1. Introduction

1.1 Problem Overview

1.1.1 Domain Background

This project was inspired by the Kaggle competition Customer Churn Prediction 2020. This competition is about predicting whether a customer will change telecommunications provider, something known as "churning". The reason why this is important is that the technical progress and the increasing number of operators globally have made the industry competitive [1]. Companies are working hard to survive in this competitive market depending on multiple strategies.

There are often three main strategies for generating more revenue within a business [2]:

1. Acquiring new customers
2. Upselling existing customers
3. Increase the retention period of customers

However, comparing these strategies taking the value of return on investment (RoI) of each into account has shown that the third strategy is the most profitable strategy. The reason being is that retaining an existing customer costs much lower than acquiring a new one, in addition to being considered much easier than the upselling strategy. To apply the third strategy, we need to decrease the potential of customer churn by putting systems in place to do so. Hence why exploring machine learning techniques for predicting customer churn can provide huge financial benefits to companies.

1.1.2 Related Work

Many approaches were applied to predict churn in telecom companies. Most of these approaches have used machine learning and data mining. The majority of related work focused on applying only one method of data mining to extract knowledge, and the others focused on comparing several strategies to predict churn.

Gavril et al. [3] presented an advanced methodology of data mining to predict churn for prepaid customers using a dataset for call details of 3333 customers with 21 features, and a dependent churn parameter with two values: Yes/No. The author used AUC to measure the performance of the algorithms. The AUC values were 99.10%, 99.55% and 99.70% for Bayes Networks, Neural networks and support vector machine, respectively.

He et al. [4] proposed a model for prediction based on the Neural Network algorithm in order to solve the problem of customer churn in a large Chinese telecom company which contains about 5.23 million customers. The prediction accuracy standard was the overall accuracy rate, and reached 91.1%.

Idris [5] proposed an approach based on genetic programming with AdaBoost to model the churn problem in telecommunications. The model was tested on two standard data sets. One by Orange Telecom and the other by cell2cell, with 89% accuracy for the cell2cell dataset and 63% for the other one.

Huang et al. [6] studied the problem of customer churn in the big data platform. The goal of the researchers was to prove that big data greatly enhances the process of predicting churn depending on the volume, variety, and velocity of the data. Dealing with data from the Operation Support department and Business Support department at China's largest telecommunications company needed a big data platform to engineer the fractures. Random Forest algorithm was used and evaluated using AUC.

Ahmad et al. [7] investigated the churn problem by developing a model using machine learning techniques on a big data platform and building a new way of feature engineering and selection. Their study measured the performance of the model using the AUC and obtained an AUC value of 93.3% after applying the XGBOOST algorithm.

Lalwani et al. [8] tacked the churn prediction problem by using a gravitational search algorithm for feature selection while several machine learning models were applied, namely, logistic regression, naive bayes, support vector machine, random forest, decision trees. The highest AUC score of 84% was achieved by both Adaboost and XGboost classifiers.

1.1.3 Datasets and Inputs

The training dataset contains 4250 samples. Each sample contains 19 features and 1 boolean variable churn which indicates the class of the sample. The 19 input features and 1 target variable are:

File Descriptions

All of these files can be found in dataset folder within the submission:

- **train.csv** - the training set. Contains 4250 lines with 20 columns. 3652 samples (85.93%) belong to class churn=no and 598 samples (14.07%) belong to class churn=*yes*
- **test.csv** - the test set. Contains 750 lines with 20 columns: the index of each sample and the 19 features (missing the target variable churn).
- **sampleSubmission.csv** - a sample submission file in the correct format

Data Fields

1. state, *string*. 2-letter code of the US state of customer residence
2. account_length, *numerical*. Number of months the customer has been with the current telco provider
3. area_code, *string*=area_code_AAA where AAA = 3 digit area code.
4. international_plan, *string*,(yes/no). The customer has international plan.
5. voice_mail_plan, *string*, (yes/no). The customer has voice mail plan.
6. number_vmail_messages, *numerical*. Number of voice-mail messages.
7. total_day_minutes, *numerical*. Total minutes of day calls.
8. total_day_calls, *numerical*. Total minutes of day calls.
9. total_day_charge, *numerical*. Total charge of day calls.
10. total_eve_minutes, *numerical*. Total minutes of evening calls.
11. total_eve_calls, *numerical*. Total number of evening calls.
12. total_eve_charge, *numerical*•. Total charge of evening calls.
13. total_night_minutes, *numerical*. Total minutes of night calls.
14. total_night_calls, *numerical*. Total number of night calls.
15. total_night_charge, *numerical*. Total charge of night calls.
16. total_intl_minutes, *numerical*. Total minutes of international calls.
17. total_intl_calls, *numerical*. Total number of international calls.
18. total_intl_charge, *numerical*. Total charge of international calls
19. number_customer_service_calls, *numerical*. Number of calls to customer service
20. churn, *categorical*, (yes/no). Customer churn - target variable.

1.2 Problem statement

1.2.1 Problem

Customer churn is a major problem and one of the most important concerns for large companies. Due to the impact customer churn has on the revenues of companies it is important to develop means to predict customer churn.

Customer churn is a common problem for businesses that provide subscription services. It is often difficult to determine when exactly a customer is likely to churn, due to the fact that it could be caused by several reasons.

The challenge is to predict whether a customer will churn or not using 19 input features defined in the dataset. The area under the curve (AUC) score of the model will be the main metric for determining its success.

1.2.2 Solution

With the advancements in artificial intelligence, the possibility to predict customer churn has increased significantly.

Having looked at the current literature, most research has been focused on using single estimators for making predictions. This project intends on taking an alternative approach by building an ensemble voting classifier comprised of 3 models that have the highest AUC.

The models that I used in the project were:

Logistic Regression (benchmark):
This is a statistical model that models the probability of an event taking place by having the log-odds of the event be a linear combination of one or more independent variables. [10]

LightGBM:
LightGBM is an implementation of the gradient-boosted trees algorithm that adds two novel techniques for improved efficiency and scalability: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). [11]

CatBoost:
CatBoost is an implementation of the gradient-boosted trees algorithm that introduces ordered boosting and an innovative algorithm for processing categorical features. [12]

XGBoost:
XGBoost is an implementation of the gradient-boosted trees algorithm that combines an ensemble of estimates from a set of simpler and weaker models. [13]

Random Forests:
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. [14]

Ensemble Voting Classifier:
Ensemble voting classifiers this is a meta-classifier for combining similar or conceptually different machine learning classifiers for classification via majority or plurality voting. [15]

1.2.3 Solution workflow:

The workflow for approaching a solution:

1. **Data Analysis:** understand the datasets
2. **Features Transformation:** convert variables into features. Standardize/normalize features, apply numerical transformations
3. **Features Selection:** select relevant features
4. **Machine Learning Models:** Train different models and perform hyperparameter tuning
5. **Evaluation:** evaluate the performance of each model, and check the possibilities of combining them to achieve an optimal model
6. **Deployment:** deploy the trained model to a streamlit [16] web app

1.3 Evaluation metrics

After a quick look at the dataset, I noticed that the dataset was imbalanced. Moreover, for business reasons, the model must be able to predict each class very well, therefore the Area Under the Curve (AUC) score will be used for evaluating the model. Additionally, the decision tree diagram below further justifies why the AUC score would be the most relevant metric to use to evaluate the model because we will need probabilities to determine the propensity to churn and both classes are equally important. AUC score has also been used for churn prediction analysis in other studies [8][7].

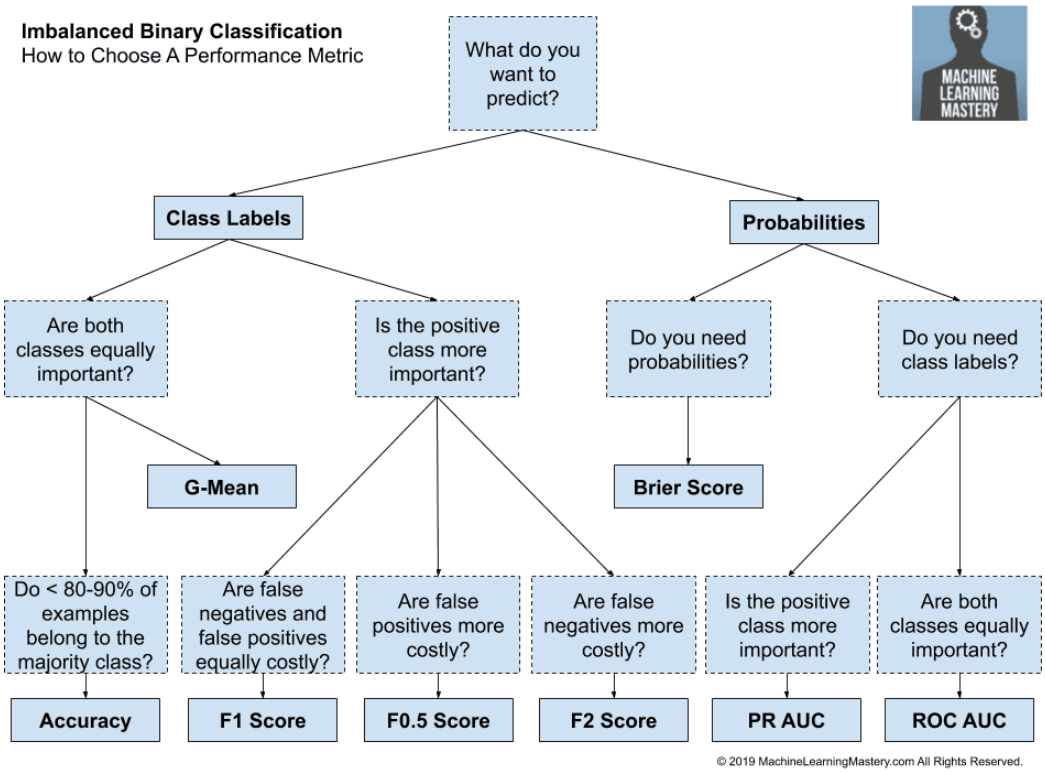

Figure 1: Decision tree diagram for choosing a binary classification performance metric. Image source [9]

2. Method

2.1 Data Pre-processing

2.1.1 Loading and cleaning the dataset

As I loaded the dataset, I checked for any null values and inconsistent data types. Upon looking at the dataset, one can see that each column has an equal amount of non-null values, indicating that there are no instances of missing data. However, the data types in the dataset are of type object, int64, and float64 indicating varying data types and a mixture of categorical and numerical variables. Lastly, I checked for duplicate values and did not find any.

2.1.2 Creating new input features

Upon looking at the dataset, I created additional features total_minutes, total_calls and total charges to observe their correlations with churn and their impact on the model's performance.

Next, I one-hot encoded the categorical features to make all the features numerical so that I could create a correlation matrix of all features. Lastly, I label encoded the target variables where 0 = 'not churned' and 1 = 'churned'.

2.1.3 Train/Validation Split

I created a validation set in order to evaluate the performance of the model and to see how well the model could generalise on unseen data. The training and validation sets were split within an 80/20 split in a stratified fashion.

2.2 Analysing and visualising the dataset

2.2.1 Visualising the class distribution

I first visualised the class distribution to have a better understanding of the split between churned and not churned. From figure 2, we can see that 85.94% of the dataset was classified as not churned, indicating that the dataset was highly imbalanced. Figure 2: Class Frequency distribution of the Churn Dataset

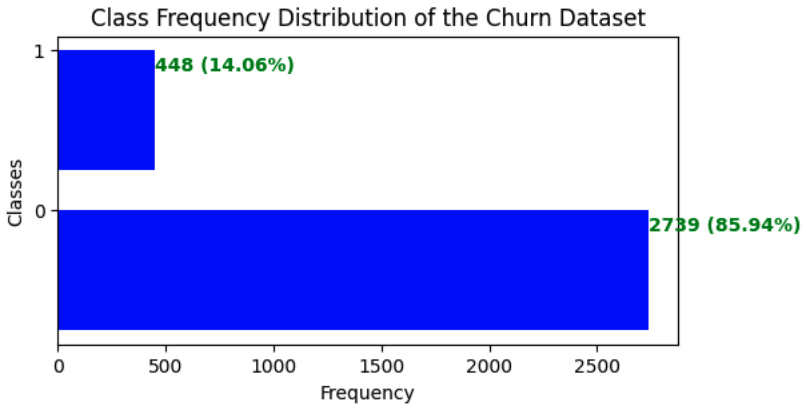

Figure 2: Class Frequency distribution of the Churn Dataset

2.2.2 Visualising the distributions of the numerical features

Most of the features tend to form a normal distribution. However, number_service_calls, total international calls, number_vmail_messages are more right-skewed than others.

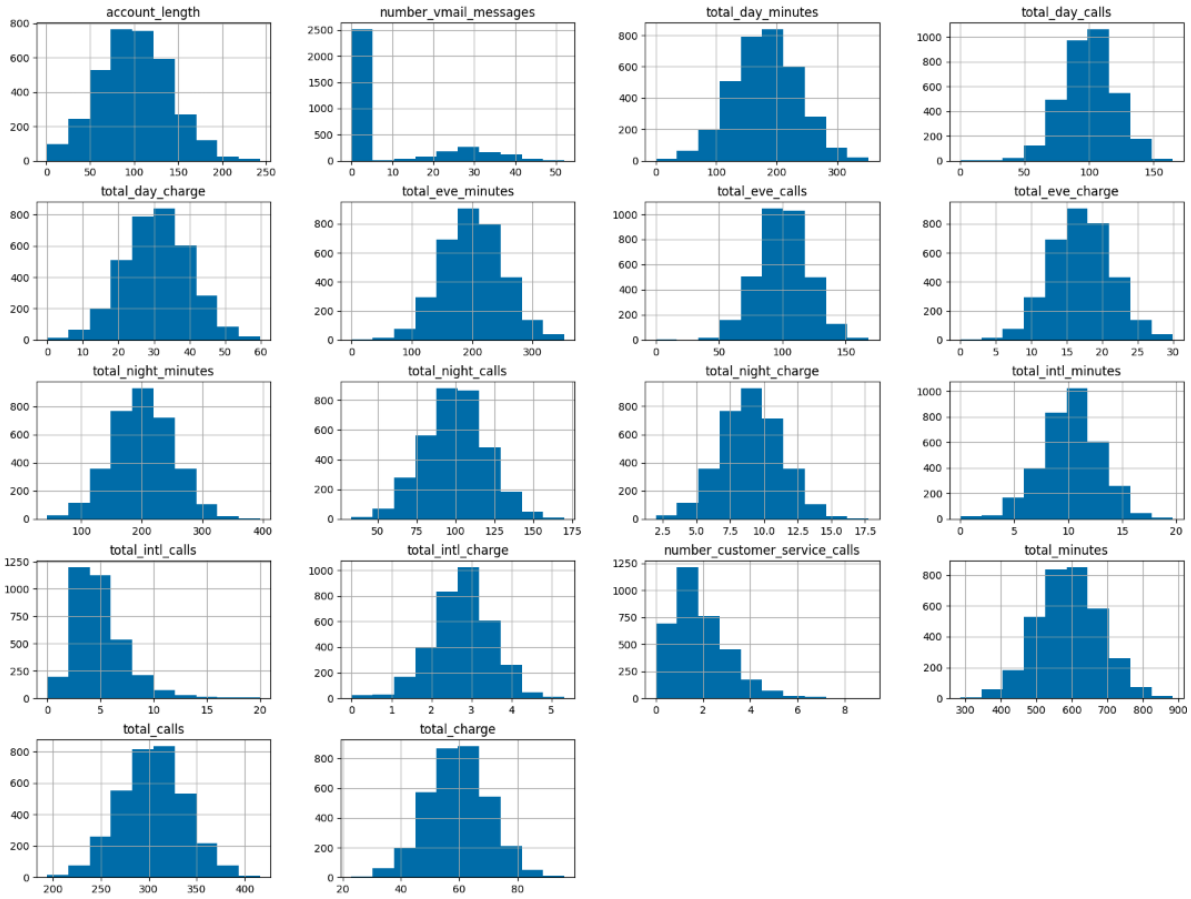

Figure 3: Histograms of the numerical features within the dataset

2.2.3 Correlation Matrix

To have a better understanding of the relationship between the features and the target variable I created two correlation matrices.

One for the top 10 features with a positive correlation with churn and another for the top 10 features with a negative correlation with churn. As we can see from figure 4 below, "international_plan_yes" is the feature with the highest correlation with churn followed by "total_charge". This is not surprising as it could be expected that an increase in charges could cause a customer to switch to another provider that could offer a better deal.

Moreover, total_day_minutes, total_day_charge and total_minutes all have a high correlation with one another, thus suggesting multicollinearity. This could be problematic for a logistic regression model however I'll also be training other tree-based models so they shouldn't be too affected by this. On the other hand, I noticed that "voice_mail_plan_yes" had the lowest correlation with churn, followed by number_of_vmail messages. Not sure why that would be the case but interesting to observe.

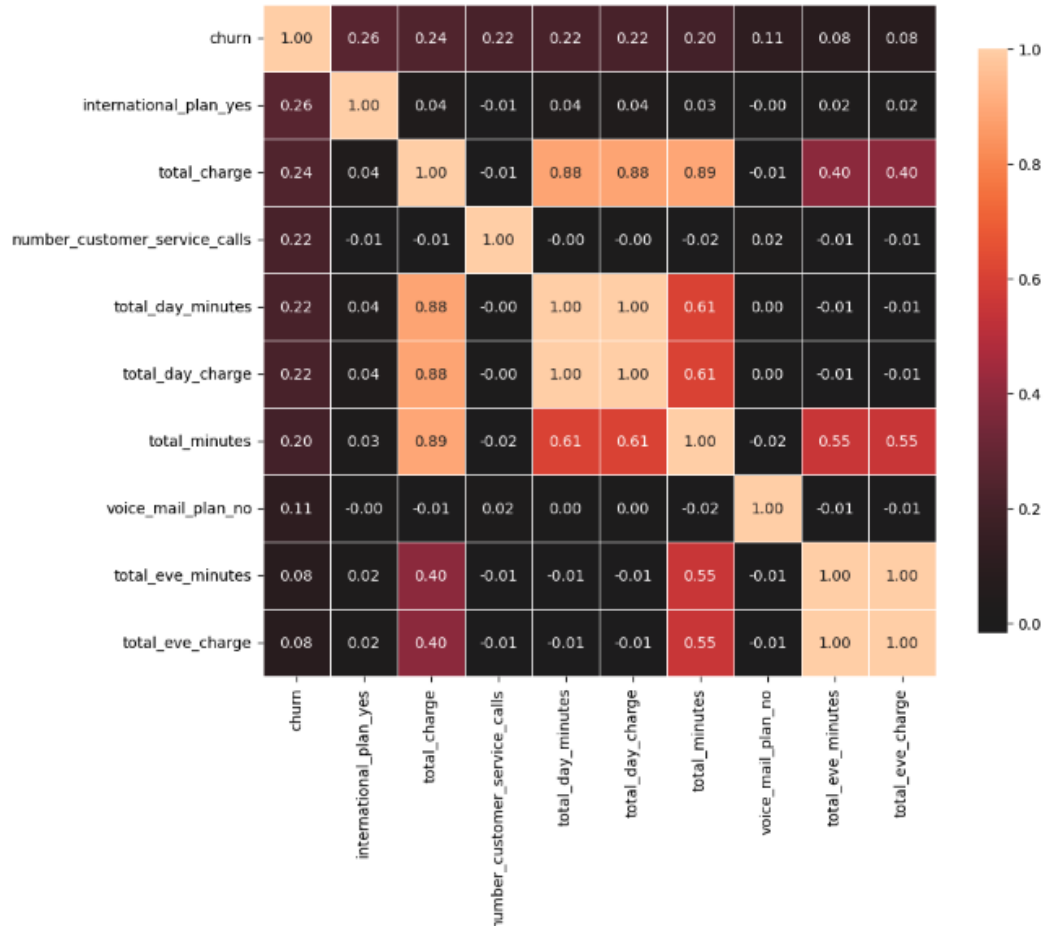

Figure 4: Top 10 features with a positive correlation with churn

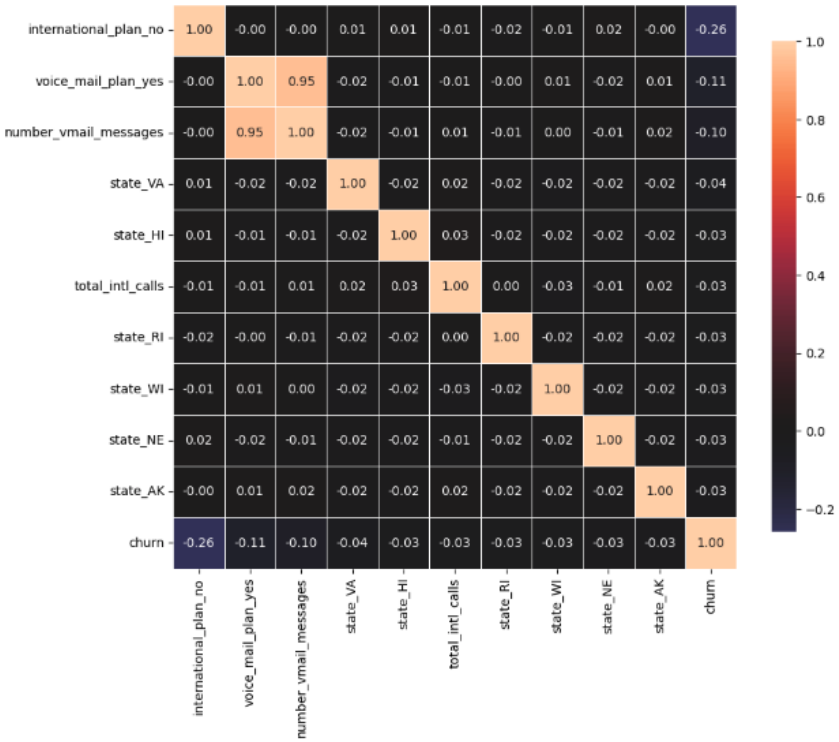

Figure 5: Top 10 features that have a negative correlation with churn

2.2.4 Pair Plot

Visualising the pair plot was very useful for having a good understanding of the pairwise relationship between the numerical features. We can see that evening_minutes and evening_charge features have a very strong positive correlation. Moreover, as seen earlier, all the numerical features tend to form a normal distribution

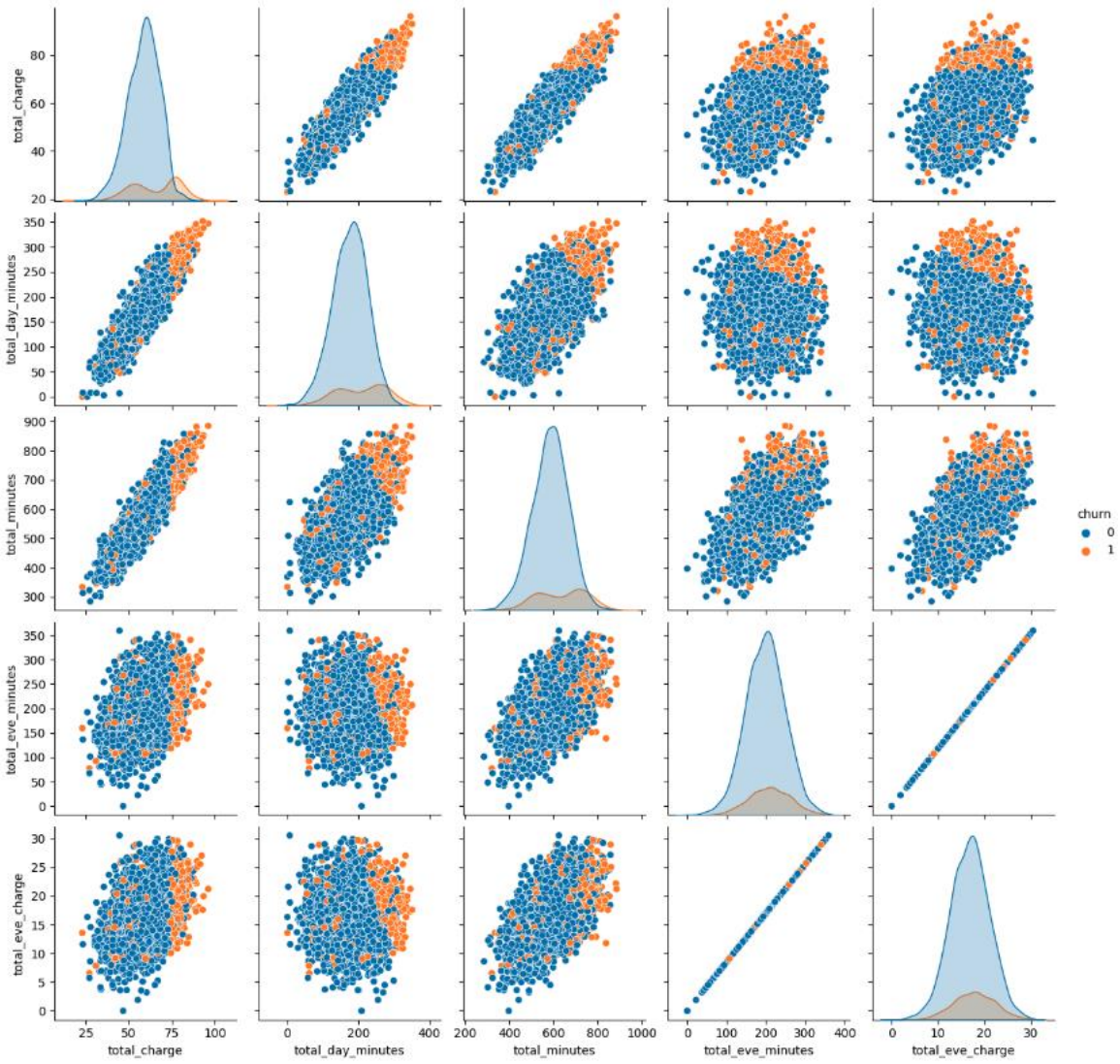

Figure 6: Pairplot of all the numerical features vs churn. The diagonal plots are a univariate distribution plot drawn to show the marginal distribution of the data in each column

2.3 Feature selection

In order to pick the best features, I created 3 different datasets and then trained a model using 5-fold cross-validation to find the best dataset to use for hyperparameter tuning.

The datasets that I created were:

1. Full dataset – all features
2. Top 10 features that correlated with churn
3. Recursive Feature Elimination (RFE)

Additionally, RFE is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the model's **coef_** or **feature_importances_** attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model. [17]

In order to optimise RFE, I chose different base models to implement the feature selection and then trained and computed the AUC scores for the different datasets using the LightGBM model. I found that the XGBoost model had the highest AUC score on average, with a score of 94%. (see Figure 7).

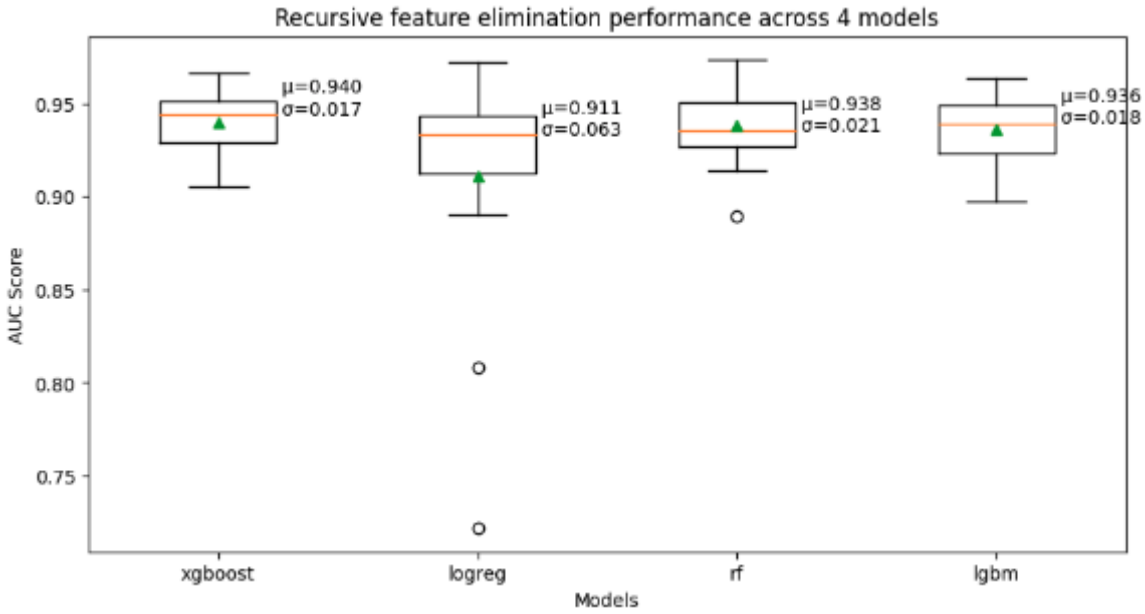

Figure 7: Boxplots of the AUC scores of the 4 base models that were used for RFE optimisation

Moreover, to finally select the features, I used XGBoost as the base model and found that the seven features it selected were:

1. Number_vmail_messages
2. Total_intl_minutes
3. Total_intl_calls
4. Number_customer_service_calls
5. Total_charge
6. Area_code_area_code_415
7. International_plan_no

2. 4 Train the models

2.4.1 Find the top-performing dataset

In order to find the top-performing dataset, I trained a LightGBM model with default hyperparameters and evaluated the AUC scores with 5-fold cross-validation (see Figure 8) and found that the RFE dataset had the highest average AUC score of 93.8%.

Figure 8: Box plot showing LightGBM's training performance across the 3 datasets with 5 fold-cross validation

2.4.2 Find the top-performing model

Table 1: Table showing the AUC scores of the six machine learning modles trained on the RFE datasets with 5-fold cross-validation

Model	Mean AUC Score	Standard deviation
Logistic Regression (bench mark)	0.797	0.026
LightGBM	0.938	0.022
XGBoost	0.937	0.021
Random Forests	0.937	0.021
CatBoost	0.936	0.022
Ensembling Voting Classifier	0.934	0.023

Next, I trained and scored six models with their default hyperparameters using the RFE dataset to find the top-performing model. Interestingly, the LightGBM model was the top-performing model while Logistic regression was the least-performing model.

2.4.3 Hyperparameter tuning of top models

For hyperparameter tuning the models, I used Bayesian search. Bayesian search works by constructing a posterior distribution of functions (gaussian process) that best describes the function you want to optimize. As the number of observations grows, the posterior distribution improves, and the algorithm becomes more certain of which regions in parameter space are worth exploring and which are not, as seen in the picture below. [18]

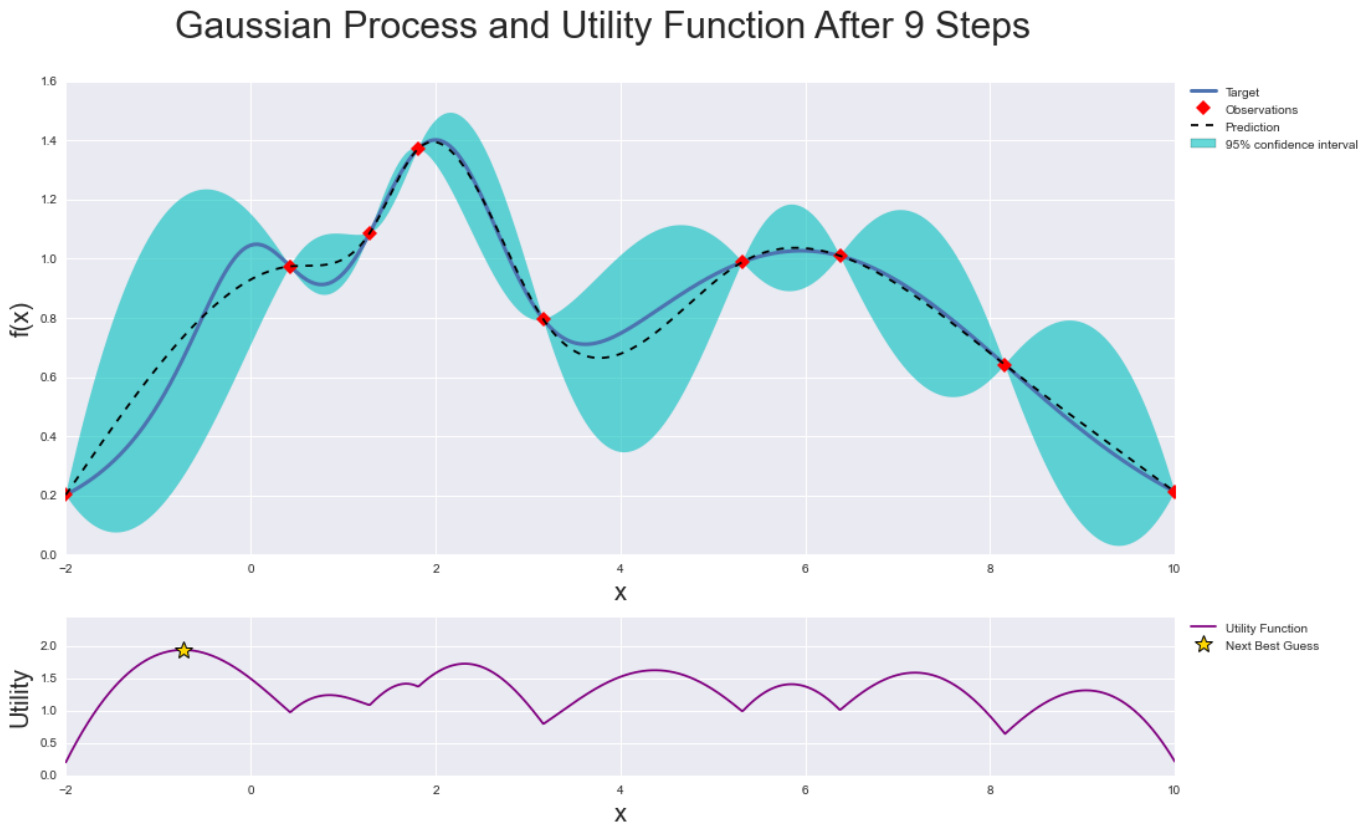

Figure 9: Image showing an example of the bayesian search process after 9 steps

I mainly focused on tuning the tree-based models like LightGBM, Random Forest, XGBoost and CatBoost. I set the Bayesian search to stop after 60 iterations and ensured that it evaluated the fits with 5-fold cross-validation. Below is a screenshot of the search spaces created for the hyperparameter tuning:

```
# Setting the search space
lgbm_search_spaces = {
    'learning_rate': Real(0.01, 1.0, 'log-uniform'),      # Boosting learning rate
    'n_estimators': Integer(30, 5000),                    # Number of boosted trees to fit
    'num_leaves': Integer(2, 512),                        # Maximum tree leaves for base learners
    'max_depth': Integer(-1, 256),                        # Maximum tree depth for base learners, <=0 means no limit
    'subsample': Real(0.01, 1.0, 'uniform'),              # Subsample ratio of the training instance
    'subsample_freq': Integer(1, 10),                     # Frequency of subsample, <=0 means no enable
    'colsample_bytree': Real(0.01, 1.0, 'uniform'),       # Subsample ratio of columns when constructing each tree
    'reg_lambda': Real(1e-9, 100.0, 'log-uniform'),       # L2 regularization
    'reg_alpha': Real(1e-9, 100.0, 'log-uniform'),        # L1 regularization
   }

rf_search_spaces = {
    'bootstrap': Categorical([True, False]),              # Method of selecting samples for training each tree
    'max_depth': Integer(1, 200),                         # Maximum number of levels in tree
    'max_features': Categorical(['auto', 'sqrt']),        # Number of features to consider at every split
    'min_samples_leaf': Integer(1, 5),                    # Minimum number of samples required at each leaf node
    'min_samples_split': Integer(2, 10),                  # Minimum number of samples required to split a node
    'n_estimators': Integer(200, 2000)}                   # Number of trees in the random forest

xgb_search_spaces = {
    'learning_rate': Real(0.01, 1.0, 'uniform'),
     'max_depth': Integer(2, 12),
     'subsample': Real(0.1, 1.0, 'uniform'),
     'colsample_bytree': Real(0.1, 1.0, 'uniform'), # subsample ratio of columns by tree
     'reg_lambda': Real(1e-9, 100., 'uniform'), # L2 regularization
     'reg_alpha': Real(1e-9, 100., 'uniform'), # L1 regularization
     'n_estimators': Integer(50, 5000)}

cat_search_spaces = {
    'iterations': Integer(10, 2000),
    'depth': Integer(1, 12),
    'learning_rate': Real(0.01, 1.0, 'log-uniform'),
    'random_strength': Real(1e-9, 10, 'log-uniform'), # randomness for scoring splits
    'bagging_temperature': Real(0.0, 1.0), # settings of the Bayesian bootstrap
    'l2_leaf_reg': Integer(2, 100), # L2 regularization
   }
```

Figure 10: Bayesian spaces for the 4 machine learning models

3. Evaluation of the validation set

Once all the models were finely tuned, I combined them to form the ensemble voting classifier and compared their performances.

From Table 2, we can see that the Random Forests model was the top-performing model with an AUC score of 0.916. Although the score is slightly lower than that seen on the training set, the models are able to still generalise pretty well. The best model also outperforms the benchmark logistic regression model.

Model	AUC Score	Accuracy
Logistic Regression (benchmark)	0.797	0.88
LightGBM	0.900	0.97
XGBoost	0.909	0.96
Random Forests	0.916	0.97
CatBoost	0.900	0.97
Ensembling Voting Classifier	0.911	0.97

Table 2: Table showing the AUC scores and accuracies of the models after evaluating the validation set

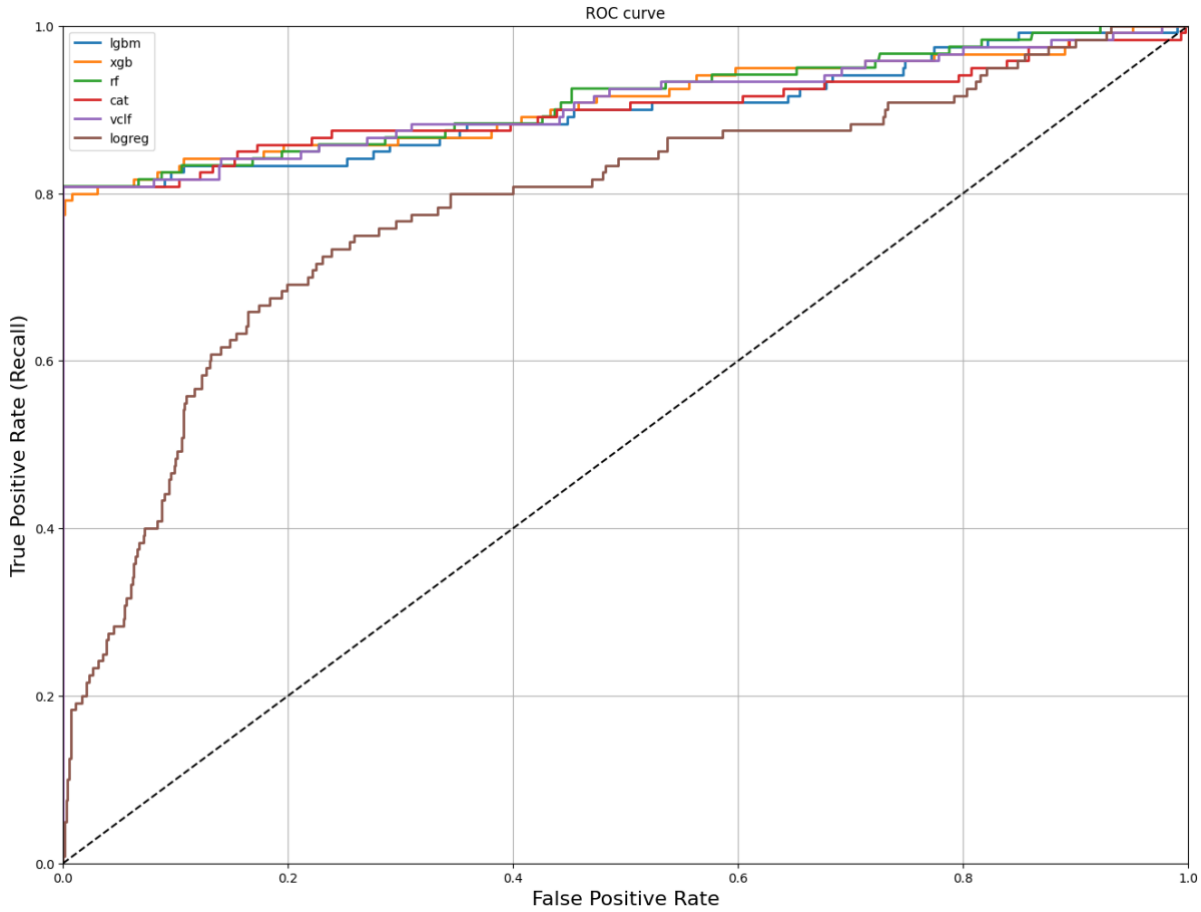

Figure 11: ROC Curve of the models after evaluating the validation set

I also plotted the ROC curve for all the models to have a better understanding of the trade-off between the True Positive rate and the False Positive rate. Noticed the benchmark logistic regression model appeared to be further away from the top left, indicating that it had the worst performance. Interestingly, the Random Forests model (green), appears to be the model that's closest to the top when the False Positive rate > 0.8.

4. Model Deployment on Streamlit

The model I chose for deployment was the finely tuned Random forest model due to its top performance in terms of AUC score.
I then saved the model in a pickle file and then built a web app using Streamlit so that any user could use the model for predictions. I also only used the features selected by recursive feature elimination to minimise the options needed for predictions.

Instructions on how to run the app can be found in the README.md.

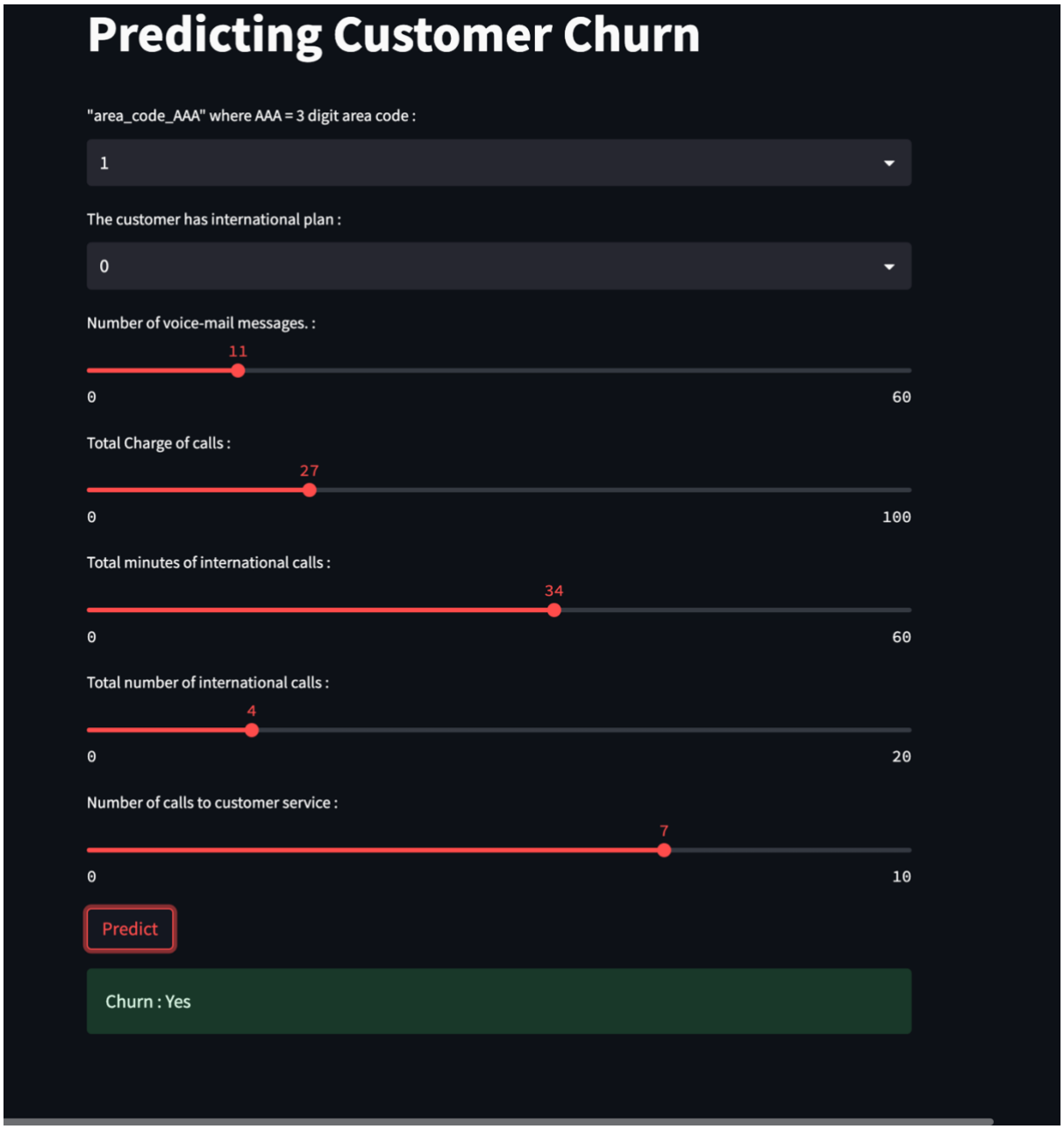

Figure 12: Screenshot of the UI built for the churn prediction model using streamlit

5. Conclusions and justification

This project focused on tackling the churn problem by building a classification machine-learning model that was able to predict churn with an accuracy of 97% and an AUC score of 91.6% on the validation set. The results indicate the final Random Forest model is able to generalise well and provide a lot of value for the telecommunication business given that cost acquisition costs are pretty high and retention is very important. Interestingly the ensemble voting classifier was not the top performer, thus highlighting that its ensemble method was outperformed by the bagging method seen in the random forests. Importantly, the final model beat the benchmark model in terms of performance and is considered an excellent model for industry standards, given that there was no solution in place before.

6. References

1. Gerpott TJ, Rams W, Schindler A. Customer retention, loyalty, and satisfaction in the German mobile cellular telecommunications market. Telecommun Policy. 2001;25:249–69

2. Wei CP, Chiu IT. Turning telecommunications call details to churn prediction: a data mining approach. Expert Syst Appl. 2002;23(2):103–12.

3. Brandusoiu I, Toderean G, Ha B. Methods for churn prediction in the prepaid mobile telecommunications industry. In: International conference on communications. 2016. p. 97–100.

4. He Y, He Z, Zhang D. A study on prediction of customer churn in fixed communication network based on data mining. In: Sixth international conference on fuzzy systems and knowledge discovery, vol. 1. 2009. p. 92–4.

5. Idris A, Khan A, Lee YS. Genetic programming and adaboosting based churn prediction for telecom. In: IEEE international conference on systems, man, and cybernetics (SMC). 2012. p. 1328–32.

6. Bingquan Huang, Mohand Tahar Kechadi, Brian Buckley, Customer churn prediction in telecommunications, Expert Systems with Applications, Volume 39, Issue 1,2012, Pages 1414-1425,

7. Ahmad, A.K., Jafar, A. & Aljoumaa, K. Customer churn prediction in telecom using machine learning in big data platform. J Big Data 6, 28 (2019)

8. Lalwani, P., Mishra, M.K., Chadha, J.S. et al. Customer churn prediction system: a machine learning approach. Computing 104, 271–294 (2022)

9. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

10. Sperandei S. Understanding logistic regression analysis. Biochem Med (Zagreb). 2014 Feb 15;24(1):12-8. doi: 10.11613/BM.2014.003. PMID: 24627710; PMCID: PMC3936971.

11. Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems* 30 (2017).

12. Dorogush, Anna Veronika, Vasily Ershov, and Andrey Gulin. "CatBoost: gradient boosting with categorical features support." *arXiv preprint arXiv:1810.11363* (2018).

13. Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.

14. Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.

15. Dietterich, Thomas G. "Ensemble methods in machine learning." *International workshop on multiple classifier systems*. Springer, Berlin, Heidelberg, 2000.

16. https://streamlit.io/

17. Chen, Xue-wen, and Jong Cheol Jeong. "Enhanced recursive feature elimination." *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. IEEE, 2007.

18. Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." Advances in neural information processing systems 25 (2012).