Efthimios Vlahos

AMS 561

Final Project

26 April, 2023

Final Project Report: American Express Default Prediction

Introduction

The convenience of credit cards has become an indispensable aspect of modern life, facilitating daily purchases without the need to carry large amounts of cash. However, as credit card issuers extend credit to customers, the challenge of predicting the likelihood of payment default arises. This is a complex problem that has been addressed by many existing solutions, with opportunities for further improvements.

In this report, I present the results of the American Express - Default Prediction Kaggle competition, an old data science competition hosted on the Kaggle platform by American Express. The objective of the competition was to predict which credit card customers were likely to default on their payments.

Data Preprocessing

The data for this report consists of aggregated profile features for each customer at each statement date. These features have been anonymized and normalized and can be broadly categorized into five groups: Delinquency (D_*), Spend (S_*), Payment (P_*), Balance (B_*), and Risk (R_*). The Delinquency variables provide information on customer payment behavior, while Spend variables describe their transaction patterns. The Payment variables relate to the amount and

timing of customer payments. Balance variables provide insight into the customer's debt and available credit, and Risk variables offer information on the likelihood of default. Overall, these anonymized and normalized features provide a comprehensive view of customer credit card usage and behavior, forming the basis of the analysis and modeling presented in this report.

Snapshot of shapes of training, label, and merged training Dataframe :

```
In [7]: #Shape of sample training dataframe
        print('Shape of dataset is:', df_train_sample.shape)

        #Print summary of dataframe (df)
        df_train_sample.info()

        Shape of dataset is: (100000, 190)
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 100000 entries, 0 to 99999
        Columns: 190 entries, customer_ID to D_145
        dtypes: float64(185), int64(1), object(4)
        memory usage: 145.0+ MB
```

```
In [12]: #General properties of labels df
         train_labels_df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 458913 entries, 0 to 458912
         Data columns (total 2 columns):
          #   Column       Non-Null Count   Dtype
         ---  ------       --------------   -----
          0   customer_ID  458913 non-null  object
          1   target       458913 non-null  int64
         dtypes: int64(1), object(1)
         memory usage: 7.0+ MB
```

```
: #Merging of training data, inner takes intersection of the two dataframes based on customer_ID key
  df_train = pd.merge(df_train_sample, train_labels_df, how="inner", on=["customer_ID"])
```

```
: df_train.info()

  <class 'pandas.core.frame.DataFrame'>
  Int64Index: 100000 entries, 0 to 99999
  Columns: 191 entries, customer_ID to target
  dtypes: float64(185), int64(2), object(4)
  memory usage: 146.5+ MB
```

Observations:

- So there are 100,000 rows (training examples) with 191 columns (features) in the merged training dataframe now containing the labels. Specifically: 185 features (Columns) as dtype float64, 2 features (Column) as dtype int64 and 4 features (Columns) as dtype object

Missing Data

Dealing with missing data is a critical step in any data analysis task, and there are several approaches to handling it. One common method is to use code that identifies missing values and either removes them or imputes them with estimated values based on other features. In this report, during the data preprocessing phase, I used Python code to drop columns in the training and test dataset that had more than 70% null values (NAN's) in their respective columns. This led to a reduced Dataframe for the training set and testing set which resulted in 25 less features/columns the potential models for classification can work with near the end of the project.

Snapshot of reduced training Dataframe after removing columns with > 70% NAN's and Barplot of count of features:

```
#Drop variables with missing values >=70% in the train dataframe
num_cols=0
for col in df_train.columns:
    if (df_train[col].isnull().sum()/len(df_train[col])*100) >=70:
        df_train.drop(axis=1,labels=col,inplace=True)
        print("Dropping column", col)
        num_cols+=1
print("Total number of columns dropped in train dataframe", num_cols)
```

```
#New training df characteristics
df_train.info()
```

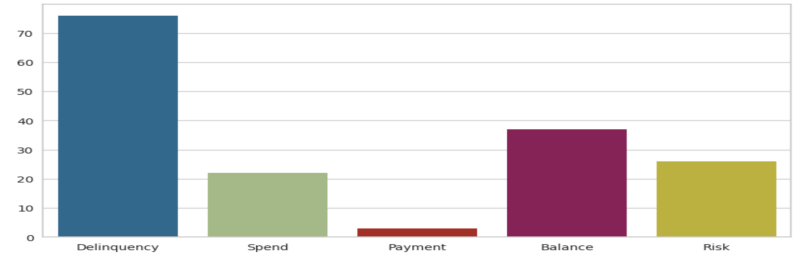
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100000 entries, 0 to 99999
Columns: 166 entries, customer_ID to targe
dtypes: float64(160), int64(2), object(4)
memory usage: 127.4+ MB
```

```
#New test df sample characteristics
df_test_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 100000 entries, 00000469ba478561f23
caf1772b0512becef11a1eda0948b8b8908576
Columns: 164 entries, S_2 to D_145
dtypes: float64(160), int64(1), object(3)
memory usage: 125.9+ MB
```

```
#Barplot counting each type of feature
Dict = {'Delinquency': len(D_columns), 'Spend': len(S_columns), 'Payment': len(P_columns), 'Balance': len(B_columns),

plt.figure(figsize=(10,5))
sns.barplot(x=list(Dict.keys()), y=list(Dict.values()));
```



Exploratory Data Analysis

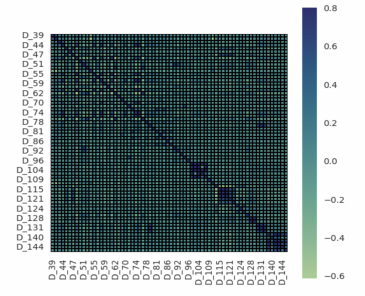
Exploratory Data Analysis (EDA) was conducted to gain a better understanding of the competition dataset. I used Python libraries such as Pandas, Matplotlib, and Seaborn to visualize and summarize the data for each type of variable. Through the analysis, it was revealed that the dataset was imbalanced. I also observed that some features had high correlation with each other, indicating potential redundancy in the feature set. Additionally, I identified missing values in several features, which were subsequently handled using imputation techniques. Overall, the EDA provided

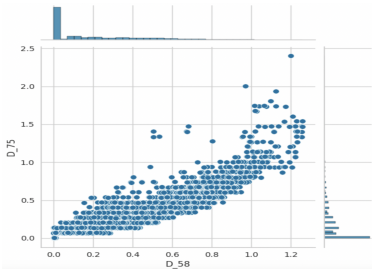
valuable insights into the dataset, enabling us to make informed decisions in the subsequent modeling stages of the competition.

D_*:Delinquency Variables

The exploratory data analysis of the delinquency variables, which was similarly done for all the features, revealed interesting findings. I used various Python libraries, primarily matplotlib and seaborn, to visualize and analyze the distribution of the D variables. The distribution plot indicated that some variables had extreme values, which were essentially binary variables with values of 1 or 0. We also observed skewness in some of the distributions, with a few variables appearing to follow a normal distribution.

Plots of correlation matrix and joint distribution of two correlated features D_* Variables:



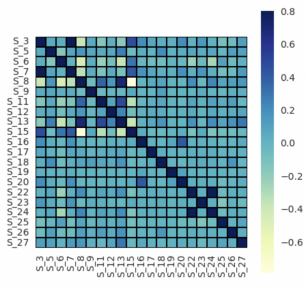


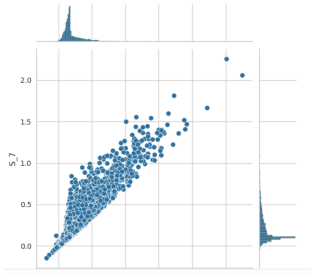
S_*: Spend variables

The exploratory data analysis of the Spend variables provided valuable insights into the customers' spending behavior. The distribution plot indicated that the majority of the customers had low spend values, with a few outliers having extremely high spend values. I also observed that some S variables had a linear relationship with each other, indicating a potential redundancy in the feature

set. Additionally, the analysis revealed missing values in some of the S variables, which were subsequently handled using imputation techniques.

Plots of correlation matrix and joint distribution of two correlated features S_* Variables:

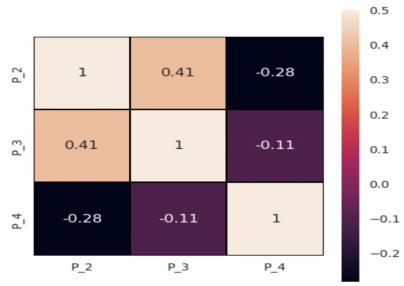


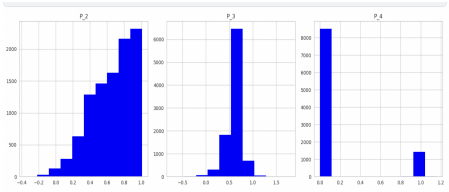


Rest of Features: Payment (P_*), Balance, and Risk

The Payment, Balance, and Risk variables provided valuable insights into the customers' financial behavior. Just like the EDA for the Delinquency and Spend variables, I used Python libraries such as Pandas, Matplotlib, and Seaborn, visualized and summarized the data to gain insights. The EDA revealed that the majority of customers had a low payment and balance amount, with a few customers having high values. For the Risk variables, I observed that most of the variables had a skewed distribution, indicating the presence of outliers.

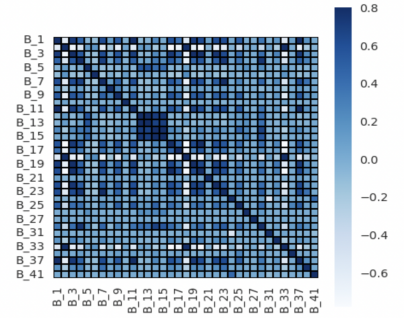
Various plots of Payment, Balance, and Risk features:

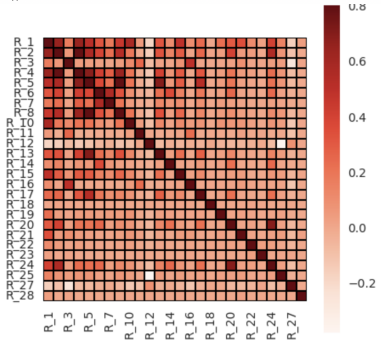




Observation:

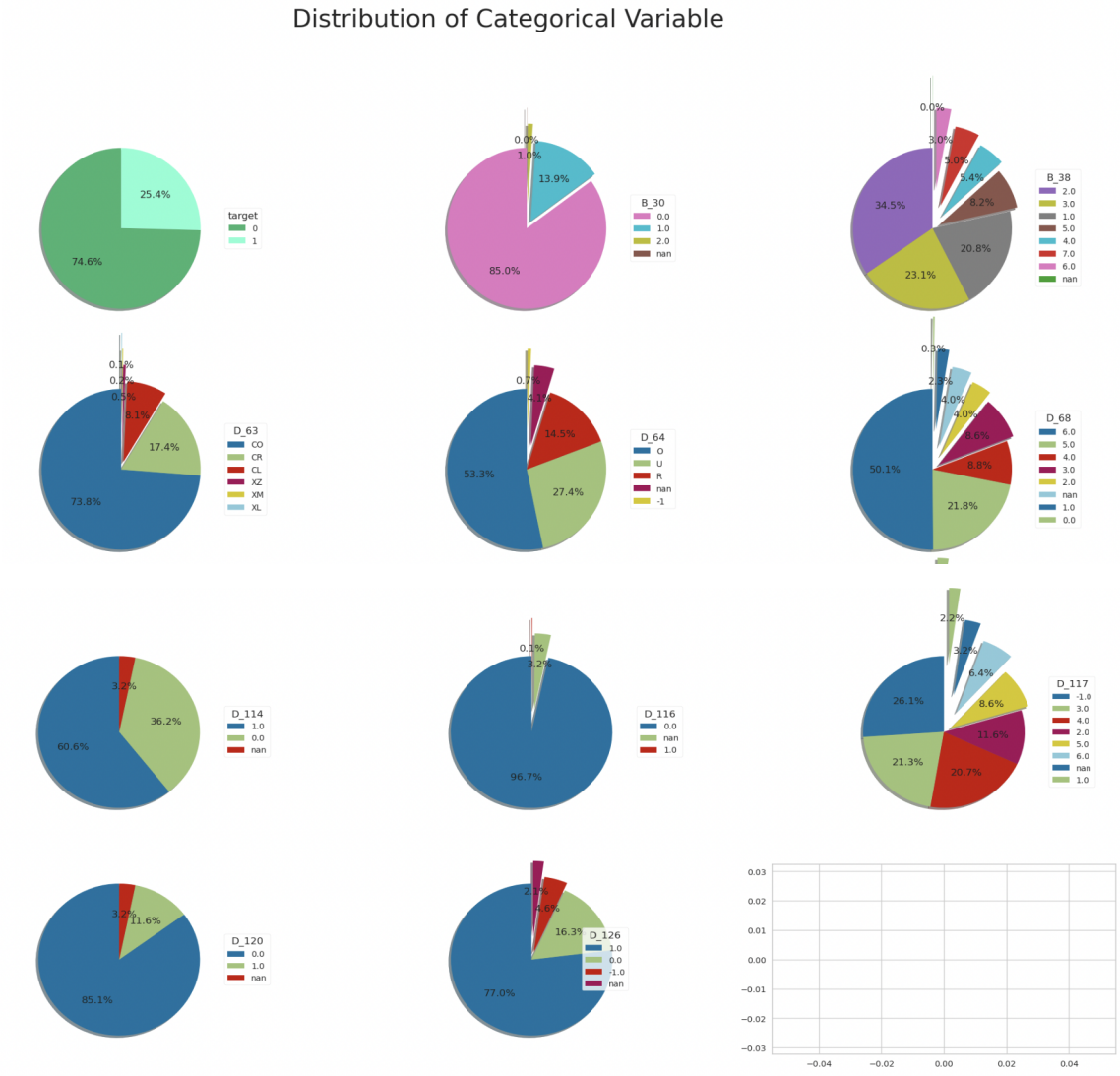
- P_2 variable appears to have negative skewed distribution (longer left tail)
- P_3 variable appears to have normal distribution
- P_4 variable appears to have bimodal distribution





Categorical variables

The categorical variables played a crucial role in predicting the likelihood of default by the customers and are inevitable in any data science project. Given the categorical variables, I printed pie charts of each categorical feature to try and get a grasp of its respective distribution, including the target variable in the training set.



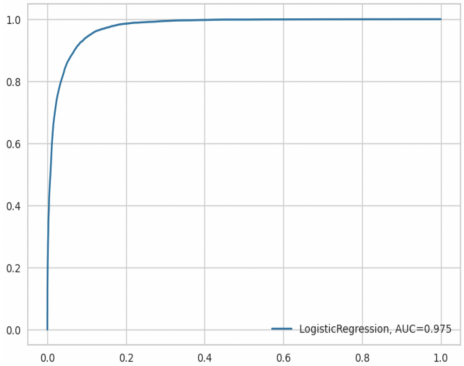
Model Building/Evaluation

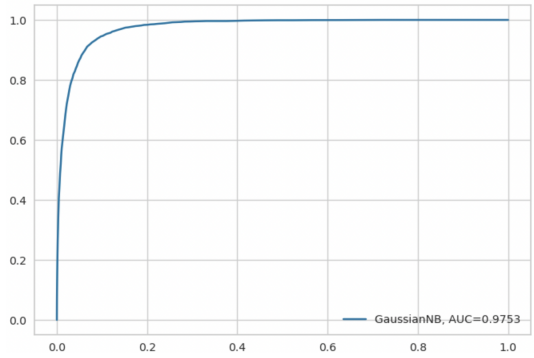
In this study, I compared the performance of several machine learning classification models, specifically Logistic Regression, Random Forest, Naive Bayes, KNN, and Gradient Boosting for default prediction using metrics such as accuracy, precision, recall, F-1 score, validation scores, and AUC/ROC scores to evaluate model performance. All the models performed considerably well and there really wasn't one clear cut favorite. It is worth noting that while the AUC/ROC scores provide a good indication of a model's overall performance, it is also important to consider other metrics such as precision, recall, and F1 score, which can provide a more nuanced understanding of a model's strengths and weaknesses.

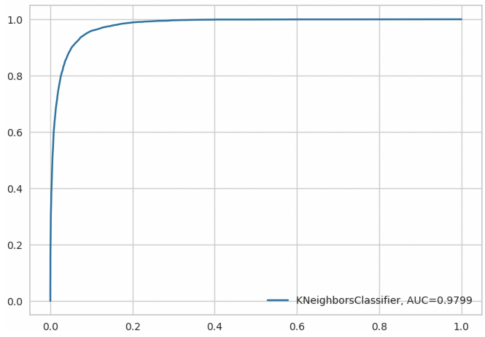
To make the code more clear and organized, I created a pipeline that streamlined the entire process from data pre-processing to model evaluation. The pipeline consisted of several stages, including data cleaning, feature engineering and reduction, model selection, and hyperparameter tuning. The data cleaning stage involved imputing missing values and removing outliers. The feature engineering stage involved encoding categorical variables using one-hot encoding. The model selection stage involved trying out several machine learning algorithms (ones mentioned earlier) to find the best-performing model. Finally, hyperparameter tuning was performed on the selected model to optimize its performance. The pipeline allowed me to iterate through different models relatively quickly and experiment with various approaches.

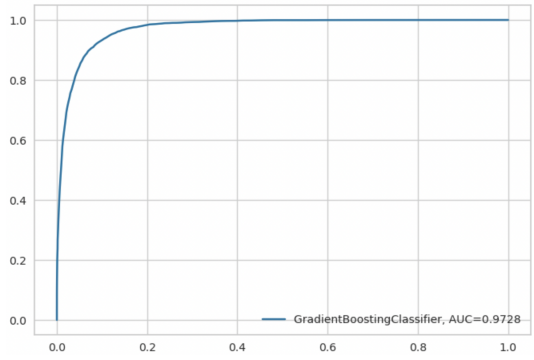
Table of scores for each model on test set (No cross validation) and ROC/AUC curves

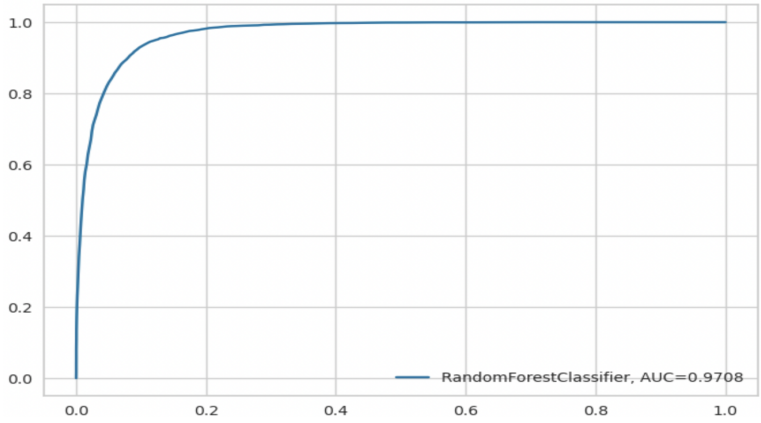
Metric	Logistic Regression	Naive Bayes	KNN	Gradient Boosting	Random Forest
Accuracy	.92	.92	.93	.91	.91
F1-Score	.90	.90	.91	.89	.89
Precision	.88	.88	.89	.87	.87
AUC/ROC	.975	.975	.980	.9728	.9708











Conclusion

In conclusion, the machine learning models that were developed and evaluated performed exceptionally well in predicting whether a customer will default. The AUC/ROC scores provide reliable indication of the overall performance of the models. Overall, the competition has demonstrated the potential of machine learning in predicting default, which can have significant implications for credit risk management in the banking and financial sector.

References

https://www.kaggle.com/competitions/amex-default-prediction/overview

https://www.kaggle.com/code/eraikako/credit-card-default-prediction-educational-series/notebook#Feature-Engineering