CREDIT CARD USAGE UNDERSTANDING FOR NEXT MONTH PAYMENT PREDICTION

TRAN MINH NAM, HUYNH NHAT ANH, LE HOAI THUC NHI,
NGUYEN TRAN NHAT QUOC, NGUYEN DAC HOANG PHU
VIETTEL DIGITAL TALENT PROGRAM 2023

ABSTRACT

This report focuses on credit card usage understanding for next month's payment prediction, specifically looking at customers' default payments in Taiwan. The study suggests that understanding user behavior is important for predicting next-month payments and gives further insights.

KEYWORDS

Imbalance dataset, Clustering, Getting Insight

1 INTRODUCTION

In today's data-driven world, companies are constantly seeking insights from their data to improve their business operations and decision-making. Our team has embarked on a project to identify insights from data, with a focus on the following tasks. Firstly, we identified the problem and use case we wanted to implement. We then gathered potential data for the use case from the company's data warehouse. Next, we applied data analysis skills, data preprocessing techniques, data mining, and model building to identify a solution for predicting the outcome for the use case. Finally, we presented our proposed ideas and solutions in a comprehensive report. Our main objective is to extract meaningful insights from the data that can help the company make informed decisions and improve their business operations. By performing data analysis and building predictive models, we hope to uncover hidden patterns and relationships in the data that can reveal new opportunities and potential areas for improvement. As part of our mission, our team also selected a dataset that allows us to apply the data analysis techniques and skills we have learned. We chose a language corpus dataset that contains both numerical and categorical data, has a large number of instances, and is particularly imbalanced. Working with this dataset will enable us to deepen our understanding of the limitations of certain techniques and gain insight into the various steps and techniques used in data projects. This will help us to develop a better understanding of the data analysis process and increase our confidence as we continue on this path. By grappling with the challenges presented by this imbalanced dataset, we will gain valuable experience and knowledge that we can apply to future data analysis projects.

Use-case. The team's goal is to identify corresponding use cases to analyze the available data.

- Visualize data using charts and analyze correlations and relationships between important attributes to create an overview of the data.
- Develop a model capable of predicting customer behavior based on data.
- Clustering customers to gain insights into their behavior.

Related Work.

- Understand and gain an overview of the data through information about the schema, type, and chart format.
- Then, perform feature selection to remove redundant features to improve the quality of the data for model learning.
- Investigate the meaning of the label column and determine what corresponding problems can be solved with this dataset.
- Experiment with multiple different models to achieve the highest accuracy possible for the chosen label column.
- Conclude and provide meaningful insights into the model learning and hidden behavior in the data through the mining process.

2 BACKGROUND

2.1 Data collection

The data for this study was collected in Taiwan and aimed to predict the probability of default payments for customers. The research was conducted by I-Cheng Yeh, who works in the Department of Information Management at Chung Hua University and the Department of Civil Engineering at Tamkang University in Taiwan.

The dataset includes a binary response variable, default payment, with a value of 1 for defaulters and 0 for non-defaulters. The study used 23 explanatory variables to predict the probability of default. The first variable, X1, represents the amount of credit given to the customer, including individual consumer credit and supplementary credit. The second variable, X2, indicates the gender of the customer, with a value of 1 for males and 2 for females. The third variable, X3, shows the education level of the customer, with a value of 1 for graduate school, 2 for university, 3 for high school, and 4 for others. The fourth variable, X4, represents the marital status of the customer, with a value of 1 for married, 2 for single, and 3 for others. The fifth variable, X5, indicates the age of the customer in years.

The remaining 18 variables, X6 to X23, are related to the payment history, bill statements, and previous payment amounts of the customer. Variables X6 to X11 track the past monthly payment records from April to September 2005, with values ranging from -1 (pay duly) to 9 (payment delay for nine months and above). Variables X12 to X17 represent the amount of the billing statement for the months from April to September 2005. Variables X18 to X23 show the number of previous payments made by the customer for the months from April to September 2005.

The dataset can provide valuable insights into the factors that contribute to default payments, allowing financial institutions to improve their risk management practices and make better-informed

Project1 in VDT, gen 3, (2023)
(C).

decisions regarding credit approvals and risk assessments for their customers. The dataset includes information on various demographic and payment-related factors, which can be used to identify significant risk factors and take steps to mitigate them. Overall, the dataset can inform decision-making processes for financial institutions seeking to reduce the risk of default payments.

2.2 Data selection and transformation

In statistics and machine learning, feature selection is the task of selecting a subset of features from the dataset, where data relevant to the analysis tasks are retrieved. Feature selection is applied when we want to, help our machine learning algorithm train faster, reduce model complexity by removing redundant features, find precious features for data mining, enhance generalization, and avoid overfitting.

There are two kinds of selection: supervised and unsupervised. In unsupervised selection, we do not use the target attribute for selecting and evaluating the feature importance, in contrast with the supervised selection approach, where we use the target variable to remove irrelevant input features. The group of supervised methods can be categorized into 3 kinds: filter method, wrapper method, and embedded method (intrinsic).

Figure 1 is a graphical representation that demonstrates the hierarchical structure of different feature selection methods. The methods are organized into categories based on their characteristics and the underlying approach utilized.

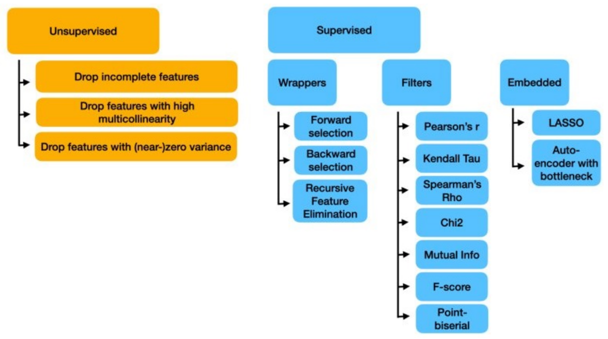


Figure 1: Hierachical structure of feature selection methods

2.2.1 Filter method. Filter feature selection methods use mathematical techniques to understand how each input variable relates to the target variable, and these mathematical scores are used to decide which input variables will be used in the model.

The scores used in filter-based feature selection are usually calculated for each input variable independently of the target variable. This type of calculation is known as univariate statistical measures. However, this method does not take into account how different input variables may interact with each other, which can affect the filtering process.

The techniques related to this method can be listed as follows:

- Univariate methods
- Information gain
- Fischer score
- Correlation Matrix with Heatmap

2.2.2 Wrapper method. Wrapper feature selection methods generate multiple models using different sets of input features and then choose the features that result in the best-performing model based on a performance metric. These methods are not worried about the types of variables used, although they can be computationally demanding. Some wrapper methods are forward selection, backward elimination, exhaustive feature selection, etc.

2.2.3 Embedded method. The embedded method is a popular technique for feature selection in machine learning that combines elements of both filter and wrapper methods. Unlike filter methods that rely on statistical measures to rank the importance of each feature and wrapper methods that use a trial-and-error approach to evaluate subsets of features, the embedded method integrates feature selection into the training process itself. It is often used with tree-based models, such as decision trees, random forests, and gradient boosting machines.

2.3 Dealing imbalance dataset problem

Data imbalance is one of the common phenomena of a binary classification problem. It usually appears in any field: healthcare, financial economics, traffic, email spam, or venture capital-related fields. It usually occurs because: data is null too much leading to data deletion (there is a way to fix null by imputation data), data is secure for some reason (medical), and data is biased. subjective orientation about a certain characteristic (for example, data related to the season, the dry season often occurs more than the rainy season, ..) Data is considered unbalanced when the balance ratio is 50:50, 60:40 is still considered balanced. A ratio greater than the stated ratio is considered unbalanced. The severe imbalance is 90:10. Dealing with imbalance dataset, we have a few methods:

- **Method 1 - Collect more data to correct the imbalance ratio:** with image data: crawl data from google photos, and Pinterest (a quite good effort to check images and related labels), use image creation models like GAN, and VAEs to create fake images, data augmentation,..., data Whether the language can be further created by replacing words in a sentence with a thesaurus, with null numeric data, simple models can be relied upon to link existing information such as linear regression,... In some specific areas, it is difficult to collect more data.
- **Method 2 - Use other metrics:** Accuracy is a commonly used metric to measure the accuracy of the model but it is not reliable enough, let's assume the case: if a data set has a ratio of 99:1, its accuracy if the model learns perfectly is 99%, not good for class with the smaller distribution. Thus we have other metrics. These metrics will not be too large to lead to misinterpretation of accuracy, and at the same time, they focus more on assessing the accuracy of the minority group, which we want to forecast more accurately than the majority group.
- **Method 3 - Sampling dataset:** is the steps to change the dataset so that the dataset is balanced with 2 main strategies: Undersampling: delete data from the majority class (for multiple datasets) and Oversampling: copy data from minority class (for small dataset). This is a fairly fast method and is often used when starting to solve problems.

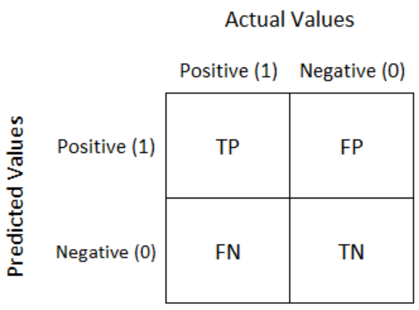


Figure 2: Some metric such as: precision, recall, F1-score, TPR construct from confusion matrix

- **Method 4 - Generate synthetic dataset:** The oversampling techniques of SMOTE and ADASYN are utilized to tackle imbalanced datasets by combining data samples to increase the minority class size, where the nearest neighbor samples are selected and linear combinations are performed to generate synthetic samples, and the neighbor selection process can be based on kNN or SVM algorithms.
- **Method 5 - Use other model**: Various algorithms and techniques have been developed, including decision trees, random forests, K-nearest neighbors (KNN), support vector machines (SVM), multilayer perceptron (MLP), and linear gradient boosting, each with their own strengths and weaknesses. The choice of algorithm depends on the specific characteristics of the dataset and analysis goals.
- **Method 6 - Collect more attributes**: A poor performing model may be a result of missing important variables that have a significant impact on determining the behavior of the minority group, highlighting the importance of domain knowledge for data scientists before building a model, as many important input variables may not be easily discernible without an understanding of the classification domain, and collecting expert opinions is an important measure to supplement variables and produce high-quality datasets for model training, known as feature selection.

Solving the problem of imbalanced datasets can help improve the accuracy and performance of machine learning models, as well as enable better identification and understanding of the minority class, which can have important practical applications, such as fraud detection, disease diagnosis, and predictive maintenance. Additionally, it can help prevent bias and discrimination, ensure fairness and equity in decision-making, and ultimately lead to more effective and impactful use of data in various fields and industries.

2.4 Understanding data

Data understanding begins with data exploration, which involves visually inspecting and identifying the data trends. Its aim is to get to know the data, identify patterns and relationships, and gain insights into the data's underlying trends and characteristics. A thorough understanding will help to make informed decisions about the business problem.

The data understanding step typically involves exploring the data using various techniques, including data visualization, data profiling, and data sampling. Data visualization techniques, such as scatter plots, histograms, and heat maps, can help analysts identify patterns and trends in the data. Data profiling techniques can provide insights into the data's structure, quality, completeness, and consistency. Data sampling techniques can be used to create subsets of the data to identify specific patterns or relationships.

Regarding data visualization, four of the most widely used libraries for this technique are Pandas, Numpy, Matplotlib, and Seaborn.

- NumPy is a fundamental library for scientific computing in Python that provides support for large, multi-dimensional arrays and matrices. It is the foundation for many other scientific computing libraries in Python, including Pandas and SciPy. NumPy provides efficient and high-performance numerical operations on arrays, such as element-wise operations, linear algebra, and Fourier transforms. It is also highly optimized for performance and memory usage, making it ideal for large-scale data analysis.
- Pandas is a powerful library for data manipulation and analysis that provides data structures and tools for working with structured data. It is built on top of NumPy and provides two main data structures, Series and DataFrame, that are designed to handle different types of data. Pandas offers a variety of powerful tools for data manipulation, including data selection, filtering, grouping, merging, and reshaping. These operations can be applied to the data structures in a flexible and intuitive way, making it easy to perform complex data manipulations with just a few lines of code.
- Matplotlib is a powerful library for creating static, animated, and interactive visualizations in Python. It provides a range of tools for creating plots, charts, histograms, and more. Matplotlib is highly customizable and can be used to create complex and beautiful visualizations. It can be used in combination with Pandas and NumPy to create sophisticated data visualizations.
- Seaborn is a library built on top of Matplotlib that provides a higher-level interface for creating statistical visualizations in Python. It provides a range of visualization types, such as scatter plots, line plots, and heatmaps, and includes support for complex statistical plots such as regression models and distribution plots. Seaborn also provides a range of customization options for creating visually appealing and informative plots.

2.5 Clustering

Clustering credit card users help the company to tailor their marketing strategies and offer personalized financial products and services to their customers. However, the high dimensionality of credit card usage data makes it challenging to obtain meaningful clusters using traditional clustering techniques. In this section, we will explore the use of Principal Component Analysis (PCA) in combination with clustering techniques, K-means, hierarchical clustering, and Gaussian mixture models, to cluster credit card users based on their spending patterns.

- Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of high-dimensional data by transforming the data into a lower-dimensional space. PCA identifies the directions in which the data varies the most and projects the data onto those directions. The new dimensions obtained through PCA are called principal components, and they contain most of the variation in the original data. PCA can help in reducing the noise and redundancy in the data, making it easier to analyze and visualize.
- K-means works by dividing a set of observations into a predetermined number of clusters (K), where each observation belongs to the cluster with the closest mean, each iteration the algorithm updates the mean of each cluster until it reaches convergence. Result of the K-means algorithm is a set of K clusters, where each data point belongs to one of the clusters.
- Hierarchical clustering is a clustering technique that groups data points into a tree-like structure called a dendrogram. The algorithm works by iteratively merging the closest pairs of data points or clusters until all data points belong to a single cluster. The result of hierarchical clustering is a tree-like structure that shows the relationships between the clusters. Hierarchical clustering can be agglomerative, where each data point is initially considered as a separate cluster and then merged, or divisive, where all data points are initially considered as a single cluster and then split recursively.
- Gaussian Mixture Model (GMM) is a probabilistic model used for clustering. GMM assumes that the data points are generated from a mixture of Gaussian distributions. The algorithm tries to estimate the parameters of the Gaussian distributions using the Expectation-Maximization (EM) algorithm. GMM is a soft clustering technique, which means that each data point has a probability of belonging to each of the clusters rather than being assigned to a single cluster.

3 IMPLEMENTATION

3.1 Feature selection

There are two kinds of features in this dataset: categorical and numerical. Therefore, we treat each of the two group separately using different algorithms.

The categorical attributes are `"SEX"`, `"EDUCATION"`, `"MARRIAGE"`, `"AGE_GROUP"`, `"PAY_0"`, `"PAY_2"`, `"PAY_3"`, `"PAY_4"`, `"PAY_5"`, `"PAY_6"` and the numerical attributes are `"LIMIT_BAL"`, `"BILL_AMT1"`, `"BILL_AMT2"`, `"BILL_AMT3"`, `"BILL_AMT4"`, `"BILL_AMT5"`, `"BILL_AMT6"`, `"PAY_AMT1"`, `"PAY_AMT2"`, `"PAY_AMT3"`, `"PAY_AMT4"`, `"PAY_AMT5"`, `"PAY_AMT6"`.

The new categorical `"AGE_GROUP"` is derived from the numerical feature `"AGE"`, where each bin is a range of 10, starting from 20. In this case, the bins are defined to be 20-29, 30-39, 40-49, 50-59, 60-69, and 70-79 years old.

The histogram for the `"AGE_GROUP"` is in Figure ??. It is clear that the majority of the customers fall into the age groups 2x and 3x, which are 20-29 and 30-39 years old, respectively. There are fewer customers in the older age groups, with a significant drop in the number of customers in the 5x age group (50-59 years old). Additionally, there are very few customers in the 6x and 7x age groups, indicating that there are very few older customers in the dataset. This information can be useful for understanding the age distribution of the dataset and for identifying potential age-related patterns in the data.

3.1.1 *ANOVA F-value For numerical feature selection.*

One-way ANOVA test can be used to find the relationship between numeric and a categorical variable

Here, we have the testing hypothesis:

- Null hypothesis H_0: two groups have the same variance.
- Alternate hypothesis H_1: at least one of the group have a different variance

If two groups have similar amounts of variation, it means that the character isn't significant. We can leave it out when selecting features. But if there's a difference in variance, we shouldn't leave out the feature.

The basic idea is that we will find $F{-}score = (variance_{between\ groups} / variance_{within\ groups})$ and compare it with the critical value obtained from the F-value table to accept or reject the null hypothesis.

If the value `variance_between / variance_within`, the p-value, is less than the critical value, for $p < 0.05$ since we mean that the confidence > 95%, it means that they belong to the same population and hence are co-related. We select the top k co-related features according to the score returned by Anova.

```
fvalue_selector = SelectKBest(f_classif, k=10)
fvalue_selector.fit(X_num_scaled, y)

df_score =
     pd.DataFrame(fvalue_selector.pvalues_,columns=['p_values'])
df_score['score'] = fvalue_selector.scores_
df_score['columns'] = X_num_scaled.columns

df_score.sort_values(by='score', ascending=False,
     inplace=True)
df_score
```

The result data frame is shown in Table 1.

idx	p_values	score	columns
0	$1.302244 \times 10^{-157}$	724.068539	LIMIT_BAL
7	1.146488×10^{-36}	160.403810	PAY_AMT1
8	3.166657×10^{-24}	103.291524	PAY_AMT2
10	6.830942×10^{-23}	97.188000	PAY_AMT4
9	1.841770×10^{-22}	95.218011	PAY_AMT3
11	1.241345×10^{-21}	91.429801	PAY_AMT5
12	3.033589×10^{-20}	85.089045	PAY_AMT6
1	6.673295×10^{-4}	11.580532	BILL_AMT1
2	0.01395736	6.044238	BILL_AMT2
3	0.01476998	5.944388	BILL_AMT3
4	0.07855564	3.094745	BILL_AMT4
5	0.2416344	1.371087	BILL_AMT5
6	0.3521225	0.865820	BILL_AMT6

Table 1: F-value For numerical feature selection

From the result table, we get lists of features where the p-value < 0.05:

```
selected_num_features = df_score[df_score['p_values'] <
    0.05]['columns'].tolist()
```

The selected numerical features are 'LIMIT_BAL', 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT4', 'PAY_AMT3', 'PAY_AMT5', 'PAY_AMT6', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3'.

3.1.2 Chi-Square test for categorical feature selection. We calculate chi-squared scores between each of the non-negative categorical features with the target variable. This score measures how the distribution of target class Y across different categories of the input feature X compared to the distribution of the target variable.

```
chi_selector = SelectKBest(score_func=chi2, k=5)
fit = chi_selector.fit(X_cat_encoded, y)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(X_cat_encoded.columns)

f_scores = pd.concat([df_columns, df_scores], axis=1)
f_scores.columns = ['Feature', 'Score']
f_scores
```

The result is listed in Table 2. Based on these scores, we can see that the PAY_0, PAY_2, PAY_3, PAY_4 features have the highest scores, indicating a strong association with the target variable. The SEX, EDUCATION, MARRIAGE, and AGE_GROUP features have relatively lower scores, indicating a weaker association with the target variable.

Feature	Score
SEX	18.983326
EDUCATION	7.931259
MARRIAGE	3.120010
AGE_GROUP	7.824658
PAY_0	2015.175489
PAY_2	1600.272996
PAY_3	1296.927064
PAY_4	1081.317089
PAY_5	421.891456
PAY_6	367.152234

Table 2: Chi-square scores for categorical features

In conclusion, the selected categorical features are 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5'.

3.1.3 Embedded method with tree-based model. Embedded methods are a class of feature selection techniques that perform feature selection during the model training process. Tree-based models like XGBoost use embedded methods to identify the most important features for the given task. During the model training process, XGBoost calculates a score for each feature that reflects its importance in the decision-making process. This score is based on how much each feature reduces the objective function in each split of the decision tree. Hence, we can use XGBoost to extract important features.

```
import xgboost
import matplotlib.pyplot as plt

model = xgboost.XGBClassifier(seed=42)
X_train = pd.concat([X_cat_encoded, X_num_scaled], axis=1,
    ignore_index=True)
model.fit(X_train, y)

# plot the graph of feature importances for better
    visualization
feat_imp = pd.Series(model.feature_importances_,
    index=X.columns)
feat_imp.nlargest(20).plot(kind='barh')

plt.figure(figsize=(8,6))
plt.show()
```

The top-20 score features based on the XGBoost model are visualized in Figure 3.

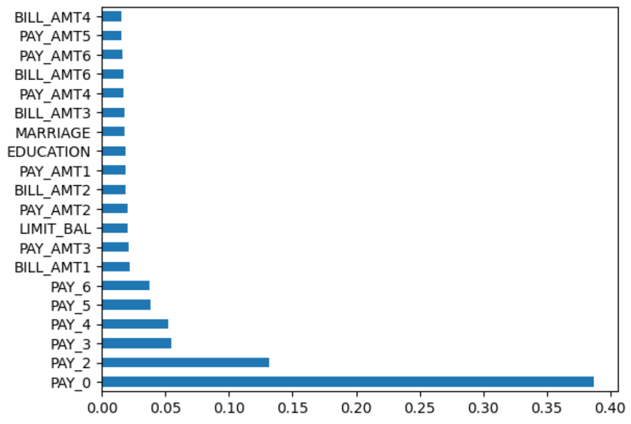


Figure 3: Feature importance ranking by XGBoost Classifier

Compared with the ANOVA test and Chi-square test, the embedded method with XGBoost has a small difference. PAY_AMT4, PAY_AMT5, PAY_AMT6 are not considered as key features by the tree-based model. EDUCATION, MARRIAGE, and PAY_6 are not important via the statistical models.

3.2 Predict bill pay next month

We will first extract the data related to the feature selected in the previous step, and take the "default payment next month" column as the label, If the value is 0, the customer is likely not to pay, and vice versa. We will plot how the data imbalance rate is:

We recognize that the severe class imbalance requires handling to improve model performance; thus, we will first normalize the continuous variables to a specific range to prevent them from exceeding the bounds of the learning model. Next, we partition the dataset into three sets: train, validation, and test, where the validation set is used to assess the baseline model's performance, which will be used to apply various methods to address the data imbalance.

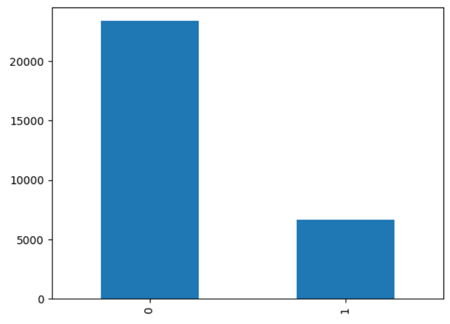


Figure 4: Ratio approximate 80/20

```
//train/val/test is: 80/10/10
//split id
id_pos = np.where(y.values.reshape(-1)==0)[0]
id_neg = np.where(y.values.reshape(-1)==1)[0]

//shuffle id
np.random.shuffle(id_pos)
np.random.shuffle(id_neg)

//train
id_train_neg = id_neg[:int(len(id_neg)*0.8)]
id_train_pos = id_pos[:int(len(id_pos)*0.8)]
id_train = np.concatenate((id_train_neg,
     id_train_pos),axis=0)

//val
id_val_neg = id_neg[int(len(id_neg) *
     0.8):int(len(id_neg)*0.9)]
id_val_pos = id_pos[int(len(id_pos) *
     0.8):int(len(id_pos)*0.9)]
id_val = np.concatenate((id_val_neg, id_val_pos),axis=0)

//test
id_test_neg = id_neg[int(len(id_neg) * 0.9):]
id_test_pos = id_pos[int(len(id_pos) * 0.9):]
id_test = np.concatenate((id_test_neg, id_test_pos),axi =0)
```

We will create a baseline model based on the non-parametric method of Random Forest, which will run GridSearch to find the necessary hyperparameters and use the validation set on the f1-score metric to select the best solution.

```
//create pipeline to run model
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier())
])

//set parameter to grid-search use
params = {
    'classifier__n_estimators': [100, 500, 800],
    'classifier__max_depth': [5, 10],
    'classifier__min_samples_split': [200, 400],
    'classifier__class_weight' : [None,"balanced"],
    'classifier__max_features' : [10,"auto","sqrt"]
}
grid_search = GridSearchCV(pipeline, params, cv=5,
     scoring='f1')
grid_search.fit(data_train[features],
     data_train[target].values.ravel())

//run and best solution is model
model_baseline = RandomForestClassifier(n_estimators=500,
                             max_depth=10,
                             min_samples_split=200,
                             random_state=12,
                             class_weight="balanced",
                             max_features="auto")
```

The first method the team wants to use is undersampling, where we will sequentially undersampling the data with ratios of 30:70, 40:60, and 50:50 and then use the generated data to train the model.

```
//use example setting with ratio: 30/70
//set up data
np.random.shuffle(id_train_pos)
id_train_pos_30_70 = id_train_pos[:12835]
id_train_30_70 = np.concatenate((id_train_pos_30_70,
     id_train_neg), axis = 0)
data_train_30_70 = df.iloc[id_train_30_70]

//train model
under_sampling_30_70 =
train_and_get_metrics(model_baseline,data_30_70)
```

The second method we will use is oversampling through techniques such as naive random oversampling and data generation methods like ADASYN and SMOTE. Here, we need to set up a pipeline to both generate new data and feed it into the training process.

```
//Naive random over-sampling
naive_random_oversampling =
     make_pipeline(RandomOverSampler(sampling_strategy=1,
     random_state=0), model_baseline)
NAIVE_RANDOM_O =
     train_and_get_metrics(naive_random_oversampling,dataset)

//SMOTE over-sampling
smote_random_model = make_pipeline(SMOTE(sampling_strategy
     = 1, random_state=0), model_baseline)
SMOTE_RO = train_and_get_metrics(smote_random_model,dataset)
```

Another testing method that we will consider is penalizing the model by increasing the weight for the minority class and decreasing the weight for the majority class. We will adjust the weights by creating a dictionary specifying the desired weights in the class_weight parameter of the model. The weight ratios we will test are: 0: 0.1, 1: 0.9, 0: 0.2, 1: 0.8, and 0: 0.3, 1: 0.7.

Finally, we will experiment with other non-parametric models such as linear gradient boosting and XGBoost. Additionally, we will extract numerical features separately for SVM and logistic regression models to learn and compare with the methods we used above.

```
model_pen2 = RandomForestClassifier(n_estimators=500,
                                max_depth=10,
                                min_samples_split=400,
                                random_state=12,
                                class_weight={0: 0.2,
                                              1: 0.8},
                                max_features="auto")
penalty_method = train_and_get_metrics(model_pen2,dataset)

\\XGBOOST TRAINING
model_xgb = xgb.XGBClassifier(random_state=42, n_estimators
     = 800)
xgb_method = train_and_get_metrics(model_xgb,dataset)
```

After completing the experiments, we will have a table comparing the results of each model with the dataset. We will select the models with the best ROC and F1-scores as the main models. We will also visualize the results using charts and graphs to provide better insights, which will be discussed in the results section.

3.3 Clustering

3.3.1 Choose the number of clusters. For each combination of the number of components of PCA and the number of clusters, we performed PCA on the training data and transformed the testing data, then applied K-means clustering on the transformed testing data and computed the Silhouette score for each data point. The Silhouette score measures how well each data point fits into its assigned cluster and ranges from -1 to 1, with higher values indicating better cluster assignments. The average Silhouette score is used across all data points as the evaluation metric for each combination of hyperparameters. With $n_{compoments} = 2$ and $n_{cluster} = 5$ yield the best result.

4 RESULTS AND CONCLUSION

4.1 Graphs and Data Visualization

(1) Histograms We first draw histograms in order to have a visualization of the data that is measured on an interval scale of other data. Based on these histograms, by analyzing the shape and distribution of the graphs, we would want to get some insights and understanding into the attributes that are drawn together.

(a) Histogram of default payments based on age

According to the data, the group most likely to default on payments are young individuals aged between 25 and 35, with a frequency of more than 1150. This group is followed by people aged between 20 and 25, who have a frequency of around 1000. The third group most likely to default is middle-aged individuals aged between 35 and 45, with a frequency of approximately 900. As individuals get older, the frequency of default payments decreases.

(b) Histogram of default payments based on customers' limit balance
Based on the chart, it can be observed that individuals with lower limit balances are more prone to default. Conversely,

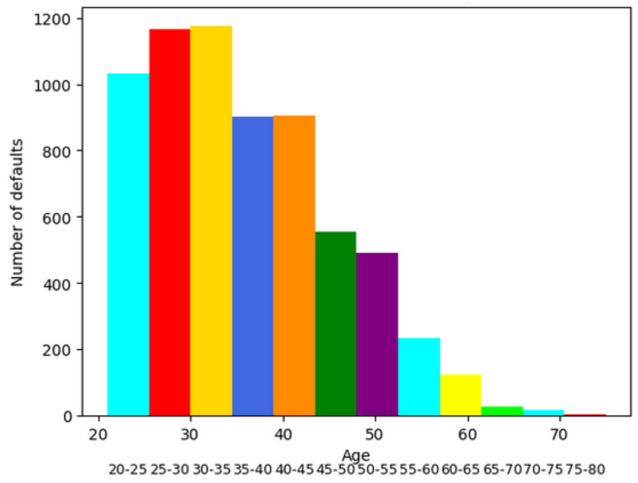


Figure 5: Age of default payment customers

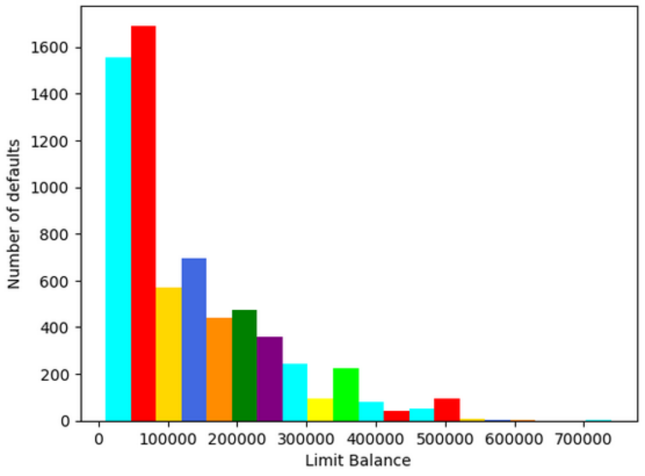


Figure 6: Limit balance of default payment customers

those with higher limit balances are less likely to default. The largest number of defaults occurred in the group with limit balances ranging from $50, 000 to $60, 000, which totaled over 1,600 defaults. The second largest group of defaults came from those with limit balances ranging from $0 to $50,000, which totaled around 1,570 defaults. As the limit balance increases from $100,000, there is a clear decline in the number of defaults.

(c) Scatter plot We would then want to draw some scatter plots in order to analyze the relationship and trends between the two variables that are being drawn. Specifically, the graph will display the correlation that shows what happens to one variable when the other variable change.
The scatter plot drawn above is between the customer's payment amount on the x-axis with the customer's bill payment of that corresponding month on the y-axis where the graphs are ordered from left to right, upper to bottom

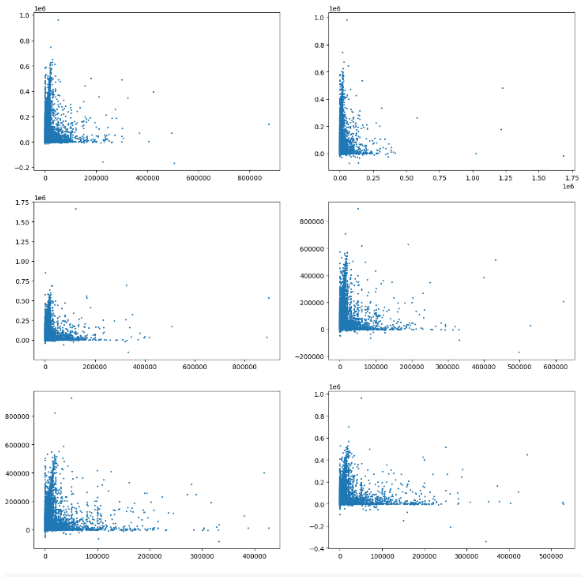


Figure 7: Monthly payment amount vs. the corresponding monthly bill payment

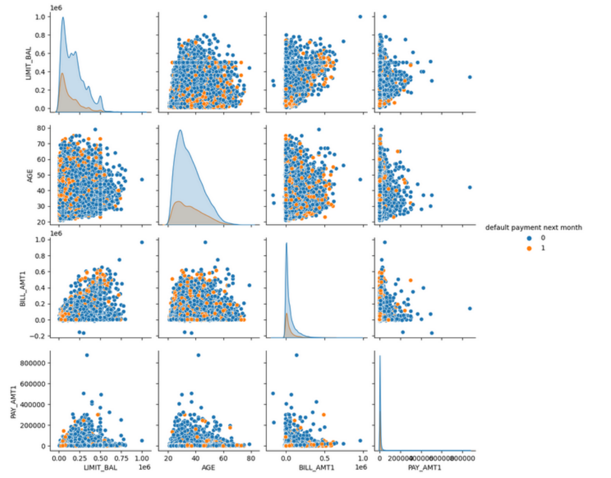


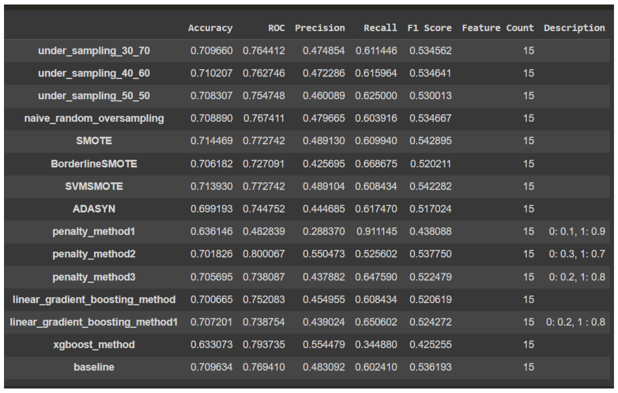
Figure 8: Pair plot between customers' limit balance, age, bill payment, and payment amount

based on the months. We can observe that there is a higher proportion of customers whose bill payment 1 (bill payment in September 2005) is high compared to the proportion of customers whose bill payment is high in the later months. However, the customer's payment amount has a higher distribution in the graph of payment 6 (payment in April 2005). From these observations, it is shown that the amount of customers' monthly payments decreases over time while the bill payments amount are increasing monthly.

(d) Pair plot Pair plots are being graphed to have a better understanding of the distribution of each attribute as well as the relationship of the attributes that are being graphed through the pairwise plot method.
By focusing on the pair plot between customers' limit balance and payment amount, we can observe that the distribution of customers who default on the next month's payment lies is higher when the limit balance is low and the payment amount of the previous month is low. Furthermore, having a low bill amount but with a higher limit balance would result in more customers being able to pay off their payment in the next month based on the pair plot between the limit balance and bill amount.

4.2 Feedback On A Model Predict "Payment Next Month"

After running multiple methods with the baseline model and experimenting with various different models, a table of results was obtained by applying metrics such as accuracy, f1-score, ROC, precision, and recall. The use of multiple metrics provides an overall view of the model's performance, from which the best model for the specific problem can be selected.



	Accuracy	ROC	Precision	Recall	F1 Score	Feature Count	Description
under_sampling_30_70	0.709660	0.764412	0.474854	0.611446	0.534562	15	
under_sampling_40_60	0.710207	0.762746	0.472286	0.615964	0.534641	15	
under_sampling_50_50	0.708307	0.754748	0.460089	0.625000	0.530013	15	
naive_random_oversampling	0.708890	0.767411	0.479665	0.603916	0.534667	15	
SMOTE	0.714469	0.772742	0.489130	0.609940	0.542895	15	
BorderlineSMOTE	0.706182	0.727091	0.425695	0.668675	0.520211	15	
SVMSMOTE	0.713930	0.772742	0.489104	0.608434	0.542282	15	
ADASYN	0.699193	0.744752	0.444685	0.617470	0.517024	15	
penalty_method1	0.636146	0.482839	0.288370	0.911145	0.438088	15	0: 0.1, 1: 0.9
penalty_method2	0.701826	0.800067	0.550473	0.525602	0.537750	15	0: 0.3, 1: 0.7
penalty_method3	0.705695	0.738087	0.437882	0.647590	0.522479	15	0: 0.2, 1: 0.8
linear_gradient_boosting_method	0.700665	0.752083	0.454955	0.608434	0.520619	15	
linear_gradient_boosting_method1	0.707201	0.738754	0.439024	0.650602	0.524272	15	0: 0.2, 1 : 0.8
xgboost_method	0.633073	0.793735	0.554479	0.344880	0.425255	15	
baseline	0.709634	0.769410	0.483092	0.602410	0.536193	15	

Figure 9: Result with run different method and model

Based on the results, we have selected the two best-performing models with the highest ROC and F1-score: the model with a penalty of 0.3 and 0.7 (highest ROC score), and the model using SMOTE (highest F1-score). We will now plot the ROC curve for each model and make some observations.

Based on the observations, we can see that the model only has a relatively decent accuracy and F1-score, which is around 0.54, while the Baseline model has achieved an accuracy of 0.53. Additionally, the high ROC score may depend heavily on the majority class, which may not fully reflect the impact of the minority class. We also noticed that there are still some outliers present after clustering, which may affect the results of the group. However, the model can still provide some insights into customer behavior, and if given more time, the team can try other methods such as removing outliers from the data for training, using numerical features to apply to logistic regression, MLP, or dividing customers into groups and training on each group separately.

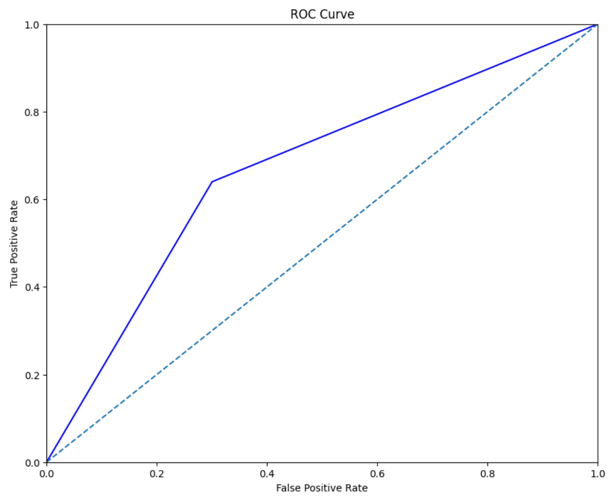


Figure 10: ROC chart with Penalty2 Model

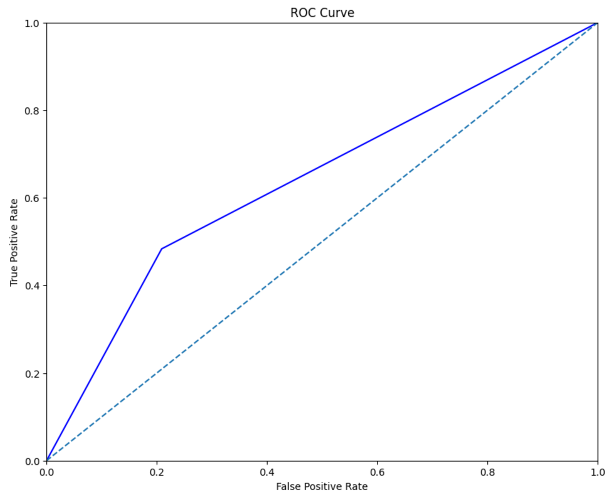


Figure 11: ROC chart with SMOTE Model

4.3 Clustering Result

After trying other clustering algorithms such as Gaussian Mixture Models (GMM) and Hierarchical Clustering, it is important to analyze the results of the K-Means clustering algorithm because of its interpretability. K-Means provides easily interpretable clusters that can help us understand the underlying structure of the data and identify any patterns or trends that may be present.

Additionally, we can use the K-Means results to inform further analysis and decision-making. For example, we can identify high-risk customers in the high-risk cluster and take steps to mitigate the risk of default, such as offering credit counseling or adjusting credit limits. We can also use the cluster assignments to segment customer groups for targeted marketing or to identify areas for improvement in the bank's risk management strategies.

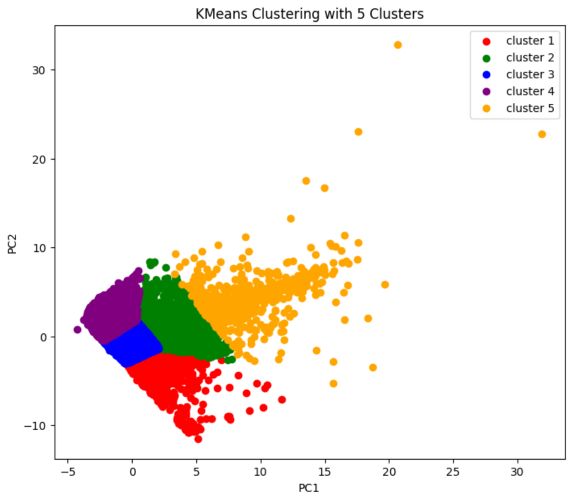


Figure 12: Clustering with K-means

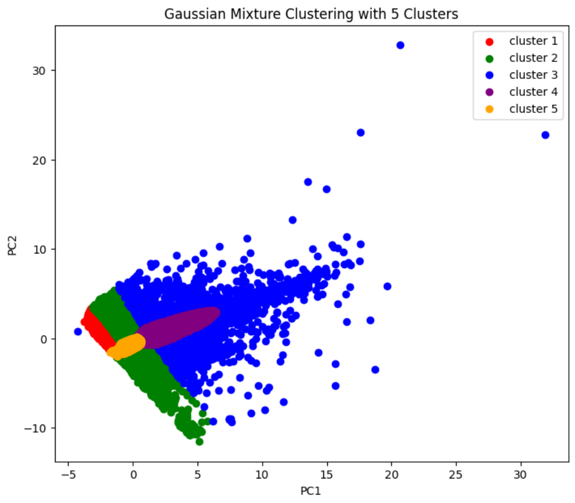


Figure 13: Clustering with GMM

- High-risk customers with low credit limits, low bill payments, and low previous payments
- Medium-risk customers with low bill payments and high previous payments
- Low-risk customers with aggressive bill payments and high previous payments
- Low-risk customers with low bill payments, low previous payments, and high credit limits
- Customers with high bill payments in the beginning but reduced over time due to high previous payments, who may be low risk but require further analysis to determine their risk level.

5 REFERENCES

[1]: **Feature Selection Methods and How to Choose Them** https://neptune.ai/blog/feature-selection-methods

[2]: **Feature Selection Techniques in Machine Learning** https://www.kaggle.com/code/piyushagni5/feature-selection-techniques-in-machine-learning

[3]: **Comprehensive Guide on Feature Selection** https://www.kaggle.com/code/prashant111/comprehensive-guide-on-feature-selection/notebook

[4]: **Best way to apply Feature Selection Techniques in ML.** https://www.kaggle.com/getting-started/186082

[5]: **Somte for handling imbalance dataset:** https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[6]: **5 Techniques to Handle Imbalanced Data For a Classification Problem** https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[7]: **K-Means Clustering in Python: A Practical Guide** https://realpython.com/k-means-clustering-python/