Title: Prediction on Credit Card Consumer Churn

Kaggle Project:https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers
Notebook:https://www.kaggle.com/code/thomaskonstantin/bank-churn-data-exploration-and-churn-prediction

1. Project Introduction

Customer churn, or customer attrition, is a critical issue for businesses in the financial sector, especially those dealing with credit cards. Successfully predicting which customers are likely to close their accounts can help companies develop targeted strategies to retain customers and reduce losses. However, this is a challenging task due to the complex nature of consumer behavior, which is influenced by numerous factors.

Previous attempts, such as the Kaggle notebook by Thomas Konstantin, have employed different models like the Random Forest Classifier, AdaBoost Classifier, and Support Vector Machines (SVM) to predict churn. It focused on using different machine learning algorithms, with varying levels of success. However, these previous works often overlooked the importance of handling missing values in the dataset, which, as we discovered, can significantly impact the performance of the prediction model.

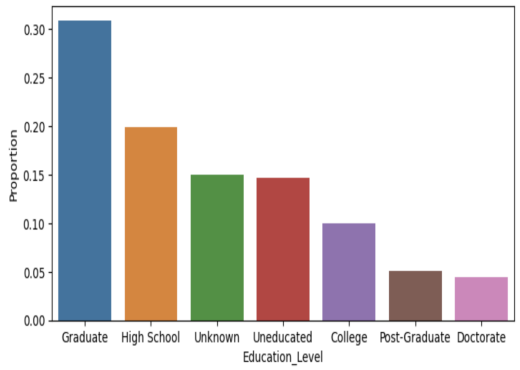
In this project, we explore the use of machine learning techniques to predict customer churn using a dataset of credit card customers. We used two different models - a Decision Tree Classifier and a K-Nearest Neighbors (KNN) Classifier - and two different imputation techniques to handle missing data (Nawale, 2022). Our goal is to determine which combination of model and imputation technique yields the best results.

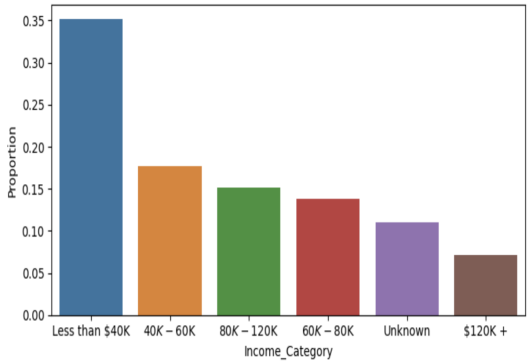
2. Project Approaches

Our approach to this problem was largely based on data preprocessing, exploratory data analysis (EDA), feature engineering, model building, and model evaluation.

Data Preprocessing and EDA:
First, we loaded the dataset using Pandas. This dataset comprised 23 features and 10127 records. We dropped irrelevant features such as "Marital_Status", "CLIENTNUM", and the "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count" related columns, as they do not provide meaningful insights for our analysis.

Subsequently, we conducted a preliminary analysis to determine the presence of any missing values. We discovered that our dataset had no missing data, which looks like an ideal scenario. However, when we took a closer look into the distribution of categorical variables, we found out that there are missing values, shown as “Unknown”, in variables “Education_Level” and “Income_Category”.





Missing Values Imputation:
These missing values are not detected or handled by the most voted Kaggle notebook. We decided to handle these missing values instead of removing them, as they are important and relevant variables and the missing values take up more than 10% in each variable. We applied KNN imputer and mode imputer to fill the missing values after label encoding all the categorical variables (Jason, 2022). When employing KNN Imputer, to find the optimal number of 'k' neighbors, we used a simple grid search strategy and found 'k=3' to be optimal.

Model Building and Evaluation:
We then used these two processed datasets to train two different types of models: a Decision Tree Classifier and a K-Nearest Neighbors (KNN) Classifier. For model optimization, we employed GridSearchCV for hyperparameter tuning. For the Decision Tree model, we tuned 'max_depth' and 'min_samples_leaf', and for the KNN model, we tuned 'n_neighbors'.

The models were evaluated using the F1 score as our primary metric. We also recorded the training time for each model as an additional performance measure. To better understand our models' performance, we plotted the Precision-Recall curve for each model on each dataset.

3. Project Analysis

Data:
The dataset we used for this project is a CSV file from Kaggle, with information about customers and whether they have churned or not. This data includes demographic data (like age, gender, and income level), as well as behavioral data (like credit card category, credit limit, and total transaction count). In total, there are 10127 entries, with 23 original features, reduced to 19 after we dropped irrelevant or redundant ones. There were no null values in the data, but there were some "Unknown" values in two categorical features that we treated as missing and imputed.

Algorithms:
For missing values imputation, we choose KNN imputation and mode imputation as they are suitable for categorical variables but work in different ways, and the evaluation is decided by comparison between the model performance from two different datasets after imputation.

For building models, we choose Decision Tree and K-Nearest Neighbors (KNN). The choice of these two models was based on their simplicity and interpretability. Decision Trees can model nonlinear relationships, and they are easy to understand and visualize. On the other hand, KNN, as a

non-parametric method, doesn't make any assumptions about the data's distribution and can adapt quickly as we collect new data.

We use F1 score as our evaluation metric, given its effectiveness in dealing with imbalanced data. The F1 score is a harmonic mean of precision and recall, which are both important for our problem at hand. Given that our dataset is imbalanced with fewer instances of customers who churned than those who didn't, using accuracy as a metric could give us misleading results.

The Precision-Recall Curve to visualize the trade-off between precision and recall for different threshold settings, because we think it would be better to focus a bit more on recall for detecting all the customers who are going to leave the service.

Experimental details:
For hyperparameter tuning, we used the GridSearchCV method with 5-fold cross-validation from scikit-learn. This method exhaustively searches through a specified subset of the hyperparameter space of a learning algorithm. In our case, for the Decision Tree model, we considered a range of values for 'max_depth' (3, 5, 7, 9, 11) and 'min_samples_leaf' (from 1 to 5). For the KNN model, we considered values for 'n_neighbors'(3, 5, 7, 9, 11) .
The model configuration and training time are shown below:

	Model configuration	Training time
Decision Tree with KNN imputation	{'max_depth': 7, 'min_samples_leaf': 1}	8.310558319091797 seconds
Decision Tree with Mode imputation	{'max_depth': 7, 'min_samples_leaf': 1}	6.88030481338501 seconds
KNN model with KNN imputation	{'n_neighbors': 11}	2.824738979339599 seconds
KNN model with Mode imputation	{'n_neighbors': 11}	2.925180912017822 seconds

Grid-search for Decision Tree on dataset KNN imputation: (Same setting with dataset mode imputation)

	rank_test_score	mean_test_score	param_max_depth	param_min_samples_leaf
0	21	0.950953	3	1
1	21	0.950953	3	2
2	21	0.950953	3	3
3	21	0.950953	3	4
4	21	0.950953	3	5
5	16	0.959567	5	1
6	20	0.959350	5	2
7	17	0.959509	5	3
8	18	0.959509	5	4
9	19	0.959438	5	5
10	1	0.965656	7	1
11	2	0.965600	7	2
12	6	0.964639	7	3
13	3	0.965537	7	4
14	4	0.965280	7	5
15	11	0.963968	9	1
16	10	0.964171	9	2
17	7	0.964451	9	3
18	9	0.964233	9	4
19	5	0.964797	9	5
20	13	0.962486	11	1
21	15	0.962066	11	2
22	14	0.962383	11	3
23	12	0.963393	11	4
24	8	0.964428	11	5

Grid-search for KNN on dataset KNN imputation: (Same setting with dataset mode imputation)

	rank_test_score	mean_test_score	param_n_neighbors
0	5	0.933737	3
1	4	0.935447	5
2	2	0.937022	7
3	3	0.936697	9
4	1	0.938471	11

Results:

1. **F1 Score on two models * two datasets**

   Upon evaluating our models, we found that both Decision Tree and KNN performed remarkably well, with the Decision Tree model having a slight edge.

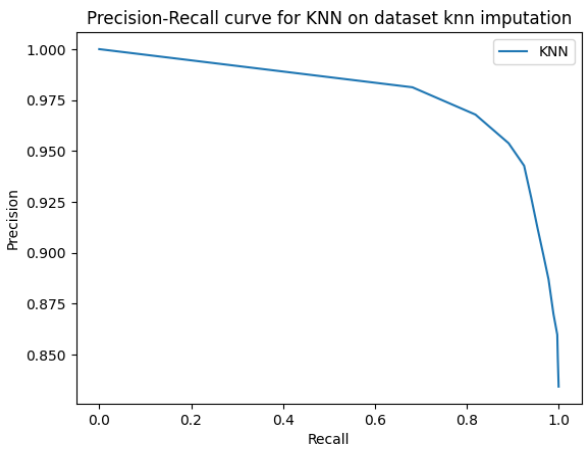
   However, given that the datasets processed by the two imputation techniques turned out to be identical, both models returned the same F1 scores, which is different from what we expected:
   The best F1 scores for the Decision Tree models were:
   - dt_knn imputation: 0.9665492957746479
   - dt_most frequent imputation: 0.9665492957746479

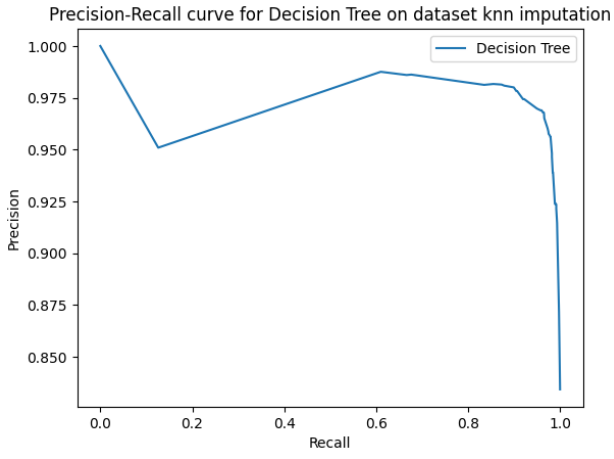
   The best F1 scores for the KNN models were:
   - knn_knn imputation: 0.932600520682673
   - knn_most frequent imputation: 0.932600520682673

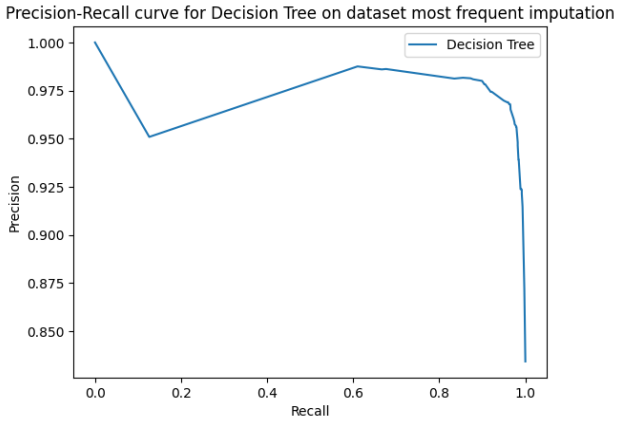
   Even though we included other relevant variables in KNN imputation, the KNN and mode imputations led to the same results, we might conclude that the missing data was likely missing completely at random *(*Buuren, 2018*)*, where the missingness is completely random and doesn't depend on any other variable.

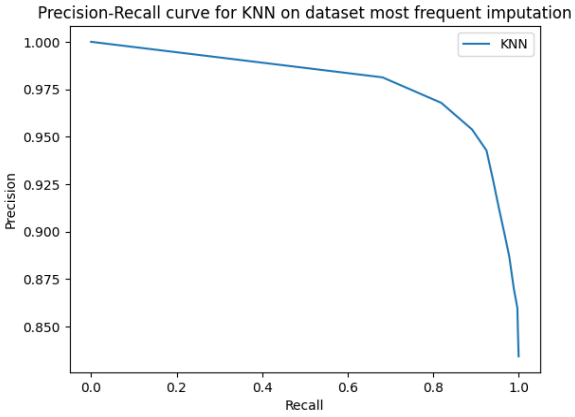
2. **PR Curve on two models * two datasets**

   And the precision-recall curves were analyzed to understand their performance across different thresholds. We noticed that the curve for the Decision Tree model was more skewed towards the right corner compared to the KNN model. This indicates that the Decision Tree model was better at maintaining a high recall rate without sacrificing precision, making it a more effective model for this dataset and for detecting potential missing customers.





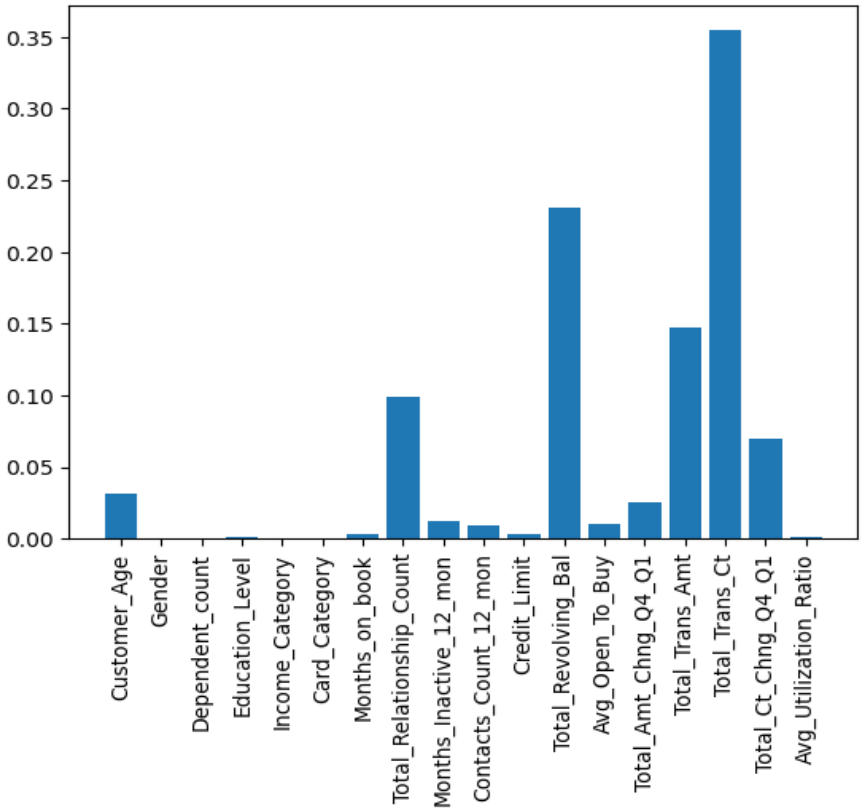




3. Conclusion

Recommendation to have the best model efficiently:

- Simpler mode imputation to handle missing value in categorical variables
- Decision Tree model with max_depth as 7, min_samples_leave as 1
- Feature Importance: In the best Decision Tree model, suggest paying attention to the following features in below plot, as they were identified as important for predicting customer churn.



Feature importance in the best decision tree model

Reference:

Buuren, S. van. (2018). 1.2 Concepts of MCAR, MAR and MNAR. Flexible Imputation of Missing Data. https://stefvanbuuren.name/fimd/sec-MCAR.html

Nawale, T. (2022, May 25). How to deal with missing values in machine learning. Medium. https://medium.com/geekculture/how-to-deal-with-missing-values-in-machine-learning-98e47f025b9c

Jason, T.(2022, June 24) kNN Imputation for Missing Values in Machine Learning. Data Preparation. https://machinelearningmastery.com/knn-imputation-for-missing-values-in-machine-learning/