Credit Risk Customer

Musa Erkam Akçınar
Tobb Etü University
makcinar@etu.edu.tr

Abstract- ***In the realm of financial institutions and lending practices, the accurate assessment of credit risk plays a pivotal role in maintaining a healthy portfolio and ensuring the stability of the lending ecosystem. The CREDIT_RISK_CUSTOMER dataset stands as a representative collection of real-world data, encompassing a spectrum of attributes that influence credit risk. Harnessing the power of machine learning to extract insights and predictions from this data holds immense potential for refining risk evaluation processes.***

.

I. INTRODUCTION

In this project, our objective is to achieve the most accurate performance on the Credit_risk_customer dataset using various machine learning techniques. This will be accomplished by utilizing a portion of the pre-labeled data for training and another portion for testing purposes. The ultimate goal is to identify and determine the most suitable model that can yield the highest performance on this dataset. The link of presentation is https://youtu.be/DvzsYW9dZiM

II. DATA SET, DATA FEATURES AND ATTRIBUTES

The data set link:
https://www.kaggle.com/datasets/ppb00x/credit-risk-customers

A. About The Data

The provided below are the attributes of the dataset, explained in a simple manner

checking_status: This attribute likely represents the status of the checking account of the applicant, indicating their financial stability and management.

duration: The duration refers to the time for which the credit is being sought. This could provide insights into the loan's tenure and potential repayment capacity.

credit_history: This attribute may encapsulate the past credit history of the applicant, indicating their track record of repaying debts.

purpose: Purpose signifies the reason for which the credit is being requested, offering insights into the nature of the investment or expenditure.

credit_amount: This represents the amount of credit being requested, offering a quantitative measure of the financial requirement.

savings_status: Savings status likely indicates the applicant's level of savings or financial reserve, indicating their ability to handle unexpected financial obligations.

employment: Employment denotes the current job status of the applicant, shedding light on their stability and income source.

installment_commitment: This attribute could signify the percentage of income committed to repaying existing loans, revealing the applicant's debt burden.

personal_status: Personal status may represent the marital or family status of the applicant, which could influence their financial stability.

other_parties: Other parties could refer to additional individuals associated with the credit application, which might have an impact on the applicant's ability to repay.

property_magnitude: Property magnitude could provide insights into the applicant's assets or the collateral they possess.

age: The age of the applicant is a crucial factor in assessing credit risk, as it could reflect their career stage and potential income growth.

other_payment_plans: This attribute might indicate any existing alternate payment plans the applicant has, which could influence their repayment capability.

housing: Housing refers to the applicant's current living arrangement, which could provide insights into their stability and financial commitments.

existing_credits: This attribute signifies the number of existing credits the applicant holds, indicating their debt load.

job: Job type provides information about the applicant's occupation, which is relevant to their income and stability.

num_dependents: Num dependents represents the number of dependents the applicant has, which could influence their financial responsibilities.

own_telephone: This attribute indicates whether the applicant owns a telephone, which might serve as a proxy for stability and communication.

foreign_worker: Foreign worker indicates whether the applicant is a foreign worker, potentially affecting their financial standing and commitment.

class: The class attribute likely represents the target variable, indicating whether the applicant is classified as a good or bad credit risk.

```
# Changing object to category
cat= credit.select_dtypes(exclude=[np.number])
for i in list(cat.columns):
    credit[i]=credit[i].astype('category')

# Cheking Data Info
credit.info()
```

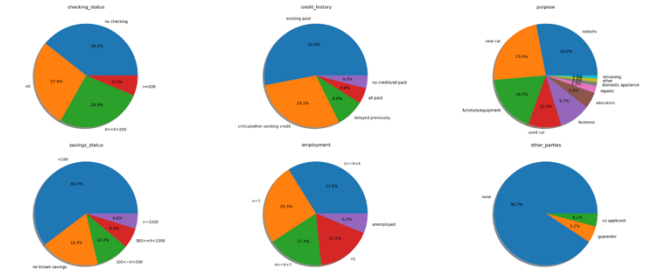
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   checking_status         1000 non-null   category
 1   duration                1000 non-null   float64
 2   credit_history          1000 non-null   category
 3   purpose                 1000 non-null   category
 4   credit_amount           1000 non-null   float64
 5   savings_status          1000 non-null   category
 6   employment              1000 non-null   category
 7   installment_commitment  1000 non-null   float64
 8   personal_status         1000 non-null   category
 9   other_parties           1000 non-null   category
 10  residence_since         1000 non-null   float64
 11  property_magnitude      1000 non-null   category
 12  age                     1000 non-null   float64
 13  other_payment_plans     1000 non-null   category
 14  housing                 1000 non-null   category
 15  existing_credits        1000 non-null   float64
 16  job                     1000 non-null   category
 17  num_dependents          1000 non-null   float64
 18  own_telephone           1000 non-null   category
 19  foreign_worker          1000 non-null   category
 20  class                   1000 non-null   category
dtypes: category(14), float64(7)
memory usage: 71.0 KB
```

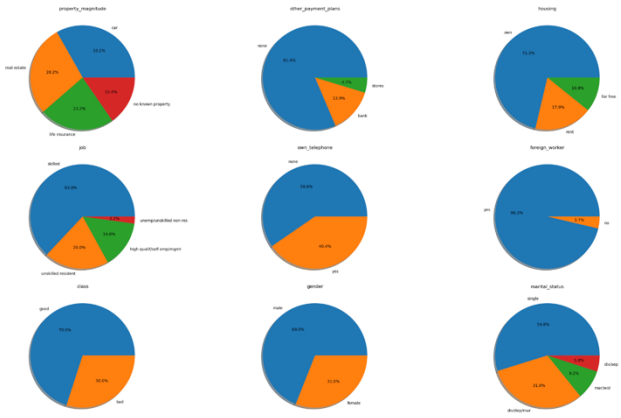
```
In [10]: # Cheking categorical features
         cat=credit.select_dtypes(exclude=[np.number])
         cat.columns

Out[10]: Index(['checking_status', 'credit_history', 'purpose', 'savings_status',
                'employment', 'other_parties', 'property_magnitude',
                'other_payment_plans', 'housing', 'job', 'own_telephone',
                'foreign_worker', 'class', 'gender', 'marital_status'],
               dtype='object')
```

B. Data Visualation

Visualizing Categorical Features

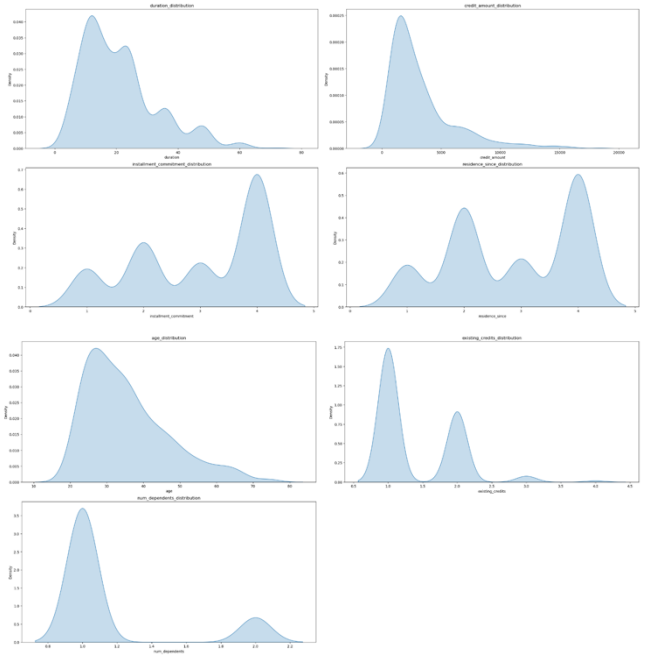
```
lst=['checking_status', 'credit_history', 'purpose', 'savings_status','employment', 'other_parties', 'property_magnitud
    'foreign_worker', 'class', 'gender', 'marital_status']
plt.figure(figsize=(25,25),layout='constrained')
for i in range(len(lst)):
    plt.subplot(5,3,i+1)
    a=credit[lst[i]].value_counts()
    lbl=a.index
    plt.pie(x=a,labels=lbl,autopct="% .1f%%",shadow=True)
    plt.title(lst[i])
plt.show()
```





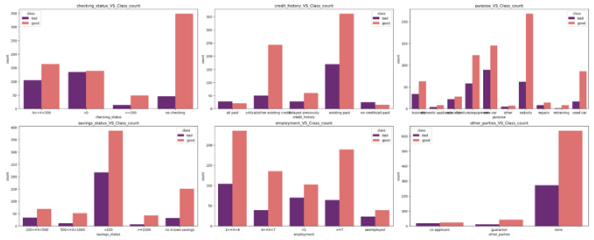
Visualizing Numerical Features

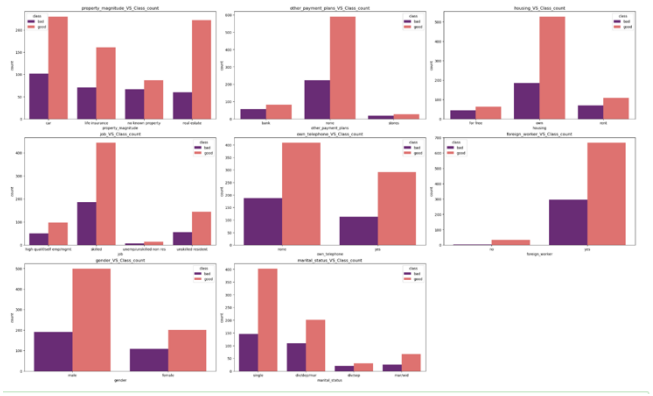
```
lst=['duration', 'credit_amount', 'installment_commitment',
    'residence_since', 'age', 'existing_credits', 'num_dependents']
plt.figure(figsize=(25,25),layout='constrained')
for i in range(len(lst)):
    plt.subplot(4,2,i+1)
    sns.kdeplot(data=credit,x=lst[i],shade=True)
    plt.title(lst[i]+'_distribution')
plt.show()
```



Visualizing Categorical Features against the target feature

```
In [17]: lst=['checking_status', 'credit_history', 'purpose', 'savings_status','employment', 'other_parties', 'property_magnitud
             'foreign_worker', 'gender', 'marital_status']
         plt.figure(figsize=(25,25),layout='constrained')
         for i in range(len(lst)):
             plt.subplot(5,3,i+1)
             sns.countplot(data=credit,x=lst[i],palette='magma',hue='class')
             plt.title(lst[i]+'_VS_Class_count')
         plt.show()
```





C. *Data Scaling*

Within this project, data undergo scaling through the application of the min-max scaling method. Min-Max Scaling is a fundamental data preprocessing technique utilized to standardize numerical features within a specified range, usually between 0 and 1. By transforming the values of each feature while maintaining their relative proportions, Min-Max Scaling ensures that features with varying scales can be uniformly compared and integrated into machine learning models. This technique involves subtracting the minimum value of each feature from its original value and then dividing by the range (difference between maximum and minimum values). As a result, features are brought onto a consistent scale, aiding algorithms that are sensitive to input feature magnitudes. Min-Max Scaling finds applications in domains like image processing and economic analysis, where disparate scales can impede accurate analysis and modeling. Its role in enhancing convergence, stability, and performance of machine learning algorithms underscores its significance in the data preprocessing pipeline.

```
# Label Encoding
for i in list(cat.columns):
    credit[i+'_encoder']=LabelEncoder().fit_transform(credit[i])

# Taking only numerical features
num=credit.select_dtypes(include=[np.number])

# Data Scaling
scl=pd.DataFrame(MinMaxScaler().fit_transform(num.to_numpy()),columns=num.columns)

# Scaled Data
scl.head()
```

	duration	credit_amount	installment_commitment	residence_since	age	existing_credits	num_dependents	checking_status_encoder
0	0.029412	0.050567	1.000000	1.000000	0.857143	0.333333	0.0	0.333333
1	0.647059	0.313690	0.333333	0.333333	0.053571	0.000000	0.0	0.000000
2	0.117647	0.101574	0.333333	0.666667	0.535714	0.000000	1.0	1.000000
3	0.558824	0.419941	0.333333	1.000000	0.464286	0.000000	1.0	0.333333
4	0.294118	0.254209	0.666667	1.000000	0.607143	0.333333	1.0	0.333333

credit_history_encoder	purpose_encoder	...	other_parties_encoder	property_magnitude_encoder	other_payment_plans_encoder	housing_encoder	job_encoder
0.25	0.666667	...	1.0	1.000000	0.5	0.5	0.333333
0.75	0.666667	...	1.0	1.000000	0.5	0.5	0.333333
0.25	0.222222	...	1.0	1.000000	0.5	0.5	1.000000
0.75	0.333333	...	0.5	0.333333	0.5	0.0	0.333333
0.50	0.444444	...	1.0	0.666667	0.5	0.0	0.333333

r	own_telephone_encoder	foreign_worker_encoder	class_encoder	gender_encoder	marital_status_encoder
3	1.0	1.0	1.0	1.0	1.0
3	0.0	1.0	0.0	0.0	0.0
)	0.0	1.0	1.0	1.0	1.0
3	0.0	1.0	1.0	1.0	1.0
3	0.0	1.0	0.0	1.0	1.0

D. *Corelation Matrix*

III. USED MODELS

Within this project, The data has been randomly divided, with 70% allocated for training and the remaining 30% for testing. Also, four distinct models have been employed, namely logistic regression, decision tree, k-nearest neighbor, and random forest models. We can elaborate on each of these models sequentially as follows:

Logistic Regression: Logistic regression is a widely used statistical method for binary classification. It estimates the probability of a given input belonging to one of two possible classes, making it suitable for scenarios where the target variable is categorical. By fitting a logistic function to the data, it computes the likelihood of each class and assigns the input to the class with the higher probability.

Decision Tree: The decision tree model is a versatile tool for both classification and regression tasks. It creates a tree-like structure where each internal node represents a decision based on a feature, leading to subsequent nodes or leaves that

indicate the predicted outcome. Decision trees are intuitive to interpret and can handle complex relationships in data.

K-Nearest Neighbor (K-NN): K-NN is a non-parametric algorithm used for classification and regression. It classifies a new data point based on the majority class of its k-nearest neighbors in the training data. The choice of 'k' influences the level of noise reduction and the algorithm's sensitivity to local variations.

Random Forest: Random forest is an ensemble learning method that combines multiple decision trees to enhance predictive performance. Each tree is trained on a different subset of the data and features. The final prediction is determined by aggregating the predictions of individual trees, resulting in improved accuracy and reduced overfitting.

These models collectively provide a diversified set of tools to address different types of data and problem domains, contributing to a comprehensive analysis within the project.

IV. RESULTS

The performance metrics including accuracy, F1 score, confusion matrix, and classification report were evaluated for four different models: logistic regression, decision tree, k-nearest neighbors (KNN), and random forest. Based on the F1 scores, the models were ranked in the following order: Random Forest achieved the highest F1 score, followed by Logistic Regression, K-Nearest Neighbors, and finally Decision Tree. This suggests that, among the tested models, Random Forest exhibited the best overall performance in terms of F1 score.

```
********** Logistic Regression**********


Accuracy :  0.7266666666666667
F1 Score :  0.8247863247863247
==================================================
Confusion Matrix :
 [[ 25  61]
 [ 21 193]]
==================================================
Classification Report :
              precision    recall  f1-score   support

         0.0       0.54      0.29      0.38        86
         1.0       0.76      0.90      0.82       214

    accuracy                           0.73       300
   macro avg       0.65      0.60      0.60       300
weighted avg       0.70      0.73      0.70       300
```

```
**********Decision Tree**********


Accuracy :  0.69
F1 Score :  0.7801418439716313
==================================================
Confusion Matrix :
 [[ 42  44]
 [ 49 165]]
==================================================
Classification Report :
              precision    recall  f1-score   support

         0.0       0.46      0.49      0.47        86
         1.0       0.79      0.77      0.78       214

    accuracy                           0.69       300
   macro avg       0.63      0.63      0.63       300
weighted avg       0.70      0.69      0.69       300
```

```
**********K Nearest Neighbor**********


Accuracy :  0.6866666666666666
F1 Score :  0.7882882882882883
==================================================
Confusion Matrix :
 [[ 31  55]
 [ 39 175]]
==================================================
Classification Report :
              precision    recall  f1-score   support

         0.0       0.44      0.36      0.40        86
         1.0       0.76      0.82      0.79       214

    accuracy                           0.69       300
   macro avg       0.60      0.59      0.59       300
weighted avg       0.67      0.69      0.68       300
```

```
**********Random Forest**********


Accuracy :  0.7466666666666667
F1 Score :  0.8354978354978355
==================================================
Confusion Matrix :
 [[ 31  55]
 [ 21 193]]
==================================================
Classification Report :
              precision    recall  f1-score   support

         0.0       0.60      0.36      0.45        86
         1.0       0.78      0.90      0.84       214

    accuracy                           0.75       300
   macro avg       0.69      0.63      0.64       300
weighted avg       0.73      0.75      0.72       300
```

Model Performance Comparison

```
In [16]: # Creating tabular format for better comparison
         tbl=pd.DataFrame()
         tbl['Model']=pd.Series(['Logistic Regreession','Decision Tree','KNN','Random Forest'])
         tbl['Accuracy']=pd.Series([acc_lr,acc_dt,acc_knn,acc_rf])
         tbl['F1_Score']=pd.Series([f1_lr,f1_dt,f1_knn,f1_rf])
         tbl.set_index('Model')
```

Out[16]:

Model	Accuracy	F1_Score
Logistic Regreesion	0.726667	0.824786
Decision Tree	0.690000	0.780142
KNN	0.686667	0.788288
Random Forest	0.746667	0.835498

```
# Best model on the basis of F1 Score
tbl.sort_values('F1_Score',ascending=False)
```

	Model	Accuracy	F1_Score
3	Random Forest	0.746667	0.835498
0	Logistic Regreesion	0.726667	0.824786
2	KNN	0.686667	0.788288
1	Decision Tree	0.690000	0.780142

V. CONCLUSIONS

A. *Summary*

In this study, the data was initially scaled to ensure uniformity, followed by the training of four distinct machine learning models: logistic regression, decision tree, k-nearest neighbors (KNN), and random forest. Subsequently, a comprehensive evaluation encompassing accuracy, F1 score, confusion matrix, and classification report was conducted for these models. The comparative analysis of these metrics revealed insights into their performance. Ultimately, the results indicate that, after scaling the data and evaluating the models, the random forest model exhibited the highest F1 score, showcasing its superior predictive capabilities among the tested models.

B. *Key findings, Benefits, and Contributions of the Study*

As a result of this study, several key insights were gained. Firstly, the importance of data scaling as a preprocessing step was highlighted, leading to improved model convergence and performance. Secondly, through the comprehensive evaluation of multiple machine learning models, including logistic regression, decision tree, k-nearest neighbors, and random forest, we gained a deeper understanding of their strengths and weaknesses in predicting the target variable. This evaluation allowed us to identify the random forest model as the most effective in terms of F1 score.

The significance of this study lies in its contribution to informed decision-making. By rigorously comparing and assessing various models, we provide valuable guidance for selecting an appropriate model when faced with similar predictive tasks. Additionally, our work emphasizes the importance of preprocessing techniques, like data scaling, which can significantly impact the performance of machine learning models. Overall, this study enhances our understanding of model evaluation, selection, and preprocessing, ultimately leading to more accurate and reliable predictions in practical applications.

C. *Future Works*

Based on the outcomes of this project, several potential avenues for future research and development emerge. Firstly, an exploration of ensemble techniques could be undertaken to further enhance the predictive capabilities of the models. Combining the strengths of various models, such as the Random Forest and K-Nearest Neighbors, might yield even more robust results.

Secondly, investigating the impact of feature engineering could offer valuable insights. Fine-tuning the input variables by creating new features or selecting relevant ones could potentially improve the models' performance.

Additionally, a deeper analysis of the dataset's characteristics and potential biases could lead to refinements in the preprocessing steps. Addressing any data imbalances or outliers might result in more accurate and generalizable models.

Furthermore, the study could be extended by incorporating more advanced algorithms, such as gradient boosting or support vector machines, to evaluate their effectiveness on the given task.

Lastly, exploring the application of deep learning techniques, like neural networks, could provide a novel perspective on the predictive modeling problem, potentially leading to even better performance.

In summary, there are numerous promising paths for future research, ranging from model ensembles and feature engineering to advanced algorithms and deep learning, all of which could contribute to the continuous improvement of predictive modeling in similar contexts.

REFERENCES

[1] https://en.wikipedia.org/wiki/Random_forest

[2] https://en.wikipedia.org/wiki/Decision_tree

[3] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[4] https://en.wikipedia.org/wiki/Linear_regression

[5] https://scikit-learn.org/stable/