Abstract

Customer churn is a critical aspect of subscription-based businesses seeing that keeping customers is much more economical than acquiring new ones. Many authors have used random forest models to tackle this issue [2][3], while many others have opted for deep learning models instead [1]. However, Extreme Gradient Boosting (XGBoost), a powerful and now popular tree-based model has not yet been thoroughly explored in this scenario, let alone compared to deep learning models in terms of performance and computational efficiency. This project's goal was to first experiment with under-sampling, rather than the predominant Synthetic Minority Over-sampling Technique (SMOTE), to address class imbalance; then create both an XGBoost and a Deep Neural Network (DNN) model, compare their performances and running times to determine whether, based on the IBM Telco Customer Churn dataset, deep learning models are needed to address customer churn or if simpler, and yet robust and modern tree-based models, such as XGBoost, could be a more economical and practical choice while being comparable in terms of performance; lastly, this project measured performance by using Distribution Optimally Balanced Stratified Cross-Validation (DOB-SCV) to avoid dataset shift, particularly on a dataset marked by class imbalance. For this particular dataset, both for the XGBoost and DNN classifiers, over-sampling with SMOTE yielded about two percent better performance for the evaluation metrics used, but at the expense of longer run times. The XGBoost classifier either outperformed or performed at the same level as the DNN classifier in terms of Accuracy, AUC, F1-Score, Precision and Recall while keeping run times of about thirty-two times faster than the DNN models' run times, which is a strong indication that the tree-based model is not only a more computationally efficient model, but also just as good, if not better, in terms of performance for this particular problem.

1. Introduction

The telecommunications industry has had considerable growth in the past few decades [1], especially considering the rise in fast speed mobile internet [3] and all the possibilities that come with it. If mere twenty years ago that industry was dedicated exclusively to phone service, now it encompasses much more. Mobile phones are now much more than just for making of receiving phone calls: They allow for writing and receiving emails, crafting and editing documents, playing games, making video calls and much more. Because of that, this industry is now intertwined with many others and the services offered by one are often bundled with those of others. A customer who buys a phone from company A will download games from company B using a browser from company C, pay for it using their bank account from company D, tell their friends about it on social media company E, all on one same device. Many of the services and products that can be accessed on a phone are often subscription-based rather than a one-time payment kind of commitment. For the customers, having the freedom to choose from so many different organizations is definitely beneficial, but it raises concerns for the companies offering those services [3]. One of the key factors which will determine a company's profitability, especially in a time where so many of them provide online services, is whether customers a loyal to their brand or not. Customer Churn is the term coined to describe a customer's decision to leave a given company [2], which can happen due to a myriad of reasons: Competition for said customers via higher quality or lower cost services and products; Lack of money during challenging times; or simply no longer enjoying a company's product. Whatever reason it may be, losing a customer is always costly as it will require marketing efforts to either re-gain their trust or replace them with new customers [1][2][3]. On the other hand, keeping one's customers engaged can be achieved at a relatively low cost as long as there is enough business intelligence to support that effort [3].

A significant part of the effort to retain customers starts with understanding exactly why they might leave so that decisions can be made to prevent that from happening [3]. Customer churn models try to accomplish exactly that and, while developing and deploying these models is a fairly common practice, many of them were once tree-based models, namely Random Forests. Once very popular, tree-based models seem to have lost their appeal in the midst of the rise of Deep Neural Network models. These models can be outstandingly accurate and just as flexible [5], with applications stemming from medical imaging, drug discovery, fraud detection, customer support, self-driving cars, product recommendation, image search and recognition, dynamic pricing, language translation, customer churn prediction and many others. However, these models also take a very significant amount of computing power and electricity, raising not only concerns regarding the amount of time to train and deploy them but also regarding how such increase in power consumption affects the environment and whether their overall impact on society is positive in light of that [4][5]. Because of all of these concerns, the model used on this project was intended as a compromise between performance and computational efficiency. To achieve that, a more modern and robust version of the so well-known Random Forests was used. Extreme Gradient Boosting models (XGBoost) build on Random Forests by accounting for each tree's errors at each step, executing their predictions sequentially rather than in parallel. These models are capable of handling complex data relationships and avoid overfitting, all the while being computationally efficient.

The data frame used for this project was the well-known IBM Telco Customer Churn dataset available on Kaggle, which contains seven thousand and forty-three rows and fifty columns. For more information on it, please see "Data Frame" and "Data Frame Summary" in the appendix. This project started with an exploratory data analysis to make sure there was a full understanding of the data, allowing for proper pre-processing and modeling. The first step was to find out the data type of each one of the fifty columns (forty-nine features and one target). Once it was determined that no feature needed to be cast to a different data type, the features were organized into three lists: str_cols, num_cols and to_drop. The first and second lists were used to hold string and numeric features, respectively, whereas the last one held features to be dropped. The last list was not meant to serve as a means for feature selection, but rather as a dataset cleaning as some features were just noise, had zero variance, had too great a proportion of null values, were redundant or correlated with other features. A couple of features had to be removed for special reasons, such as "Churn Score" and "Customer Status Stayed". The first was a score calculated by IBM to determine the likelihood of a given customer churning, the goal of this project, and the latter measured whether a customer had churned at the end of the quarter, much like the target itself. Histograms were plotted to investigate the distribution of each feature, revealing that most of them were either skewed or bimodal, which required either transforming or removing outliers as can be seen on "Histograms Compiled" in the appendix. Box-plots for each feature were also used to determine features with outliers as can be seen on "Box-Plots Compiled" in the appendix. An outlier analysis was performed and determined that those instances accounted for three quarters of all churners. More importantly, nine hundred of those outliers were customer with low satisfaction scores who churned, and are therefore important for building predictions. Senior citizens and customers under thirty years of age who churned were also outliers, but accounted for about half of all churners as well. Overall, given those circumstances, and keeping in mind the data frame had only seven thousand and forty-three instances, it was decided to keep the outliers and normalize the distributions with transformations. Some of those transformations yielded bimodal distributions, which were discretized by a given threshold so as to split the distribution into two features. Both sets of images can be seen on the appendix as "New Histograms Compiled" and "Bar Charts Compiled". At that point, old features (such as the original version of the discretized features) were dropped as well and all features were scaled to allow for better training of the chosen models, which can be seen on "Descriptive Statistics Scaled" on the appendix. Features were then properly selected combining three different methods and target imbalance was addressed prior to training, testing and evaluating the classifiers.

2. Literature Review

Saha et al. (2024) [1] was also concerned about the impact of customer churn in the telecommunications industry and decided to address that with a deep neural networks model. In fact, the authors developed their own, highly customized, deep learning model, the "ChurnNet". They decided to treat target imbalance by over-sampling it using different variations of the Synthetic Minority Over-sampling Technique (SMOTE) and achieved impressive accuracy results of 95.59%, 96.94%, and 97.52% on three different benchmark datasets.

Wagh et al. (2024) [2] approached customer churn much like Saha et al. (2024) [1], including the use of SMOTENN for over-sampling. However, these authors decided to stick with the true and tested tree models both for prediction and feature selection. They used both a Decision Tree and a Random Forest model to select features based on their feature importance scores and these same models achieved impressive 93% and 99% accuracy scores, respectively.

Wu et al. (2021) [3] aimed to produce a more comprehensive study rather than diving deep on any aspect of it. These authors tested six different models (Decision Tree, Random Forest, Ada Boost, Linear Regression, Naïve Bayes and Multi-Layer Perceptron) on three different benchmark datasets, with and without over-sampling with SMOTE. They measured performance in terms of Accuracy, Precision, Recall, F-1 and AUC, and achieved much more modest results, perhaps due to lack of pre-processing. These authors also performed customer segmentation, which lies outside the scope of this project.

While the authors above concerned themselves with tackling customer Churn in varied ways, Mehlin et al. (2023) [4] and Gowda et al. (2024) [5] focused on the rise of Deep Learning Models and how they affect the environment due to the fact that these are models that required tremendous computational power, thus leading to high energy consumption. If, in one hand, Mehlin et al. (2023) [4] seems to be more optimistic of the Deep Learning scape in regards to sustainability and makes an effort to catalogue the effort's development, on the other hand, Gowda et al. (2024) [5] seems to lie on the side of caution and proposes incorporating energy consumption as part of the performance evaluation for Deep Learning models. In fact, the authors go as far as to suggest a metric that evaluates accuracy per unit of electricity consumed. The works of Mehlin et al. (2023) [4] and Gowda et al. (2024) [5] paint an accurate portrait of the academic community: Excitement and support of state of the art models and their potential versus a more careful, environmentally and socially concerned approach.

This project differs from those of Saha et al. (2024) [1], Wagh et al. (2024) [2] and Wu et al. (2021) [3] firstly because it uses both a state-of-the-art tree-based model and a deep learning model with the intent to compare their performance and their computational efficiency, with the goal of making a recommendation on how to approach customer churn. Unlike Saha et al. (2024) [1], the deep learning model used on this project is not customized, but rather a simple Dense Neural Network (DNN) model to prioritize generalization and understanding. Unlike Wagh et al. (2024) [2], this project employed an Extreme Gradient Boosting (XGBoost) Classifier, rather than older tree-based models such as Decision Tree and Random Forest. Differently from Wu et al. (2021) [3], this project focuses on the comparison between two models built off of one dataset, rather than a broad comparison between many models built off of many datasets. This project also differs from those of all three authors on how target distribution imbalance is approached: All the three authors use over-sampling, more specifically SMOTE or one of its variations, while this project makes a comparison between under-sampling and over-sampling with SMOTE, prioritizing model's explainability to a non-technical audience. This project also differs from the works of Saha et al. (2024) [1], Wagh et al. (2024) [2] and Wu et al. (2021) [3] when it comes to pre-processing and feature selection. While the differences in pre-processing might not be so significant, the differences between what was done in terms of feature selection by these three authors and what was done in terms of feature selection on this project is a lot clearer as Saha et al. (2024) [1] and Wu et al. (2021) [3] do not seem to perform any feature selection and Wagh et al. (2024) [2] uses the simple feature importance from the Decision Tree and Random Forest models, while this project uses three distinct methods (two wrappers and one filter) together, on top of a preliminary manual feature selection, to come up with a robust set of features for the classifiers. Lastly, this project also differs from that of the other authors on how models are evaluated, as on this project a custom function was developed to use a more robust cross validation method to try and mitigate the possibility of dataset shift.

3. Methodology

3.1 Transformations

The dataset used has a large number of features, but not enough instances that any of them could easily be discarded. Because of that, as well as the fact that the outliers were informative, outlier analysis led to transformations rather than outlier removal or caping. However, performing multiple transformations per feature to find the one that best normalizes the distribution was not practical, thus requiring an automated transformation method. The PowerTransformer method from the Scikit-Learn library was used with the method parameter set to yeo-johnson. Unlike the Box-Cox transformation, the Yeo-Johnson transformation can address negative numbers as well as positive ones. The Yeo-Johnson uses maximum likelihood estimation to determine an optimal parameter lambda's value, which will determine the transformation to be applied. Some of the skewed distributions turned into bimodal ones and, by setting thresholds and using the where method from the numpy library, those continuous variables were replaced by multiple discretized features, allowing the classifiers to derive more meaningful information from each sub-distribution.

3.2 Feature Selection

Feature selection started with simple analysis to determine features that were either constant, noise, redundant, of high null value incidence or correlated with other features. To determine whether a feature was constant, histograms (for continuous features) and number of unique values (for categorical) were used. A few categorical features were considered noise if they had more than five unique values, but most of the features considered noise had many more than just five unique values with some of them having only unique values. Some features had both continuous and discretized versions and only one of them was kept in the data frame. Features with more than ten percent of null values were also removed from the data frame, rather than eliminating instances. To evaluate multicollinearity, both multiple iterations of correlation matrices and VIF scores were used, which can be seen in the appendix as "First Correlation Matrix" or "First VIF Scores" to "Fourth Correlation Matrix" or "Fourth VIF Scores".

Once this simple analysis was finished and a few features were removed, three distinct feature selection methods were applied: Scikit Learn's Recursive Feature Elimination (RFE), Select From Model and Select K Best. Recursive Feature Elimination simply fits a model, Random Forest in this case, calculates the most important features for that model and eliminates the least important one, repeating this process until the desired number of features is achieved. Select From Model follows a similar process by fitting a model, also Random Forest, and calculating feature importance but, instead of iteratively removing least important features, it sets a threshold, mean in this case, and removes all features below said threshold. Making for a less thorough but faster method. Select K Best uses a statistical scoring function, mutual information in this case, to calculate scores for each feature then ranks them based on said scoring function and selects the k best as per user input. The features selected by each model were placed in a data frame with the count of times they were selected so as to give an overall importance ranking. All twelve features in said ranking were kept in the data frame and can be seen in the appendix as "Features Selected".

3.3 Class Imbalance

The target distribution was plotted, which can be seen in the appendix as "Target Distribution", to give an idea of how imbalanced it was and whether under-sampling was a possibility. Under-sampling was achieved very easily by using the sample method from the pandas library, concatenating under-sampled majority (non-churned) and the minority class (churned) and shuffling the resulting data frame. Over-sampling was accomplished by using the SMOTE method from the imblearn library. Both under-sampled and over-sampled data frames were kept so as to experiment with both to find out how the classifiers would perform on each one of them.

3.4 Training, Testing and Evaluating models

The first step to start training, testing and evaluating the classifiers was to create a Dense Neural Network (DNN) class to define its structure, which included three hidden layers with batch normalization. The choice of three hidden layers was made in an attempt to balance performance and run time, and the batch normalization was chosen to stabilize learning and regularize the model to try and achieve more robust and faster convergence. This class also includes the forward propagation, with a relu activation function for the three hidden layers as well as a sigmoid activation function for the output layer.

The next step was to create a customized Distribution Optimally Balanced Stratified Cross-Validation (DOB-SCV) function, which takes a data frame, a target column's name, model type, number of splits for the cross validation, random state, learning rate for either model, number of estimators and maximum depth for the XGBoost classifier, epochs, learning rate and batch size for the DNN model. The function then uses the StratifiedKFold method from the Scikit-Learn library to create the stratified k folds' indexes. It then splits into training and evaluation for either XGBoost and DNN classifiers. For the XGBoost classifier, it uses these indexes to split the data frame into train and test sets, fits a given model, makes the predictions, calculates and stores Accuracy, AUC, F1, Precision and Recall scores as well as Confusion Matrices for each iteration. For the DNN classifier, it also creates a "device" variable set to cuda (GPU), if possible, and CPU otherwise; Converts train and test datasets into pytorch tensors; Creates a data loader object for mini-batch training; Initializes model, loss criterion (set to BCELoss), and optimizer (set to Adam) and then finally trains the model for each epoch. Iterating over epochs starts by putting the model in training mode, then iterating each batch of features and target from the train loader. It creates the mini-batches, clears gradients from previous iterations, calculates and stores the output, calculates and stores the loss, backpropagates the error to calculate the gradient of the loss with respect to each parameter and updates the model parameters using the gradients calculated by backpropagation. The next and final step for the DNN model is to calculate performance metrics and store their results. Results are then aggregated by averages and both aggregated and individual metrics are printed and returned.

A customized search grid function was created to receive the parameters for the DOB-SCV function as inputs, create four hundred and eighty unique combinations of parameters for the XGBoost model and two hundred and sixteen unique combinations of parameters for the DNN classifier using the product method from the itertools library, then call the DOB-SCV function for each parameter combination to return a list with results for each one of them. Results for both the XGBoost and DNN classifiers on under-sampled and over-sampled data frames were stored into pandas data frames, and those result data frames were stored into csv files for easier posterior access. Indexes for the best result for each performance metric, for each classifier, for each data frame were gathered to create tables with all combinations of parameters and their results, allowing to determine which classifier was best suited for the task while keeping in mind performance metrics as well as run times and explainability to an untrained audience. Once a classifier was chosen, its results were further investigated by plotting a line chart with a given hyper parameter on the x axis and the performance metrics on the y axis, which can be seen in the appendix as "Performance Scores".

4. Results

Results for model "A", those of index 400 on the "Under Sampled XGBoost Results" in the appendix, come from a less complex model, with a learning rate of 0.1 and only 100 trees with a maximum depth of 3. It also uses a ten-fold cross validation evaluation, which is industry standard, making it the most appealing XGBoost Classifier on the under sampled dataset. Results for model "B", those of index 83 on the "Over Sampled XGBoost Results" in the appendix, are better than those of indexes 20 and 302 and virtually the same as those of index 338. However, they were achieved in much better run time than that of index 338, making it the most appealing XGBoost Classifier on the over sampled dataset. Results for model "C", those of index 75 on the "Under Sampled DNN Results" in the appendix, are very comparable to those of the remaining indexes for the same table, but at a much lower run time of 16 seconds whereas the run times for the other models on that table range from 31 to 94 seconds. That makes model "C" the most appealing DNN Classifier model on the under sampled dataset. Results for model "D", those of index 13 on the "Over Sampled DNN Results" in the appendix, are very comparable to those of the remaining indexes for the same table, but at a lower run time of 117 seconds whereas the run times for the other models on that table range from 126 to 351 seconds. That makes model "D" the most appealing DNN Classifier model on the over sampled dataset.

When comparing results for model "A" to the results for model "B", it is clear that scores for the latter are slightly higher (at about 2%) but at an unusual number of folds for the cross-validation evaluation and at a higher maximum depth, thus leading to a higher run time. More importantly, however, is the fact that if using the under sampled dataset there is no need for explaining, potentially to a non-technical audience, the creation and use of synthetic data, making model “A” easier to sell than model “B” while at the same performance level. The results for model “A” are at the same performance level as the results for model “C”, but run in about half a second instead of about 16 seconds. Results for model “A” range from 2% worse (for Average Recall) to the same (for Average AUC) of those from model “D”, but were achieved in about half a second rather than 117 seconds and did not require the creation and use of synthetic data. Therefore, the most appealing Classifier is model “A”, the XGBoost Classifier built off of the under-sampled data frame.

5. Conclusions and future work

Based on the grid search results it seemed like the XGBoost Classifier built off of the under-sampled data frame, with a learning rate of 0.1 and only 100 trees with a maximum depth of 3, evaluated with a ten-fold DOB-SCV, was the best model. However, based on “Performance Scores” in the appendix, it is clear that there is little to be gained after 40 trees, and after 60 trees some of the performance metrics worsen. The best classifier for this particular dataset, therefore, is an XGBoost classifier built off of the under-sampled data frame, with a learning rate of 0.1, only 60 trees with a maximum depth of 3, evaluated with a ten-fold DOB-SCV. This model runs in about half a second whereas the best DNN classifier runs in about 16 seconds, or roughly 32 times the run time of the chosen XGBoost classifier. The initial expectation was that the tree-based model would compromise in performance to achieve better run times and power consumption than the Deep Learning model. However, the XGBoost Classifier also delivered performance at the same level or better, depending on the metric.

While it is clear that the tree-based model is a better choice for predicting customer churn on this particular dataset, that might not be the case on a different one. The goal of this project was to conduct a thorough analysis of both models and, because of that, exploring other datasets was not practical. Although the target variable was imbalanced, the minority class accounted for roughly a quarter of all instances, which might have affected the classifiers’ performances positively, thus motivating future work to investigate further which kind of model is better suited for predicting customer churn.

6. References

1. **ChurnNet: Deep Learning Enhanced Customer Churn Prediction in Telecommunication Industry**
   Saha, S., Saha, C., Haque, M. M., Alam, M. G. R., & Talukder, A. (2024). ChurnNet: Deep learning enhanced customer churn prediction in telecommunication industry. *IEEE Access*, 12, 4471-4484. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10380579
2. **Customer Churn Prediction in Telecom Sector Using Machine Learning Techniques**
   Wagh, S. K., Andhale, A. A., Wagh, K. S., Pansare, J. R., Ambadekar, S. P., & Gawande, S. H. (2024). Customer churn prediction in telecom sector using machine learning techniques. *Results in Control and Optimization*, 14, 100342. Retrieved from https://www.sciencedirect.com/science/article/pii/S2666720723001443
3. **Integrated Churn Prediction and Customer Segmentation Framework for Telco Business**
   Wu, S., Yau, W.-C., Ong, T.-S., & Chong, S.-C. (2021). Integrated churn prediction and customer segmentation framework for telco business. *IEEE Access*, 9, 62118-62129. Retrieved from https://ieeexplore.ieee.org/document/9406002
4. **Towards Energy-Efficient Deep Learning: An Overview of Energy-Efficient Approaches Along the Deep Learning Lifecycle**
   Mehlin, V., Schacht, S., & Lanquillon, C. (2023). Towards energy-efficient deep learning: An overview of energy-efficient approaches along the deep learning lifecycle. *Proceedings of the 2023 International Conference on Energy and AI*. Retrieved from https://arxiv.org/abs/2303.01980
5. **Watt for What: Rethinking Deep Learning's Energy-Performance Relationship**
   Gowda, S. N., Hao, X., Li, G., Narayana Gowda, S., Jin, X., & Sevilla-Lara, L. (2024). Watt for what: Rethinking deep learning's energy-performance relationship. *arXiv*. Retrieved from https://arxiv.org/abs/2310.06522

7. Appendix

Image 1 - Data Frame

Column Name	Data Type	% Null Instances	Number of Unique Values
Customer ID	str	0%	7043
Gender	str	0%	2
Age	numpy.int64	0%	NA
Under 30	str	0%	2
Senior Citizen	str	0%	2
Married	str	0%	2
Dependents	str	0%	2
Number of Dependents	numpy.int64	0%	NA
Country	str	0%	1
State	str	0%	1
City	str	0%	1106
Zip Code	numpy.int64	0%	NA
Latitude	numpy.float64	0%	NA
Longitude	numpy.float64	0%	NA
Population	numpy.int64	0%	NA
Quarter	str	0%	1
Referred a Friend	str	0%	2
Number of Referrals	numpy.int64	0%	NA
Tenure in Months	numpy.int64	0%	NA
Offer	float	55%	NA
Phone Service	str	0%	2
Avg Monthly Long Distance Charges	numpy.float64	0%	NA
Multiple Lines	str	0%	2
Internet Service	str	0%	2
Internet Type	str	22%	3
Avg Monthly GB Download	numpy.int64	0%	NA
Online Security	str	0%	2
Online Backup	str	0%	2
Device Protection Plan	str	0%	2
Premium Tech Support	str	0%	2
Streaming TV	str	0%	2
Streaming Movies	str	0%	2
Streaming Music	str	0%	2
Unlimited Data	str	0%	2
Contract	str	0%	3
Paperless Billing	str	0%	2
Payment Method	str	0%	3
Monthly Charge	numpy.float64	0%	NA
Total Charges	numpy.float64	0%	NA
Total Refunds	numpy.float64	0%	NA
Total Extra Data Charges	numpy.int64	0%	NA
Total Long Distance Charges	numpy.float64	0%	NA
Total Revenue	numpy.float64	0%	NA
Satisfaction Score	numpy.int64	0%	NA
Customer Status	str	0%	3
Churn Label	str	0%	2
Churn Score	numpy.int64	0%	NA
CLTV	numpy.int64	0%	NA
Churn Category	str	73%	5
Churn Reason	str	73%	20

Image 2 – Data Frame Summary

Number of instances	7043
Number of columns	50
Numeric Features	40%
Categorical Features	60%
Features with null values	8%
% Null values from	22%
% Null values to	73%
Constant Features	6%
Noise Features	6%
Redundant Features	8%

Image 3 – Histograms Compiled

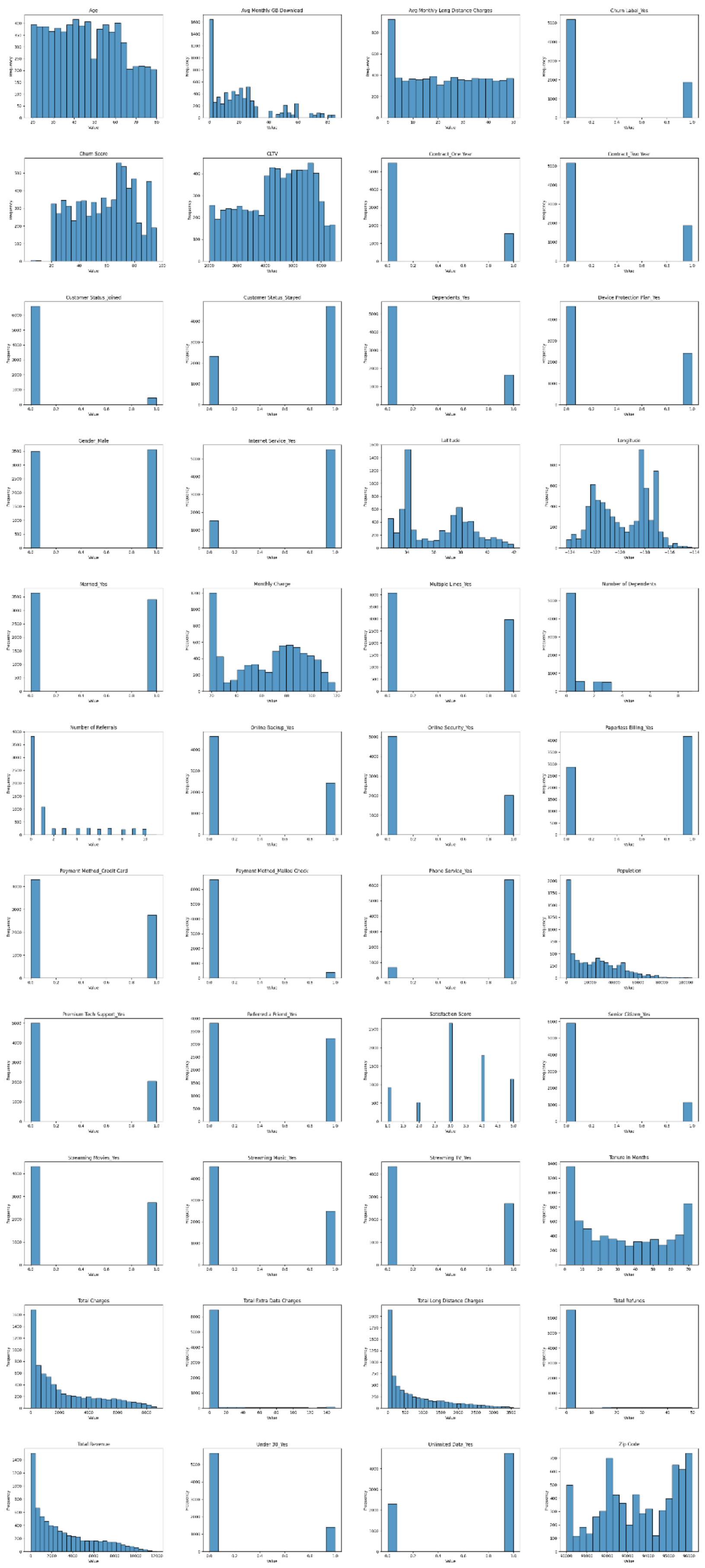


Image 4 – Box Plots Compiled

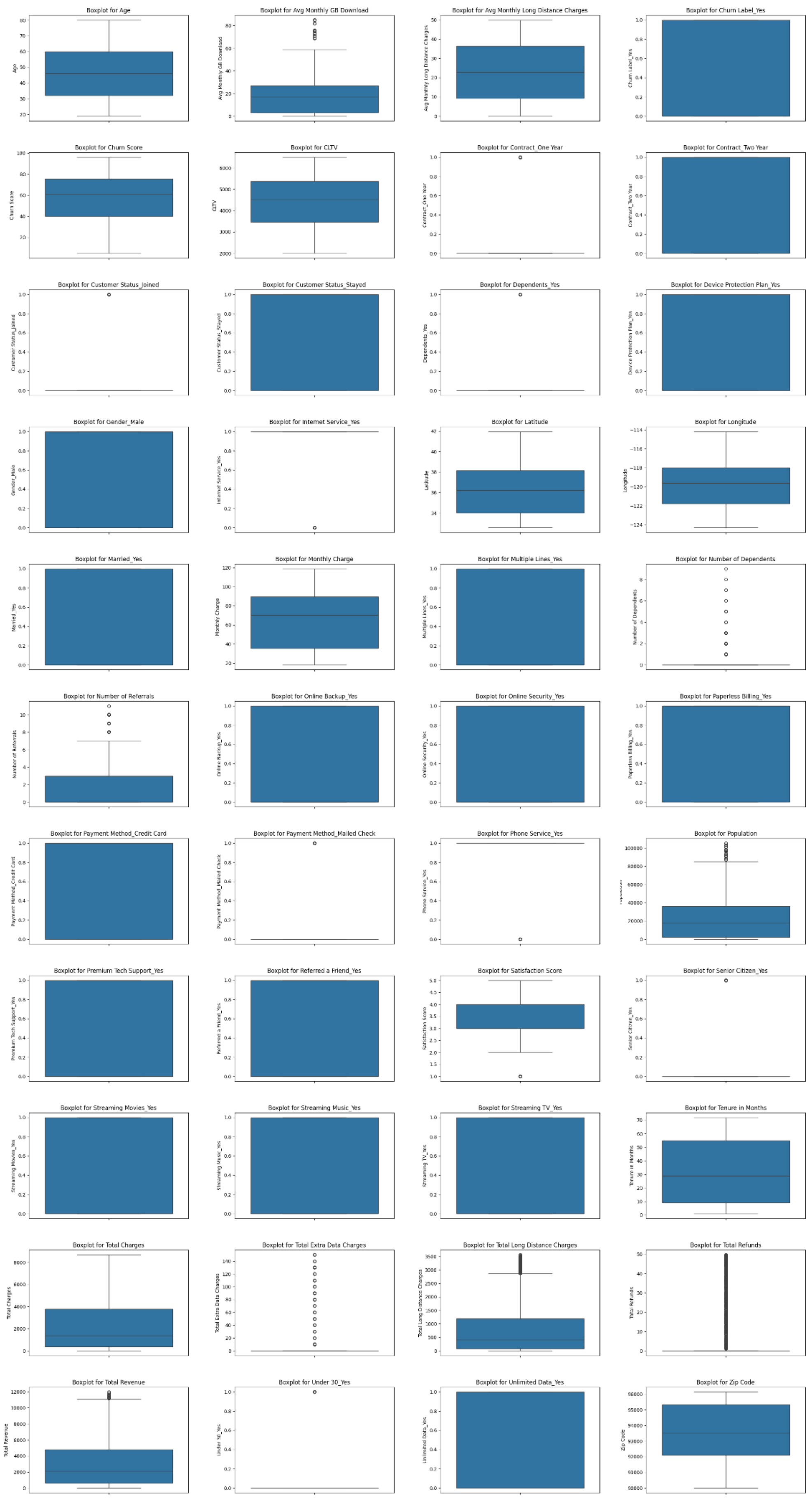
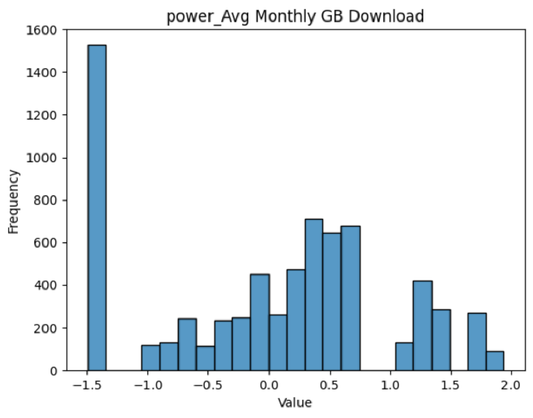
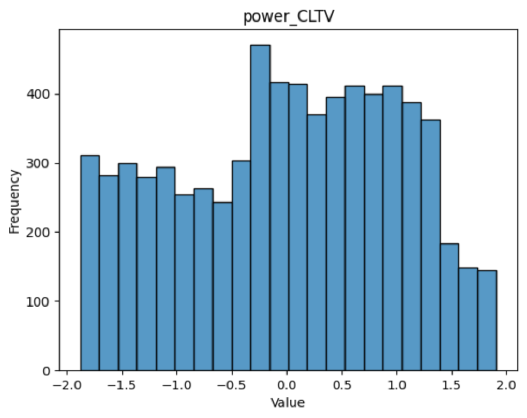
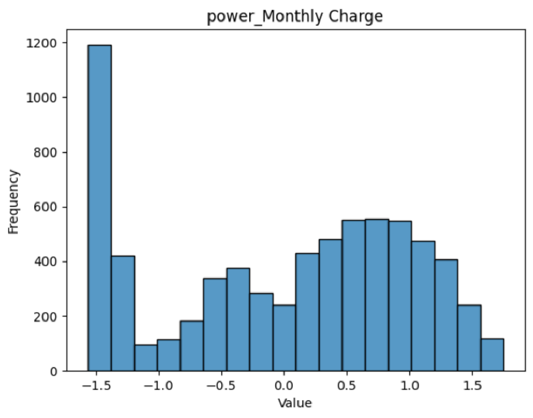
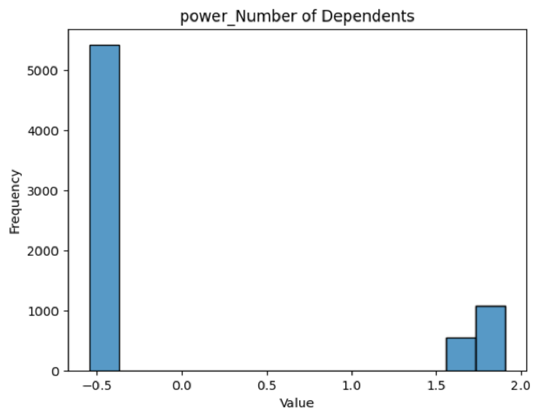


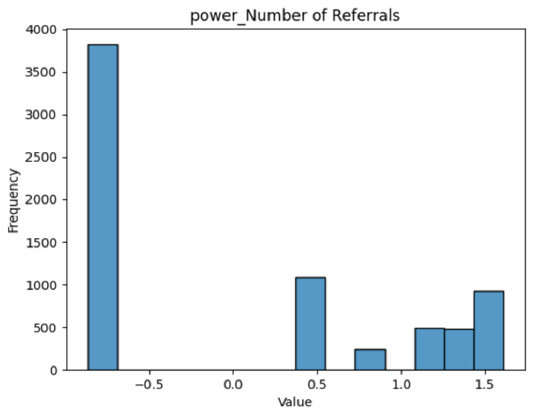
Image 5 – New Histograms Compiled

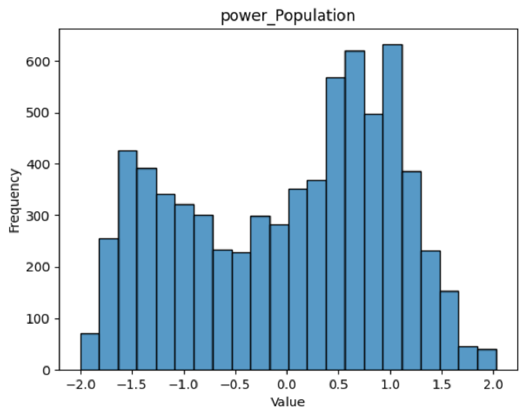


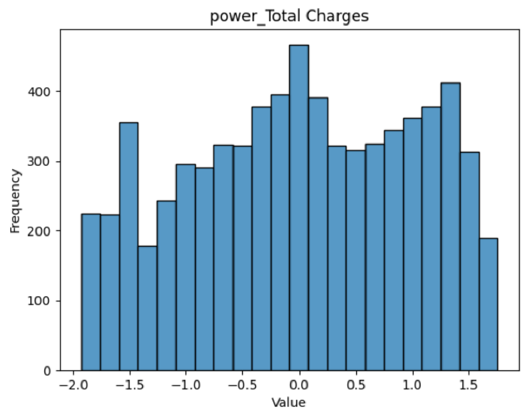












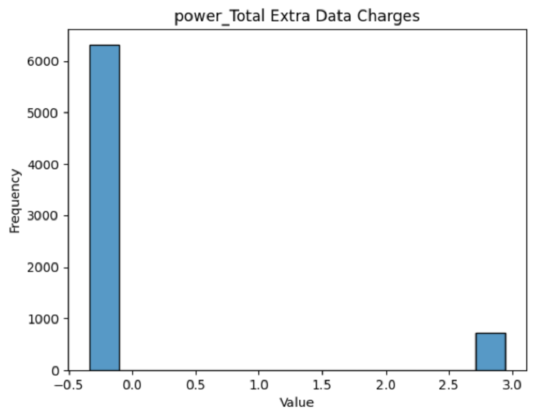


Image 6 – Bar Charts Compiled

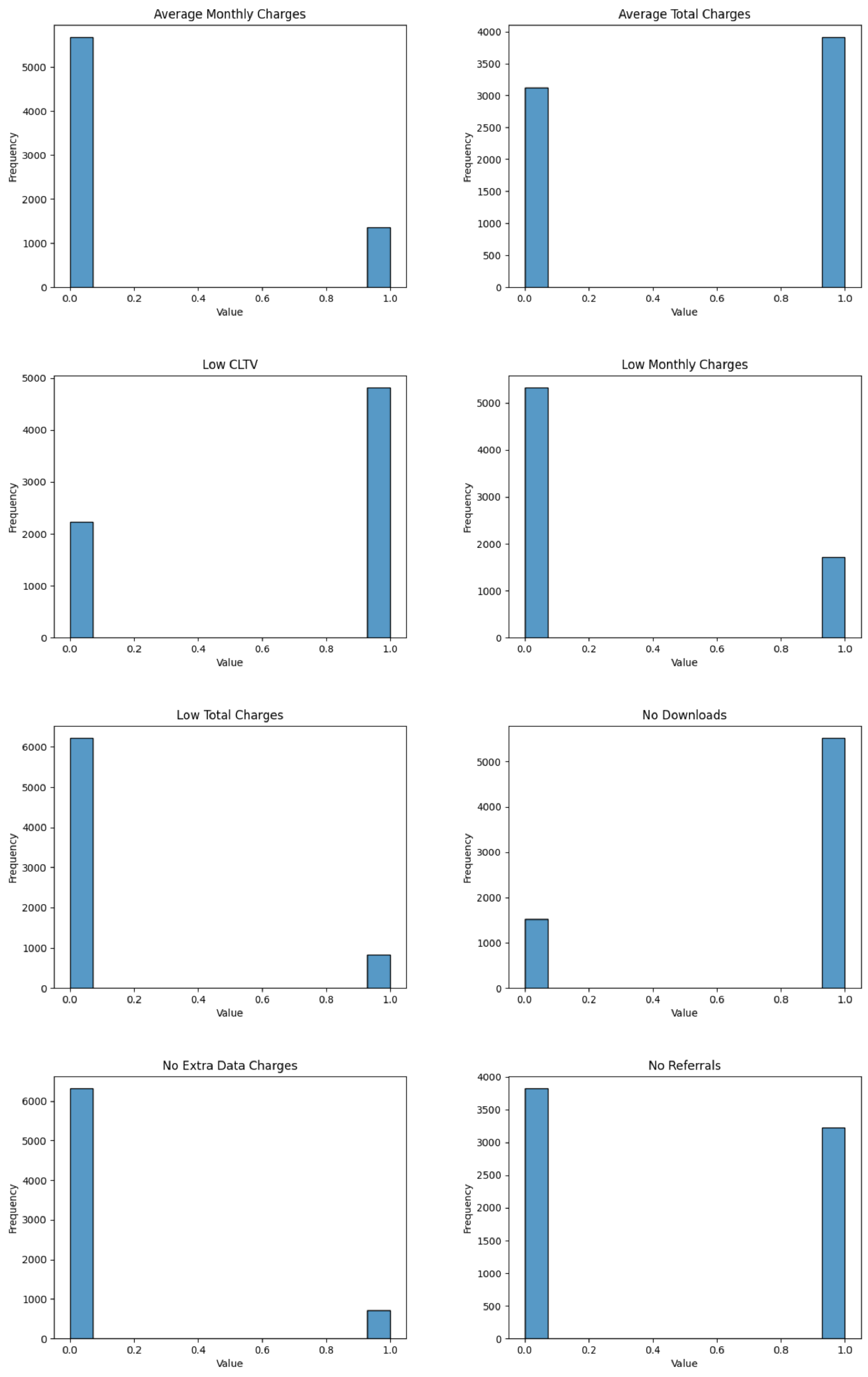


Image 7 – Descriptive Statistics Scaled

Column Name	count	mean	std	min	25%	50%	75%	max
Satisfaction Score	7043	0.56	0.30	0.00	0.50	0.50	0.75	1.00
Total Refunds	7043	0.04	0.16	0.00	0.00	0.00	0.00	1.00
Longitude	7043	0.45	0.21	0.00	0.25	0.47	0.63	1.00
Churn Score	7043	0.59	0.23	0.00	0.38	0.62	0.77	1.00
Age	7043	0.45	0.27	0.00	0.21	0.44	0.67	1.00
Latitude	7043	0.39	0.26	0.00	0.15	0.39	0.60	1.00
Avg Monthly Long Distance Charges	7043	0.46	0.31	0.00	0.18	0.46	0.73	1.00
Zip Code	7043	0.57	0.30	0.00	0.34	0.57	0.87	1.00
Total Long Distance Charges	7043	0.21	0.24	0.00	0.02	0.11	0.33	1.00
Total Revenue	7043	0.25	0.24	0.00	0.05	0.17	0.40	1.00
Streaming Music_Yes	7043	0.35	0.48	0.00	0.00	0.00	1.00	1.00
Under 30_Yes	7043	0.20	0.40	0.00	0.00	0.00	0.00	1.00
Paperless Billing_Yes	7043	0.59	0.49	0.00	0.00	1.00	1.00	1.00
Multiple Lines_Yes	7043	0.42	0.49	0.00	0.00	0.00	1.00	1.00
Married_Yes	7043	0.48	0.50	0.00	0.00	0.00	1.00	1.00
Premium Tech Support_Yes	7043	0.29	0.45	0.00	0.00	0.00	1.00	1.00
Customer Status_Joined	7043	0.06	0.25	0.00	0.00	0.00	0.00	1.00
Customer Status_Stayed	7043	0.67	0.47	0.00	0.00	1.00	1.00	1.00
Internet Service_Yes	7043	0.78	0.41	0.00	1.00	1.00	1.00	1.00
Device Protection Plan_Yes	7043	0.34	0.48	0.00	0.00	0.00	1.00	1.00
Streaming TV_Yes	7043	0.38	0.49	0.00	0.00	0.00	1.00	1.00
Phone Service_Yes	7043	0.90	0.30	0.00	1.00	1.00	1.00	1.00
Churn Label_Yes	7043	0.27	0.44	0.00	0.00	0.00	1.00	1.00
Online Security_Yes	7043	0.29	0.45	0.00	0.00	0.00	1.00	1.00
Contract_One Year	7043	0.22	0.41	0.00	0.00	0.00	0.00	1.00
Contract_Two Year	7043	0.27	0.44	0.00	0.00	0.00	1.00	1.00
Gender_Male	7043	0.50	0.50	0.00	0.00	1.00	1.00	1.00
Online Backup_Yes	7043	0.34	0.48	0.00	0.00	0.00	1.00	1.00
Streaming Movies_Yes	7043	0.39	0.49	0.00	0.00	0.00	1.00	1.00
Payment Method_Credit Card	7043	0.39	0.49	0.00	0.00	0.00	1.00	1.00
Payment Method_Mailed Check	7043	0.05	0.23	0.00	0.00	0.00	0.00	1.00
Unlimited Data_Yes	7043	0.67	0.47	0.00	0.00	1.00	1.00	1.00
Senior Citizen_Yes	7043	0.16	0.37	0.00	0.00	0.00	0.00	1.00
Referred a Friend_Yes	7043	0.46	0.50	0.00	0.00	0.00	1.00	1.00
Dependents_Yes	7043	0.23	0.42	0.00	0.00	0.00	0.00	1.00
Low Tenure	7043	0.42	0.49	0.00	0.00	0.00	1.00	1.00
Small Population	7043	0.66	0.47	0.00	0.00	1.00	1.00	1.00
Low Total Charges	7043	0.12	0.32	0.00	0.00	0.00	0.00	1.00
Average Total Charges	7043	0.56	0.50	0.00	0.00	1.00	1.00	1.00
Low CLTV	7043	0.68	0.47	0.00	0.00	1.00	1.00	1.00
Low Monthly Charges	7043	0.24	0.43	0.00	0.00	0.00	0.00	1.00
Average Monthly Charges	7043	0.19	0.39	0.00	0.00	0.00	0.00	1.00
No Referrals	7043	0.46	0.50	0.00	0.00	0.00	1.00	1.00
No Extra Data Charges	7043	0.10	0.30	0.00	0.00	0.00	0.00	1.00
No Downloads	7043	0.78	0.41	0.00	1.00	1.00	1.00	1.00

Image 8 – First Correlation Matrix

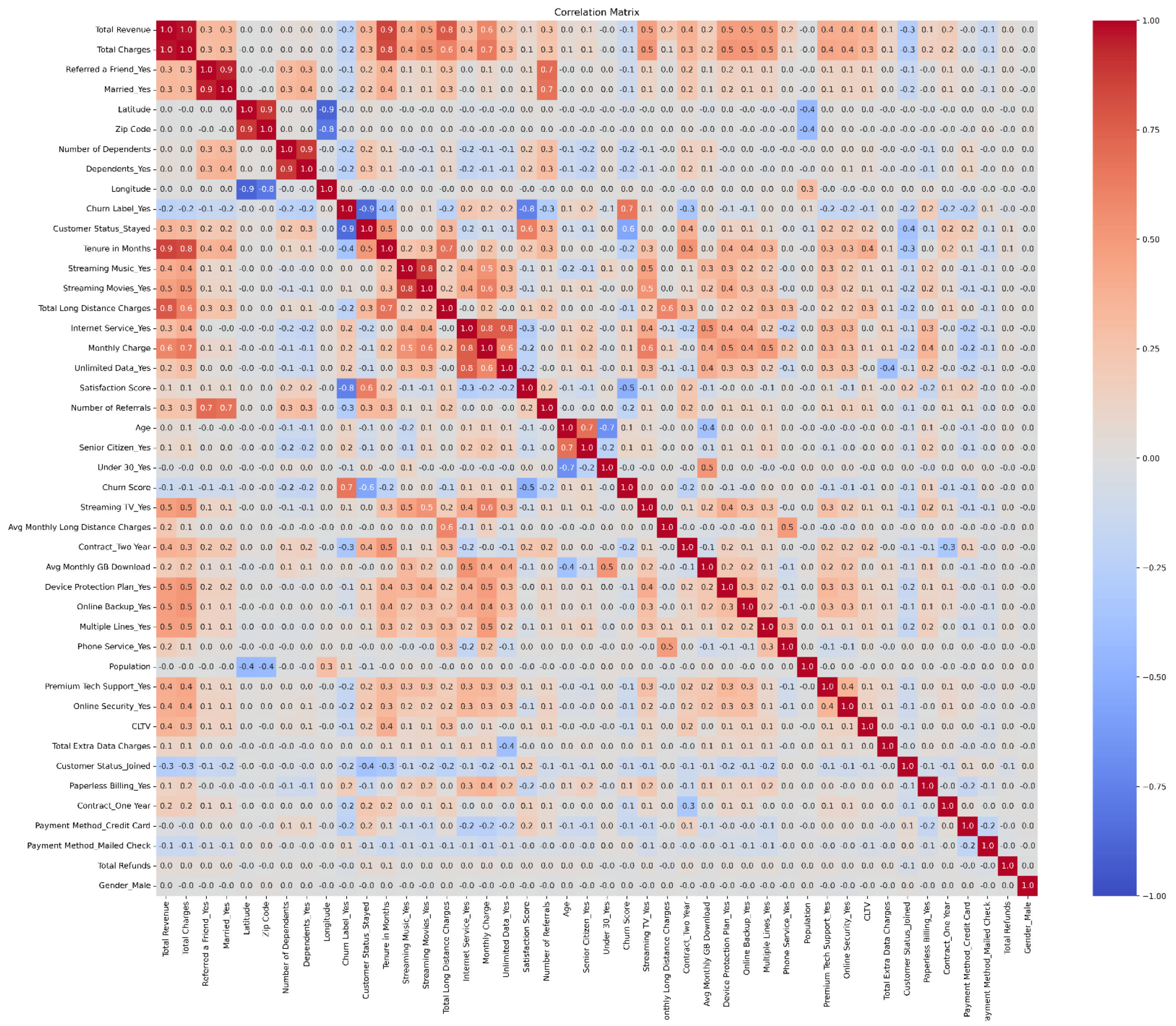


Image 9 – Second Correlation Matrix

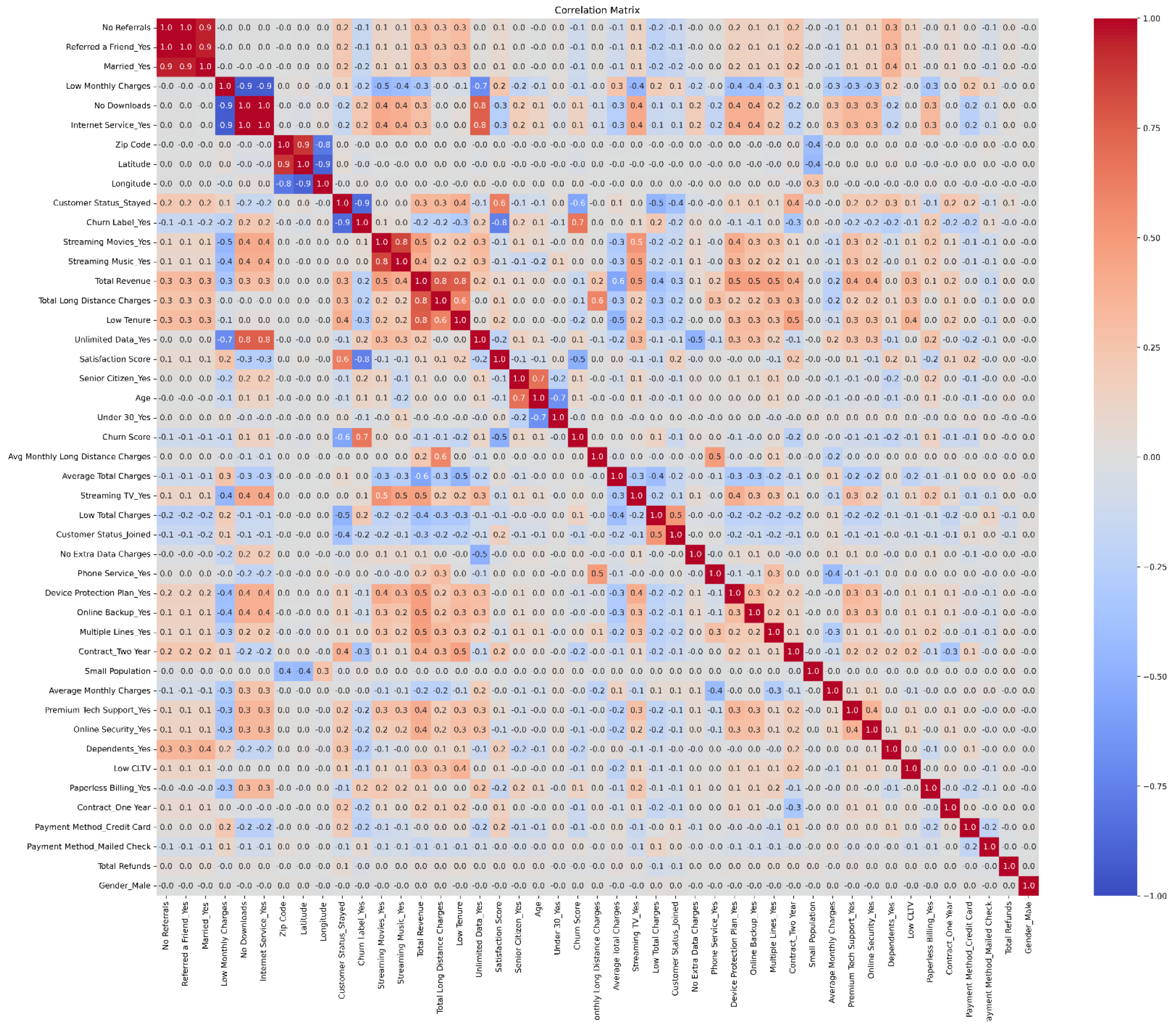


Image 10 – Third Correlation Matrix

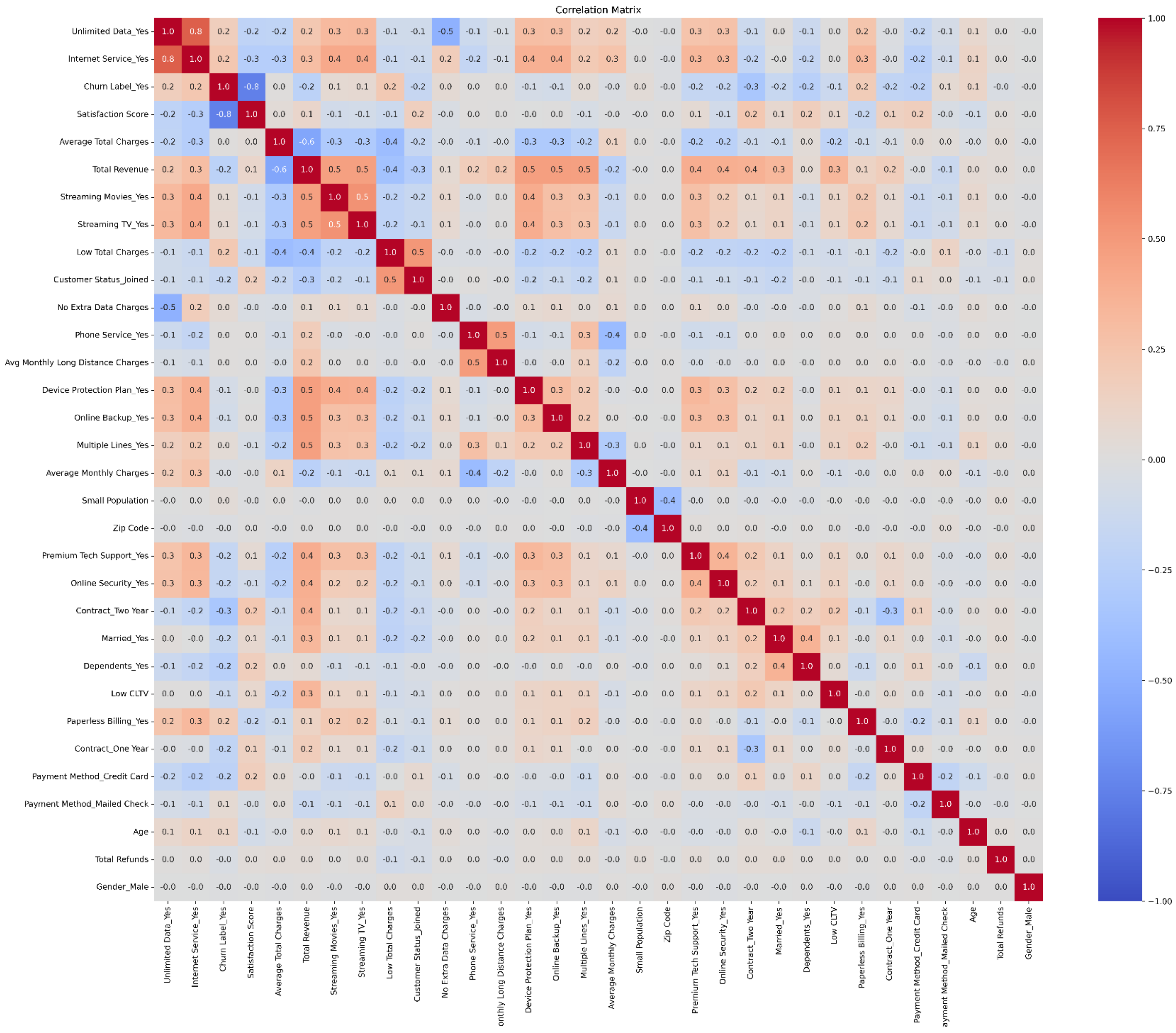


Image 11 – Fourth Correlation Matrix

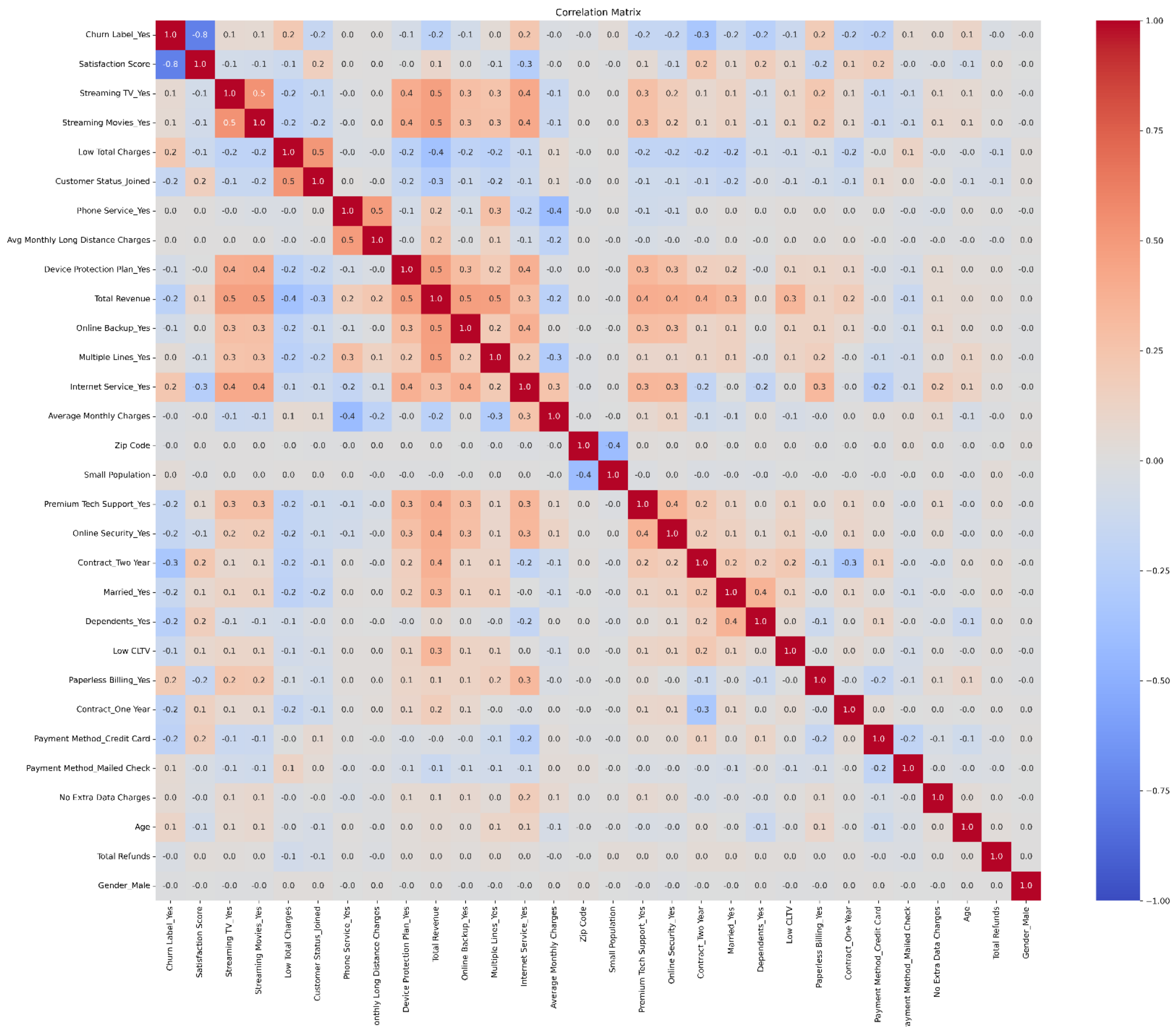
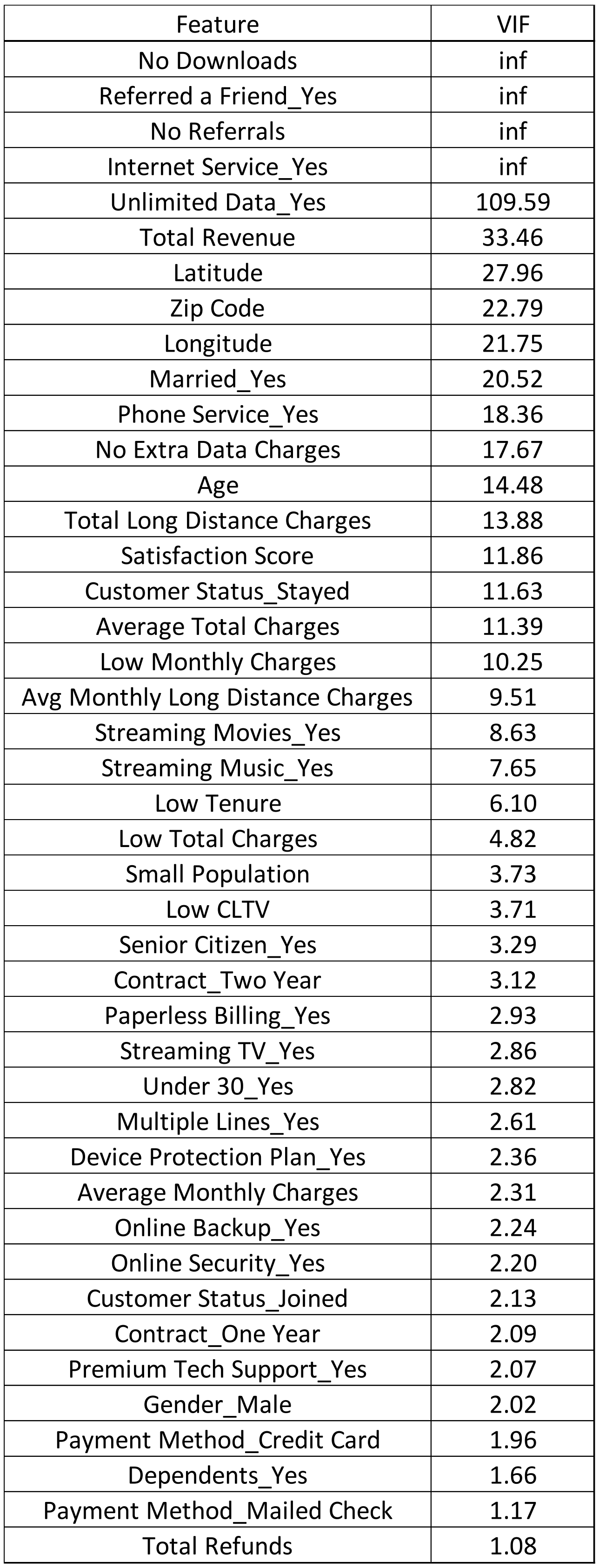


Image 12 – First VIF

Feature	VIF
Total Extra Data Charges	inf
Total Charges	inf
Total Long Distance Charges	inf
Total Revenue	inf
Total Refunds	inf
Zip Code	7267.49
Longitude	7153.69
Latitude	488.42
Monthly Charge	80.78
Internet Service_Yes	47.59
Age	35.60
Phone Service_Yes	27.86
Tenure in Months	25.72
Satisfaction Score	21.98
Referred a Friend_Yes	20.94
Married_Yes	20.58
Unlimited Data_Yes	18.10
CLTV	17.81
Customer Status_Stayed	11.41
Avg Monthly Long Distance Charges	10.12
Streaming Movies_Yes	9.14
Streaming Music_Yes	7.64
Dependents_Yes	6.56
Number of Dependents	6.18
Avg Monthly GB Download	5.35
Under 30_Yes	3.56
Streaming TV_Yes	3.37
Senior Citizen_Yes	3.31
Contract_Two Year	3.31
Number of Referrals	3.09
Paperless Billing_Yes	2.97
Multiple Lines_Yes	2.78
Population	2.75
Device Protection Plan_Yes	2.41
Online Backup_Yes	2.28
Online Security_Yes	2.19
Contract_One Year	2.12
Premium Tech Support_Yes	2.07
Customer Status_Joined	2.04
Gender_Male	2.03
Payment Method_Credit Card	1.99
Payment Method_Mailed Check	1.18

Image 13 – Second VIF



Feature	VIF
No Downloads	inf
Referred a Friend_Yes	inf
No Referrals	inf
Internet Service_Yes	inf
Unlimited Data_Yes	109.59
Total Revenue	33.46
Latitude	27.96
Zip Code	22.79
Longitude	21.75
Married_Yes	20.52
Phone Service_Yes	18.36
No Extra Data Charges	17.67
Age	14.48
Total Long Distance Charges	13.88
Satisfaction Score	11.86
Customer Status_Stayed	11.63
Average Total Charges	11.39
Low Monthly Charges	10.25
Avg Monthly Long Distance Charges	9.51
Streaming Movies_Yes	8.63
Streaming Music_Yes	7.65
Low Tenure	6.10
Low Total Charges	4.82
Small Population	3.73
Low CLTV	3.71
Senior Citizen_Yes	3.29
Contract_Two Year	3.12
Paperless Billing_Yes	2.93
Streaming TV_Yes	2.86
Under 30_Yes	2.82
Multiple Lines_Yes	2.61
Device Protection Plan_Yes	2.36
Average Monthly Charges	2.31
Online Backup_Yes	2.24
Online Security_Yes	2.20
Customer Status_Joined	2.13
Contract_One Year	2.09
Premium Tech Support_Yes	2.07
Gender_Male	2.02
Payment Method_Credit Card	1.96
Dependents_Yes	1.66
Payment Method_Mailed Check	1.17
Total Refunds	1.08

Image 14 – Third VIF

Feature	VIF
Internet Service_Yes	134.38
Unlimited Data_Yes	109.41
No Extra Data Charges	17.64
Phone Service_Yes	13.62
Total Revenue	13.12
Average Total Charges	7.12
Satisfaction Score	5.60
Zip Code	4.87
Avg Monthly Long Distance Charges	4.81
Age	3.82
Low CLTV	3.58
Small Population	3.27
Low Total Charges	3.22
Paperless Billing_Yes	2.89
Contract_Two Year	2.83
Streaming Movies_Yes	2.76
Streaming TV_Yes	2.72
Multiple Lines_Yes	2.58
Married_Yes	2.55
Device Protection Plan_Yes	2.31
Online Backup_Yes	2.20
Online Security_Yes	2.03
Premium Tech Support_Yes	2.02
Gender_Male	2.00
Contract_One Year	1.97
Average Monthly Charges	1.91
Payment Method_Credit Card	1.90
Dependents_Yes	1.61
Customer Status_Joined	1.61
Payment Method_Mailed Check	1.15
Total Refunds	1.08

Image 15 – Fourth VIF

Feature	VIF
Phone Service_Yes	12.46
Internet Service_Yes	10.50
Total Revenue	7.29
Satisfaction Score	5.36
Zip Code	4.60
Avg Monthly Long Distance Charges	4.56
Age	3.75
Low CLTV	3.52
Small Population	3.10
Paperless Billing_Yes	2.87
Contract_Two Year	2.77
Streaming Movies_Yes	2.75
Streaming TV_Yes	2.72
Multiple Lines_Yes	2.58
Married_Yes	2.53
Device Protection Plan_Yes	2.31
Online Backup_Yes	2.20
Online Security_Yes	2.03
Premium Tech Support_Yes	2.02
Gender_Male	1.98
Contract_One Year	1.96
Average Monthly Charges	1.88
Payment Method_Credit Card	1.87
Low Total Charges	1.76
Dependents_Yes	1.61
Customer Status_Joined	1.60
No Extra Data Charges	1.16
Payment Method_Mailed Check	1.14
Total Refunds	1.08

Image 16 – Target Distribution

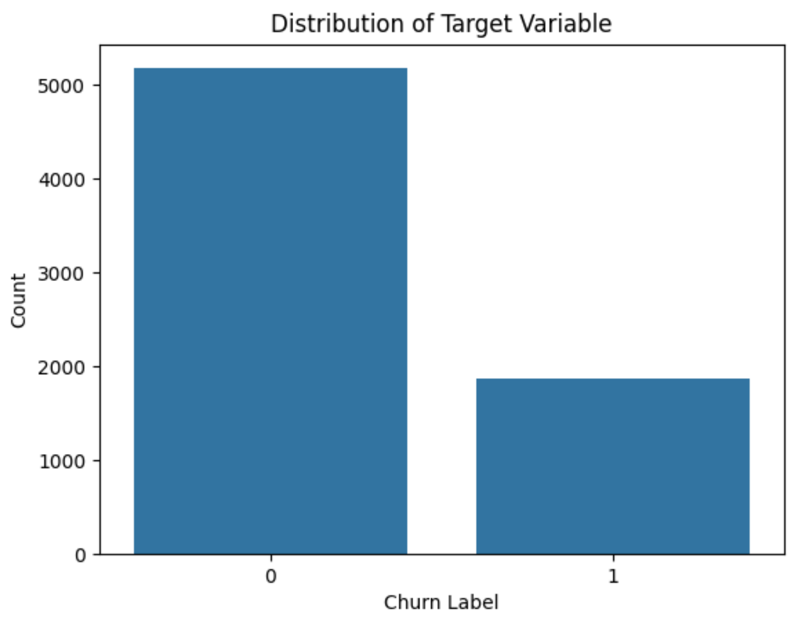


Image 17 – Features Selected

Feature	Count
Contract_Two Year	3
Satisfaction Score	3
Total Revenue	3
Customer Status_Joined	3
Zip Code	2
Age	2
Contract_One Year	2
Dependents_Yes	2
Payment Method_Credit Card	1
Married_Yes	1
Online Security_Yes	1
Avg Monthly Long Distance Charges	1
Internet Service_Yes	1

Image 18 – Under Sampled XGBoost Results

Index	400	241	178	221
xgb_learning_rate	0.1	0.1	0.1	0.0001
n_estimators	100	100	500	100
max_depth	3	4	5	4
cv_folds	10	8	7	7
run_time	0.55	0.53	2.11	0.44
Average Accuracy	0.95	0.96	0.95	0.91
Average AUC	0.99	0.99	0.99	0.98
Average F1 Score	0.95	0.96	0.95	0.92
Average Precision	0.94	0.95	0.96	0.87
Average Recall	0.96	0.96	0.95	0.98

Image 19 – Over Sampled XGBoost Results

Index	338	83	20	302
xgb_learning_rate	0.1	0.1	0.01	0.0001
n_estimators	500	100	100	100
max_depth	5	6	3	5
cv_folds	9	6	5	8
run_time	3.60	0.78	0.41	0.74
Average Accuracy	0.97	0.97	0.93	0.94
Average AUC	1.00	1.00	0.99	0.99
Average F1 Score	0.97	0.97	0.93	0.94
Average Precision	0.98	0.98	1.00	0.91
Average Recall	0.97	0.97	0.86	0.98

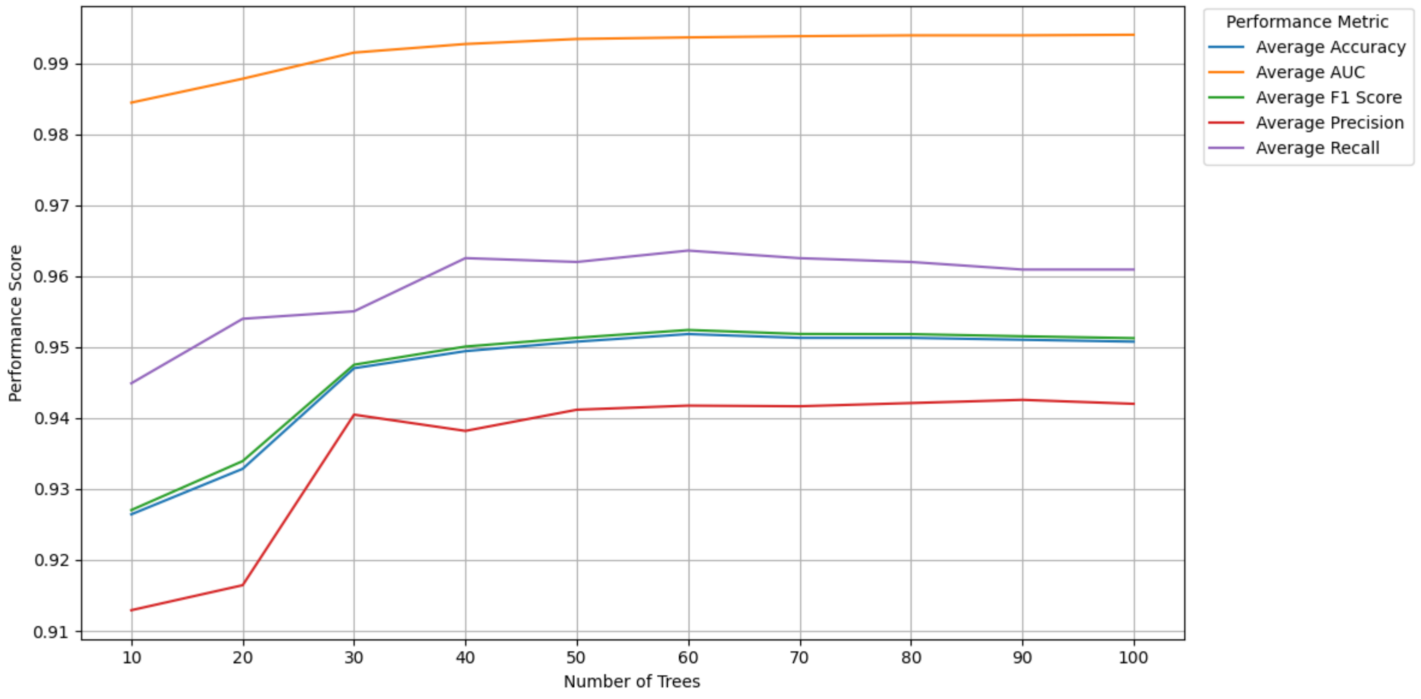
Image 20 – Under Sampled DNN Results

Index	2	168	75	76	94
epochs	25	75	25	25	75
learning_rate	0.001	0.00001	0.0001	0.0001	0.0001
batch_size	16	64	64	32	32
cv_folds	5	9	7	7	7
run_time	40.48	66.54	16.43	31.52	94.15
Average Accuracy	0.94	0.95	0.95	0.95	0.95
Average AUC	0.99	0.99	0.99	0.99	0.99
Average F1 Score	0.95	0.95	0.95	0.95	0.95
Average Precision	0.93	0.95	0.95	0.94	0.95
Average Recall	0.96	0.94	0.95	0.96	0.95

Image 21 – Over Sampled DNN Results

Index	27	100	13	190
epochs	100	100	50	50
learning_rate	0.001	0.001	0.0001	0.001
batch_size	64	32	32	32
cv_folds	5	7	5	10
run_time	126.07	351.28	117.07	263.63
Average Accuracy	0.97	0.97	0.97	0.97
Average AUC	1.00	1.00	1.00	1.00
Average F1 Score	0.97	0.97	0.97	0.97
Average Precision	0.96	0.96	0.95	0.96
Average Recall	0.97	0.98	0.98	0.98

Image 22 – Performance Scores


Performance Metric
Average Accuracy
Average AUC
Average F1 Score
Average Precision
Average Recall
Performance Score
Number of Trees