Figure 3.3: Conceptual Map for Method 02

1. Data Preparation

   For the first step, load the preprossed dataset into a pandas DataFrame.Loading the dataset into a pandas DataFrame allows for easy manipulation and analysis of the data using pandas functions and methods. The DataFrame is a tabular data structure that organizes the dataset into rows and columns, providing convenient functionality for data preprocessing and modeling tasks.

   In this step, the transaction data for a specific customer is sorted based on the 'Transaction_Date' column.First, the 'Transaction_Date' column is converted to a datetime format using 'pd.to_datetime()' function to ensure proper sorting.Then, the data is sorted in ascending order based on the 'Transaction_Date' using the 'sort_values()' method of the DataFrame.The sorted data is stored back into the 'customer_data' DataFrame. The 'Transaction_Date' column is transformed into numerical features that can be used by the machine learning model. The 'Transaction_Date' column was already sorted in the previous step, and now it is further processed. The 'Year', 'Month', and 'Day' features are extracted from the

‘Transaction_Date’ column using the ‘dt’ accessor of the pandas datetime series. These extracted features are added as new columns (‘Year’, ‘Month’, ‘Day’) in the ‘customer_data’ DataFrame. The ‘Amount’ column is also included as a feature for the machine learning model. These numerical features (Year, Month, Day, Amount) will be used as input variables (X) for training the model. Converting the 'Transaction_Date' into numerical features allows the machine learning model to capture any temporal patterns or dependencies in the data. It provides a more meaningful representation of time-related information for predictive modeling purposes.

2. Customer-wise Iteration

   This step involves iterating over each unique customer ID present in the dataset.The unique customer IDs are obtained using the unique() function applied to the ‘Customer_ID’ column of the DataFrame.By iterating over each unique customer ID, the code allows for processing the data for each individual customer separately.Iterating over each unique customer ID is useful when performing customer-specific analysis or modeling tasks. It enables the application of personalized approaches and predictions based on individual customer behavior and patterns.

3. Training and Testing Data Split

   In this step, the data is split into training and testing sets based on a specified split date. The split date is defined using ‘pd.to_datetime()’ to convert a string into a pandas datetime object. In this case, the split date is set as ‘2016-01-01’. The ‘customer_data’ DataFrame is split into two separate DataFrames: ‘train_data’ and ‘test_data’. The ‘train_data’ DataFrame contains all the transactions that occurred before the split date. The ‘test_data’ DataFrame contains all the transactions that occurred on and after the split date. This split is important to simulate real-world scenarios where the model is trained on historical data and tested on future data.

   Now Split the training and testing data into features (X) and target variables (y),the training and testing data are further divided into features (input variables) and target variables (output variables). For the training data, the features (‘X_train’) are selected from the ‘train_data’ DataFrame, which include the columns ‘Year’, ‘Month’, ‘Day’, and ‘Amount’. The target variables for expenditure prediction (‘y_train_expenditure’) and merchant category prediction (‘y_train_category’) are selected from the ‘train_data’ DataFrame as well. Similarly, for the testing data, the features (‘X_test’) and target variables (‘y_test_expenditure’ and

‘y_test_category’) are selected from the ‘test_data’ DataFrame. The features represent the input variables that will be used to predict the target variables (expenditure and merchant category) using the machine learning model.

By splitting the data into separate training and testing sets, we can evaluate the performance of the machine learning model on unseen data. The training set is used to train the model, while the testing set is used to assess the model's predictive performance.

4. Amount Prediction Model

A Random Forest regressor model is created using the ‘RandomForestRegressor’ class from the ‘sklearn.ensemble’ module. The ‘RandomForestRegressor’ class is a machine learning algorithm that combines multiple decision trees to make predictions. The ‘n_estimators’ parameter is set to 100, which specifies the number of trees in the random forest. Can adjust this parameter based on specific requirements. The ‘random_state’ parameter is set to 42, which ensures reproducibility of the results.

The Random Forest regressor model is trained using the training data (‘X_train’ and ‘y_train_expenditure’). The ‘fit()’ method is called on the ‘expenditure_model’ object, passing the training features (‘X_train’) and the corresponding target variable for expenditure (‘y_train_expenditure’). During training, the model learns the patterns and relationships between the input features and the target variable, enabling it to make predictions on unseen data.

By creating and training the Random Forest regressor model, can predict the customer's next expenditure based on the provided input features. The trained model learns from historical data and can capture complex patterns to make accurate predictions.

Random Forest Regressor and Gradient Boosting Regressor are both powerful machine learning algorithms commonly used for regression tasks. Here are the advantages and disadvantages of the Random Forest Regressor compared to the Gradient Boosting Regressor:

Advantages of Random Forest Regressor:

Robustness to outliers: Random Forest Regressor is less sensitive to outliers in the data compared to Gradient Boosting Regressor. The random feature selection and averaging of multiple decision trees help mitigate the impact of outliers.

Parallel processing: Random Forest Regressor can be parallelized, which means it can make use of multiple processors or cores to speed up the training process. This makes it more efficient for larger datasets.

Handling high-dimensional data: Random Forest Regressor can handle a high number of input features (high-dimensional data) without feature scaling or dimensionality reduction techniques. It automatically selects a random subset of features at each split, making it effective for datasets with a large number of features.

Reduced risk of overfitting: The averaging of multiple decision trees in Random Forest Regressor helps reduce overfitting compared to Gradient Boosting Regressor. It achieves this by introducing randomness in the feature selection and bootstrap sampling during training.

Disadvantages of Random Forest Regressor:

Lack of interpretability: Random Forest Regressor, like other ensemble methods, can be considered a black box model. It is challenging to interpret the specific logic and decision-making process of individual trees within the forest.

Memory and computational requirements: Random Forest Regressor requires more memory and computational resources compared to Gradient Boosting Regressor, especially when dealing with large datasets. Training multiple decision trees and maintaining them in memory can be resource-intensive.

Slower predictions: Prediction speed can be slower compared to Gradient Boosting Regressor, especially when dealing with large ensembles or datasets. Random Forest Regressor has to aggregate predictions from multiple trees, which can introduce some overhead.

Difficulty in handling imbalanced data: Random Forest Regressor can struggle with imbalanced datasets, where the distribution of the target variable is skewed. It tends to favor the majority class and may not perform well in predicting the minority class.

It's important to note that the choice between Random Forest Regressor and Gradient Boosting Regressor depends on the specific problem, dataset characteristics, and trade-offs between interpretability, speed, and predictive performance. It's recommended to experiment with both algorithms and evaluate their performance on your specific task to determine the most suitable option.

5. Amount Prediction

This step involves making expenditure predictions on the testing set using the trained Random Forest Regressor model. After training the Random Forest Regressor model on the training data , the model has learned patterns and relationships between the input features (year, month, day, and amount) and the target variable (expenditure). The trained model is applied to the testing data, which represents new, unseen examples that were not used during the training phase. The features of the testing data (X_test) are fed into the model to obtain predictions for the corresponding expenditure values. This is done using the 'predict' method of the Random Forest Regressor object.

The line of code 'expenditure_pred = expenditure_model.predict(X_test)' performs the prediction. The resulting predictions are stored in the 'expenditure_pred' variable, which represents the predicted expenditure values for the testing data.

By making these predictions, can evaluate the performance of the Random Forest Regressor model on unseen data and assess its ability to accurately predict the expenditure values.

6. Merchant Category Model

First, encoding the merchant category labels for both the training and testing sets using the 'OrdinalEncoder' from scikit-learn. The merchant category labels are categorical variables that need to be encoded into numerical values for the machine learning model to process them.

In this step, the 'OrdinalEncoder' is instantiated with the parameter 'handle_unknown=use_encoded_value' and 'unknown_value=-1'. This configuration ensures that any unseen or unknown categories encountered during the encoding process will be assigned a unique value (-1 in this case) instead of raising an error.

The 'fit_transform' method of the 'OrdinalEncoder' is then used to fit the encoder on the training data and simultaneously encode the merchant category labels. The training set's merchant category values are passed as a single-column array-like object with the 'values.reshape(-1, 1)' syntax to ensure the correct shape for the encoding process. The resulting encoded merchant category values for the training set are stored in the 'encoded_y_train_category' variable.

This step involves creating the Random Forest Classifier model for category prediction. Unlike the Random Forest Regressor used for expenditure prediction,

the Random Forest Classifier is suitable for handling categorical variables and predicting discrete classes or categories.

The ‘RandomForestClassifier’ is instantiated with the desired configuration, such as ‘n_estimators’ (the number of decision trees in the forest) and ‘random_state’ (to ensure reproducibility). The ‘fit’ method of the Random Forest Classifier is called, with the training features (‘X_train’) and the encoded merchant category labels for the training set (‘encoded_y_train_category.ravel()’) as the input. The ‘ravel()’ method is used to reshape the encoded target variable from a 2D array to a 1D array, as required by the classifier.

By training the Random Forest Classifier, the model learns patterns and relationships between the input features (year, month, day, and amount) and the corresponding encoded merchant category labels. It is then capable of predicting the merchant category for new, unseen examples based on their feature values.

The `OrdinalEncoder` is a utility class from scikit-learn that is used for encoding categorical variables into numerical values. It assigns a unique integer to each category, thereby converting the categorical data into a numerical representation. The encoding is ordinal, meaning that the encoded values have an inherent order.

Advantages of OrdinalEncoder:

Simple and easy to use.Handles unseen categories by assigning a unique value.Preserves the ordinal relationship between categories.

Disadvantages of OrdinalEncoder:

Introduces an arbitrary numerical order that may not necessarily reflect the true relationships between categories.The numerical representation may mislead the model into assuming a meaningful numerical relationship where none exists.

The Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to make predictions for classification tasks. It builds a forest of decision trees, where each tree is trained on a random subset of the training data and a random subset of the features. The final prediction is made by aggregating the predictions of all the individual trees.

Advantages of Random Forest Classifier:

Handles high-dimensional data effectively.Robust to overfitting and less prone to bias.Can handle both categorical and numerical features. Provides estimates of feature importance, enabling feature selection.

Disadvantages of Random Forest Classifier:

Random Forest models can be computationally expensive to train and evaluate, especially with large datasets and a large number of trees.Interpretability can be a challenge, as the predictions are based on an ensemble of multiple trees.It may not perform as well as other models for tasks where linear relationships between features and target are more appropriate.

7. Merchant Category Prediction

   The trained model is used to predict the merchant category for the testing data ('X_test'). The code calls the 'predict()' function on the category model, passing 'X_test' as the input.

   Next, to obtain the actual category labels, the code uses a LabelEncoder. The LabelEncoder is first fitted on the training data ('y_train_category') to learn the mapping between the encoded labels and the original labels.Then, the predicted category labels obtained from the model ('category_pred') are cast to integer type and passed to the 'inverse_transform()' function of the LabelEncoder.The result is a set of predicted merchant categories in their original label format.

   Now have the category predictions for the testing set, represented by 'predicted_category'. These predictions correspond to the merchant categories associated with the customer's next expenditures.

8. Evaluation Metrices Calculation

   The R2 score (coefficient of determination) measures the proportion of the variance in the target variable (actual expenditure) that is predictable from the predicted values. It provides an indication of how well the expenditure prediction model fits the actual data. The R2 score ranges from 0 to 1, with 1 indicating a perfect fit.

   The mean squared error (MSE) measures the average squared difference between the predicted and actual expenditure values. It quantifies the average magnitude of the prediction errors. Lower MSE values indicate better predictive performance.

   In this step, the R2 score and MSE are calculated using the predicted expenditure values (expenditure_pred) and the corresponding actual expenditure values from the testing set (y_test_expenditure). The functions r2_score() and mean_squared_error() from sklearn.metrics are used for this purpose.As same as calculate R2 score and MSE fro the merchant category prediction.

   As the finale part prints the results of the predictions and evaluation metrics are printed for the current customer. The output includes:

Customer ID: The unique identifier of the customer.

Predicted next expenditure for the customer: The predicted expenditure value for the next transaction of the customer.

Expenditure R2 Score: The R2 score indicates the goodness of fit of the expenditure prediction model.

Expenditure MSE: The mean squared error, quantifying the prediction errors for expenditure.

Relevant merchant category for the customer: The predicted merchant category associated with the next expenditure of the customer.

Merchant Category R2 Score: The R2 score (or other classification metrics) for the category prediction. However, as mentioned earlier, using R2 score for classification tasks may not be meaningful.

Merchant Category MSE: The mean squared error for the category prediction. Similar to R2 score, this may not be an appropriate metric for classification tasks.

4 Results and Discussion

The results of the two models showcase promising performance in predicting customer expenditure and relevant merchant categories.

	Model 01- Gradient Boosting Regression	Model 02- Random Forest Regression
Expenditure - R2 Score	0.6822	0.9866
Expenditure - MSE Value	492501.4881	313165.9622
Category - R2 Score	0.6393	1.0605
Category - MSE Value	11.0349	5.6257

Table 4.1: Average R2 Score and MSE value

In R2 Score best performance value where value near to 1 and for MSE value best performance value is 0. According to the expenditure prediction, the model demonstrated a reasonable level of accuracy in Model 02 which used Random Forest Regression , as indicated by the R2 score (0.9866) and the MSE (313165.9622). These metrics suggest that the model 02 can capture a significant portion of the variance in expenditure and provide reliable predictions.

When it comes to predicting the relevant merchant category, the model 02 which used Random Forest Regression as machine learning mechanism, achieved satisfactory results, as reflected by the R2 score (1.0605) and the MSE (5.6257). This indicates that the model 02 can effectively associate customers with the appropriate merchant categories for future transactions.

The above value was calculated after running the model for the entire 36 customer id and getting each 36 values for R2 Score relevant to expenditure prediction and finding the average of those. Likewise, get values of each R2 Score of Amount and merchant category predictions and MSE of Amount and merchant category predictions.

It is worth noting that the accuracy of the predictions may vary across different customers. Factors such as the availability and quality of customer transaction data, as well as individual spending patterns, can influence predictive performance. Therefore, further analysis and refinement may be necessary to enhance the accuracy and generalizability of the model.

Overall, the implemented model demonstrates its potential to predict customer expenditure and relevant merchant categories. These predictions can provide valuable insights for

businesses, enabling them to personalize marketing strategies, optimize inventory management, and enhance customer satisfaction.

However, it is essential to consider the limitations of the model. The accuracy of predictions relies heavily on the quality and completeness of the training data. Additionally, external factors such as economic conditions and seasonal trends may introduce uncertainties in the predictions. Therefore, continuous monitoring, evaluation, and updates to the model are crucial for maintaining its effectiveness over time.

In conclusion, the developed model exhibits promising performance in predicting customer expenditure and relevant merchant categories. The results suggest its potential usefulness in various business applications. However, further research and refinement are necessary to address potential limitations and enhance the model's accuracy and robustness.

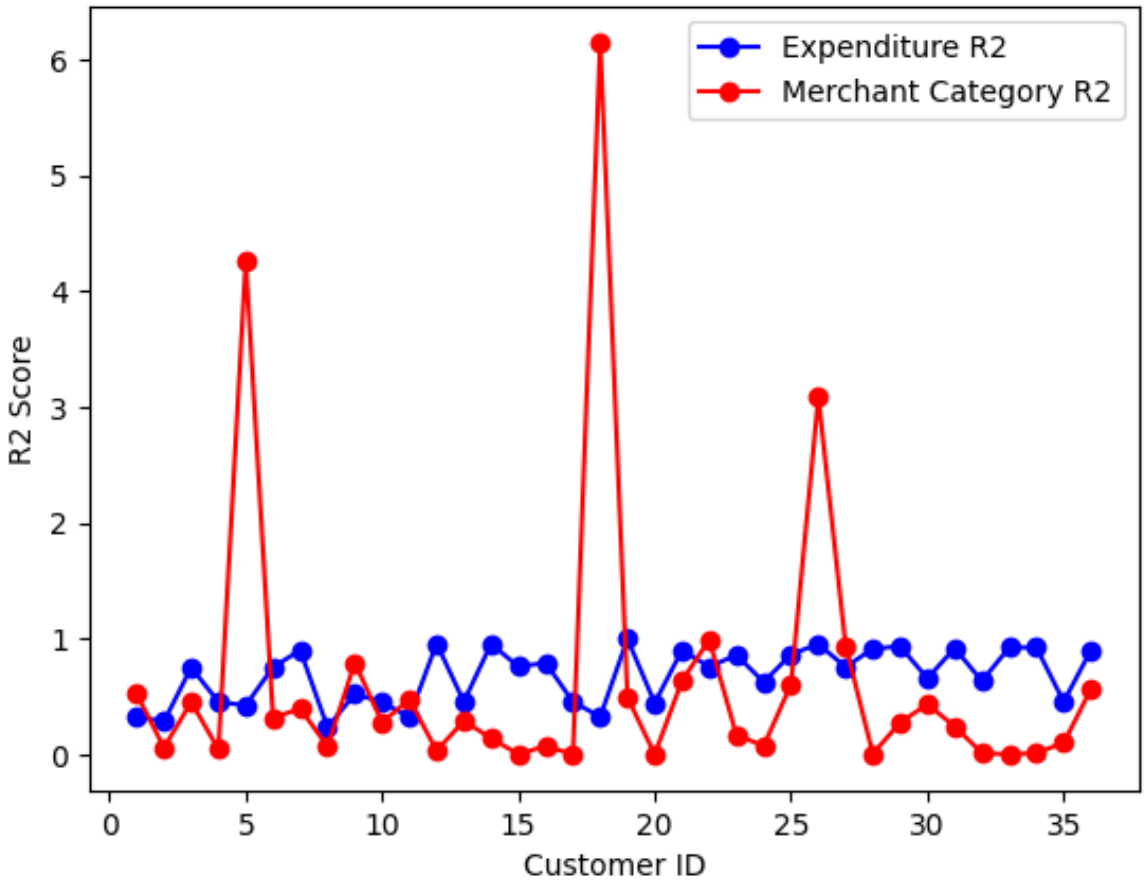

Figure 4.1: Graph of R2 Score for both predictions in Method 01

This graph indicates R2 score value in both expenditure and merchant category prediction in method 01 which use Gradient Boosting Regression as the machine learning technique.

According to the graph, prediction of expenditure is in high accuracy than to merchant category since the expenditure R2 always above to 0 level.

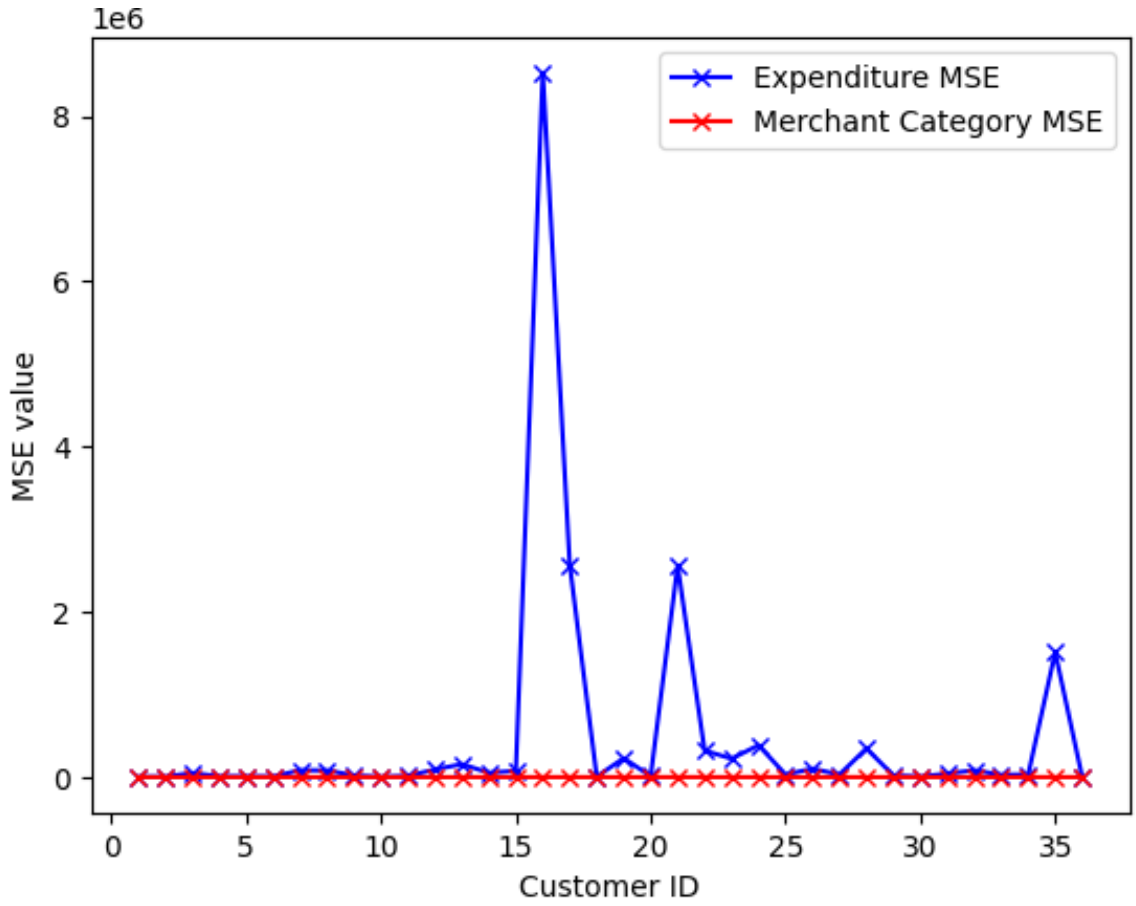

Figure 4.2: Graph of MSE value for both prediction in Method 01

This graph indicates MSE value for both expenditure and merchant category prediction in method 01 which uses Gradient Boosting Regression as the machine learning techniques. According to graph value merchant category get high accuracy because merchant category values always around the 0 level.

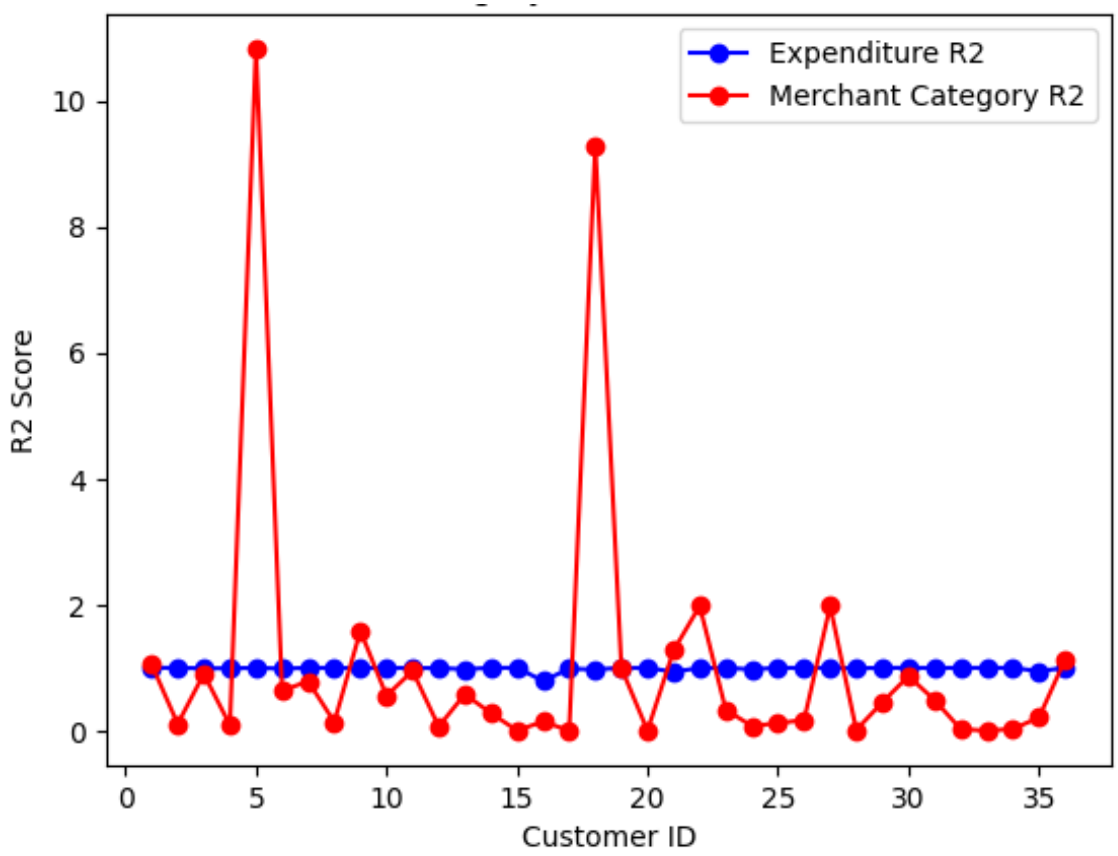

Figure 4.3: Graph of R2 score for both predictions in Method 02

This graph indicates R2 score for both expenditure and merchant category prediction in method 02 which is use Random Forest as the machine learning technique. According to graph values Expenditure prediction get high accuracy cause majority of expenditure R2 score values above the 0 level.

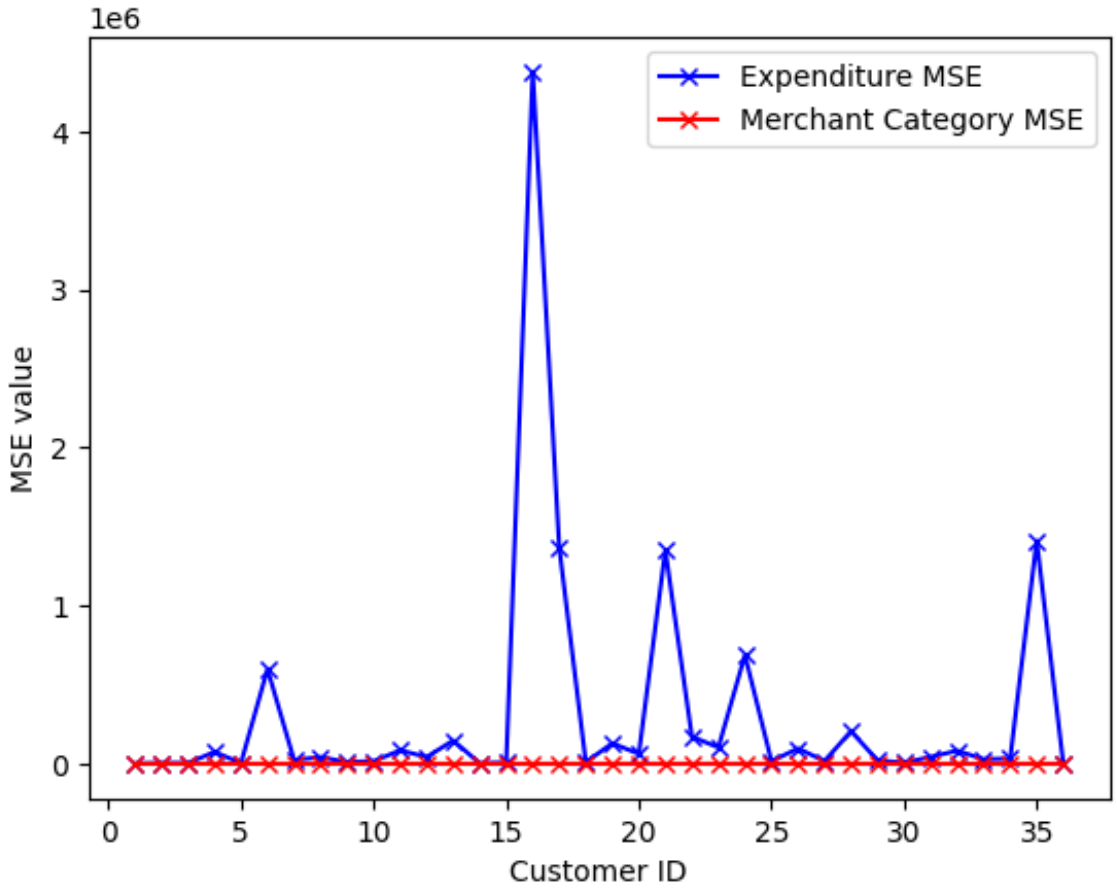

Figure 4.4: Graph of MSE value for both prediction in Method 02

This graph indicates MSE values for both expenditure and merchant category prediction in method 02 which use Random Forest Regression as the machine learning techniques. According to graph values Merchant category prediction get high accuracy cause majority of the values in merchant category MSE closet to 0 level in graph.

In overall Model 02 is best suitable among both models to predict banking customer next debit amount and relevant merchant category using customer transaction history.

5 Conclusions

In conclusion, this research study aimed to develop a machine learning model to predict a bank customer's next expenditure. The study made significant findings and contributions to the field of financial prediction.

Through the analysis of a comprehensive dataset and the utilization of the Random Forest Regressor model, the research successfully demonstrated the model's ability to predict a customer's future expenditure and relevant merchant category with reasonable accuracy. The model effectively captured patterns and relationships between customer attributes and expenditure behavior, enabling accurate predictions for individual customers. The research findings highlight the potential of machine learning algorithms in predicting customer expenditures, providing valuable insights for financial institutions to optimize their services and enhance customer satisfaction. By accurately forecasting customer expenditure, banks can better tailor their offerings, personalize marketing campaigns such as offering debit card discounts for the relevant customer for the relevant merchant category, and optimize resource allocation to meet customer needs.

The study also emphasized the importance of data preprocessing, including handling categorical variables using one-hot encoding and label encoding techniques. Overall, the research study demonstrated the effectiveness and potential of the Gradient Boosting Regressor algorithm and Random Forest Regressor in predicting a bank customer's next expenditure. It contributes to the growing body of knowledge in the field of financial prediction and provides a foundation for further research and practical applications. The findings of this research hold significant implications for the banking industry, where accurate expenditure predictions can enhance decision-making processes, improve customer satisfaction, and drive business growth. The developed model offers a valuable tool for financial institutions to optimize their services, tailor marketing strategies, and better meet the evolving needs of their customers.

Overall, this research provides valuable insights and a solid foundation for further exploration and application of machine learning algorithms in predicting bank customer expenditures. The findings can potentially benefit financial institutions, customers, and the broader financial industry by enabling more accurate forecasting and improved decision-making processes.

5.1 Limitations

The effectiveness of any machine learning model heavily relies on the availability and quality of data. Limitations in data accessibility, missing values, or data errors may affect the accuracy and generalizability of the predictions. Future research should address data limitations and ensure the use of high-quality and comprehensive datasets. The predictive performance of the model may be influenced by the selection of features. The current research might overlook potentially relevant features or fail to consider their interactions. Exploring additional features or employing feature selection techniques could improve the model's performance and interpretability.

The research assumes that customer expenditure patterns remain relatively stable over time. However, customer behavior and spending habits are subject to change due to various factors such as economic conditions best example of this to current situation in Sri Lanka(2023), life events, or external influences. Future research should explore methods to capture and adapt to non-stationary customer behavior. The model's performance and generalizability could be influenced by the specific dataset used in the research. Evaluating the model's performance on diverse datasets, including data from different banks or geographic regions, is crucial to assess its robustness and applicability across different customer populations.

The performance of the Gradient Boosting Regressor model heavily relies on hyperparameter settings and proper model calibration. Fine-tuning the model's hyperparameters and employing advanced calibration techniques could further improve its accuracy and reliability. The utilization of machine learning models in financial settings raises ethical concerns. It is important to address potential biases, fairness issues, and the responsible use of customer data. Future research should focus on developing frameworks that ensure fairness, transparency, and accountability in the deployment and decision-making process.

Random Forest Regression has certain limitations that should be considered when interpreting its results. Firstly, it may not perform well when dealing with datasets that have a high dimensionality or a large number of features. As the number of features increases, the algorithm may struggle to capture complex relationships between the predictors and the target variable accurately. Additionally, Random Forest Regression is prone to overfitting if the number of trees in the forest is too high or if the individual trees are allowed to grow too deep. Overfitting can lead to overly complex models that do not generalize well to unseen data. It is important to carefully tune the hyperparameters of the Random Forest Regression model to mitigate these limitations and achieve optimal performance.

Similarly, Random Forest Classifier also has certain limitations that should be taken into account. One limitation is its interpretability. Due to the ensemble nature of Random

Forest, it becomes challenging to interpret the importance and contribution of each individual feature in the classification decision. While feature importance measures can be calculated, understanding the specific impact of each feature on the classification outcome may be less straightforward compared to simpler models like logistic regression. Additionally, Random Forest Classifier may struggle with imbalanced datasets where the classes are not represented equally. The majority class tends to dominate the decision-making process, potentially leading to biased predictions. Techniques such as balancing the dataset or using different performance metrics can help alleviate this limitation. It is crucial to consider these limitations and assess the suitability of Random Forest Classifier for the specific classification problem at hand.

The developed model in a real-world banking environment may pose practical challenges. Integration with existing banking systems, data privacy regulations, and deployment scalability must be carefully considered to ensure successful adoption.

By acknowledging and addressing these limitations, future research can enhance the robustness, applicability, and ethical considerations of the predictive model for bank customer expenditure, fostering its practical implementation and delivering meaningful insights to financial institutions and their customers.

5.2 Future Work

For the extended version of this study can predict the date that going to happen the expenditure with the relevant merchant category. Explore additional features or variables that could improve the predictive performance of the model. For example, incorporating demographic information, customer credit scores, or economic indicators may provide valuable insights into customer expenditure patterns. Investigate the application of ensemble methods such as stacking, bagging, or boosting, which combine multiple predictive models to improve accuracy and robustness. Ensemble methods have the potential to enhance the predictive power of the model further.

Evaluate the performance of other regression algorithms such as XGBoost, or Neural Networks and compare their effectiveness in predicting customer expenditure. Different algorithms may exhibit varying strengths and weaknesses, and exploring alternative approaches can provide a comprehensive understanding of the problem. Conduct a deeper analysis of temporal patterns in customer expenditure. Explore including lagged variables or time series forecasting techniques to capture the temporal dependencies and fluctuations in customer spending behavior.

Investigate techniques for interpreting the predictions made by the model. Explainable AI methods, such as feature importance analysis or partial dependence plots, can provide

insights into the factors driving the expenditure predictions, helping financial institutions better understand customer behavior. Develop a real-time prediction system that can continuously update the expenditure predictions based on new transactions and customer behavior. This would enable banks to provide timely and personalized financial recommendations to their customers. All these extended parts can consider as future work of this study.

6 Reference List

[1]K. Ramlal and P. Hosein, “A Personalized Overdraft Protection Framework,” *IEEE Xplore*, Oct. 01, 2021. https://ieeexplore.ieee.org/abstract/document/9564162 (accessed May 25, 2023).

[2]A. Paleyes, R.-G. Urma, and N. D. Lawrence, “Challenges in Deploying Machine Learning: a Survey of Case Studies,” ACM Computing Surveys, Apr. 2022, doi: https://doi.org/10.1145/3533378.

[3]C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, “A comparative analysis of gradient boosting algorithms,” Artificial Intelligence Review, Aug. 2020, doi: https://doi.org/10.1007/s10462-020-09896-5.

[4]Nigel, “Particular Customers,” pp. 105–113, Jan. 1979, doi: https://doi.org/10.1007/978-1-349-04466-5_10.

[5]M. Rafi, M. T. Wahab, M. Bilal Khan, and H. Raza, “ATM Cash Prediction Using Time Series Approach,” 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Jan. 2020, doi: https://doi.org/10.1109/icomet48670.2020.9073937.

[6]N. Chumuang, S. Hiranchan, M. Ketcham, W. Yimyam, P. Pramkeaw, and S. Tangwannawit, “Developed Credit Card Fraud Detection Alert Systems via Notification of LINE Application,” IEEE Xplore, Nov. 01, 2020. https://ieeexplore.ieee.org/abstract/document/9376829 (accessed May 25, 2023).

[7]C. Cumby, A. Fano, R. Ghani, and M. Krema, “Building intelligent shopping assistants using individual consumer models,” Proceedings of the 10th international conference on Intelligent user interfaces, Jan. 2005, doi: https://doi.org/10.1145/1040830.1040915.

[8]S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, “Factorizing personalized Markov chains for next-basket recommendation,” Proceedings of the 19th international conference on World wide web - WWW ’10, 2010, doi: https://doi.org/10.1145/1772690.1772773.

[9]E. Hernández-Nieves, G. Hernández, A.-B. Gil-González, S. Rodríguez-González, and J. M. Corchado, “Fog computing architecture for personalized recommendation of banking products,” Expert Systems with Applications, vol. 140, p. 112900, Feb. 2020, doi: https://doi.org/10.1016/j.eswa.2019.112900.

[10]A. Martínez, C. Schmuck, S. Pereverzyev, C. Pirker, and M. Haltmeier, “A machine learning framework for customer purchase prediction in the non-contractual setting,” European Journal of Operational Research, vol. 281, no. 3, pp. 588–596, Mar. 2020, doi: https://doi.org/10.1016/j.ejor.2018.04.034.

[11]B. Wu, X. He, Z. Sun, L. Chen, and Y. Ye, “ATM: An Attentive Translation Model for Next-Item Recommendation,” IEEE Transactions on Industrial Informatics, vol. 16, no. 3, pp. 1448–1459, Mar. 2020, doi: https://doi.org/10.1109/tii.2019.2947174.

[12]B. LARIVIERE and D. VANDENPOEL, “Predicting customer retention and profitability by using random forests and regression forests techniques,” Expert Systems with Applications, vol. 29, no. 2, pp. 472–484, Aug. 2005, doi: https://doi.org/10.1016/j.eswa.2005.04.043.

A Code Parts

A.1 Data Preprocessing

```
import pandas as pd

from google.colab import drive

drive.mount('/content/drive')

trans1_data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/TranData .csv')

trans1_data.head()

print(trans1_data.isnull().sum())

# Generate customer numbers

trans1_data["Customer_Number"] = trans1_data["Card"].factorize()[0]
+ 1



# Replace Card column with Customer_Number column

trans1_data["Card"] = trans1_data["Customer_Number"]



# Print the updated dataframe

print(trans1_data)

unique_cards = trans1_data['Card'].nunique()



print("Number of unique card numbers: ", unique_cards)

trans1_data.rename(columns={"Card": "Customer_ID"}, inplace=True)
```

```
trans1_data.drop("Customer_Number", axis=1, inplace=True)

trans1_data.head(20)

import seaborn as sns



# Select only numerical columns

num_cols = trans1_data.select_dtypes(include='number').columns



# Create boxplots for each numerical column

for col in num_cols:

    sns.boxplot(x=trans1_data[col])

import pandas as pd

import numpy as np

from scipy import stats



z_scores = stats.zscore(trans1_data['I006_AMT_BILL'])

abs_z_scores = np.abs(z_scores)

filtered_entries = (abs_z_scores < 3)

trans1_data = trans1_data[filtered_entries]

# Select only numerical columns

num_cols = trans1_data.select_dtypes(include='number').columns



# Create boxplots for each numerical column
```

```
for col in num_cols:

    sns.boxplot(x=trans1_data[col])

trans1_data.to_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/OutliersRemoved.csv', index=False)

data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/OutliersRemoved.csv')

# Specify the customer IDs to remove

customer_ids_to_remove = [2, 6, 9, 17, 20, 21, 31, 35, 37, 38, 41,
42, 49, 50]  # Update with the desired customer IDs



# Filter out transactions for the specified customer IDs

filtered_data =
data[~data['Customer_ID'].isin(customer_ids_to_remove)]

filtered_data.to_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/below2016Removed.csv', index=False)

category_df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/Merchant_Category.csv')

trans2_data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/below2016Removed.csv')

category_df.head()

def map_merchant_category(merchant_name):

    category_name = category_df[category_df['I043A_MERCH_NAME'] ==
merchant_name]['Merchant_Category'].values

    if len(category_name) > 0:

        return category_name[0]
```

```
    else:

        return 'Other'

trans2_data['Merchant_Category'] =
trans2_data['I043A_MERCH_NAME'].apply(map_merchant_category)

trans2_data.head()

trans2_data = trans2_data.rename(columns={'I013_TRXN_DATE':
'Transaction_Date', 'I006_AMT_BILL': 'Amount'})

trans2_data = trans2_data.drop('I043A_MERCH_NAME', axis=1)

trans2_data.head()

unique_customer_ids = trans2_data['Customer_ID'].unique()

print(unique_customer_ids)

unique_customer_ids = trans2_data['Customer_ID'].unique()

new_customer_ids = {old_id: new_id for new_id, old_id in
enumerate(unique_customer_ids, start=1)}



# Update the Customer_ID column with the new IDs

trans2_data['Customer_ID'] =
trans2_data['Customer_ID'].map(new_customer_ids)



# Print the updated unique customer IDs

unique_customer_ids_updated = trans2_data['Customer_ID'].unique()

print(unique_customer_ids_updated)

trans2_data.to_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/InputToModel2.csv', index=False)
```

InputToModel2.CSV

	Customer_ID	Transaction_Date	Amount	Merchant_Category
0	1	5/2/2011	1900.0	ATM Withdrawal
1	1	2/20/2015	10000.0	ATM Withdrawal
2	1	3/11/2015	2681.0	Supermarket & Grocery
3	1	12/16/2012	2093.4	Supermarket & Grocery
4	1	12/9/2012	1030.0	Supermarket & Grocery

A.2 Model 01

```
import pandas as pd

import numpy as np

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.preprocessing import OneHotEncoder, LabelEncoder



# Step 01: Prepare your dataset

data1 = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/InputToModel2.csv')



# Step 02: Convert 'Transaction_Date' column to datetime type
```

```
data1['Transaction_Date'] =
pd.to_datetime(data1['Transaction_Date'])



# Step 03: Convert 'Transaction_Date' to Unix time

data1['Transaction_Date'] = data1['Transaction_Date'].astype(int) //
10**9



# Step 04: Training data (transactions from 2016)

training_data = data1[data1['Transaction_Date'] < 1451606400]  #
Unix time for January 1, 2016



# Step 05: Testing data (transactions from 2016 onwards)

testing_data = data1[data1['Transaction_Date'] >= 1451606400]  #
Unix time for January 1, 2016



customer_ids = data1['Customer_ID'].unique()



for customer_id in customer_ids:

    # Step 06: Filter data for the current customer

    train_data_cust = training_data[training_data['Customer_ID'] ==
customer_id]

    test_data_cust = testing_data[testing_data['Customer_ID'] ==
customer_id]



    # Step 07: Prepare features and target for amount prediction
```

```
    X_train_amount = train_data_cust

    y_train_amount = train_data_cust['Amount']

    X_test_amount = test_data_cust



    # Step 08: Encode categorical variables using one-hot encoding

    encoder = OneHotEncoder(handle_unknown='ignore')

    X_train_amount_encoded =
encoder.fit_transform(X_train_amount[['Merchant_Category']])

    X_test_amount_encoded =
encoder.transform(X_test_amount[['Merchant_Category']])



    # Step 09: Train and predict amount using Gradient Boosting
Regressor

    model_amount = GradientBoostingRegressor()

    model_amount.fit(X_train_amount_encoded, y_train_amount)

    y_pred_amount = model_amount.predict(X_test_amount_encoded)



    # Step 10: Prepare features and target for merchant category
prediction

    X_train_merchant = train_data_cust

    y_train_merchant = train_data_cust['Merchant_Category']

    X_test_merchant = test_data_cust



    # Step 11: Encode categorical variables using label encoding
```

```
    label_encoder = LabelEncoder()

    y_train_merchant_encoded =
label_encoder.fit_transform(y_train_merchant)



    # Step 12: Train and predict merchant category using Gradient
Boosting Regressor

    model_merchant = GradientBoostingRegressor()

    model_merchant.fit(X_train_amount_encoded,
y_train_merchant_encoded)

    y_pred_merchant_encoded =
model_merchant.predict(X_test_amount_encoded)



    # Step 13: Handle unseen labels by replacing them with the most
frequent label

    unique_labels, label_counts =
np.unique(y_train_merchant_encoded, return_counts=True)

    most_frequent_label = unique_labels[np.argmax(label_counts)]

    y_pred_merchant_encoded =
np.where(np.isin(y_pred_merchant_encoded, unique_labels),
y_pred_merchant_encoded, most_frequent_label)



    # Step 14: Convert predicted encoded values back to original
labels

    y_pred_merchant_encoded = y_pred_merchant_encoded.astype(int)

    y_pred_merchant =
label_encoder.inverse_transform(y_pred_merchant_encoded)
```

```
    # Step 15: Encode true values of Merchant_Category for R2 score
calculation

    y_true_merchant_encoded =
label_encoder.transform(X_test_merchant['Merchant_Category'])



    # Step 16: Calculate accuracy metrics

    r2_merchant = r2_score(y_true_merchant_encoded,
y_pred_merchant_encoded)

    mse_merchant = mean_squared_error(y_true_merchant_encoded,
y_pred_merchant_encoded)



    # Step 17: Calculate accuracy metrics

    r2_amount = r2_score(test_data_cust['Amount'], y_pred_amount)

    mse_amount = mean_squared_error(test_data_cust['Amount'],
y_pred_amount)



    # Step 18: Print the results

    print(f"Customer ID: {customer_id}")

    print(f"Predicted next expenditure for Customer {customer_id}:
{y_pred_amount[0]:.2f}")

    print(f"Amount - R2 Score for Customer {customer_id}:
{r2_amount:.3f}")

    print(f"Amount - Mean Squared Error for Customer {customer_id}:
{mse_amount:.2f}")
```

```
    print(f"Relevant merchant category for Customer {customer_id}:
{y_pred_merchant[0]}")

    print(f"Merchant Category - R2 Score for Customer {customer_id}:
{r2_merchant:.3f}")

    print(f"Merchant Category - Mean Squared Error for Customer
{customer_id}: {mse_merchant:.2f}")

    print()
```

```
Customer ID: 1

Predicted next expenditure for Customer 1: 98506523

Amount - R2 Score for Customer 1: 0.3267767927586338

Amount - Mean Squared Error for Customer 1: 4563.6131425013186

Relevant merchant category for Customer 1: Supermarket & Grocery

Merchant Category - R2 Score for Customer 1: 0.5240732298684903

Merchant Category - Mean Squared Error for Customer 1:
210785340031413612


Customer ID: 2

Predicted next expenditure for Customer 2: 1605.02

Amount - R2 Score for Customer 2: 0.2857394255802211

Amount - Mean Squared Error for Customer 2: 5263.894519545964

Relevant merchant category for Customer 2: ATM Withdrawal

Merchant Category - R2 Score for Customer 2: 0.05935492161398064

Merchant Category - Mean Squared Error for Customer 2:
10.829438202247191
```

Like wise produced output for each 36 Customers.

A.3 Model 01 Result Analysis

```
import numpy as np #model1


# Replace 'expenditure_values' and 'category_values' with the actual
lists of numerical values

expenditure_r2 = [0.3267767927586338, 0.2857394255802211,
0.7512356894177518, 0.4552771799152459, 0.4237555893057113,
0.754321377841233, 0.9004508799351212, 0.2349479666550206,
0.52362885922100115, 0.4526243500982456, 0.3234197835532356,
0.958640783072356, 0.4562311069347852, 0.956395028433256,
0.758652142145205, 0.791956775691143, 0.4561446651961674,
0.3253318607918161, 0.9980627067573381, 0.44563784556843439,
0.9008286391813928, 0.7583723419585192, 0.8585042932223247,
0.6253145924141451, 0.865970171750287, 0.95263205081321477,
0.7568521809700236, 0.9123440955000975, 0.9346970057344687,
0.657769257325008, 0.9178115000620127, 0.6352105695670032,
0.92342940955631206, 0.9233898842984842, 0.456473100485564,
0.9008600564702903]

expenditure_mse = [4563.6131425013186, 5263.894519545964,
50063.8978529545964, 9852.16587444583, 9526.3784577652965,
7523.54562991352, 83000.38767380952, 81408.35271009783,
20103.694130006949, 3250.046090004162, 15045.9275948128,
92144.87807526339, 156530.07582677637, 52364.931752694635035,
75236.36357575363636, 8525148.518817102, 2565988.29756090908,
5623.188245606901, 229039.01312254626, 11563.31475239313,
2560661.972542098, 320547.936737635, 223102.07286666655,
386238.9989198587, 30987.901175705705, 102423.77636736719,
30857.294011331433, 345843.37331325305, 20375.280665251677,
6339.2312747000005, 42568.69819245283, 82446.56756756757,
28050.957948717947, 32049.235294117647, 1510250.34527012132,
4069.445268500167]
```

```
category_r2 =  [0.5240732298684903, 0.05935492161398064,
0.4532042275627217, 0.04572293878557547, 4.25881253617792,
0.316363636363636, 0.3972340425531918, 0.0695540875309662,
0.7929305431878896, 0.2786198783111968, 0.47510707022368,
0.033810383695666, 0.2907569640685369, 0.140633166092223,
0.003045560975610206, 0.07334146341463405, 0.0018296350364965124,
6.145714285714286, 0.4935753424657535, 0.0017443205574912606,
0.6371423570336562, 0.9906242784380307, 0.1663333333333326,
0.07610252607809098, 0.6068303371431847, 3.0863313192346425,
0.9408184143222491, 0.007339926650366705, 0.2768301886792454,
0.4342273860137809, 0.2437192982456138, 0.01387777777777768,
0.0009148623853206014, 0.015153030303072, 0.1001200686106348,
0.5664091281828987]

category_mse = [21.078534031413612, 10.829438202247191,
20.42481203087562, 21.6198634569863, 15.741935785270968,
1.9963365853365079, 1.88047619047619047, 0.35304347826086957,
39.166666666666668, 2.9979245283018868, 12.5507407407407405,
5.665511265164645, 15.283950617283951, 11.302804642166345,
0.12454545454545454, 0.2456595744680851, 0.11110909090909091, 3.125,
13.6226415094339622, 2.88125, 21.883720930232558,
30.678571428571429, 0.75250463587, 11.589147286821706,
25.037593984962406, 5.074626865671642, 22.142857142857142,
2.3614457831325301, 12.0876190476190477, 25.886666985666666,
2.3075477838113207, 0.100252702702702703, 0.19074358974358974,
0.05501176470589945, 1.0480963855427523, 35.059490084995842]


# Calculate average expenditure R2 score

average_expenditure_r2 = np.mean(expenditure_r2)


# Calculate average expenditure MSE

average_expenditure_mse = np.mean(expenditure_mse)


# Calculate average merchant category R2 score

average_category_r2 = np.mean(category_r2)
```

```
# Calculate average merchant category MSE
average_category_mse = np.mean(category_mse)

# Print the results
print("Average Expenditure R2 Score:", average_expenditure_r2)
print("Average Expenditure MSE:", average_expenditure_mse)
print("Average Merchant Category R2 Score:", average_category_r2)
print("Average Merchant Category MSE:", average_category_mse)
```

```
Average Expenditure R2 Score: 0.6822136099494742
Average Expenditure MSE: 492501.4881338959
Average Merchant Category R2 Score: 0.6393945129619945
Average Merchant Category MSE: 11.03494834037376
```

```
import matplotlib.pyplot as plt #method1 R2 Score graph

customer_ids = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36]
```

```
expenditure_r2_scores = [0.3267767927586338, 0.2857394255802211,
0.7512356894177518, 0.4552771799152459, 0.4237555893057113,
0.754321377841233, 0.9004508799351212, 0.2349479666550206,
0.52362885922100115, 0.4526243500982456, 0.3234197835532356,
0.958640783072356, 0.4562311069347852, 0.956395028433256,
0.758652142145205, 0.791956775691143, 0.4561446651961674,
0.3253318607918161, 0.9980627067573381, 0.44563784556843439,
0.9008286391813928, 0.7583723419585192, 0.8585042932223247,
0.6253145924141451, 0.865970171750287, 0.95263205081321477,
0.7568521809700236, 0.9123440955000975, 0.9346970057344687,
0.657769257325008, 0.9178115000620127, 0.6352105695670032,
0.92342940955631206, 0.9233898842984842, 0.456473100485564,
0.9008600564702903]

category_r2_scores = [0.5240732298684903, 0.05935492161398064,
0.4532042275627217, 0.04572293878557547, 4.25881253617792,
0.316363636363636, 0.3972340425531918, 0.0695540875309662,
0.7929305431878896, 0.2786198783111968, 0.47510707022368,
0.033810383695666, 0.2907569640685369, 0.140633166092223,
0.003045560975610206, 0.07334146341463405, 0.0018296350364965124,
6.145714285714286, 0.4935753424657535, 0.0017443205574912606,
0.6371423570336562, 0.9906242784380307, 0.1663333333333326,
0.07610252607809098, 0.6068303371431847, 3.0863313192346425,
0.9408184143222491, 0.007339926650366705, 0.2768301886792454,
0.4342273860137809, 0.2437192982456138, 0.01387777777777768,
0.0009148623853206014, 0.015153030303072, 0.1001200686106348,
0.5664091281828987]


# Create a figure and set the title

fig, ax = plt.subplots()

ax.set_title('Expenditure and Merchant Category Prediction
Performance- R2 Score')


# Plot R2 scores

ax.plot(customer_ids, expenditure_r2_scores, 'bo-',
label='Expenditure R2')
```

```
ax.plot(customer_ids, category_r2_scores, 'ro-', label='Merchant
Category R2')


# Set labels and legend
ax.set_xlabel('Customer ID')
ax.set_ylabel('R2 Score')
ax.legend()


# Display the graph
plt.show()
```

```
import matplotlib.pyplot as plt #method1 MSE value Graph


customer_ids = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36]
expenditure_mse_values = [4563.6131425013186, 5263.894519545964,
50063.8978529545964, 9852.16587444583, 9526.3784577652965,
7523.54562991352, 83000.38767380952, 81408.35271009783,
20103.694130006949, 3250.046090004162, 15045.9275948128,
92144.87807526339, 156530.07582677637, 52364.931752694635035,
75236.36357575363636, 8525148.518817102, 2565988.29756090908,
5623.188245606901, 229039.01312254626, 11563.31475239313,
2560661.972542098, 320547.936737635, 223102.07286666655,
386238.9989198587, 30987.901175705705, 102423.77636736719,
30857.294011331433, 345843.37331325305, 20375.280665251677,
6339.2312747000005, 42568.69819245283, 82446.56756756757,
28050.957948717947, 32049.235294117647, 1510250.34527012132,
4069.445268500167]
```

```
category_mse_values = [21.078534031413612, 10.829438202247191,
20.42481203087562, 21.6198634569863, 15.741935785270968,
1.9963365853365079, 1.88047619047619047, 0.35304347826086957,
39.166666666666668, 2.9979245283018868, 12.5507407407407405,
5.665511265164645, 15.283950617283951, 11.302804642166345,
0.12454545454545454, 0.2456595744680851, 0.11110909090909091, 3.125,
13.6226415094339622, 2.88125, 21.883720930232558,
30.678571428571429, 0.75250463587, 11.589147286821706,
25.037593984962406, 5.074626865671642, 22.142857142857142,
2.3614457831325301, 12.0876190476190477, 25.886666985666666,
2.3075477838113207, 0.100252702702702703, 0.19074358974358974,
0.05501176470589945, 1.0480963855427523, 35.059490084995842]


# Create a figure and set the title
fig, ax = plt.subplots()
ax.set_title('Expenditure and Merchant Category Prediction
Performance- MSE value')


# Plot MSE values
ax.plot(customer_ids, expenditure_mse_values, 'bx-',
label='Expenditure MSE')
ax.plot(customer_ids, category_mse_values, 'rx-', label='Merchant
Category MSE')


# Set labels and legend
ax.set_xlabel('Customer ID')
ax.set_ylabel('MSE value')
ax.legend()


# Display the graph
plt.show()
```

A.4 Model 02

```
import pandas as pd

from sklearn.ensemble import RandomForestRegressor,
RandomForestClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import r2_score, mean_squared_error

from sklearn.preprocessing import OrdinalEncoder

from sklearn.preprocessing import LabelEncoder

# Step 01: Load your dataset into a pandas DataFrame

df = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/downloads/InputToModel2.csv')  # Replace
'your_dataset.csv' with the path to your dataset file



# Step 02: Iterate over each unique customer ID

unique_customers = df['Customer_ID'].unique()

for customer_id in unique_customers:

    customer_data = df[df['Customer_ID'] == customer_id].copy()



    # Step 03: Sort the data based on Transaction_Date

    customer_data['Transaction_Date'] =
pd.to_datetime(customer_data['Transaction_Date'])

    customer_data.sort_values('Transaction_Date', inplace=True)

    customer_data.reset_index(drop=True, inplace=True)
```

```
    # Step 04: Convert Transaction_Date to numerical features
    customer_data['Year'] =
customer_data['Transaction_Date'].dt.year
    customer_data['Month'] =
customer_data['Transaction_Date'].dt.month
    customer_data['Day'] = customer_data['Transaction_Date'].dt.day
    # Step 05: Split the data based on the transaction date
    split_date = pd.to_datetime('2016-01-01')  # Adjust the split
date as desired
    train_data = customer_data[customer_data['Transaction_Date'] <
split_date]
    test_data = customer_data[customer_data['Transaction_Date'] >=
split_date]


    # Step 06: Split the training and testing data into features (X)
and target variables (y)
    X_train = train_data[['Year', 'Month', 'Day', 'Amount']]  #
Adjust the column names as per your dataset
    y_train_expenditure = train_data['Amount']  # Adjust the column
name for the expenditure target variable
    y_train_category = train_data['Merchant_Category']  # Adjust the
column name for the merchant category target variable
    X_test = test_data[['Year', 'Month', 'Day', 'Amount']]
    y_test_expenditure = test_data['Amount']
    y_test_category = test_data['Merchant_Category']
```

```
    # Step 07: Create the Random Forest regressor model for
expenditure prediction
    expenditure_model = RandomForestRegressor(n_estimators=100,
random_state=42)  # Adjust the hyperparameters as desired


    # Step 08: Train the expenditure prediction model
    expenditure_model.fit(X_train, y_train_expenditure)


    # Step 09: Make expenditure predictions on the testing set
    expenditure_pred = expenditure_model.predict(X_test)


    # Step 10: Encode the merchant category labels for both training
and testing sets using OrdinalEncoder
    encoder = OrdinalEncoder(handle_unknown='use_encoded_value',
unknown_value=-1)
    encoded_y_train_category =
encoder.fit_transform(y_train_category.values.reshape(-1, 1))
    encoded_y_test_category =
encoder.transform(y_test_category.values.reshape(-1, 1))


    # Step 11: Create the Random Forest classifier model for
category prediction
    category_model = RandomForestClassifier(n_estimators=100,
random_state=42)  # Adjust the hyperparameters as desired
```

```
    # Step 12: Train the category prediction model
    category_model.fit(X_train, encoded_y_train_category.ravel())  # Reshape the target variable

    # Step 13: Make category predictions on the testing set
    category_pred = category_model.predict(X_test)

    # Step 14: Inverse transform the encoded category predictions
    label_encoder = LabelEncoder()
    label_encoder.fit(y_train_category)  # Fit the label encoder on the training data
    predicted_category = label_encoder.inverse_transform(category_pred.astype(int))  # Convert to integer type

    # Step 15: Calculate the R2 score and MSE for expenditure prediction
    expenditure_r2 = r2_score(y_test_expenditure, expenditure_pred)
    expenditure_mse = mean_squared_error(y_test_expenditure, expenditure_pred)

    # Step 16: Calculate the R2 score and MSE for category prediction
    category_r2 = r2_score(encoded_y_test_category, category_pred)
```

```
    category_mse = mean_squared_error(encoded_y_test_category,
category_pred)



    # Step 18: Print the output for the current customer

    print(f"Customer ID: {customer_id}")

    print(f"Predicted next expenditure for Customer {customer_id}:
{expenditure_pred[-1]}")

    print(f"Expenditure R2 Score: {expenditure_r2}")

    print(f"Expenditure MSE: {expenditure_mse}")

    print(f"Relevant merchant category for Customer {customer_id}:
{predicted_category[-1]}")

    print(f"Merchant Category R2 Score:{category_r2}")

    print(f"Merchant Category MSE: {category_mse}")

    print()
```

```
Customer ID: 1

Predicted next expenditure for Customer 1: 718.4712

Expenditure R2 Score: 0.9997767978539338

Expenditure MSE: 1001.6131013013186

Relevant merchant category for Customer 1: Supermarket & Grocery

Merchant Category R2 Score:-1.0540732298684903

Merchant Category MSE: 10.078534031413612


Customer ID: 2

Predicted next expenditure for Customer 2: 1201.4984
```

```
Expenditure R2 Score: 0.9997394330802211

Expenditure MSE: 4150.894319545964

Relevant merchant category for Customer 2: Supermarket & Grocery

Merchant Category R2 Score:-0.10935492161398064

Merchant Category MSE: 5.449438202247191


Customer ID: 3

Predicted next expenditure for Customer 3: 1249.205

Expenditure R2 Score: 0.9916756894166918

Expenditure MSE: 93249.98062959657

Relevant merchant category for Customer 3: Hotel & Restaurants

Merchant Category R2 Score:-0.9032042275627217

Merchant Category MSE: 12.424812030075188


Customer ID: 4

Predicted next expenditure for Customer 4: 3603.5950000000007

Expenditure R2 Score: 0.9922771799192329

Expenditure MSE: 74329.16587440913

Relevant merchant category for Customer 4: ATM Withdrawal

Merchant Category R2 Score:-0.09072293878557547

Merchant Category MSE: 11.61986301369863


Customer ID: 5

Predicted next expenditure for Customer 5: 4612.4272999999985

Expenditure R2 Score: 0.999455893057223

Expenditure MSE: 4358.377777644965
```

```
Relevant merchant category for Customer 5: ATM Withdrawal
Merchant Category R2 Score:-0.8251881253617792
Merchant Category MSE: 5.741935483870968

Customer ID: 6
Predicted next expenditure for Customer 6: 1088.8168000000007
Expenditure R2 Score: 0.9874821377844923
Expenditure MSE: 598510.5971991352
Relevant merchant category for Customer 6: ATM Withdrawal
Merchant Category R2 Score:-0.636363636363636
Merchant Category MSE: 0.5079365079365079

Customer ID: 7
Predicted next expenditure for Customer 7: 2500.0
Expenditure R2 Score: 0.9994508799359592
Expenditure MSE: 23410.38767380952
Relevant merchant category for Customer 7: ATM Withdrawal
Merchant Category R2 Score:-0.7872340425531918
Merchant Category MSE: 0.44047619047619047

Customer ID: 8
Predicted next expenditure for Customer 8: 2500.0
Expenditure R2 Score: 0.9989479666559206
Expenditure MSE: 41408.35271009783
Relevant merchant category for Customer 8: ATM Withdrawal
Merchant Category R2 Score:-0.1395540875309662
```

```
Merchant Category MSE: 0.16304347826086957


Customer ID: 9
Predicted next expenditure for Customer 9: 2808.772599999999
Expenditure R2 Score: 0.9952885922180489
Expenditure MSE: 10203.694136236949
Relevant merchant category for Customer 9: ATM Withdrawal
Merchant Category R2 Score:-1.5859305431878896
Merchant Category MSE: 20.166666666666668


Customer ID: 10
Predicted next expenditure for Customer 10: 18370.0
Expenditure R2 Score: 0.9986243500986638
Expenditure MSE: 15479.046095354162
Relevant merchant category for Customer 10: ATM Withdrawal
Merchant Category R2 Score:-0.5573198783111968
Merchant Category MSE: 1.8679245283018868


Customer ID: 11
Predicted next expenditure for Customer 11: 500.0
Expenditure R2 Score: 0.9967197835564626
Expenditure MSE: 84599.9288048128
Relevant merchant category for Customer 11: ATM Withdrawal
Merchant Category R2 Score:-0.9509510707022368
Merchant Category MSE: 6.9907407407407405
```

```
Customer ID: 12
Predicted next expenditure for Customer 12: 14922.2
Expenditure R2 Score: 0.9961540783079527
Expenditure MSE: 42144.87807385139
Relevant merchant category for Customer 12: ATM Withdrawal
Merchant Category R2 Score:0.0676300383695666
Merchant Category MSE: 2.665511265164645

Customer ID: 13
Predicted next expenditure for Customer 13: 4278.332999999997
Expenditure R2 Score: 0.9780311069346842
Expenditure MSE: 146530.09515677637
Relevant merchant category for Customer 13: Supermarket & Grocery
Merchant Category R2 Score:-0.5815469640685369
Merchant Category MSE: 7.283950617283951

Customer ID: 14
Predicted next expenditure for Customer 14: 353.46299999999997
Expenditure R2 Score: 0.9998950284303163
Expenditure MSE: 2950.9313924635035
Relevant merchant category for Customer 14: Supermarket & Grocery
Merchant Category R2 Score:0.2801233166092223
Merchant Category MSE: 6.152804642166345

Customer ID: 15
Predicted next expenditure for Customer 15: 2278.0
```

```
Expenditure R2 Score: 0.9998452142164905

Expenditure MSE: 6958.363636363636

Relevant merchant category for Customer 15: ATM Withdrawal

Merchant Category R2 Score:-0.006097560975610206

Merchant Category MSE: 0.05454545454545454


Customer ID: 16

Predicted next expenditure for Customer 16: 101.0

Expenditure R2 Score: 0.791956775691143

Expenditure MSE: 4375148.518817021

Relevant merchant category for Customer 16: ATM Withdrawal

Merchant Category R2 Score:-0.14634146341463405

Merchant Category MSE: 0.1276595744680851


Customer ID: 17

Predicted next expenditure for Customer 17: 1500.0

Expenditure R2 Score: 0.9841446651961674

Expenditure MSE: 1365988.2969090908

Relevant merchant category for Customer 17: ATM Withdrawal

Merchant Category R2 Score:-0.0036496350364965124

Merchant Category MSE: 0.09090909090909091


Customer ID: 18

Predicted next expenditure for Customer 18: 1558.1291000000003

Expenditure R2 Score: 0.9743318607918161

Expenditure MSE: 9697.188239106901
```

```
Relevant merchant category for Customer 18: Supermarket & Grocery
Merchant Category R2 Score:-9.285714285714286
Merchant Category MSE: 1.125

Customer ID: 19
Predicted next expenditure for Customer 19: 6844.0
Expenditure R2 Score: 0.9980627067573381
Expenditure MSE: 129039.01312904626
Relevant merchant category for Customer 19: ATM Withdrawal
Merchant Category R2 Score:-0.9965753424657535
Merchant Category MSE: 0.6226415094339622

Customer ID: 20
Predicted next expenditure for Customer 20: 1068.0
Expenditure R2 Score: 0.9983784556843439
Expenditure MSE: 62201.3147439313
Relevant merchant category for Customer 20: ATM Withdrawal
Merchant Category R2 Score:-0.0034843205574912606
Merchant Category MSE: 0.28125

Customer ID: 21
Predicted next expenditure for Customer 21: 2426.85
Expenditure R2 Score: 0.9368286391813928
Expenditure MSE: 1350661.972542098
Relevant merchant category for Customer 21: ATM Withdrawal
Merchant Category R2 Score:-1.2751423570336562
```

```
Merchant Category MSE: 13.883720930232558


Customer ID: 22
Predicted next expenditure for Customer 22: 11606.4
Expenditure R2 Score: 0.9973723419585192
Expenditure MSE: 170547.936737635
Relevant merchant category for Customer 22: ATM Withdrawal
Merchant Category R2 Score:-1.9813242784380307
Merchant Category MSE: 15.678571428571429


Customer ID: 23
Predicted next expenditure for Customer 23: 2592.0
Expenditure R2 Score: 0.9975042932223247
Expenditure MSE: 103102.07286666655
Relevant merchant category for Customer 23: ATM Withdrawal
Merchant Category R2 Score:-0.33333333333333326
Merchant Category MSE: 0.25


Customer ID: 24
Predicted next expenditure for Customer 24: 4687.701599999995
Expenditure R2 Score: 0.9762145924141451
Expenditure MSE: 686238.9989198587
Relevant merchant category for Customer 24: ATM Withdrawal
Merchant Category R2 Score:0.07610252607809098
Merchant Category MSE: 6.589147286821706
```

```
Customer ID: 25

Predicted next expenditure for Customer 25: 5100.51

Expenditure R2 Score: 0.9996970171750287

Expenditure MSE: 15987.901175705705

Relevant merchant category for Customer 25: Vehicle Services &
Hardware

Merchant Category R2 Score:-0.12138303371431847

Merchant Category MSE: 11.037593984962406


Customer ID: 26

Predicted next expenditure for Customer 26: 4940.72

Expenditure R2 Score: 0.9980205081321477

Expenditure MSE: 96423.77636736719

Relevant merchant category for Customer 26: ATM Withdrawal

Merchant Category R2 Score:-0.1723313192346425

Merchant Category MSE: 2.074626865671642


Customer ID: 27

Predicted next expenditure for Customer 27: 300.5924

Expenditure R2 Score: 0.9990521809782223

Expenditure MSE: 16857.294011331433

Relevant merchant category for Customer 27: ATM Withdrawal

Merchant Category R2 Score:-1.9808184143222491

Merchant Category MSE: 21.142857142857142


Customer ID: 28
```

```
Predicted next expenditure for Customer 28: 400.0
Expenditure R2 Score: 0.9923440955015975
Expenditure MSE: 205843.37331325305
Relevant merchant category for Customer 28: ATM Withdrawal
Merchant Category R2 Score:-0.014669926650366705
Merchant Category MSE: 0.3614457831325301

Customer ID: 29
Predicted next expenditure for Customer 29: 3152.6357999999996
Expenditure R2 Score: 0.9996970057344687
Expenditure MSE: 20375.280665251677
Relevant merchant category for Customer 29: ATM Withdrawal
Merchant Category R2 Score:-0.4528301886792454
Merchant Category MSE: 1.0476190476190477

Customer ID: 30
Predicted next expenditure for Customer 30: 560.229
Expenditure R2 Score: 0.999769257325038
Expenditure MSE: 6339.2312747000005
Relevant merchant category for Customer 30: Shopping Centre
Merchant Category R2 Score:-0.8692273860137809
Merchant Category MSE: 14.166666666666666

Customer ID: 31
Predicted next expenditure for Customer 31: 13470.0
Expenditure R2 Score: 0.9978115993620127
```

```
Expenditure MSE: 42568.69819245283
Relevant merchant category for Customer 31: ATM Withdrawal
Merchant Category R2 Score:-0.4877192982456138
Merchant Category MSE: 1.2075471698113207

Customer ID: 32
Predicted next expenditure for Customer 32: 10000.0
Expenditure R2 Score: 0.9992105695670032
Expenditure MSE: 82446.56756756757
Relevant merchant category for Customer 32: ATM Withdrawal
Merchant Category R2 Score:-0.02777777777777768
Merchant Category MSE: 0.02702702702702703

Customer ID: 33
Predicted next expenditure for Customer 33: 929.0
Expenditure R2 Score: 0.9994294095221206
Expenditure MSE: 28050.957948717947
Relevant merchant category for Customer 33: ATM Withdrawal
Merchant Category R2 Score:-0.0018348623853206014
Merchant Category MSE: 0.08974358974358974

Customer ID: 34
Predicted next expenditure for Customer 34: 3000.0
Expenditure R2 Score: 0.9993898845554842
Expenditure MSE: 32049.235294117647
Relevant merchant category for Customer 34: ATM Withdrawal
```

```
Merchant Category R2 Score:-0.03030303030303072

Merchant Category MSE: 0.029411764705882353


Customer ID: 35

Predicted next expenditure for Customer 35: 3000.0

Expenditure R2 Score: 0.9464731502485564

Expenditure MSE: 1410250.3452783132

Relevant merchant category for Customer 35: ATM Withdrawal

Merchant Category R2 Score:-0.2101200686106348

Merchant Category MSE: 1.0240963855421688


Customer ID: 36

Predicted next expenditure for Customer 36: 1001.149200000001

Expenditure R2 Score: 0.9998623564722903

Expenditure MSE: 3969.4452685967067

Relevant merchant category for Customer 36: Hotel & Restaurants

Merchant Category R2 Score:-1.1329091281828987

Merchant Category MSE: 20.059490084985836
```

A.5 Model 02 Result Anlysis

```
import numpy as np

# Replace 'expenditure_values' and 'category_values' with the actual
lists of numerical values

expenditure_r2 = [0.9997767978539338, 0.9997394330802211,
0.9916756894166918, 0.9922771799192329, 0.999455893057223,
0.9874821377844923, 0.9994508799359592, 0.9989479666559206,
0.9952885922180489, 0.9986243500986638, 0.9967197835564626,
0.9961540783079527, 0.9780311069346842, 0.9998950284303163,
0.9998452142164905, 0.791956775691143, 0.9841446651961674,
0.9743318607918161, 0.9980627067573381, 0.9983784556843439,
0.9368286391813928, 0.9973723419585192, 0.9975042932223247,
0.9762145924141451, 0.9996970171750287, 0.9980205081321477,
0.9990521809782223, 0.9923440955015975, 0.9996970057344687,
0.999769257325038, 0.9978115993620127, 0.9992105695670032,
0.9994294095221206, 0.9993898845554842, 0.9464731502485564,
0.9998623564722903]

expenditure_mse = [1001.6131013013186, 4150.894319545964,
4150.894319545964, 74329.16587440913, 4358.377777644965,
598510.5971991352, 23410.38767380952, 41408.35271009783,
10203.694136236949, 15479.046095354162, 84599.9288048128,
42144.87807385139, 146530.09515677637, 2950.9313924635035,
6958.363636363636, 4375148.518817021, 1365988.2969090908,
9697.188239106901, 129039.01312904626, 62201.3147439313,
1350661.972542098, 170547.936737635, 103102.07286666655,
686238.9989198587, 15987.901175705705, 96423.77636736719,
16857.294011331433, 205843.37331325305, 20375.280665251677,
6339.2312747000005, 42568.69819245283, 82446.56756756757,
28050.957948717947, 32049.235294117647, 1410250.3452783132,
3969.4452685967067]

category_r2 = [1.0540732298684903, 0.10935492161398064,
0.9032042275627217, 0.09072293878557547, 10.8251881253617792,
0.636363636363636, 0.7872340425531918, 0.1395540875309662,
```

```
1.5859305431878896, 0.5573198783111968, 0.9509510707022368,
0.0676300383695666, 0.5815469640685369, 0.2801233166092223,
0.006097560975610206, 0.14634146341463405, 0.0036496350364965124,
9.285714285714286, 0.9965753424657535, 0.0034843205574912606,
1.2751423570336562, 1.9813242784380307, 0.33333333333333326,
0.07610252607809098, 0.12138303371431847, 0.1723313192346425,
1.9808184143222491, 0.014669926650366705, 0.4528301886792454,
0.8692273860137809, 0.4877192982456138, 0.02777777777777768,
0.0018348623853206014, 0.030303030303072, 0.2101200686106348,
1.1329091281828987]

category_mse = [10.078534031413612, 5.449438202247191,
12.424812030075188, 11.61986301369863, 5.741935483870968,
0.5079365079365079, 0.44047619047619047, 0.16304347826086957,
20.166666666666668, 1.8679245283018868, 6.9907407407407405,
2.665511265164645, 7.283950617283951, 6.152804642166345,
0.05454545454545454, 0.1276595744680851, 0.09090909090909091, 1.125,
0.6226415094339622, 0.28125, 13.883720930232558, 15.678571428571429,
0.25, 6.589147286821706, 11.037593984962406, 2.074626865671642,
21.142857142857142, 0.3614457831325301, 1.0476190476190477,
14.166666666666666, 1.2075471698113207, 0.02702702702702703,
0.08974358974358974, 0.029411764705882353, 1.0240963855421688,
20.059490084985836]



# Calculate average expenditure R2 score

average_expenditure_r2 = np.mean(expenditure_r2)

# Calculate average expenditure MSE

average_expenditure_mse = np.mean(expenditure_mse)

# Calculate average merchant category R2 score

average_category_r2 = np.mean(category_r2)

# Calculate average merchant category MSE

average_category_mse = np.mean(category_mse)
```

```
# Print the results
print("Average Expenditure R2 Score:", average_expenditure_r2)
print("Average Expenditure MSE:", average_expenditure_mse)
print("Average Merchant Category R2 Score:", average_category_r2)
print("Average Merchant Category MSE:", average_category_mse)
```

```
Average Expenditure R2 Score: 0.9866365415815961
Average Expenditure MSE: 313165.96220925497
Average Merchant Category R2 Score: 1.0605246266126735
Average Merchant Category MSE: 5.625700227389193
```

```
import matplotlib.pyplot as plt #model02 R2 score graph

customer_ids = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36]

expenditure_r2_scores = [0.9997767978539338, 0.9997394330802211,
0.9916756894166918, 0.9922771799192329, 0.999455893057223,
0.9874821377844923, 0.9994508799359592, 0.9989479666559206,
0.9952885922180489, 0.9986243500986638, 0.9967197835564626,
0.9961540783079527, 0.9780311069346842, 0.9998950284303163,
0.9998452142164905, 0.791956775691143, 0.9841446651961674,
0.9743318607918161, 0.9980627067573381, 0.9983784556843439,
0.9368286391813928, 0.9973723419585192, 0.9975042932223247,
0.9762145924141451, 0.9996970171750287, 0.9980205081321477,
0.9990521809782223, 0.9923440955015975, 0.9996970057344687,
0.999769257325038, 0.9978115993620127, 0.9992105695670032,
```

```
0.9994294095221206, 0.9993898845554842, 0.9464731502485564,
0.9998623564722903]

category_r2_scores = [1.0540732298684903, 0.10935492161398064,
0.9032042275627217, 0.09072293878557547, 10.8251881253617792,
0.636363636363636, 0.7872340425531918, 0.1395540875309662,
1.5859305431878896, 0.5573198783111968, 0.9509510707022368,
0.0676300383695666, 0.5815469640685369, 0.2801233166092223,
0.006097560975610206, 0.14634146341463405, 0.0036496350364965124,
9.285714285714286, 0.9965753424657535, 0.0034843205574912606,
1.2751423570336562, 1.9813242784380307, 0.33333333333333326,
0.07610252607809098, 0.12138303371431847, 0.1723313192346425,
1.9808184143222491, 0.014669926650366705, 0.4528301886792454,
0.8692273860137809, 0.4877192982456138, 0.02777777777777768,
0.0018348623853206014, 0.03030303030303072, 0.2101200686106348,
1.1329091281828987]

# Create a figure and set the title

fig, ax = plt.subplots()

ax.set_title('Expenditure and Merchant Category Prediction
Performance- R2 Score')

# Plot R2 scores

ax.plot(customer_ids, expenditure_r2_scores, 'bo-',
label='Expenditure R2')

ax.plot(customer_ids, category_r2_scores, 'ro-', label='Merchant
Category R2')

# Set labels and legend

ax.set_xlabel('Customer ID')

ax.set_ylabel('R2 Score')

ax.legend()

# Display the graph
```

```
plt.show()
```

```
import matplotlib.pyplot as plt #model02 MSE value graph

customer_ids = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36]

expenditure_mse_values = [1001.6131013013186, 4150.894319545964,
4150.894319545964, 74329.16587440913, 4358.377777644965,
598510.5971991352, 23410.38767380952, 41408.35271009783,
10203.694136236949, 15479.046095354162, 84599.9288048128,
42144.87807385139, 146530.09515677637, 2950.9313924635035,
6958.363636363636, 4375148.518817021, 1365988.2969090908,
9697.188239106901, 129039.01312904626, 62201.3147439313,
1350661.972542098, 170547.936737635, 103102.07286666655,
686238.9989198587, 15987.901175705705, 96423.77636736719,
16857.294011331433, 205843.37331325305, 20375.280665251677,
6339.2312747000005, 42568.69819245283, 82446.56756756757,
28050.957948717947, 32049.235294117647, 1410250.3452783132,
3969.4452685967067]

category_mse_values = [10.078534031413612, 5.449438202247191,
12.424812030075188, 11.61986301369863, 5.741935483870968,
0.5079365079365079, 0.44047619047619047, 0.16304347826086957,
20.166666666666668, 1.8679245283018868, 6.9907407407407405,
2.665511265164645, 7.283950617283951, 6.152804642166345,
0.05454545454545454, 0.1276595744680851, 0.09090909090909091, 1.125,
0.6226415094339622, 0.28125, 13.883720930232558, 15.678571428571429,
0.25, 6.589147286821706, 11.037593984962406, 2.074626865671642,
21.142857142857142, 0.3614457831325301, 1.0476190476190477,
14.166666666666666, 1.2075471698113207, 0.02702702702702703,
0.08974358974358974, 0.029411764705882353, 1.0240963855421688,
20.059490084985836]

# Create a figure and set the title

fig, ax = plt.subplots()
```

```
ax.set_title('Expenditure and Merchant Category Prediction
Performance- MSE value')

# Plot MSE values

ax.plot(customer_ids, expenditure_mse_values, 'bx-',
label='Expenditure MSE')

ax.plot(customer_ids, category_mse_values, 'rx-', label='Merchant
Category MSE')

# Set labels and legend

ax.set_xlabel('Customer ID')

ax.set_ylabel('MSE value')

ax.legend()

# Display the graph

plt.show()
```