Credit Scoring Project

Pınar Yazgan

May 2021

1 Introduction

Credit risk management is vital for banks to survive. Therefore, before the loan application and disbursement process, the study of predicting the behavior of the consumer who will use credit has a critical place in the finance and banking sector. In this context, it is a definite need to determine whether the loan is payable or not. It is envisaged to be able to give repayable loans with machine learning algorithms and the loan application process is expected to be completed more consciously and correctly. Credit scoring transactions are carried out between the bank and the customer in the loan application. The purpose of this is basically to evaluate whether people will actually pay off the loan they will receive. This is called credit scoring in machine learning. After the transactions, a positive or negative feedback is made to the person applying for the loan. There are many metrics that evaluate in this direction such as status of existing checking account of the customer, Duration of the credit (requested by the customer from the bank) in month, Purpose of the credit, Credit amount, Savings account/bonds of the customer, Present employment of the customer since, Installment rate in percentage of disposable income, Other debtors or guarantors for the credit, Present residence of the customer since (in years), Property owned by the customer, Age of the customer, Housing situation of the customer, Job situation of the customer, Number of existing credits of the customer at this bank, if the customer is a foreign worker...

2 Objective

German Credit Dataset classifies 1000 people described by a set of contains categorical/symbolic attributes as good or bad credit risk. My goal is to predict if this loan credit would be a risk for the bank or not and if the loan amount is given to the applicant, will they pay back or will not pay back? There are many loan applications that need to be examined every day. It would be helpful to develop a Machine Learning predictive model that helps managers in approving or rejecting a loan application. In below, for this purpose I applied various classification, clustering algorithms and Frequent pattern mining algorithm to uncover some patterns in the data.

3 Problem Statement

Create a Predictive model which can tell us approve a loan application or not?
• Target Variable: Risk
• Predictors: Property owned by the customer, Age of the customer, Housing situation of the customer, Job situation of the customer etc.
• Risk=1 means the loan was a good decision.
• Risk=0 means the loan was a bad decision.

4 Exploratory Data Analysis (EDA)

Exploratory Data Analysis helps us to understand the overall data. The volume of data, the types of columns etc.
Firstly; I looked how many missing values are there for each column using isnull().sum(). I saw that there is no null data in my dataset.
Then I looked distribution of target variable with value_counts(). Because If target variable's distribution is too skewed then the predictive modeling cant be possible. So when making Classification, I have to make sure there is a balance in the distribution of each class otherwise it impacts the Machine Learning algorithms ability to learn all the classes.
I saw 700 good customer and 300 bad customer in target data. The data distribution of the target variable is satisfactory and there are sufficient number of rows for each category to learn from. Then I identified which columns are Quantitative, Categorical or Qualitative. This helps for selection of columns.
I looked column type and Uniquecount to identify which columns are Quantitative, Categorical or Qualitative using info() and nunique().

Column Name	Column Type	UniqueCount
Credit amount	int64	921
Duration in month	int64	33
Age in years	int64	53
Number of existing credits at this bank	int64	4
Number of people being liable to provide maintenance for	int64	2
Installment rate in percentage of disposable income	int64	4
Present residence since	int64	4
Status of existing checking account	object	4
Credit history	object	5
Purpose	object	10
Savings account/bonds	object	5
Present employment since	object	5
Personal status and sex	object	4
Other debtors / guarantors	object	3
Property	object	4
Other installment plans	object	3
Housing	object	3
Job	object	4
Telephone	object	2
foreign worker	object	2
Risk	object	2

Table 1: Column type and unique count

I saw credit amount,duration in month and age in years columns are continuous quantitative. Although Number of existing credits at this bank, Number of people being liable to provide maintenance for , Installment rate in percentage of disposable income , Present residence since columns are quantitative but their unique counts are too low. So I thought these columns are qualitative like categorical object type columns. This helped for selection of columns. Then, I looked at each column carefully and ask, does this column affect the Target variable? Does this column affect the approval or rejection of loan? If the answer is "No" I removed the column from the data.

5 Feature Selection

Feature selection is a process where we automatically select those features in data that contribute most to the prediction variable or output. In below, I defined variables as Categorical Vs Continuous and applied statistical feature selection methods accordingly.

5.1 Statistical Feature Selection (Categorical Vs Continuous) Using ANOVA Test

Analysis of variance(ANOVA) is performed to check if there is any relationship between the given continuous and categorical variable.

Assumption(H0): There is NO relation between the given variables.

ANOVA Test result: Probability of H0 being true.
I made variance analysis using anova test for continuous columns (credit amount,duration in month and age in years) and target variable(Risk). If the ANOVA P-Value is < 0.05, that means we can reject H0 and we can say this column is correlated with target variable otherwise is NOT correlated with TargetVariable.

ANOVA Test Results:
Credit amount is correlated with Risk — P-Value: 8.797572373533373e-07
Duration in month is correlated with Risk — P-Value: 6.488049877187189e-12
Age in years is correlated with Risk — P-Value: 0.003925339398278295
I found ['Credit amount', 'Duration in month', 'Age in years'] are correlated with Risk. So I can use these columns for developing a prediction model to classify the customers as good or bad.

5.2 Statistical Feature Selection (Categorical Vs Categorical) Using Chi-Square Test

Chi-Square test is conducted to check the correlation between two categorical variables.

Assumption(H0): The two columns are NOT related to each other.

Chi-Square Test Result: The Probability of H0 being True.

I made Chi-Square test to check the correlation between Categorical columns and target data. If the P-Value is < 0.05, that means we can reject H0 and we can say this column is correlated with target variable. Otherwise this column is NOT correlated with Target(Risk) variable.

Chi-Square Test Results:

• Status of existing checking account is correlated with Risk — P-Value: 1.2189020722893755e-26
• Credit history is correlated with Risk — P-Value: 1.2791872956750918e-12
• Purpose is correlated with Risk — P-Value: 0.00011574910079691635
• Savings account/bonds is correlated with Risk — P-Value: 2.761214238568249e-07
• Present employment since is correlated with Risk — P-Value: 0.0010454523491402522
• Personal status and sex is correlated with Risk — P-Value: 0.02223800546926877
• Other debtors / guarantors is correlated with Risk — P-Value: 0.036055954027247206
• Property is correlated with Risk — P-Value: 2.8584415733250017e-05
• Other installment plans is correlated with Risk — P-Value: 0.0016293178186473534
• Housing is correlated with Risk — P-Value: 0.00011167465374597684
• Job is NOT correlated with Risk — P-Value: 0.5965815918843431
• Telephone is NOT correlated with Risk — P-Value: 0.2788761543035742
• foreign worker is correlated with Risk — P-Value: 0.015830754902852885
• Installment rate in percentage of disposable income is NOT correlated with Risk — P-Value: 0.1400333122128481
•Present residence since is NOT correlated with Risk — P-Value: 0.8615521320413175
• Number of existing credits at this bank is NOT correlated with Risk — P-Value: 0.4451440800083001
• Number of people being liable to provide maintenance for is NOT correlated with Risk — P-Value: 1.0

I found ['Status of existing checking account','Credit history','Purpose','Savings account/bonds','Present employment since', 'Personal status and sex', 'Other debtors / guarantors', 'Property', 'Other installment plans', 'Housing','foreign worker'] columns are correlated with Target variable Risk. So I can use these columns for developing a prediction model.
After feature selection part I removed columns are not correlated with Risk And I saw that [credit amount,duration in month, age in years columns, 'Status of existing checking account','Credit history','Purpose','Savings account/bonds','Present employment since', 'Personal status and sex', 'Other debtors / guarantors', 'Property', 'Other installment plans', 'Housing','foreign worker'] columns are correlated with Risk column. So I used these columns in ML algorithms to classify the customers as good or bad.

6 Data Preprocessing For Machine Learning

List of Data Preprocessing steps before data can be used for machine learning algorithms.
1.Converted Binary nominal Categorical columns (foreign-worker,Risk) to numeric using get_dummies().
Because One-hot encoding converts it into n variables, while dummy encoding converts it into n-1 variables. For binary columns 1 variable is sufficient.
2.Converted other categorical columns to numeric using OneHotEncoder.
3.I applied MinmaxScalar Normalization technique in this part. The goal of normalization is to make every datapoint has the same scale so each feature is equally important and equally affect the result. For this reason, I applied minmaxScalar Normalization on numeric variables (credit amount,duration in month, age in years columns) for producing better results. Using MinmaxScalar for every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1. Normalization is expecially important when using distance based algorithms like KNN, or Neural Networks.

After applied preprocessing part I got preprocessing datasets(dfx_processed, dfy_processed) for using ML algorithms.

7 Classification

Classification is a supervised learning concept where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under. In this project I developed a prediction model to classify the customers as good or bad. Many machine learning algorithms cannot work with categorical data directly. So I converted the categories into numbers as I mentinoned above. Then I applied 4 classification models (Decisiontree,Randomforest,Knn,Svm)
List of process for these 4 classification models.

1.Firstly I applied the train-test split procedure (test_size=0.3) to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.
2.Defined the range of possible values for all hyperparameters.
3.Applied models using hyper parameter sets.
4.Selected best hyper parameters sets with hyper parameter tuning.I prefered GridSearch approach for hyperparameter tuning.

Hyperparameter tuning is the process of determining the right combination of hyperparameters that allows the model to maximize model performance. This technique chooses a set of optimal estimators from each algorithm that (might) produces the highest accuracy score on the given dataset. GridSearchCV is an approach to hyperparameter tuning that will methodically build and evaluate a model. GridSearchCV method helps finding the set of optimal estimators in each algorithm.

5.Created Model on training data using best hyper parameter sets.
6.Measured accuracy score and f1 score on test data and printed confusion-matrix.
7.Imported k-fold cross-validation method and run 10 fold cross validation on a given algorithm.

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. It gives our model the opportunity to train on multiple train-test splits. K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called folds. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set.
8.Passed full data X and y because the K fold will split the data and automatically choose train/test.
9.Measured accuracy score and f1 score with using 10 fold cross validation.

7.1 Which Performance Metrics To Choose?

Accuracy: It's the ratio of the correctly labeled subjects to the whole pool of subjects. Accuracy is the fraction of predictions our model got right.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

Precision: Precision is a measure for the correctness of a positive prediction. Attempts to answer the following question: What proportion of positive identifications was actually correct?

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: Recall – or the true positive rate – is the measure for how many true positives get predicted out of all the positives in the dataset.Attempts to answer the following question: What proportion of actual positives was identified correctly?

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1 score: A good F1 score means that you have low false positives and low false negatives. An F1 score is considered perfect when it's 1 , while the model is a total failure when it's 0 .

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

7.2 DecisionTree

Decision tree is one of the most powerful and popular algorithm for classification and prediction. A decision tree typically starts with a single node, which branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch off into other possibilities. This gives it a treelike shape. Decision trees are built using a heuristic called recursive partitioning. This approach is also commonly known as divide and conquer.
Basic Divide-and-Conquer Algorithm :
1.Select a test for root node.
2.Create branch for each possible outcome of the test.
3.Split instances into subsets.
4.One for each branch extending from the node.
5.Repeat recursively for each branch, using only instances that reach the branch.
6.Stop recursion for a branch if all its instances have the same class.

I tried to find the best hyperparameters by doing a grid search with the values shown below for the decision tree algorithm.

max_features: ['auto', 'sqrt', 'log2'], The number of features to consider when looking for the best split.
min_samples_split: [2,3,4,5,6,7,8,9,10,11,12,13,14,15], The minimum number of samples required to split an internal node
min_samples_leaf: [1,2,3,4,5,6,7,8,9,10,11], This is the minimum number of samples, or data points, that are required to be present in the leaf node

After my analysis, I saw that my decision tree model will work best with the following hyperparameters.

Result: max_features: log2, min_samples_leaf: 3, min_samples_split: 9, random_state: 123
I selected these hyperparameters and created the model on Training Data. Measured accuracy and f1 score on Testing Data and looked confusion matrix.
Train Test Split Approach Results:
Accuracy score: 0.69
F1 score: 0.78
Confusion Matrix:
[[44 52]
[42 162]]
This confusion matrix show us that our model gave real labels for 206 samples. 94 samples are incorrectly labeled by our model.
Kfold Cross Validation Approach Results:
Then I imported k fold cross validation function. Run 10 Fold Cross validation in this algorithm. Passed full data X and y because the K-fold will split the data and automatically choose train/test.

Accuracy values for 10 K-fold Cross Validation: [0.67 0.7 0.71 0.77 0.72 0.68 0.68 0.71 0.67 0.66]
Final average accuracy score for 10-kfold Cross Validation: 0.7
F1 values for 10-kfold Cross Validation: [0.68669654 0.68690476 0.72224618 0.77121335 0.7088 0.66402852 0.68284245 0.69929088 0.6716264 0.6626142]
Final average F1 score for 10-kfold Cross Validation:0.7
I saw that cross validation increased accuracy score and model performance.

7.3 RandomForestClassifier

The random forest is a classification algorithm consisting of many decision trees. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset.
I tried to find the best hyperparameters by doing a grid search with the values shown below for the Random Forest algorithm.

criterion: ['gini','entropy'], The function to measure the quality of a split.
n_estimators: [10,15,20,25,30], Number of trees
min_samples_leaf: [1,2,3], This is the minimum number of samples present in the leaf node.(external node)
min_samples_split: [3,4,5,6,7], The minimum number of samples required to split an internal node
n:[-1]: This means that the computation will be dispatched on all the CPUs of the computer. After my analysis, I have determined that my random forest model will work best with the following hyperparameters.
Result: criterion='entropy', min_samples_leaf=2, min_samples_split=3, n_estimators=30, n_jobs=-1
Train Test Split Approach Results:
Accuracy Score: 0.73
Confusion Matrix:
[[37 59]
[21 183]]
This confusion matrix show us that our model gave real labels for 220 samples. 80 samples are incorrectly labeled by our model.
F1 score: 0.82
kfold Cross Validation Approach Results:
Accuracy values for 10-kfold Cross Validation:
[0.8 0.72 0.79 0.77 0.75 0.73 0.75 0.79 0.76 0.78]
Final Average Accuracy score for 10-kfold Cross Validation: 0.76
F1 values for 10-kfold Cross Validation:
[0.8 0.69386667 0.79154777 0.77121335 0.72831541 0.70800834 0.73944295 0.77239637 0.7472 0.77326858]
Final Average F1 score for 10-kfold Cross Validation: 0.75
I saw that cross validation increased accuracy score and model performance.

7.4 Support Vector Classification-SVM

SVM or Support Vector Machine is an algorithm that is used in classification and regression problems. This algorithm creates a line or a hyperplane which separates the data into classes so that we can easily put the new data point to the correct category in the future. Maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyperplane. This distance is called as Margin. The hyperplane for which the margin is maximum is the optimal hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

I tried to find the best hyperparameters by doing a grid search with the values shown below for the Support Vector Machine algorithm.
C (Regularisation): [6,7,8,9,10,11,12] C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimisation how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. When C is high it will classify all the data points correctly, also there is a chance to overfit.
Kernel: ['linear','rbf'], The main function of the kernel is to take low dimensional input space and transform it into a higher-dimensional space. It is mostly useful in non-linear separation problem.
Gamma: It defines how far influences the calculation of plausible line of separation.When gamma is higher, nearby points will have high influence; low gamma means far away points also be considered to get the decision boundary. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class). Therefore, models with very large gamma values tend to overfit.After my analysis, I have determined that my SVM model will work best with the following hyperparameters.
Result:'C': 6, 'kernel': 'linear'
Train Test Split Approach Results:
Accuracy score: 0.75
Confusion Matrix:
[[46 50]
[24 180]]
This confusion matrix show us that our model gave real labels for 226 samples. 74 samples are incorrectly labeled by our model.
F1 score: 0.83
kfold Cross Validation Approach Results:
Accuracy values for 10 kfold Cross Validation:
[0.84 0.64 0.76 0.77 0.75 0.77 0.63 0.8 0.77 0.77]
Final Average Accuracy values for 10 kfold Cross Validation: 0.76
F1 values for 10 kfold Cross Validation:
[0.8351214 0.61305613 0.75608392 0.7658375 0.73341677 0.75440456 0.61956113 0.79388052 0.76610658 0.76150656]
Final Average F1 score for 10 kfold Cross Validation:0.75

7.5 K Nearest Neighbors

K Nearest Neighbors stores all available cases and classifies new cases based on a smilarity measure(distance function).Distance functions are euclidean,manhattan, minkowski. A new instance is classified by a majority votes for its neighbor classes. The object is assigned to the most common class amongst its K nearest neighbors (measured by a distance function)
To classify the unknown data point using the KNN algorithm:
1.Normalize the numeric data then find the distance between the unknown data point and all training data points.
2.Sort the distance and find the nearest k data points.
3.Classify the unknown data point based on the most instances of nearest k points.
Firstly I tuned hyperparameters of algorithms using GridSearchCV. Then I used Cross validation for testing the effectiveness of a machine learning models.
I tried to find the best hyperparameters by doing a grid search with the values shown below for the K Nearest Neighbors algorithm.
n_neighbors: list(range(1, 10)), Number of neighbors to use by default for kneighbors queries.
leaf_size: list(range(1,10)), the memory required to store the tree.
weights: ['distance', 'uniform']
uniform: Uniform weights. All points in each neighborhood are weighted equally.
distance: Weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
algorithm: ['auto', 'ball_tree','kd_tree','brute']
After my analysis, I have determined that my KNeighbors model will work best with the following hyperparameters.
Result:'algorithm': 'auto', 'leaf_size': 1, 'n_neighbors': 7, 'weights': 'uniform'
Train Test Split Approach Results:
Accuracy score: 0.72
Confusion Matrix:
[[32 64]
[20 184]]
This confusion matrix show us that our model gave real labels for 216 samples. 84 samples are incorrectly labeled by our model.
F1 score: 0.81
Kfold Cross Validation Approach Results:
Accuracy values for 10 kfold Cross Validation:
[0.81 0.65 0.75 0.72 0.77 0.68 0.71 0.73 0.73 0.68]
Final Average Accuracy values for 10 kfold Cross Validation: 0.72
F1 values for 10 kfold Cross Validation:
[0.78419831 0.60614657 0.71904762 0.70403509 0.73877589 0.64083536 0.68221388 0.69617021 0.69580507 0.67020884]
KNN Final Average F1 score for 10-kfold Cross Validation: 0.69

8 Result Of Comparing Classifications Models

DecisionTree: Confusion Matrix: [[44 52] [42 162]] Train Test Split Result: Accuracy score: 0.69 F1 score: 0.78 Kfold Cross Validation Result: Accuracy Score: 0.70 F1 Score: 0.70	**RandomForest:** Confusion Matrix: [[37 59] [21 183]] Train Test Split Result: Accuracy score: 0.73 F1 score: 0.82 Kfold Cross Validation Result: Accuracy Score: 0.76 F1 Score: 0.75
Support Vector Machine: Confusion Matrix: [[46 50] [24 180]] Train Test Split Result: Accuracy Score: 0.75 F1 Score: 0.83 Kfold Cross Validation Result: Accuracy Score: 0.76 F1 Score: 0.75	**KNearestNeighbors:** Confusion Matrix: [[32 64] [20 184]], Train Test Split Result: Accuracy Score: 0.72 F1 Score: 0.81 Kfold Cross Validation Result: Accuracy Score: 0.72 F1 Score: 0.69

Table 2: Result Of Comparing Classifications Models

I used Cross validation for testing the effectiveness of a machine learning models. Then I compared accuracy and f1-weighted values. Because I thought the weighted average of f1 scores would give a more accurate result. I saw that, the best model that gives us the best accuracy and f1-weighted is the Support Vector Machine (SVM) and Randomforest. But we know bad loan prediction success is as important as good loan prediction success in finance industry. Because labeling bad debts as good debt will result in loans to those who should not be given credit. So I prefer SVM algorithm for credit scoring solution.

9 Clustering

There are two types of evaluation methods to evaluate clustering quality. One is an external evaluation where the truth labels in the data sets are known in advance and the other is internal evaluation in which the evaluation is done with data set itself without true labels.
External performance measures are:
Jaccard index,Precision,Recall,F-measure
Internal performance measures are:

Silhouette Index:
• sil(x) [-1, 1]
• sil(x) = 1 x is far away from the neighboring clusters.
• sil(x) = 0 x is on the boundary between two neighboring clusters.
• sil(x) =-1 x is probably assigned to the wrong cluster.

Sum Group Of Square Errors:

$$W = \sum_{i=1}^{k} \sum_{\substack{\mathbf{x} \in D, \\ p(\mathbf{x})=i}} \|\mathbf{x} - \mathbf{c}_i\|^2$$

In this project I used internal performance measures, the Silhouette Index and Sum group of Square Errors.

9.1 Hierarchical Clustering

Hierarchical clustering is an alternative approach to kmeans clustering for identifying groups in the dataset. It does not require us to prespecify the number of clusters. Hierarchical clustering has an added advantage over K-means clustering. It results in an attractive tree-based representation of the observations, called a dendogram.

Hierarchical clustering algorithms group similar objects into groups called clusters. There are two types of hierarchical clustering algorithms:
Agglomerative is a Bottom up approach. Starts with many small clusters and merge them together to create bigger clusters. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster.
Divisive is a Top down approach. Starts with a single cluster than break it up into smaller clusters.
In hyperparameters, there are two important concepts: One is the way in which distance between clusters is calculated (linkage) and how distance is measured (affinity).

Linkages: “ward”, “complete”, “average”, “single”, (default=”ward”)
Single Linkage:
The distance between two clusters is the shortest distance between two points in each cluster
Complete Linkage: The distance between two clusters is the longest distance between two points in each cluster.
Average Linkage: The distance between clusters is the average distance between each point in one cluster to every point in other cluster.
Ward Linkage: The distance between clusters is the sum of squared differences within all clusters.
Affinity: Options available in sklearn are : “euclidean”, “l1”, “l2”, “manhattan”, “cosine”, or “precomputed”

Firstly, I used Hierarchical Clustering and looked Silhouette Score of various affinity and linkage values. I found best Silhouette Score for Hieararchial Clustering is 0.179 using manhattan affinity and single linkage.
Silhouette Score for euclidean ward :0.145
Silhouette Score for euclidean average:0.148
Silhouette Score for euclidean complete:0.132
Silhouette Score for euclidean single:0.079
Silhouette Score for manhattan average:0.170
Silhouette Score for manhattan complete:0.128
Silhouette Score for manhattan single:0.179

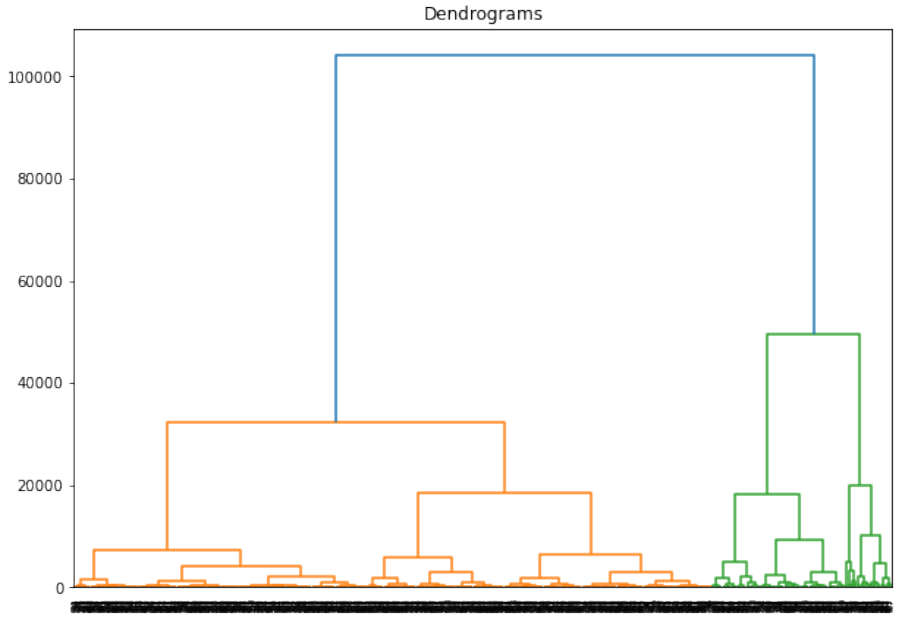

Figure 1: Dendogram

9.2 K-Means Clustering

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data. The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.

K Means Steps
Step1: The first step is to select the number of clusters, k.
Step2: Next, we randomly select the centroid for each cluster. For example, if we have 2 clusters, so k is equal to 2 here.
Step3: Then, assign all the points to the closest cluster centroid.
Step4: After we have assigned all of the points to either cluster, the next step is to recompute the centroids of newly formed clusters.
Step5: Repeat steps 3 and 4.

K Means Hyper Parameters
random_state: This is setting a random seed.We can set it to any number we want. In this project I used 123.
init: This is where you can set the initial cluster centroids.
n_clusters: Number of clusters
In cluster analysis, the Elbow Method is one of the most popular methods to determine this optimal number of clusters k.

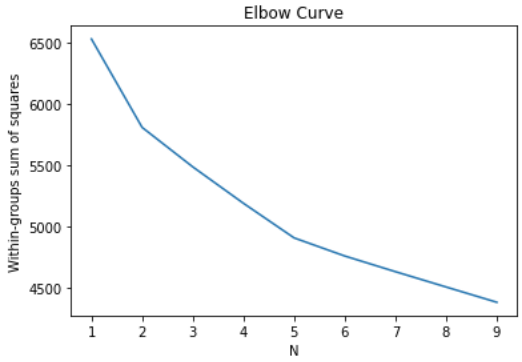

Figure 2: Elbow Curve
Firstly I used elbow method for finding optimal number of clusters k. Then I looked Silhouette scores for k-means algorithm. Their values are very close.
Silhouette score for 2 clusters k-means : 0.35
Silhouette score for 3 clusters k-means : 0.23
Silhouette score for 4 clusters k-means : 0.188
Silhouette score for 5 clusters k-means : 0.167
Silhouette score for 6 clusters k-means : 0.163
Silhouette score for 7 clusters k-means : 0.148
Silhouette score for 8 clusters k-means : 0.154

Best result is for 2 cluster 0.35 . But I prefered 3 clusters to reach meaningfull results. As an example, I looked at the distribution of columns(duration in month, credit amount,age and Personal status and sex in clusters.

9.3 Clustering Results

Number of customers in clusters:
1: 353
2: 331
3: 316

1.cluster:Mean of duration of credit is almost 18. Credit amount is near 2800. These graphs shows us that people in 1. Clusters use more credit and their mean of ages is 37. There are 25 female:divorced/separated/married people, 13 male:divorced/separated people, 50 male:married/widowed people, 260 "male:single" people
2.cluster:Mean of duration of credit is almost 25. Credit amount is near 4000. Their mean of ages is 35. There are 125 female:divorced/separated/married people, 13 male:divorced/separated people, 18 male:married/widowed people, 180 "male:single" people
3.cluster:Mean of duration of credit is almost 20. Credit amount is near 3000. Their mean of ages is 33. There are 160 female:divorced/separated/married people, 23 male:divorced/separated people, 30 male:married/widowed people, 100 "male:single" people

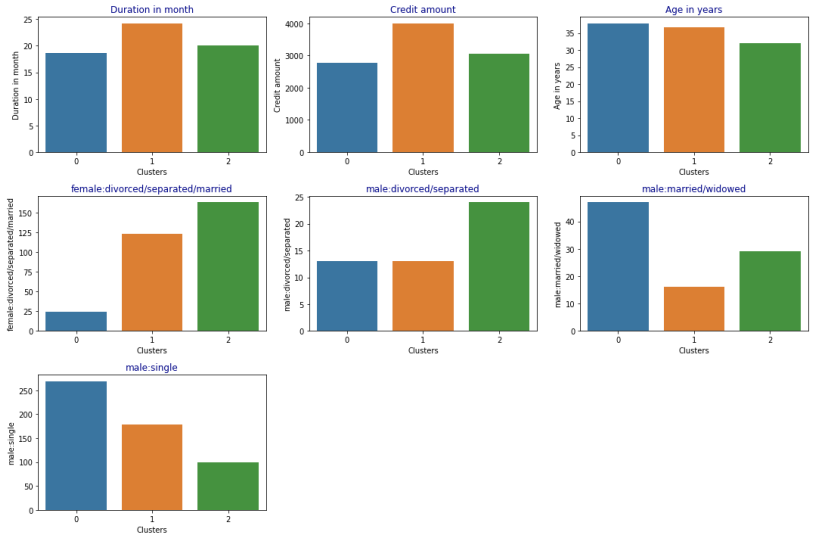

Table 3: Clustering Results

10 Frequent Pattern Mining

Frequent Pattern Mining is a Data Mining subject with the objective of extracting frequent itemsets from a database.

10.1 Frequent Pattern Mining Metrics

Support: This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears. Support: (number of transactions in which this itemset) / number of all transactions.
Confidence: X ->Y. This is measured by the proportion of transactions with item X, in which item Y also appears. Estimated by fraction of transactions that contain X and Y among all transactions containing X. Confidence: transactions containing X and Y/transactions containing X
Lift: This says how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is.

10.2 Apriori

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases.

Using the apriori principle, the number of itemsets that have to be examined can be pruned, and the list of popular itemsets can be obtained in these steps:
Step 1. Start with itemsets containing just a single item.
Step 2. Determine the support for itemsets. Keep the itemsets that meet your minimum support threshold, and remove itemsets that do not.
Step 3. Using the itemsets you have kept from Step 1, generate all the possible itemset configurations.
Step 4. Repeat Steps 2 and 3 until there are no more new itemsets.

Firstly I selected categorical values then I used mlxtend.preprocessing.TransactionEncoder that transforms dataset into an array format suitable for the mlxtend library. TransactionEncoder learned unique items from the dataset and transformed each transaction into a one-hot encoded boolean numpy array.
I assigned 0.4 to the minimum support value. If lift value more than 1 there is a positive correlation between columns. So I chose lift metric and assigned 1 to the minthreshold value. I saw some patterns between columns. These patterns can be used when classifying customers. If I decrease minsupport I can see more patterns between columns.

antecedents	consequents	antecedent support	consequent support	support	confidence
(own)	(male:single)	0.713	0.548	0.408	0.572230
(male:single)	(own)	0.548	0.713	0.408	0.744526
(none)	(existing credits paid back duly till now)	0.979	0.530	0.521	0.532176
(existing credits paid back duly till now)	(none)	0.530	0.979	0.521	0.983019
(<100 EUR, none)	(own)	0.586	0.713	0.419	0.715017
(own)	(<100 EUR, none)	0.713	0.586	0.419	0.587658
(none)	(own)	0.979	0.713	0.699	0.713994
(own)	(none)	0.713	0.979	0.699	0.980365
(<100 EUR)	(own)	0.603	0.713	0.430	0.713101
(own)	(<100 EUR)	0.713	0.603	0.430	0.603086

Table 4: Results Of Apriori

Results:

Male and single people have own house more than others.

Most of people have own house have Savings account/bonds less than 100 euro.