SMARTINTERNZ

Credit Card Approval Prediction Using Machine Learning

Project Report

introduction

1.1 overview

Credit risk as the board in banks basically centers around deciding the probability of a customer's default or credit decay and how expensive it will end up being assuming it happens. It is important to consider major factors and predict beforehand the probability of consumers defaulting given their conditions. Which is where a machine learning model comes in handy and allows the banks and major financial institutions to predict whether the customer, they are giving the loan to, will default or not. This project builds a machine learning model with the best accuracy possible using python. First we load and view the dataset. The dataset has a combination of both mathematical and non-mathematical elements, that it contains values from various reaches, in addition to that it contains a few missing passages. We preprocess the dataset to guarantee the AI model we pick can make great expectations. After the information is looking great, some exploratory information examination is done to assemble our instincts. Finally, we will build a machine learning model that can predict if an individual's application for a credit card will be accepted. Using various tools and techniques we then try to improve the accuracy of the model. This project uses Jupyter notebook for python programming to build the machine learning model. Using Data Analysis and Machine Learning, we attempted to determine the most essential parameters for obtaining credit card acceptance in this project.

1.2 Purpose :

It uses personal information and data submitted by credit card applicants to predict the probability of future defaults and credit card borrowings. The bank is able to decide whether to issue a credit card to the applicant. Credit scores can objectively quantify the magnitude of risk.

2 LITERATURE SURVEY

2.1 Existing problem :

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.

2.2 Proposed solution

This a classic example of supervised learning. We have been provided with a fixed number of features for each data point, and our aim will be to train a variety of Supervised Learning algorithms on this data, so that , when a new data point arises, our best performing classifier can be used to categorize the data point as a positive example or negative. Exact details of the number and types of algorithms used for training is included in the 'Algorithms and Techniques' sub-section of the 'Analysis' part

.

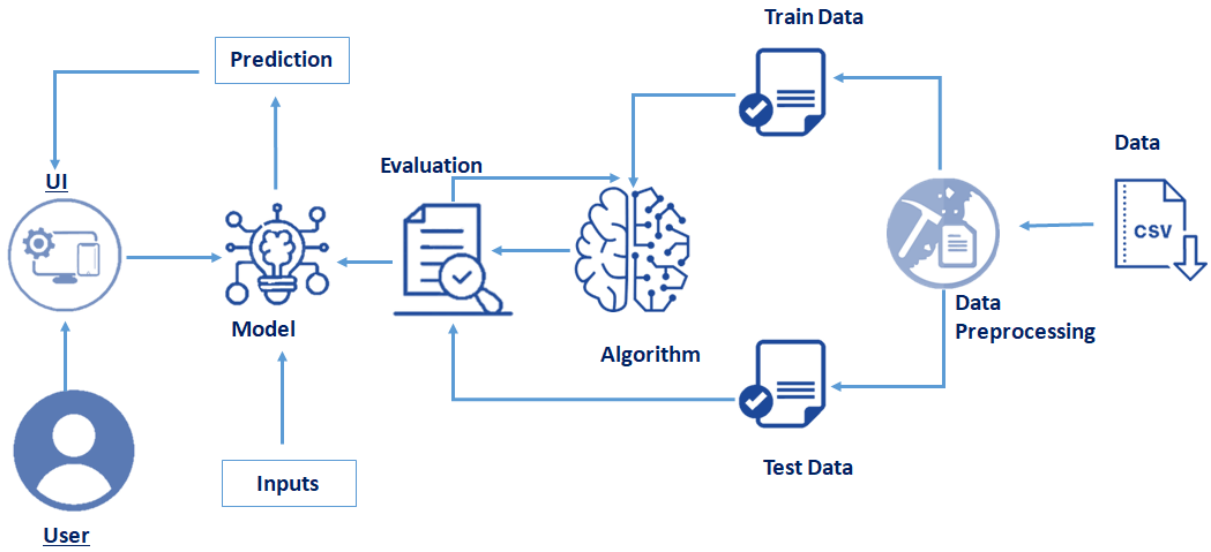
This project focuses on the related works of various authors on credit card approval such that algorithms were implemented using Jupyter that is a machine learning software written in Python. Various attributes that are essential in the prediction of credit card were examined and the dataset of applicants were also evaluated.

This project compares various classification algorithms such as Random Forest,KNN classification Algorithm with an aim to identify the best technique. Based on this study, Random Forest with the highest accuracy outperformed the other algorithms and can be further utilised in the prediction of credit card recommended to the user.

Later by using Flask app create html files and create an user interface to display whether the candidate is eligible for credit card or not.

3.THEORITICAL ANALYSIS

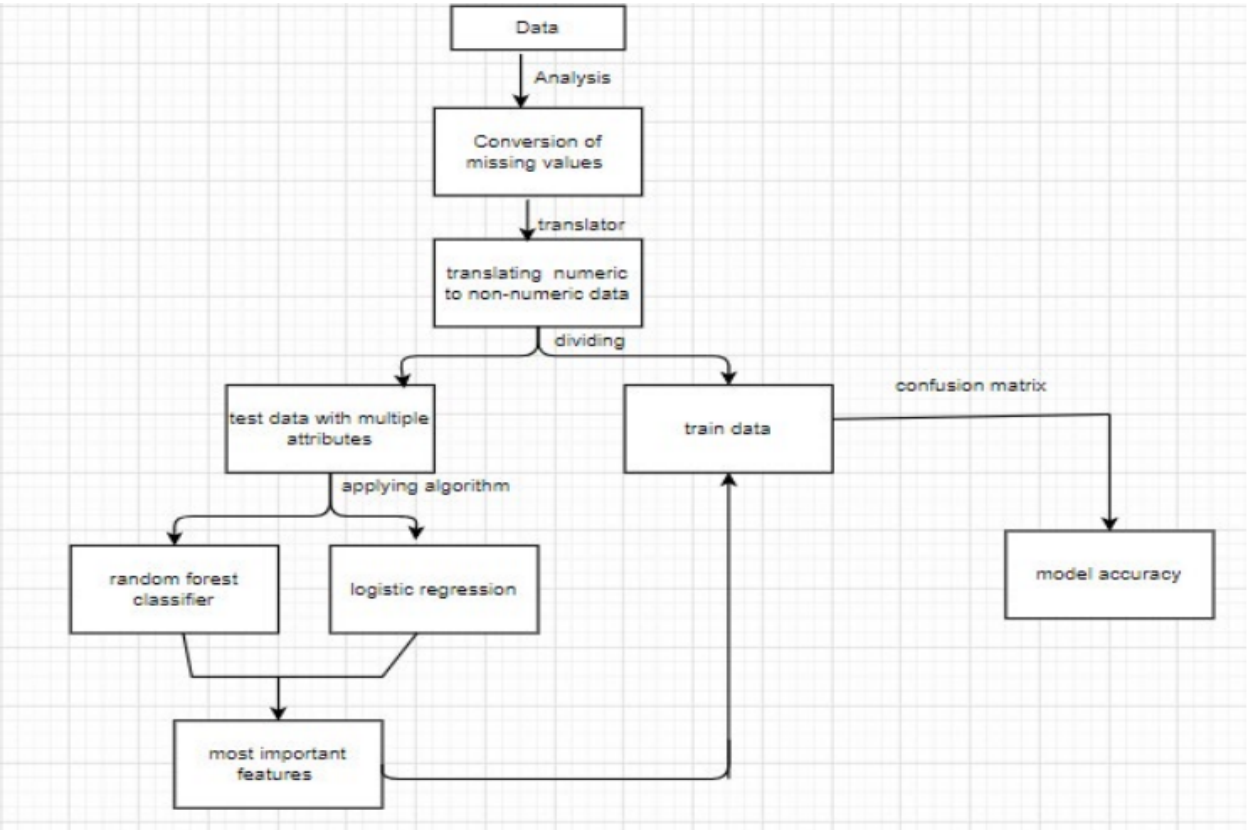
3.1 Block Diagram :



3.2 Software Requirements :

- ANACONDA NAVIGATOR
- JUPYTER NOTEBOOK
- SPIDER

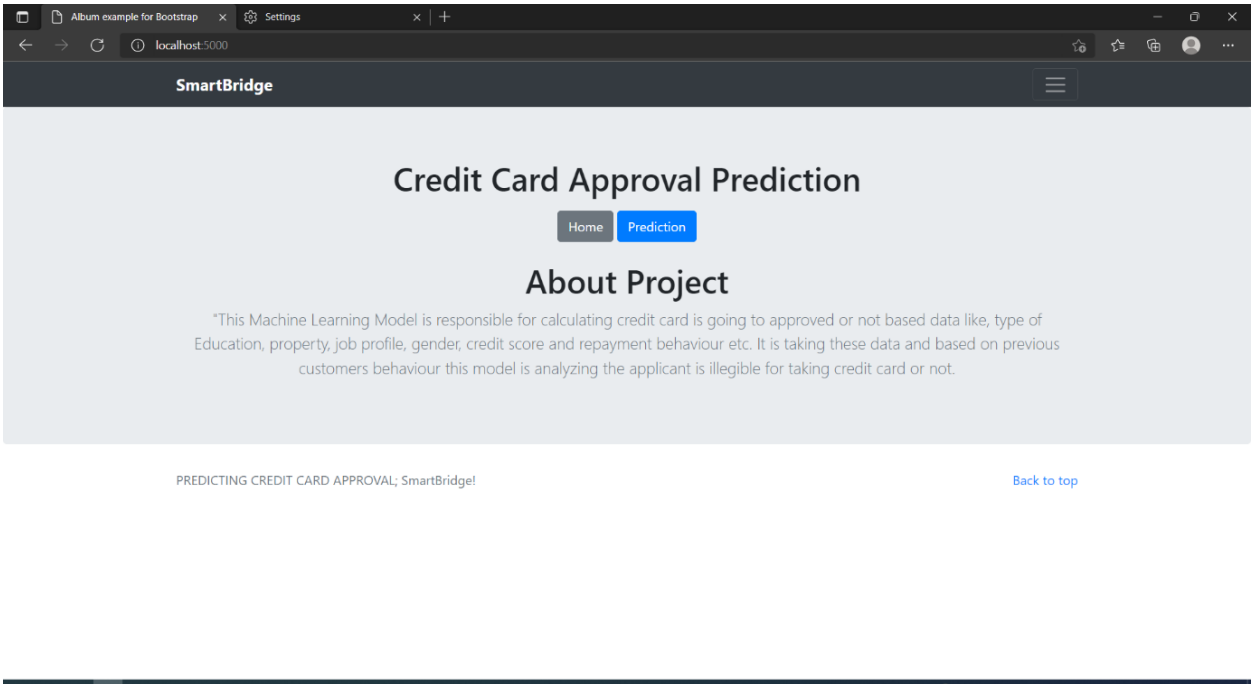
3.3 FLOW CHART



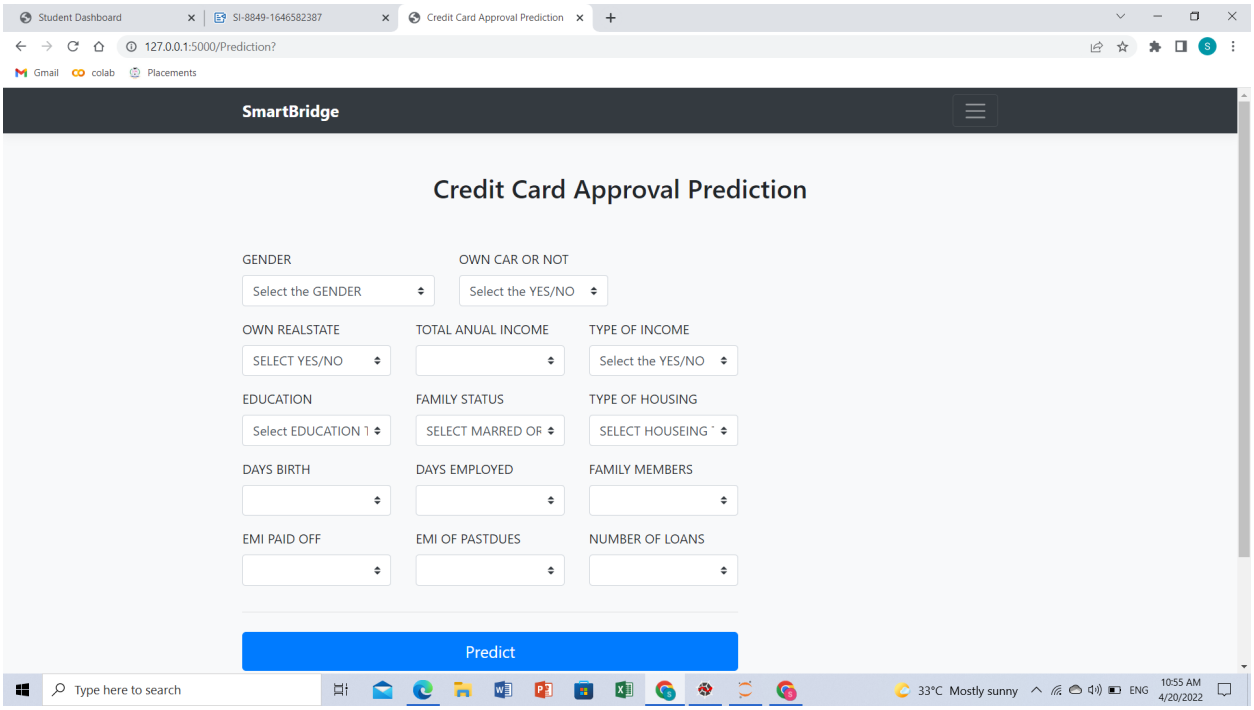
4. Project Flow

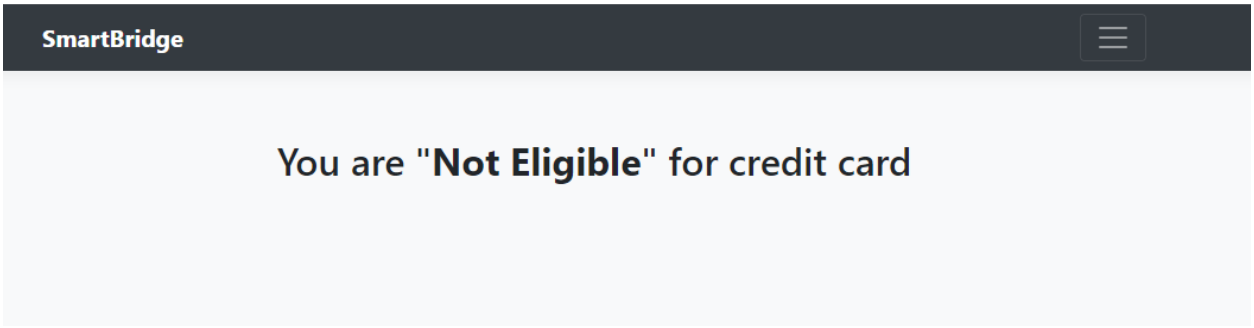
- Data Collection.
- Data Visualization
- Data Pre-processing
- Model Building
- Application Building

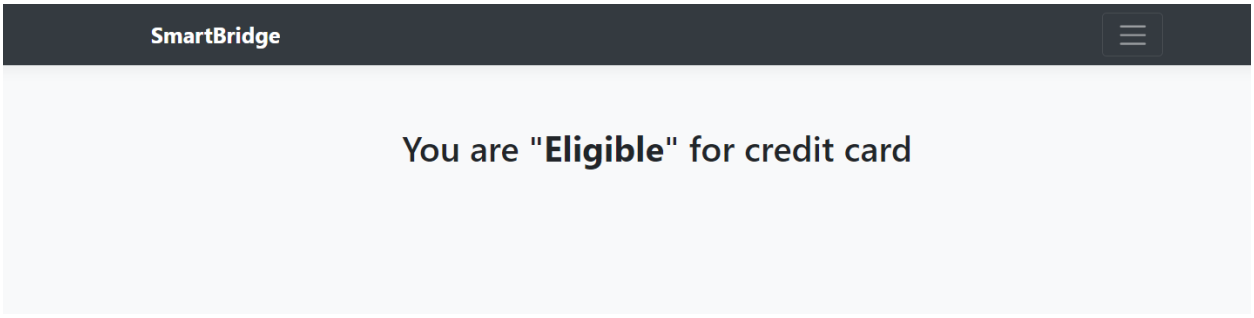
5.RESULT



This is our home page where we get to know the summary of the project.







As we see the predicted output is displayed on the User Interface

6 . ADAVANTAGES AND DISADAVANTAGES :

6.1 ADAVANTAGES

- Opportunity to build credit
- Earn rewards such as cash back or miles points
- Protection against credit card fraud
- Free credit score information
- No foreign transaction fees

6.2 DISADAVANTAGES

- Minimum due trap
- Hidden costs
- Ease of overuse
- High intrest rate

7. CONCLUSION :

Currently, factors considered are regular details related to gender, age of the consumer, his/hercredit reports and worthiness, yearly income, and the number of years he/she has been working.Further, to improve this work, various other factors or conditions can be considered like theirhistory related to any offense and their assets which can be both physical and liquid cash. Thesefeatures can improve the model to be more effective and can help the institutes to make betterdecisions so that they can avoid experiencing frauds and loss. Various classification algorithmscan be used to build a model and compare the rates or levels of accuracy to improve the model for better use.

7. FUTURE SCOPE :

From this initial analysis, we are able to conclude that the most significant factors in determining the outcome of a credit application are Employment, Income, Credit Score and Prior Default.

Based on these insights, we can work on building some predictive models. They can be used by analysts in financial sector and be incorporated to automate the credit approval process. These results can also serve as a source of information for the consumers.

This study can include the classifiers to use TensorFlow as the library to increase accuracy, precision, and the other parameters. This study uses only seven supervised Machine Learning models which can be increased and some more models including both supervised and unsupervised models can be used to find best model.

8.REFERENCE :

- K. Chaudhary, J. Yadav, and B. Mallick, “A review of fraud detection techniques: Credit card,” International Journal of Computer Applications

- Youtube videos by simplilearn, Datacamp, Codebasics, eduraka!.
- Smartinternz tutorial classes help me for completion of project.

9.APPENDIX

9.1 Flask Code

```
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
import os
app=Flask(__name__)# initializing a flask app
#filepath="I:\SmartBridge Projects\Co2 emission\co2.pickle"
#model=pickle.load(open(co2.pickle,'rb'))

with open('model.pkl','rb') as handle:
    model = pickle.load(handle)

@app.route('/')# route to display the home page
def home():
    return render_template('index.html') #rendering the home page
@app.route('/Prediction',methods=['POST','GET'])
def prediction(): # route which will take you to the prediction page
    return render_template('index1.html')
@app.route('/Home',methods=['POST','GET'])
def my_home():
    return render_template('index.html')

@app.route('/predict',methods=["POST","GET"])# route to show the predictions in a web UI
def predict():
    #  reading the inputs given by the user
    input_feature=[float(x) for x in request.form.values() ]
    features_values=[np.array(input_feature)]
```

```
feature_name=['CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','AMT_INCOME_TOTAL','NAM
E_INCOME_TYPE','NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','DAY
S_BIRTH','DAYS_EMPLOYED','CNT_FAM_MEMBERS','paid_off','#_of_pastdues','no_loan']
    x=pd.DataFrame(features_values,columns=feature_name)

     # predictions using the loaded model file
    pred=model.predict(x)
    print(pred)
    if pred==0:
        prediction = "Eligible"
    else:
        prediction = "Not Eligible"



    #prediction="Prediction is:"+str(predic)

     # showing the prediction results in a UI
    return render_template("result.html",prediction=prediction)
if __name__=="__main__":

    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=False,use_reloader=False)
```

SOURCE CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import classification_report,confusion_matrix,f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
app = pd.read_csv('application_record.csv')
credit = pd.read_csv('credit_record.csv')
app.head()
app.tail()
credit.head()
credit.tail()
print("Number of people working status:")
print(app['OCCUPATION_TYPE'].value_counts())
sns.set(rc ={'figure.figsize':(18,6)})
sns.countplot(x='OCCUPATION_TYPE',data=app,palette = 'Set2')
print('Types of house of the people:')
print(app['NAME_HOUSING_TYPE'].value_counts())
sns.set(rc = {'figure.figsize':(15,4)})
sns.countplot(x='NAME_HOUSING_TYPE',data=app,palette ='Set2')
print("Income types of the Person:")
print(app['NAME_INCOME_TYPE'].value_counts())
sns.set(rc ={'figure.figsize':(8,5)})
```

sns.countplot(x='NAME_INCOME_TYPE',data=app,palette ='Set2')

flg, ax=plt.subplots(figsize=(8,6))

sns.heatmap(app.corr(),annot=True)

app.describe()

app.info()

```
def unique_values():
    a = app.CODE_GENDER.unique()
    print("-------CODE_GENDER---------")
    print(a)
    print()
    b = app.FLAG_OWN_CAR.unique()
    print("--------FLAG_OWN_CAR---------")
    print(b)
    print()
    c = app.FLAG_OWN_REALTY.unique()
    print("-------FLAG_OWN_REALTY---------")
    print(c)
    print()
    d = app.CNT_CHILDREN.unique()
    print("---------CNT_CHILDREN------")
    print(d)
    print()
    e = app.NAME_INCOME_TYPE.unique()
    print("---------NAME_INCOME_TYPE------")
    print(e)
    print()
    f = app.NAME_EDUCATION_TYPE.unique()
    print("-------NAME_EDUCATION_TYPE--------")
    print(f)
    print()
    g = app.NAME_FAMILY_STATUS.unique()
    print("------NAME_FAMILY_STATUS------")
    print(g)
    print()
    h = app.NAME_HOUSING_TYPE.unique()
    print("------NAME_HOUSING_TYPE------")
    print(h)
    print()
    i = app.OCCUPATION_TYPE.unique()
```

```
    print("------OCCUPATION_TYPE-----")
    print(i)
    print()
    j = app.CNT_FAM_MEMBERS.value_unique()
    print("-------CNT_FAM_MEMBERS------")
    print(j)
    print()
    return unique_values

app.drop_duplicates(subset =
['CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','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'],keep = 'first',inplace = True

app.isnull().mean()

def data_cleansing(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)

    data['DAYS_BIRTH'] = np.abs(data['DAYS_BIRTH']/365)
    data['DAYS_EMPLOYED'] = data['DAYS_EMPLOYED']/365

    housing_type = {'House / apartment' : 'House / apartment',
                    'with parents' : 'With parents',
                    'Municipal apartment' : 'House / apartment',
                    'Rented apartment' : 'House / apartment',
                    'Office apartment' : 'House / apartment',
                    'co-op apartment' : 'House / apartment'}
    income_type = {'Commercial associate' : 'Working',
                   'State servant' : 'Woking',
                   'Working' : 'Working',
                   'Pensioner' : 'Pensioner',
                   'Student' : 'Student'}
    education_type = {'Secondary / secondary special':'secondary',
                      'Lower secondary' : 'Secondary',
                      'Higher education' : 'Higher education',
                      'Incomplete higher' : 'Higher education',
                      'Academic degree' : 'Academic degree'}
```

```
    family_status = {'Single / not married' : 'Single',
                     'Separated' : 'Single',
                     'widow' : 'Single',
                     'Civil marriage' : 'Married',
                     'Married' : 'Married'}
    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

credit.head()
```

credit.shape

credit.info

```
grouped = credit.groupby('ID')

pivot_tb = credit.pivot(index = 'ID',columns = 'MONTHS_BALANCE',values = 'STATUS')
pivot_tb['open_month'] = grouped['MONTHS_BALANCE'].min()
pivot_tb['end_month'] = grouped['MONTHS_BALANCE'].max()
pivot_tb['window'] = pivot_tb['end_month'] - pivot_tb['open_month']
pivot_tb['window'] += 1

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

# setting Id column to merge with app data.
pivot_tb['ID'] = pivot_tb.index
pivot_tb.head(15)
pivot_tb.tail(15)


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

target = pd.DataFrame()
target['ID'] = pivot_tb.index
target['paid_off'] = pivot_tb['paid_off'].values
target['#_of_pastdues'] = pivot_tb['pastdue_1-29'].values + pivot_tb['pastdue_30-59'].values
+ pivot_tb['pastdue_60-89'].values + pivot_tb ['pastdue_90-119'].values
+ pivot_tb['pastdue_120-149'].values + pivot_tb['pastdue_over_150'].values
```

```
target['no_loan'] = pivot_tb['no_loan'].values
target['target'] = feature_engineering_target(pivot_tb)
credit_app = app.merge(target, how ='inner' ,on = 'ID')
credit_app.drop('ID',axis = 1,inplace = True)


from sklearn.preprocessing import LabelEncoder
cg=LabelEncoder ()
oc=LabelEncoder ()
own_r= LabelEncoder ()
it = LabelEncoder ()
et = LabelEncoder ()
fs= LabelEncoder ()
ht = LabelEncoder ()
nf = LabelEncoder ()
nh = LabelEncoder ()
o =  LabelEncoder ()
credit_app['CODE_GENDER'] = cg.fit_transform(credit_app[ 'CODE_GENDER'])
credit_app['FLAG_OWN_CAR'] = oc.fit_transform(credit_app['FLAG_OWN_CAR'])
credit_app['FLAG_OWN_REALTY']= own_r.fit_transform(credit_app[ 'FLAG_OWN_REALTY'])
credit_app['NAME_INCOME_TYPE'] = it.fit_transform (credit_app['NAME_INCOME_TYPE'])
credit_app['NAME_EDUCATION_TYPE'] = et.fit_transform(credit_app['NAME_EDUCATION_TYPE'])
credit_app['NAME FAMILY STATUS'] = fs.fit_transform(credit_app['NAME_FAMILY_STATUS'])
credit_app['NAME HOUSING_TYPE'] = ht.fit_transform(credit_app['NAME_HOUSING_TYPE'])
credit_app['NAME_FAMILY_STATUS'] = nf.fit_transform(credit_app['NAME_FAMILY_STATUS'])
credit_app['NAME_HOUSING_TYPE'] = nf.fit_transform(credit_app['NAME_HOUSING_TYPE'])
credit_app['OCCUPATION_TYPE'] = nf.fit_transform(credit_app['OCCUPATION_TYPE'])

x = credit_app[credit_app.drop('target', axis = 1).columns]
y = credit_app['target']
xtrain,xtest,ytrain,ytest = train_test_split(x,y,train_size = 0.8, random_state = 0)


def logistic_reg(xtrain,xtest, ytrain, ytest):
  lr-LogisticRegression (solver-'liblinear')
  lr.fit(xtrain, ytrain)
  ypred-lr.predict(xtest)
  print('***LogisticRegression***')
  print('Confusion matrix')
  print(confusion_matrix(ytest,ypred))
  print('Classification report')
  print(classification_report(ytest, ypred))
```

```
def random_forest(xtrain, xtest, ytrain,ytest):
  rf=RandomForestClassifier()
  rf.fit(xtrain, ytrain)
  ypred-rf.predict(xtest)
  print('***RandomForestClassifier***')
  print('Confusion matrix')
  print(confusion_matrix(ytest,ypred))
  print('Classification report')
  print(classification_report(ytest,ypred))


def g_boosting (xtrain, xtest, ytrain,ytest):
   gb=GradientBoostingClassifier()
   gb.fit(xtrain, ytrain)
   ypred=gb.predict(xtest)
   print('***Gradient BoostingClassifier***')
   print('Confusion matrix')
   print(confusion_matrix(ytest,ypred))
   print('Classification report')
   print(classification_report(ytest, ypred))

def d_tree(xtrain,xtest, ytrain,ytest):
   dt=DecisionTreeClassifier()
   dt.fit(xtrain, ytrain)
   ypred=dt.predict(xtest)
   print('***DecisionTreeClassifier***')
   print('Confusion matrix')
   print(confusion_mattix(ytest,ypred))
   print('Classification report')
```

```
  print(classification_report(ytest,ypred))


def compare_model (xtrain,xtest, ytrain,ytest):
  logistic_reg(xtrain, xtest, ytrain,ytest)
  print('-'*100)
  random_forest(xtrain, xtest, ytrain,ytest)
  print('-'*100)
  g_boosting(xtrain, xtest, ytrain,ytest)
  print('-'*100)
  d_tree(xtrain, xtest, ytrain,ytest)

dt = DecisionTreeClassifier()
dt.fit(xtrain,ytrain)
ypred = dt.predict(xtest)

import pickle
pickle.dump(dt,open("model.pk1","wb"))
```