AI-ML FOUNDATIONS: FACTORS INFLUENCING CHURN IN TELCO

By

Allan Stalker
astalker.ieu2022@student.ie.edu

Cloe Chapotot
cchapotot.ieu2022@student.ie.edu

Marcela Funabashi
mmendesgimen.ieu2022@student.ie.edu

Matias Arevalo
marevalo.ieu2022@student.ie.edu

Moritz Göbbels
mgobbels.ieu2022@student.ie.edu

Pilar Guerrero
pguerrero.ieu2022@student.ie.edu

Vittorio Fialdini
vpereirafial.ieu2022@student.ie.edu

Abstract:
The paper uses the prediction of churning for a fictional telecommunications company as a real-world business situation to analyze. Customers churning, regardless of the industry of the business, results in loss and high costs of recuperation. Hence, it is important to observe which features make the probability of a customer to churn higher and accurately predict recall. Two classification models were applied and the team decided to use the Logistic Regression for the rest of the analysis as it proved to yield great results while being less complex and has better interpretability. Four hypotheses were tested to find the relationship between variables that lead to a positive churn label, finding that customers with a longer contract period are less likely to churn and that customers subscribed to multiple services are more likely to churn when they have low satisfaction.

Keywords: classification models, recall, logistic regression, random forest, churn prediction, and interpretability.

1. Introduction

Machine learning models can be applied to analyze behaviors to solve business questions. When exploring different industries, telecommunications appeared to be one of the most interesting cases to observe as it is indispensable from the consumer side, and from the business side, global telecommunications has a market size of USD 2.32 trillion in 2024 and is growing at a CAGR of 6.15% from 2024 to 2034[1]. Despite the industry's escalating growth, it faces complicated challenges that put its future profitability at stake, competition being one of them. The telecommunications market is currently very saturated with industry giants that dominate the market and smaller companies that attempt to get a piece of the pie. This over-saturation and struggle for market share reduces profit margins for all.

On the other side of the token, by having many service and product providers in the telecommunications industry, consumers can leverage the choice of provider. Given that these services have low differentiation, they become highly substitutable hence, consumers choose the company that gives them the best offer for their money. This raises the issue of brand switching and churning.

This paper will explore the prediction of churning, through a classification model, for a fictional telecommunications company based in California called Telco. It will perform an extensive machine learning analysis to determine the factors that may increase the probability of a customer churning. The idea is to look at variables related to the customers, such as geographical location and demographic characteristics, and some of the Telco company such as the characteristics of the contracted plan.

Furthermore, the goal of this study is to identify the factors that would lead to a higher probability of churning, helping businesses protect themselves with a better strategy. Being able to predict this behavior can save businesses millions of dollars annually as losing a customer doesn't only refer to a forfeit of revenues but also implies high costs in trying to replace the customer.

2. Hypotheses

H1: Customers with monthly contracts are more likely to churn compared to ones with yearly and bi-yearly contracts.

H2: In counties with very-low income per capita, a higher churn rate is associated with women than with men, and the opposite behavior is expected in non very-low income counties.

H3: Customers paying via mailed checks have a higher churn rate compared to those using bank withdrawals or credit cards.

H4: Customers subscribing to multiple company services are paradoxically more likely to churn if their satisfaction score drops below a threshold of 3, as their expectations of service quality are higher.

3. Data, Sources, Model And Variables Used

3.1. Data and sources used to obtain it.

Three datasets were merged and used to conduct the analysis. The main dataset "Telco customer churn (11.1.3+)" contains information about customers of a fictional Telco company, and reasons why they left the company. This dataset was sourced from Kaggle[2] and published by the user Al Fath Terry, who compiled the data from IBM Business Analytics[3]. This dataset also includes zip codes, which were used to identify customers' respective California counties. For this, a second dataset called "California Zip Codes" was merged, which was sourced from UnitedStatesZipCodes.org[4]. Finally, a third dataset, "Measures of Income in State of California", was used to merge information about the income per capita for each respective county. This was sourced from the Employment Development Department[5] of the state of California.

---

[1] Shivarkar, A. (n.d.). Telecommunication Market Size to Surpass USD 4.21 Trn by 2034. Precedence Research. Retrieved November 21, 2024, from https://www.precedenceresearch.com/telecommunication-market

[2] "Telco Customer Churn (11.1.3+)," accessed November 20, 2024, https://www.kaggle.com/datasets/alfathterry/telco-customer-churn-11-1-3.

[3] "Telco Customer Churn (11.1.3+)," <date>, https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/11/telco-customer-churn-1113.

[4] "U.S. ZIP Codes: Free ZIP Code Map and Zip Code Lookup," UnitedStatesZipCodes, accessed November 20, 2024, https://www.unitedstateszipcodes.org.

[5] "California LaborMarketInfo, The Economy," accessed November 20, 2024, https://labormarketinfo.edd.ca.gov/cgi/databrowsing/localAreaProfileQSMoreResult.asp?viewAll=yes&viewAllUS=&currentPage=1&currentPageUS=&sortUp=I.INCOME&sortDown=&criteria=income&categoryType=population+census+data&geogArea=0604000009&timeseries=&more=More+Areas&menuChoice=localAreaPro&printerFriendly=&BackHistory=-4&goTOPageText=.

3.2. Variables used and form of measurement.

The following variables pertain directly to the hypotheses previously stated. The other variables present in the dataset can be found in **Appendix 1**.

- **Gender**: The customer's gender: Male, Female.
- **Contract**: Indicates the customer's current contract type: Month-to-Month, One Year, Two Year.
- **Payment Method**: Indicates how the customer pays their bill: Bank Withdrawal, Credit Card, Mailed Check.
- **Satisfaction Score**: A customer's overall satisfaction rating of the company from 1 (Very Unsatisfied) to 5 (Very Satisfied).
- **County Income Per Capita**: Income per capita of the county where the customer lives.

The following variables were used to create a new feature "Add Ons", which refers to the number of company services the customer was subscribed to. The specifics of this transformation are discussed in section 3.5: Feature Engineering. For simplicity purposes, this new feature is the one used in section 3.3: Descriptive Analysis.

- **Phone Service**: Indicates if the customer subscribes to home phone service with the company: Yes, No.
- **Multiple Lines**: Indicates if the customer subscribes to multiple telephone lines with the company: Yes, No.
- **Internet Service**: Indicates if the customer subscribes to Internet service with the company: True, False.
- **Online Security**: Indicates if the customer subscribes to an additional online security service provided by the company: Yes, No.
- **Online Backup**: Indicates if the customer subscribes to an additional online backup service provided by the company: Yes, No.
- **Device Protection Plan**: Indicates if the customer subscribes to an additional device protection plan for their Internet equipment provided by the company: Yes, No.
- **Premium Tech Support**: Indicates if the customer subscribes to an additional technical support plan from the company with reduced wait times: Yes, No.
- **Streaming TV**: Indicates if the customer uses their Internet service to stream television programming from a third-party provider: Yes, No.
- **Streaming Movies**: Indicates if the customer uses their Internet service to stream movies from a third-party provider: Yes, No.
- **Streaming Music**: Indicates if the customer uses their Internet service to stream music from a third-party provider: Yes, No.
- **Unlimited Data**: Indicates if the customer has paid an additional monthly fee to have unlimited data downloads/uploads: Yes, No.

The following is the target variable used in the study.

- **Churn Label**: Yes = the customer left the company this quarter. No = the customer remained with the company.

3.3. Descriptive Analysis of the Variables

a. Numerical Variables

The table in **Appendix 2.1** shows the main statistics of the numerical variables pertinent to the hypotheses. They give an idea of the distribution of the customers' counties' income per capita and the number of services they are subscribed to within the company, respectively.

To detect the presence of outliers, boxplots of all numerical attributes in the dataset were plotted and are shown in **Appendix 2.2**. While some features such as "Age", "Add Ons" and "Tenure in Months" do not present outliers, other variables show data beyond 1.5 times the distribution's IQR–notably the "Number of Dependents" and "Total Extra Data Charges". The decision made was to keep them throughout the preprocessing stage, and eventually evaluate their impact on the models.

Appendix 2.3 shows the correlation matrix between all numerical attributes in the dataset. The correlation coefficients reveal that there are no significant correlations between the attributes, eliminating the concern about redundant predictors and potential complexities for the models.

b. Categorical Variables

Appendix 2.4 shows the histograms of the nominal categorical variables referred to in the hypotheses, with the last one being the target variable. A brief view suggests a relatively balanced distribution across categories, which are further analyzed and discussed in section 3.6: Resampling Techniques.

3.4. Model Approach

The model approach should be based on a classification model given the problem consists of predicting a qualitative target variable. Moreover, a classification model will allow categorizing customers into known discrete labels (churn and not churn).

According to Forbes[6], it can cost up to 5-7 times more to acquire a customer than to maintain one. Thus, in theoretical terms, the model should prioritize classifying churners correctly rather than reducing incorrectly classified customers that remain. In technical terms, where Positive (1) is churn and Negative (0) is not churn, the model should primarily focus on reducing false negatives. Thus, to assess the goodness of the final models, the Recall indicator should be used, as a recall (TPR) of 1.0 indicates zero false negatives and thus a perfect detection of churning customers. Overall, it should be a priority to know which customers will churn in order to focus efforts on maintaining them.

3.5. Feature Engineering and Feature Selection

a. New Features

In order to evaluate hypothesis two (H2), the California counties' income per capita was evaluated according to the Department of Housing and Urban Development (HUD)'s method of determining very-low income areas and households – those in the 25th percentile within the State[7]. After calculating the 25th percentile of California counties, a dichotomous feature – "Low Income" – was created to determine whether the client resides in a very-low income county (1) or not (0). Additionally, hypothesis four (H4) requires an evaluation of customer behavior in terms of the number of company services they are subscribed to. By transforming all of the individual service subscriptions features into a dichotomous variable where subscribed is coded by 1 and non-subscribed is coded by 0, a feature with the addition of customers' subscriptions was created and named "Add Ons". Finally, to reduce the complexity and potential overfitting attributed to encoding zip codes, the "County" attribute was used to assign customers to greater Regions in the State of California. This new feature was named "Regions".

b. Filter Methods

i. Variance

The first filter method applied for feature selection was VarianceThreshold from the Scikit-learn library. As the graph in **Appendix 2.5** reveals, most features were shown to have a relatively low variance after scaling, with some features like "Age," "Tenure in Months," and "Monthly Charges" having a high variance, which is a good indicator for predictive potential as they show more cases of the target value churn rate. However, with all above a threshold of 0.01 – the lowest being "Number of Dependents" at 0.0114 – no attributes were removed at this point.

ii. Features–Features Correlation

Appendix 2.6 depicts the feature-feature correlation matrix, which gives insight into the presence of any multicollinearity within the data. A few instances of high multicollinearity are noted and theoretically consistent, such as with attributes "Total Charges" and "Total Revenue" with a coefficient of 0.97 and "Add Ons" and "Monthly Charge" at 0.84.

iii. Features–Target Correlation and M_k

The graph in **Appendix 2.7** depicts how strongly each feature correlates with the target "Churn Label". Features like "Internet Service," "Monthly Charges," "Paperless Billing," and "Unlimited Data" have higher positive correlations. On the other hand other columns such as "Contract," "Tenure in Months," and "Satisfaction Score" have negative correlations, implying they may act as deterrents for churn. Aimed at finding the features with the lowest feature-feature correlation and highest feature-target correlation, a test was also conducted using the formula $M = \frac{k\overline{corr}_{cf}}{\sqrt{k+k(k-1)\overline{corr}_{ff}}}$ where the k-th feature to be included will be the one which maximizes M, $\overline{corr}_{cf}$ is the average feature-class correlation, and $\overline{corr}_{ff}$ is the average feature-feature correlation. The graph in **Appendix 2.8** shows that the results, in general, are consistent with the Features–Target order of correlation and thus suggests that the

[6] Saravana Kumar, "Council Post: Customer Retention Versus Customer Acquisition," Forbes, accessed November 23, 2024, https://www.forbes.com/councils/forbesbusinesscouncil/2022/12/12/customer-retention-versus-customer-acquisition/.

[7] "Methodology for Determining Section 8 Income Limits," n.d.

correlation between the feature and the target variable (i.e., how strongly a feature is associated with the target) is not significantly altered by the correlation between the feature and other features in the dataset.

iv. Mutual Information

An attempt at understanding the contribution of each feature in determining churn was made by measuring Mutual Information, which quantifies the "amount of information" obtained about the target when observing each attribute. The results shown in **Appendix 2.9** demonstrate that attributes with a relatively high negative correlation with the target variable, such as "Satisfaction Score" and "Tenure in Months", seem to give the most information about churn. Meanwhile and consistently, features in the middle of the M_k graph demonstrate a very low mutual information score.

Overall, the decision was made to keep all attributes for the models' creation and then evaluate the impact of any relationships between the attribute and their contributions.

3.6. Resampling Technique.

In order to assess whether any kind of resampling techniques need to be applied, the variables of the data frame were checked for their class sizes. In this case, the guidance used was a 90% to 10% rule of thumb for the imbalance; meaning that if there was a variable which contains 90% or more of a single class, it would need to be resampled. However, based on the results obtained in the graph in **Appendix 3**, none of the variables used in the model breaks the 90:10 class size rule. Finally, in the case of the target variable "Churn Label," one can observe quite a high difference between the 2 classes: class "No" (73.46%) and class "Yes" (26.54%). While this difference might seem high and problematic it is actually a good representation of real industry data and will therefore be a good indicator to train the model, which does not need any resampling.

3.7. Train/Validation/Test Split

The final dataset was split into training, validation, and testing sets based on principles learned throughout the course. 50% of the entries were allocated to create the training split (3521 rows). Furthermore, the remaining 50% was divided equally to create the validation and testing split (1761–1761 rows). Entries were selected randomly to minimize selection bias.

3.8. Preprocessing

a. Missing Values

The first steps taken to find missing values were constructing a heatmap of these entries and finding a sum per column. This indicated the total extent of the missing information. With that, it was possible to find multiple columns that contained missing values, those being "Internet Type", "Offer", "Churn Reason" and "Churn Category". These columns had missing values for similar reasons; "Internet Type" had missing values for customers that did not hire an internet service so the solution was to impute "No Internet" for every missing value in that column. As for "Offer", it also only had missing values for customers that did not receive marketing offers, meaning that it was possible to impute "No offer received" for those values. "Churn Reason" and "Churn Category" columns were dropped due to the fact that all missing values were clients that had not churned in the quarter.

b. Dropping Variables

Columns that provided no additional information had to be dropped. During this phase, the columns deleted were "Customer ID", "Dependents", "Country", "State", "City", "Quarter", "Customer Status", "county" and "zip". All of the the location features were substituted by the representative variable "County Income Per Capita", which was merged into the dataset. Variables were also dropped if they had only one unique value such as "Quarter", where all the values were "Q3". The last column that had to be dropped was the Churn Score, since it is a score calculated by the IBM Analytics model. This can bias the model and should not be used as a parameter as it is already a type of prediction.

c. Label Encoding

The label encoder function was used to transform nominal categorical columns into numerical ones. Binary attributes were encoded as 0 or 1, to more easily integrate them into logistic predictors in the future. For other nominal categorical features, one hot encoding was used as they do not follow ordinal behavior.

4. Model Research Development and Results

For the selection of the model, correctly identifying customers who will churn from the telco company is treated with more value. As a result, recalling actual positives that were classified correctly as positives will be a guiding measure for the performance of the models.

Initially, the linear separability of classes was examined. This involved analyzing scatter plots coded with class labels, as illustrated in Appendix 4.1 and the output of the code. From the graphs, it was concluded that, although classes are not perfectly linearly separable, there is still enough separability to create a logistic regression as a baseline model and then explore more complex models such as Random Forests, which do not require linear separability to assess both options and performance.

4.1. Logistic Regression Model

The first model built is a Logistic Regression as it provides a binary outcome prediction ideal for churn rate and allows for probability predictions as well as interpretable coefficients to later test the hypotheses. For the feature selection, a forward feature selection scoring with "recall" was used as it selects the combination of features that optimize the identification of positive instances in the class. Within this forward features selection, a cross validation fold of 5 was used to ensure robust performance (based on recall) on different train test splits to avoid overfitting. **Appendix 4.2** shows the list of features selected for the specific Logistic regression technique proposed by the SFS to maximize recall. This selection of features is coherent with the previous analysis of feature variance and correlation in the previous **Section 3.5b.** The SFS selects features with higher variance and/ or higher absolute value for the M_k metric including, 'Internet Type 2', 'Monthly charge', 'Satisfaction Rate', and 'Tenure in Months'.

To interpret the performance of the model, predictions were made using both the training and test data, and various classification metrics were analyzed. These metrics included Accuracy, Precision, Recall, and F1 score, along with a Classification Report and a Classification Matrix (based on the test set). These can be found in the **Appendix 4.3**, **4.4**, and **4.5**, respectively. From the results, it's observable how the training and testing metrics are generally close which indicates the model should not be overfitted and is performing similarly with unseen data. The metrics ranging from 0.92 to 0.96 across all metrics also imply a low bias. However, there seems to be a generally lower recall in both the training and testing which is counterintuitive for the objective of prioritizing the correct prediction of churn rate. The model might be performing better on the 0 label, yet this is expected due to the class imbalance (there's more data to train the model on predicting not churning). Due to these adequate performance metrics, the features provided by the SFS were maintained.

To further analyze the model, cross-validation was conducted using 5 folds with the data reserved for testing. The results are provided in **Appendix 4.6**. Once again, the metrics generally demonstrated high values, indicating low bias and strong performance across new data. Variance within different folds was also generally low. Further analysis on bias and variance of the model will be performed on **section 4.4**.

To further enhance the model and align with the business objective of improving recall, an optimized threshold was determined using the ROC Curve. **Appendix 4.7** depicts the ROC Curve– plots the True Positive Rate against the False Positive Rate to visualize the separability across the two different classes across all thresholds. Moreover a 'best threshold' was calculated as the one maximizing the square root of the (tpr *(1-fpr)). This Geometric mean is maximized at the threshold of 0.3634. By lowering the threshold from the standard value of 0.5 to 0.3634, a higher recall is achieved. A lower cutoff results in more instances being classified as positive, which aligns with the goal of improving recall. **Appendix 4.8** shows the new classification metrics based on the new fine tuned threshold. Recall is improved from 0.921 in the test data to 0.955 with the new defined threshold, of course this is at the expense of a lower precision (from 0.92 to 0.89). However, this trade-off aligns with the business priorities, as the cost of losing a client is more significant than the cost of targeting marketing efforts toward customers who may churn regardless.

For the interpretability of the model and analysis of the hypothesis it is possible to use the coefficients of the logistic regression exponentiated and calculate the percentage change in odds ratio. A table with the exponentiated coefficients of the model is available in **Appendix 4.14**. This renders it an adequate model for understanding and quantifying the varying impact of different features on Churn rate. This interpretability will be further useful when analyzing the different hypotheses suggested for this project.

4.2. Random Forest Model

The second model created was a Random Forest model. In the initial creation of the model without setting hyperparameters, there were issues of overfitting with the model perfectly classifying all the values in the training

set except for 2, which ultimately led to worse results in the validation set. These results can be seen in **Appendix 4.9** visually representing how the data was being classified through a confusion matrix. In addition, the target metric recall was the lowest performing metric and was far from the score which was aimed for.

A method used to reduce overfitting was introducing hyperparameters and finding the optimal parameters for the chosen specific metric. In order to determine the best set of Hyper Parameters for the problem and model, grid search cross validation was used. This method tested 180 unique combinations on 5 different subsets of data for a total of 900 fits. This ensures that the model is able to achieve a strong balance between underfitting and overfitting, leading to better generalization on the testing data later on. The results are shown below and will be the hyperparameters used for the model. The Ideal hyperparameters for predicting high accuracy in predicting churners are seen in the **Appendix 4.10**.

Moreover, creating the best model required selecting the optimal features. Random Forest's embedded feature importance allows us to perform feature selection and identify the optimal subset of features that maximizes recall for the churn prediction class. In **Appendix 4.11**, the relative feature importance of each variable is shown visually. There was a potential issue with satisfaction score being around 0.5 leading to very high reliance and that specific metric. Moreover, regarding the ideal features to use for the model, different thresholds were created for feature importance and created subsets based on the threshold to later train the model. It was then possible to determine the optimal number of features and which features to optimize the recall for the churn prediction class of the model. Using the validation set, the optimal threshold was determined to be 0.004, which resulted in the selection of 29 out of 50 instances. This threshold not only had the highest recall metric for churn class but additionally highest overall accuracy.

By combining hyperparameter optimization and feature selection, a Random Forest model was successfully developed that prioritizes recall for the churn prediction class while maintaining robust overall performance. Thanks to the model, the telecom company would be able to successfully predict the majority of the churners ensuring they can retain customers which are likely to churn before they end up churning. In the **Appendix 4.12** the evaluation metrics of the model on the testing set can be seen along with the confusion matrix in **Appendix 4.13**.

4.3. Choice of Model and Justification

Modeling Technique

After analyzing both a Random Forest and a Logistic Regression model, the Logistic Regression was selected, despite both models demonstrating high performance. The key reasons for this decision include:

1) Interpretability, given that for this project the objective is not uniquely building the best model but being able to interpret the hypothesis and test relationships within the data. Although random forest feature importance allows us to understand the main features influencing the model as it is an ensemble of decision trees, understanding specific decision paths may be challenging compared to simpler models. By selecting Logistic Regression, the model's coefficients can be exponentiated to derive the change in the odds ratio. This allows for interpreting the likelihood of belonging to a category other than the base class for categorical variables or determining the effect of a unit change for numerical variables. Such property enables analysis of the hypothesis and aids telco companies in not only predicting churn rate, but having available information on how to influence this churn rate by directing marketing offers to the correct customers.
2) Additionally in the analysis of classification metric between both models, when using random forest there was an overreliance on satisfaction score when predicting churn, with a feature importance of 0.49, the model may disproportionately weight this single feature, potentially overlooking other important factors as well as satisfaction score may not always be most accurate metric as it is based on the customer opinion rather than objective data. In logistic regression Satisfaction score also had a dominant effect given that the exponentiated coefficient for this variable is 0.0021 conveying a reduction in odds of 99.79% *(1-0.21%)* per each unit increase in customer satisfaction. While both models suggest the importance of this feature, the direct interpretability and quantification of the effect in Logistic Regression allows us to better assess and understand the influence of this dominant feature.
3) Moreover despite carrying out a Random Forest after the initial inspection which depicted how classes where not perfectly linearly separable across all variables the Logistic Regression performed well on classification metrics as seen in **Appendix 4.8**, it was reasonable to assume that the model is handling the data effectively and there is no need for extra complexity in handling complex relationships. Therefore a random forest model may be overcomplicating the problem.
4) Another advantage of the Logistic regression model would be the possibility it offers to be fine tuned with a specific lower threshold which directly addresses the issue of prioritizing recall. Although in Random Forests

precision and recall may be influenced indirectly through parameter tuning (e.g increasing tree depth, increasing max nodes etc,) this would come at the cost of the random forest making more splits and creating finer partitions of the feature space introducing therefore more complexity to the interpretability of model and possible overfitting. These tradeoffs of fine tuning a complex Random Forest seem counterintuitive with such low bias in a simpler more interpretable logistic regression. As observable in the results below Logistic Regression due to the fine tuning of the threshold better prioritizes recall, although at the expense of precision, while maintaining high accuracy and F1 scores.

Random Forest Model	**Logistic Regression Model**
Model Evaluation Metrics (Test Set): Accuracy: 0.9625 Precision: 0.9653 Recall: 0.8910 F1-Score: 0.9267	The metrics for the test dataset are: Precision: 0.887 Recall: 0.955 Accuracy: 0.956 F1 Score: 0.920

Justification Selected Features and Parameter Tuning within Logistic model

Within the selected modeling technique, two important decisions were made in the feature selection and fine tuning of the logistic regression.

1) For the feature selection of the model, it was decided to select those provided by the sequential Forward feature selection with a cross validation of 5 folds and a scoring of recall, to optimize for the correct prediction of Churn rate (label 1). The rationale behind this choice is based on the business objective where incorrectly predicting a churning customer (false negative) is more costly than wrongly predicting churn (false positive) for a telco company.
2) Moreover, a threshold of 0.3634 was chosen for the model. This selection is justified through the development of the model in **Section 4.1** yet the basic rationale is improving separability of classes and by lowering the threshold from the standard (0.5) improving recall.

4.4 Analysis of the Chosen Model

Analyzing Bias and Variance:

Using the chosen Logistic Model, a superficial awareness of bias and variance can be deducted by comparing the metrics of the training and testing data on **Appendix 4.4**. There is a very good similarity between the training and testing metrics which suggests that the model has low variance and generalizes well to new data. Iit also indicates low bias as the model is capturing the underlying patterns well on both datasets.

- The difference in precision is minimal (0.888 vs. 0.871), implying that the model maintains similar true positive rates across training and testing sets.
- Recall is slightly higher on the testing set compared to the training set (0.968 vs. 0.952), suggesting that the model is not underfitting.
- The accuracy difference is also very small, indicating no significant overfitting.
- The consistency in F1 score (0.919 for training vs. 0.917 for testing) further supports that there is no major bias or variance issue affecting model predictions.

In addition, the results from the 5-fold cross-validation can be referenced to provide further insights into the model's ability to generalize effectively (Graph 4.6). The consistent metrics across the folds without significant fluctuations suggest a well-balanced model with minimal bias and variance.

To further evaluate the model's features and performance, a more rigorous cluster analysis was conducted. PCA was implemented to reduce dimensionality and then K-Means clustering was performed to identify patterns among different segments of the data. This allowed us to understand how the model behaves across various sub-groups within the dataset and uncover any biases or performance disparities that may not be visible in an overall analysis.

Using the Elbow Method and Silhouette Scores, it was identified that 3 clusters were optimal (**Appendix 4.21**). The relatively low Silhouette Score of 0.23 (**Appendix 4.22**) suggests limited separation between clusters, which might indicate overlapping or less distinct groups. K-Means Clustering was used to create the clusters, and

labels were assigned to both the training and testing datasets to evaluate model performance across the identified groups.

As can be seen in **Appendix 4.23**, Cluster 0 displayed balanced precision, recall, and accuracy, indicating consistent but slightly lower model performance compared to the other clusters. Recall was higher in the testing set compared to training, implying some variance within this cluster. Cluster 1 exhibited strong performance with high recall and F1 score, suggesting low bias and variance, with consistent metrics across both training and testing sets. Cluster 2 had lower recall on both training and testing sets, indicating the model often missed positive cases in this group. This suggests higher bias, possibly due to unique characteristics in the cluster that made classification more challenging.

In **Appendix 4.24**, the characteristics of each cluster were ambiguous. Cluster 0 comprised customers with moderate engagement levels across features, resulting in balanced but average model performance. Cluster 1 included customers with stable service engagement, high contract retention, and consistent usage patterns. This predictability led to strong model performance. Cluster 2 showed greater variability, especially in features like senior citizen status and dependents. This diversity likely contributed to the model's lower recall, indicating challenges in generalizing well for this cluster.

Overall, the clusters lacked clear distinctions, reflected in the low Silhouette Score of 0.23. The overlap between clusters complicated the identification of unique group behaviors. Nevertheless, it is still possible to interpret that cluster 2 exhibited some bias, indicated by lower recall scores, suggesting the model struggled to correctly identify positive cases for this group. Clusters 0 and 1 showed strong, consistent performance, suggesting the model effectively captured patterns for these groups. To enhance the model, targeted adjustments for poorly performing clusters like Cluster 2 could improve overall accuracy and fairness. A limitation of this analysis is the low Silhouette Score (0.23), which suggests significant overlap between clusters, potentially reducing the effectiveness of distinguishing distinct sub-groups.

Analysis of Hypotheses

Hypothesis 1: *Customers with monthly contracts are more likely to churn compared to ones with yearly and bi-yearly contracts.*

Model Findings

Using the Logistic Regression model, the hypotheses can be evaluated by comparing the baseline category set to monthly contracts and the coefficients for one-year and two-year contracts to measure the difference in churn likelihood relative to the baseline.

Coefficients

Feature	Coefficient	Odds Ratio	Interpretation
Contract_1 (One Year)	-1.937	0.144	Customers with one-year contracts are 85.6% less likely to churn compared to monthly contracts.
Contract_2 (Two Year)	-3.401	0.033	Customers with two-year contracts are 96.7% less likely to churn compared to monthly contracts.

Key Interpretations

1. Contract_1 (One Year): The negative coefficient (−1.937-1.937) and odds ratio (0.1440.144) indicate that customers with one-year contracts are significantly less likely to churn than those with monthly contracts. Specifically, the likelihood of churn decreases by approximately 85.6% for one-year contract holders.
2. Contract_2 (Two Year): The coefficient (−3.401-3.401) and odds ratio (0.0330.033) demonstrate an even greater reduction in churn likelihood for customers with two-year contracts compared to monthly contracts. These customers are about 96.7% less likely to churn.

Both coefficients for one-year and two-year contracts are negative and highly significant, indicating a substantial reduction in churn likelihood for these groups relative to the baseline. Two-year contracts show the greatest retention benefit, with customers being the least likely to churn.

Hypothesis 3: *Customers paying via mailed checks have a higher churn rate compared to those using bank withdrawal or credit cards.*

Logistic Regression Results:

1. Feature Evaluated: Payment Method_2 (Mailed Checks)
    - Coefficient: -0.247
    - Odds Ratio: 0.781
2. The coefficient of -0.247 indicates that customers paying via mailed checks have lower odds of churning compared to the baseline category (bank withdrawal). The odds ratio of 0.781 implies that customers using mailed checks are approximately 21.9% less likely to churn than those using bank withdrawal (1 - 0.781 = 0.219). This coefficient is weak and does not suggest a significant effect, however it does contradict the hypothesis immediately as the relationship predicted is the opposite of what is actually happening.
3. Feature Selection Context:
    - The chosen model used a feature selection process to retain only one payment method variable: Payment Method_1 (Credit Cards). However, by directly evaluating Payment Method_2 it was then possible to manually examine the relationship between mailed checks and churn. This step allows us to draw insights despite Payment Method_2 being excluded in the selected feature set for the primary model.

Overall the analysis does not support the hypothesis that customers paying via mailed checks have a higher churn rate.

Evaluating Dual Relationship Hypotheses:

Hypothesis 2, Gender and Income Interaction on Churn: *In counties with very-low income per capita, a higher churn rate is associated with women than with men, and the opposite behavior is expected in non very-low income counties.*

Methodology:

To test this hypothesis, a more elaborate logistic model was employed, incorporating the following features:

1. Gender (Female): Binary indicator for gender.
2. Low Income (Very-Low): Binary indicator for customers residing in counties with very-low per capita income.
3. Gender * Low Income Interaction: Interaction term representing the combined effect of gender and income status.

The interaction can be formally represented as:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1(\text{Gender}) + \beta_2(\text{Low Income}) + \beta_3(\text{Gender} \times \text{Low Income})$$

This methodology differs from the approach used in Hypothesis 1, where the logistic regression model included only categorical variables related to contract types. Here, the interaction term is specifically included to capture the dual relationship hypothesis between gender and income. This allowed us to test how the effect of gender on churn changes depending on income status.

Results:

The logistic regression model produced the following coefficients and odds ratios:

Feature	Coefficient	Odds Ratio	Interpretation

Gender (Female)	0.001834	1.001836	Gender alone has a negligible effect on churn; females have nearly the same odds of churn as males.
Low Income (Very-Low)	-0.097327	0.907259	Customers in very-low-income counties are 9.3% less likely to churn than those in higher-income counties.
Gender * Low Income Interaction	-0.052196	0.949143	Females in very-low-income counties are 5.1% less likely to churn compared to males in non-very-low-income counties.

The intercept of -1.0000 represents the baseline log-odds of churn when all predictors are set to zero. It was not a relevant focus of interpretation for this hypothesis.
Interpretation:

1. The Gender (Female) coefficient indicates that gender alone has an almost negligible effect on churn rates. This suggests that, independent of income, being female does not strongly predict churn.
2. The Low Income (Very-Low) coefficient shows that customers in very-low-income counties are slightly less likely to churn compared to those in non-very-low-income counties.
3. The Gender * Low Income Interaction coefficient indicates that, contrary to the hypothesis, females in very-low-income counties are less likely to churn than males, rather than more likely.

The results do not support Hypothesis 2. Instead, the interaction term suggests that females in very-low-income counties exhibit a slightly reduced churn rate, which again is the opposite of what was expected. While the interaction term is statistically significant, the magnitude of its effect is relatively small, suggesting that the hypothesised gender-income relationship does not strongly influence churn.

Hypothesis 4: *Customers subscribing to multiple company services are paradoxically more likely to churn if their satisfaction score drops below a threshold of 3, as their expectations of service quality are higher.*

Methodology:

For this hypothesis, the aim was to understand how the number of services subscribed (represented by the "Add Ons" variable) interacts with low satisfaction scores (below a threshold of 3). Similar to hypothesis 2, it was tested whether customers who subscribed to multiple services and experienced low satisfaction were more likely to churn due to their higher expectations of service quality.

As before, the following model was created:

1. Low Satisfaction: A binary variable, where a value of 1 represents customers whose satisfaction score is below 3, and 0 represents those with a score of 3 or higher.
2. Interaction Term: A term between Add Ons and Low Satisfaction to capture any combined effect of these variables on churn.
3. Logistic Regression Model: A logistic regression model with three features:
    - Add Ons: The number of additional services subscribed by the customer.
    - Low Satisfaction: Whether the satisfaction score is below 3.
    - Add Ons * Low Satisfaction Interaction: The interaction between the number of add-ons and satisfaction score.

More formally, the interaction can be expressed as:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1(\text{Add Ons}) + \beta_2(\text{Low Satisfaction}) + \beta_3(\text{Add Ons} \times \text{Low Satisfaction})$$

Results:
The results of the logistic regression model are as follows:

Feature	Coefficient	Odds Ratio	Interpretation
Add Ons	-1.937	0.144	For each additional service subscribed, the likelihood of churn decreases significantly (about 85.6% less likely to churn), when considered independently.

Low Satisfaction	3.733	41.801	Indicates that customers with a satisfaction score below 3 are much more likely to churn, with the odds of churn being approximately 42 times greater for these customers compared to those with higher satisfaction scores.
Add Ons * Low Satisfaction	1.878	6.537	Suggests that when a customer subscribes to multiple services and has a low satisfaction score, the likelihood of churn increases substantially (by about 6.5 times) compared to those with low satisfaction but fewer services

Interpretation:

1. Add Ons:
   The coefficient for Add Ons is -1.937, and the corresponding odds ratio is 0.144. This could imply that customers who have more services are generally more loyal, unless affected by other factors.
2. Low Satisfaction:
   The coefficient for Low Satisfaction is 3.733, and the odds ratio is 41.801. This reinforces the idea that low satisfaction is a strong predictor of churn.
3. Interaction Term (Add Ons * Low Satisfaction):
   The coefficient for the interaction term is 1.878, and the odds ratio is 6.537. This supports the hypothesis that customers with more services and low satisfaction have a higher likelihood of churning, likely due to higher expectations from the multiple services they are subscribed to.

Based on these results, Hypothesis 4 is supported. Customers who subscribe to multiple services (Add Ons) and have low satisfaction scores (below 3) are indeed more likely to churn, with the interaction term showing a significant increase in the likelihood of churn. This suggests that the expectations of customers with multiple services are high, and when their satisfaction drops, their likelihood of churn rises substantially.

5. Conclusion

To find relationships among variables that could explain churn rate, the team conducted two types of classification models. The first was a Logistic Regression classification model which used forward feature selection that resulted in an optimization of recall. The selected variables had high variance and significant correlation with the target metric. The performance metrics derived from the Logistic Regression all indicated low bias and good generalization, but the recall was lower due to class imbalance – this was then improved with the ROC curve at the expense of precision.

The second model was the Random Forest which initially suffered from overfitting as it perfectly classified the training data but performed poorly in validation. After conducting hyperparameter optimization, the model was refined, improving recall and removing overfitting. This model put a lot of emphasis into the variable satisfaction, which wasn't great as it risks over-reliance on a single feature.

After examining both, the team decided that the Logistic Regression was the best model for the purpose of this analysis as interpretability is one of the most important characteristics to be considered. This model achieved a recall of 0.968 in the testing set, giving a good guidance measure for the real-world application of the model. The Logistic Regression, being more interpretable, permits an easier testing of hypotheses and allows for more actionable insights for telecommunications companies. In addition, the Logistic Regression had a more balanced performance compared to Random Forest (which relied heavily on a single variable) without increasing complexity.

With this, the team learned the importance of not overlooking simple solutions as oftentimes they turn out to be the most effective. At the beginning of the process, a higher complexity was associated with the possibility of better results or predictions, but as the work progressed, this idea slowly started to fade. The team learned that even if there is a very slight optimization in the results, interpretability is still more important when it comes to applying analysis to real-world problems.

Furthermore, when proving the hypotheses it was found that customers with monthly-contracts are associated with higher churn rates and that customers subscribed to multiple services are more likely to churn if their satisfaction score drops below 3, supporting hypothesis 1 and 4. When reflecting on customer behaviors, it makes intuitive sense that the more services a customer experiences, the higher their expectations will be; making it easier for them to cancel their subscription as soon as their expectation is not met.

The rejection of hypothesis 2 brings many questions as it states that customers in very-low income counties are less likely to churn compared to those in non very-low income counties. It also demonstrates that gender in itself has a negligible effect on churn rate. In addition, the rejection of hypothesis 3 came as a surprise to the team demonstrating that a more traditional method exhibited a lower probability of churning.

The results of these tests, and especially of the hypotheses, give insight into the factors that can lead to customers churning. Even though the results obtained are for the fictional company Telco, the team believes that they can be generalized to the telecommunications industry. That being said, it is important to ask more questions and investigate deeper, like why do customers paying via mailed checks have a less probability of churning? Is it related to their age? If so, does this indicate a common brand loyalty behavior for that generation? Finding answers to more and more questions will help businesses in this field save millions of dollars by knowing which type of clients to target and avoiding the costs associated with recuperating churning clients.

6. Appendix

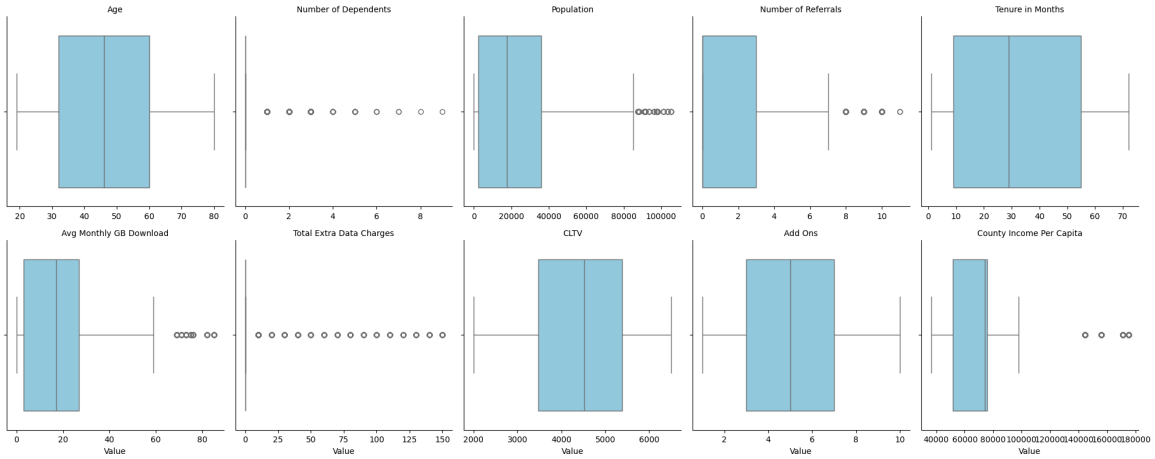
Appendix 1: All other variables in the dataset.

- **Customer ID**: A unique ID that identifies each customer.
- **Age**: The customer's current age, in years, at the time the fiscal quarter ended.
- **Under 30**: No for customers that are over the age of 30 and Yes for customer under the age of 30
- **Senior Citizen**: Indicates if the customer is 65 or older: Yes, No.
- **Married**: Indicates if the customer is married: Yes, No.
- **Dependents**: Indicates if the customer lives with any dependents: Yes, No. Dependents could be children, parents, grandparents, etc.
- **Number of Dependents**: Indicates the number of dependents that live with the customer.
- **Country**: The country of the customer's primary residence.
- **State**: The state of the customer's primary residence.
- **City**: The city of the customer's primary residence.
- **Zip**: The zip code of the customer's primary residence.
- **Latitude**: The latitude of the customer's primary residence.
- **Longitude**: The longitude of the customer's primary residence.
- **Population**: A current population estimate for the entire zip code area.
- **Quarter**: The fiscal quarter that the data has been derived from (e.g. Q3).
- **Referred a Friend**: Indicates if the customer has ever referred a friend or family member to this company: Yes, No.
- **Number of Referrals**: Indicates the number of referrals to date that the customer has made.
- **Tenure in Months**: Indicates the total amount of months that the customer has been with the company by the end of the quarter specified above.
- **Offer**: Identifies the last marketing offer that the customer accepted, if applicable.
- **Phone Service**: Indicates if the customer subscribes to home phone service with the company: Yes, No.
- **Avg Monthly Long Distance Charges**: Indicates the customer's average long distance charges, calculated to the end of the quarter specified above.
- **Internet Type**: Indicates the type of internet service subscribed by the customer: None, Fiber Optic, DSL, Cable.
- **Avg Monthly GB Download**: Indicates the customer's average download volume in gigabytes, calculated to the end of the quarter specified above.
- **Paperless Billing**: Indicates if the customer has chosen paperless billing: Yes, No.
- **Monthly Charge**: Indicates the customer's current total monthly charge for all their services from the company.
- **Total Charges**: Indicates the customer's total charges, calculated to the end of the quarter specified above.
- **Total Refunds**: Indicates the customer's total refunds, calculated to the end of the quarter specified above.
- **Total Extra Data Charges**: Indicates the customer's total charges for extra data downloads above those specified in their plan, by the end of the quarter specified above.
- **Total Long Distance Charges**: Indicates the customer's total charges for long distance above those specified in their plan, by the end of the quarter specified above.
- **Total Revenue**: Total revenue earned by the company from the specific customer
- **Customer Status**: Indicates the status of the customer at the end of the quarter: Churned, Stayed, or Joined.
- **Churn Score**: A value from 0-100 that is calculated using the predictive tool IBM SPSS Modeler.
- **CLTV**: Customer Lifetime Value.
- **Churn Category**: A high-level category for the customer's reason for churning: Attitude, Competitor, Dissatisfaction, Other, Price.
- **Churn Reason**: A customer's specific reason for leaving the company.
- **County**: Where the customer lives based on the ZIP Code.

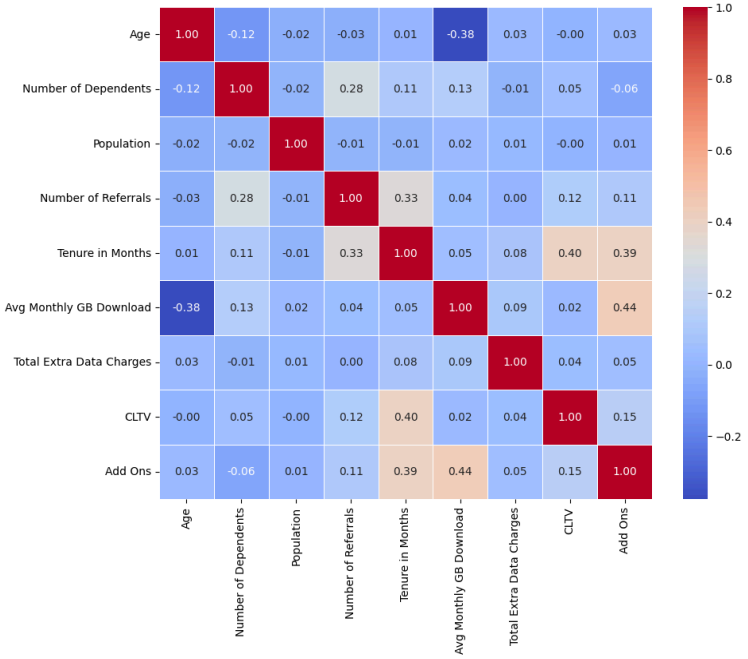
Appendix 2.1: Summary Statistics of Numerical Variables Pertinent to Hypotheses

	County Income Per Capita	Add Ons
count	7043.000000	7043.000000
mean	72850.579725	4.751384
std	28667.498530	2.691717
min	36314.000000	1.000000
25%	51788.000000	3.000000
50%	74142.000000	5.000000
75%	75720.000000	7.000000
max	175070.000000	10.000000

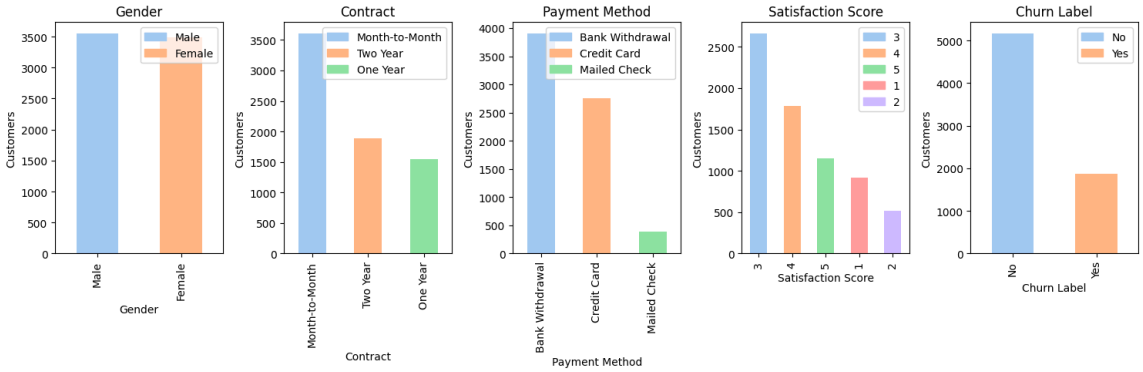
Appendix 2.2: Boxplots of Dataset Numerical Variables



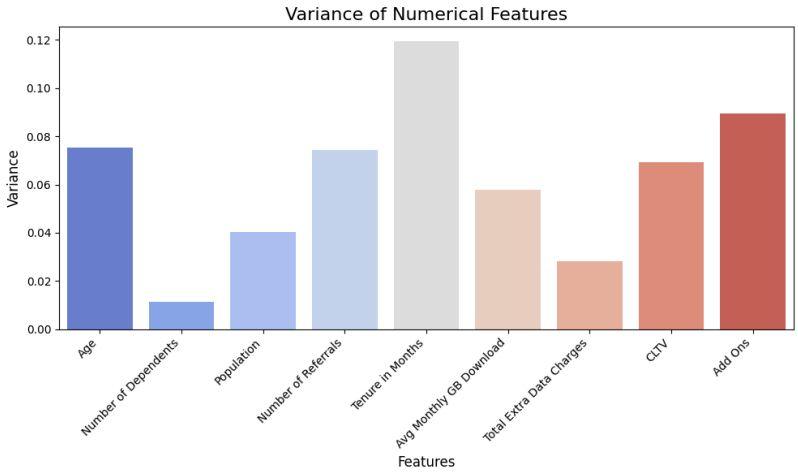
Appendix 2.3: Correlation Matrix of Numerical Attributes



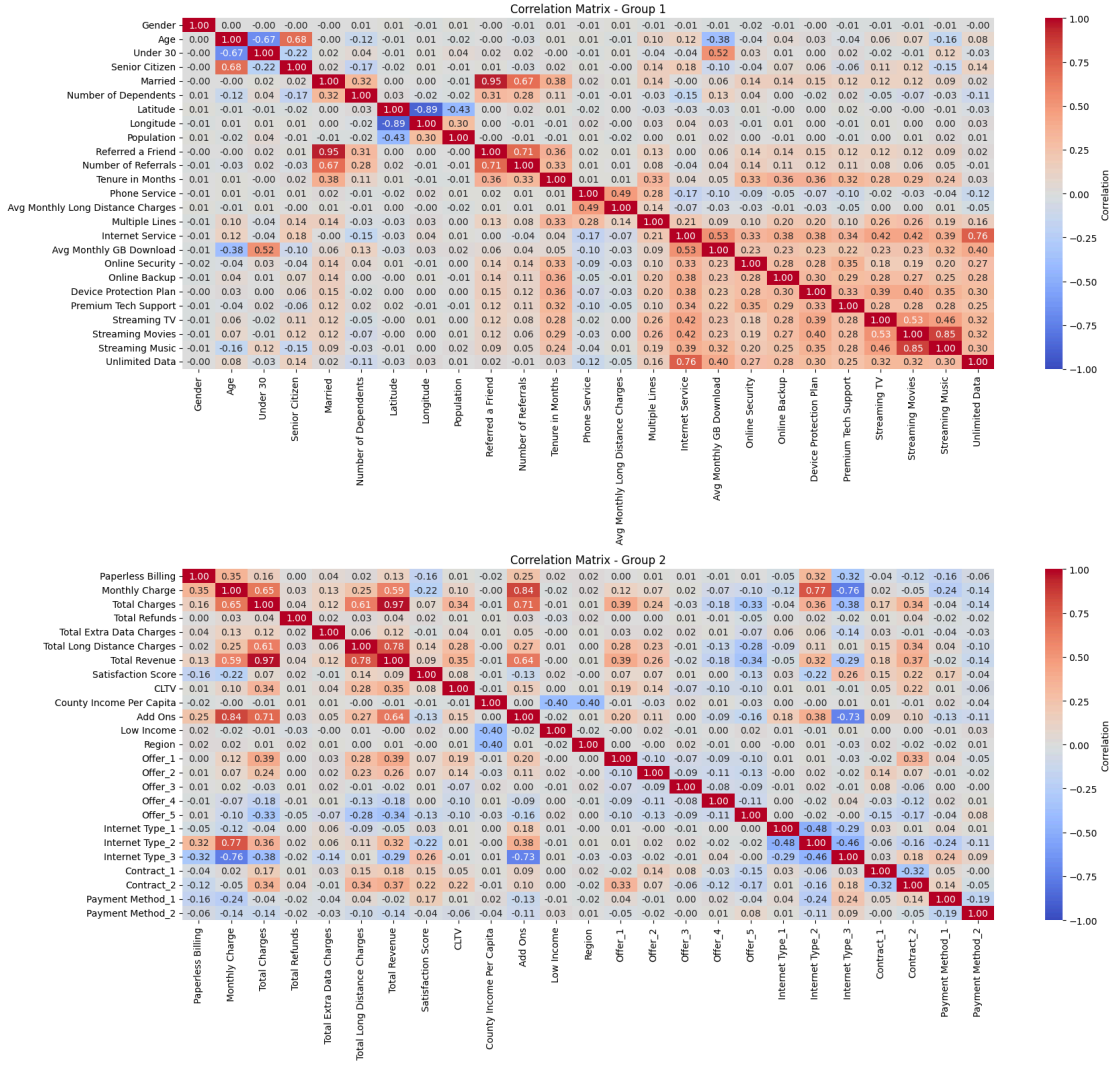
Appendix 2.4: Histograms of Nominal Attributes



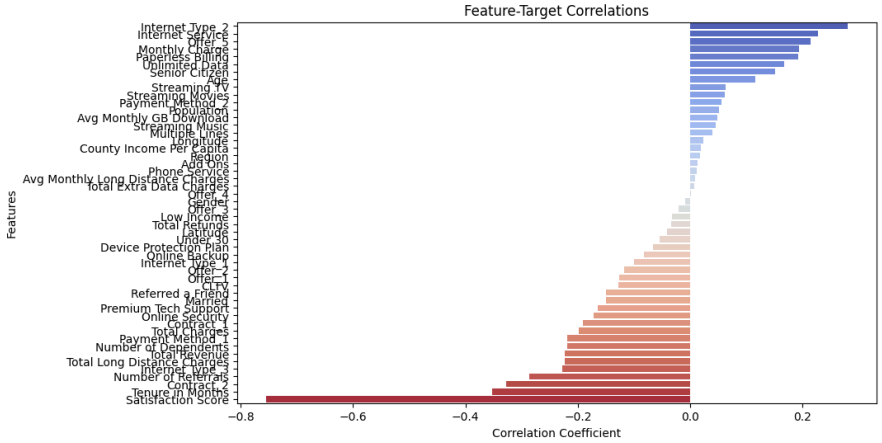
Appendix 2.5: Variance of Numerical Features



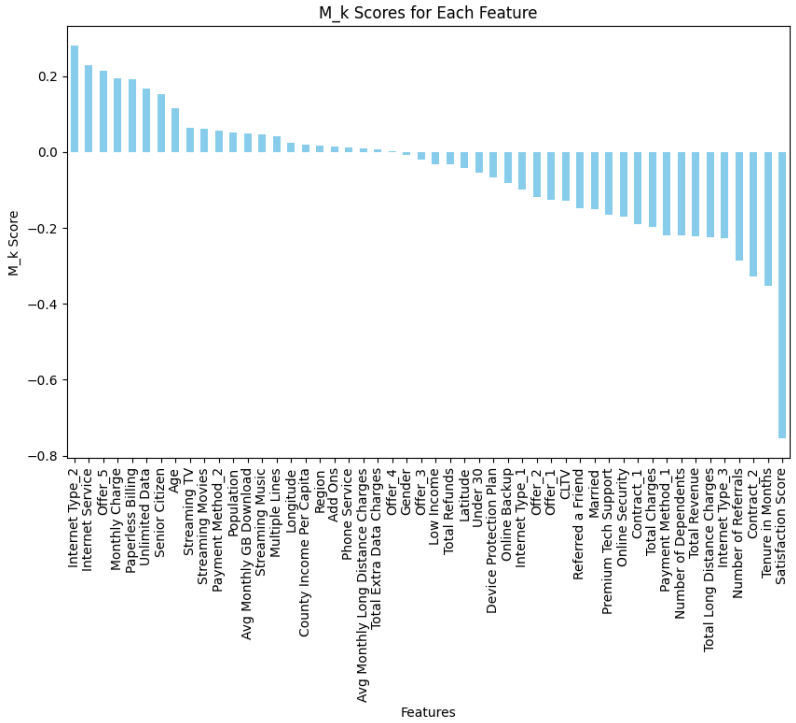
Appendix 2.6: Feature-Feature Correlations



Appendix 2.7: Feature-Target Correlations



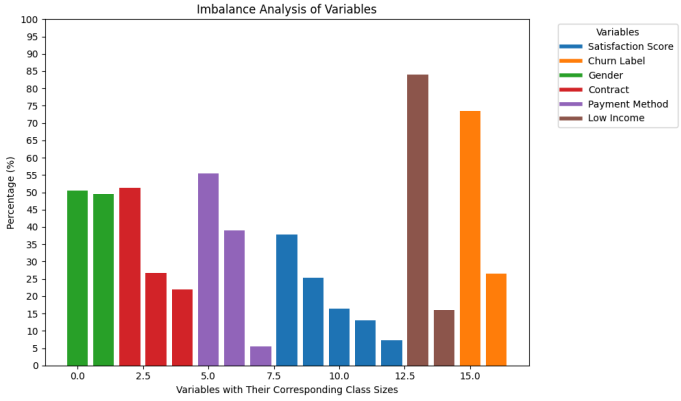
Appendix 2.8: M_k Metric Results



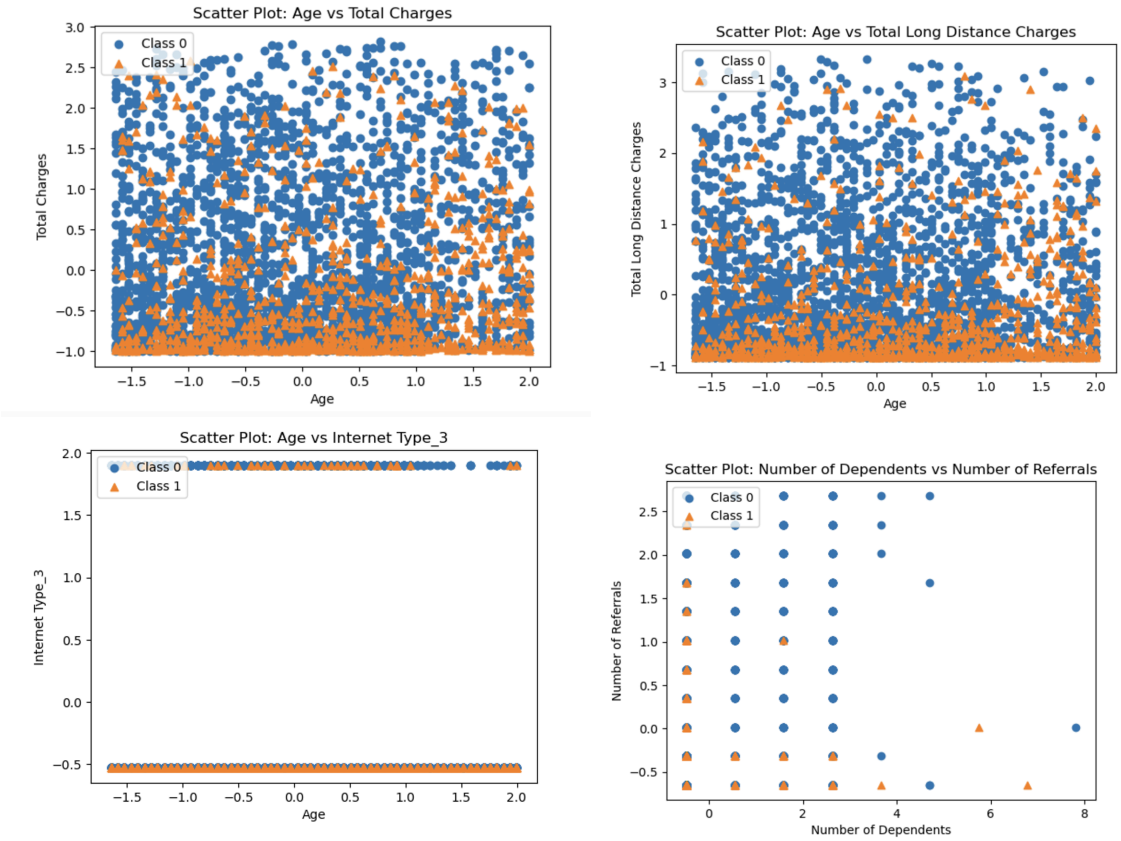
Appendix 2.9: Mutual Information Results

Feature	Mutual Information
Satisfaction Score	0.412971
Tenure in Months	0.076486
Number of Referrals	0.073307
Contract_2	0.067436
Monthly Charge	0.051433
Internet Type_2	0.051415
Add Ons	0.050404
Total Long Distance Charges	0.048022
Total Charges	0.042595
Total Revenue	0.040274
Internet Service	0.035467
Longitude	0.033514
Latitude	0.031631
Avg Monthly GB Download	0.031536
Number of Dependents	0.031145
Internet Type_3	0.028774
Contract_1	0.027483
Population	0.025732
Paperless Billing	0.022842
Payment Method_1	0.022834
Offer_5	0.021800
Online Security	0.018217
Premium Tech Support	0.016921
Offer_1	0.014989
Streaming TV	0.013980
Age	0.012993
Unlimited Data	0.012676
Online Backup	0.009364
Referred a Friend	0.007104
County Income Per Capita	0.006750
Married	0.006517
Total Extra Data Charges	0.006351
Device Protection Plan	0.006090
Gender	0.005290
Senior Citizen	0.004764
Internet Type_1	0.004760
Low Income	0.003037
Phone Service	0.002956
Streaming Music	0.002790
Under 30	0.002274
Region	0.002157
Offer_4	0.001717
Payment Method_2	0.001225
CLTV	0.000554
Offer_3	0.000306
Total Refunds	0.000000
Offer_2	0.000000
Streaming Movies	0.000000
Multiple Lines	0.000000
Avg Monthly Long Distance Charges	0.000000

Appendix 3: Imbalance Analysis of Variables Directly Pertinent to Hypotheses



Appendix 4.1: Linear Separability Within Classes



Appendix 4.2: Features Selected Logistic Regression.

```
Selected features: ['Gender', 'Senior Citizen', 'Married', 'Number of Dependents', 'Longitude', 'Referred a Friend', 'Number of Referrals',
'Tenure in Months', 'Avg Monthly Long Distance Charges', 'Online Security', 'Online Backup', 'Device Protection Plan', 'Unlimited Data', 'Mon
thly Charge', 'Total Extra Data Charges', 'Total Long Distance Charges', 'Satisfaction Score', 'Region', 'Offer_1', 'Offer_2', 'Offer_5', 'In
ternet Type_2', 'Contract_1', 'Contract_2', 'Payment Method_1']
```

Appendix 4.3: Test and Train Metrics for SFS Logistic Regression

```
Metrics(y_train, y_train_pred_f, data='train')
```

```
The metrics for the train dataset are:
Precision: 0.949
Recall: 0.915
Accuracy: 0.964
F1 Score: 0.932
```

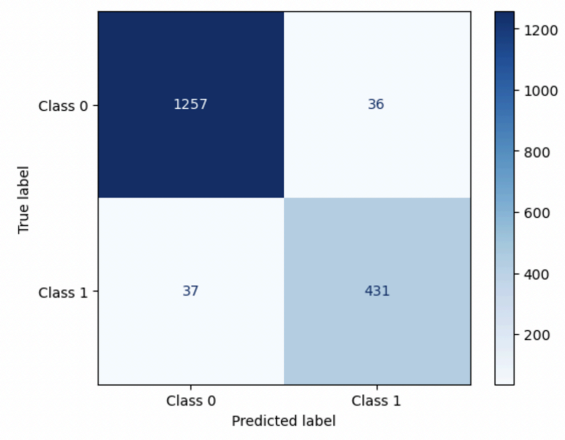
```
Metrics(y_test, y_test_pred_f, data='test')
```

```
The metrics for the test dataset are:
Precision: 0.923
Recall: 0.921
Accuracy: 0.959
F1 Score: 0.922
```

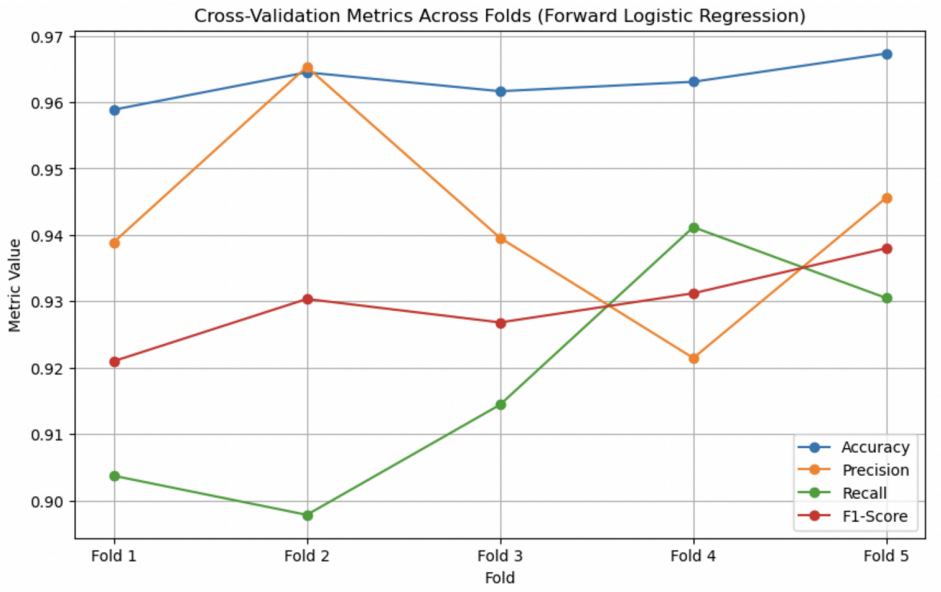
Appendix 4.4: Classification Report. (On Test data)

```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1293
           1       0.92      0.92      0.92       468

    accuracy                           0.96      1761
   macro avg       0.95      0.95      0.95      1761
weighted avg       0.96      0.96      0.96      1761
```

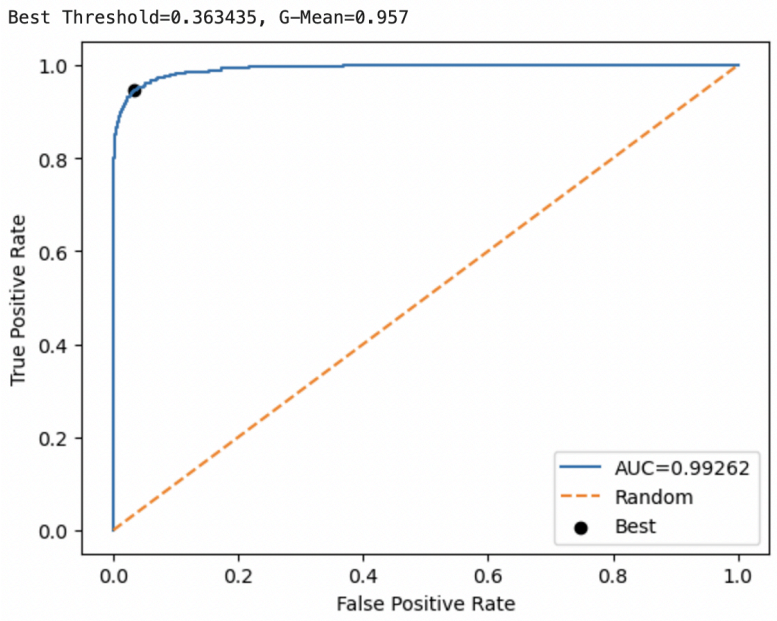
Appendix 4.5: Classification Matrix Logistic Regression. (On Test data)



Appendix 4.6: Cross Validation Logistic Regression



Appendix 4.7: ROC Curve Logistic Regression



Appendix 4.8: Logistic Regression Metrics - Threshold Optimised

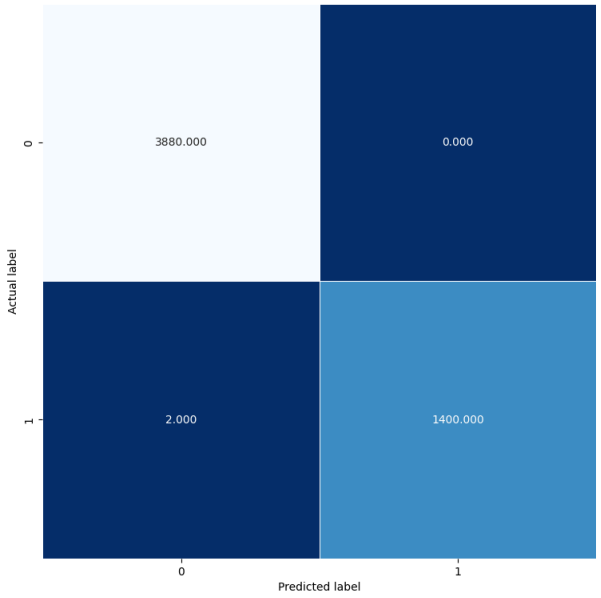
```
Metrics(y_train, y_train_pred_f_t, data='train')
```

```
The metrics for the train dataset are:
Precision: 0.909
Recall: 0.948
Accuracy: 0.961
F1 Score: 0.928
```

```
Metrics(y_test, y_test_pred_f_t, data='test')
```

```
The metrics for the test dataset are:
Precision: 0.887
Recall: 0.955
Accuracy: 0.956
F1 Score: 0.920
```

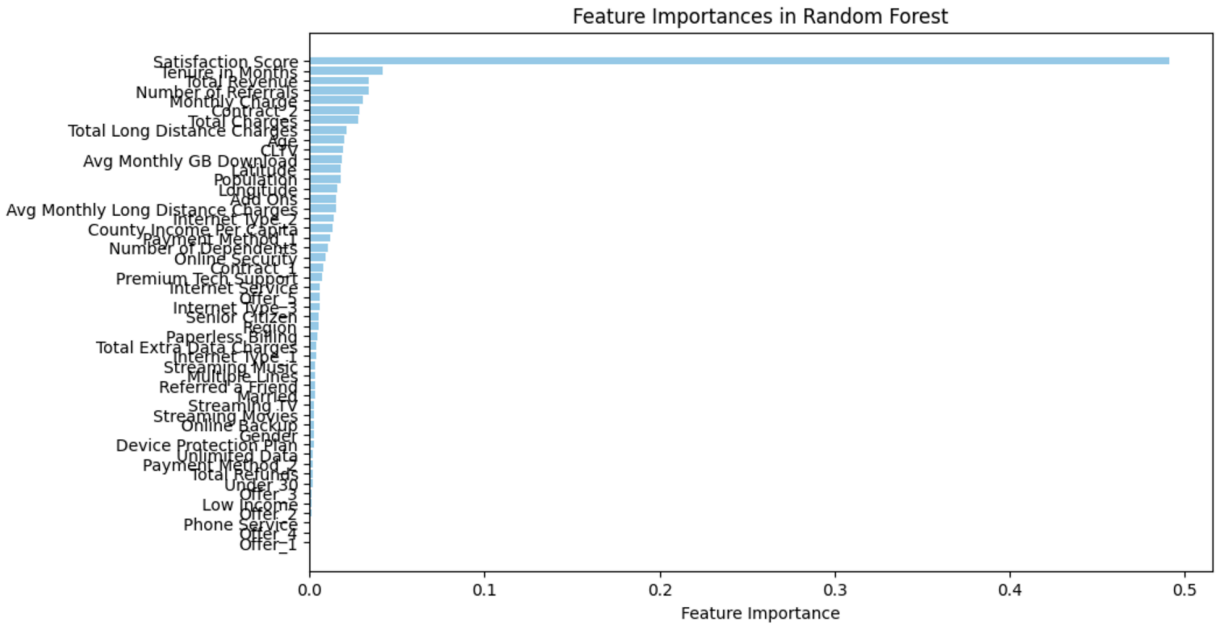
Appendix 4.9: Confusion Matrix Random Forest for training data.



Appendix 4.10: Ideal Hyperparameter to optimize recall for churn class

```
RandomForestClassifier(max_depth=10, max_leaf_nodes=100, min_samples_split=10,
                       n_estimators=200)
```

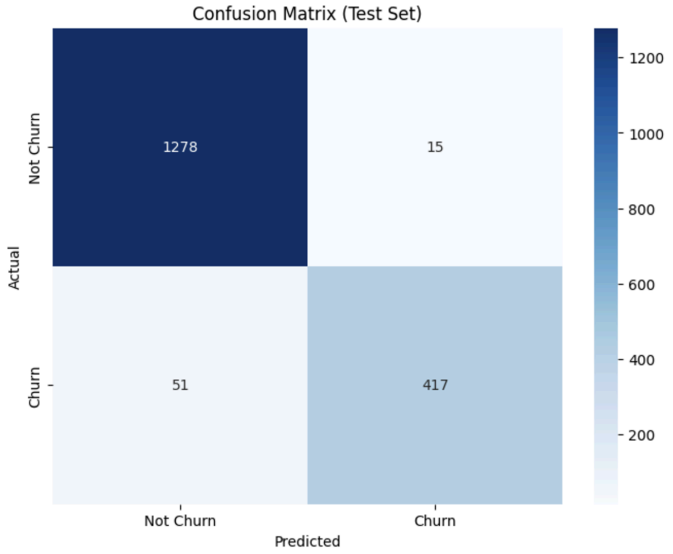
Appendix 4.11: Feature importance by Variable



Appendix 4.12: Random Forest evaluation metrics on test set after hyperparameter tuning and feature selection

```
Model Evaluation Metrics (Test Set):
Accuracy:  0.9625
Precision: 0.9653
Recall:    0.8910
F1-Score:  0.9267
ROC-AUC:   0.9882
```

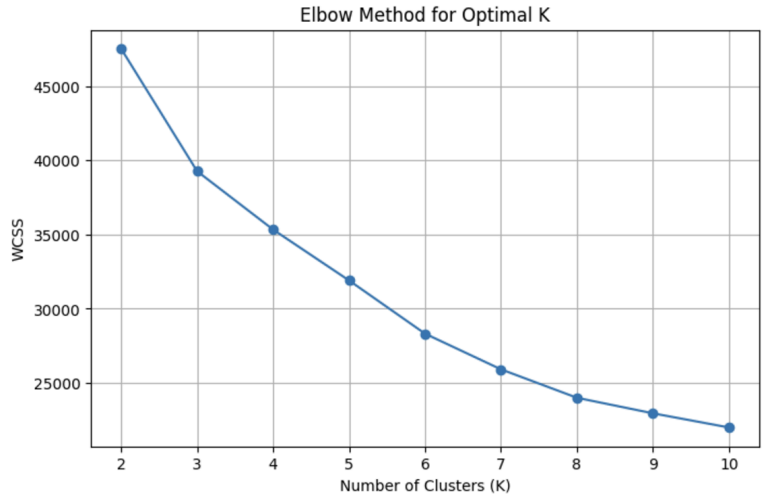
Appendix 4.13: Random Forest Confusion Matrix of test set after hyperparameter tuning and feature selection



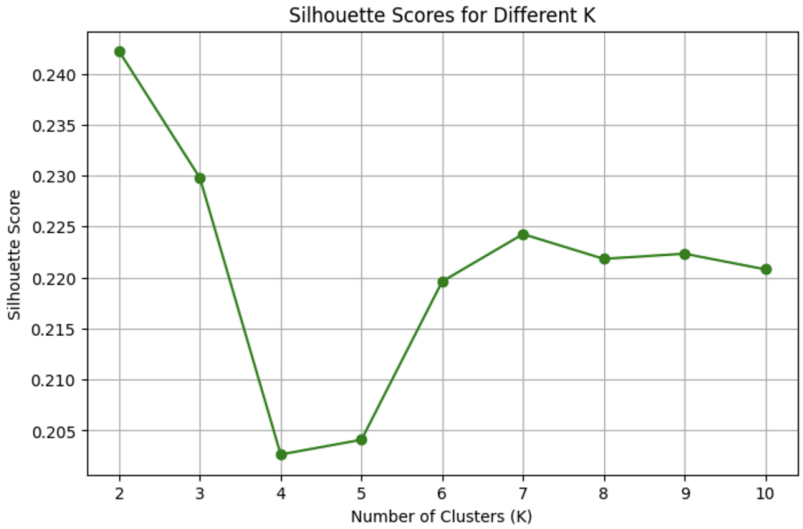
Appendix 4.14: Logistic Regression Coefficients & Exponentiated for Odds Ratios

	Feature	Coefficient	Odds Ratio
0	Gender	-0.009644	0.990402
1	Senior Citizen	0.295439	1.343716
2	Married	-0.056012	0.945527
3	Number of Dependents	-0.602722	0.547320
4	Longitude	0.034058	1.034644
5	Referred a Friend	0.851771	2.343795
6	Number of Referrals	-1.717371	0.179538
7	Tenure in Months	-0.934770	0.392676
8	Avg Monthly Long Distance Charges	-0.247141	0.781031
9	Online Security	-1.293775	0.274234
10	Online Backup	-0.102383	0.902684
11	Device Protection Plan	0.022061	1.022306
12	Unlimited Data	0.064334	1.066448
13	Monthly Charge	0.736667	2.088962
14	Total Extra Data Charges	-0.020293	0.979911
15	Total Long Distance Charges	0.510996	1.666951
16	Satisfaction Score	-6.164972	0.002102
17	Region	-0.017247	0.982901
18	Offer_1	0.303078	1.354020
19	Offer_2	-0.017467	0.982685
20	Offer_5	0.186375	1.204874
21	Internet Type_2	-0.059008	0.942699
22	Contract_1	-0.250176	0.778663
23	Contract_2	-0.792149	0.452871
24	Payment Method_1	-0.355728	0.700663
25	Intercept	-4.069572	0.017085

Appendix 4.21 Elbow Analysis for Clusters



Appendix 4.22: Silhouette scores for Cluster



Appendix 4.23: Training and testing metrics per cluster

Training set metrics for clusters

Cluster	Precision	Recall	Accuracy	F1_Score
0	0.929032	0.935065	0.982067	0.932039
1	0.905371	0.973865	0.935417	0.938370
2	0.891892	0.622642	0.973626	0.733333

Testing set metrics for clusters

Cluster	Precision	Recall	Accuracy	F1_Score
0	0.859155	0.968254	0.978495	0.910448
1	0.887805	0.973262	0.920680	0.928571
2	0.956522	0.709677	0.979879	0.814815

Appendix 4.24: Key Features of Each cluster

```
Cluster Summary (Mean and Standard Deviation per Feature):

           Gender           Senior Citizen             Married            \
             mean       std           mean       std      mean       std
Cluster
0        0.004946  1.000230       0.113996  1.093929  0.695411  0.750969
1       -0.027222  1.000122       0.116352  1.095593 -0.640246  0.739501
2        0.016785  1.000055      -0.328077  0.539563  0.112891  0.997812

        Number of Dependents           Longitude            ...   Offer_5  \
                        mean       std      mean       std  ...      mean
Cluster                                                     ...
0                   0.170994  1.119014  0.017037  1.011555  ... -0.344690
1                  -0.360989  0.522372  0.089717  0.990529  ...  0.318859
2                   0.351810  1.198801 -0.118156  0.979308  ... -0.046501

                Internet Type_2           Contract_1           Contract_2  \
              std            mean       std       mean       std       mean
Cluster
0        0.213355        0.297744  0.997491   0.242192  1.126656   0.400849
1        1.293127        0.316395  0.994295  -0.243267  0.782545  -0.471426
2        0.941130       -0.851535  0.193276   0.077802  1.048744   0.213306

                Payment Method_1
              std            mean       std
Cluster
0        1.123148       -0.030657  0.992897
1        0.531300       -0.276654  0.894107
2        1.086088        0.444115  1.001593
```

7. Code

Creating the Dataset:

```
# importing the dataset from kaggle
import pandas as pd
df = pd.read_csv('telco.csv')
df.rename(columns={"Zip Code": "zip"}, inplace=True)

# importing the dataset for the zipcodes
zip = pd.read_csv('/content/zip_code_database.csv')
cal = zip[zip['state'] == 'CA'][['county', 'zip']]

# merging the zipcodes into the kaggle dataset
merged_data1 = pd.merge(df, cal, on="zip", how="left")

# importing the income dataset
econ = pd.read_excel('/content/income202447.xlsx')
econ.rename(columns={"California LaborMarketInfo": "county"}, inplace=True)
econ.to_csv('.income202447', index=False)
econ = econ[econ['Unnamed: 3'] == 'Per Capita Personal Income - BEA'][['county', 'Unnamed: 4']]

# merging the income dataset into the merged dataset that we did previously
merged_data1['county'] = merged_data1['county'].str.strip().str.lower()
econ['county'] = econ['county'].str.strip().str.lower()
final_dataset = pd.merge(merged_data1, econ, on="county", how="left")
```

```
final_dataset['Unnamed: 4'] = final_dataset['Unnamed: 4'].astype(int)
final_dataset.rename(columns={"Unnamed: 4": "County Income Per Capita"}, inplace=True)

# adding variable of services subscribed
columns_to_convert = ['Phone Service', 'Internet Service', 'Online Security', 'Online Backup', 'Device Protection Plan', 'Pre
mapping = {'Yes': 1, 'No': 0, 'True': 1, 'False': 0}
add_ons = final_dataset[columns_to_convert].replace(mapping)
final_dataset['Add Ons'] = add_ons.sum(axis=1)

# adding variable to indicate if lower income or not
import numpy as np
threshold = np.percentile(final_dataset['County Income Per Capita'].unique(), 25)
final_dataset['Low Income'] = (final_dataset['County Income Per Capita'] < threshold).astype(int).astype('object')
# using my county variable to transform it into a new variable for regions so I reduce the categories for the OHE
county_to_region = {
    "northern california": [
        "trinity county", "lassen county", "kings county", "del norte county", "siskiyou county",
        "shasta county", "humboldt county", "tehama county", "modoc county", "plumas county",
        "sierra county", "mendocino county", "glenn county", "butte county", "colusa county",
          "sutter county", "yuba county", "lake county", "yolo county", "sacramento county", "placer county", "nevada county",
    ],
    "central california": [
        "merced county", "tulare county", "madera county", "kern county", "fresno county",
        "stanislaus county", "san joaquin county", "kings county", "calaveras county",
        "tuolumne county", "mariposa county", "san benito county"
    ],
    "bay area": [
        "marin county", "alameda county", "san mateo county", "contra costa county",
        "san francisco county", "santa clara county", "napa county", "sonoma county", "solano county"
    ],
    "southern california": [
        "los angeles county", "orange county", "san diego county", "ventura county",
        "riverside county", "san bernardino county", "imperial county"
    ],
    "coastal region": [
        "santa cruz county", "santa barbara county", "san luis obispo county", "monterey county"
    ],
    "eastern region": [
        "mono county", "inyo county", "alpine county"
    ]
}

county_to_region_flat = {}
for region, counties in county_to_region.items():
    for county in counties:
        county_to_region_flat[county] = region


final_dataset["Region"] = final_dataset["county"].map(county_to_region_flat)

# target variable to the end of the DataFrame
churn_label = 'Churn Label'
columns = [col for col in final_dataset.columns if col != churn_label]
columns.append(churn_label)
```

```
# Reorder the DataFrame
final_dataset = final_dataset[columns]
```

Dropping Variables:

```
columns_to_drop = [
    "Customer ID", "Dependents", "Country", "State", "City", "Quarter",
    "Customer Status", "Churn Category", "Churn Reason", "county","Churn Score", "zip"
]

final_dataset = final_dataset.drop(columns=columns_to_drop, axis=1)
```

Missing Values:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(final_dataset.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data Heatmap')
plt.show()

final_dataset.isnull().sum(axis=0)
```

Imputation of NA

```
final_dataset["Internet Type"].fillna("No Internet", inplace=True) # Those with no internet service
final_dataset["Offer"].fillna("No marketing offer received", inplace=True) # Those with no marketing offer
```

Main Statistics of Pertinent Numerical Variables

```
final_dataset[['County Income Per Capita', 'Add Ons']].describe()
```

Histograms

```
import matplotlib.pyplot as plt
import seaborn as sns
categorical_columns = [
    "Gender",
    "Contract",
    "Payment Method",
    "Satisfaction Score",
    "Churn Label"
]

# Create a single figure with multiple subplots
fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(15, 5))
axes = axes.flatten()  # Flatten the axes array to make it easier to loop through

for i, column in enumerate(categorical_columns):
    ax = axes[i]  # Get the current subplot axis
    value_counts = final_dataset[column].value_counts()
    colors = sns.color_palette("pastel", len(value_counts))
    value_counts.plot(kind='bar', color=colors, ax=ax)
    ax.set_title(f"{column}")
    ax.set_ylabel("Customers")
    ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
    legend_labels = value_counts.index.tolist()
```

```
    handles = [plt.Rectangle((0, 0), 1, 1, color=colors[i]) for i in range(len(legend_labels))]
    ax.legend(handles, legend_labels, loc="upper right")
    for ax in axes[len(categorical_columns):]:  # Skip axes corresponding to columns
        ax.set_visible(False)
    plt.tight_layout()
    plt.show()

Outliers
numerical_variables = [
    "Age",
    "Number of Dependents",
    "Population",
    "Number of Referrals",
    "Tenure in Months",
    "Avg Monthly GB Download",
    "Total Extra Data Charges",
    "CLTV",
    "Add Ons",
    "County Income Per Capita"
]
melted_df = final_dataset[numerical_variables].melt(var_name="Variable", value_name="Value")
g = sns.FacetGrid(melted_df, col="Variable", col_wrap=5, sharex=False, sharey=False, height=4)
g.map(sns.boxplot, "Value", color="skyblue")
g.set_titles("{col_name}")
g.set_axis_labels("Value", "")
g.tight_layout()
plt.show()

import pandas as pd
numerical_variables = [
    "Age",
    "Number of Dependents",
    "Population",
    "Number of Referrals",
    "Tenure in Months",
    "Avg Monthly GB Download",
    "Total Extra Data Charges",
    "CLTV",
    "Add Ons"
]
def count_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return len(outliers)
outliers_count = {}
for column in numerical_variables:
    outliers_count[column] = count_outliers(final_dataset, column)
print(outliers_count)

Correlation Matrix Between Numerical Variables
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt
numerical_variables = [
    "Age",
    "Number of Dependents",
    "Population",
    "Number of Referrals",
    "Tenure in Months",
    "Avg Monthly GB Download",
    "Total Extra Data Charges",
    "CLTV",
    "Add Ons"
]
correlation_matrix = final_dataset[numerical_variables].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(
    correlation_matrix,
    annot=True,
    fmt=".2f",
    cmap="coolwarm",
    linewidths=0.5,
    cbar=True
)
plt.show()

Categorical Variables Encoding
final_dataset.dtypes

from sklearn.preprocessing import LabelEncoder
categorical_columns = [
    "Gender",
    "Under 30",
    "Senior Citizen",
    "Married",
    "Referred a Friend",
    "Offer",
    "Phone Service",
    "Multiple Lines",
    "Internet Service",
    "Internet Type",
    "Online Security",
    "Online Backup",
    "Device Protection Plan",
    "Premium Tech Support",
    "Streaming TV",
    "Streaming Movies",
    "Streaming Music",
    "Unlimited Data",
    "Contract",
    "Paperless Billing",
    "Payment Method",
    "Low Income",
    "Region",
    "Satisfaction Score",
    "Churn Label"
]
label_encoder = LabelEncoder()
```

```
for col in categorical_columns:
    if col in final_dataset.columns:
        final_dataset[col] = label_encoder.fit_transform(final_dataset[col])
categorical_columns = ["Offer", "Internet Type", "Contract", "Payment Method"]
final_dataset = pd.get_dummies(final_dataset, columns=categorical_columns, drop_first=True,dtype= int)
```

Feature Selection
Filter Methods (Numerical Variables):
1. Variance

```
from sklearn.feature_selection import VarianceThreshold
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
numerical_variables = [
    "Age",
    "Number of Dependents",
    "Population",
    "Number of Referrals",
    "Tenure in Months",
    "Avg Monthly GB Download",
    "Total Extra Data Charges",
    "CLTV",
    "Add Ons"
]
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(final_dataset[numerical_variables])
variance = pd.Series(scaled_data.var(axis=0), index=numerical_variables)
print("Variance of features:\n", variance)
threshold = 0.01
selector = VarianceThreshold(threshold=threshold)
selector.fit(scaled_data)
selected_features =list(pd.DataFrame(scaled_data,
columns=numerical_variables).columns[selector.get_support(indices=True)])
print("\nFeatures selected with variance above threshold ({}):\n{}".format(threshold, selected_features))

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.barplot(x=variance.index, y=variance.values, palette="coolwarm")
plt.title("Variance of Numerical Features", fontsize=16)
plt.xlabel("Features", fontsize=12)
plt.ylabel("Variance", fontsize=12)
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```

2. Correlation Between Features

```
import matplotlib.pyplot as plt
import seaborn as sns
aux2 = final_dataset.loc[:, final_dataset.columns != 'Churn Label']
mid_index = len(aux2.columns) // 2
group1 = aux2.iloc[:, :mid_index]
group2 = aux2.iloc[:, mid_index:]
corr_matrix_group1 = group1.corr()
corr_matrix_group2 = group2.corr()
plt.figure(figsize=(18, 16))
plt.subplot(2, 1, 1)  # (rows, columns, index)
```

```
sns.heatmap(corr_matrix_group1, annot=True, cmap='coolwarm', fmt='.2f', vmin=-1, vmax=1, center=0, cbar_kws={'label': 'Corre
plt.title('Correlation Matrix - Group 1')
plt.subplot(2, 1, 2)  # (rows, columns, index)
sns.heatmap(corr_matrix_group2, annot=True, cmap='coolwarm', fmt='.2f', vmin=-1, vmax=1, center=0, cbar_kws={'label': 'Correlation})
plt.title('Correlation Matrix - Group 2')
plt.tight_layout()
plt.show()
```

3. Correlation With Target

```
# Calculate correlations with the target (feature-class correlation)
target_corr = final_dataset.corr()['Churn Label'].drop('Churn Label')
target_corr_sorted = target_corr.sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=target_corr_sorted.values, y=target_corr_sorted.index, palette='coolwarm')
plt.title('Feature-Target Correlations')
plt.xlabel('Correlation Coefficient')
plt.ylabel('Features')
plt.show()
```

4. Mutual Information

```
from sklearn.feature_selection import mutual_info_classif
X = final_dataset.loc[:, final_dataset.columns != 'Churn Label']
y = final_dataset['Churn Label']
mutual_info = mutual_info_classif(X, y)
mi_scores = pd.DataFrame({
   'Feature': X.columns,
   'Mutual Information': mutual_info
}).sort_values(by='Mutual Information', ascending=False)
print(mi_scores)
```

EXTRA: Metric M

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
X = final_dataset.loc[:, final_dataset.columns != 'Churn Label']
y = final_dataset['Churn Label']
def calculate_m_k(target_corr, feature_corr, selected_features, candidate_feature):
   k = len(selected_features) + 1
   corr_cf = target_corr[candidate_feature]
   if not selected_features:
      corr_ff = 0  # No feature-feature correlation for the first feature
   else:
      corr_ff = feature_corr.loc[selected_features, candidate_feature].mean()
   denominator = np.sqrt(k + k * (k - 1) * max(corr_ff, 0))  # this is to avoid negatives
   if denominator == 0:  # Handle divide-by-zero scenario
      return 0
   m_k = (k * corr_cf) / denominator
   return m_k
def calculate_all_mk_scores(data, target):
   # calculate feature-target and feature-feature correlations
   target_corr = data.corrwith(target)  # Feature-target correlation
   feature_corr = data.corr()          # Feature-feature correlation matrix
   mk_scores = {}
   for feature in data.columns:
```

```
        mk_scores[feature] = calculate_m_k(target_corr, feature_corr, [], feature)
    return pd.Series(mk_scores)
# calculate M_k scores for each feature
mk_scores = calculate_all_mk_scores(X, y)
plt.figure(figsize=(10, 6))
mk_scores.sort_values(ascending=False).plot(kind='bar', color='skyblue')
plt.title("M_k Scores for Each Feature")
plt.xlabel("Features")
plt.ylabel("M_k Score")
plt.xticks(rotation=90)
plt.show()

# For this we are using the old dataframe (final_dataset before encoding) again - created a copy earlier
# Compute value counts and percentages for each column
columns_to_check = ['Gender', 'Contract', 'Payment Method', 'Satisfaction Score', 'Low Income','Churn Label']
summary = {}
for column in columns_to_check:
    # Convert column to string type to handle mixed types
    df_copy[column] = df_copy[column].astype(str)
    counts = df_copy[column].value_counts(normalize=True) * 100
    summary[column] = counts.to_dict()  # Convert to dictionary for easier handling in plain Python
# Create a summary table for percentages
percentages_table = []
for column, value_counts in summary.items():
    for value, percentage in value_counts.items():
        percentages_table.append({
            'Column': column,
            'Value': value,
            'Percentage': round(percentage, 2)
        })
# Display the percentages
print("Proportion Analysis of Variables:")
for entry in percentages_table:
    print(f"Column: {entry['Column']}, Value: {entry['Value']}, Percentage: {entry['Percentage']}%")

import matplotlib.pyplot as plt
# prepare data for the bar graph
columns = [entry['Column'] for entry in percentages_table]
values = [entry['Value'] for entry in percentages_table]
percentages = [entry['Percentage'] for entry in percentages_table]
unique_columns = list(set(columns))
color_map = {col: f"C{i}" for i, col in enumerate(unique_columns)}
colors = [color_map[col] for col in columns]
# Plot the bar graph
plt.figure(figsize=(12, 6))
bars = plt.bar(range(len(percentages_table)), percentages, color=colors)
plt.xlabel('Variables with Their Corresponding Class Sizes')
plt.ylabel('Percentage (%)')
plt.title('Imbalance Analysis of Variables')
plt.ylim(0, 100)  # Set y-axis range to 0-100
plt.yticks(range(0, 101, 5))  # Ticks every 5
# legend for the variables
legend_labels = [plt.Line2D([0], [0], color=color_map[col], lw=4) for col in unique_columns]
plt.legend(legend_labels, unique_columns, title="Variables", loc='upper left', bbox_to_anchor=(1.05, 1))
plt.tight_layout(rect=[0, 0, 0.85, 1])  # Leave space on the right for the legend
plt.show()
```

Train/Validation/Test Split

```
from sklearn.model_selection import train_test_split

X = final_dataset.loc[:, final_dataset.columns != 'Churn Label']
y = final_dataset['Churn Label']
## SCALE DATASET
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# First, split into 50% training and 50% temporary (validation + test)
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.50, random_state=1234, shuffle=True, stratify=y
)
# Then, split the temporary set (50%) into 25% validation and 25% test
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.50, random_state=1234, shuffle=True, stratify=y_temp
)
print(f"Training set: {len(X_train)} rows")
print(f"Validation set: {len(X_val)} rows")
print(f"Test set: {len(X_test)} rows")
```

Random Forest

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(random_state=1234)
rfc.fit(X_train, y_train)
```

In [ ]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, classification_report

import seaborn as sns
import matplotlib.pyplot as plt

def evaluate_model(model, X_train, y_train, X_test, y_test):
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)
    y_test_prob = model.predict_proba(X_test)[:, 1] if hasattr(model, "predict_proba") else None


    accuracy = accuracy_score(y_test, y_test_pred)
    precision = precision_score(y_test, y_test_pred)
    recall = recall_score(y_test, y_test_pred)
    f1 = f1_score(y_test, y_test_pred)
    auc = roc_auc_score(y_test, y_test_prob) if y_test_prob is not None else None

    print("Model Evaluation Metrics (Test Set):")
    print(f"Accuracy:  {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall:    {recall:.4f}")
    print(f"F1-Score:  {f1:.4f}")
```

```
    print("\nClassification Report :")
    print(classification_report(y_test, y_test_pred))


    cm = confusion_matrix(y_test, y_test_pred)
    plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["Not Churn", "Churn"], yticklabels=["Not
Churn", "Churn"])
    plt.title("Confusion Matrix ")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

In [ ]:

```
evaluate_model(rfc, X_train, y_train, X_train, y_train)
```

Random Forest: Hyper Parameter Tuning

In [ ]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score, make_scorer



recall_scorer = make_scorer(recall_score, pos_label=1) # to focus on correctly predicting Churners

param_grid = {
    'n_estimators': [100, 150, 200],
    'criterion': ['gini', 'entropy'],
    'max_depth': [4, 5, 6, 7, 10],
    'max_leaf_nodes': [50, 100],
    'min_samples_split': [10, 20, 25]
}


rf_model = RandomForestClassifier(random_state=42)


grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    cv=5,
    scoring=recall_scorer,
)

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

print("Best Hyperparameters:", best_params)
```

In [ ]:

```
Optimised_rfc = RandomForestClassifier(criterion= 'gini', max_depth= 10, max_leaf_nodes= 100, min_samples_split= 10, n_estimators= 200,random_state=1234)
Optimised_rfc.fit(X_train, y_train)
```

Random Forest: Selecting Features

In [ ]:

```
importance = Optimised_rfc.feature_importances_
names = X.columns
df_imp = pd.DataFrame({"Feature": names, "Importance": importance}).sort_values(by="Importance", ascending=False)
print(df_imp)
```

In [ ]:

```
import matplotlib.pyplot as plt

# Plot feature importances
plt.figure(figsize=(10, 6))
plt.barh(df_imp["Feature"], df_imp["Importance"], color="skyblue")
plt.gca().invert_yaxis()  # Invert the y-axis to show the most important features on top
plt.xlabel("Feature Importance")
plt.title("Feature Importances in Random Forest")
plt.show()
```

In [ ]:

```
importance = rfc.feature_importances_
names = X.columns

# Create a DataFrame for feature importances
df_imp = pd.DataFrame({"Feature": names, "Importance": importance}).sort_values(by="Importance", ascending=False)


thresholds = [0,0.0025,0.004, 0.005,0.0075, 0.01, 0.02]


subsets = {}

for threshold in thresholds:
    selected_features = df_imp[df_imp["Importance"] >= threshold]["Feature"].tolist()
    ignored_features = df_imp[df_imp["Importance"] < threshold]["Feature"].tolist()

    selected_indices = [X.columns.get_loc(col) for col in selected_features]
    subsets[threshold] = {
        "selected_features": selected_features,
        "ignored_features": ignored_features,
        "X_train": X_train[:, selected_indices],
        "X_test": X_test[:, selected_indices]
    }

    # Print the subset information
    print(f"\nThreshold: {threshold}")
    print(f"Selected Features (Importance >= {threshold}): {len(selected_features)}")
    print(f"Ignored Features (Importance < {threshold}): {len(ignored_features)}")
```

In [ ]:

```
from sklearn.metrics import recall_score


results = {}
for threshold, subset in subsets.items():

    X_train_subset = subset["X_train"]
    X_test_subset = subset["X_test"]


    Optimised_rfc.fit(X_train_subset, y_train)
    y_val_pred = Optimised_rfc.predict(X_test_subset)

    accuracy = accuracy_score(y_test, y_val_pred)
    recall = recall_score(y_test, y_val_pred, pos_label=1)
    print(f"Validation Accuracy for threshold {threshold}: {accuracy:.4f}")
    print(f"Validation Recall (Class 1) for threshold {threshold}: {recall:.4f}")
    print()
    report = classification_report(y_test, y_val_pred)
    results[threshold] = {
        "accuracy": accuracy,
        "recall_class_1": recall,
        "report": report
    }
```

Random Forest: Final Model with Hyperparameter tuning and feature selection

In [ ]:

```
from sklearn.metrics import accuracy_score, recall_score, classification_report

final_threshold = 0.004
final_subset = subsets[final_threshold]


X_train_final = final_subset["X_train"]
X_test_final = final_subset["X_test"]


final_rfc = RandomForestClassifier(
    criterion='gini',
    max_depth=10,
    max_leaf_nodes=100,
    min_samples_split=10,
    n_estimators=200,
    random_state=1234
)


final_rfc.fit(X_train_final, y_train)
y_test_pred = final_rfc.predict(X_test_final)

evaluate_model(final_rfc, X_train_final, y_train, X_test_final, y_test)
```

2. Logistic Regression

In [28]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SequentialFeatureSelector
```

In [ ]:

```
## Study Linear Separability of classes
X_train_df = pd.DataFrame(X_train, columns=X.columns)
exclude_columns = set(binary_columns + categorical_columns)
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
numeric_columns = [
   col for col in X_train_df.select_dtypes(include=numerics).columns
   if col not in exclude_columns
]
y_train = y_train.reset_index(drop=True)

# Loop through pairs of numeric columns and plot scatter plots
for index in range(len(numeric_columns) - 1):
   for index2 in range(index + 1, len(numeric_columns)):
      col1 = numeric_columns[index]
      col2 = numeric_columns[index2]

      # Plot for class 0
      plt.scatter(
         X_train_df.loc[y_train == 0, col1],
         X_train_df.loc[y_train == 0, col2],
         marker='o',
         label='Class 0'
      )

      # Plot for class 1
      plt.scatter(
         X_train_df.loc[y_train == 1, col1],
         X_train_df.loc[y_train == 1, col2],
         marker='^',
         label='Class 1'
      )

      # Adding labels and legend
      plt.xlabel(col1)
      plt.ylabel(col2)
      plt.legend(loc='upper left')
      plt.title(f'Scatter Plot: {col1} vs {col2}')
      plt.show()
```

2.1 Forward Feature Selection

In [ ]:

```
# Initialize the logistic regression model
log_reg_forwards = LogisticRegression(max_iter=1000)

# Initialize the Sequential Feature Selector with backward selection
sfs_forwards = SequentialFeatureSelector(log_reg_forwards, n_features_to_select='auto', direction='forward', scoring='recall', cv = 5)
```

```
# By setting cv = 5 the feature selector will use 5-fold cross-validation to evaluate the performance of different subsets of features at each step.

# Fit the selector on the training data
sfs_forwards.fit(X_train, y_train)

# Get the selected features
selected_features_forwards = X.columns[sfs_forwards.get_support()].tolist()

# Display selected features
print(f"Selected features: {selected_features_forwards}")
```

In [ ]:

```
#Train a logistic regression model using only the selected features
X_train_selected_forwards = X_train[:, sfs_forwards.get_support()]
X_test_selected_forwards = X_test[:, sfs_forwards.get_support()]

log_reg_forwards.fit(X_train_selected_forwards, y_train)
```

In [ ]:

```
# Get predicted probabilities for the positive class (class 1)
y_train_pred_proba_f = log_reg_forwards.predict_proba(X_train_selected_forwards)[:, 1]
y_test_pred_proba_f = log_reg_forwards.predict_proba(X_test_selected_forwards)[:, 1]

# Set the threshold for classification
threshold = 0.5

# Convert the predicted probabilities to class labels using the threshold
y_train_pred_f = (y_train_pred_proba_f >= threshold).astype(int)
y_test_pred_f = (y_test_pred_proba_f >= threshold).astype(int)
```

In [ ]:

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
def Metrics(y_true, y_pred, data='train'):
    print(f'The metrics for the {data} dataset are:')
    print('Precision: %.3f' % precision_score(y_true, y_pred))
    print('Recall: %.3f' % recall_score(y_true, y_pred))
    print('Accuracy: %.3f' % accuracy_score(y_true, y_pred))
    print('F1 Score: %.3f' % f1_score(y_true, y_pred))
    print()
    print()
    print()
```

In [ ]:

```
# Calculate and display metrics for the training set
Metrics(y_train, y_train_pred_f, data='train')
```

In [ ]:

```
# Calculate and display metrics for the testing set
Metrics(y_test, y_test_pred_f, data='test')
```

In [ ]:

```
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

# Assuming y_test_pred_f is the predictions for the test set from the forward model
```

```
# Print the classification report
print("Classification Report:")
print(classification_report(y_test, y_test_pred_f))

# Generate and display the confusion matrix
conf_matrix = confusion_matrix(y_true=y_test, y_pred=y_test_pred_f)
print("Confusion Matrix:")
print(conf_matrix)

# Display the confusion matrix visually
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=['Class 0', 'Class 1'])
disp.plot(cmap='Blues')
```

2.2 Logistic Regression: Cross Validation

In [ ]:

```
from sklearn.model_selection import cross_validate
scoring_metrics = ['accuracy', 'precision', 'recall', 'f1']
cv_scores_forwards = cross_validate(log_reg_forwards, X_train_selected_forwards, y_train,
                                    cv=5, scoring=scoring_metrics)
```

In [ ]:

```
print("Cross-validation results for forward logistic regression:")
print(f"Accuracy: {cv_scores_forwards['test_accuracy']}")
print(f"Precision: {cv_scores_forwards['test_precision']}")
print(f"Recall: {cv_scores_forwards['test_recall']}")
print(f"F1-Score: {cv_scores_forwards['test_f1']}")

# Print the average score for each metric for forwards logistic regression
print(f"Average Accuracy: {cv_scores_forwards['test_accuracy'].mean()}")
print(f"Average Precision: {cv_scores_forwards['test_precision'].mean()}")
print(f"Average Recall: {cv_scores_forwards['test_recall'].mean()}")
print(f"Average F1-Score: {cv_scores_forwards['test_f1'].mean()}")
```

In [ ]:

```
# Cross-validation results
folds = [f"Fold {i+1}" for i in range(len(cv_scores_forwards['test_accuracy']))]
accuracy_metric = cv_scores_forwards['test_accuracy']
precision_metric = cv_scores_forwards['test_precision']
recall_metric= cv_scores_forwards['test_recall']
f1_score_metric = cv_scores_forwards['test_f1']

# Plot metrics
plt.figure(figsize=(10, 6))
plt.plot(folds, accuracy_metric, marker='o', label='Accuracy')
plt.plot(folds, precision_metric, marker='o', label='Precision')
plt.plot(folds, recall_metric, marker='o', label='Recall')
plt.plot(folds, f1_score_metric, marker='o', label='F1-Score')

# Add labels, legend, and title
plt.title("Cross-Validation Metrics Across Folds (Forward Logistic Regression)")
plt.xlabel("Fold")
plt.ylabel("Metric Value")
plt.legend()
plt.grid(True)
plt.show()
```

2.3 Logistic Regression Tuning: Find the best threshold

In [ ]:

```
from sklearn import metrics
# Get the predicted probabilities for the positive class
y_pred_proba_f = log_reg_forwards.predict_proba(X_train_selected_forwards)[:, 1]

# Compute the False Positive Rate (FPR), True Positive Rate (TPR), and thresholds
fpr, tpr, thresholds = metrics.roc_curve(y_train, y_pred_proba_f)

# Calculate the AUC (Area Under the Curve)
auc = metrics.roc_auc_score(y_train, y_pred_proba_f)

# Calculate G-Mean (Geometric Mean) for optimization
gmeans = np.sqrt(tpr * (1 - fpr))

# Find the best threshold that maximizes the G-Mean
ix = np.argmax(gmeans)
print(f'Best Threshold={thresholds[ix]:.6f}, G-Mean={gmeans[ix]:.3f}')

# Plot the ROC curve
plt.plot(fpr, tpr, label="AUC=" + str(auc.round(5)))

# Plot a random classifier for comparison
plt.plot([0, 1], [0, 1], linestyle='--', label='Random')

# Mark the best threshold on the plot
plt.scatter(fpr[ix], tpr[ix], marker='o', color='black', label='Best')

# Set plot labels and legend
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)

# Show the plot
plt.show()
```

In [ ]:

```
# get best threshold
# 0.3634
```

In [ ]:

```
# Get predicted probabilities for the positive class (class 1)
y_train_pred_proba_f = log_reg_forwards.predict_proba(X_train_selected_forwards)[:, 1]
y_test_pred_proba_f = log_reg_forwards.predict_proba(X_test_selected_forwards)[:, 1]

# Set the threshold for classification
threshold = 0.3634

# Convert the predicted probabilities to class labels using the threshold
y_train_pred_f_t = (y_train_pred_proba_f >= threshold).astype(int)
y_test_pred_f_t = (y_test_pred_proba_f >= threshold).astype(int)
```

In [ ]:

```
# Calculate and display metrics for the training set
```

```
Metrics(y_train, y_train_pred_f_t, data='train')
```

In [ ]:

```
# Calculate and display metrics for the training set
Metrics(y_test, y_test_pred_f_t, data='test')
```

With this threshold seems to prioritise recall. In our interest for our model

2.4 Get Parameters of Logistic Regression to Study Hypothesis

In [ ]:

```
# Extract coefficients and intercept
selected_features_forwards
coefficients = log_reg_forwards.coef_  # Coefficients for each feature
intercept = log_reg_forwards.intercept_  # Intercept term


print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

In [ ]:

```
# Extract coefficients
coefficients = log_reg_forwards.coef_[0] # For binary classification, take the first (and only) row
odds_ratios = np.exp(coefficients)

# Create a DataFrame to pair feature names with coefficients
coefficients_df = pd.DataFrame({
   'Feature': selected_features_forwards,  # Use the list of selected features
   'Coefficient': coefficients,
   'Odds Ratio': odds_ratios
})


# Add the intercept as a separate row
coefficients_df = pd.concat([
   coefficients_df,
            pd.DataFrame({'Feature': ['Intercept'], 'Coefficient': log_reg_forwards.intercept_, 'Odds Ratio':
np.exp(log_reg_forwards.intercept_)})
], ignore_index=True)

coefficients_df
```

PCA and Clustering

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score, precision_score, recall_score, f1_score, accuracy_score
from sklearn.linear_model import LogisticRegression

# Assuming you have the following variables defined:
```

```
# - selected_features_forwards: List of selected feature names
# - X_train_selected_forwards: Training features (DataFrame or NumPy array)
# - X_test_selected_forwards: Testing features (DataFrame or NumPy array)
# - y_train: Training target
# - y_test: Testing target

# 1. Convert NumPy arrays to DataFrames if necessary
if isinstance(X_train_selected_forwards, np.ndarray):
  X_train_selected_forwards = pd.DataFrame(X_train_selected_forwards, columns=selected_features_forwards)
  print("Converted X_train_selected_forwards to DataFrame.")

if isinstance(X_test_selected_forwards, np.ndarray):
  X_test_selected_forwards = pd.DataFrame(X_test_selected_forwards, columns=selected_features_forwards)
  print("Converted X_test_selected_forwards to DataFrame.")

# 2. Verify the conversion and column consistency
print(f"Columns in X_train_selected_forwards: {X_train_selected_forwards.columns.tolist()}")
print(f"Columns in X_test_selected_forwards: {X_test_selected_forwards.columns.tolist()}")

# 3. Concatenate training and testing data for clustering
X_combined = pd.concat([X_train_selected_forwards, X_test_selected_forwards], axis=0).reset_index(drop=True)
print(f"Combined dataset shape: {X_combined.shape}")

# 4. Apply PCA for Dimensionality Reduction
pca = PCA(n_components=5, random_state=42)  # Adjust n_components as needed
X_pca = pca.fit_transform(X_combined)

# 5. Determine Optimal Number of Clusters using Elbow Method and Silhouette Score
wcss = []
sil_scores = []
K_range = range(2, 11)

for k in K_range:
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(X_pca)
  wcss.append(kmeans.inertia_)

  sil = silhouette_score(X_pca, kmeans.labels_)
  sil_scores.append(sil)

# Plot Elbow Method
plt.figure(figsize=(8,5))
plt.plot(K_range, wcss, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()

# Plot Silhouette Scores
plt.figure(figsize=(8,5))
plt.plot(K_range, sil_scores, marker='o', color='green')
plt.title('Silhouette Scores for Different K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Silhouette Score')
plt.grid(True)
```

```
plt.show()
```

Creating clusters

In [ ]:

```
# Choose optimal K based on plots (e.g., K=4)
optimal_k = 3
kmeans_optimal = KMeans(n_clusters=optimal_k, random_state=42)
cluster_labels_optimal = kmeans_optimal.fit_predict(X_pca)

# Assign cluster labels to combined data
X_combined['Cluster'] = cluster_labels_optimal

# Split cluster labels back to train and test sets
n_train = X_train_selected_forwards.shape[0]

X_train_clustered = X_train_selected_forwards.copy()
X_train_clustered['Cluster'] = cluster_labels_optimal[:n_train]

X_test_clustered = X_test_selected_forwards.copy()
X_test_clustered['Cluster'] = cluster_labels_optimal[n_train:]

print(f"Training set with clusters: {X_train_clustered.shape}")
print(f"Testing set with clusters: {X_test_clustered.shape}")

# 6. Define Function to Evaluate Model Performance per Cluster
def evaluate_model_per_cluster(model, X, y, cluster_column='Cluster', threshold=0.363435):
  clusters = X[cluster_column].unique()
  metrics = []

  for cluster in clusters:
      mask = X[cluster_column] == cluster
      X_cluster = X[mask].drop(columns=[cluster_column])
      y_cluster = y[mask]

      if len(X_cluster) == 0:
          continue  # Skip empty clusters

      y_proba = model.predict_proba(X_cluster)[:, 1]
      y_pred = (y_proba >= threshold).astype(int)

      precision = precision_score(y_cluster, y_pred, zero_division=0)
      recall = recall_score(y_cluster, y_pred, zero_division=0)
      accuracy = accuracy_score(y_cluster, y_pred)
      f1 = f1_score(y_cluster, y_pred, zero_division=0)

      metrics.append({
          'Cluster': cluster,
          'Precision': precision,
          'Recall': recall,
          'Accuracy': accuracy,
          'F1_Score': f1
      })

  metrics_df = pd.DataFrame(metrics).sort_values(by='Cluster')
  return metrics_df
```

Testing the clusters

In [ ]:

```
n_train = X_train_selected_forwards.shape[0]

# Ensure that the index is reset to match cluster labels
X_train_clustered = X_train_selected_forwards.copy().reset_index(drop=True)
X_train_clustered['Cluster'] = cluster_labels_optimal[:n_train]

X_test_clustered = X_test_selected_forwards.copy().reset_index(drop=True)
X_test_clustered['Cluster'] = cluster_labels_optimal[n_train:]

y_train = y_train.reset_index(drop=True)
y_test = y_test.reset_index(drop=True)

# Reset indices for all relevant DataFrames and Series
X_train_clustered = X_train_clustered.reset_index(drop=True)
X_test_clustered = X_test_clustered.reset_index(drop=True)
y_train = y_train.reset_index(drop=True)
y_test = y_test.reset_index(drop=True)


# Evaluate performance per cluster on the training set
train_metrics_df = evaluate_model_per_cluster(log_reg_forwards, X_train_clustered, y_train)

# Evaluate performance per cluster on the testing set
test_metrics_df = evaluate_model_per_cluster(log_reg_forwards, X_test_clustered, y_test)

# Display metrics
print("Training Set Metrics per Cluster:")
print(train_metrics_df)

print("\nTesting Set Metrics per Cluster:")
print(test_metrics_df)
```

Visualizing features accross clusters

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# List of features to visualize across clusters
features_to_visualize = [
    'Gender', 'Senior Citizen', 'Married', 'Number of Dependents',
    'Longitude', 'Referred a Friend', 'Number of Referrals',
    'Tenure in Months', 'Avg Monthly Long Distance Charges',
    'Online Security', 'Online Backup', 'Device Protection Plan',
    'Unlimited Data', 'Monthly Charge', 'Total Extra Data Charges',
    'Total Long Distance Charges', 'Satisfaction Score', 'Region',
    'Offer_1', 'Offer_2', 'Offer_5', 'Internet Type_2',
    'Contract_1', 'Contract_2', 'Payment Method_1'
]

# Loop through each feature and plot a boxplot by cluster
for feature in features_to_visualize:
    if feature in X_combined.columns:
```

```
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Cluster', y=feature, data=X_combined)
    plt.title(f'{feature} Distribution across Clusters')
    plt.xlabel('Cluster')
    plt.ylabel(feature)
    plt.grid(True)
    plt.show()
else:
    print(f"Feature '{feature}' not found in X_combined columns.")
```

Inspecting cluster centroids

In [ ]:

```
import pandas as pd

# Get cluster centroids
centroids = kmeans_optimal.cluster_centers_

# Create a DataFrame with centroids for better readability
centroids_df = pd.DataFrame(centroids, columns=[f'PC{i+1}' for i in range(centroids.shape[1])])
print("Centroids of each cluster in PCA space:\n")
print(centroids_df)
```

Calculate mean and std for each feature per cluster

In [ ]:

```
# Calculate mean and std for each feature per cluster
cluster_summary = X_combined.groupby('Cluster').agg(['mean', 'std'])

print("\nCluster Summary (Mean and Standard Deviation per Feature):\n")
print(cluster_summary)
```

Hypothesis 1

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression

# Prepare X and y
X = final_dataset[['Contract_1', 'Contract_2']]
y = final_dataset['Churn Label']

# Fit Logistic Regression
log_reg = LogisticRegression(max_iter=1000, random_state=1234)
log_reg.fit(X, y)

# Extract coefficients
coefficients = log_reg.coef_[0]
intercept = log_reg.intercept_[0]

# Create a DataFrame for coefficients
contract_coefficients = pd.DataFrame({
    'Feature': ['Contract_1 (One Year)', 'Contract_2 (Two Year)'],
    'Coefficient': coefficients
})
contract_coefficients['Odds Ratio'] = np.exp(contract_coefficients['Coefficient'])
```

```
# Display results
print(contract_coefficients)
print(f"Intercept: {intercept}")
```

Hypothesis 3

In [ ]:

```
# Extract the first element of the tuple (list of feature names)
feature_names = selected_features[0]

# Find the coefficient for 'Payment Method_2'
payment_method_coeff = log_reg_forwards.coef_[0][feature_names.index('Payment Method_2')]
odds_ratio = np.exp(payment_method_coeff)

# Display the coefficient and odds ratio
print(f"Payment Method_2 Coefficient: {payment_method_coeff:.3f}")
print(f"Odds Ratio: {odds_ratio:.3f}")
```

Hypothesis 2 - Dual Relationship with Gender and income

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Create interaction term
final_dataset['Gender_Binary'] = final_dataset['Gender']  # Assuming Gender is already binary encoded
final_dataset['Low_Income_Binary'] = final_dataset['Low Income'].astype(int)  # Convert to numeric if needed
final_dataset['Gender_LowIncome_Interaction'] = final_dataset['Gender_Binary'] *
final_dataset['Low_Income_Binary']

# Select features for the model
X = final_dataset[['Gender_Binary', 'Low_Income_Binary', 'Gender_LowIncome_Interaction']]
y = final_dataset['Churn Label']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Train logistic regression model
log_reg_interaction = LogisticRegression(max_iter=1000)
log_reg_interaction.fit(X_train, y_train)

# Extract coefficients and intercept
coefficients = log_reg_interaction.coef_[0]
intercept = log_reg_interaction.intercept_[0]

# Create a DataFrame for coefficients
interaction_coefficients = pd.DataFrame({
   'Feature': ['Gender (Female)', 'Low Income (Very-Low)', 'Gender * Low Income Interaction'],
   'Coefficient': coefficients,
   'Odds Ratio': np.exp(coefficients)
})

# Display the coefficients and odds ratios
print("Interaction Model Coefficients:")
print(interaction_coefficients)
```

```
print(f"Intercept: {intercept}")
```

Hypothesis 4 - Dual relationship with satisfaction score and add ons

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression

# Step 1: Create a binary variable for satisfaction score below the threshold
final_dataset['Low Satisfaction'] = (final_dataset['Satisfaction Score'] < 3).astype(int)

# Step 2: Create an interaction term between "Add Ons" and "Low Satisfaction"
final_dataset['Add Ons * Low Satisfaction'] = final_dataset['Add Ons'] * final_dataset['Low Satisfaction']

# Step 3: Prepare the features and target variable
X = final_dataset[['Add Ons', 'Low Satisfaction', 'Add Ons * Low Satisfaction']]
y = final_dataset['Churn Label']

# Step 4: Fit the logistic regression model
log_reg_h4 = LogisticRegression(max_iter=1000, random_state=1234)
log_reg_h4.fit(X, y)

# Step 5: Extract coefficients and odds ratios
coefficients_h4 = log_reg_h4.coef_[0]
intercept_h4 = log_reg_h4.intercept_[0]

# Create a DataFrame for results
interaction_results_h4 = pd.DataFrame({
    'Feature': ['Add Ons', 'Low Satisfaction', 'Add Ons * Low Satisfaction'],
    'Coefficient': coefficients_h4
})
interaction_results_h4['Odds Ratio'] = np.exp(interaction_results_h4['Coefficient'])

# Display results
print("Interaction Model Coefficients:")
print(interaction_results_h4)
print(f"Intercept: {intercept_h4}")
```

Bibliography

"California LaborMarketInfo, The Economy." Accessed November 20, 2024. https://labormarketinfo.edd.ca.gov/cgi/databrowsing/localAreaProfileQSMoreResult.asp?viewAll=yes&viewAllUS=&currentPage=1&currentPageUS=&sortUp=I.INCOME&sortDown=&criteria=income&categoryType=population+census+data&geogArea=0604000009&timeseries=&more=More+Areas&menuChoice=localAreaPro&printerFriendly=&BackHistory=-4&goTOPageText=.

Kumar, Saravana. "Council Post: Customer Retention Versus Customer Acquisition." Forbes. Accessed November 23, 2024. https://www.forbes.com/councils/forbesbusinesscouncil/2022/12/12/customer-retention-versus-customer-acquisition/.

"Methodology for Determining Section 8 Income Limits," n.d.

"Telco Customer Churn (11.1.3+)." Accessed November 20, 2024. https://www.kaggle.com/datasets/alfathterry/telco-customer-churn-11-1-3.

"Telco Customer Churn (11.1.3+)," <date>. https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/11/telco-customer-churn-1113.

UnitedStatesZipCodes. "U.S. ZIP Codes: Free ZIP Code Map and Zip Code Lookup." Accessed November 20, 2024. https://www.unitedstateszipcodes.org.