

# 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 :**

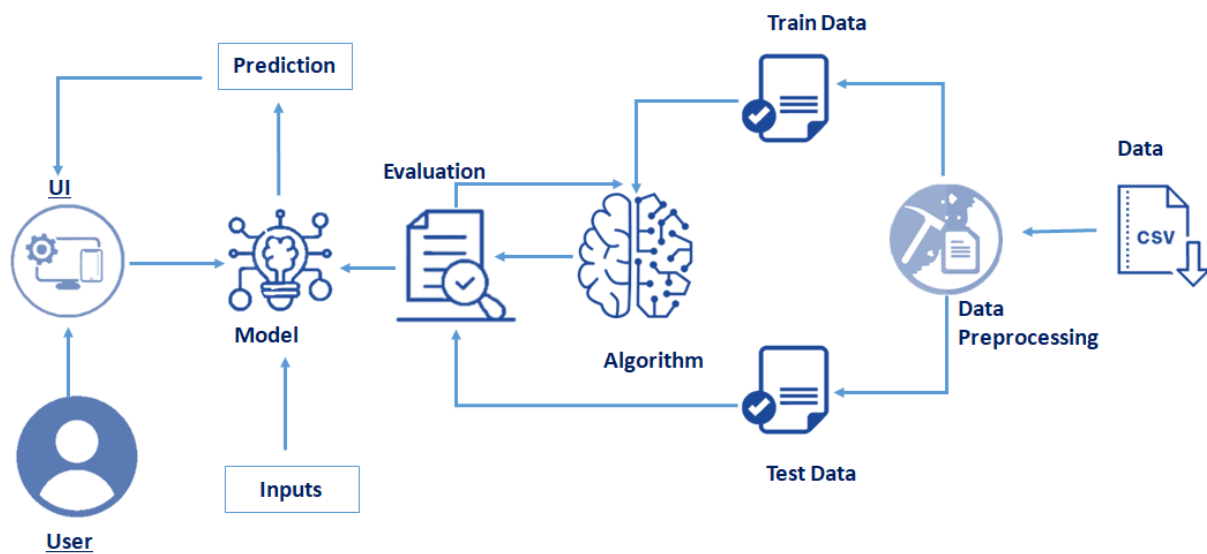
This is a prediction problem in which we need to predict whether a person is eligible for a credit card or not.

### **2.2 Proposed solution**

It means the combination of software, hardware, other products or equipment, and any and all services (including any installation, implementation, training, maintenance and support services) necessary to implement the solution described by user in its Proposal.

## **3. THEORETICAL ANALYSIS**

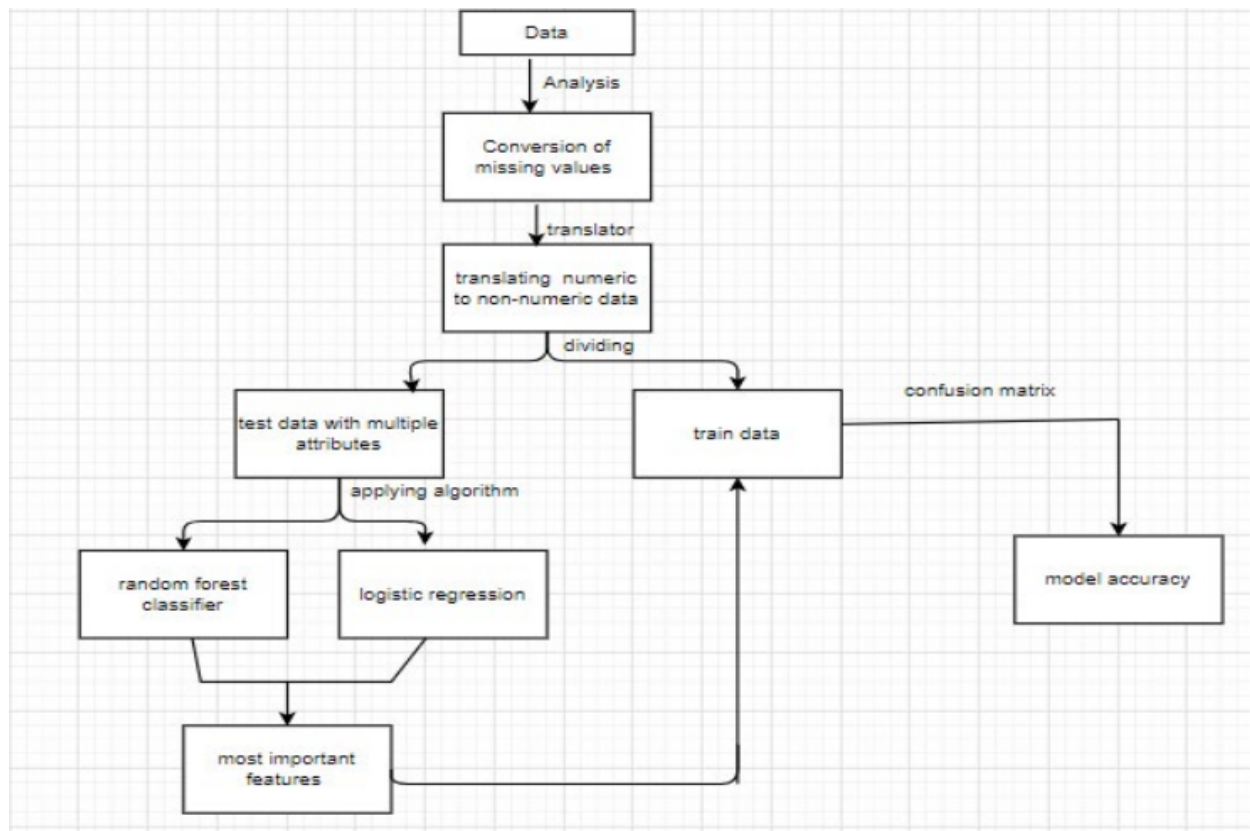
### 3.1 Block Diagram :



### 3.2 Software Requirements :

- ANACONDA NAVIGATOR
- JUPYTER NOTEBOOK
- SPIDER

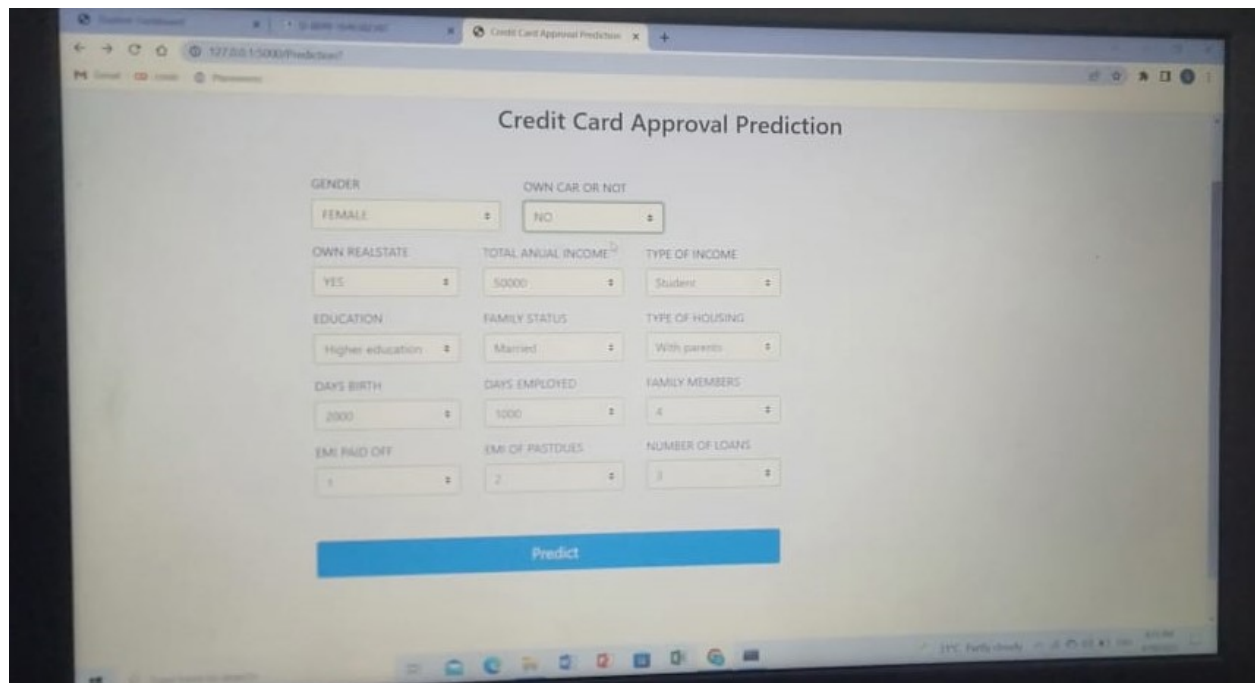
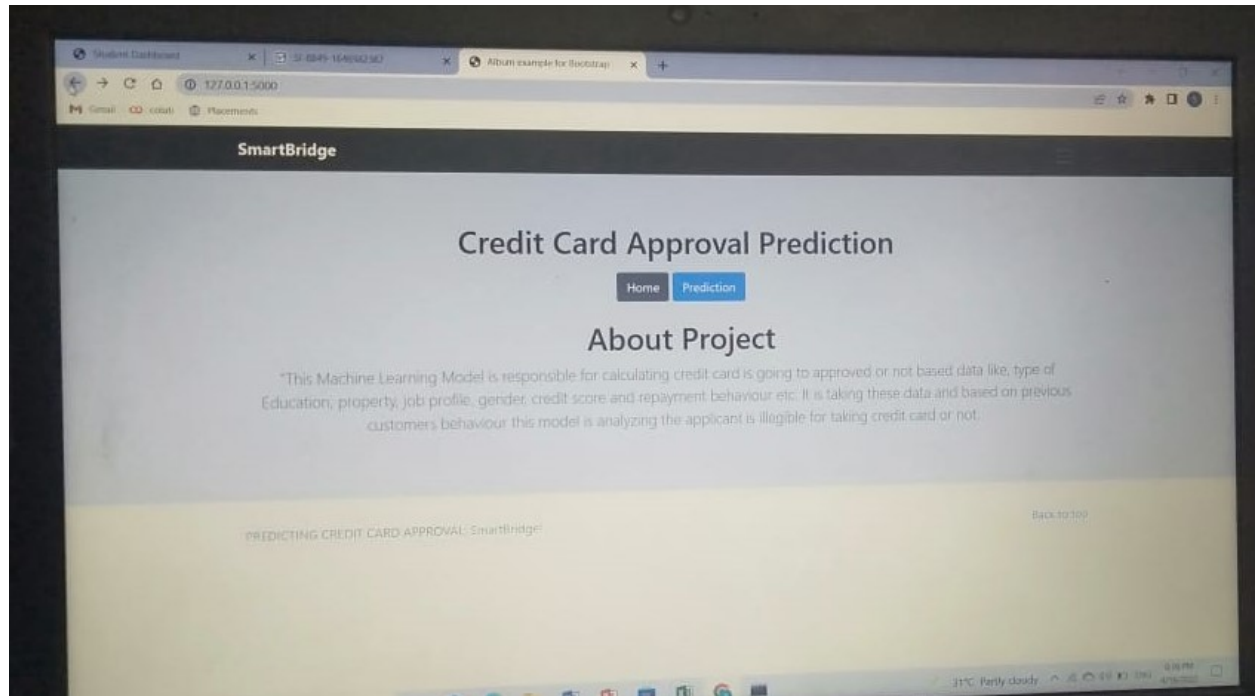
### 3.3 FLOW CHART

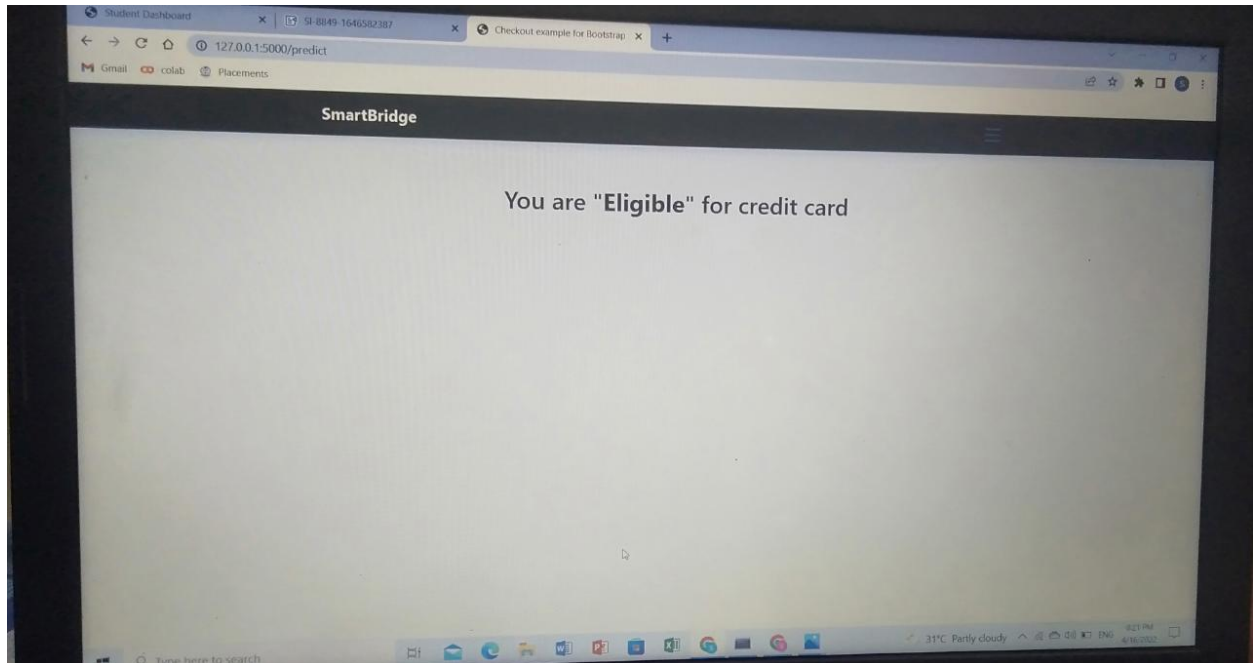


### 4. Project Flow

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

## 5.RESULT





## 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/her credit 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 their history related to any offense and their assets which can be both physical and liquid cash. These features can improve the model to be more effective and can help the institutes to make better decisions so that they can avoid experiencing frauds and loss. Various classification algorithms can be used to build a model and compare the rates or levels of accuracy to improve the model for better use.

## 7. FUTURE SCOPE :

machine learning needs you to know computer programming, statistics and data evaluation, the future scope of your machine learning career can also be in **leadership roles in automation or analytics environments**

## 8. REFERENCE :

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

## 9. APPENDIX

### 9.1 source code

```

D: > Credit card approval prediction > Loan Approval Prediction > Flask > app.py > ...
1  # importing the necessary dependencies
2  from flask import Flask,request,render_template
3  import numpy as np
4  import pandas as pd
5  import pickle
6  import os
7  app=Flask(__name__)# initializing a flask app
8  #filepath="I:\SmartBridge Projects\Co2 emission\co2.pickle"
9  #model=pickle.load(open(co2.pickle,'rb'))
10
11 with open('model.pkl','rb') as handle:
12     model = pickle.load(handle)
13
14 @app.route('/')# route to display the home page
15 def home():
16     return render_template('index.html') #rendering the home page
17 @app.route('/Prediction',methods=['POST','GET'])
18 def prediction(): # route which will take you to the prediction page
19     return render_template('index1.html')
20 @app.route('/Home',methods=['POST','GET'])
21 def my_home():
22     return render_template('index.html')

```

```

23     return render_template('index.html')
24
25 @app.route('/predict',methods=['POST','GET'])
26 def predict():
27
28     input_feature=[float(x) for x in request.form.values() ]
29     features_values=[np.array(input_feature)]
30     feature_name=['CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','AMT_INCOME_TOTAL','NAME_INCOME_TYPE','NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS']
31     x=pd.DataFrame(features_values,columns=feature_name)
32
33     # predictions using the loaded model file
34     pred=model.predict(x)
35     print(pred)
36     if pred==0:
37         prediction = "Eligible"
38     else:
39         prediction = "Not Eligible"
40
41     #prediction="Prediction is:"+str(predic)
42
43     # showing the prediction results in a UI
44     return render_template("result.html",prediction=prediction)
45 if __name__=="__main__":
46
47     # app.run(host='0.0.0.0', port=8080,debug=True)    # running the app
48     port=int(os.environ.get('PORT',5000))
49     app.run(port=port,debug=False,use_reloader=False)

```

SOURCE CODE:



```
In [2]: 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
```

```
In [3]: app = pd.read_csv('application_record.csv')
credit = pd.read_csv('credit_record.csv')
```

```
In [4]: app.head()
```

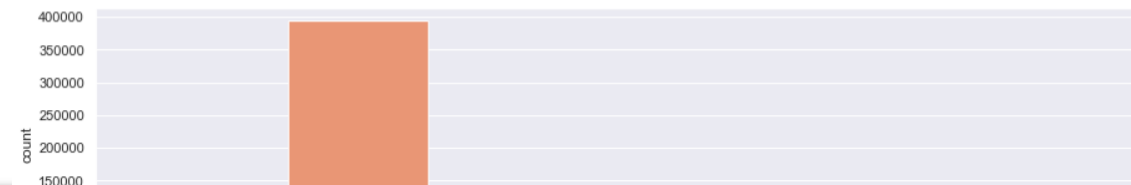
```
Out[4]:
```

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	N
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	

```
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')
```

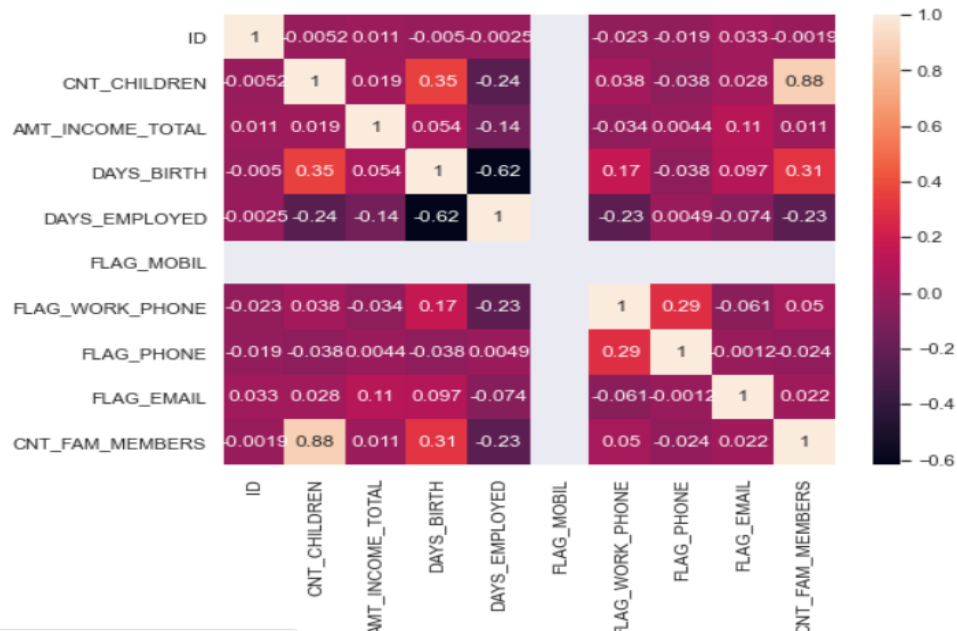
```
Types of house of the people:
House / apartment      393831
With parents           19077
Municipal apartment    14214
Rented apartment       5974
Office apartment       3922
Co-op apartment        1539
Name: NAME_HOUSING_TYPE, dtype: int64
```

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



```
flg, ax=plt.subplots(figsize=(8,6))
sns.heatmap(app.corr(),annot=True)
```

```
<AxesSubplot:>
```



```
app.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 438557 entries, 0 to 438556
Data columns (total 18 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   ID                                     438557 non-null  int64
1   CODE_GENDER                           438557 non-null  object
2   FLAG_OWN_CAR                           438557 non-null  object
3   FLAG_OWN_REALTY                         438557 non-null  object
4   CNT_CHILDREN                           438557 non-null  int64
5   AMT_INCOME_TOTAL                       438557 non-null  float64
6   NAME_INCOME_TYPE                       438557 non-null  object
7   NAME_EDUCATION_TYPE                    438557 non-null  object
8   NAME_FAMILY_STATUS                     438557 non-null  object
9   NAME_HOUSING_TYPE                      438557 non-null  object
10  DAYS_BIRTH                             438557 non-null  int64
11  DAYS_EMPLOYED                           438557 non-null  int64
12  FLAG_MOBIL                             438557 non-null  int64
13  FLAG_WORK_PHONE                         438557 non-null  int64
14  FLAG_PHONE                             438557 non-null  int64
15  FLAG_EMAIL                             438557 non-null  int64
16  OCCUPATION_TYPE                         304354 non-null  object
17  CNT_FAM_MEMBERS                         438557 non-null  float64
dtypes: float64(2), int64(8), object(8)
memory usage: 60.2+ MB
```

```
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 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"))
```

THANK YOU

