CREDIT CARD DEFAULTER ANALYSIS

SUBMISSION BY:

1. MR. VISHAL TAD (EBEON0123797353)
2. MR. MANU PANDEY (EBEON0123704146)
3. MR DEEP BHOYAR (EBEON0123730724)

INDEX

Executive Summary

- In today's world credit cards have become a lifeline to a lot of people so banks provide us with credit cards. Now we know the most common issue there is in providing these kinds of deals are people not being able to pay the bills. These people are what we call "defaulters." Credit card default happens when you have become severely delinquent on your credit card payments. Missing credit card payments once or twice does not count as a default. A payment default occurs when you fail to pay the Minimum Amount Due on the credit card for a few consecutive months

- Objective of our project is to predict which customer might default in upcoming months. The research aims at developing a mechanism to predict the credit card default beforehand and to identify the potential customer base that can be offered various credit instruments to invite minimum default.

- The purpose of this project is to conduct quantitative analysis on credit card default risk by applying various classification machine learning models. Despite machine learning and big data have been adopted by the banking industry, the current applications are mainly focused on credit score predicting. Heavily relying on credit scores could cause banks to miss valuable customers who are new immigrants with repaying power but little to no credit history. This analysis is a machine learning application on default risk itself and the predictor features do not include credit score or credit history. Due to the regulatory constraints that banks are facing.

- From this study, we discovered a few interesting insights which may or may not hold for other datasets. We learned the most important predictors of default are not human characteristics, but the most recent 2 months' payment status and customers' credit limit. The conventional thinking of younger people tends to have higher default risk is proven to be only partially true in this dataset. Also, surprisingly, customers being inactive for months does not mean they have no default

1. Introduction

1.1 Background

- Credit risk has traditionally been the greatest risk among all the risks that the banking and credit card industry are facing, and it is usually the one requiring the most capital. This can be provenby industry business reports and statistical data. For example, “The Federal Reserve Bank of New York measures credit card delinquencies based on the percent of balances that are at least 90 days late. For the third quarter of 2019, that rate was about 8%, about the same level as in the previous quarter.”[2] Thus, assessing, detecting and managing default risk is the key factor in generating revenue and reducing loss for the banking and credit card industry.

- Despite machine learning and big data have been adopted by the banking industry, the current applications are mainly focused on credit score predicting. The disadvantage of heavily relyingon credit score is banks would miss valuable customers who come from countries that are traditionally underbanked with no credit history or new immigrants who have repaying power but lack credit history. According to a literature review report on analyzing credit risk using machine and deep learning models, “credit risk management problems researched have been around credit scoring; it would go a long way to research how machine learning can be applied to quantitative areas for better computations of credit risk exposure by predicting probabilities of default.” [3]

- The purpose of this project is to conduct quantitative analysis on credit card default risk by using interpretable machine learning models with accessible customer data, instead of credit score or credit history, with the goal of assisting and speeding up the human decision making process.

1.2 Method Overview

- **Data**: We acquire the data from a public dataset from Kaggle. The original dataset can be found at the UCI Machine Learning Repository. Lichman, M. (2013). UCI Machine Learning Repository Irvine, CA: University of California, School of Information and Computer Science.

- **Models:** Due to the scope of the project and lack of computational resources, this analysis is not intended to be exhaustive, we only applied 6 classification machine learning models - Logistic Regression, Random Forest, KNN, NB, Decision Tree and SVM .

- **Tools:** The programming is done in Python. Scikit-learn Python libraries are utilized for machine learning modelling. For data analysis and visualization, we use NumPy, Pandas, matplotlib and seaborn.

2. Data Exploration

This dataset contains information on default payments, demographic factors, credit limit, history of payments, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. It includes 30,000 rows and 25 columns, and there is no credit score or credit history information. Data dictionary is available in Appendix A.

Overall, the dataset is very clean, but there are several undocumented column values. As a result, most of the data wrangling effort was spent on searching information and interpreting the columns. More details about the data cleaning can be found in this Jupyter Notebook.

The purpose of exploratory data analysis is to identify the variables that impact payment default likelihood and the correlations between them. We use graphical and statistical data exploratoryanalysis tools to check every categorical variable. Each starts with a visualization and is followed by a statistical test to verify the findings.

The main findings from exploratory analysis are as following:

- Females have more delayed payment than maless in this dataset. Keep in mind that this finding only applies to this dataset, it does not imply this is true for other datasets.
- Customers with higher education have less default payments .
- Customers aged between 30-50 have the lowest delayed payment rate, while younger groups (20-30) and older groups (50-70) all have higher delayed payment rates. However, the delayed rate drops slightly again in customers older than 70.
- There appears to be no correlation between default payment and marital status.
- Customers being inactive does not mean they have no default risk.

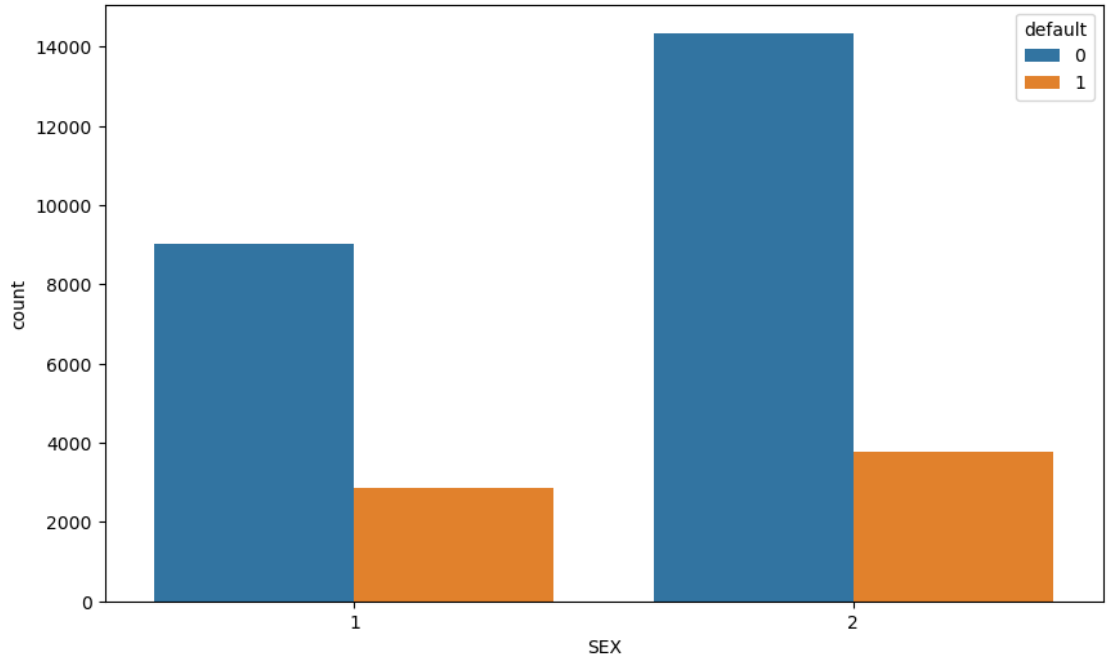
Exploratory Data Analysis

In our EDA part we have carried out the basic and important initials commits; below are a few to name:

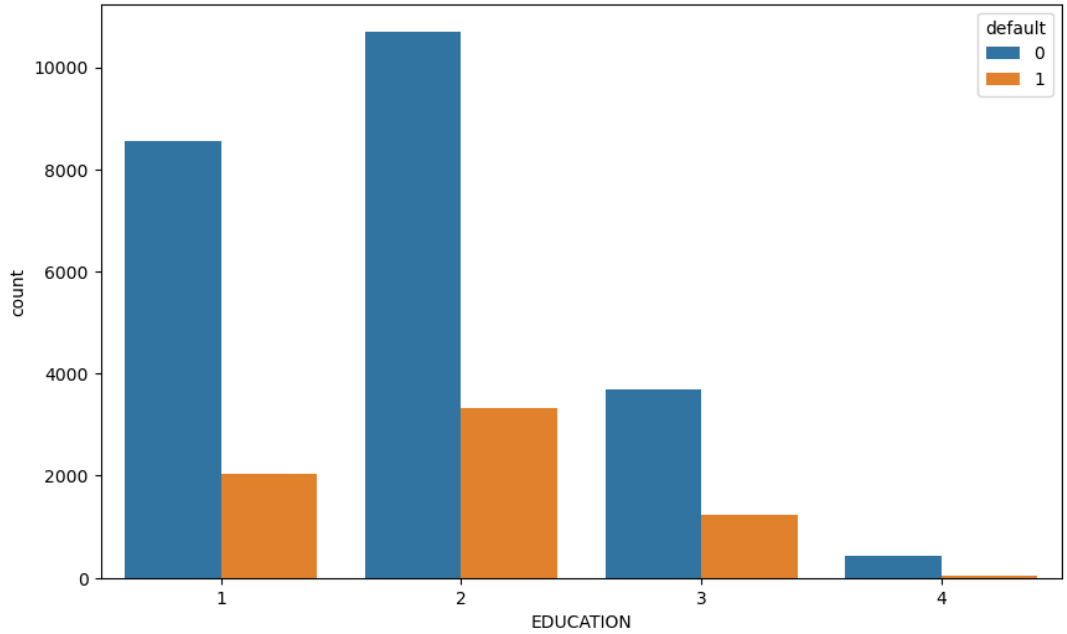
- Loading the dataset
- Checking the shape
- Checking info
- Getting statistical information from describe
- Checking and handling null values
- Checking duplicate values
- Taking detailed look at number of columns
- Renaming columns as per our convenience
- Taking count of the categorical values for entities
- Finally, proceeding with our visualization

2.1 Gender Variable

From the below gender to count plot we can conclude that females tend to have more default than men.



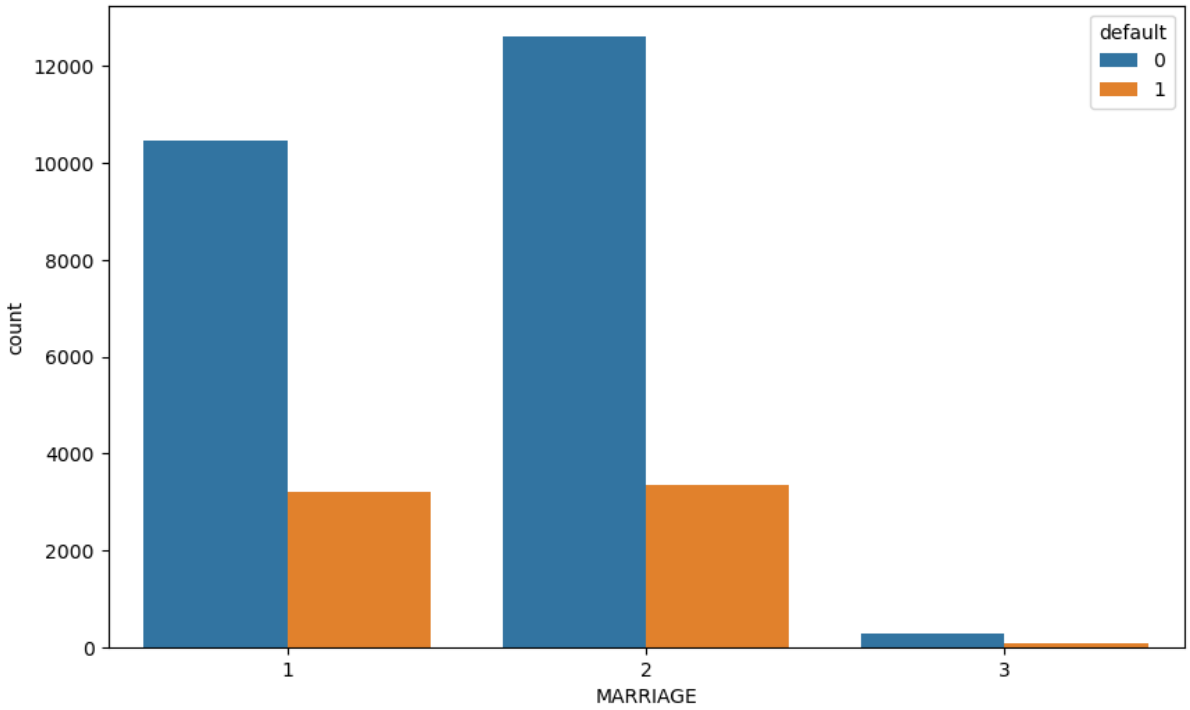
2.2 Education Variable



People having university level education are defaulting more than others.

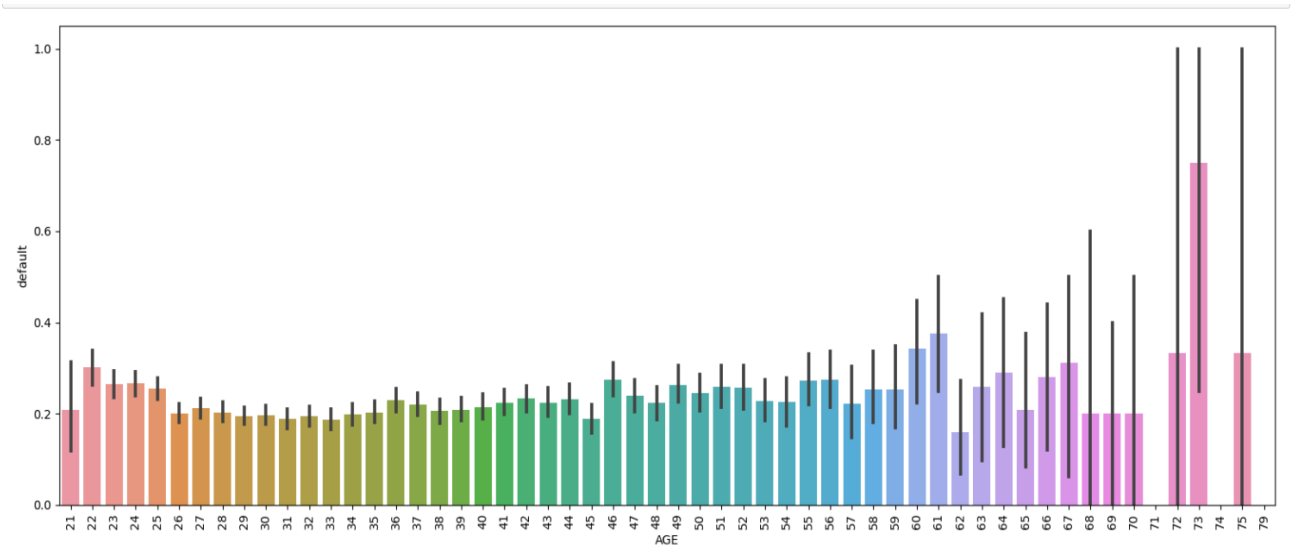
2.3 Marriage Variable

From the marriage to count plot we can examine that single , non-married people are more likely to default the payments.



2.4 Age Variable:

From the Age to default plot we can notice the defaulters by the age criteria.



3. Modeling

Modeling in machine learning refers to the process of creating a mathematical representation, or model, that captures the underlying patterns and relationships in a given dataset. This model is trained using algorithms and techniques that enable it to learn from the data and make predictions or decisions.

Here are the key steps involved in the modeling process:

- Defining the problem: Clearly articulate the problem you want to solve using machine learning. Identify the type of problem, such as classification, regression, clustering, or reinforcement learning.

- Data collection and preprocessing: Gather the relevant data for your problem domain. This data may consist of structured or unstructured data, text, images, or other types. Preprocess the data by cleaning, transforming, and normalizing it to make it suitable for modeling.

- Model selection: Choose an appropriate machine learning algorithm or model that suits your problem and data. The selection depends on various factors, including the type of problem, the available data, and the desired outcomes. Common models include decision trees, support vector machines, neural networks, and ensemble methods.

- Training the model: Use the training dataset to train the selected model. During the training process, the model learns the patterns and relationships in the data by adjusting its internal parameters. This is typically done through an optimization process that minimizes a predefined loss function.

- Model evaluation: Assess the performance of the trained model using evaluation metrics specific to the problem type. For example, classification models can be evaluated using metrics like accuracy, precision, recall, and F1 score. Regression models may be evaluated using metrics such as mean squared error or R-squared.

- Model tuning and optimization: Fine-tune the model by adjusting hyperparameters, such as learning rate, regularization strength, or the number of hidden layers in a neural network. This process aims to optimize the model's performance on unseen data and prevent overfitting or underfitting.

3.1 LOGISTIC REGRESSION:

- Logistic regression is a statistical model used for binary classification problems. It is a supervised learning algorithm that predicts the probability of an instance belonging to a certain class. Despite its name, logistic regression is primarily used for classification rather than regression.

- During training, the logistic regression model estimates the weights that minimize the difference between the predicted probabilities and the true class labels in the training data. This is typically done using maximum likelihood estimation or other optimization algorithms, such as gradient descent.

- Logistic regression has several advantages. It is computationally efficient, easy to implement, and provides interpretable coefficients that indicate the influence of each input feature on the predicted probability. Additionally, logistic regression can handle both categorical and continuous input features, and it can be extended to handle multi-class classification problems.

- However, logistic regression assumes a linear relationship between the input features and the log-odds of the positive class. If the relationship is not linear, feature transformations or more complex models may be necessary. Additionally, logistic regression is sensitive to outliers and may struggle with imbalanced datasets.

- Logistic regression is widely used in various domains, including healthcare, finance, marketing, and social sciences, where binary classification problems are common. Many machine learning libraries and frameworks provide implementations of logistic regression, making it easily accessible for practitioners.

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

logistic = LogisticRegression(random_state = 42)
logistic.fit(X_train,y_train)
y_pred1 = logistic.predict(X_val)
acc_1 = accuracy_score(y_val, y_pred1)
score1 = round(acc_1*100,4)
print('The accuracy score of logistic regression is {}%'.format(score1))
```

```
The accuracy score of logistic regression is 81.25%
```

3.2 KNN:

- KNN, short for k-nearest neighbors, is a non-parametric algorithm used for both classification and regression tasks in machine learning. It is a simple yet powerful algorithm that relies on the similarity of instances in the feature space.

- In KNN, the training dataset consists of labeled instances with their associated features. When making predictions for a new, unlabeled instance, KNN looks at the k nearest neighbors in the training data based on a distance metric (e.g., Euclidean distance). The predicted class or value for the new instance is determined by majority voting (for classification) or averaging (for regression) among its k nearest neighbors.

- KNN does not involve a training phase like many other algorithms. It directly uses the training data to make predictions for new instances. However, it may require preprocessing steps like feature scaling to ensure that all features contribute equally to the distance calculation.

- KNN is known for its simplicity, interpretability, and ability to handle non-linear decision boundaries. However, it can be computationally expensive for large datasets, as the distance calculations need to be performed for each prediction. Additionally, KNN assumes that nearby instances are likely to have similar labels or values, which may not hold in all cases.

- Overall, KNN is a versatile algorithm that can be used in various domains and works well for datasets with discernible patterns and relatively small feature spaces.

K Nearest Neighbor

```
from sklearn.neighbors import KNeighborsClassifier

k_neighbors = KNeighborsClassifier()
k_neighbors.fit(X_train,y_train)
y_pred2 = k_neighbors.predict(X_val)
acc_2 = accuracy_score(y_val, y_pred2)
score2 = round(acc_2*100,4)
print('The accuracy score of k nearest neighbors is {}%'.format(score2))
```

```
The accuracy score of k nearest neighbors is 79.3542%
```

3.3 NAÏVE BAYES:

- Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with an assumption of independence between features. It is a simple yet effective algorithm that is widely used for text classification and other classification tasks.

- The "naive" in Naive Bayes refers to the assumption of feature independence. Although this assumption is often violated in real-world scenarios, Naive Bayes can still perform well and provide fast and efficient predictions.

- There are different variants of Naive Bayes based on the type of features and distribution assumptions. The most common ones include:

- Multinomial Naive Bayes: Suitable for discrete features, such as word counts in text classification tasks.

- Gaussian Naive Bayes: Suitable for continuous features that follow a Gaussian (normal) distribution.

- Bernoulli Naive Bayes: Suitable for binary features, such as presence or absence of a particular attribute.

- Naive Bayes is known for its simplicity, efficiency, and ability to handle high-dimensional data. It can work well with small to medium-sized datasets and is particularly useful for text classification tasks, spam filtering, sentiment analysis, and document categorization. However, the assumption of feature independence can limit its performance when the features are dependent on each other.

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
naive_bayes = GaussianNB()
naive_bayes.fit(X_train,y_train)
y_pred3 = naive_bayes.predict(X_val)
acc_3 = accuracy_score(y_val, y_pred3)
score3 = round(acc_3*100,4)
print('The accuracy score Naive bayes is {}%'.format(score3))
```

```
The accuracy score Naive bayes is 76.0417%
```

3.4 DECISION TREES:

- A decision tree is a supervised machine learning algorithm that can be used for both classification and regression tasks. It models decisions and their possible consequences as a tree-like structure, where each internal node represents a feature or attribute, each branch represents a decision rule, and each leaf node represents a class label or a predicted value.

- Decision trees have several advantages:
- They are easy to understand and interpret, as the decision rules and splits are represented graphically.
- Decision trees can handle both categorical and numerical features.
- They can capture non-linear relationships and interactions between features.
- Decision trees can handle missing values by using surrogate splits.
- They can handle both classification and regression tasks.

- However, decision trees can be prone to overfitting, especially if the tree grows too deep or if the dataset has noisy or irrelevant features. To mitigate overfitting, techniques like pruning, setting a maximum depth, or using ensemble methods such as random forests or gradient boosting can be employed.

- Overall, decision trees are widely used due to their simplicity, interpretability, and effectiveness in various domains, such as finance, healthcare, and customer segmentation.

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train,y_train)
y_pred4 = decision_tree.predict(X_val)
acc_4 = accuracy_score(y_val, y_pred4)
score4 = round(acc_4*100,4)
print('The accuracy score of Decision tree is {}%'.format(score4))
```

```
The accuracy score of Decision tree is 71.625%
```

3.5 RANDOM FOREST:

- Random Forest is an ensemble machine learning algorithm that combines the predictions of multiple decision trees to make more accurate and robust predictions. It is a versatile algorithm that can be used for both classification and regression tasks.

The key advantages of Random Forest are:

- Reduced overfitting: The randomness in the feature selection and the bootstrap aggregating process helps reduce overfitting by introducing diversity and reducing the variance in the ensemble.

- Robustness to outliers and noise: Random Forest can handle outliers and noisy data by considering multiple decision trees and aggregating their predictions.

- Feature importance: Random Forest provides a measure of feature importance based on the average reduction in impurity or the average decrease in the splitting criterion across all the trees. This information can be useful for feature selection and understanding the importance of different features in the prediction process.

- Random Forest is widely used in various domains due to its robustness, accuracy, and ability to handle high-dimensional datasets. It is particularly effective when there are complex interactions or non-linear relationships between features. Additionally, Random Forest can handle missing values and does not require extensive data preprocessing.

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(random_state = 42)
random_forest.fit(X_train, y_train)

ye = random_forest.predict(X_train)
print(accuracy_score(y_train, ye))

y_pred5 = random_forest.predict(X_val)
acc_5 = accuracy_score(y_val, y_pred5)
score5 = round(acc_5*100, 4)
print('The accuracy score of random forest is {}%'.format(score5))
```

```
0.9994791666666667
The accuracy score of random forest is 80.8125%
```

.

3.6 SVM:

- SVM, or Support Vector Machines, is a powerful supervised machine learning algorithm used for both classification and regression tasks. It is particularly effective when dealing with complex decision boundaries and high-dimensional data.

- The main idea behind SVM is to find a hyperplane that separates the data points of different classes with the largest margin. The hyperplane is a decision boundary that divides the feature space into different regions corresponding to different classes. SVM aims to maximize the margin between the support vectors (data points closest to the decision boundary) of different classes.

- SVM can also be extended to handle multiclass classification by using techniques such as one-vs-rest or one-vs-one.

SVM has several advantages, including:

- Ability to handle high-dimensional data and complex decision boundaries.
- Robustness against overfitting due to the use of the margin and regularization.
- Flexibility through the choice of different kernel functions.
- Theoretical foundation and strong generalization performance.

- However, SVM's main limitations include the need for careful tuning of hyperparameters, sensitivity to the choice of kernel function and parameters, and computational complexity for large datasets.

SVM

```
from sklearn.svm import SVC
model6 = SVC()
model6.fit(X_train, y_train)
y_pred6 = model6.predict(X_val)
acc_6 = accuracy_score(y_val, y_pred6)
score6=round(acc_6*100,4)

print("The accuracy score of SVM is {}%:".format(score6))
```

```
The accuracy score of SVM is 81.125%:
```

4. Model Evaluation

- From all the applied machine learning algorithms we can initially see that our model for Logistic Regression and SVM gives us the highest accuracy of 81.25 and 81.12.

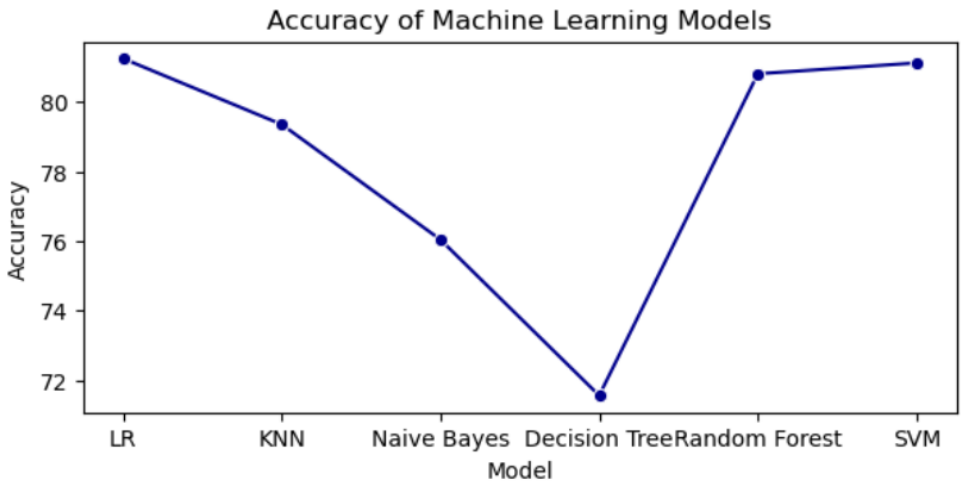
```
data = {'Machine Learning Model':['Logistic Regression','K Nearest Neighbor','Naive Bayes',
                                  'Decision Tree','Random Forest','SVM'],
        'Accuracy Score':[score1,score2,score3,score4,score5,score6]}

df_new= pd.DataFrame(data)
df_new
```

	Machine Learning Model	Accuracy Score
0	Logistic Regression	81.2500
1	K Nearest Neighbor	79.3542
2	Naive Bayes	76.0417
3	Decision Tree	71.5625
4	Random Forest	80.8125
5	SVM	81.1250

```
models = ['LR','KNN','Naive Bayes','Decision Tree','Random Forest','SVM']
accuracies = [score1, score2,score3,score4,score5,score6]

plt.figure(figsize=(7,3))
sns.lineplot(x=models, y=accuracies, marker='o', color='darkblue')
plt.title('Accuracy of Machine Learning Models')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```



Here , we can look at the accuracy plotting of our ML models.

5. Bagging Concept.

- Bagging, short for Bootstrap Aggregating, is a machine learning ensemble technique that combines multiple models to improve the overall performance and robustness of predictions. It is commonly used in classification and regression tasks.

- The main idea behind bagging is to create multiple subsets of the training data through a process called bootstrapping, where each subset is generated by sampling the original dataset with replacement. This means that some instances may appear multiple times in a subset, while others may be omitted. Then, a separate model is trained on each subset using a base learning algorithm.

- During the prediction phase, each model independently makes predictions on unseen data. For classification tasks, the final prediction is typically determined by majority voting, where the class that receives the most votes from the individual models is selected. For regression tasks, the final prediction is often the average of the predictions made by the individual models.

- The key advantage of bagging is that it reduces variance and helps to mitigate overfitting. By training multiple models on slightly different subsets of data, bagging introduces diversity among the models, which leads to more robust predictions. Additionally, bagging can improve the generalization ability of the ensemble by reducing the impact of outliers or noisy instances in the training data.

- Overall, bagging is a powerful technique for improving the performance and stability of machine learning models by combining multiple models trained on bootstrapped subsets of the data.

Logistic Regression

```
model_bgc = BaggingClassifier(LogisticRegression(),n_estimators=10,random_state=42,oob_score=True)
```

```
model_bgc.fit(X_train,y_train)
```

```
BaggingClassifier(base_estimator=LogisticRegression(), oob_score=True,
                  random_state=42)
```

```
model_bgc.oob_score_
```

```
0.8180729166666667
```

```
model_bgc_cv = cross_val_score(model_bgc,X,y,cv=5)
model_bgc_cv
```

```
array([0.80433333, 0.808     , 0.81966667, 0.8265    , 0.82216667])
```

```
accuracy1 =np.average(model_bgc_cv)
score_bg_lr=round(accuracy1*100,4)

print("The accuracy score of Logistic Regression after bagging is {}%:".format(score_bg_lr))
```

```
The accuracy score of Logistic Regression after bagging is 81.6133%:
```

SVM

```
model_bgc1 = BaggingClassifier(SVC(),n_estimators=10,random_state=42,oob_score=True)
```

```
model_bgc1.fit(X_train,y_train)
```

```
BaggingClassifier(base_estimator=SVC(), oob_score=True, random_state=42)
```

```
model_bgc1.oob_score_
```

```
0.8156770833333333
```

```
model_bgc1_cross_val = cross_val_score(model_bgc1,X,y,cv=3)
model_bgc1_cross_val
```

```
array([0.8082, 0.8117, 0.8208])
```

```
np.average(model_bgc1_cross_val)
```

```
0.8135666666666665
```

```
accuracy2 =np.average(model_bgc1_cross_val)
score_bg_svm=round(accuracy2*100,4)

print("The accuracy score of SVM after bagging is {}%:".format(score_bg_svm))
```

```
The accuracy score of SVM after bagging is 81.3567%:
```

6. Conclusion

- From the above machine learning models, we can conclude that Logistic Regression gives us the highest accuracy.
- After implementation of bagging concept, Logistic Regression still tops is terms of accuracy with 81.61 %.
- Based on the exploratory data analysis, we discover that human characteristics are not the most important predictors of default.
- Model can be improved with more data and computational resources.
- We can also see that most credit card users are females and so are the highest number of defaulters.
- People aged between 20 to 30 have moderate default conditions.
- People aged above 61 are more likely to default the payments.
- Highly educated persons are less likely to default whereas, less educated people come under the highly defaulted category.

7. References

1. Addo, P.M.; Guegan, D.; Hassani, B. Credit Risk Analysis Using Machine and Deep Learning Models. Risks 2018, 6, 38. https://www.mdpi.com/2227-9091/6/2/38 14

2. Simeon Kostadinov: "My Analysis from 50+ papers on the Application of ML in Credit Lending": https://towardsdatascience.com/my-analysis-from-50-papers-on-the-application-of-ml-incredit-lending-b9b810a3f38

3. Khyati Mahendru: "How to Deal with Imbalanced Data using SMOTE ":https://medium.com/analytics-vidhya/balance-your-data-using-smote-98e4d79fcddb

4. https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset/discussion