LOAN ELIGIBILITY PREDICTION

A SKILLING PROJECT REPORT

Submitted towards the Professional course

19CS3021S Machine Learning

190030433 E Jhasha sri

190030485 G G Sanjana

190030519 G Venkatesh

BACHELOR OF TECHNOLOGY

Branch: Computer Science and Engineering

(DEEMED TO BE UNIVERSITY)

October 2021

Department of Computer Science and Engineering

K L Deemed to be University,

Green Fields, Vaddeswaram,

Guntur District, A.P, 522502.

INDEX

ABSTRACT

Our project title is Loan Eligibility prediction. A loan is when one or more people, organizations, or other entities lend money to other people, organizations, or entities. The recipient incurs a debt for which he or she is generally responsible for paying interest until the loan is repaid.

The project's objective is to ensure that a person, institution, or organization applying for a loan is verified thoroughly before sanctioning them a loan. Several criteria like gender, education, number of dependents, to name a few, have to be taken into consideration before approving the loan. The project aims at automating the procedure, thus, helping in reducing the time and energy and making the process more efficient. Two types of data – train data and test data – are made from the input dataset. The train data is used to train the Machine Learning Model and determine its accuracy. The test data is used to output the loan eligibility predictions.

INTRODUCTION

A monetary loan is when one or more persons, organizations, or other entities lend money to other people, organizations, or entities. The borrower incurs a debt for which he or she is generally responsible for paying interest until the loan is repaid along with the principal amount borrowed. Nowadays, sanctioning of loans has become a significant function of the financial institutions or banking sector. Loans are also one of the significant sources of income for banks. Banks apply interests on loans which are then sanctioned to their borrowers. While sanctioning a loan, the lender needs to have an assurance of earning their money back along with interest. Thus, identifying the creditworthiness of an individual or an organization is highly important before sanctioning the loan. In this project, we focus mainly on monetary loans. The project aims to thoroughly verify the borrower and perform a background check based on several variables like gender, income, employment status, etc., to ensure whether the borrower is creditworthy and can be sanctioned the loan or not.

REQUIREMENTS ELICITATION

Software Details:

- Python modules with machine learning libraries
- Windows Operating System
- Google Colab

Hardware Details:

System with:

- i5 processor
- 256 MB RAM or higher
- 256GB Hard disk

PROBLEM MODELLING

Each model in our project is made to work as follows:

- The train data is first fit into the model
- The outputs are predicted based on the provided attributes
- The accuracy is then computed by comparing the outputs to the already existing outcomes in the train data.
- Differences in predictions are recorded as errors, which are minimized to the greatest extent feasible.
- The test data is then fitted into the model to provide accurate predictions for each application if the accuracy is sufficient.
- Thus, the machine learning model is deployed to obtain

accurate dichotomous predictions for multiple independent entries.

Steps:

Loading Data:

We first divide the dataset into train data and test data by selecting the target variable. During this process we may delete some unique columns.

Analyzing data:

We have 32 features and 1 target variable, i.e. Loan_Default in the training dataset. We have similar features in the test dataset as the training dataset except for the Loan_Default. We will predict the Loan_Default using the model built using the train data.

Pre-Processing:

After analysing the data, we pre-process the data by removing null values, by visualization of data, pre-processing the data by using python libraries like label encoder etc…

Model Selection:

After Pre-Processing we select the best model for our requirements of predicting the target variable. Some of the models we selected for out project is

1. Classification:
    - Decision Tree Classifier
    - Random Forest Classifier
    - Naïve Bayes Classifier
    - KNN Classifier
2. Regression:
    - Logistic Regression

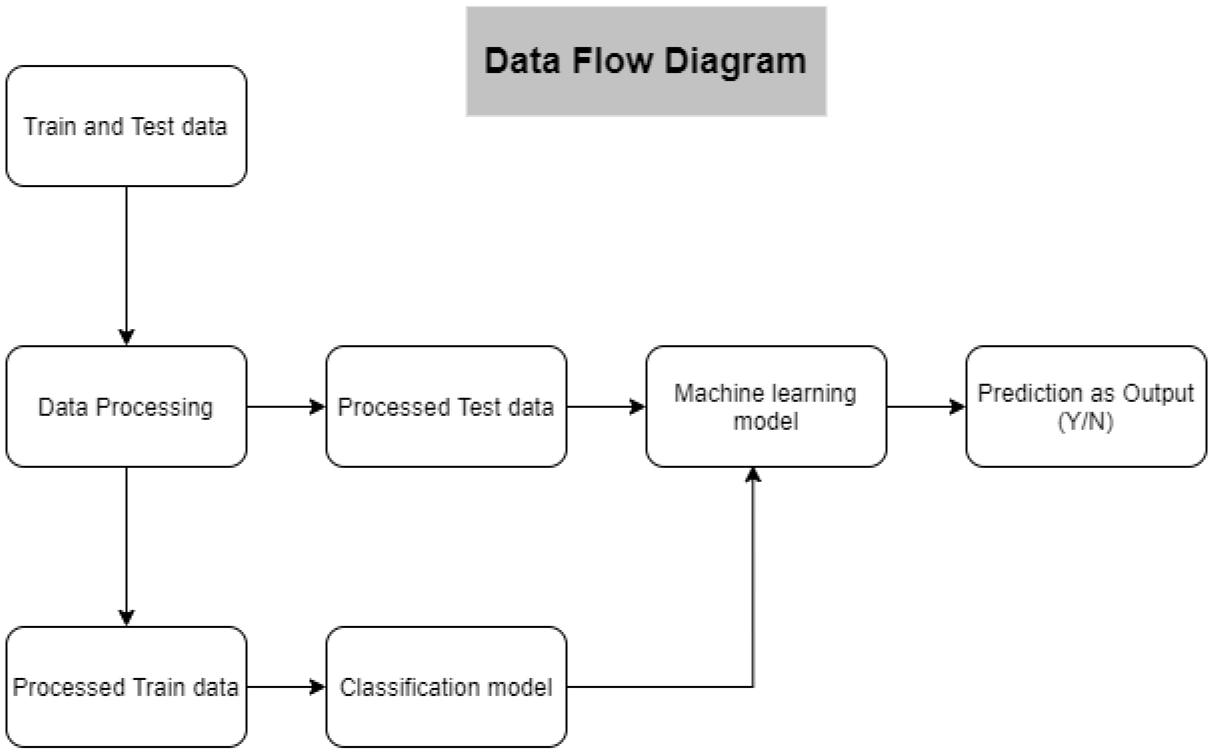
Fitting the Model:

After model selection we use python libraries and we fit our data into that model, and we train our model using the train data set.

Predicting:

After training the model we predict the accuracy, and we draw confusion matrix. We test our data with the model for accuracy.

SYSTEM DESIGN



Our project consists of three phases

1. Preprocessing
2. Training
3. Testing

They work as follows:

- All the raw data is fed into the machine which cleans and processes it for accurate testing and training.

- The train data is fed into the classification model, which determines the accuracy of the model.

- The test data set is fed into the machine learning model.

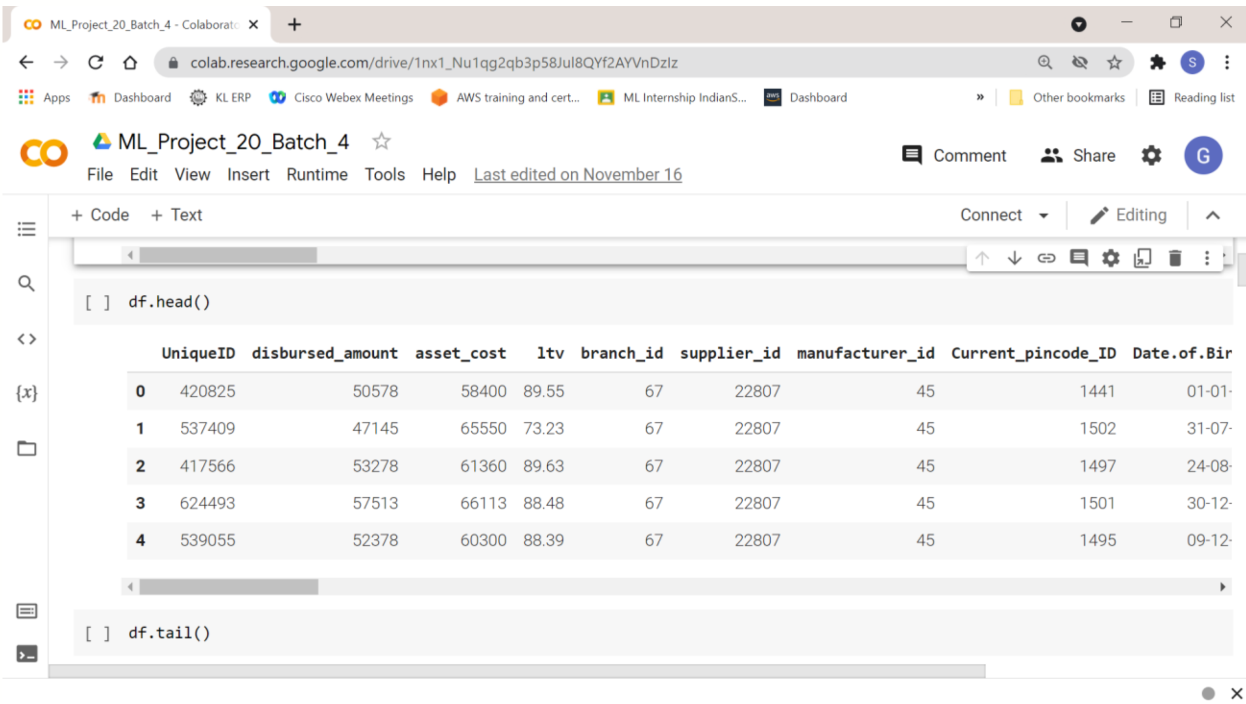
- Finally, using the information given by the classification model and the train data set, the Machine Learning Model predicts whether the applicant is worthy of the monetary loan or not.

Algorithms used in this project are:

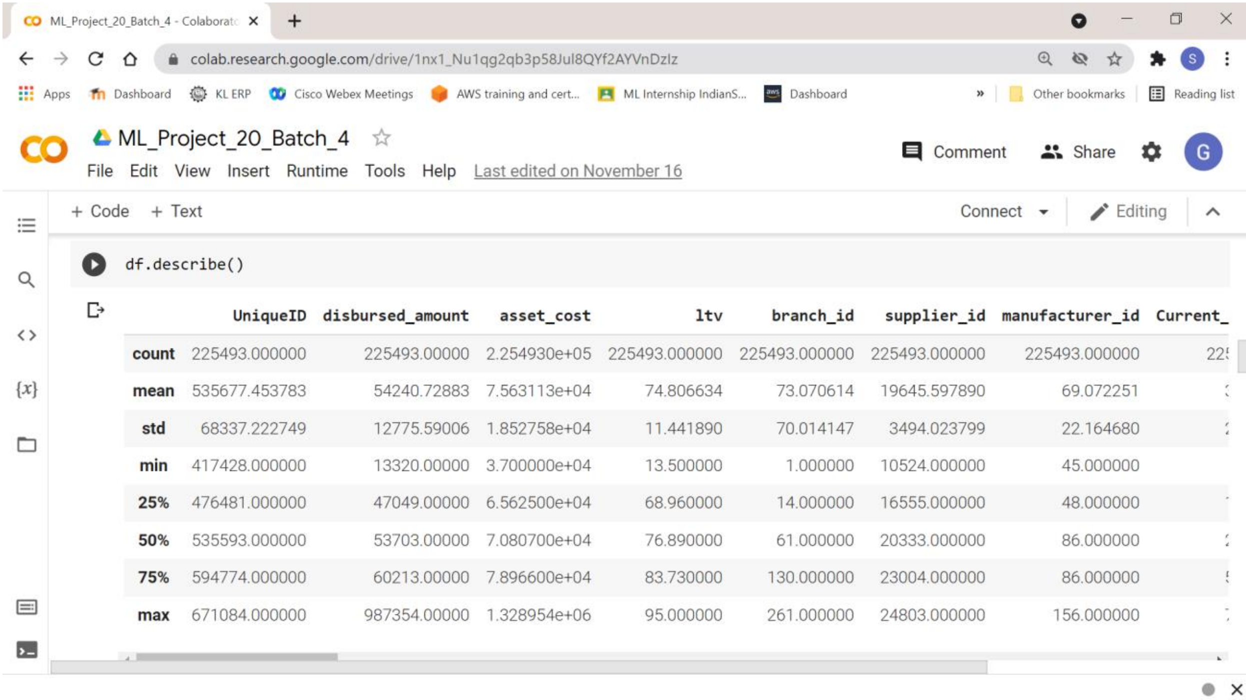
1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest Classifier
4. Naïve Bayes Classifier
5. KNN Classifier

IMPLEMENTATION

```
import numpy as np
import pandas as pd
df=pd.read_csv("/content/train.csv")
df
df.head()
```



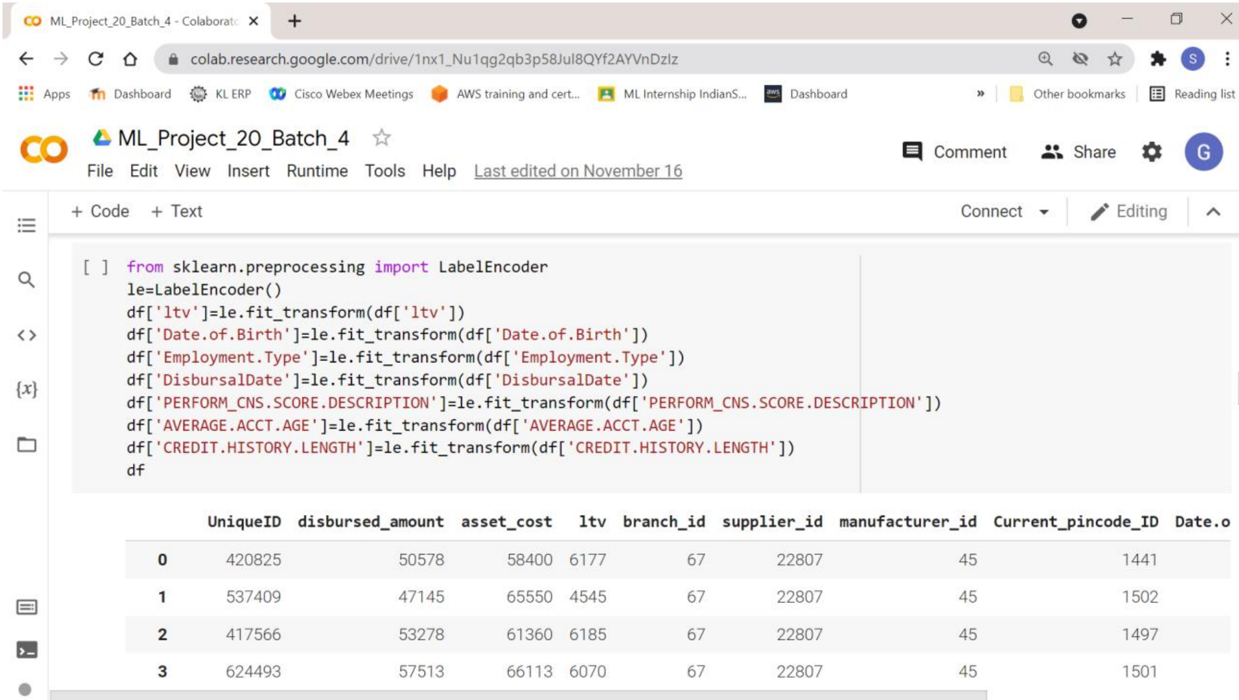

```
df.tail()
df.isnull().sum()
df.dropna(inplace=True)
df.isnull().sum()
df.info()
df.describe()
```




```
df.describe()
```

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Current_
count	225493.000000	225493.00000	2.254930e+05	225493.000000	225493.000000	225493.000000	225493.000000	22
mean	535677.453783	54240.72883	7.563113e+04	74.806634	73.070614	19645.597890	69.072251	
std	68337.222749	12775.59006	1.852758e+04	11.441890	70.014147	3494.023799	22.164680	
min	417428.000000	13320.00000	3.700000e+04	13.500000	1.000000	10524.000000	45.000000	
25%	476481.000000	47049.00000	6.562500e+04	68.960000	14.000000	16555.000000	48.000000	
50%	535593.000000	53703.00000	7.080700e+04	76.890000	61.000000	20333.000000	86.000000	
75%	594774.000000	60213.00000	7.896600e+04	83.730000	130.000000	23004.000000	86.000000	
max	671084.000000	987354.00000	1.328954e+06	95.000000	261.000000	24803.000000	156.000000	

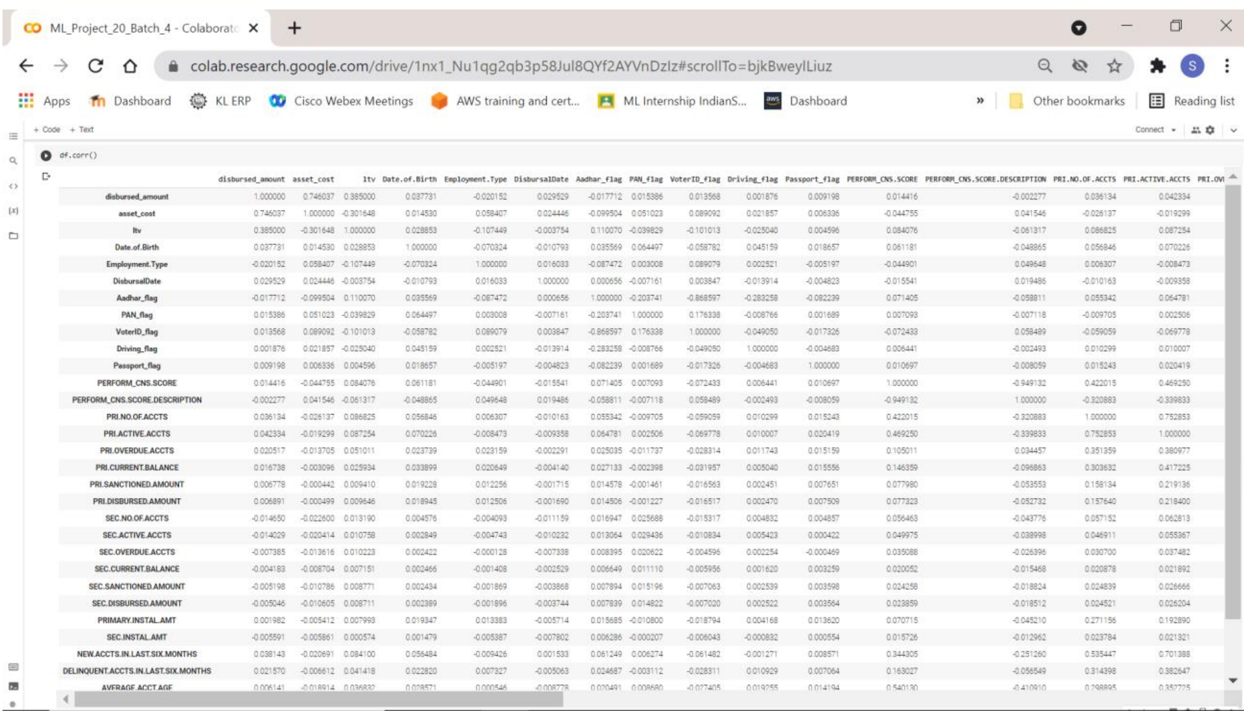
```
df.shape
df.columns
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['ltv']=le.fit_transform(df['ltv'])
df['Date.of.Birth']=le.fit_transform(df['Date.of.Birth'])
df['Employment.Type']=le.fit_transform(df['Employment.Type'])
df['DisbursalDate']=le.fit_transform(df['DisbursalDate'])
df['PERFORM_CNS.SCORE.DESCRIPTION']=le.fit_transform(df['PERFORM_CNS.SCORE
.DESCRIPTION'])
df['AVERAGE.ACCT.AGE']=le.fit_transform(df['AVERAGE.ACCT.AGE'])
df['CREDIT.HISTORY.LENGTH']=le.fit_transform(df['CREDIT.HISTORY.LENGTH'])
df
```




```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['ltv']=le.fit_transform(df['ltv'])
df['Date.of.Birth']=le.fit_transform(df['Date.of.Birth'])
df['Employment.Type']=le.fit_transform(df['Employment.Type'])
df['DisbursalDate']=le.fit_transform(df['DisbursalDate'])
df['PERFORM_CNS.SCORE.DESCRIPTION']=le.fit_transform(df['PERFORM_CNS.SCORE.DESCRIPTION'])
df['AVERAGE.ACCT.AGE']=le.fit_transform(df['AVERAGE.ACCT.AGE'])
df['CREDIT.HISTORY.LENGTH']=le.fit_transform(df['CREDIT.HISTORY.LENGTH'])
df
```

	UniqueID	disbursed_amount	asset_cost	ltv	branch_id	supplier_id	manufacturer_id	Current_pincode_ID	Date.o
0	420825	50578	58400	6177	67	22807	45	1441	
1	537409	47145	65550	4545	67	22807	45	1502	
2	417566	53278	61360	6185	67	22807	45	1497	
3	624493	57513	66113	6070	67	22807	45	1501	

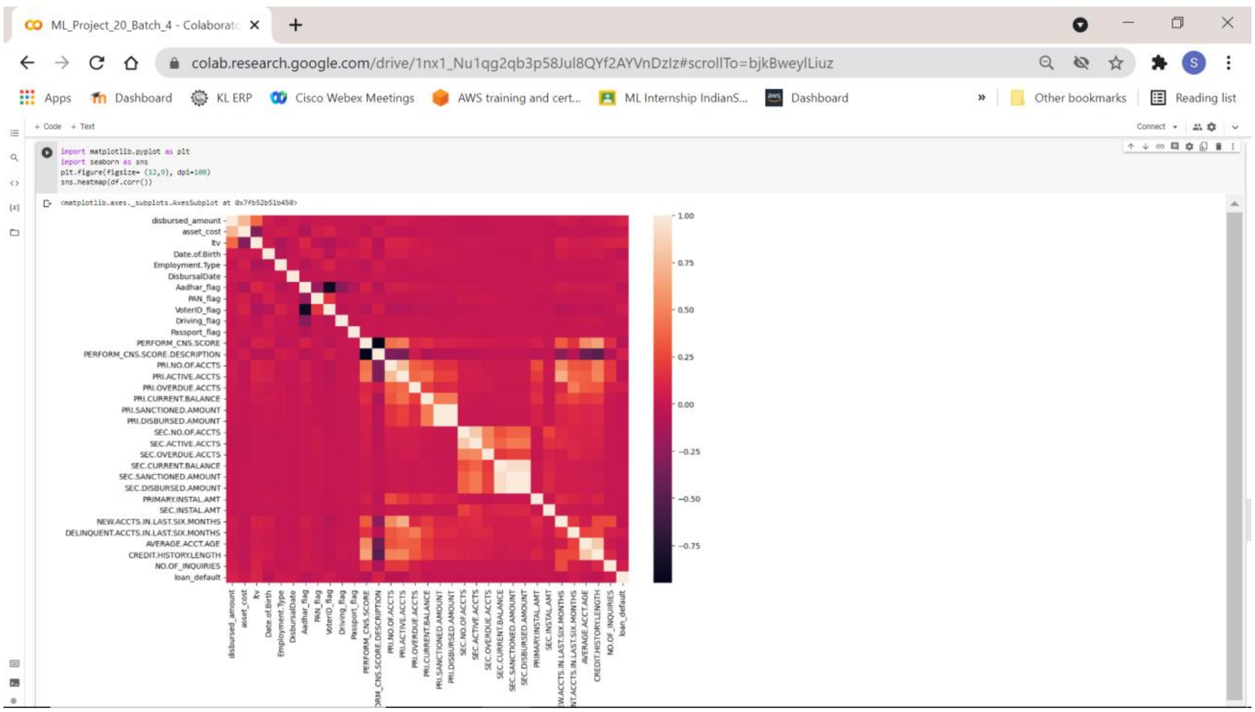
```
df.info()
df.describe()
df.corr()
```



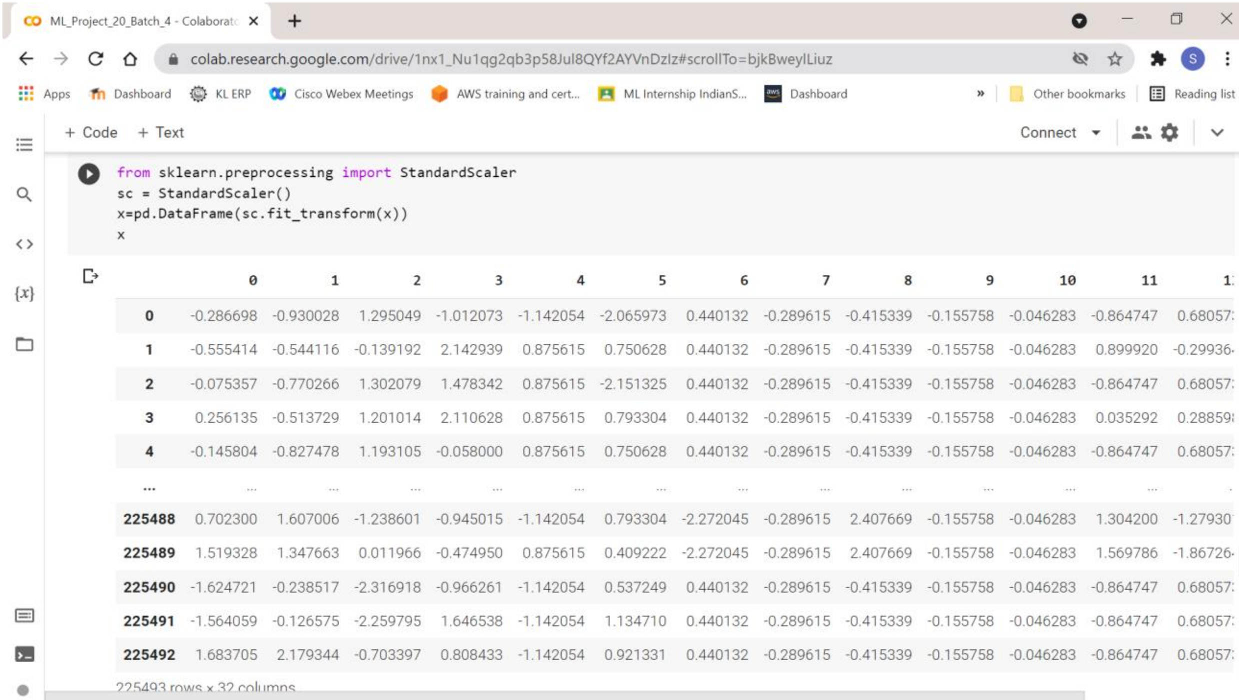

```
df.corr()
```

	disbursed_amount	asset_cost	ltv	Date.of.Birth	Employment.Type	DisbursalDate	Aadhar_flag	PAN_flag	VoterID_flag	Driving_flag	Passport_flag	PERFORM_CNS.SCORE	PERFORM_CNS.SCORE.DESCRIPTION	PRI.NO.OF.ACCTS	PRI.ACTIVE.ACCTS	PRI.OV
disbursed_amount	1.000000	0.746037	0.385000	0.037731	-0.020152	0.029529	-0.017712	0.015386	0.013568	0.001876	0.009198	0.014416	-0.002277	0.036134	0.042334	
asset_cost	0.746037	1.000000	-0.301648	0.014530	0.058407	0.024446	-0.099504	0.051023	0.089092	0.021857	0.006336	-0.044755	0.041546	-0.026137	-0.019299	
ltv	0.385000	-0.301648	1.000000	0.028853	-0.107449	-0.003754	0.110070	-0.039829	-0.101013	-0.025040	0.004596	0.084076	-0.061317	0.086825	0.087254	
Date.of.Birth	0.037731	0.014530	0.028853	1.000000	-0.070324	-0.010793	0.035569	0.064497	-0.058782	0.045159	0.018657	0.061181	-0.048865	0.056846	0.070226	
Employment.Type	-0.020152	0.058407	-0.107449	-0.070324	1.000000	0.016033	-0.087472	0.003008	0.089079	0.002521	-0.005197	-0.044901	0.049648	0.006307	-0.008473	
DisbursalDate	0.029529	0.024446	-0.003754	-0.010793	0.016033	1.000000	0.000656	-0.007161	0.003847	-0.013914	-0.004823	-0.015541	0.019486	-0.010163	-0.009358	
Aadhar_flag	-0.017712	-0.099504	0.110070	0.035569	-0.087472	0.000656	1.000000	-0.203741	-0.868597	-0.283258	-0.082239	0.071405	-0.058811	0.055342	0.064781	
PAN_flag	0.015386	0.051023	-0.039829	0.064497	0.003008	-0.007161	-0.203741	1.000000	0.176338	-0.008766	0.001689	0.007093	-0.007118	-0.009705	0.002506	
VoterID_flag	0.013568	0.089092	-0.101013	-0.058782	0.089079	0.003847	-0.868597	0.176338	1.000000	-0.049050	-0.017326	-0.072433	0.058489	-0.059059	-0.069778	
Driving_flag	0.001876	0.021857	-0.025040	0.045159	0.002521	-0.013914	-0.283258	-0.008766	-0.049050	1.000000	-0.004683	0.006441	-0.002493	0.010299	0.010007	
Passport_flag	0.009198	0.006336	0.004596	0.018657	-0.005197	-0.004823	-0.082239	0.001689	-0.017326	-0.004683	1.000000	0.010697	-0.008059	0.015243	0.020419	
PERFORM_CNS.SCORE	0.014416	-0.044755	0.084076	0.061181	-0.044901	-0.015541	0.071405	0.007093	-0.072433	0.006441	0.010697	1.000000	-0.949132	0.422015	0.469250	
PERFORM_CNS.SCORE.DESCRIPTION	-0.002277	0.041546	-0.061317	-0.048865	0.049648	0.019486	-0.058811	-0.007118	0.058489	-0.002493	-0.008059	-0.949132	1.000000	-0.320883	-0.339833	
PRI.NO.OF.ACCTS	0.036134	-0.026137	0.086825	0.056846	0.006307	-0.010163	0.055342	-0.009705	-0.059059	0.010299	0.015243	0.422015	-0.320883	1.000000	0.752853	
PRI.ACTIVE.ACCTS	0.042334	-0.019299	0.087254	0.070226	-0.008473	-0.009358	0.064781	0.002506	-0.069778	0.010007	0.020419	0.469250	-0.339833	0.752853	1.000000	
PRI.OVERDUE.ACCTS	0.020517	-0.013705	0.051011	0.023739	0.023159	-0.002291	0.025035	-0.011737	-0.028314	0.011743	0.015159	0.105011	0.034457	0.351359	0.380977	
PRI.CURRENT.BALANCE	0.016738	-0.003096	0.025934	0.033899	0.020649	-0.004140	0.027133	-0.002398	-0.031957	0.005040	0.015556	0.146359	-0.096863	0.303632	0.417225	
PRI.SANCTIONED.AMOUNT	0.006778	-0.000442	0.009410	0.019228	0.012256	-0.001715	0.014578	-0.001461	-0.016563	0.002451	0.007651	0.077980	-0.053553	0.158134	0.219136	
PRI.DISBURSED.AMOUNT	0.006891	-0.000499	0.009646	0.018945	0.012506	-0.001690	0.014506	-0.001227	-0.016517	0.002470	0.007509	0.077323	-0.052732	0.157640	0.218400	
SEC.NO.OF.ACCTS	-0.014650	-0.022600	0.013190	0.004576	-0.004093	-0.011159	0.016947	0.025688	-0.015317	0.004832	0.004857	0.056463	-0.043776	0.057152	0.062813	
SEC.ACTIVE.ACCTS	-0.014029	-0.020414	0.010758	0.002849	-0.004743	-0.010232	0.013064	0.029436	-0.010834	0.005423	0.000422	0.049975	-0.038998	0.046911	0.055367	
SEC.OVERDUE.ACCTS	-0.007385	-0.013616	0.010223	0.002422	-0.000128	-0.007338	0.008395	0.020622	-0.004596	0.002254	-0.000469	0.035088	-0.026396	0.030700	0.037482	
SEC.CURRENT.BALANCE	-0.004183	-0.008704	0.007151	0.002466	-0.001408	-0.002529	0.006649	0.011110	-0.005956	0.001620	0.003259	0.020052	-0.015468	0.020878	0.021892	
SEC.SANCTIONED.AMOUNT	-0.005198	-0.010786	0.008771	0.002434	-0.001869	-0.003868	0.007894	0.015196	-0.007063	0.002539	0.003598	0.024258	-0.018824	0.024839	0.026666	
SEC.DISBURSED.AMOUNT	-0.005046	-0.010605	0.008711	0.002399	-0.001896	-0.003744	0.007839	0.014822	-0.007020	0.002522	0.003564	0.023859	-0.018512	0.024521	0.026204	
PRIMARY.INSTAL.AMT	0.001982	-0.005412	0.007993	0.019347	0.013383	-0.005714	0.015685	-0.010800	-0.018794	0.004168	0.013620	0.070715	-0.045210	0.271156	0.192890	
SEC.INSTAL.AMT	-0.005591	-0.005861	0.000574	0.001479	-0.005387	-0.007802	0.006286	-0.000207	-0.006043	-0.000832	0.000554	0.015726	-0.012962	0.023784	0.021321	
NEW.ACCTS.IN.LAST.SIX.MONTHS	0.038143	-0.020691	0.084100	0.056484	-0.009426	0.001533	0.061249	0.006274	-0.061482	-0.001271	0.008571	0.344305	-0.251260	0.535447	0.701388	
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	0.021570	-0.006612	0.041418	0.022820	0.007327	-0.005063	0.024687	-0.003112	-0.028311	0.010929	0.007064	0.163027	-0.056549	0.314398	0.382647	
AVERAGE.ACCT.AGE	0.006141	-0.018914	0.036832	0.028571	0.000546	-0.008778	0.020491	0.008680	-0.027405	0.019255	0.014194	0.540130	-0.410910	0.298895	0.352725	

```
df.drop(['UniqueID','branch_id','supplier_id','manufacturer_id','MobileNo_
Avl_Flag','State_ID','Employee_code_ID','Current_pincode_ID'],axis=1,inpla
ce=True)
df
df.corr()
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize= (12,9), dpi=100)
sns.heatmap(df.corr())
```




```
x=df.drop('loan_default',axis=1)
x
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```



```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x=pd.DataFrame(sc.fit_transform(x))
x
```

	0	1	2	3	4	5	6	7	8	9	10	11	1:
0	-0.286698	-0.930028	1.295049	-1.012073	-1.142054	-2.065973	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	-0.864747	0.68057:
1	-0.555414	-0.544116	-0.139192	2.142939	0.875615	0.750628	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	0.899920	-0.29936
2	-0.075357	-0.770266	1.302079	1.478342	0.875615	-2.151325	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	-0.864747	0.68057:
3	0.256135	-0.513729	1.201014	2.110628	0.875615	0.793304	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	0.035292	0.28859
4	-0.145804	-0.827478	1.193105	-0.058000	0.875615	0.750628	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	-0.864747	0.68057:
...	...	...	...	...	...	...	...	...	...	...	...	...	.
225488	0.702300	1.607006	-1.238601	-0.945015	-1.142054	0.793304	-2.272045	-0.289615	2.407669	-0.155758	-0.046283	1.304200	-1.27930
225489	1.519328	1.347663	0.011966	-0.474950	0.875615	0.409222	-2.272045	-0.289615	2.407669	-0.155758	-0.046283	1.569786	-1.86726
225490	-1.624721	-0.238517	-2.316918	-0.966261	-1.142054	0.537249	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	-0.864747	0.68057:
225491	-1.564059	-0.126575	-2.259795	1.646538	-1.142054	1.134710	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	-0.864747	0.68057:
225492	1.683705	2.179344	-0.703397	0.808433	-1.142054	0.921331	0.440132	-0.289615	-0.415339	-0.155758	-0.046283	-0.864747	0.68057:

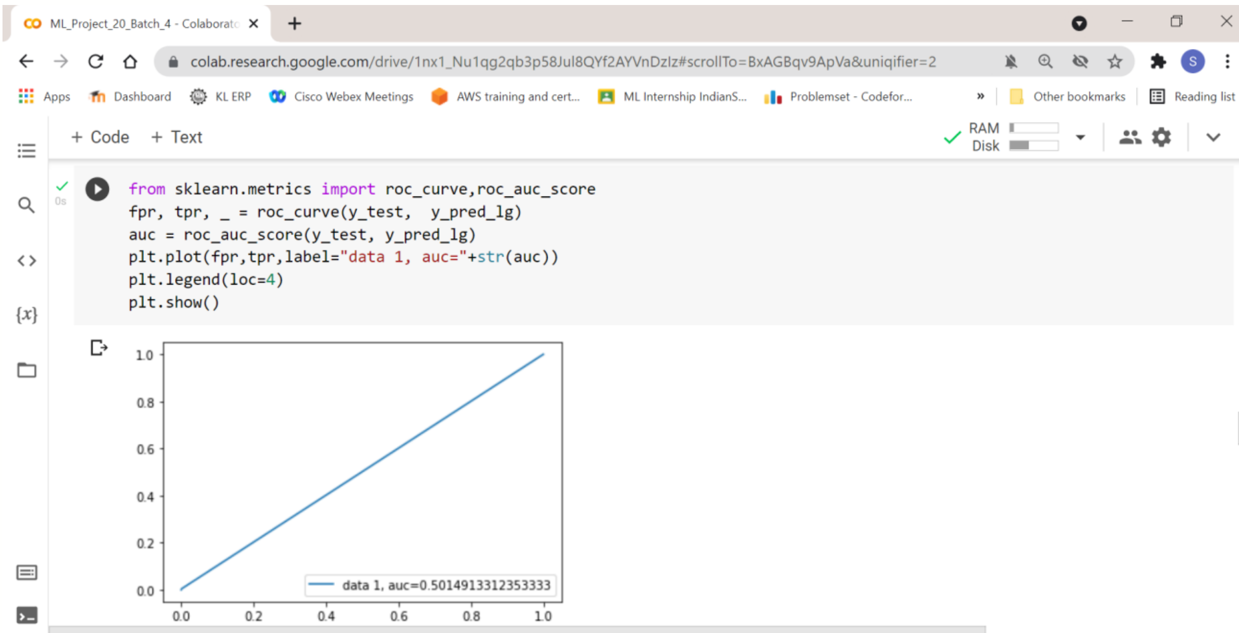
225493 rows × 32 columns

```
x=pd.DataFrame(sc.fit_transform(x))
x
y=df['loan_default']
y
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=42)
x_train
x_test
y_train
y_test
```

Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(x_train,y_train)
y_pred_lg=logistic_regression_model.predict(x_test)
logistic_regression_df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred_lg
})
logistic_regression_df
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred_lg )
accuracy_percentage = 100 * accuracy
accuracy_percentage
from sklearn.metrics import roc_curve,roc_auc_score
fpr, tpr, _ = roc_curve(y_test,  y_pred_lg)
```

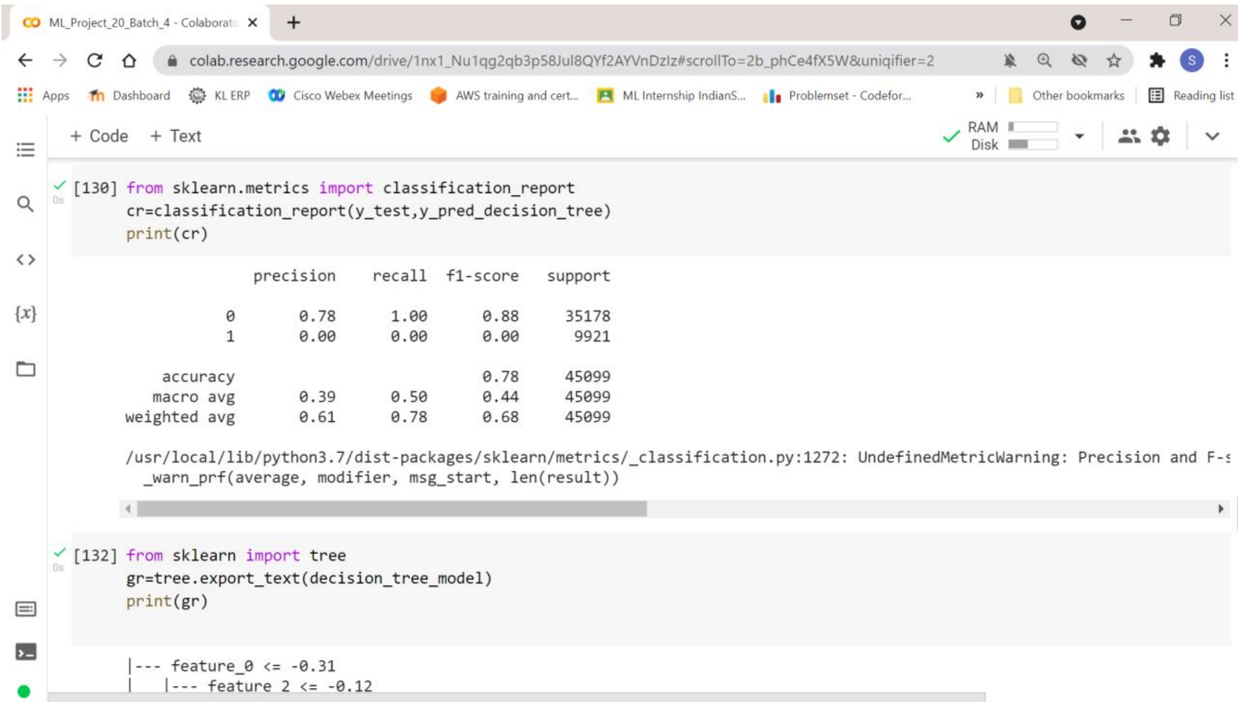
```
auc = roc_auc_score(y_test, y_pred_lg)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

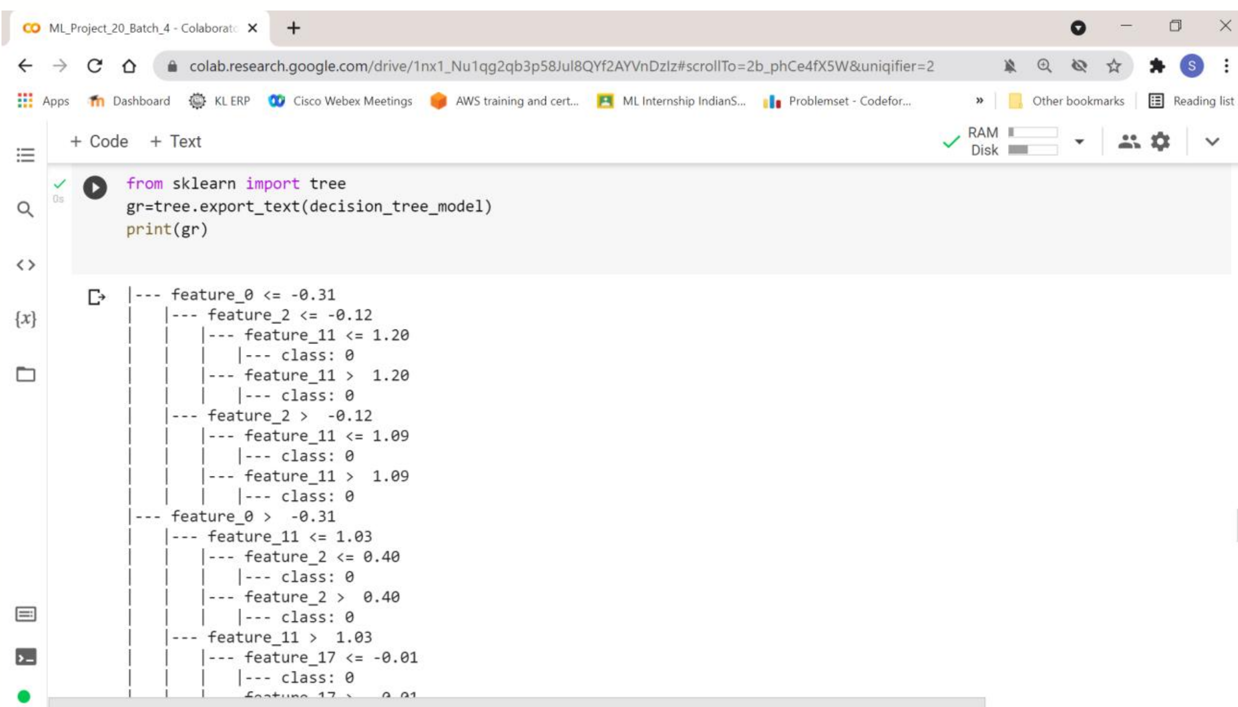


Decision Tree Classifier:

```
from sklearn.tree import DecisionTreeClassifier
decision_tree_model=DecisionTreeClassifier(criterion='gini',max_depth=3)
decision_tree_model.fit(x_train,y_train)
y_pred_decision_tree=decision_tree_model.predict(x_test)
decision_tree_df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred_decision
_tree})
decision_tree_df
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred_decision_tree)
pd.DataFrame(cm)
accuracy = accuracy_score(y_test,y_pred_decision_tree )
accuracy_percentage = 100 * accuracy
accuracy_percentage
from sklearn.metrics import classification_report
cr=classification_report(y_test,y_pred_decision_tree)
print(cr)
from sklearn import tree
gr=tree.export_text(decision_tree_model)
print(gr)
a=pd.DataFrame(x).columns.values
a
```

```python
b=pd.DataFrame(y).columns.values
b
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(25,20))
tree.plot_tree(decision_tree_model,feature_names=a,class_names=b,filled=True)
```





ML_Project_20_Batch_4 - Colaborat

colab.research.google.com/drive/1nx1_Nu1qg2qb3p58Jul8QYf2AYVnDzlz#scrollTo=2b_phCe4fX5W&uniqifier=2

Apps | Dashboard | KL ERP | Cisco Webex Meetings | AWS training and cert... | ML Internship IndianS... | Problemset - Codefor... | Other bookmarks | Reading list

+ Code + Text

```
|--- feature_0 <= -0.31
|   |--- feature_2 <= -0.12
|   |   |--- feature_11 <= 1.20
|   |   |   |--- class: 0
|   |   |--- feature_11 >  1.20
|   |   |   |--- class: 0
|   |--- feature_2 >  -0.12
|   |   |--- feature_11 <= 1.09
|   |   |   |--- class: 0
|   |   |--- feature_11 >  1.09
|   |   |   |--- class: 0
|--- feature_0 >  -0.31
|   |--- feature_11 <= 1.03
|   |   |--- feature_2 <= 0.40
|   |   |   |--- class: 0
|   |   |--- feature_2 >  0.40
|   |   |   |--- class: 0
|   |--- feature_11 >  1.03
|   |   |--- feature_17 <= -0.01
|   |   |   |--- class: 0
|   |   |--- feature_17 >  -0.01
|   |   |   |--- class: 0
```

ML_Project_20_Batch_4 - Colaborat

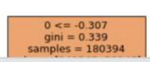
colab.research.google.com/drive/1nx1_Nu1qg2qb3p58Jul8QYf2AYVnDzlz#scrollTo=2b_phCe4fX5W&uniqifier=2

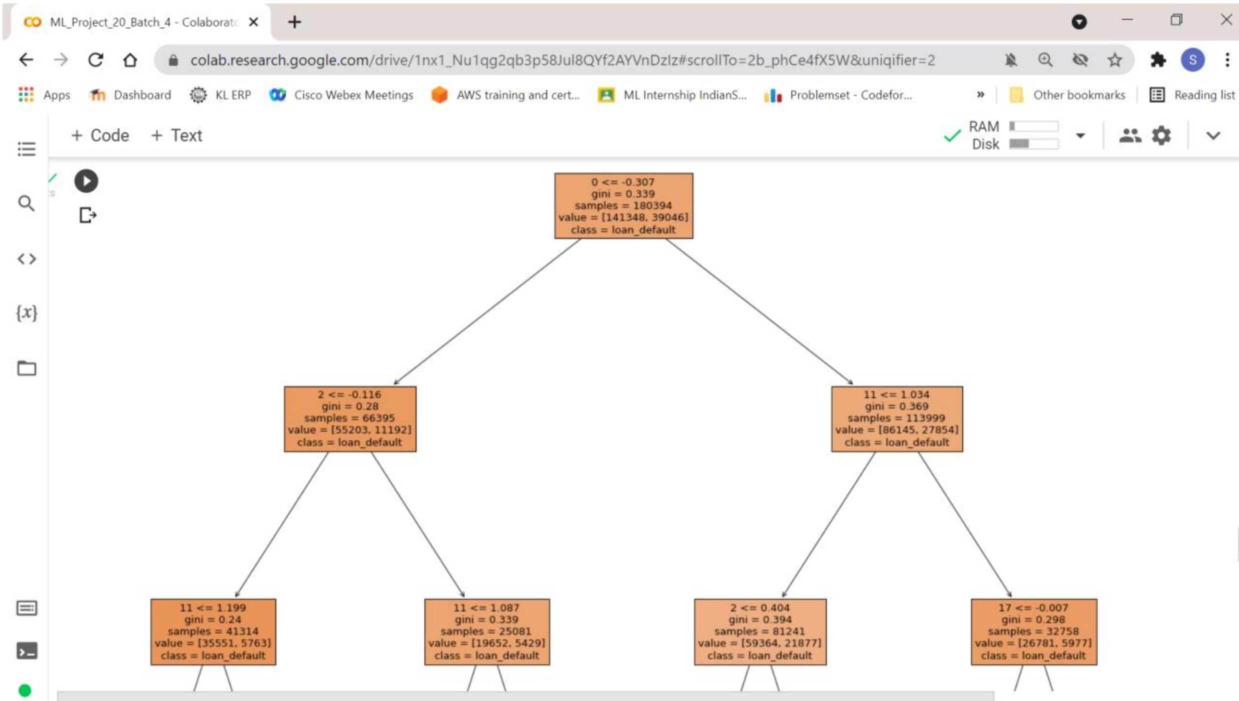
Apps | Dashboard | KL ERP | Cisco Webex Meetings | AWS training and cert... | ML Internship IndianS... | Problemset - Codefor... | Other bookmarks | Reading list

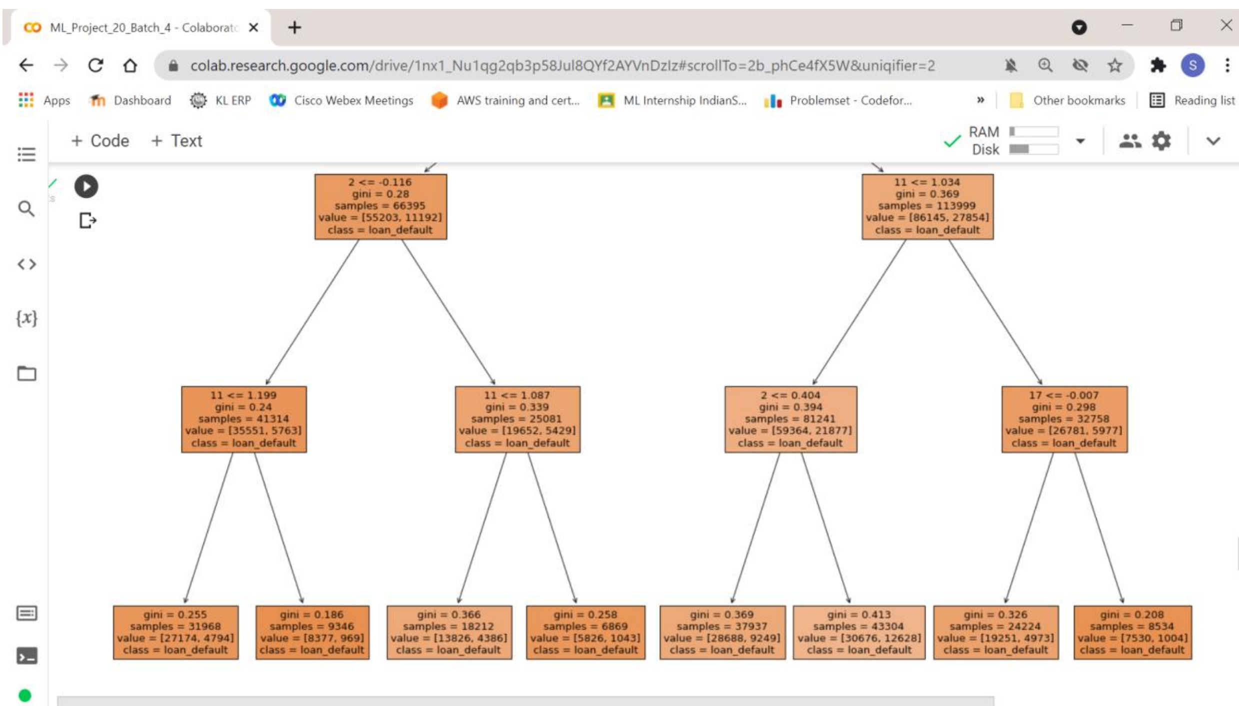
+ Code + Text

```python
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(25,20))
tree.plot_tree(decision_tree_model,feature_names=a,class_names=b,filled=True)
```

```
[Text(697.5, 951.3000000000001, '0 <= -0.307\ngini = 0.339\nsamples = 180394\nvalue = [141348, 39046]\nclass = loan_defau
 Text(348.75, 679.5, '2 <= -0.116\ngini = 0.28\nsamples = 66395\nvalue = [55203, 11192]\nclass = loan_default'),
 Text(174.375, 407.70000000000005, '11 <= 1.199\ngini = 0.24\nsamples = 41314\nvalue = [35551, 5763]\nclass = loan_defaul
 Text(87.1875, 135.89999999999998, 'gini = 0.255\nsamples = 31968\nvalue = [27174, 4794]\nclass = loan_default'),
 Text(261.5625, 135.89999999999998, 'gini = 0.186\nsamples = 9346\nvalue = [8377, 969]\nclass = loan_default'),
 Text(523.125, 407.70000000000005, '11 <= 1.087\ngini = 0.339\nsamples = 25081\nvalue = [19652, 5429]\nclass = loan_defau
 Text(435.9375, 135.89999999999998, 'gini = 0.366\nsamples = 18212\nvalue = [13826, 4386]\nclass = loan_default'),
 Text(610.3125, 135.89999999999998, 'gini = 0.258\nsamples = 6869\nvalue = [5826, 1043]\nclass = loan_default'),
 Text(1046.25, 679.5, '11 <= 1.034\ngini = 0.369\nsamples = 113999\nvalue = [86145, 27854]\nclass = loan_default'),
 Text(871.875, 407.70000000000005, '2 <= 0.404\ngini = 0.394\nsamples = 81241\nvalue = [59364, 21877]\nclass = loan_defau
 Text(784.6875, 135.89999999999998, 'gini = 0.369\nsamples = 37937\nvalue = [28688, 9249]\nclass = loan_default'),
 Text(959.0625, 135.89999999999998, 'gini = 0.413\nsamples = 43304\nvalue = [30676, 12628]\nclass = loan_default'),
 Text(1220.625, 407.70000000000005, '17 <= -0.007\ngini = 0.298\nsamples = 32758\nvalue = [26781, 5977]\nclass = loan_def
 Text(1133.4375, 135.89999999999998, 'gini = 0.326\nsamples = 24224\nvalue = [19251, 4973]\nclass = loan_default'),
 Text(1307.8125, 135.89999999999998, 'gini = 0.208\nsamples = 8534\nvalue = [7530, 1004]\nclass = loan_default')]
```



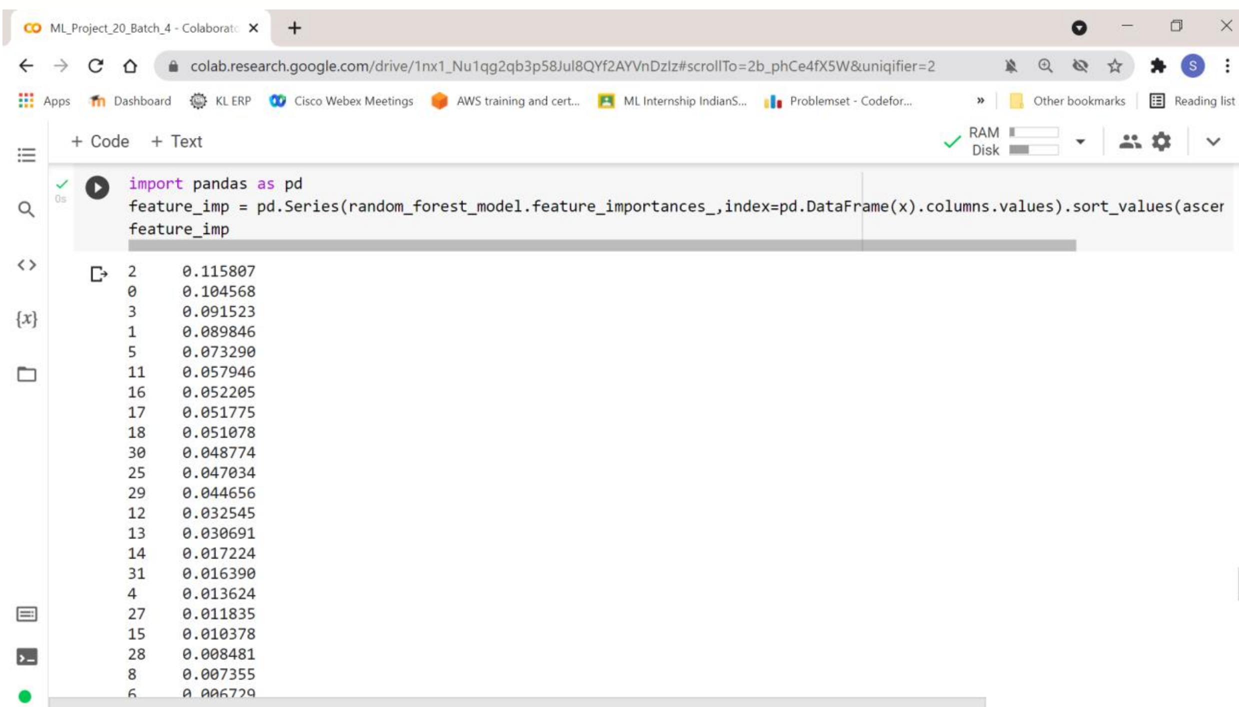


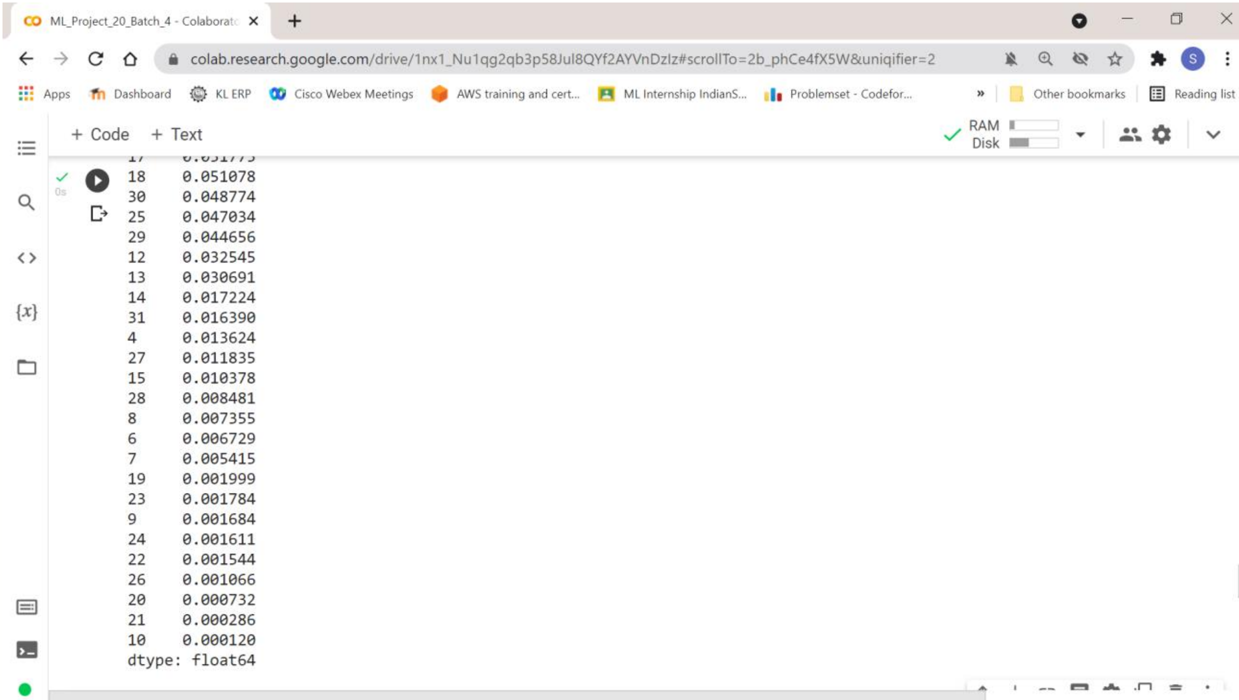


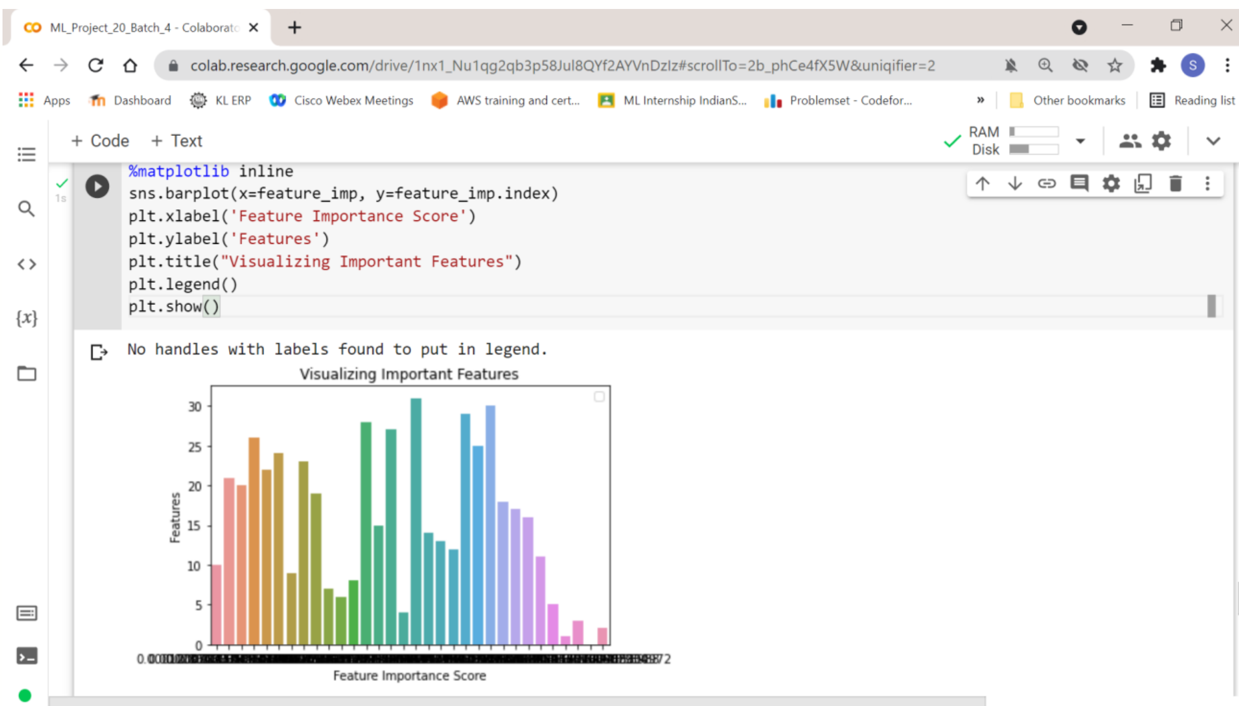
Random Forest Classifier:

```
from sklearn.ensemble import RandomForestClassifier
random_forest_model=RandomForestClassifier(criterion='entropy',max_depth=
20,min_samples_leaf=5)
random_forest_model.fit(x_train,y_train)
```

```
y_pred_random_forest=random_forest_model.predict(x_test)
random_forest_df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred_random_f
orest})
random_forest_df
accuracy = accuracy_score(y_test,y_pred_random_forest )
accuracy_percentage = 100 * accuracy
accuracy_percentage
import pandas as pd
feature_imp = pd.Series(random_forest_model.feature_importances_,index=pd.
DataFrame(x).columns.values).sort_values(ascending=False)
feature_imp
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```



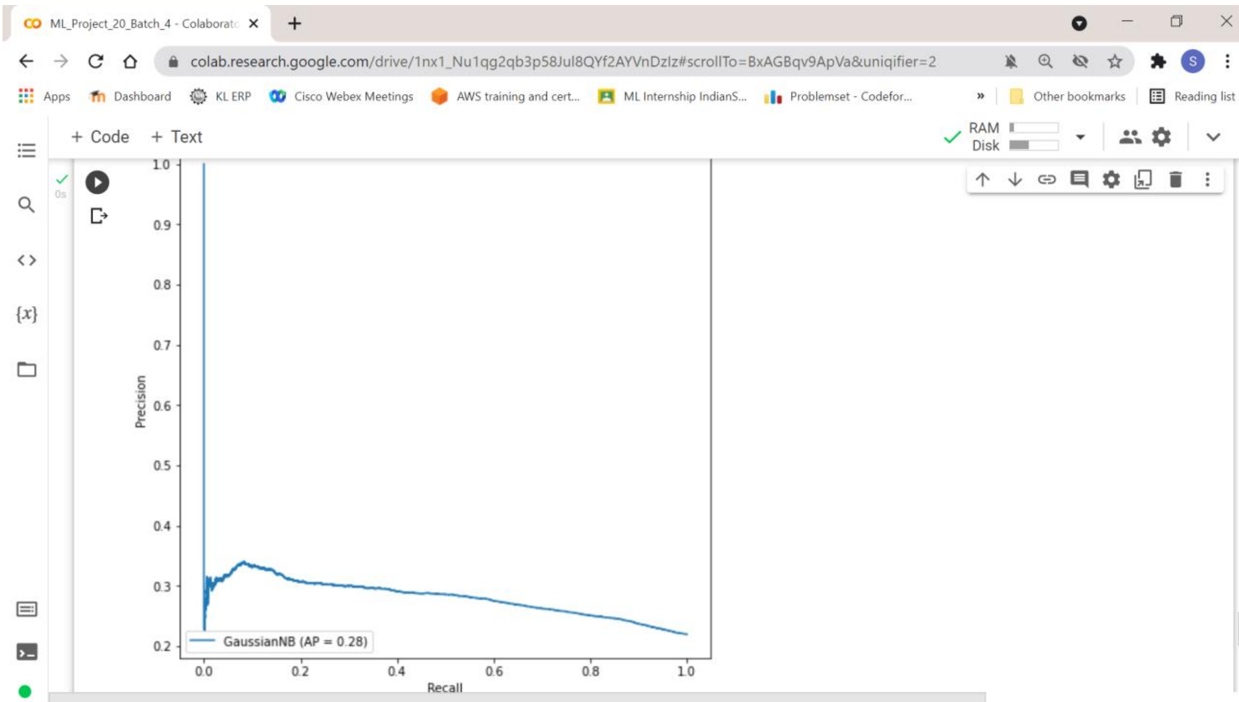




Naïve Bayes Classifier:

```
from sklearn.naive_bayes import GaussianNB
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(x_train,y_train)
y_pred_nb=naive_bayes_model.predict(x_test)
naive_bayes_df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred_nb})
```

```
naive_bayes_df
accuracy = accuracy_score(y_test,y_pred_nb )
accuracy_percentage = 100 * accuracy
accuracy_percentage
from sklearn import metrics
fig, ax = plt.subplots(1,1, figsize=(8,8))
metrics.plot_precision_recall_curve(naive_bayes_model, x_test, y_test, ax=
ax)
```

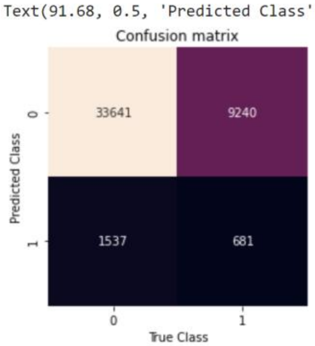


KNN Classifier:

```
from sklearn.neighbors import KNeighborsClassifier
knn_model=KNeighborsClassifier(n_neighbors=4)
knn_model.fit(x_train, y_train)
y_pred_knn = knn_model.predict(x_test)
knn_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_knn})
knn_df
accuracy = accuracy_score(y_test,y_pred_knn )
accuracy_percentage = 100 * accuracy
accuracy_percentage
from sklearn.metrics import confusion_matrix
mat=confusion_matrix(y_test,y_pred_knn)
sns.heatmap(mat.T,square=True,annot=True,fmt='d',cbar=False)
plt.title('Confusion matrix')
plt.xlabel('True Class')
plt.ylabel('Predicted Class')
```

```python
from sklearn.metrics import confusion_matrix
mat=confusion_matrix(y_test,y_pred_knn)
sns.heatmap(mat.T,square=True,annot=True,fmt='d',cbar=False)
plt.title('Confusion matrix')
plt.xlabel('True Class')
plt.ylabel('Predicted Class')
```

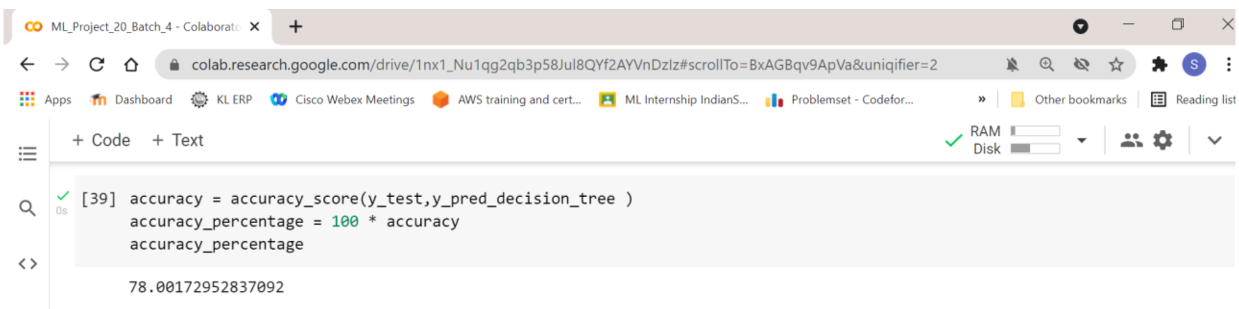
```
Text(91.68, 0.5, 'Predicted Class')
```



RESULTS

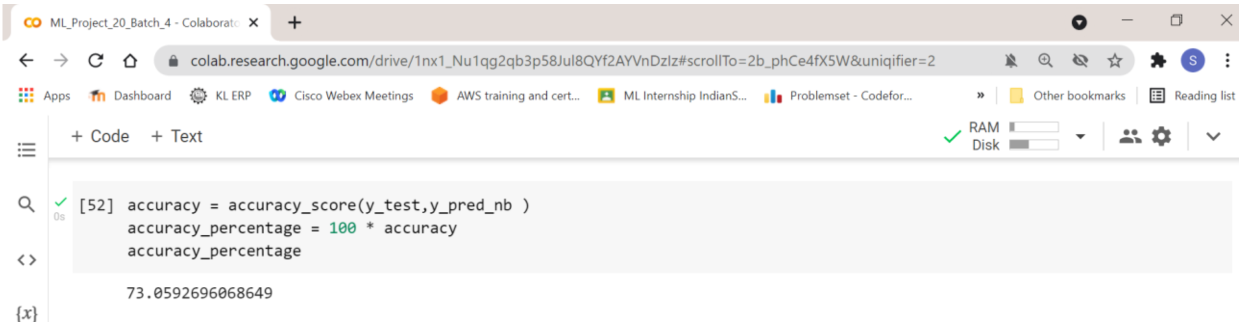
Logistic Regression:

Decision Tree Classifier:

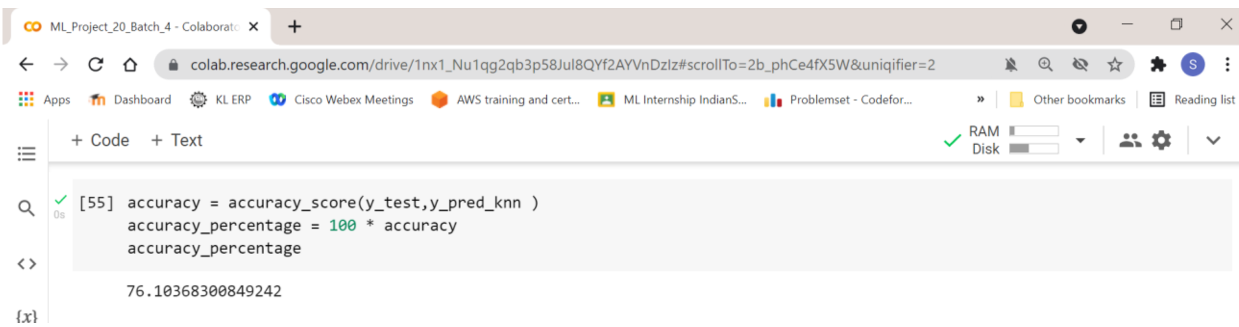


Random Forest Classifier:

Naïve Bayes Classifier:



KNN Classifier:



Model	Accuracy
Logistic Regression	77.9973%
Decision Tree Classifier	78.0017%
Random Forest Classifier	78.0527%
Naïve Bayes Classifier	73.0527%
KNN Classifier	76.0592%

Random Forest Classifier is found to be the best fitted model with our data obtaining an accuracy of 78.0527%.

CONCLUSION

Thus, we notice that using Credit history length is not only the accurate measurement of a borrower's credibility. Many more parameters must be taken into consideration. But this process, done manually, is time taking and inefficient. Our project gives a solution by constructing Machine Learning based models. It is inclusive of all the parameters needed to evaluate the creditworthiness of a client. The model is trained to produce results with satisfactory accuracy, after which it produces accurate results as to whether a borrower should be lent money or not without any tedious manual work.