To Churn or Not to Churn: Predicting Bank Customer Behavior Through Machine Learning Algorithms

Kuan-Chen Liu Mallika Chandra Rhea Sethi

***Abstract*—This paper aims to address the issue of bank churn by implementing a predictive model that can be used for customer retention strategy. The selected sample includes data from 10,000 customers. Seven models have been selected for comparison, including K-Nearest Neighbor, Logistic Regression, Neural Network, Naïve Bayes, Support Vector Machine, Random Forest, and a stacked model. The performance of these models is assessed using metrics such as accuracy score, AUC, and recall rate. The results show that the Random Forest has the highest accuracy score at 86.2% and the highest recall rate at 48%. The Feedforward Neural Network attains the highest AUC at 0.85. We conclude that the Random Forest is the best performing model out of the seven. The model then achieves an accuracy score of 86.04%, a recall rate of 46%, and an AUC of 0.711 for the testing set. Nevertheless, misclassifications are seen in corner cases for outliers, i.e., customers with feature values atypical of their true class. The study offers the potential of Random Forest Classifier model to generate insights into customer churn and the development of customer retention strategies for banks.**

I. Introduction

The banking and finance sectors are undergoing rapid advancements in the current economic landscape. In the past, customer churn rates were relatively low, but with the increased competition in the banking industry, compounded with the boom of fintech, customer churn has become an increasingly significant concern for financial institutions. In light of these challenges, it has become increasingly crucial for financial institutions to not only comprehend but also effectively manage customer churn rates. Additionally, acquiring new customers can be costlier and more time-consuming than retaining present customers. Therefore, through our paper we aim to provide a predictive modelling algorithm for banks to narrow down upon the characteristics of the customers who have a propensity to churn, and enable the banks to employ potential strategies to retain them.

II. Data

A. Data Source

The dataset used for this paper is sourced from Kaggle, which comprises information of 10,000 customers and encompasses 10 unique features. The customer data consists of both demographic and engagement features, such as gender, age, estimated salary, credit score, geography, tenure, balance, number of products used, customer's credit card ownership status with the bank, and the customer's active membership status. The dataset was carefully selected to enable an in-depth analysis of the customer behavior and characteristics in the context of the banking industry.

B. Exploratory Analysis

Table 1 presents summary statistics of the continuous variables, providing an overview of their distribution. The age range is broad, covering individuals from 18 to 92 years old, which gives a good representative sample of the population. It is worth noting that the mean of balance, $76,485.89, is significantly lower than the median of $97,198.54 due to 3617 accounts with a 0 balance. These inactive accounts may offer insights into customer churn. The average number of products used is low, at 1.53, which indicates that the sample consists of individuals who do not have extensive banking or financial needs.

The correlation heatmap presented in Figure 1 illustrates the relationships between the features. The results indicate that Age has the strongest correlation with account closure, with a coefficient of 0.29. On the other hand, Tenure, HasCrCard, and

TABLE I: Summary Statistics

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
count	10000	10000	10000	10000	10000	10000
mean	650.5288	38.9218	5.0128	76485.89	1.5302	100090.2
std	96.6533	10.48781	2.892174	62397.41	0.581654	57510.49
min	350	18	0	0	1	11.58
25%	584	32	3	0	1	51002.11
50%	652	37	5	97198.54	1	100193.9
75%	718	44	7	127644.2	2	149388.2
max	850	92	10	250898.1	4	199992.5

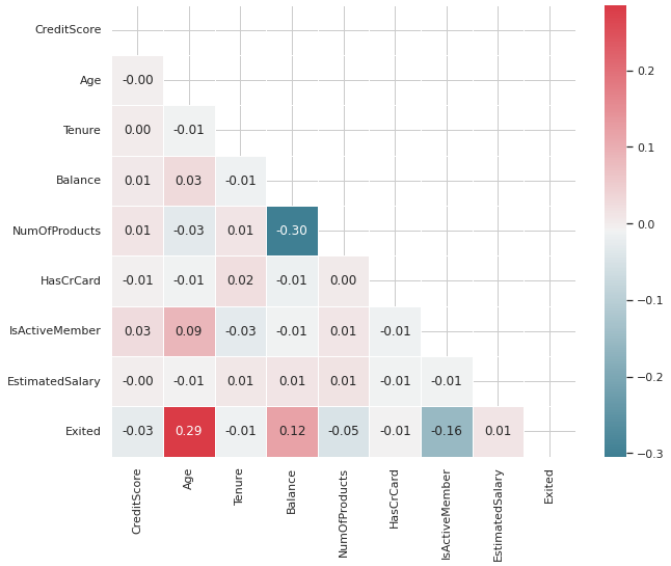


Fig. 1: Correlation heatmap

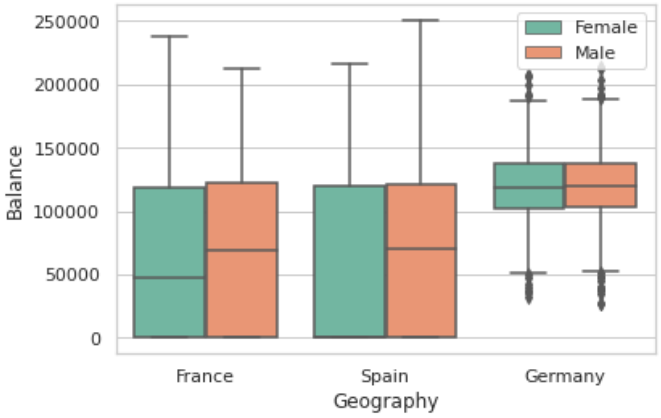


Fig. 2: Box plots with categorical variables

EstimatedSalary display the weakest correlations, with coefficients of 0.01. Notably, the number of products a customer owns and their balance exhibit a significantly negative correlation.

In Figure 2, a detailed analysis is presented on the relationship between the categorical variables Geography and Gender with Balance. The results reveal that Germany displays a smaller interquartile range, as well as a significantly higher median Balance value, in comparison to France and Spain. It is observed that the median Balance value of male and female in Germany is similar, while France and Spain exhibit a noticeable disparity in the median Balance value between the two genders, with male Balance being higher than female Balance. Unexpectedly, the median Balance value of Spanish females is recorded as 0.

C. Data Preprocessing

For our research, we employ the following preprocessing strategies:

1) We use **dummy variables** to represent categorical variables, such as Gender and Geography, instead of encoding them as numerical values. This is done to avoid the possibility of the machine learning algorithm interpreting the numerical values as having a specific order or ranking, which could result in biased predictions.
2) We apply **min-max scaling** to ensure that each feature has a similar scale.
   Overall, these preprocessing techniques are critical in preparing our dataset for effective machine learning model training.
3) Additionally, the **data is split into training, validation, and testing sets** with a ratio of 50:25:25 to evaluate the performance of different machine learning models and to prevent overfitting. This ensures that our models are

generalizable and can perform well on new, unseen data.

III. Model Evaluation

For the classification problem, seven machine learning models are chosen: K-Nearest Neighbor, Logistic Regression, Neural Network, Naïve Bayes, Support Vector Machine, and two ensemble models - Random Forest Classifier and a stacked model of several heterogeneous weak learners.

A. K-Nearest Neighbours

K-Nearest Neighbours is a non-parametric algorithm used for classification tasks. Given a dataset of labeled examples, the algorithm predicts the class label of a new example by finding the k nearest neighbors in the feature space and assigning the most frequent class among them as the label of the new example.

K here is an important hyperparameter that affects the performance of the algorithm. We first fit a KNN model with a random value of K = 3 to get a baseline model which gives an accuracy score of 80.64% on the validation set. Then, two approaches were employed to optimize this hyperparameter. First, using a grid search, the optimal value was found to be 15. However, the ”Elbow Method” suggests accuracy does not improve by much after for values of K greater than 9, as shown in the figure. Models are fit on the training set with both these values and performances compared over the validation set. K = 9 yields a slightly higher accuracy score of 81.64% than K = 15 (81.24%). Both these models outperform the baseline.

The performance of this model is further evaluated using a Confusion Matrix. We also calculate the Precision and Recall. Given our business context, predicting customers who are going to exit correctly is crucial. As such, False Negatives have a high cost. Therefore, we care more about the recall rate for the positive class and want to maximize this. For the KNN model, this is a low 22%. We also plot the ROC/AUC curve. The AUC for the model is an unimpressive 0.59, only slightly better than random guessing.

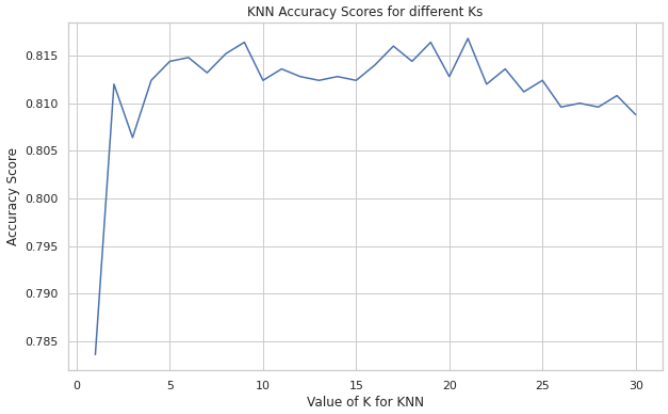
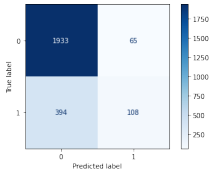
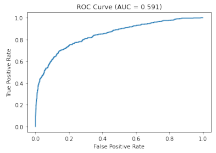


Fig. 3: Hyperparameter Tuning for KNN



(a) Confusion Matrix: K-Nearest Neighbours



(b) ROC Curve: K-Nearest Neighbours

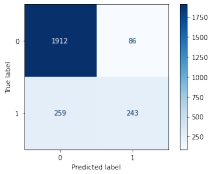
Fig. 4: K-Nearest Neighbours Results

B. Random Forest Classifier

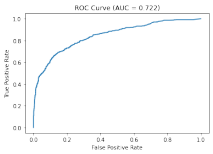
Random forest is an ensemble learning algorithm that builds a collection of decision trees using a random subset of features. It aggregates the predictions of all decision trees to output the class of a new data point. Random forest can handle high-dimensional and noisy data without overfitting, and identify important features.

This algorithm has several hyperparameters that can be optimized to improve performance. We first fit a Random Forest over the training data using the default values for all hyperparameters to use as a benchmark model. This baseline yields a good accuracy score of 85.72% over the validation set, but gives a perfect score of 100% over the training set indicating overfitting. Therefore we use grid search to tune 4 of the more important hyperparameters using 3 different values for each, resulting in a total of 405 fits in the process. We end up with optimal values of ’max depth’ at 230, ’minimum samples leaf’ at 1, ’minimum samples split’ at 3 and ’number of estimators’ at 100. We fit the model again using these values and evaluate the results.

This model has a higher validation accuracy of 86.2% and a slightly lower training accuracy of 99.84% than the baseline, suggesting higher gener-

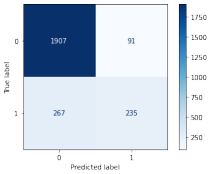


(a) Confusion Matrix: Random Forest

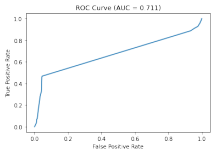


(b) ROC Curve: Random Forest

Fig. 5: Random Forest Results



(a) Confusion Matrix: Stacking



(b) ROC Curve: Stacking

Fig. 7: Stacking Results

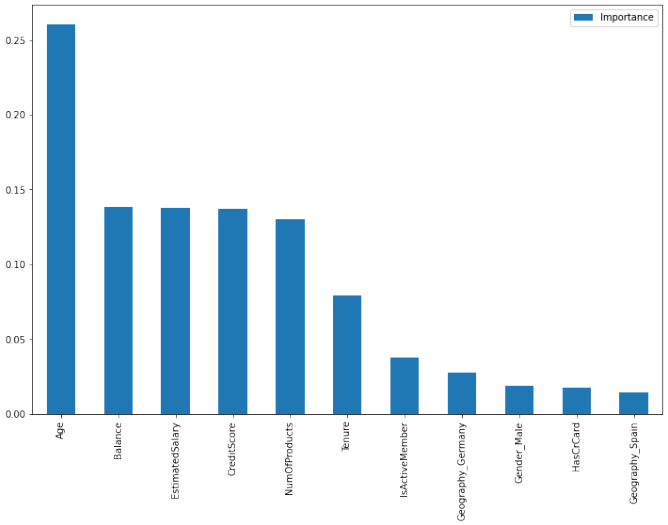


Fig. 6: Feature Importance for Random Forest

alization capabilities. From the Confusion Matrix, we can calculate the recall for the positive class to be 48%. The AUC is 72.2% as displayed. We also identify the features deemed to be the most important in this classification algorithm. "Age", "Balance" and "Estimated salary" are the top 3 predictors as shown in the figure, which also aligns with our intuition.

C. Ensemble Model: Stacking

Stacking is an ensemble machine learning technique that involves training multiple heterogeneous weak learners to make predictions on a dataset and then using these models' predictions as input features for a final meta-model which generates the final predictions. The idea behind stacking is that by combining multiple models, the final model can achieve better predictive performance than any individual model on its own.

For our paper, we use a set of 5 base learners - Logistic Regression, Random Forest Classifier, KNN, Neural Network (MLP Classifier) and Support Vector Machine. Since these are supposed to be 'weak' learners , we use the default values for their hyperparameters; stacking them together mitigates the weaknesses of these individual models and leverages their respective strengths. The base predictions are then fed into a Logistic Regression Meta Learner which outputs the final predictions. The coefficients of this Meta Learner tell us the weight it gives to each base model's predictions while making the final decision. Neural Network receives the highest weight with a coefficient of 8.55 for our model.

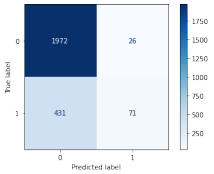
The validation accuracy for this ensemble model is 85.68%. From the Confusion Matrix, we can calculate the recall for the positive class to be 47%. The AUC is 71.1% as displayed.

D. Logistic Regression

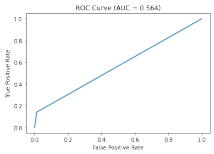
Logistic Regression is a machine learning algorithm used for classification by predicting the likelihood of an observation belonging to a particular class. The predicted probability of an outcome falling into one of the categories is calculated using the logistic function and then categorized by employing a threshold value (typically 0.5).

We use the **sklearn** library to carry out a logistic regression with the optimal hyperparameters found through GridSearch, that is, using an L2 regularization penalty, c = 0.1 and solver='lbfgs.'

After training the model on our training set, when we predict values for the validation set we observe that the model performs with an accuracy of 81.76%, indicating that the model correctly predicted the target variable for about 81.76% of the samples in the validation set. Figure 8(a) is confusion matrix we obtained through which we can infer the precision rate to be 73.2% and recall rate

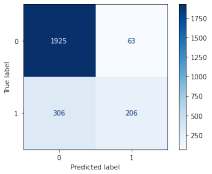


(a) Confusion Matrix: Logistic Regression

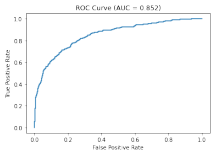


(b) AUC Curve: Logistic Regression

Fig. 8: Logistic Regression Results



(a) Confusion Matrix: Neural Network



(b) AUC Curve: Neural Network

Fig. 9: Neural Network Results

to be 14.1%. In Figure 8(b), the AUC values comes out to be 0.564. Hence, while the accuracy seemed to be high, the model's overall metric do not seem to be positive. The lower recall score indicates that the model is missing a lot of the positive cases. Lastly, the AUC curve suggests that the model is not able to discriminate well between positive and negative cases.

E. Feedforward Neural Network

A feedforward neural network is an artificial neural network that can learn complex relationships between inputs and outputs. The network receives the input data, passes it through a series of hidden layers, and produces an output. The network learns from the input-output pairs and can predict whether a customer will churn or not based on their characteristics.

We conducted a grid search in the neural network to find the optimal hyperparameters for the MLP classifier. The hyperparameters that I tested included hidden layer sizes with values of [(5,), (10,), (5,5), (10,10)], activation functions with options of logistic, tanh, and ReLu, solver with options of 'adam' and 'sgd', learning rate with values of 0.001, 0.01, and 0.1, and the maximum iterations with values of 100 and 500. This allowed me to systematically search for the best combination of hyperparameters to improve the performance of the classifier.

Our optimal model comes out to be a sequential neural network model with two layers where the first layer has 5 neurons, uses the sigmoid activation function, and includes a L2 regularization term. The second layer has a single neuron with linear activation. The model is compiled with the Adam optimizer and binary cross-entropy loss function. The model is then trained on the training data for 100 epochs.

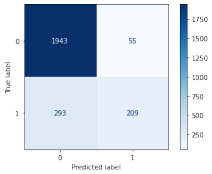
The neural network model performed well in the evaluation metric as is visible in Figure 9(a), achieving an accuracy score of 0.852, which indicates that it correctly classified a majority of instances. The precision score of 0.766 suggests that the model accurately identified positive instances almost 77% of the time, while the recall score of 0.402 indicates that the model still correctly identified over 40% of all positive instances in the test set.

Moreover, the AUC score of 0.852 in Figure 9(b) suggests that the model performs well in separating positive and negative classes, indicating that it is good at distinguishing between the two classes. Overall, these results demonstrate that the neural network model is capable of accurately classifying instances and has the potential to be used for various classification tasks.

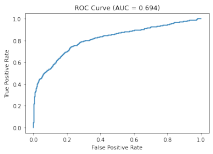
F. Support Vector Machine

The SVM algorithm aims to find a hyperplane that separates data points into distinct classes, with the goal of maximizing the margin distance between the classes. A kernel is used to transform linearly inseparable data into separable data that enables a boundary to be found. Advantages of SVM include its effectiveness in high-dimensional spaces and the robustness to outliers. On the other hand, one of the drawbacks of SVM is that it may not be appropriate for datasets with a high level of noise or target classes that overlap.

The Kernel SVM model is fine-tuned using a grid search to identify the optimal set of hyperparameters. The regularization parameter, C, is tested at several values: 2, 4, 6, 10, 100, and 1000. This parameter determines the degree to which misclassification is avoided. Gamma is tested over a range of values: 0.01, 0.1, 1, and 10. It selects the degree of curvature for the decision border. Lastly,

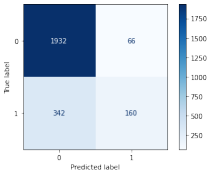


(a) Confusion Matrix: Kernel SVM

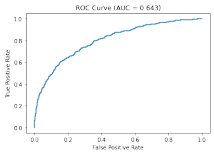


(b) ROC Curve: Kernel SVM

Fig. 10: Kernel SVM Results



(a) Confusion Matrix: Naive Bayes



(b) ROC Curve: Naive Bayes

Fig. 11: Naive Bayes Results

the RBF kernel is chosen as it can transform the data into a higher-dimensional space where it is easier to separate, which is useful for real-world applications where the data is often not separable.

The grid search results indicate that the highest score is achieved by C=100 and gamma=0.1. The model with the fine-tuned hyperparameters is then tested on the validation set. The accuracy is found to be high at 0.861, but the AUC score is lackluster at 0.694. The inconsistency between the two metrics is a common matter when there is an unequal distribution of classes in the dataset. In this case, only 20% of the labels in the validation set are 1, which may cause the algorithms to lean towards predicting the majority class. The findings are in sync with the discrepancy between positive recall rate of 0.42 and negative recall rate of 0.97. Nevertheless, it is important to note that customer churn is a relatively uncommon event compared to customers staying.

G. Naive Bayes

Naive Bayes classifier is based on Bayes' Theorem, where every feature is independent of each other. The algorithm learns the probability of every object, its features, and which class it belongs to, making it a probabilistic classifier. The model is generally fast and can handle high-dimensional data well thanks to the assumption of independence. However, such assumption can also be a disadvantage since this is not always true in real-life scenarios.

Given that the dataset contains both categorical and continuous features. To ensure that the data is not restricted to a single distribution, a mixed Naive Bayes technique is used to implement Multinoulli and Gaussian distributions.

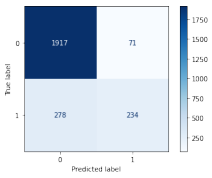
From the validation results, the model achieves a high accuracy of 0.837 with an AUC of 0.643. The false positive rate comes in at 0.033 which is considered good, as only 3.303% of the negative samples are falsely identified as positive.

IV. Results

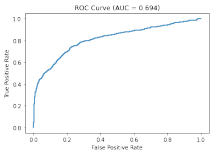
TABLE II: Model Performance

Model	Accuracy	AUC	Recall
K-Nearest Neighbors	81.64	0.59	22
Random Forest Classifier	86.2	0.722	48
Stacked Ensemble Model	85.68	0.711	47
Logistic Regression	81.76	0.564	14.1
Feedforward Neural Network	85.2	0.852	40.2
Support Vector Machine	86.1	0.694	42
Naïve Bayes	83.7	0.643	32

The table encapsulates the performance of all of our 7 models of interest over the validation set. The 3 most crucial metrics for our objective of identifying customer churn are - Accuracy Score, AUC and Recall for the positive class. Comparing these metrics across all contenders, we see that Random Forest has the highest accuracy. But, we don't solely rely on accuracy since this metric can be misleading when there is a class imbalance, which is present in our data. Feedforward Neural Network has the highest AUC, which means this model is the best at distinguishing between the two classes. Random Forest comes in second on this metric. For Recall, Random Forest again beats all the other models, meaning it is the best at predicting an exit for customers who actually exit, which is what we want to maximize (and thus minimize False Negatives). Since Random Forest tops 2 out of 3 metrics and Recall Rate holds more weight for our business problem than AUC, we conclude that Random forest is the best performing model for our dataset and objective. We fit this model on the testing set and evaluate its performance.



(a) Testing Set Confusion Matrix: Random Forest



(b) Testing Set ROC Curve: Random Forest

Fig. 12: Random Forest Testing Set Results

V. Best Performing Model: Implementation

A. Testing the Model

Fitting our best-performing model, the Random Forest classifier with the optimal hyperparameter values as obtained from the Grid Search, on the testing set, yields impressive results. The testing accuracy is 86.04%. The Confusion Matrix is calculated for a more comprehensive evaluation. From this, we infer a Recall rate of 46% for the positive class. The AUC is a reasonable 0.711. Overall, the testing set performance is only slightly worse than the validation set performance. This implies our model selection process was robust and our model of choice generalizes well to unseen data. We can, therefore, use this model for churn prediction and to derive insights into the customer base, and hopefully drive retention and thus, profits.

B. Corner Case Analysis

However, we also look into the reasons underlying the misclassifications so we can try and counter the fallacies of the model before implementation. We evaluate 5 corner cases by comparing their feature values with the mean values of each feature for each of the two classes. The intuition is that, if the model is misclassifying a sample, the sample may be exhibiting features more characteristic of the predicted class than its true class. In this sense, it is an outlier for its ground-truth label.

This hypothesis is confirmed in our corner-case analysis. Tables III and IV show the mean values by class and the values for corner cases respectively, for the top 4 most important features for the Random Forest model that we obtained from the model analysis in the previous section. Let's take the feature "Age" for an example, the most important feature. Table III shows that Class 0 has a slightly higher mean value for Age than Class 1. We see that most of the failure cases that were misclassified as negative when they were actually positive have higher values for "Age". On the other hand, the one sample misclassified as positive when it is actually negative has a low value for age, as do most of the samples under class 1. We can draw similar insights for other features.

TABLE III: Mean Feature Values

Actual	CreditScore	Age	Balance	EstdSalary
0	648.45	38.50	77936.98	100796.24
1	657.08	38.46	77460.76	109381.24

TABLE IV: Corner Case Analysis

CreditScore	Age	Balance	EstdSalary	Actual	Predicted
695	31.0	0	13998.88	0	1
845	52	0	31726.76	1	0
599	50	121159.65	4033.39	1	0
434	55	109339.17	96405.88	1	0
617	39	83348.89	7953.62	1	0

We can thus deduce that failure cases have more to do with the attributes of those samples than our model's predictive abilities. The model ingests samples' feature values and tries to align them with what is typical of both the classes. Mostly this works well. When samples themselves are outliers and more similar to the other class than their true class, the model may output the wrong result. This is a common problem in Machine Learning- data is volatile and models are reactive.

From this analysis, we conclude that we can rely on our model's results with a high degree of confidence most of the time. However, when dealing with outliers, we would need to supplement model predictions with human intuition and oversight.

VI. Conclusion and Discussion

Understanding the common characteristics of customers who have a propensity to churn from their current banks is crucial to study. Predictive modelling stands to be a strong methodology to investigate this as it provides results which can be extrapolated by the banks for their future customers as well. When employing the seven different machine learning algorithms, namely - K-Nearest

Neighbors, Random Forest Classifier, Stacking Ensemble Model, Logistic Regression, Feedforward Neural Network, Support Vector Machine, Naïve Bayes – we conclude that Random Forest Classifier is the most accurate and should be used to study environments akin to our dataset.

A. Challenges and Solution

During our study, we encountered the following challenges: a)

1) **Imbalanced Dataset:** The class distribution of our dataset was highly imbalanced due to the nature of the research question, with the majority class being the one where the target variable is 0, and only approximately 19% of the samples belonging to the other class where the target variable is 1. This caused problems with model performance evaluation metrics, In this case, because the model is predicting the majority class (0) for most of the samples, the accuracy score was misleadingly high, while the AUC score was very low. This is because AUC measures the model's ability to distinguish between the two classes, whereas accuracy only measures the proportion of correctly classified samples overall. One way to solve this challenge was to use a different evaluation metric which is more appropriate for imbalanced data, such as precision and recall, which is why employed those to further study the fit of the model. Another potential approach for future expansion of our study could be to address the class imbalance by using techniques such as undersampling, oversampling, or SMOTE.

2) **Strong Independence Assumption**: One of the challenges of using Naive Bayes is the strong independence assumption it makes–that each feature in the dataset is considered to be independent of all other features-which may not always hold true in real-world scenarios such as our dataset where the characteristics of custoemrs were correlated. This may have resulted in a loss of predictive power and suboptimal performance. In addition to our efforts for further analysis in our paper, feature engineering techniques can be employed to incorporate relevant interactions between features and improve the model's performance.

REFERENCES

1) Do, T. (2022, September 20). 5 types of classification algorithms in machine learning. Omdena. Retrieved March 7, 2023, from https://omdena.com/blog/machine-learning-classification-algorithms/
2) Molina, E. (2021, March 19). A practical guide to implementing a random forest classifier in Python. Medium. Retrieved March 10, 2023, from https://towardsdatascience.com/a-practical-guide-to-implementing-a-random-forest-classifier-in-python-979988d8a263
3) Sunkaraneni, T. (2018, March 20). Bank turnover dataset. Kaggle. Retrieved March 6, 2023, from https://www.kaggle.com/datasets/barelydedicated/bank-customer-churn-modeling

STATEMENTS OF CONTRIBUTIONS

1) Kuan-Chen Liu:
   - Abstract
   - II. Data:
     - A. Data Source
     - B. Exploratory Analysis
     - C. Data Preprocessing
   - III. F. Support Vector Machine
   - III. G. Naive Bayes
2) Mallika Chandra:
   - I. Introduction
   - III. D. Logistic Regression
   - III. E. Feedforward Neural Network
   - VI. Conclusion:
     - A. Challenges and Solution
3) Rhea Sethi:
   - III. A. K-Nearest Neighbors
   - III. B. Random Forest Classifier
   - III. C. Stacking Ensemble Model
   - IV. Results
   - V. Best Model's Implementation:
     - A. Testing the Model
     - B. Corner Case Analysis