‘Credit Score Classification’

Course: M.Sc. Data Science and Spatial Analytics

Subject: Machine learning and Advance Pyhton

By: Sahil Raj

PRN: 22070243040

SYMBIOSIS
INSTITUTE OF GEOINFORMATICS

CONTENTS:

1. Acknowledgment:
2. Abstract and Introduction
3. Dataset
4. Library used and data collection
5. Data Cleaning and Data pre-processing
6. Data Analysis
7. Geopanda Use
8. Data Visualization and Feature Extraction
9. Models Used
10. Model selection
11. Conclusion
12. References

Acknowledgment:

I would like to take this opportunity to thank everyone who helped make this endeavour successful. I owe a debt of appreciation to each and every one of the people and organisations on the list below who have supported, inspired, and led us along the journey.

I want to start by expressing my sincere gratitude to the mentors of my project, Dr. Vidya Patkar and Mr. Sahil Shah, for all of their essential help, advice, and helpful criticism. They helped me keep on track and shape my project by contributing their knowledge and thoughts.

I also appreciate Symbiosis Institute of Geoinformatics for lending me the tools and resources I needed to finish my research.

My sincere gratitude goes out to my friends and coworkers who have supported and encouraged me throughout the endeavour.

I want to express my sincere appreciation to everyone who has contributed to this initiative once more. I also hope that in the future, more people will see the value of our labour.

Abstract:

The credit score is crucial in determining whether an individual can obtain a loan, mortgage or credit card. Credit scores are classified based on the individual's credit history and financial behaviour and are used by financial institutions to assess the risk associated with lending money. In this report, we aim to analyse the credit score dataset and build a machine-learning model to classify credit scores.

INTRODUCTION:

Classifying credit scores is a crucial task in banking and business. It entails assessing a person's or an organisation's creditworthiness based on several variables, including their credit history, income, debt-to-income ratio, and other financial data. The possibility that a borrower will repay their debts on time is determined by their credit score, which is a crucial consideration when setting the terms and conditions of a loan.

In order to analyse vast volumes of data and produce precise forecasts, machine learning algorithms have been extensively used in the classification of credit scores. This study compares the effectiveness of three classification models for classifying credit scores into three groups: good, standard, and poor: "Random Forest, Logistic Regression, and Naive Bayes."

At the end of this study, we aim to shed some light on which classification model would be mostmost helpful precisely forecasting credit scores for people or organisations, which can aid financial institutions in their decision-making.

Dataset:

The given dataset has information about 100,000 and their financial profiles. It contains 27 columns with a mix of data types, including integers, floats, and object dataset is related to credit and loan analysis as it includes features related to payment delay, outstanding debt, credit utilization, and payment behaviour. Be used for classifying the credit score on selected components.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   id                        100000 non-null  int64
 1   customer_id               100000 non-null  int64
 2   month                     100000 non-null  int64
 3   name                      100000 non-null  object
 4   age                       100000 non-null  float64
 5   ssn                       100000 non-null  float64
 6   occupation                100000 non-null  object
 7   annual_income             100000 non-null  float64
 8   monthly_inhand_salary     100000 non-null  float64
 9   num_bank_accounts         100000 non-null  float64
 10  num_credit_card           100000 non-null  float64
 11  interest_rate             100000 non-null  float64
 12  num_of_loan               100000 non-null  float64
 13  type_of_loan              100000 non-null  object
 14  delay_from_due_date       100000 non-null  float64
 15  num_of_delayed_payment    100000 non-null  float64
 16  changed_credit_limit      100000 non-null  float64
 17  num_credit_inquiries      100000 non-null  float64
 18  credit_mix                100000 non-null  object
 19  outstanding_debt          100000 non-null  float64
 20  credit_utilization_ratio  100000 non-null  float64
 21  credit_history_age        100000 non-null  float64
 22  payment_of_min_amount     100000 non-null  object
 23  total_emi_per_month       100000 non-null  float64
 24  amount_invested_monthly   100000 non-null  float64
 25  payment_behaviour         100000 non-null  object
 26  monthly_balance           100000 non-null  float64
 27  credit_score              100000 non-null  object
dtypes: float64(18), int64(3), object(7)
```

LIBRARIES USED IN PYTHON:

- **Pandas**
- **Numpy**
- **Matplotlib**
- **Seaborn**
- **Psycopg2**
- **Sklearn**
- **Geopandas**

```
import warnings
warnings.filterwarnings("ignore")
warnings.filterwarnings("ignore", category=UserWarning, module='numpy')
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import psycopg2 as psy
import geopandas as gpd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.ensemble import RandomForestClassifier
```

Data collection:

Data was collected from Kaggle and it was pushed to PostgreSQL database using psycopg2 and then from there the data was fetched into data frame using pandas library.

In [4]:
```
connection=psy.connect(database='mini_project',user='postgres',password='sahilrj802')
cursor=connection.cursor()
```

In [10]:
```
cursor.execute('drop table if exists credit_score')
cursor.execute("""
    create table credit_score (

                                ID integer,Customer_ID  integer,Month integer,Name text,Age float,SSN float,Occupation text,
  Annual_Income float,
   Monthly_Inhand_Salary float,
   Num_Bank_Accounts float,
  Num_Credit_Card float,
  Interest_Rate float,
  Num_of_Loan float,
  Type_of_Loan text,
  Delay_from_due_date float,
  Num_of_Delayed_Payment float,
   Changed_Credit_Limit float,
   Num_Credit_Inquiries float,
   Credit_Mix text,
  Outstanding_Debt float,
   Credit_Utilization_Ratio float,
  Credit_History_Age float,
  Payment_of_Min_Amount text,
  Total_EMI_per_month float,
  Amount_invested_monthly float,
  Payment_Behaviour text,
  Monthly_Balance float,
  Credit_Score text
    )
""")
```

```
]:
for i in data.index:
    vals=[data.at[i,col] for col in list(data.columns)]
    query='''insert into credit_score values('%s','%s','%s','%s','%s','%s','%s','%s','%s',
    '%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s
    ','%s','%s','%s','%s'
    ,'%s','%s','%s','%s')'''%(vals[0],vals[1],vals[2],vals[3],vals[4],vals[5],vals[6],vals[8],vals[7],vals[9],vals[10],vals[11],
    cursor.execute(query)
    connection.commit()
```

```
query1='SELECT *FROM credit_score'
data = pd.read_sql_query(query1, connection)
data.head()
```

	id	customer_id	month	name	age	ssn	occupation	annual_income	monthly_inhand_salary	num_bank_accounts	...	credit_mix	outstanding
0	5634	3392	1	Aaron Maashoh	23.0	821000265.0	Scientist	1824.843333	19114.12	3.0	...	Good	8
1	5635	3392	2	Aaron Maashoh	23.0	821000265.0	Scientist	1824.843333	19114.12	3.0	...	Good	8
2	5636	3392	3	Aaron Maashoh	23.0	821000265.0	Scientist	1824.843333	19114.12	3.0	...	Good	8
3	5637	3392	4	Aaron Maashoh	23.0	821000265.0	Scientist	1824.843333	19114.12	3.0	...	Good	8
4	5638	3392	5	Aaron Maashoh	23.0	821000265.0	Scientist	1824.843333	19114.12	3.0	...	Good	8

5 rows × 28 columns

Data Cleaning:

The dataset that was available was cleaned and did not have any inconsistent format and no null values. Hence there was no need of data cleaning.

```
#checking for the null values
print(data.isnull().sum())
```

```
id                          0
customer_id                 0
month                       0
name                        0
age                         0
ssn                         0
occupation                  0
annual_income               0
monthly_inhand_salary       0
num_bank_accounts           0
num_credit_card             0
interest_rate               0
num_of_loan                 0
type_of_loan                0
delay_from_due_date         0
num_of_delayed_payment      0
changed_credit_limit        0
num_credit_inquiries        0
credit_mix                  0
outstanding_debt            0
credit_utilization_ratio    0
credit_history_age          0
payment_of_min_amount       0
total_emi_per_month         0
amount_invested_monthly     0
payment_behaviour           0
monthly_balance             0
credit_score                0
dtype: int64
```

Data Pre-processing:

- Dropping the features that are not required for the analysis.

```
In [99]: data = data.drop(['id', 'customer_id', 'ssn', 'name', 'month','age'], axis=1)
         data.describe().T
```

- Data mapping was also done :

```
#The credit mix feature tells about the types of credits and loans you have taken.
#since the column is categorical we need to transform into numbers by mapping the categories with 0,1,2
data["credit_mix"] = data["credit_mix"].map({"Standard": 1,
                                             "Good": 2,
                                             "Bad": 0})
```

- The target variable was label encoded before fitting in the models.

```
le = LabelEncoder()
data['credit_score'] = le.fit_transform(data['credit_score'])
y = np.array(data[["credit_score"]])
```

Data Analysis:

Descriptive analysis of the dataset:

]:

	count	mean	std	min	25%	50%	75%	max
annual_income	100000.0	4197.270835	3186.432497	303.645417	1626.594167	3095.905000	5957.715000	15204.633333
monthly_inhand_salary	100000.0	50505.123449	38299.422093	7005.930000	19342.972500	36999.705000	71683.470000	179987.280000
num_bank_accounts	100000.0	5.368820	2.593314	0.000000	3.000000	5.000000	7.000000	11.000000
num_credit_card	100000.0	5.533570	2.067098	0.000000	4.000000	5.000000	7.000000	11.000000
interest_rate	100000.0	14.532080	8.741330	1.000000	7.000000	13.000000	20.000000	34.000000
num_of_loan	100000.0	3.532880	2.446356	0.000000	2.000000	3.000000	5.000000	9.000000
delay_from_due_date	100000.0	21.081410	14.804560	0.000000	10.000000	18.000000	28.000000	62.000000
num_of_delayed_payment	100000.0	13.313120	6.237166	0.000000	9.000000	14.000000	18.000000	25.000000
changed_credit_limit	100000.0	10.470323	6.609481	0.500000	5.380000	9.400000	14.850000	29.980000
num_credit_inquiries	100000.0	5.798250	3.867826	0.000000	3.000000	5.000000	8.000000	17.000000
outstanding_debt	100000.0	1426.220376	1155.129026	0.230000	566.072500	1166.155000	1945.962500	4998.070000
credit_utilization_ratio	100000.0	32.285173	5.116875	20.000000	28.052567	32.305784	36.496663	50.000000
credit_history_age	100000.0	221.220460	99.680716	1.000000	144.000000	219.000000	302.000000	404.000000
total_emi_per_month	100000.0	107.699208	132.267056	0.000000	29.268886	66.462304	147.392573	1779.103254
amount_invested_monthly	100000.0	55.101315	39.006932	0.000000	27.959111	45.156550	71.295797	434.191089
monthly_balance	100000.0	392.697586	201.652719	0.007760	267.615983	333.865366	463.215683	1183.930696

The data provided contains information on several financial features of 100,000 individuals.

The first feature is the annual income, which represents the total income earned by an individual in a year. The data shows that the mean yearly income is 4197.27 units, with a minimum payment of 303.65 units and a maximum gain of 15204.63 units.

The second feature is the monthly in-hand salary, which represents the net salary earned by an individual after deductions. The data shows that the mean monthly in-hand salary is 50505.12 units, with a minimum wage of 7005.93 units and a maximum salary of 179987.28 units.

The third feature is the number of bank accounts an individual holds, with a mean value of 5.37, ranging from 0 to 11 accounts.

The fourth feature is the number of credit cards an individual holds, with a mean value of 5.53 cards ranging from 0 to 11.

The fifth feature is the interest rate, which represents the percentage charged on a loan or credit account. The data shows that the mean interest rate is 14.53%, with a minimum rate of 1% and a maximum rate of 34%. The sixth feature is the number of loans an individual has, with a mean value of 3.53 loans ranging from 0 to 9.

The seventh feature is the delay from the due date, which represents the days a payment is delayed. The data shows that the mean delay from the due date is 21.08 days, ranging from 0 to 62 days.

The eighth feature is the number of delayed payments an individual has, which has a mean value of 13.31, ranging from 0 to 25.

The ninth feature is the changed credit limit, which represents the change in credit limit on a credit account. The data shows the mean change in credit limit is 10.47 units, ranging from 0.5 to 29.98 units.

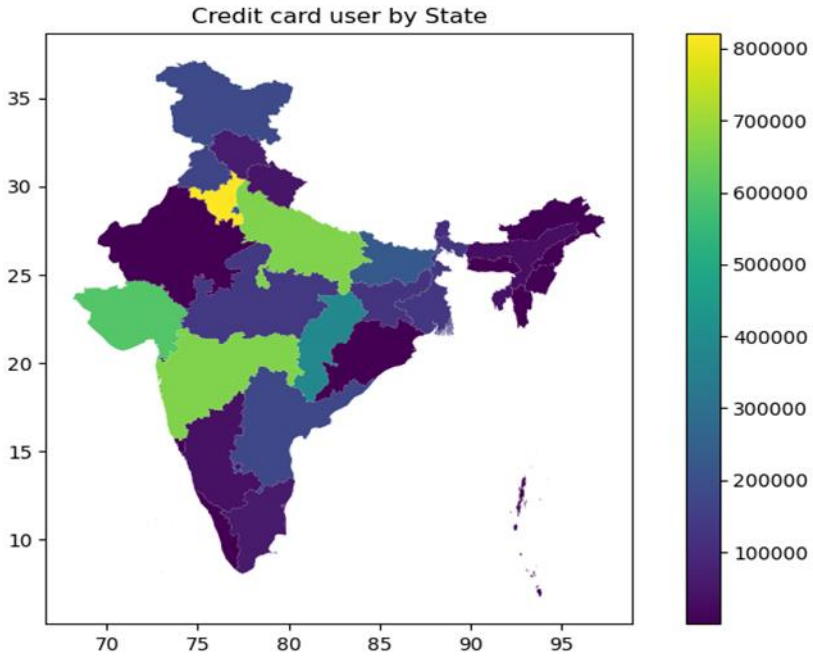
The tenth feature is the number of credit inquiries, which represents the number of times an individual's to 434.19 units.

The sixteenth and final feature is the monthly balance, which represents an individual's remaining balance after all expenses and payments have been made for a given month. The data shows that the mean monthly balance is 392.70 units, ranging from 0.007 to 1183.93 units.

Overall, this dataset provides a detailed picture of various financial factors that may affect an individual's credit score. The range and distribution of each feature can be helpful in creating models to predict creditworthiness or for identifying areas where an individual may need to improve their financial habits.

Geopanda use:

Lets have Look into the number of credit card user in India state wise



As you can see in the heat map of India, there is credit card user all over India, with the highest in Delhi and the least in the North-east region. Hence we can say that credit score is very much important for getting a credit card, so let's try to classify the credit score.

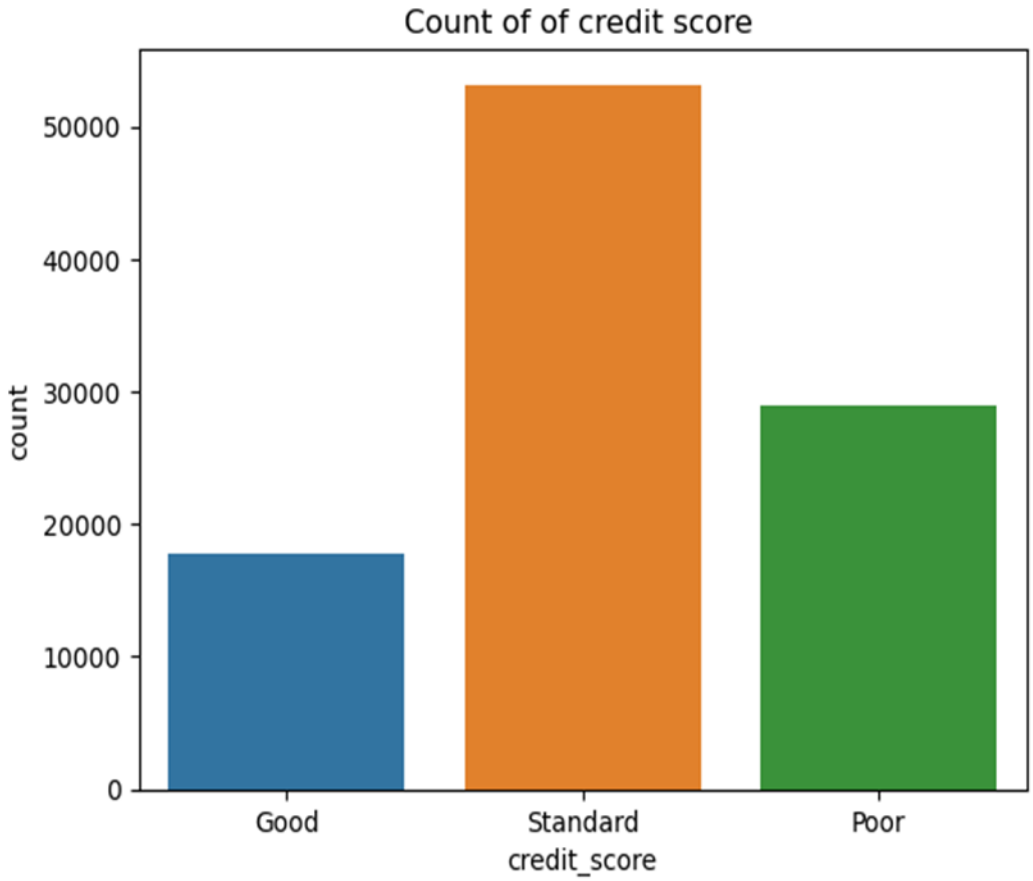
For visualising the credit card user in a dataset with credit card user was taken and India shape file was handled. The dataset for the credit card user was read in data frame using pandas and shape file was read in geo data frame and merged by mapping it with state name in the geo data frame. Then the geo data frame now contains credit card user statewise and then it was plotted using plot function of geopandas.

```
#reading the shape file
file="C:\\Users\Sahil Raj\Desktop\google data anlytics\shp file\INDIA_states.shp"
gdf=gpd.read_file(file)
credit=pd.read_excel(r"C:\Users\Sahil Raj\Desktop\credit score.xlsx")
#merging the data with the shape file
gdf=gdf.merge(credit, on="ST_NAME")
gdf
```

	ST_NAME	geometry	Credit card user
0	ANDAMAN AND NICOBAR ISLANDS	MULTIPOLYGON (((92.89889 12.91583, 92.89917 12...	5340
1	Andhra Pradesh	POLYGON ((83.94319 18.21431, 83.94236 18.21431...	176646
2	Arunachal Pradesh	POLYGON ((94.86086 27.73948, 94.86603 27.73624...	4920
3	Assam	POLYGON ((95.59917 27.22961, 95.59009 27.22952...	31520

DATA VISUALISATION:

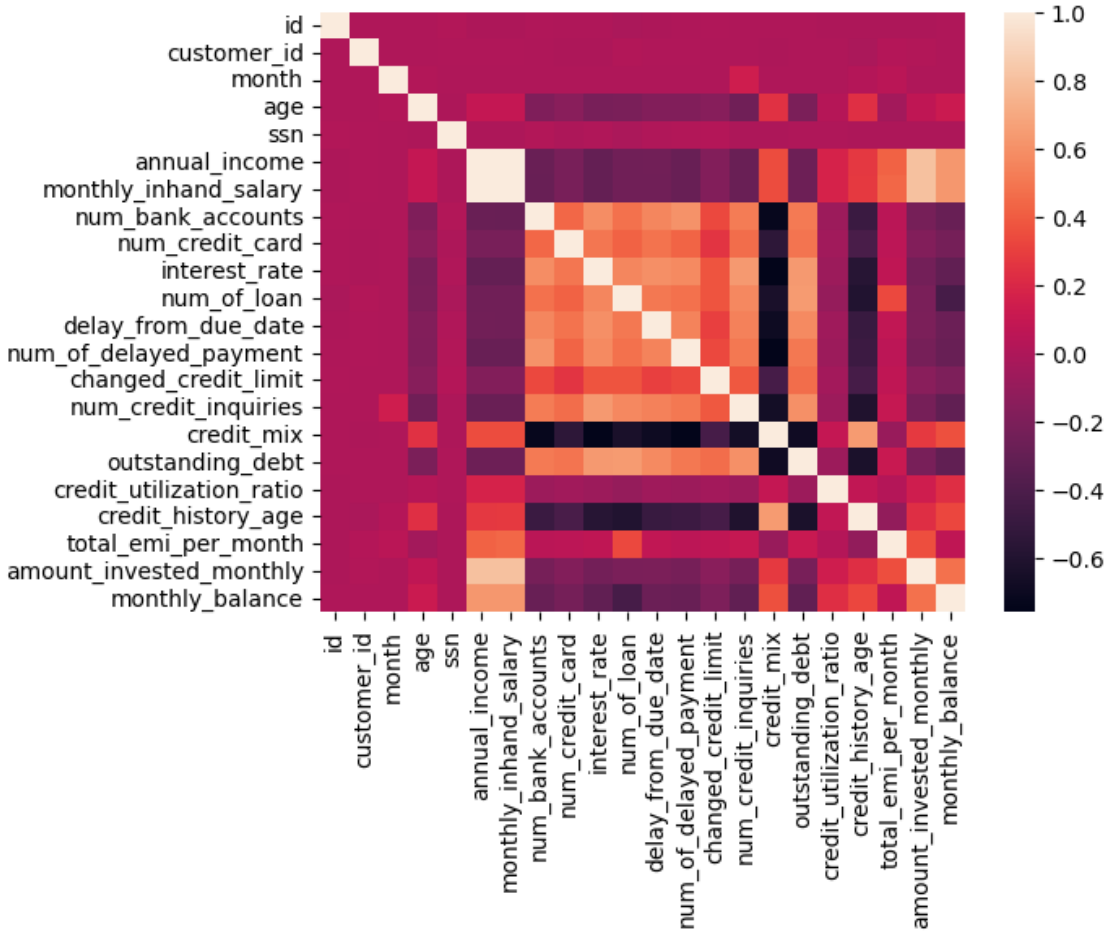
Credit Score column values:



- **We can see that there is a significant number of Good, Standard and Poor credit score classes.**

Feature selection:

Correlation matrix:



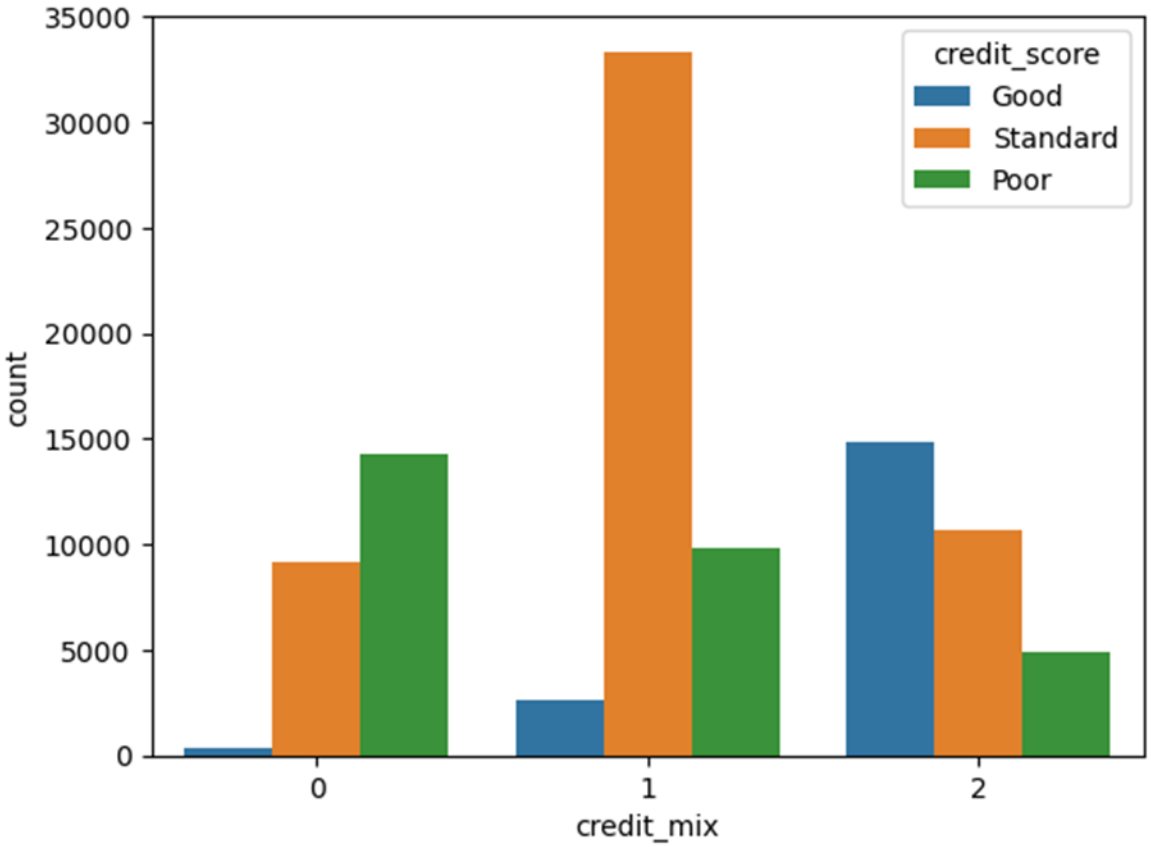
Based on the correlation matrix, we can see that some variables positively correlate with the variable('Credit_mix') and some negatively correlated.

The highest positive correlation is with the 'outstanding_debt' variable (0.186), and the highest negative correlation is with the 'delay_from_due_date' variable (-0.152).

Some other variables that have good correlation with the target variable are 'num_of_delayed_payment' (-0.105), 'num_credit_inquiries' (-0.081), 'credit_utilization_ratio' (0.064) and 'num_of_loan' (-0.064).

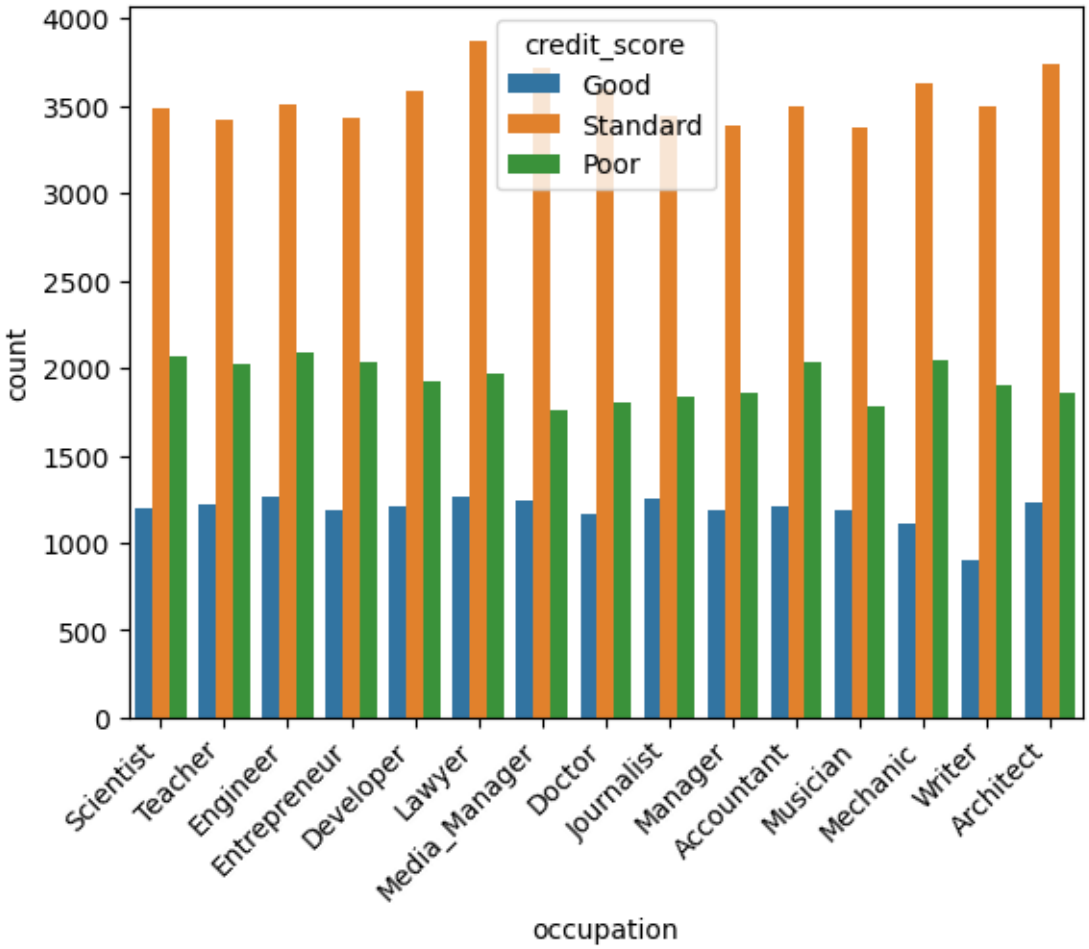
However, we should be cautious when interpreting the results of the correlation matrix as it only measures the linear relationship between variables and does not capture any non-linear relationships or causality between the variables. Additionally, it is essential to note that correlation does not imply causation, and other factors may drive the relationship between the variables.

Based on the above correlation heatmap, lets have a look into the most essential feature, that is credit_mix with the target variable credit_score.



Credit mix is an essential feature in getting a credit score; hence we can take this feature since having a good credit score having good credit mix is essential. We can interpret that correlation between credit mix, and credit score has a positive relation with each other based on this, lets see the correlation matrix.

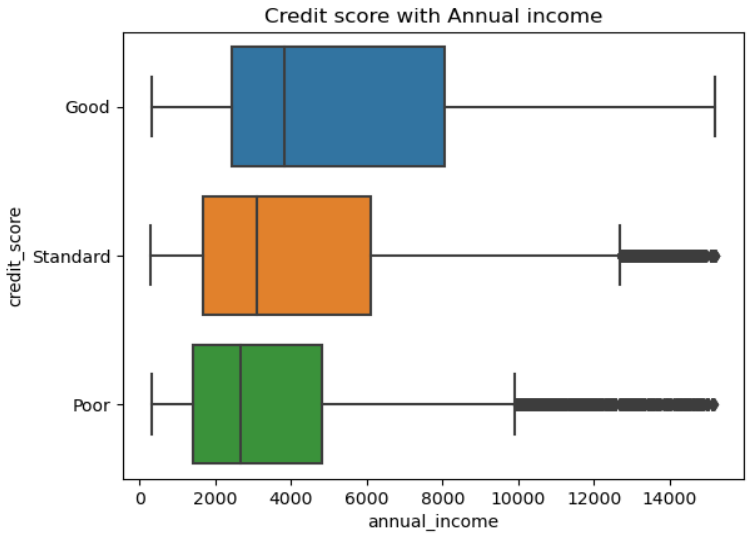
Credit score count with occupation:



Here you can see that the occupation of a person does not impact on the credit score hence we can drop this feature while training the model.

Credit score with Annual income:

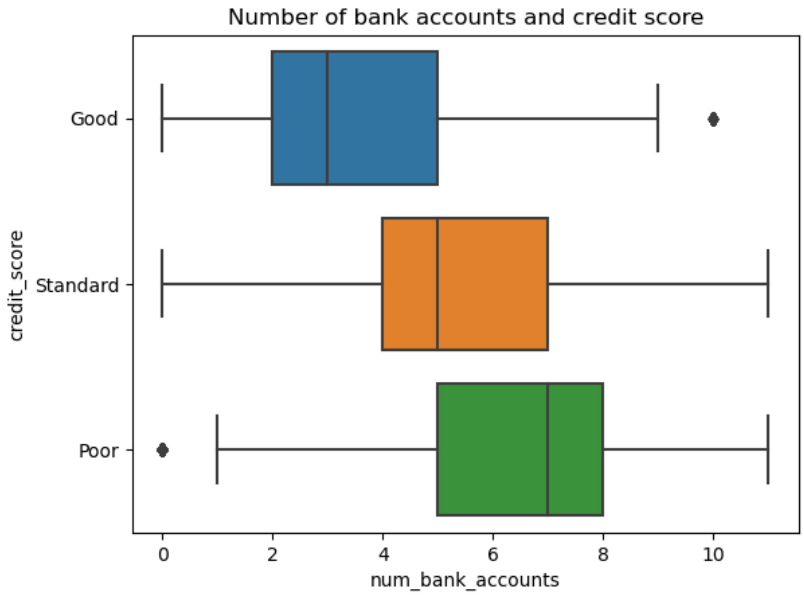
With the figure we can interpret that annual income is very important feature.



Monthly in hand salary with credit score:

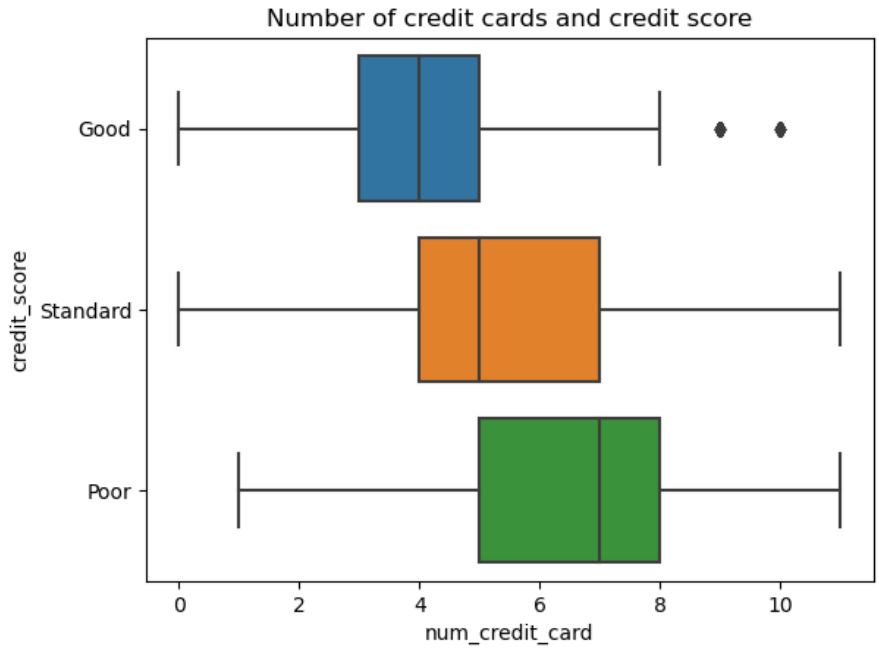
If the salary is less then the score is poor.

Number Of Bank Accounts Impacts The Credit Score:



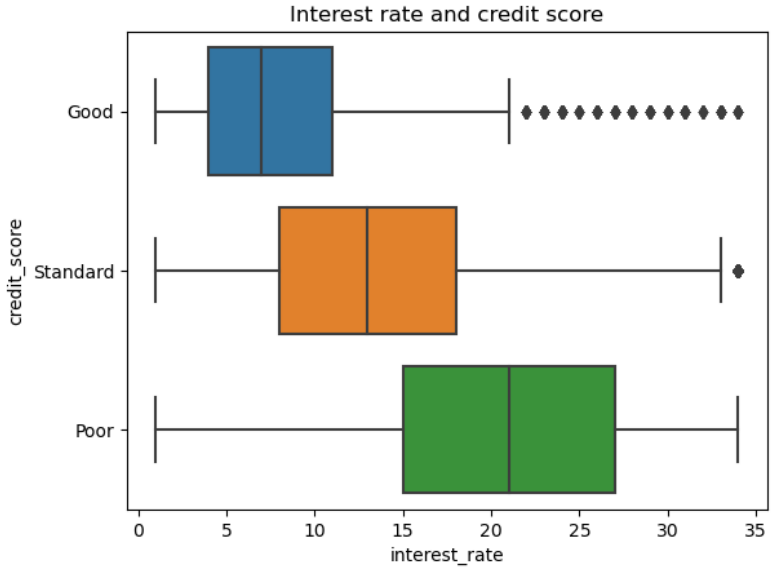
The more number of bank accounts the poor your credit score is. Feature is important for the model.

Number Of Credit Card Impacts The Credit Score Or Not:



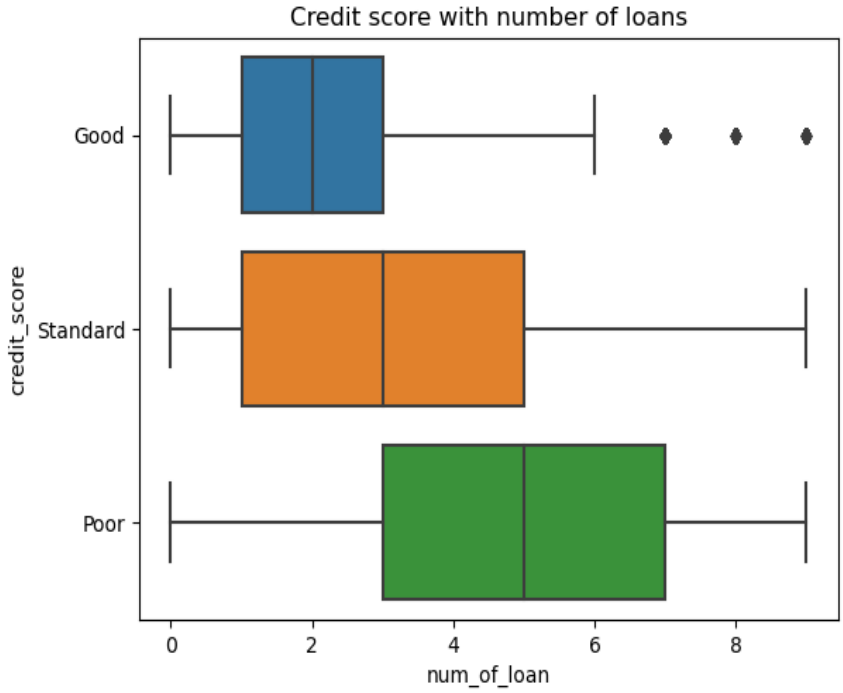
Customer should not have more credit cards, it will decrease the credit score.

Interest Rate Affect The Credit Score Or Not:



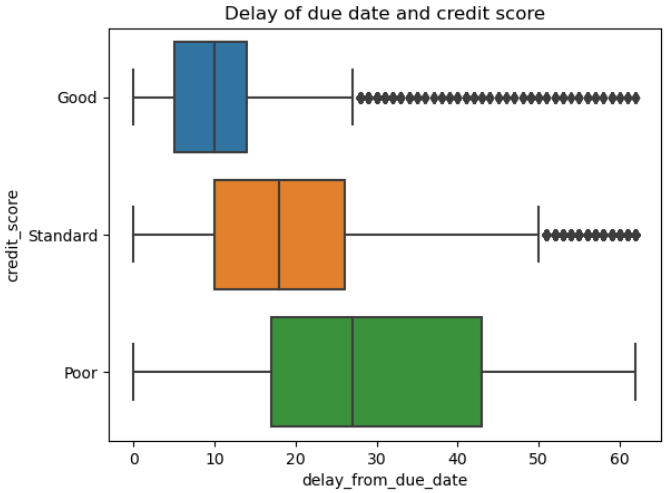
Interest rate on emi or loan should be low that is having 4% to 11% rate will have good credit score as compared to others,hence this feature is also impactful.

Number Of Loans Affect The Credit Score Or Not:



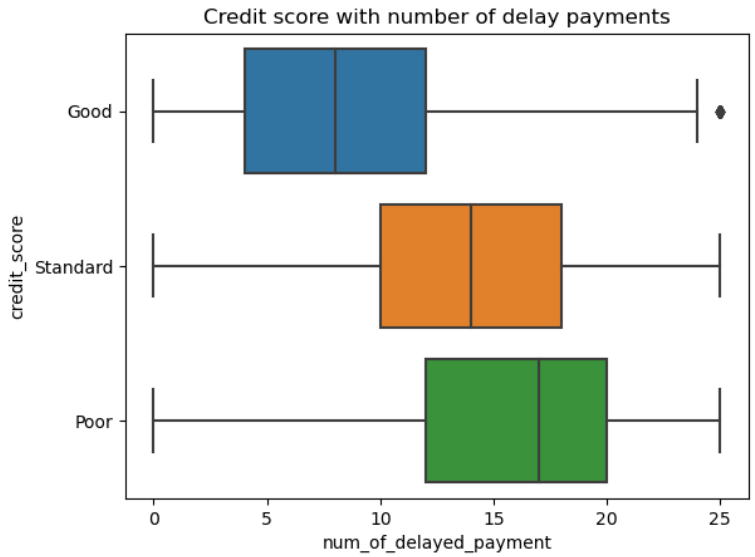
For good credit score we should not exceed 2-3 loans at a time ,hence it is useful features.

Delay Of Due Date Impacts Credit Score Or Not:



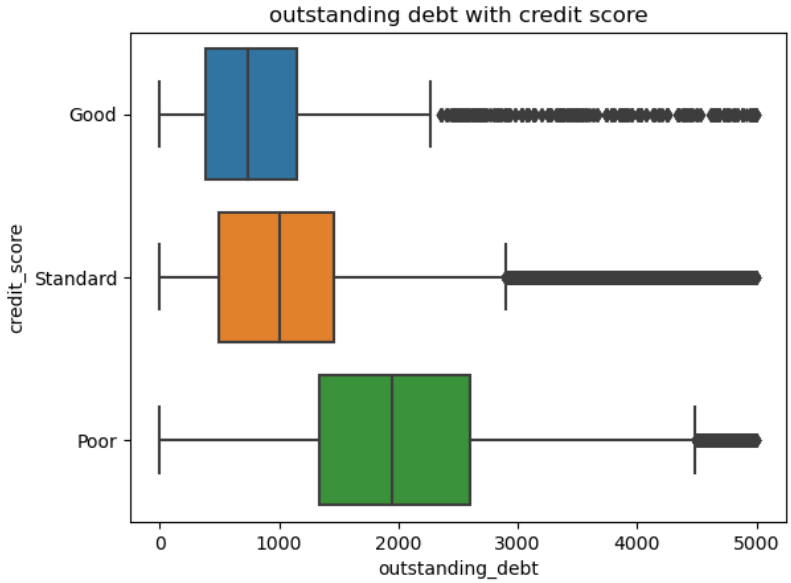
Payment of any credit amount should be in time or else it will result in bad credit score.

Number Of Times Delay Impacts Credit Score Or Not:



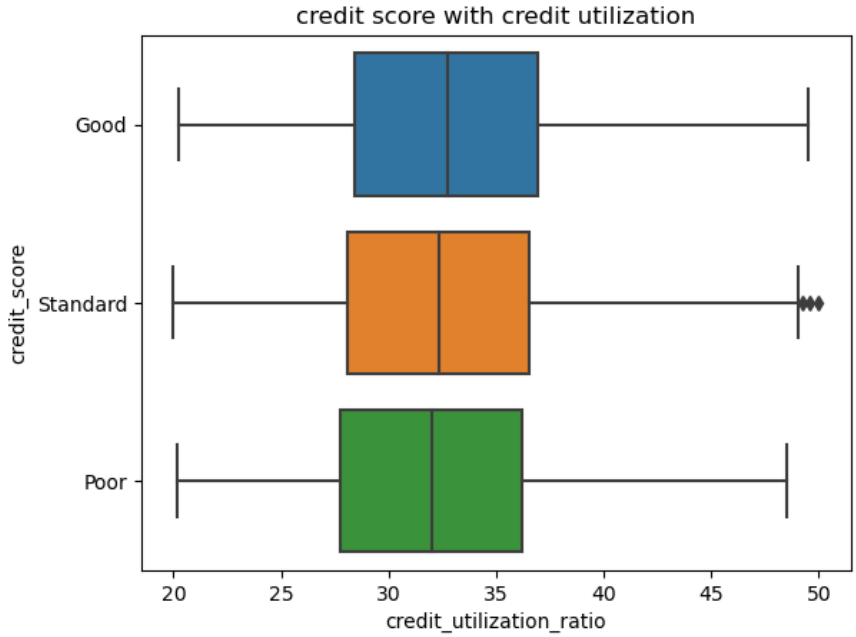
Times the payment is delayed is also not good for the score.

Outstanding Debt And Credit Score:



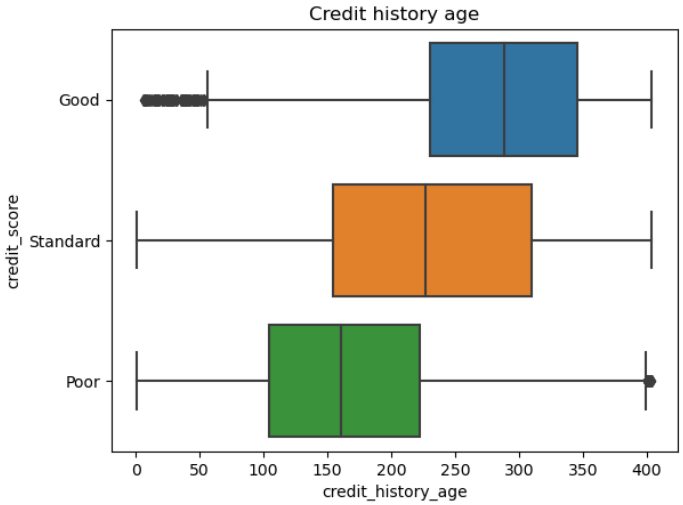
Having more outstanding debt negatively impact on credit score

Credit Utilization Ratio Impact Check:



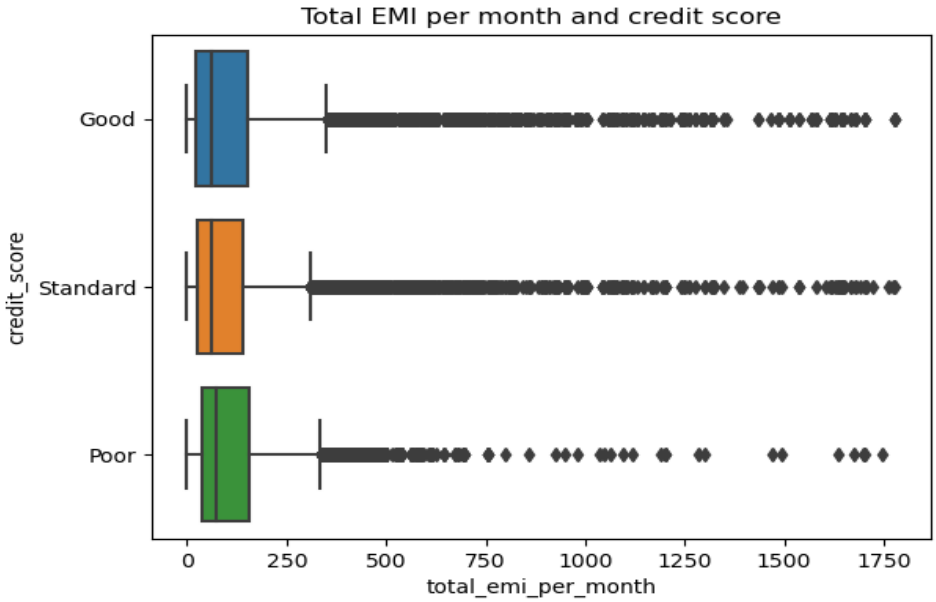
The above figure says that it does not affect the credit score and hence we drop this feature for model building.

Credit History Age Impacts The Credit Score Or Not:



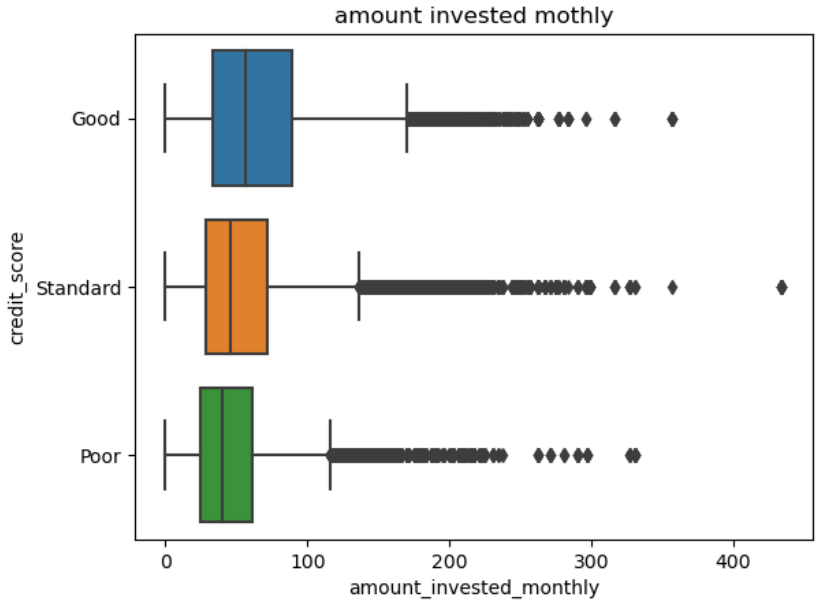
Credit history is important feature, hence this can be used for train the model

Total Emi Per Month Impact The Credit Score



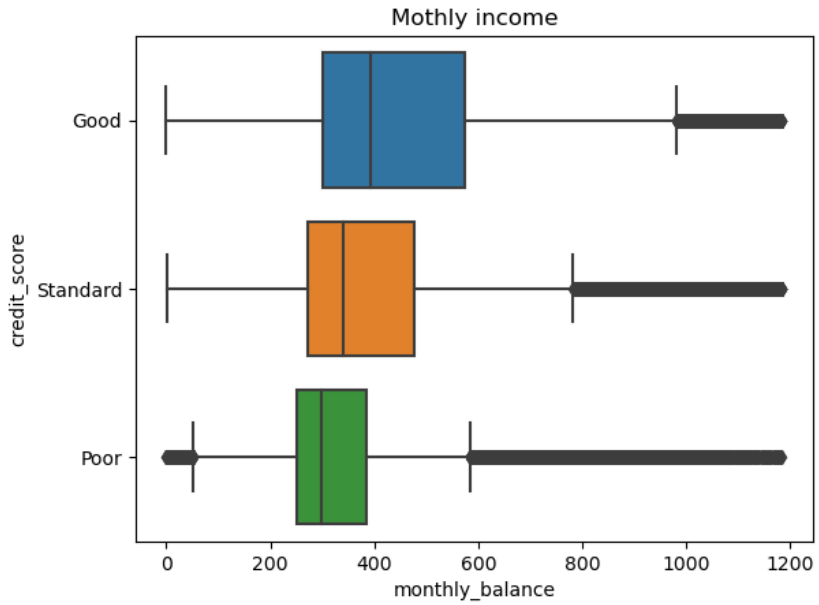
EMI is also not important for the model training.

Monthly Investment Result In Credit Score Decline Or Not:



It also does not have much impact on credit score hence we can drop this

Monthly Income Impact The Credit Score Or Not



Having a high monthly balance in your account at the end of the month is good for your credit scores

Machine -Learning Models Used:

1) **Random Forest Classifier**
2) **Logistic Regression**
3) **Naïve bayes**

Random Forest Classifier:

It is a supervised machine-learning algorithm used for classification, regression and other tasks. It is an ensemble method, which means it combines multiple decision trees to make a more accurate prediction. The basic idea of the algorithm is to create multiple decision trees for different subsets of data and then combine the predictions. Each decision tree is trained on a random sample of training data and a random subset of features. This prevents overfitting and produces more stable and accurate models. Steps to build Random Forest, classification model: 1. Randomly select 'n' features from the dataset. 2. Use these features to train a decision tree model. 3. Repeat steps 1 and 2 'm' times to create 'm' decision trees. 4. To classify a new data point, pass it through each of the 'm' decision trees, and tally the number of votes for each class. 5. The class with the most votes is the predicted class for the new data point. The dataset includes features such as income, credit score, age, and other relevant information. The goal is to build a model that can predict whether a new customer has a good or bad credit score.

Logistic Regression:

1. Input data: The first step in the Logistic Regression algorithm is to input the data. This data can be any set of observations or measurements with some associated outcome. For example, it could be a set of customer characteristics and their likelihood to purchase a particular product.
2. Model training: The next step is to train the model. This involves selecting a set of features (independent variables) from the input data that will be used to predict the outcome (dependent variable). The model is trained by estimating the parameters of the logistic function that best fit the data. These parameters are typically estimated using maximum likelihood estimation.
3. Prediction: Once the model has been trained, it can be used to predict the outcome of new data. The logistic function takes as input the values of the independent variables and produces a probability value between 0 and 1. This probability represents the likelihood of the outcome occurring given the values of the independent variables.
4. Evaluation: The final step is to evaluate the performance of the model. This is typically done by comparing the predicted outcomes to the actual outcomes in a test dataset. There are several metrics that can be used to evaluate the performance of a logistic regression model, including accuracy, precision, recall, and F1 score.

Naïve bayes:

1. Input data: The first step in the Naive Bayes algorithm is to input the data. This data can be any set of observations or measurements with some associated outcome. For example, it could be a set of emails with their corresponding labels as spam or non-spam.
2. Model training: The next step is to train the model. This involves building a probabilistic model of the input data, where each data point is represented as a set of features (independent variables) and a label (dependent variable). The model is trained by estimating the probability distributions of the features for each label. Naive Bayes assumes that the features are conditionally independent given the label, which is why it is called "naive".
3. Prediction: Once the model has been trained, it can be used to predict the label of new data. This is done by computing the posterior probability of each label given the values of the features using Bayes' rule. The label with the highest posterior probability is selected as the predicted label for the new data point.
4. Evaluation: The final step is to evaluate the performance of the model. This is typically done by comparing the predicted labels to the actual labels in a test dataset. There are several metrics that can be used to evaluate the performance of a Naive Bayes model, including accuracy, precision, recall, and F1 score.

Model building:

"Model was built according to the above pseudo code and the model selection was done on the basis of accuracy, precision, recall and f1 score."

Below are the code for model building and model report is given below:

Splitting the data into features and labels by selecting the features we found important for our model and also encoding the target variable.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
x = np.array(data[["annual_income", "monthly_inhand_salary",
                   "num_bank_accounts", "num_credit_card",
                   "interest_rate", "num_of_loan",
                   "delay_from_due_date", "num_of_delayed_payment",
                   "credit_mix", "outstanding_debt",
                   "credit_history_age", "monthly_balance"]])
le = LabelEncoder()
data['credit_score'] = le.fit_transform(data['credit_score'])
y = np.array(data[["credit_score"]])
```

Training the model:

1) **Random forest:**

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
xtrain, xtest, ytrain, ytest = train_test_split(x, y,test_size=0.2,random_state=
from sklearn.ensemble import RandomForestClassifier
model1 = RandomForestClassifier(n_estimators=100)
model1.fit(xtrain, ytrain)
y_pred1 = model1.predict(xtest)
```

2) **Logistic regression:**

LOGISTIC REGESSION CLASSIFICATION MODEL

```
|: from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder

# Create a logistic regression model
model2 = LogisticRegression(multi_class='multinomial', solver='lbfgs')

# Fit the model to the training data
model2.fit(xtrain, ytrain)

# Predict the classes of the testing set
y_pred2 = model2.predict(xtest)
```

3) **Naïve bayes:**

Naive bayes model

```
1]: from sklearn.naive_bayes import GaussianNB



# Create a Gaussian Naive Bayes model
model3 = GaussianNB()

# Fit the model to the training data
model3.fit(xtrain, ytrain)

# Predict the classes of the testing set
y_pred3 = model3.predict(xtest)
```

Model selection :

"Random Forest classification"

Random forest classifcation report

	precision	recall	f1-score	support
0	0.78	0.78	0.78	3585
1	0.80	0.84	0.82	5780
2	0.83	0.81	0.82	10635
accuracy			0.81	20000
macro avg	0.80	0.81	0.81	20000
weighted avg	0.81	0.81	0.81	20000

"Logistic regression report:"

Logistic regression classification report

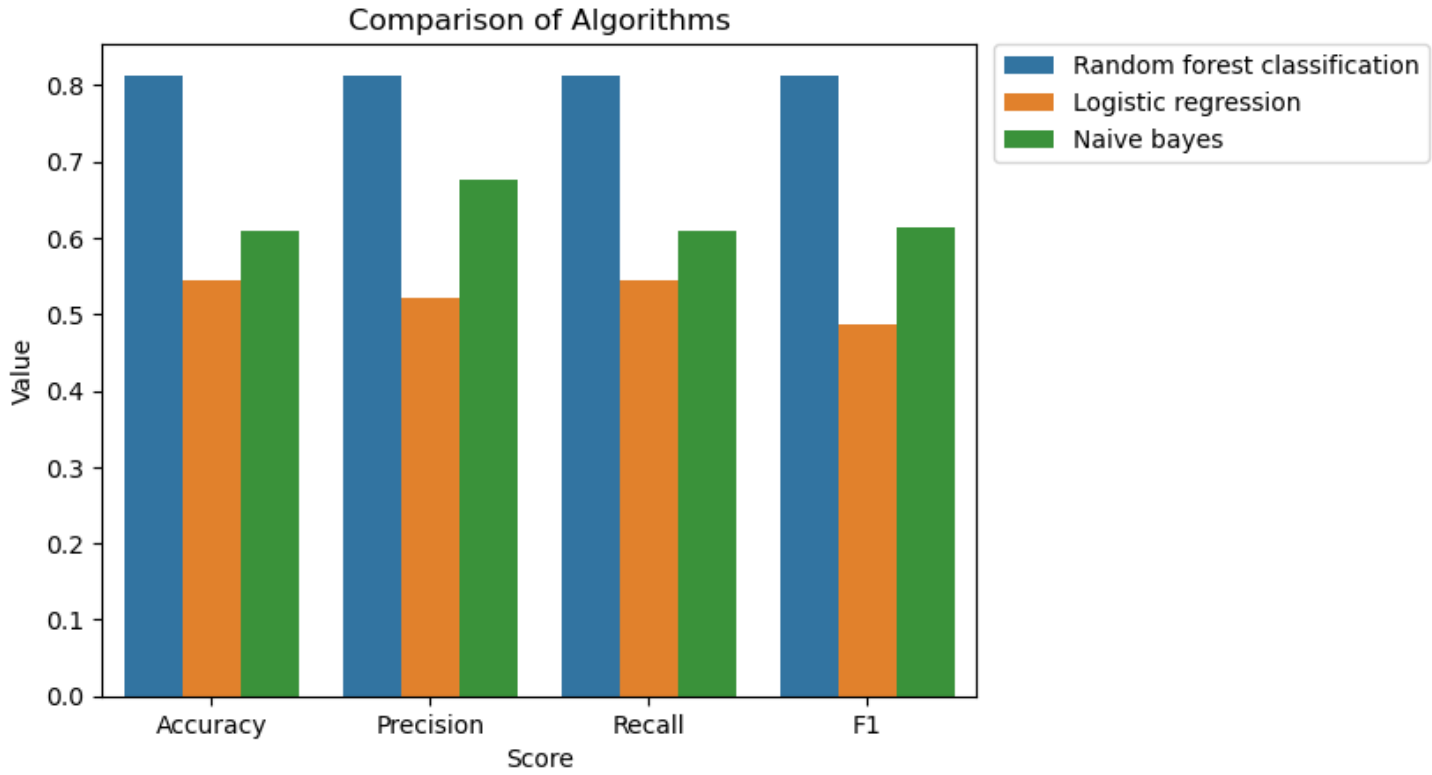
	precision	recall	f1-score	support
Good	0.55	0.12	0.19	3585
Poor	0.61	0.51	0.56	5780
Standard	0.59	0.80	0.68	10635
accuracy			0.60	20000
macro avg	0.59	0.48	0.48	20000
weighted avg	0.59	0.60	0.56	20000

"Naïve Bayes report:"

Naive bayes classification report

	precision	recall	f1-score	support
Good	0.42	0.80	0.55	3585
Poor	0.64	0.71	0.67	5780
Standard	0.78	0.49	0.61	10635
accuracy			0.61	20000
macro avg	0.61	0.67	0.61	20000
weighted avg	0.68	0.61	0.61	20000

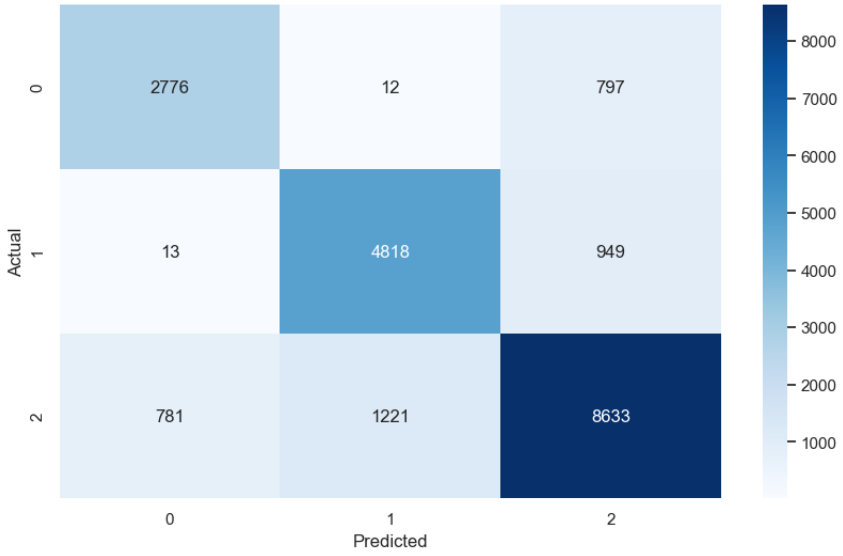
Visualising the score in bar plot:



Overall, based on this classification report, we can conclude that the Random Forest model performed well on this particular problem, with good accuracy .

Confusion matrix evaluation:

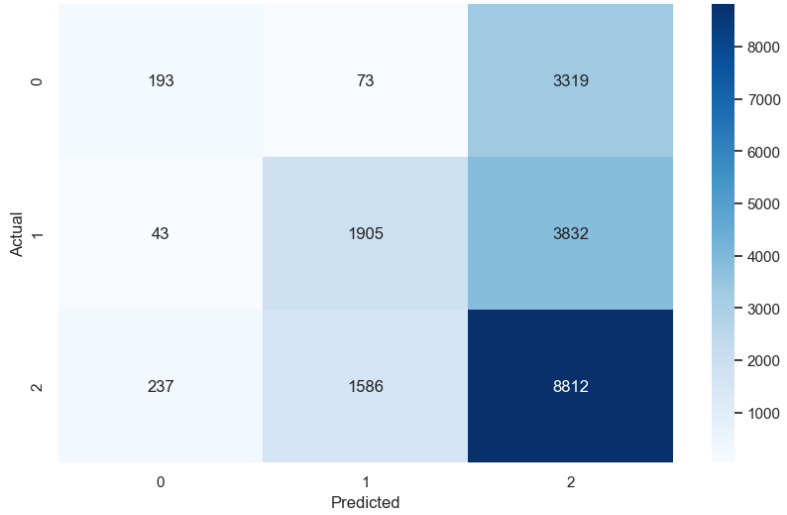
“Random forest:”



The first row indicates the samples that are actually labeled as "0" in the data. Out of 3585 samples, the model predicted 2776 correctly as "0" (true negatives), misclassified 797 as "2" (false negatives), and misclassified 12 as "1" (false positives).

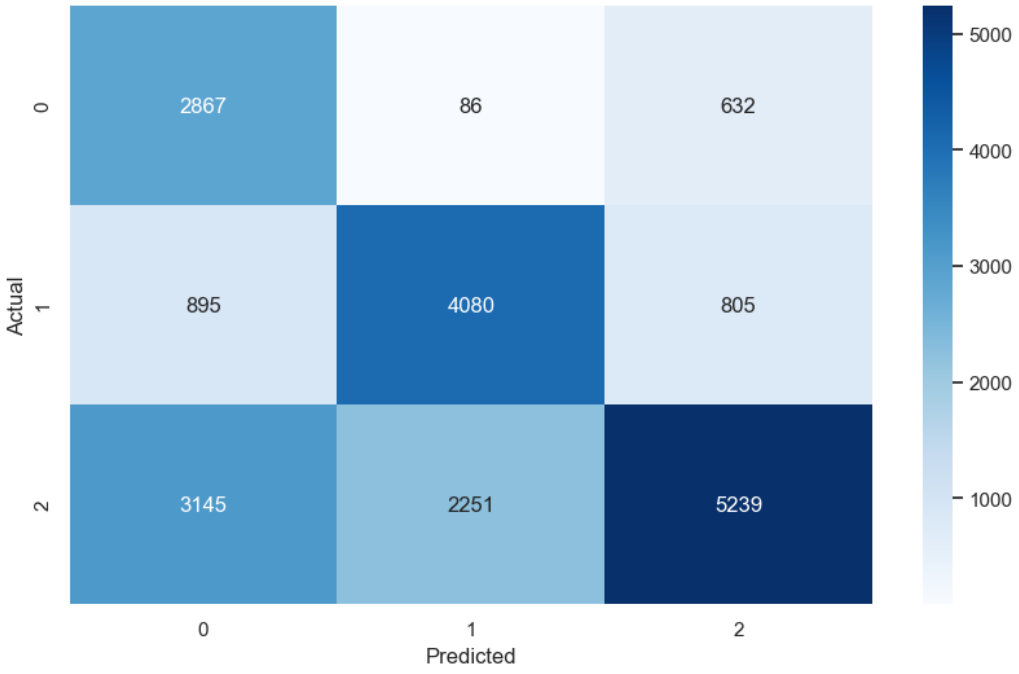
- The second row indicates the samples that are actually labeled as "1" in the data. Out of 5780 samples, the model predicted 4818 correctly as "1" (true positives), misclassified 949 as "2" (false negatives), and misclassified 13 as "0" (false positives).
- The third row indicates the samples that are actually labeled as "2" in the data. Out of 10635 samples, the model predicted 8633 correctly as "2" (true negatives), misclassified 781 as "0" (false positives), and misclassified 1221 as "1" (false positives).

“Logistic Regression:”



- The first row indicates the samples that are actually labeled as "0"in the data. Out of 3585 samples, the model predicted 193 correctly as "0" (true positives), misclassified 73 as "1" (false negatives), and misclassified 3319 as "2" (false negatives).
- The second row indicates the samples that are actually labeled as "1" the data. Out of 5780 samples, the model predicted 1905 correctly as "1" (true positives), misclassified 43 as "0" (false positives), and misclassified 3832 as "2" (false negatives).
- The third row indicates the samples that are actually labeled as "2" in the data. Out of 10435 samples, the model predicted 8812 correctly as "2" (true positives), misclassified 237 as "0" (false positives), and misclassified 1586 as "1" (false positives).

“Naïve bayes:”



- The first row indicates the samples that are actually labeled as "0" in the data. Out of 3585 samples, the model predicted 2867 correctly as "0" (true positives), misclassified 86 as "1" (false negatives), and misclassified 632 as "2" (false negatives).
- The second row indicates the samples that are actually labeled as "1" in the data. Out of 5780 samples, the model predicted 4080 correctly as "1" (true positives), misclassified 895 as "0" (false positives), and misclassified 805 as "2" (false negatives).
- The third row indicates the samples that are actually labeled as "2" in the data. Out of 10635 samples, the model predicted 5239 correctly as "2" (true positives), misclassified 3145 as "0" (false positives), and misclassified 2251 as "1" (false positives).

Prediction of new customers Credit score Using Random Forest

let's make predictions from our model by giving inputs to our model according to the features we used to train the model:

```
9]: print("Credit Score Prediction : ")
    a = float(input("Annual Income: "))
    b = float(input("Monthly Inhand Salary: "))
    c = float(input("Number of Bank Accounts: "))
    d = float(input("Number of Credit cards: "))
    e = float(input("Interest rate: "))
    f = float(input("Number of Loans: "))
    g = float(input("Average number of days delayed by the person: "))
    h = float(input("Number of delayed payments: "))
    i = input("Credit Mix (Bad: 0, Standard: 1, Good: 3) : ")
    j = float(input("Outstanding Debt: "))
    k = float(input("Credit History Age: "))
    l = float(input("Monthly Balance: "))

    features = np.array([[a, b, c, d, e, f, g, h, i, j, k, l]])
    print("Predicted Credit Score = ", model1.predict(features))
```

Output:

```
Credit Score Prediction :
Annual Income: 100000
Monthly Inhand Salary: 3000
Number of Bank Accounts: 3
Number of Credit cards: 3
Interest rate: 3
Number of Loans: 3
Average number of days delayed by the person: 0
Number of delayed payments: 0
Credit Mix (Bad: 0, Standard: 1, Good: 3) : 3
Outstanding Debt: 0
Credit History Age: 3
Monthly Balance: 100000
Predicted Credit Score =  [2]
```

Predicted score is 2 that means the credit score for this customer is standard

Conclusion :

Based on the evaluation metrics and visualizations of the three classification models “(Random Forest, Logistic Regression, and Naive Bayes)”, we can conclude that Random Forest performed the best on the credit score classification problem. It had the highest accuracy score of 0.81, along with high precision, recall, and F1 scores for all three classes.

Logistic Regression and Naive Bayes had lower accuracy scores compared to Random Forest. Logistic Regression had the most insufficient precision and recall scores for the "Good" class, while Naive Bayes had the lowest precision and F1 scores for the "Standard" class.
In summary, Random Forest appears to be the best model for this particular classification problem. However, it's worth noting that further improvements could be made to the model by tuning hyperparameters or trying different feature engineering techniques. Additionally, it may be beneficial to consider other classification models or ensemble methods to further improve the performance.

References:

- "https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/"
- "https://www.linkedin.com/pulse/exploring-scikit-learn-10-examples-leonardo-anello"
- "Machine Learning Algorithm In Sorting?? [With Code!!] » EML."
- "https://enjoymachinelearning.com/blog/machine-learning-algorithm-in-sorting/"
- "www.kaggle.com"