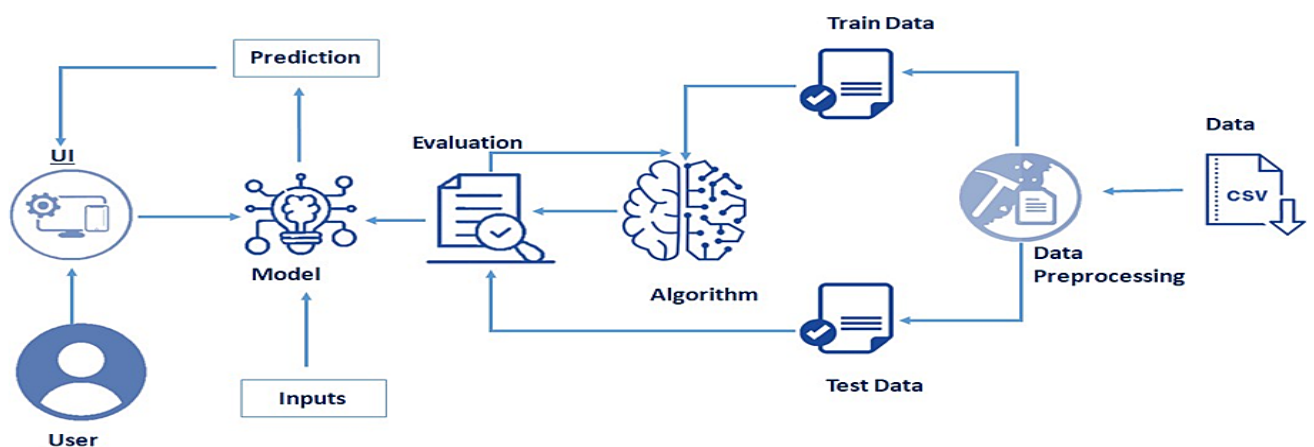


Credit Card Approval Prediction using Machine Learning

Project Description:

Nowadays, banks receive a lot of applications for issuance of credit cards. Many of them are rejected for many reasons, like high-loan balances, low-income levels, or too many inquiries on an individual's credit report. Manually analyzing these applications is error-prone and a time-consuming process. Luckily, this task can be automated with the power of machine learning and pretty much every bank does so nowadays. In this project, we will build an automatic credit card approval predictor using machine learning techniques, just like the real banks do. In this project, we will be using regression algorithms such as LogisticRegression, SGD, SVCClassifier, decisiontree, RandomForest and xgboost. We will train and test the data with these algorithms. From this the best model is selected and saved in pkl format. We will be doing flask integration and flask deployment.

Technical Architecture:

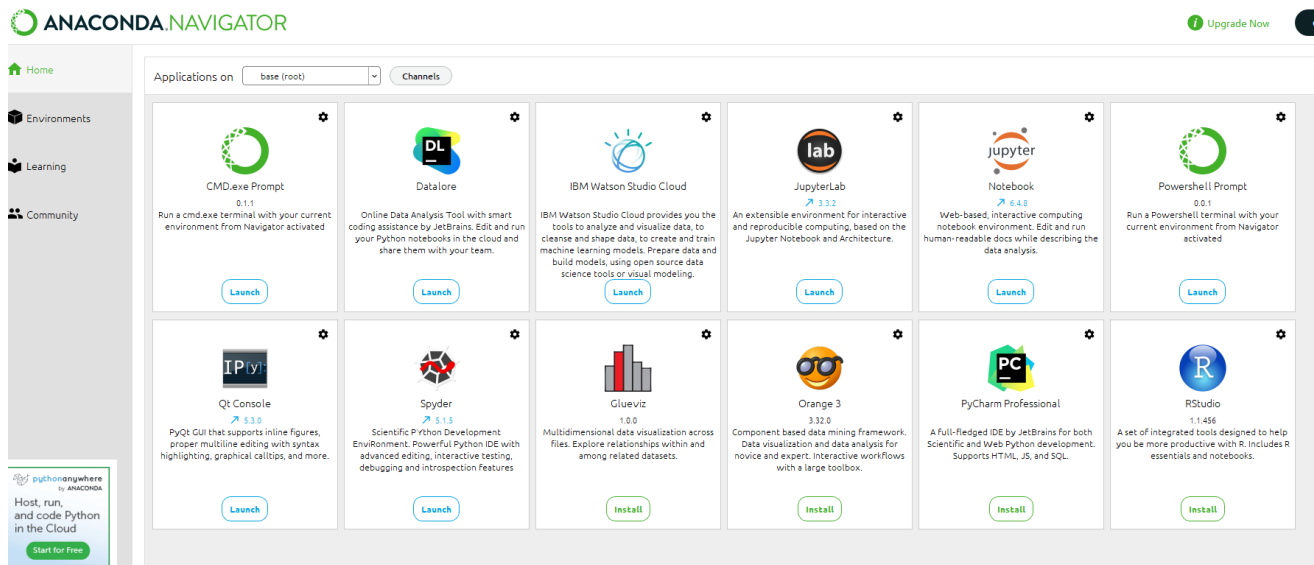


Pre requisites:

To complete this project, you must require following software's, concepts and packages

1. Anaconda navigator:

- Refer the link below to download anaconda navigator
- Link : <https://youtu.be/1ra4zH2G4o0>



2. Python packages:

- Open anaconda prompt as administrator
- Type "pip install numpy" and click enter.
- Type "pip install pandas" and click enter.
- Type "pip install pickle-mixin" and click enter
- Type "pip install Flask" and click enter.

Project Objectives :

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/capping techniques on outlier and some visualization concepts.

Gain some ideas on algorithm selection.

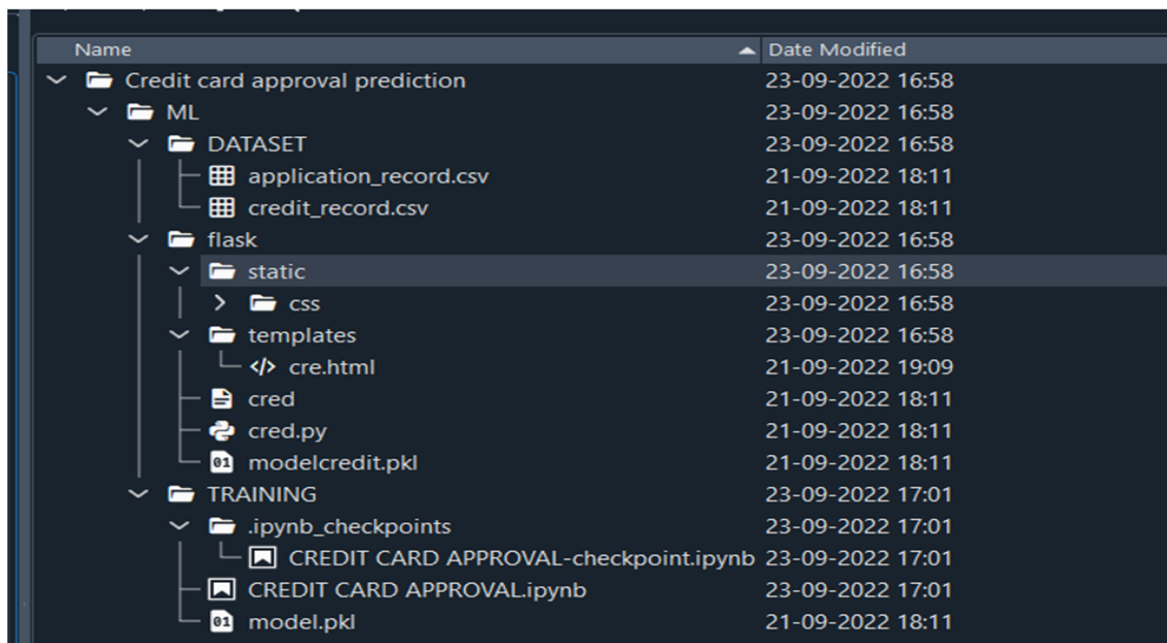
Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI To accomplish this, we have to complete all the activities listed below,
- Data Collection.
 - Collect the dataset or Create the dataset
- Data Visualization
- Multivariate analysis
- Descriptive analysis
- Data Pre-processing
 - Checking for null values
 - Drop unwanted features

- Data Cleaning and merging
- Handling categorical data
- Splitting Data into Train and Test.
- Model Building
 - Import the model building Libraries
 - Initializing the model
 - Training and testing the model
 - Evaluation of Model
 - Save the Model
- Application Building
 - Create an HTML file
 - Build a Python Code

Project Structure:

Create a Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script cred.py for scripting. modelcredit.pkl is our saved model. Further we will use this model for flask integration

Milestone 1: Data Collection:-

ML depends heavily on data, it is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

Please refer to the link given below to download the data set and to know about the dataset

<https://www.kaggle.com/namphuengauawatcharo/credit-card-approval-prediction/data>

Milestone 2: Visualizing and analysing the data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysing techniques.

Note: There is a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image.

To know about the packages refer the link given on pre requisites.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Activity 1: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

Activity 2: Handling Missing Values

- In pandas we have a function called `read_csv()` to read the dataset. As a parameter we have to give the directory of csv file.

```
dt = pd.read_csv(r"C:\Users\jagad\Dropbox\PC\Downloads\DATASET\application_record.csv")
df = pd.read_csv(r"C:\Users\jagad\Dropbox\PC\Downloads\DATASET\credit_record.csv")
```

- `head()` method is used to return top n (5 by default) rows of a DataFrame or series.

```
dt.head()
```

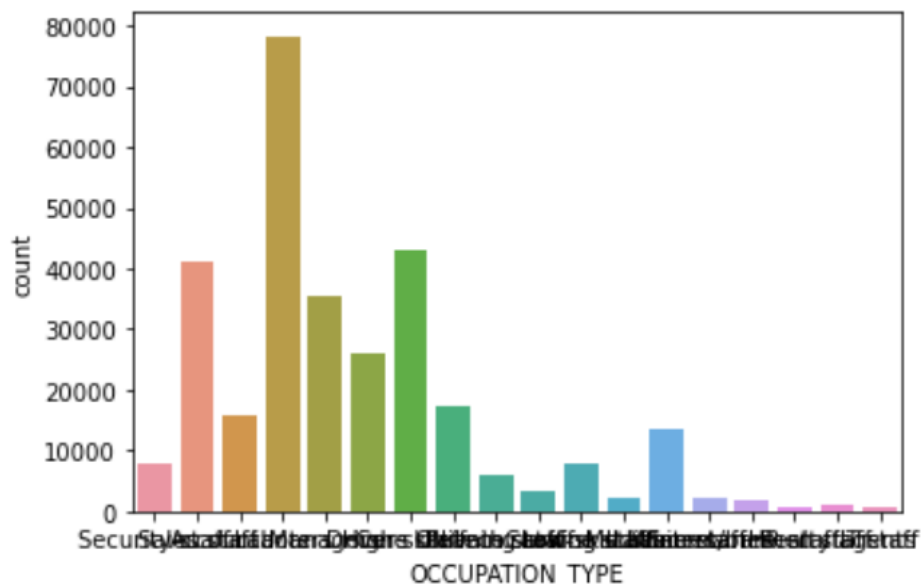
	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE
0	5008804	M	Y	Y	0	427500.0	Working
1	5008805	M	Y	Y	0	427500.0	Working
2	5008806	M	Y	Y	0	112500.0	Working
3	5008808	F	N	Y	0	270000.0	Commercial associate
4	5008809	F	N	Y	0	270000.0	Commercial associate

Activity 3: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature.

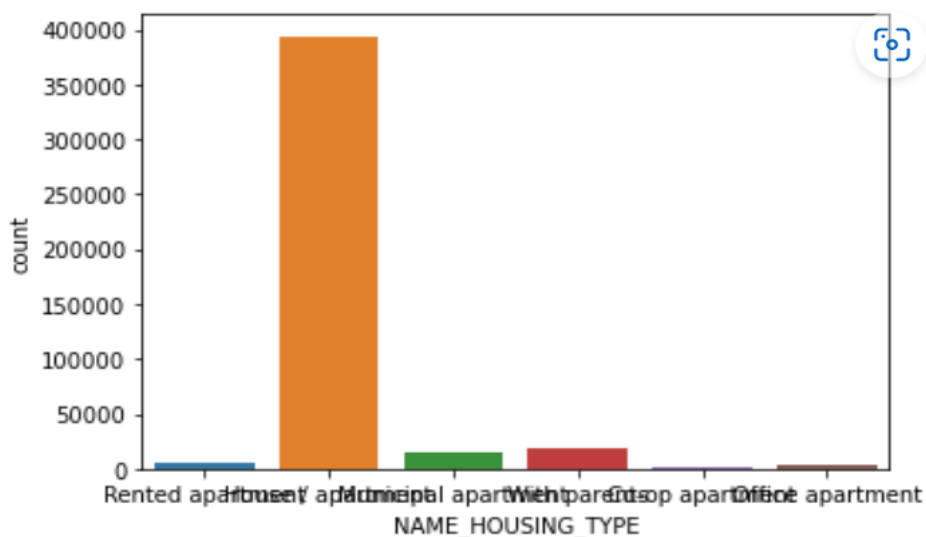
Count plot is used on occupation type feature. With the `countplot()`, we are going to count the unique category. From the below graph, we found the number of labors are high when compared to other types. For the exact count, `value counts()` are used.

```
: <AxesSubplot:xlabel='OCCUPATION_TYPE', ylabel='count'>
```



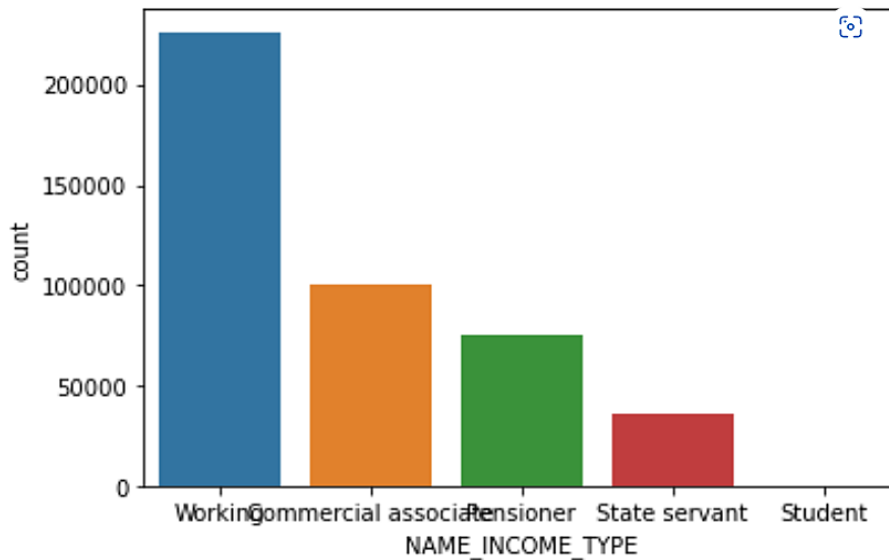
- Count plot is used on income type feature. With the `countplot()`, we are going to count the unique category. From the below graph, we found the number of working applicant are high when compared to other types. For the exact count, `value counts()` are used.

```
<AxesSubplot:xlabel='NAME_HOUSING_TYPE', ylabel='count'>
```



Count plot is used on income type feature. With the `countplot()`, we are going to count the unique category. From the below graph, we found the number of working applicant are high when compared to other types. For the exact count, value `counts()` are used.

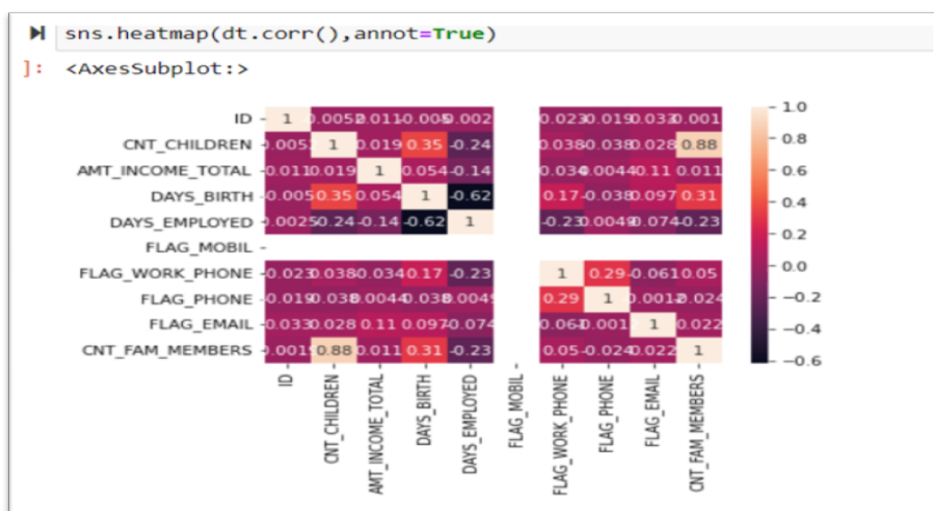
```
<AxesSubplot:xlabel='NAME_INCOME_TYPE', ylabel='count'>
```



Activity 4: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap() from seaborn package.

- To visualize the correlation of the features, heatmap() function is used. As a parameter corr() function should be passed inside heatmap. And to display the correlation percentage annot() function is used.



Activity 5: Descriptive analysis

Descriptive function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features. analysis is to study the basic features of data with the statistical process.

```
dt.describe()
```

	ID	CNT_CHILDREN	AMT_INCOME_TOTAL	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE
count	4.385570e+05	438557.000000	4.385570e+05	438557.000000	438557.000000	438557.0	438557.000000	438557.000000
mean	6.022176e+06	0.427390	1.875243e+05	-15997.904649	60563.675328	1.0	0.206133	0.287771
std	5.716370e+05	0.724882	1.100869e+05	4185.030007	138767.799647	0.0	0.404527	0.452724
min	5.008804e+06	0.000000	2.610000e+04	-25201.000000	-17531.000000	1.0	0.000000	0.000000
25%	5.609375e+06	0.000000	1.215000e+05	-19483.000000	-3103.000000	1.0	0.000000	0.000000
50%	6.047745e+06	0.000000	1.607805e+05	-15630.000000	-1467.000000	1.0	0.000000	0.000000
75%	6.456971e+06	1.000000	2.250000e+05	-12514.000000	-371.000000	1.0	0.000000	1.000000
max	7.999952e+06	19.000000	6.750000e+06	-7489.000000	365243.000000	1.0	1.000000	1.000000

Milestone 3: Data Pre-processing

As we have understood how the data is. Let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results.

This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

- To find the data type of columns `info()` function is used. It gives small information about the features.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   ID                    1048575 non-null  int64
1   MONTHS_BALANCE       1048575 non-null  int64
2   STATUS               1048575 non-null  object
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

- `Unique()` method is used to find the unique values of features. A function is defined below to find the unique values of features.

Activity 1: Drop duplicate features

Generally, applicant IDs are unique in nature. But in our dataset we found some of the IDs are repeating multiple times. To handle this we have to remove the duplicate rows. `Drop duplicates()` function from pandas is used to remove the duplicate rows. Refer the below diagram.

```
# Dropping duplicate rows
dt.drop_duplicates(subset = ['CODE_GENDER', 'FLAG_OWN_CAR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE',
                           'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
                           'FLAG_MOBIL', 'FLAG_WORK_PHONE', 'FLAG_PHONE', 'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS'],
                  inplace=True)
```

For checking the null values, `df.isnull()` function is used. To sum those null values we use `sum()` function to it. `mean()` function is used to find the impact of null values in features. From the below image we found, our dataset has no null values


```
dt.isnull().sum()
```

```
] : ID                0
    CODE_GENDER       0
    FLAG_OWN_CAR       0
    FLAG_OWN_REALTY    0
    CNT_CHILDREN       0
    AMT_INCOME_TOTAL   0
    NAME_INCOME_TYPE    0
    NAME_EDUCATION_TYPE 0
    NAME_FAMILY_STATUS  0
    NAME_HOUSING_TYPE   0
    DAYS_BIRTH         0
    DAYS_EMPLOYED       0
    FLAG_MOBIL         0
    FLAG_WORK_PHONE     0
    FLAG_PHONE         0
    FLAG_EMAIL         0
    OCCUPATION_TYPE     27383
    CNT_FAM_MEMBERS     0
    dtype: int64
```

Activity 3: Data Cleaning and merging

In this process, we are going to combine two inter-related columns. Our dataset has some negative values. Those negative values are converted into absolute values. Feature mapping is used on some categorical columns.

- values to absolute values we use `abs()` function.
- Feature mapping is done in housing type, income type, education type and A function `data_cleaning()` is defined. A column is created by adding number of family members with number of childrens.
- Six unwanted columns are dropped by `drop()` function. Refer the below image to know the columns name.
- Days birth and days employed columns have negative values. To convert the negative family type columns. (This feature mapping step is an optional step).

```
# Adding number of family members with number of children to get overall family members. data[ 'CNT_FAM_MEMBERS']
def data_clean(data):
    data['CNT_FAM_MEMBERS'] = data['CNT_FAM_MEMBERS'] + data['CNT_CHILDREN']
    dropped_cols = ['FLAG_MOBIL', 'FLAG_WORK_PHONE', 'FLAG_PHONE', 'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_CHILDREN']
    data = data.drop(dropped_cols, axis = 1)
    #converting birth years and days employed to years. data[ 'DAYS_BIRTH'] = np.abs(data[ 'DAYS_BIRTH'])/365) #Absolute
    #Cleaning up categorical values to lower the count of dummy variables. housing_type = { 'House / apartment' : 'House / apartment', 'With parents': 'With parents', 'Municipal apartment': 'Municipal apartment'
    housing_type = { 'House / apartment': 'House / apartment', 'With parents': 'With parents', 'Municipal apartment': 'Municipal apartment', 'Municipal apartment': 'Municipal apartment'
    income_type = { 'Commercial associate': 'Working', 'State servant': 'Working', 'Working': 'Working', 'Pensioner': 'Pensioner'
    education_type = { 'Secondary / secondary special': 'secondary', 'Lower secondary': 'secondary', 'Higher education': 'Higher education'
    family_status = { 'Single / not married': 'Single', 'Separated': 'Single', 'Widow': 'Single', 'Civil marriage': 'Civil marriage'
    data['NAME_HOUSING_TYPE'] = data['NAME_HOUSING_TYPE'].map(housing_type)
    data['NAME_INCOME_TYPE'] = data['NAME_INCOME_TYPE'].map(income_type)
    data['NAME_EDUCATION_TYPE'] = data['NAME_EDUCATION_TYPE'].map(education_type)
    data['NAME_FAMILY_STATUS'] = data['NAME_FAMILY_STATUS'].map(family_status)
    return data
```

Let's move to our second dataframe (cr).

To display the first five columns `head()` function is used. The `info()` method is used to find the data types of the columns.


```
df.describe()
```

	ID	MONTHS_BALANCE
count	1.048575e+06	1.048575e+06
mean	5.068286e+06	-1.913700e+01
std	4.615058e+04	1.402350e+01
min	5.001711e+06	-6.000000e+01
25%	5.023644e+06	-2.900000e+01
50%	5.062104e+06	-1.700000e+01
75%	5.113856e+06	-7.000000e+00
max	5.150487e+06	0.000000e+00

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
#   Column          Non-Null Count  Dtype
---  -
0   ID              1048575 non-null  int64
1   MONTHS_BALANCE  1048575 non-null  int64
2   STATUS          1048575 non-null  object
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

We are grouping the ID column and saving it as a variable 'grouped'.

We are using as an index ID and for column we are using MONTHS_BALANCE and STATUS as a value.

1. Minimum MONTHS_BALANCE as a open_month
2. Maximum MONTHS_BALANCE as a end_months

And for window we are subtracting end_months – open_months

```
# for credit dataset
g = df.groupby('ID')
p = df.pivot(index = 'ID', columns = 'MONTHS_BALANCE', values = 'STATUS')
p['OPEN_MONTH'] = g['MONTHS_BALANCE'].min() #0
p['END_MONTH'] = g['MONTHS_BALANCE'].max() #0
p['WINDOW'] = p['END_MONTH'] - p['OPEN_MONTH']
p['WINDOW'] = p['WINDOW'] + 1 # ADDING BECAUSE MONTH STARTS AT 0
```

```
#counting number of past dues, paid offs and no loans
p.shape
```

```
]: (45985, 64)
```

```
p['paid_off'] = p[p.iloc[:,0:61] == 'C'].count(axis=1)
p['pastdue_1-29'] = p[p.iloc[:,0:61] == '0'].count(axis=1)
p['pastdue_30-59'] = p[p.iloc[:,0:61] == '1'].count(axis=1)
p['pastdue_60-89'] = p[p.iloc[:,0:61] == '2'].count(axis=1)
p['pastdue_90-119'] = p[p.iloc[:,0:61] == '3'].count(axis=1)
p['pastdue_120-149'] = p[p.iloc[:,0:61] == '4'].count(axis=1)
p['pastdue_over_150'] = p[p.iloc[:,0:61] == '5'].count(axis=1)
p['no_loan'] = p[p.iloc[:,0:61] == 'X'].count(axis=1)
p['ID'] = p.index
```

Activity 4: Feature Engineering

Converting the multi-classification into binary classification. For a clear understanding refer the below two images.

A ratio is based method was used to create the target variable. for eg, given a client with a time period of 60 months, if the client given that there were more loans that were paid off on time compared to late payments. if a client had no loans throughout the initial approval of the credit card account. for simplicity sake I will not adjust the algorithm further and keep it at ratio decisioning. code is also not optimal, adjustment may be needed for the code to compute faster.

```

def feature_engineering_target(data):
    good_or_bad = []
    for index, row in data.iterrows():
        paid_off = row['paid_off']
        over_1 = row['pastdue_1-29']
        over_30 = row['pastdue_30-59']
        over_60 = row['pastdue_60-89']
        over_90 = row['pastdue_90-119']
        over_120 = row['pastdue_120-149'] + row['pastdue_over_150']
        no_loan = row['no_loan']
        overall_pastdues = over_1+over_30+over_60+over_90+over_120
        if overall_pastdues == 0:
            if paid_off >= no_loan or paid_off <= no_loan:
                good_or_bad.append(1)
            elif paid_off == 0 and no_loan == 1:
                good_or_bad.append(1)
        elif overall_pastdues != 0 :
            if paid_off > overall_pastdues :
                good_or_bad.append(1)
            elif paid_off <= overall_pastdues :
                good_or_bad.append(0)
        elif paid_off == 0 and no_loan != 0 :
            if overall_pastdues <= no_loan or overall_pastdues >= no_loan:
                good_or_bad.append(0)
        else:
            good_or_bad.append(1)
    return good_or_bad

```

Converting our credit data into binary format because at last we need to predict whether a person is eligible for credit card or not?

Merging two data frames with merge() function.

```

target = pd.DataFrame()
target['ID'] = p.index
target['paid_off'] = p['paid_off'].values
target['#_of_pastdues'] = p['pastdue_1-29'].values + p['pastdue_30-59'].values
+ p['pastdue_60-89'].values + p['pastdue_90-119'].values
+ p['pastdue_120-149'].values + p['pastdue_over_150'].values
target['no_loan'] = p['no_loan'].values
target['target'] = feature_engineering_target(p)
m = dt.merge(target, how = 'inner', on = 'ID')
m.drop('ID', axis = 1, inplace = True)

```

m

5]:

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	M	Y	Y	427500.0	Working	Higher education
1	M	Y	Y	112500.0	Working	secondary
2	F	N	Y	270000.0	Working	secondary
3	F	N	Y	283500.0	Pensioner	Higher education
4	M	Y	Y	270000.0	Working	Higher education
...
9692	F	N	N	180000.0	Pensioner	secondary
9693	F	N	Y	112500.0	Working	secondary
9694	M	Y	Y	90000.0	Working	secondary
9695	F	N	Y	157500.0	Pensioner	Higher education
9696	M	N	Y	112500.0	Working	secondary

9697 rows × 7 columns

Activity 5: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding technique.

There are several techniques but in our project we are using label encoding.

Label encoder is initialized and categorical feature is passed as parameter for fit_transform()

function. Label encoding uses alphabetical ordering. For the feature names refer the below diagram

```

#handling categorical value
from sklearn.preprocessing import LabelEncoder
a = LabelEncoder()
m['CODE_GENDER'] = a.fit_transform(m['CODE_GENDER'])
m['FLAG_OWN_CAR'] = a.fit_transform(m['FLAG_OWN_CAR'])
m['FLAG_OWN_REALTY'] = a.fit_transform(m['FLAG_OWN_REALTY'])
m['NAME_INCOME_TYPE'] = a.fit_transform(m['NAME_INCOME_TYPE'])
m['NAME_EDUCATION_TYPE'] = a.fit_transform(m['NAME_EDUCATION_TYPE'])
m['NAME_FAMILY_STATUS'] = a.fit_transform(m['NAME_FAMILY_STATUS'])
m['NAME_HOUSING_TYPE'] = a.fit_transform(m['NAME_HOUSING_TYPE'])

```

Activity 6: Splitting data into train and test

Now let's split the Dataset into train and test sets. For splitting training and testing data we are using the `train_test_split()` function from `sklearn`. As parameters, we are passing `x`, `y`, `train_size`, `random_state`. `X` is independent variable and `y` is dependent variable.

```
#splitting our data
from sklearn.model_selection import train_test_split
x = m[m.drop('target',axis=1).columns]
y = m['target']
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.75,random_state=0)
```

Milestone 4: Model Building:-

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance. To evaluate the performance confusion matrix and classification report is used.

Activity 1: Logistic regression model

A function named `logistic_reg` is created and train and test data are passed as the parameters. Inside the function, `LogisticRegression()` algorithm is initialized and training data is passed to the model with `.fit()` function. Test data is predicted with `.predict()` function and saved in new variable. For evaluating the model, confusion matrix and classification report is done. Refer the below image.

Activity 2: SCD model

A slowly changing dimension in data management and data warehousing is a dimension which contains relatively static data which can change slowly but unpredictably, rather than according to a regular schedule.

Activity 3: SVClassifier model

In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

Activity 4: DecisionTree model

A function named `d_tree` is created and train and test data are passed as the parameters. Inside the function, `DecisionTreeClassifier()` algorithm is initialized and training data is passed to the model with `.fit()` function. Test data is predicted with `.predict()` function and saved in new variable. For evaluating the model, confusion matrix and classification report is done. Refer the below image.

Activity 5: RandomForest model

A function named `random_forest` is created and train and test data are passed as the parameters.

Inside the function, `RandomForestClassifier()` algorithm is initialized and training data is passed to the model with `.fit()` function. Test data is predicted with `.predict()` function and saved in new variable. For evaluating the model, confusion matrix and classification report is done. Refer the below image.

Activity 6: xgboost model

A function named `g_boosting` is created and train and test data are passed as the parameters. Inside the function, `GradientBoostingClassifier()` algorithm is initialized and training data is passed to the model with `.fit()` function. Test data is predicted with `.predict()` function and saved in new variable. For evaluating the model, confusion matrix and classification report is done. Refer the below image.

```

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

def logisticRegression(x_train, x_test, y_train, y_test):
    # creating the model
    model = LogisticRegression()
    # feeding the training set into the model
    model.fit(x_train, y_train)
    # predicting the results for the test set
    y_pred = model.predict(x_test)
    # calculating the training and testing accuracies
    print('***logisticRegression***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))

```

```

from sklearn.linear_model import SGDClassifier
def SGD(x_train, x_test, y_train, y_test):
    # creating the model
    model = SGDClassifier(penalty=None)
    # feeding the training model into the model
    model.fit(x_train, y_train)
    # predicting the values for the test set
    y_pred = model.predict(x_test)
    print('***Stochastic Gradient Descent Classifier***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))

```

```

from sklearn.svm import SVC
def SVC(x_train, x_test, y_train, y_test):
    # creating the model
    model = SVC()
    # feeding the training set into the model
    model.fit(x_train, y_train)
    # predicting the results for the test set
    y_pred = model.predict(x_test)
    # calculating the training and testing accuracies
    print('***Support Vector Classifier***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))

```

```

from sklearn.tree import DecisionTreeClassifier
def decisionTree(x_train, x_test, y_train, y_test):
    dt = DecisionTreeClassifier()
    dt.fit(x_train, y_train)
    y_pred = dt.predict(x_test)
    print('***DecisionTreeClassifier***')
    print("Training accuracy :", dt.score(x_train, y_train))
    print("Testing accuracy :", dt.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test, y_pred))
    print('Classification report')
    print(classification_report(y_test, y_pred))

```

```

from sklearn.ensemble import RandomForestClassifier
def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train, y_train)
    y_pred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print("Training accuracy :", rf.score(x_train, y_train))
    print("Testing accuracy :", rf.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test, y_pred))
    print('Classification report')
    print(classification_report(y_test, y_pred))

```

```

from sklearn.ensemble import GradientBoostingClassifier
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train, y_train)
    y_pred = xg.predict(x_test)
    print('***GradientBoostingClassifier***')
    print("Training accuracy :", xg.score(x_train, y_train))
    print("Testing accuracy :", xg.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test, y_pred))
    print('Classification report')
    print(classification_report(y_test, y_pred))

```

Activity 5: Save the model

Decision tree model is saved by pickle.dump() function. It saves the model as .pkl file.

```

dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
y_pred = dt.predict(x_test)

```

```

import pickle
pickle.dump(dt, open("model.pkl", "wb"))

```

Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the users where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

1. Building HTML Pages
2. Building server-side script

Activity 1: Building HTML Pages:

In our project we have created ONE HTML page, That is **cred.html**

```

<!DOCTYPE html>
<html>
<head>
  <title> Analysis</title>
  <!-- Latest compiled and minified CSS -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <link href="C:\Users\jagad\OneDrive\Desktop\Credit card approval prediction\Credit card approval prediction\ML\flask\st
  <style type="text/css">
    .result{
      color:black;
      margin-top:30px;
      margin-bottom:20px;
      font-size:25px;
      color:red;
    }
  </style>
</head>
<body>

  <div class="container">
    <div class="row">
      <div class="col-md-3"></div>
      <div class="col-md-6">
        <div class="page-header">
          <h1 style="color:red;">Credit Card Approval Prediction</h1>
        </div>
      </div>
    </div>

    <div class="container">
      <div class="row">
        <div class="col-md-3"></div>
        <div class="col-md-6">
          <form action="/data_predict" method="POST">
            <div class="col-md-6">
              <div class="form-group">
                <label for="gender" style="color:red;">Gender</label>
                <select name="gender" class="form-control" id="gender">
                  <option value="0">FEMALE</option>
                  <option value="1">MALE</option>
                </select>
              </div>
            </div>
            <div class="col-md-6">
              <div class="form-group">
                <label for="car" style="color:red;">Own Car Or NOT</label>
                <select name="car" class="form-control" id="car">
                  <option value="0">NO</option>
                  <option value="1">Yes</option>
                </select>
              </div>
            </div>
            <div class="col-md-6">
              <div class="form-group">
                <label for="estate" style="color:red;">Own Realstate</label>
                <select name="estate" class="form-control" id="estate">
                  <option value="0">No</option>
                  <option value="1">Yes</option>
                </select>
              </div>
            </div>
            <div class="col-md-6">
              <div class="form-group">
                <label for="past" style="color:red;">Past</label>
                <input type="number" class="form-control" id="past" name="past" >
              </div>
            </div>
            <div class="col-md-6">
              <div class="form-group">
                <label for="loan" style="color:red;">No of Loans</label>
                <input type="number" class="form-control" id="loan" name="loan" >
              </div>
            </div>
          </form>
          <div class="text-center">
            <button type="submit" class="btn btn-default" style="color:red;">Predict</button>
          </div>
        </div>
      </div>
    </div>

    <div class="result">
      <div class="text-center">
        <h3>You Are
          {{prediction}}
          for credit card
        </h3></div>
      </div>

      <!-- Latest compiled and minified JavaScript -->
      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    </body>
  </html>

```

Activity 2: Build Python code:

Import the libraries

Pickle: Pickle is a module in Python used for serializing and de-serializing Python objects.Flask:

Refer prior knowledge section mentioned above.

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered.

Whenever you enter the values from the predict.html page the values can be retrieved using POST Method.

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

```
from flask import Flask, request, render_template
app = Flask(__name__)
import joblib, pickle
import numpy as np
app = Flask(__name__)
ct = joblib.load("cred")
model = pickle.load(open("modelcredit.pkl", "rb"))

@app.route('/') # rendering the html template
def predict():
    return render_template('cre.html')

@app.route('/data_predict', methods=['POST']) # route for our prediction
def data_predict():

    x_test = [[(x) for x in request.form.values()]]
    #x_test = np.array(x_test)
    #x_test=ct.transform(x_test)
    pred = model.predict(x_test)
    #print(pred)
    if pred[0]==0:
        prediction="Not Eligible"
    else:
        prediction="Eligible"

    return render_template("cre.html", prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```


Activity 3: Run the App

- Open anaconda prompt from the start menu
- Navigate to the folder where your pythonscript is.
- Nowtype “python cred.py”command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs,click onthe submit button, and see the result/prediction on the web.

```
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/jagad/Dropbox/PC/Downloads/Credit card approval prediction/Credit card
approval prediction/ML/flask/cred.py', wdir='C:/Users/jagad/Dropbox/PC/Downloads/Credit card approval
prediction/Credit card approval prediction/ML/flask')
* Serving Flask app "cred" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsani)
```

Now Enter the URL, localhost:5000 on the browser, you will redirectto cred.html page.

Let's look our credpage



Credit Card Approval Prediction

Gender	Own Car Or NOT
<input type="text" value="FEMALE"/>	<input type="text" value="NO"/>
Own Realstate	Total Anual Income
<input type="text" value="No"/>	<input type="text"/>
Type of Income	Education
<input type="text" value="Pensioner"/>	<input type="text" value="Higher Education"/>
Family Status	Type of Housing
<input type="text" value="Married"/>	<input type="text" value="House / apartment"/>
Days Birth	Days Employed
<input type="text"/>	<input type="text"/>
Family Members	Emi Paid Off
<input type="text"/>	<input type="text"/>
Emi Of Pastdues	No of Loans
<input type="text"/>	<input type="text"/>

You Are for credit card

To predict your credit card eligibility, click on predict button. The output will be displayed on page.

now see the Eligible result in the credit card approval prediction.



Credit Card Approval Prediction

Gender	Own Car Or NOT
<input type="text" value="FEMALE"/>	<input type="text" value="NO"/>
Own Realstate	Total Annual Income
<input type="text" value="No"/>	<input type="text"/>
Type of Income	Education
<input type="text" value="Pensioner"/>	<input type="text" value="Higher Education"/>
Family Status	Type of Housing
<input type="text" value="Married"/>	<input type="text" value="House / apartment"/>
Days Birth	Days Employed
<input type="text"/>	<input type="text"/>
Family Members	Emi Paid Off
<input type="text"/>	<input type="text"/>
Emi Of Pastdues	No of Loans
<input type="text"/>	<input type="text"/>
<input type="button" value="Predict"/>	

You Are Eligible for credit card



And now see the not Eligible result in the credit card approval prediction.

localhost:5000/data_predict

Credit Card Approval Prediction

Gender	Own Car Or NOT
<input type="text" value="FEMALE"/>	<input type="text" value="NO"/>
Own Realstate	Total Anual Income
<input type="text" value="No"/>	<input type="text"/>
Type of Income	Education
<input type="text" value="Pensioner"/>	<input type="text" value="Higher Education"/>
Family Status	Type of Housing
<input type="text" value="Married"/>	<input type="text" value="House / apartment"/>
Days Birth	Days Employed
<input type="text"/>	<input type="text"/>
Family Members	Emi Paid Off
<input type="text"/>	<input type="text"/>
Emi Of Pastdues	No of Loans
<input type="text"/>	<input type="text"/>

You Are Not Eligible for credit card

So this is the output after entering some values.

Thank You

