Project Report
CSP571 Data Preparation and Analysis
Illinois Institute of Technology

Group 11

November 30, 2024

Contents

GitHub repository: https://github.com/PaulAnrd/CSP571-Class-project-Group-11

1 Abstract

This project involves the prediction of credit card approvals by analyzing applicant data through various machine learning techniques. The goal is to identify patterns and features that are most indicative of approval decisions. Key challenges like data imbalance, missing values, and feature selection were addressed using advanced preprocessing and modeling strategies. Several models were evaluated to achieve a balance between performance and interpretability.

2 Introduction

Predicting credit card approvals is an essential task for financial institutions, enabling effective risk management and enhanced customer experience. The aim of this project is to predict approval outcomes based on financial and demographic data of applicants. This report details the methodology, data processing steps, and results from the analysis, providing insights into key predictive factors.

3 Dataset

Dataset link: https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction

Features

```
Index(['ID', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN',
       'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'FLAG_MOBIL', 'FLAG_WORK_PHONE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'MONTHS_BALANCE',
       'STATUS'],
      dtype='object')
```

The target is "STATUS," which is the status of credit payment. As shown in the code, STATUS is a value from 0 to 6.

3.1 Data Cleaning

Before any analysis, data preprocessing and cleaning are always crucial steps, even more so when working with different types of data. In our case, the dataset contained flags like yes or no for attributes, alongside numerical data. To proceed with the analysis, we changed all qualitative data to numerical data, either binary or numerical, to create some level of numerical value for each class. In doing so, we also addressed NaN values or other elements that could have disrupted the analysis.

4 Analysis

4.1 Exploring data

4.1.1 Correlation between features

Correlation Heatmap: A heatmap was generated to explore relationships between features. It highlighted strong correlations, particularly between income and credit risk.

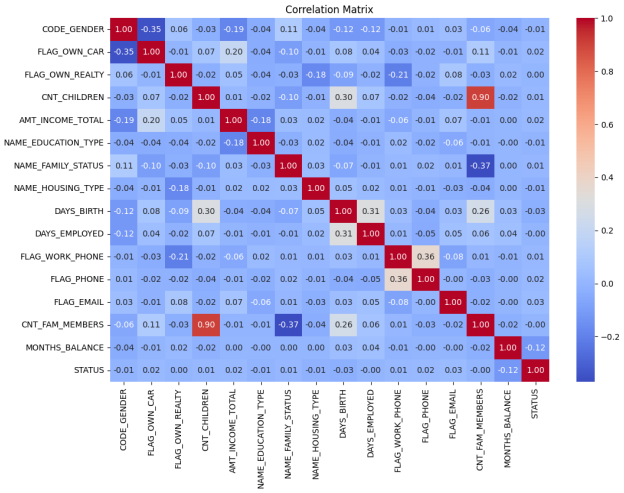


Figure 1: Correlation Heatmap of Features

4.1.2 Impact on the target

Thanks to that visualization, we can clearly see that the feature with the most important impact on the target is the monthly balance, which isn't surprising.

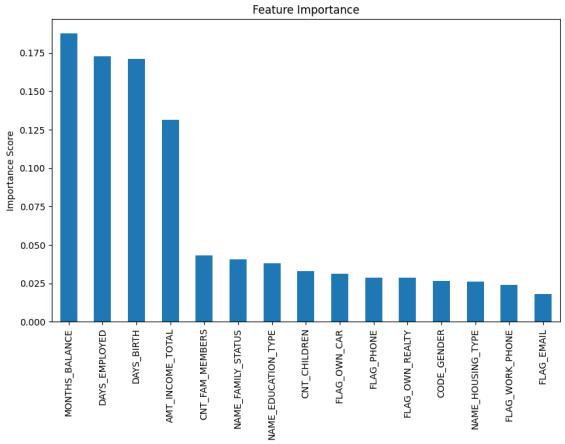


Figure 2: Features impact on the target

4.1.3 Independence assumption

With the Chi-squared test, we can verify the independence assumption. As shown in this example, the age of the customer is dependent on the target.

```
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(data['DAYS_BIRTH'], data['STATUS'])
chi2, p, dof, expected = chi2_contingency(contingency_table)

print("Chi-squared Test: p-value = ", p)
if p < 0.05:
    print("Features are not independent.")
else:
    print("Features are independent.")
```

```
Chi-squared Test: p-value =  3.092915327999291e-21
Features are not independent.
```

4.2 Visualization Strategies

Dimensionality reduction techniques, including PCA, UMAP, and t-SNE, were applied for feature visualization:

4.2.1 Correlation plot

By using this representation, we can clearly see the relationship between all features and their distribution on the target. This also highlights the number of classes of each feature, which can be challenging with this dataset.

4.2.2 PCA

Principal Component Analysis: Reduced high-dimensional data to 2D visualization

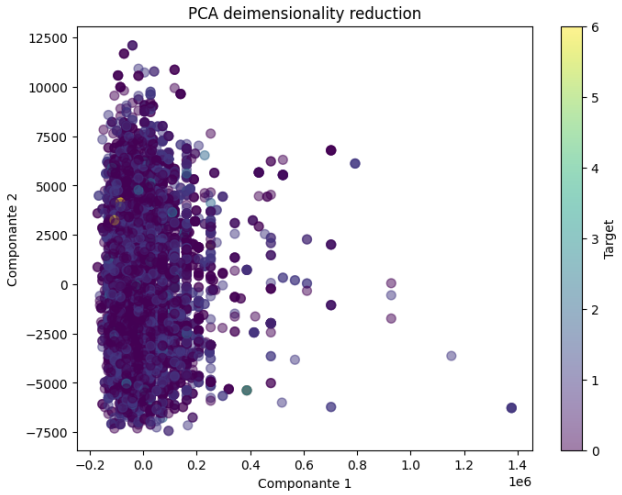


Figure 3: PCA Results: 2D Projection

4.2.3 UMAP

Uniform Manifold Approximation and Projection: Showed distinct clusters of customers based on financial and demographic features. This plot shows how the data is distributed in a lower-dimensional space. While the points are spread across the space, there is significant overlap between clusters, indicating that the classes of the target are not well-separated. This suggests features may not provide clear distinctions between the classes.

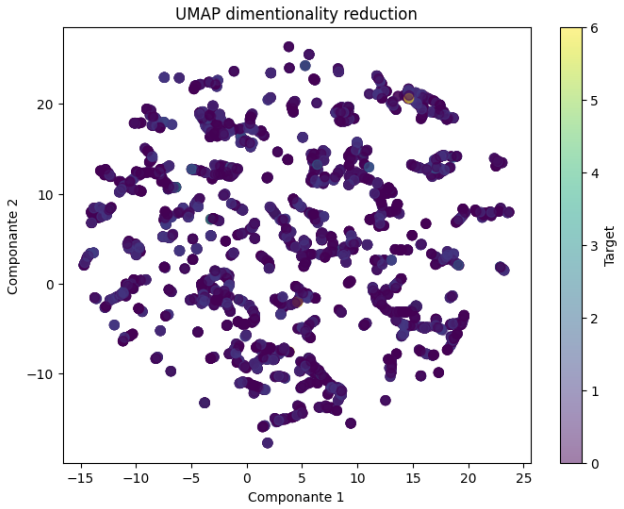


Figure 4: UMAP Visualization of Clusters

4.2.4 t-SNE

This t-SNE plot similarly reveals overlapping clusters, but it highlights some subtle separations compared to UMAP. There are slight groupings that indicate potential structure in the data, yet the clusters are not distinctly isolated. This suggests that while t-SNE captures more nuanced patterns.

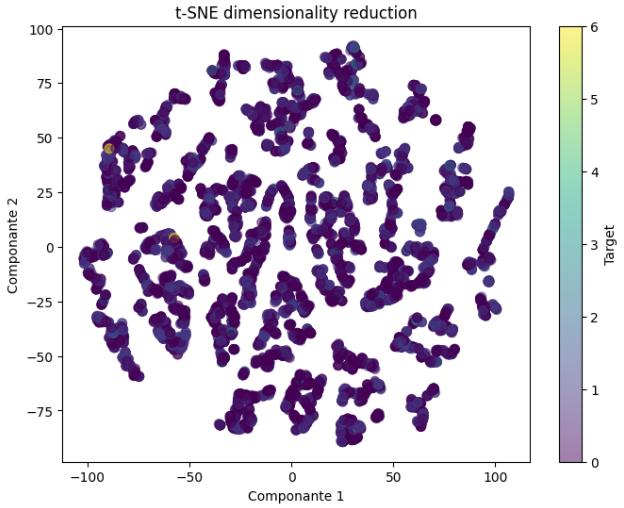


Figure 5: t-SNE Visualization of Data Clusters

4.3 Unsupervised learning techniques

By using the PCA technique, we can focus on unsupervised learning algorithms like K-means. In this example, the data is very clustered, and it is challenging to identify multiple clusters. However, the K-means algorithm predicts the following 2D representation:

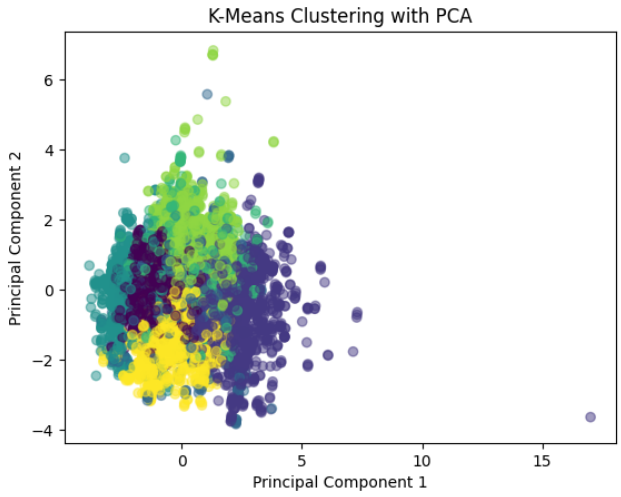


Figure 6: K-means 2D Clustering Representation

This clustering can be partially explained by the challenging dataset that groups distinct, non-linear data that is sometimes poorly distributed.

4.4 Cross Validation strategy

The scores are relatively low, indicating that the clustering structure is weak, with data points not clearly assigned to well-separated clusters. This may indicate overlapping clusters.

```
Silhouette Scores for Cross-Validation:  [0.08518711667280139, 0.10102066111979742,
    0.0875081997801321, 0.0827279467807841, 0.0893039610550859]
Mean Silhouette Score:  {0.08914957708172018}
```

4.5 Models training

	precision	recall	f1-score	support
0	0.65	0.91	0.76	1246
1	0.57	0.19	0.28	729
2	0.08	0.04	0.06	23
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.00	0.00	0.00	2
accuracy			0.63	2000
macro avg	0.19	0.16	0.16	2000
weighted avg	0.61	0.63	0.58	2000

We chose to train a gradient boosting classifier with our data. By taking a look at the model results, we can conclude that it performs at best average when classes were sufficiently represented and poorly when classes weren't represented enough. This is due to the size of the dataset and our computational power available. With more data input, the model would certainly have performed better, especially on low-represented classes.

4.5.1 Validation set for hyperparameters tuning

	precision	recall	f1-score	support
0	0.64	0.93	0.76	1246
1	0.56	0.14	0.23	729
2	0.00	0.00	0.00	23
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.00	0.00	0.00	2
accuracy			0.63	2000
macro avg	0.20	0.18	0.16	2000
weighted avg	0.60	0.63	0.56	2000

After implementing hyperparameters tuning, it slightly improved the model performance on recall. However, we can observe a slight drawback on less represented classes. With these parameters, the model appears to perform better for Class 0 but performs worse on other classes.

4.5.2 Analyse performance using cross validation

Cross-Validation Results Summary:

	train_accuracy	test_accuracy	train_f1	test_f1
count	5.000000	5.000000	5.000000	5.000000
mean	0.648625	0.628500	0.576156	0.550885
std	0.001883	0.005165	0.003520	0.007033
min	0.645938	0.624375	0.573362	0.542099
25%	0.647500	0.625000	0.573607	0.548205
50%	0.649375	0.626250	0.573903	0.550595
75%	0.649687	0.630000	0.579141	0.552053
max	0.650625	0.636875	0.580769	0.561471

The cross-validation results show consistent performance with low variance, indicated by small standard deviations for both accuracy and F1 scores across folds. The mean train accuracy is slightly higher than test accuracy, suggesting a minor degree of overfitting. Overall, the model demonstrates stable but moderate performance.

4.6 Performance improvement

Test Set Performance:

	precision	recall	f1-score	support
0	0.67	0.79	0.73	1246
1	0.50	0.36	0.42	729
2	0.20	0.04	0.07	23
6	0.00	0.00	0.00	2
accuracy			0.62	2000
macro avg	0.34	0.30	0.30	2000
weighted avg	0.60	0.62	0.61	2000

Improving classification performance is one of the main drawbacks. The model seems to identify Class 0 well but struggles with other classes. While changing to an XGBoost classifier and tuning parameters, the distribution of performance across classes improved.

4.7 More experiments

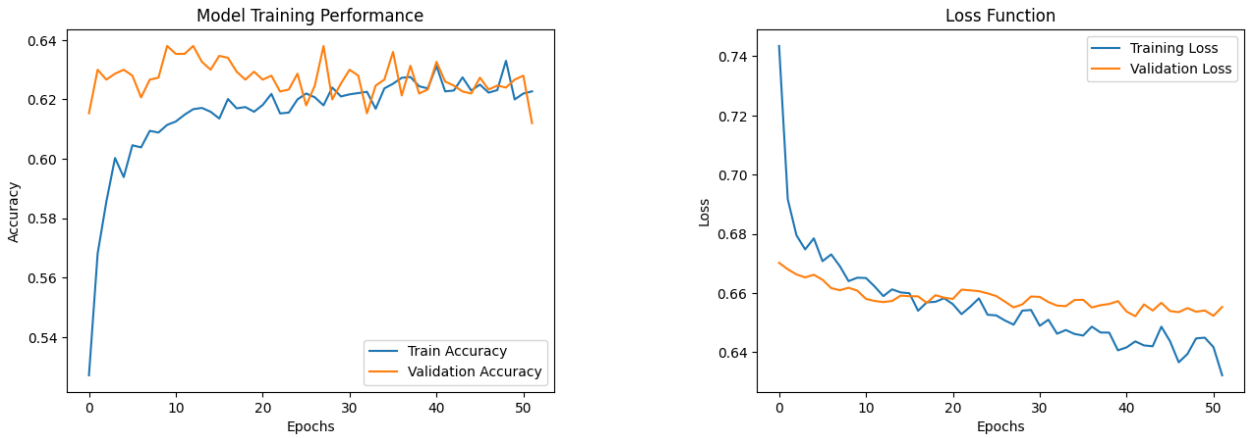
4.7.1 Neural Network approach

```
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.86      0.74       933
           1       0.48      0.24      0.32       545
           2       0.00      0.00      0.00        17
           3       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         3

    accuracy                           0.62      1500
   macro avg       0.23      0.22      0.21      1500
weighted avg       0.58      0.62      0.57      1500
```

Training the neural network model, we can see that performance starts to hit the ceiling around 200 epochs. The results seem to confirm the last model's performance, showing that the data could be pretty challenging to predict due to the overlapping clusters seen earlier. However, the neural network performs decently, given the representation of each class. Even though neural networks can usually be slow to train, the training was very comparable to other ML algorithms used in this project.



(a) Model Accuracy over training

(b) Loss function evolution

Figure 7: Neural Network performance

4.7.2 Stacked Ensemble Model

```
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.88      0.76      1246
           1       0.59      0.32      0.41       729
           2       0.00      0.00      0.00        23
           6       0.00      0.00      0.00         2

    accuracy                           0.66      2000
   macro avg       0.32      0.30      0.29      2000
weighted avg       0.64      0.66      0.63      2000
```

Finally, the Stacked Ensemble Model is a technique used to perform at its best by trading pros and cons of multiple models to predict at its best while being less subject to overfitting. The main drawback is certainly the computational time. The overall performance on prediction was among the best we have seen with this dataset, thanks to combining random forest and gradient boosting.

5 Conclusion

In this project, we applied different data analysis techniques, successfully predicting the credit default risk that could occur. This also highlights the importance of choosing the classifier wisely, creating significant performance differences among different target classes. The importance of the dataset can become a major challenge and limit models' performance. Given more time, we could have revisited data clustering.