Project Report

Done by :

1- Ali Ahmed Hamed Shaker 20201701725

2- Mina George Raouf Iskander 20201701741

3- Mohamed Ibrahim Mohamed Mazroa 20201701732

4- Mohamed Reda 20201701733

5- Diaaelden Amr 20201701720

Preprocessing Techniques

Pre-processing Techniques

1- Dropping some columns

```
# Drop multiple columns because they does not have any logical meaning.
TrainDF.drop(['Unnamed: 0','id'], axis=1, inplace=True)

# Drop Columns because it has 100% nulls
TrainDF.drop(['INPUT_VALUE_ID_FOR_judgement_lien_time'], axis=1, inplace=True)
TrainDF.drop(['RATE_ID_FOR_avg_net_deposits'], axis=1, inplace=True)
TrainDF.drop(['RATE_ID_FOR_judgement_lien_time'], axis=1, inplace=True)
TrainDF.drop(['RATE_ID_FOR_industry_type'], axis=1, inplace=True)
```

2- Filling Nulls

```
# Float (using mean)
for col1 in C1:
    TrainDF[col1].fillna(TrainDF[col1].mean(), inplace=True)
# Object (using top)
for col2 in C2:
    TrainDF[col2].fillna('A', inplace=True)
```

```
owner_1_score                                0
RATE_owner_1                                 0
CAP_AMOUNT_owner_1                           0
PERCENT_OWN_owner_1                          0
owner_2_score                                0
RATE_owner_2                                 0
CAP_AMOUNT_owner_2                           0
PERCENT_OWN_owner_2                          0
owner_3_score                                0
RATE_owner_3                                 0
CAP_AMOUNT_owner_3                           0
PERCENT_OWN_owner_3                          0
years_in_business                            0
RATE_ID_FOR_years_in_business                0
fsr                                          0
RATE_ID_FOR_fsr                              0
location                                     0
RATE_ID_FOR_location                         0
funded_last_30                               0
RATE_ID_FOR_funded_last_30                   0
judgement_lien_percent                       0
RATE_ID_FOR_judgement_lien_percent           0
INPUT_VALUE_ID_FOR_judgement_lien_amount     0
RATE_ID_FOR_judgement_lien_amount            0
INPUT_VALUE_ID_FOR_num_negative_days         0
```

3- Encoding

#Label Encoding

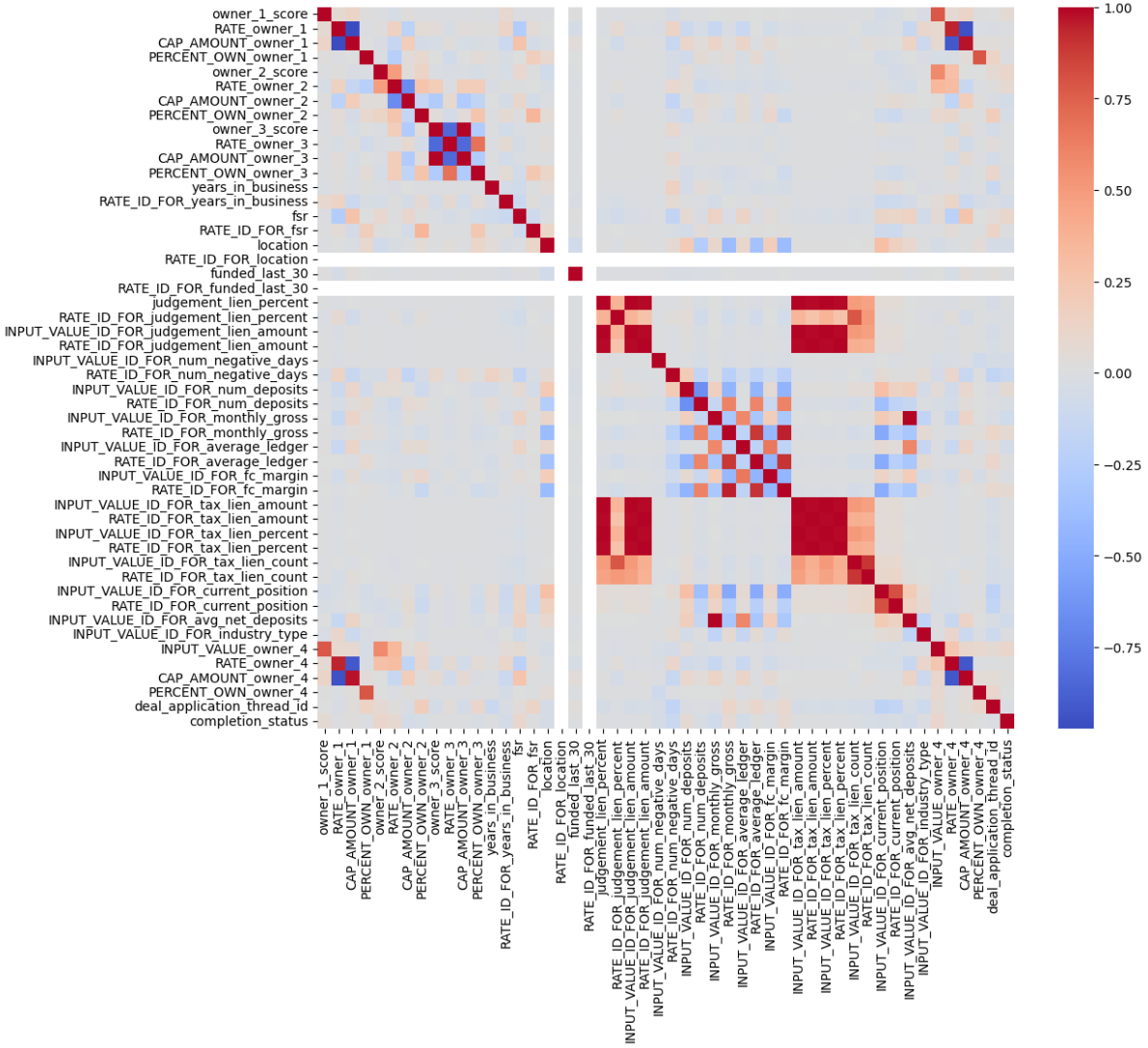
-Label encoding is a technique where categorical variables are assigned numerical labels starting from 0 to n-1, where n is the number of unique categories in the variable.

```
T_len =len(type)
for i in range(T_len):
    if type[i] =='object':
        TrainDF[column_lable[i]]= preprocessing.LabelEncoder().fit_transform(TrainDF[column_lable[i]])
```

Feature Extraction

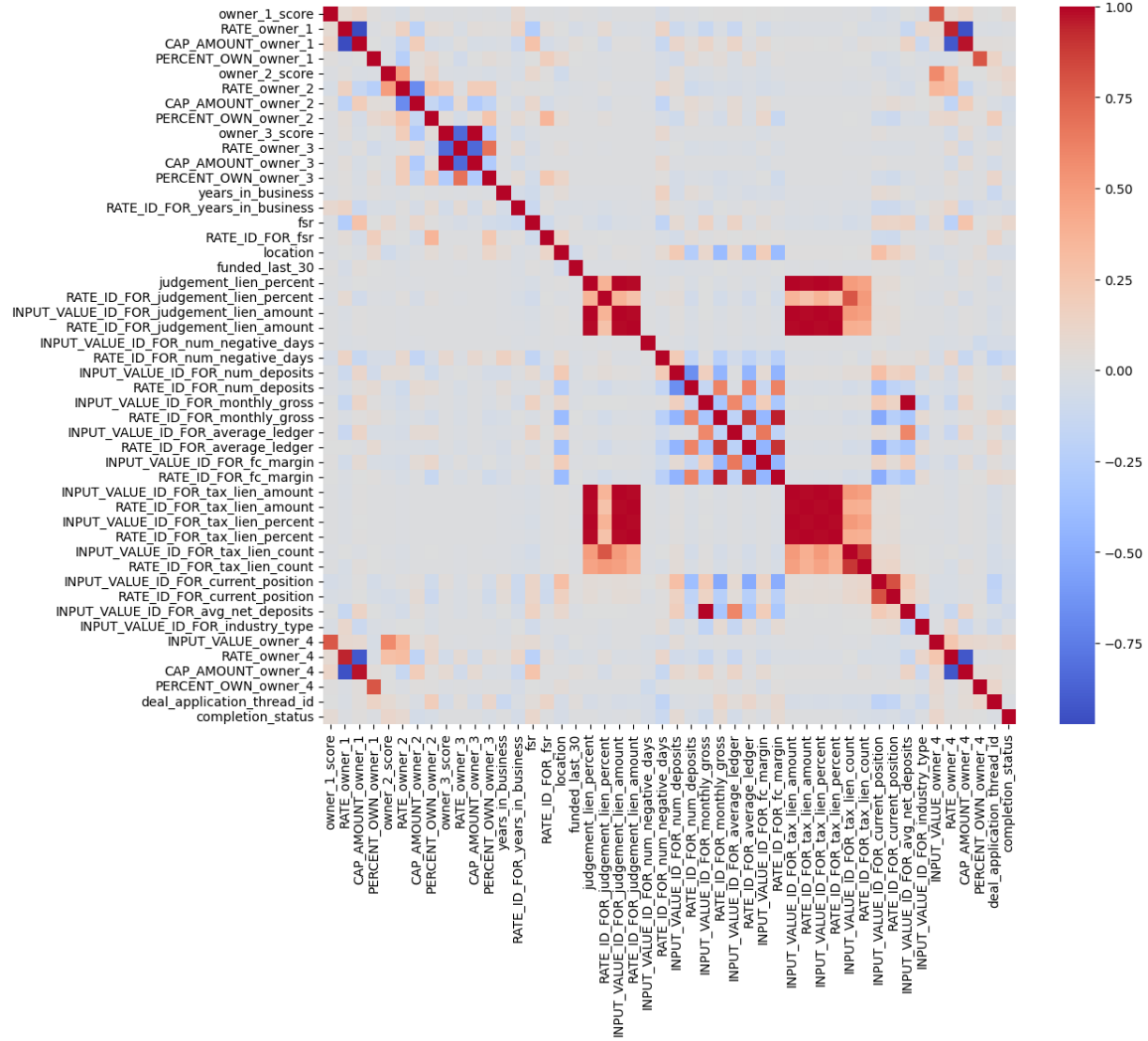
Correlation Between columns

```
# dropping unused columns (Zero Correlation)
TrainDF.drop(['RATE_ID_FOR_location'], axis=1, inplace=True)
TrainDF.drop(['RATE_ID_FOR_funded_last_30'], axis=1, inplace=True)
```



Correlation Between columns

-Correlation after dropping 2 columns because of zero correlation
('RATE_ID_FOR_location,'RATE_ID_FOR_funded_last_30')

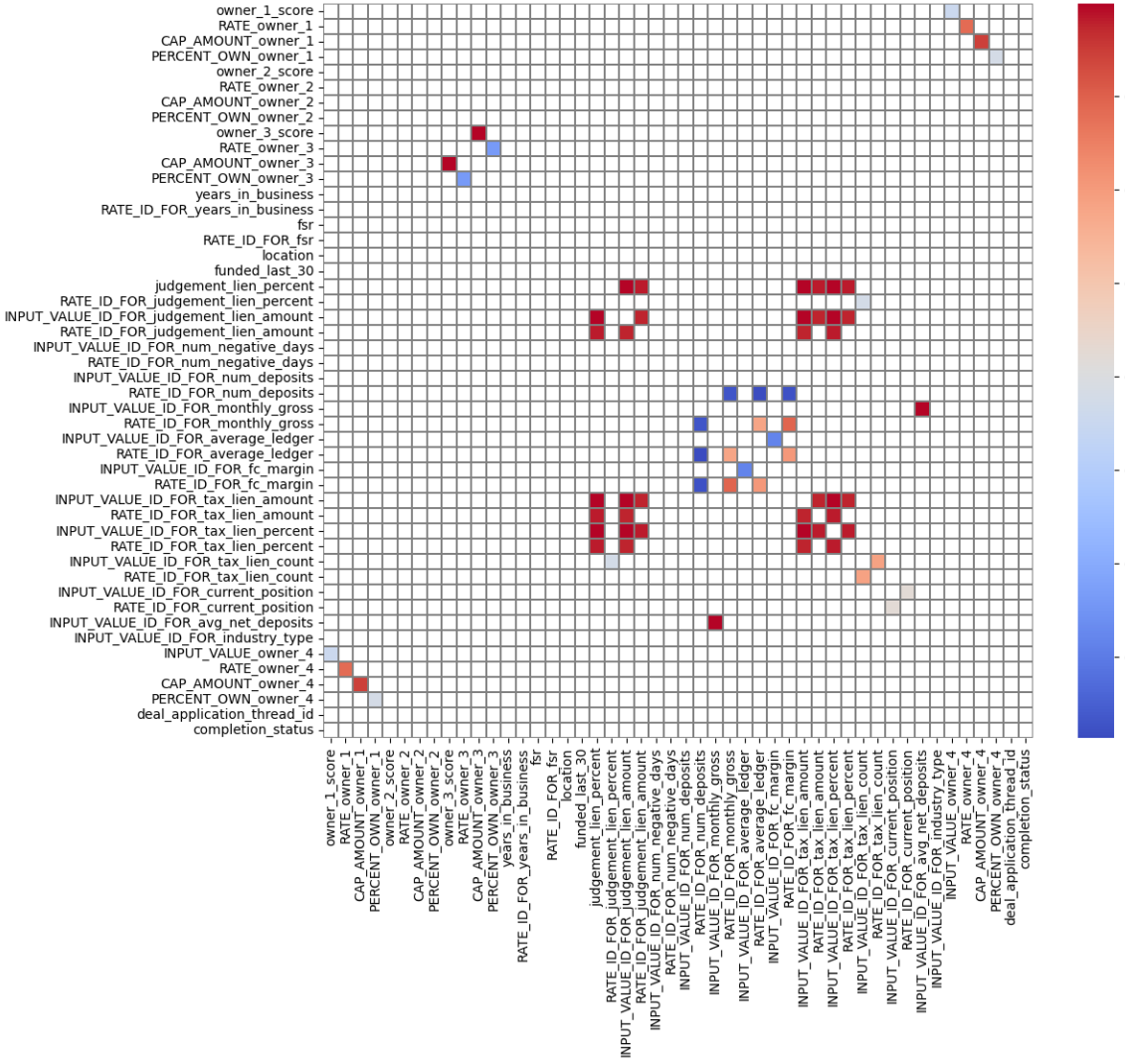


Correlation Between columns

-When we use most import columns `(data > 0.6, data < 1)` , accuracy was 88%

-When we use all columns `(data < 1)` , accuracy was 98%

-Conclusion: All columns are important



Models

Models

1- XGBoostclassifier: a powerful machine learning algorithm used for classification tasks that combines multiple weak prediction models to create a strong predictive model with high accuracy.

2-CatBoost: a machine learning algorithm designed specifically for handling categorical features, and it utilizes gradient boosting to create an accurate predictive model for classification tasks.

3-RandomForestClassifier: an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of the model for both binary and multiclass classification tasks.

4- GradientBoostingClassifier: a machine learning algorithm that sequentially builds an ensemble of weak prediction models to create a strong predictive model for classification tasks.

5-DecisionTree: is a machine learning algorithm that recursively splits data based on feature values to create a predictive model for classification or regression tasks.

6-KNN: a machine learning algorithm that classifies new data points based on the majority class of its k nearest neighbors in the feature space.

Hyperparameter Tuning

Hyperparameters

We used grid search technique in all models which has a lot of hyperparameters such as :

1- learning_rate : **It controls how much the model learns from each example.**

2- max_depth : **It determines how deep the model's decision-making process can go.**

3- n_estimators : **It represents the number of individual models or trees in an ensemble.**

4- iterations : **t defines the number of iterations or training cycles for a specific algorithm.**

5- depth : **It specifies the depth or complexity of a specific model or algorithm.**

6- n_neighbors : **It determines the number of neighbors considered in a k-nearest neighbors algorithm.**

7- weights : **It assigns weights to the neighbors based on their distance or similarity.**

8- algorithm : **It defines the algorithm used to organize and search for neighbors in a k-nearest neighbors algorithm.**

9- p : **It represents the power parameter used in distance calculations for k-nearest neighbors.**

10- criterion : **It determines the quality measure used to evaluate splits in a decision tree.**

Hyperparameters

1- XGBoostclassifier :

```
param_grid = {
    'learning_rate': [0.01, 0.1, 0.8],
    'max_depth': [3, 5, 7]
}
```

-when we used [0.01, 0.1, 0.3] accuracy decreased (learning rate)

-when we used [0.01, 0.1, 0.9] accuracy decreased (learning rate)

-When we used [2,3,5] accuracy decreased (max depth)

Hyperparameters

2-CatBoost:

```
param_grid = {
    'iterations': [50, 300],
    'learning_rate': [0.01, 0.3],
    'depth': [3, 5, 7],
}
```

- when we used [50,100,150] and [50,250] accuracy decreased (iterations)

Hyperparameters

3-Random ForestClassifier:

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 25],
}
```

- When we used [5,10,15] accuracy decreased (max depth)

Hyperparameters

4- GradientBoostingClassifier :

```
param_grid = {
    'learning_rate': [0.1, 0.8],
    'n_estimators': [100, 200]
}
```

- When we used [200,300] accuracy decreased (n_estimators)

Hyperparameters

5-DecisionTree:

```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15]
}
```

- when we used [5,10,15] and [30,50] accuracy decreased (`max_depth`)

Hyperparameters

6-KNN:

```
param_grid = {'n_neighbors': [3, 9],
              'weights': ['uniform', 'distance'],
              'algorithm': ['ball_tree', 'kd_tree', 'brute'],
              'p': [1, 2]}
```

- When we used [none,1,3,7,9] accuracy decreased (n_neighbors)

Models Evaluation

1-XGBoostclassifier

```
...   Accuracy: 98.62 %
      Precision: 98.96 %
      Recall: 98.78 %
      F1 score: 98.86 %
      Confusion matrix:
       [[104   2   2   0]
       [  0 160   0   0]
       [  0   1  84   0]
       [  0   0   0   9]]
```

2-CatBoost

```
Accuracy: 96.41 %
Precision: 97.42 %
Recall: 97.17 %
F1 score: 97.28 %
Confusion matrix:
 [[101   7   0   0]
 [  1 156   3   0]
 [  1   1  83   0]
 [  0   0   0   9]]
```

3-RandomForestClassifier

```
Accuracy: 94.75 %
Precision: 93.88 %
Recall: 95.59 %
F1 score: 94.62 %
Confusion matrix:
 [[ 98   7   3   0]
 [  1 156   2   1]
 [  1   4  80   0]
 [  0   0   0   9]]
```

4-GradientBoostingClassifier

```
Accuracy: 94.75 %
Precision: 93.88 %
Recall: 95.59 %
F1 score: 94.62 %
Confusion matrix:
 [[ 98   7   3   0]
 [  1 156   2   1]
 [  1   4  80   0]
 [  0   0   0   9]]
```

5-DecisionTree

```
Best hyperparameters: {'criterion': 'gini', 'max_depth': None}
Accuracy: 93.37 %
Precision: 89.14 %
Recall: 95.37 %
F1 score: 91.78 %
Confusion matrix:
 [[100   5   2   1]
 [ 10 146   2   2]
 [  1   1  83   0]
 [  0   0   0   9]]
```

6-KNN

```
Best hyperparameters: {'algorithm': 'ball_tree', 'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
Accuracy: 91.44 %
Precision: 93.04 %
Recall: 93.34 %
F1 score: 93.13 %
Confusion matrix:
 [[ 93  10   5   0]
 [  4 149   7   0]
 [  4   1  80   0]
 [  0   0   0   9]]
```

Models Evaluation

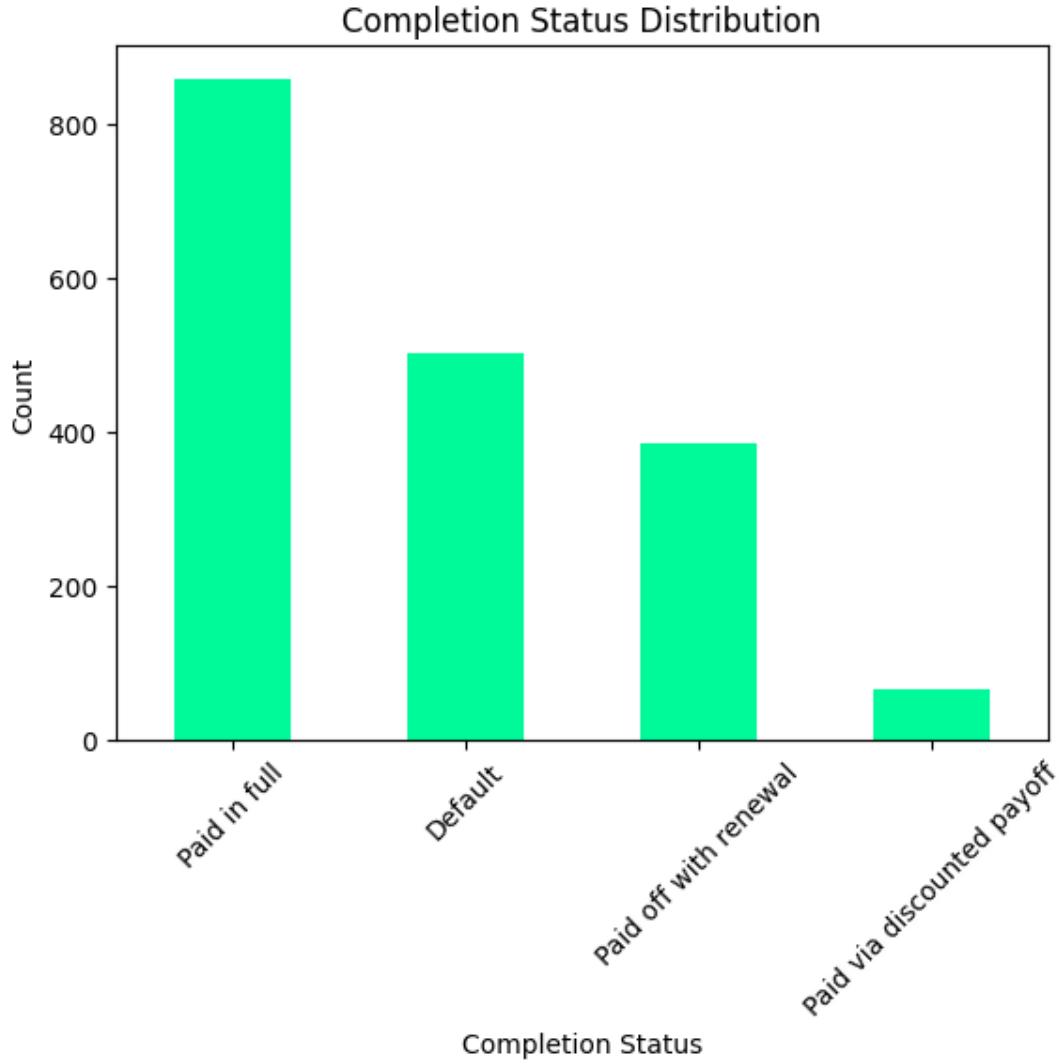
❖ The best model is " XGBoostclassifier"

```
Accuracy: 98.62 %
Precision: 98.96 %
Recall: 98.78 %
F1 score: 98.86 %
Confusion matrix:
 [[104   2   2   0]
 [  0 160   0   0]
 [  0   1  84   0]
 [  0   0   0   9]]
```

Visualization

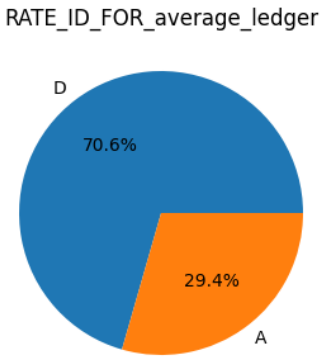
Visualization

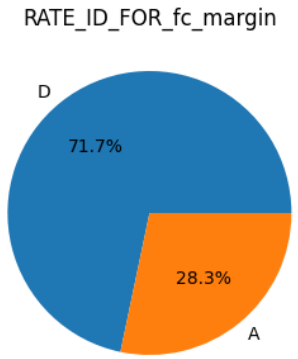
- **The plot represents the distribution of completion status for the training dataset.**

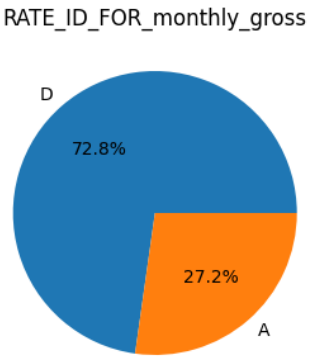


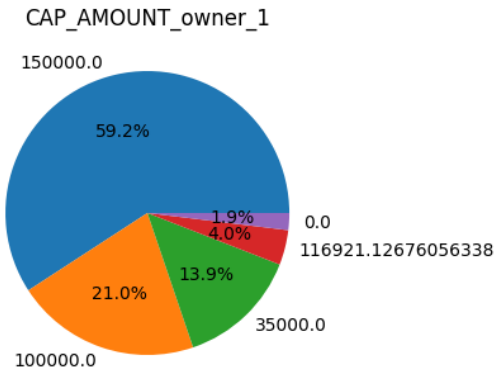
Visualization

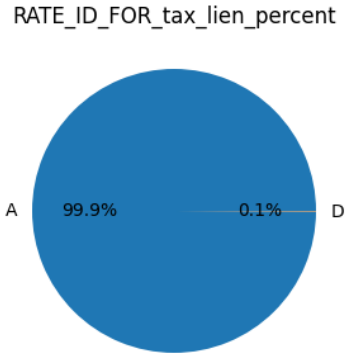
- **The pie charts for the eight columns visualize the distribution of unique values, providing an overview of their relative proportions within the training dataset**

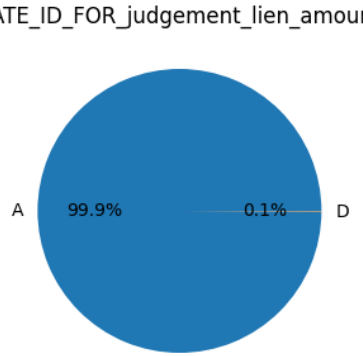


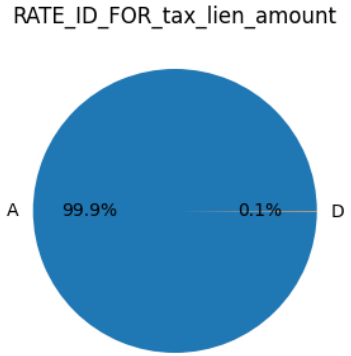


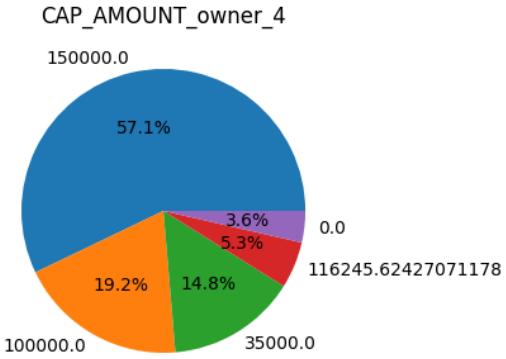






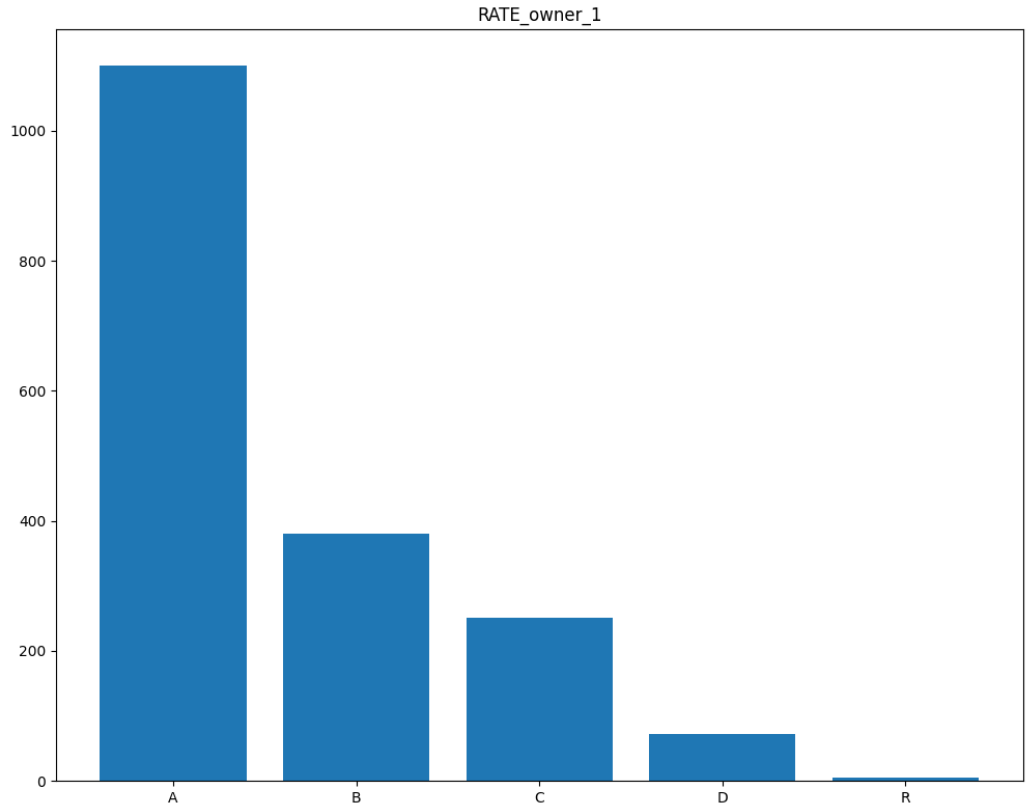


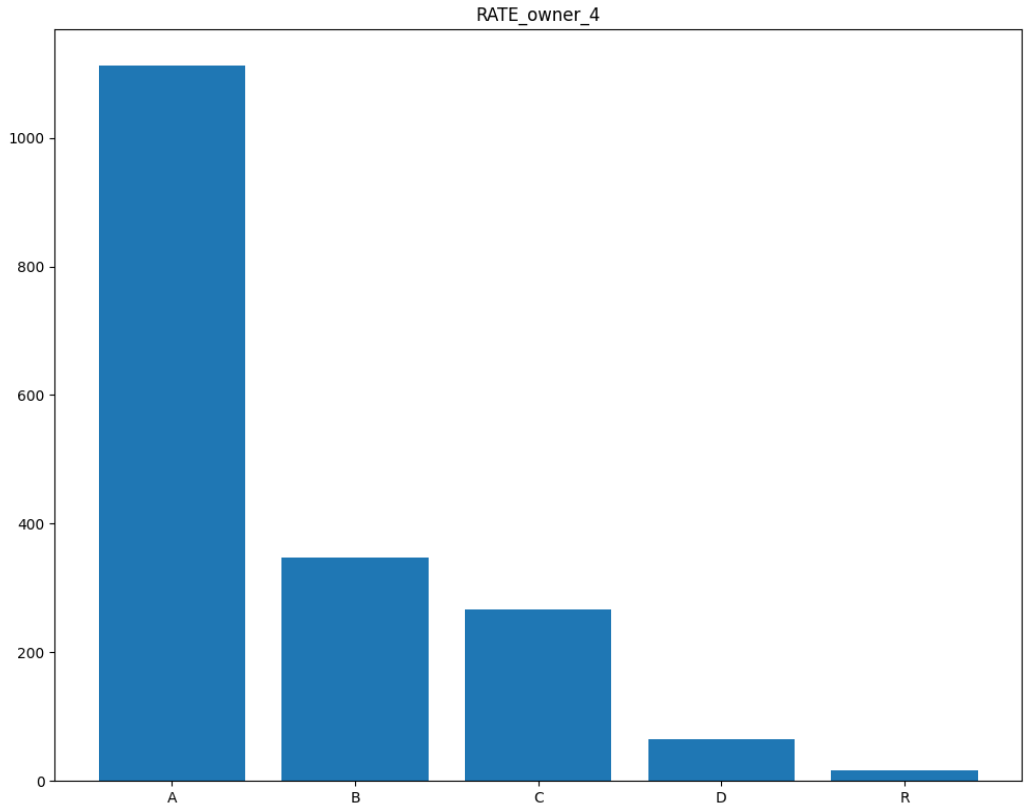




Visualization

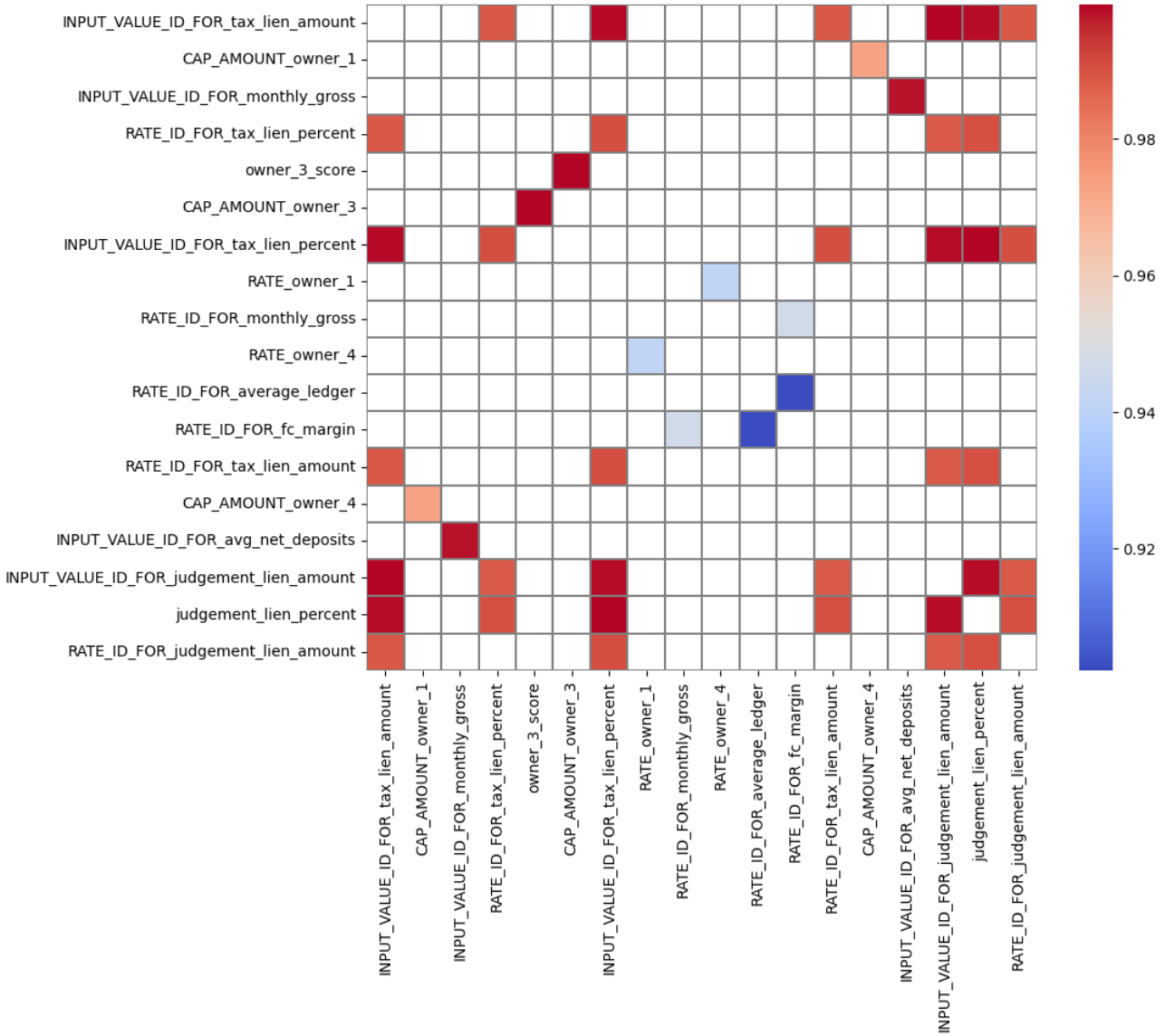
- the bar charts for the two columns depict the frequency of each unique value, offering a visual representation of their occurrence in the dataset.





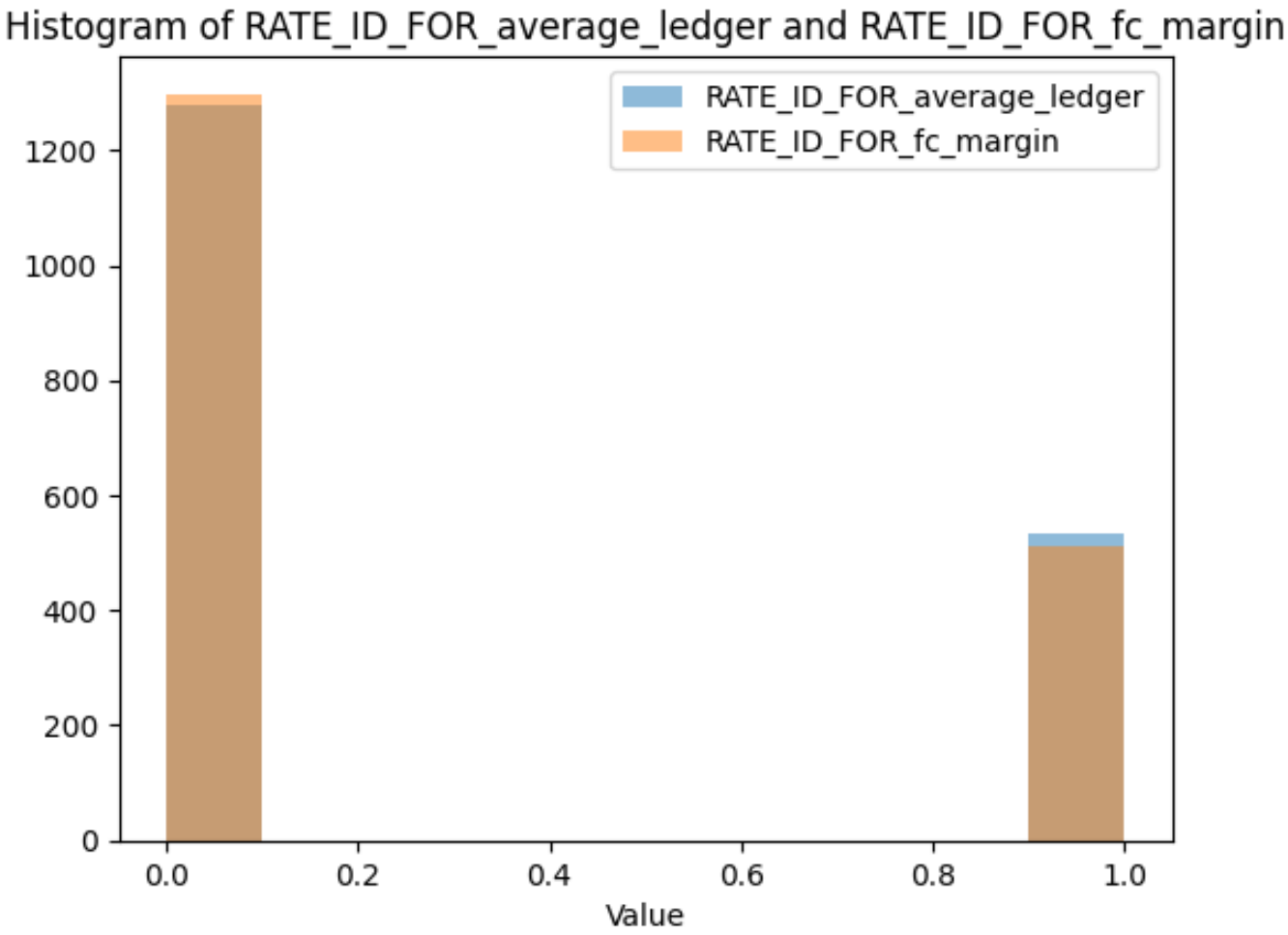
Visualization

- The plot visualizes a heatmap representing the high correlation between selected columns in the training dataset, using a coolwarm color scheme.



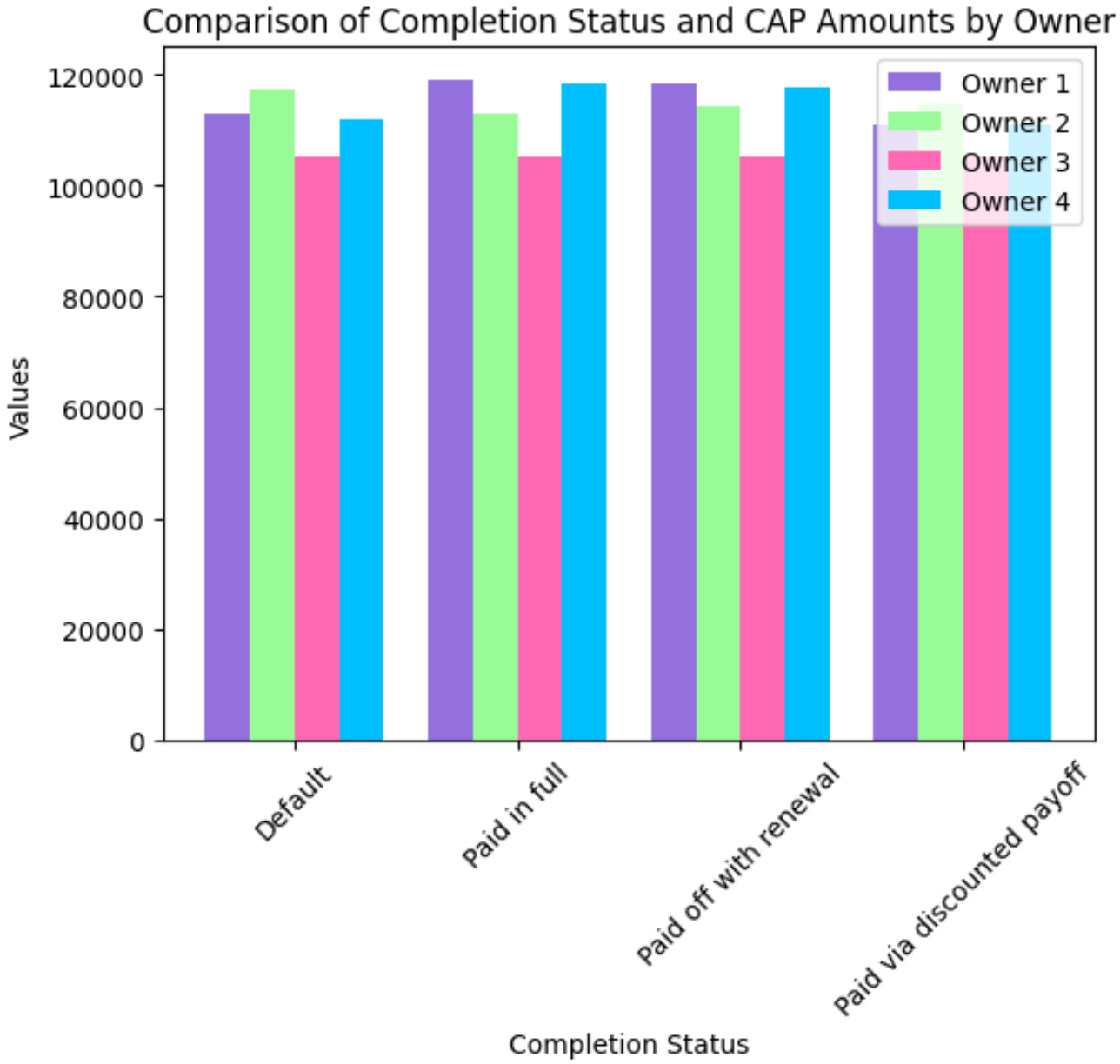
Visualization

- The plot shows two histograms, one for the 'RATE_ID_FOR_average_ledger' column and another for the 'RATE_ID_FOR_fc_margin' column. It helps visually compare the distributions of values in these two columns.



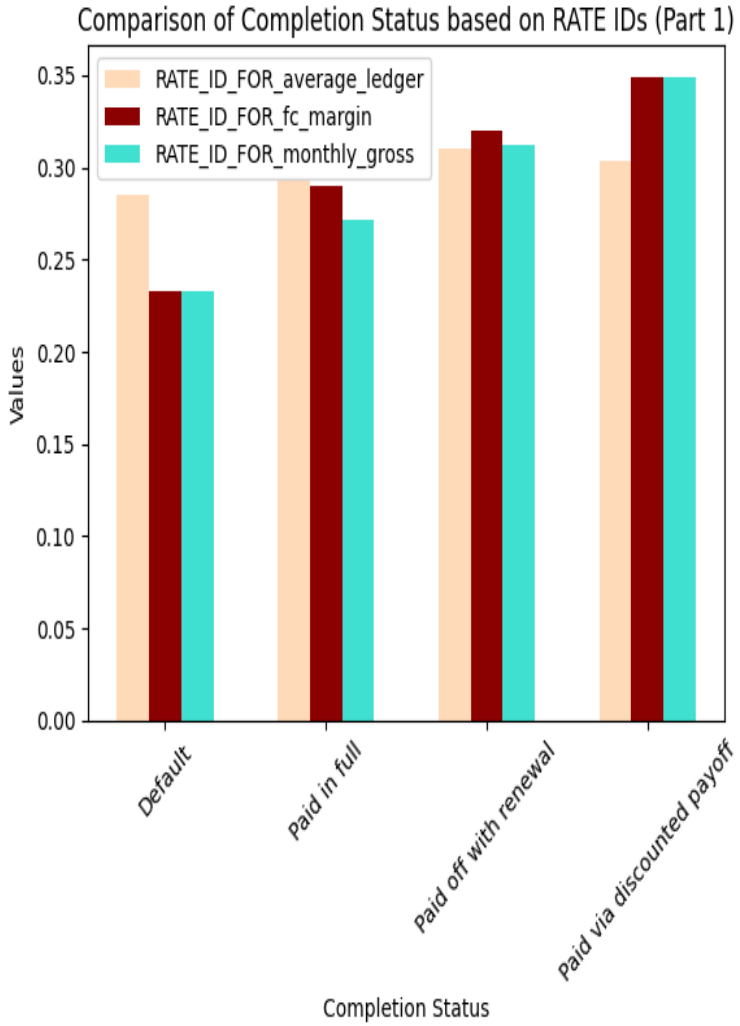
Visualization

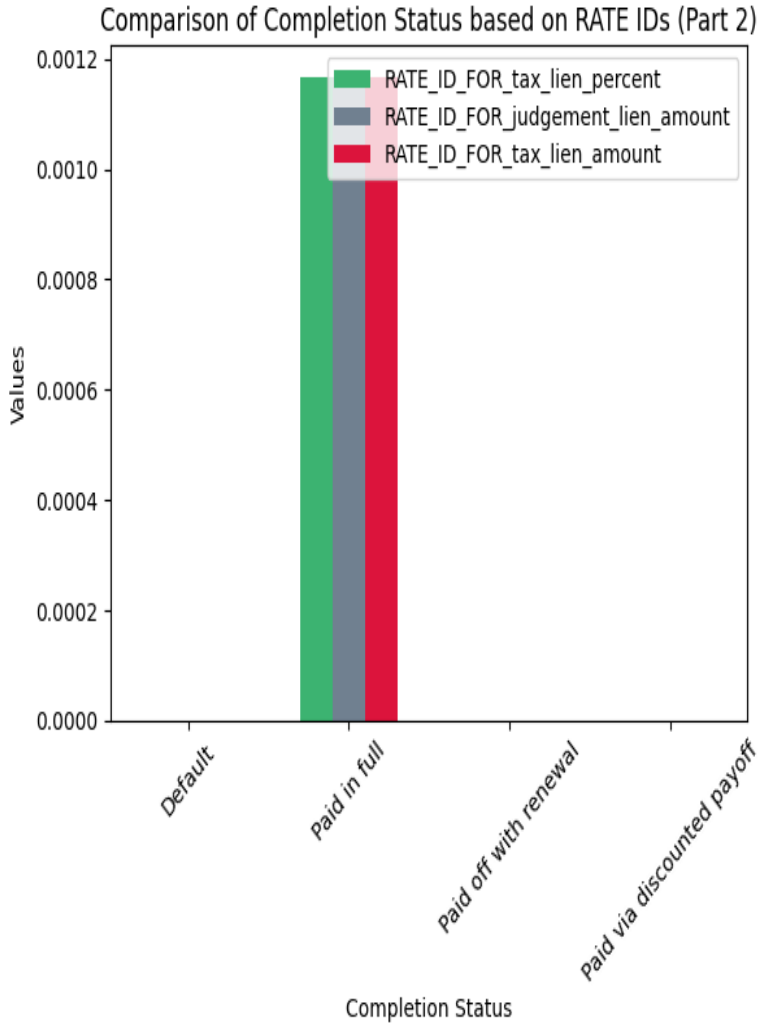
- The plot compares the average CAP amounts for different owners across completion status categories, providing insights into the relationship between completion status and CAP amounts for each owner.



Visualization

- The plot shows two charts side by side. Each chart compares the average values of different RATE IDs (categories) across different completion status groups. It helps understand how completion status relates to these RATE IDs.





Test Script

Test Script

1- Filling nulls :

We used an if statement to check each column's data type. If the data type was identified as an object, we replaced the null values with a common value. For columns that were not objects, we replaced the null values with the mean of the corresponding column in the training dataset.

2- Encoding :

The code checks if each column is categorical and, if so, applies encoding to convert the categorical values into numerical representations.

Conclusion

Conclusion

- Pre-processing Techniques : Dropping some columns , filling nulls with mean (float) and top (object) , Label encoding
- Feature Extraction : **Correlation Between columns (all columns are important)**
- Modeling : we used 6 classification models (**XGBoostclassifier , CatBoost , RandomForestClassifier , GradientBoostingClassifier , DecisionTree , KNN**)
- Hyperparameter Tuning : **we used some hyperparameters in grid search technique to get better accuracy such as (learning_rate , max_depth , n_estimators , iterations , depth ,n_neighbors , algorithm , weights , p , criterion)**
- Models Evaluation : `The best model is "XGBoostclassifier" (accuracy=98.62%)`

Conclusion

- Visualization : **In our analysis, we utilized informative plots to visually explore the dataset, gaining insights on completion status distribution, variable correlations, value distributions, and the relationship between completion status and CAP amounts for different owners. These plots provide valuable summaries that can assist in data analysis and decision-making tasks.**

- Test Script : **The code fills null values in different ways based on the data type of each column, and applies encoding to convert categorical values into numerical representations.**

Thank you