PREDICTING PERSONAL LOAN USING MACHINE LEARNING

1.INTRODUCTION

1.1 Overview:

PREDICTING PERSONAL LOAN USING MACHINE LEARNING

PROJECT DESCRIPTION

PREDICTING PERSONAL LOAN USING MACHINE LEARNING INVOLVES USING ALGORITHMS AND STATISTICAL MODELS TO ANALYZE VARIOUS DATA POINTS SUCH AS CREDIT SCORE, INCOME, EMPLOYMENT DETAILS, AND OTHER PERSONAL INFORMATION OF LOAN APPLICANTS TO ESTIMATE THE PROBABILITY OF APPROVAL FOR A PERSONAL LOAN. THE MACHINE LEARNING MODEL ANALYZES THE HISTORICAL DATA OF LOAN APPROVALS AND REJECTIONS TO FIND PATTERNS AND TRENDS THAT CAN BE USED TO MAKE PREDICTIONS. THE MODEL LEARNS THE RELATIONSHIP BETWEEN VARIOUS FACTORS AND THE LOAN APPROVAL PROCESS BY USING A RANGE OF TECHNIQUES SUCH AS LOGISTIC REGRESSION, DECISION TREES, NEURAL NETWORKS, AND SUPPORT VECTOR MACHINES. THE MODEL THEN CATEGORIZES LOAN APPLICATIONS INTO "APPROVED" OR "REJECTED" BASED ON THE LIKELIHOOD OF

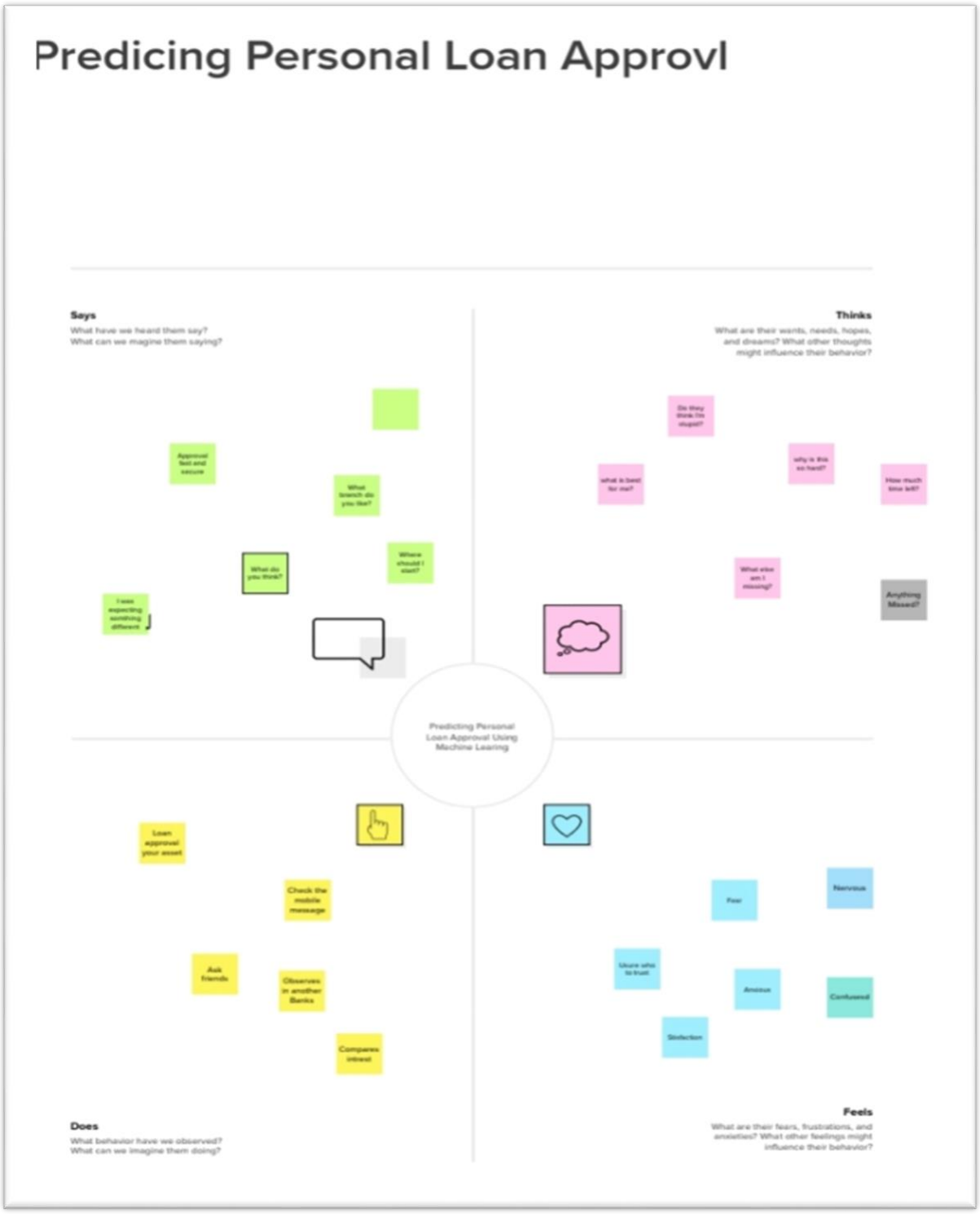
APPROVAL, AND THE LENDER USES THIS INFORMATION TO MAKE DECISIONS. THE USE OF MACHINE LEARNING IN PERSONAL LOAN PREDICTION HAS IMMENSE BENEFITS, INCLUDING INCREASED EFFICIENCY, PRODUCTIVITY, AND ACCURACY, WHICH ULTIMATELY LEAD TO BETTER LENDING DECISIONS.

1.2 PURPOSE

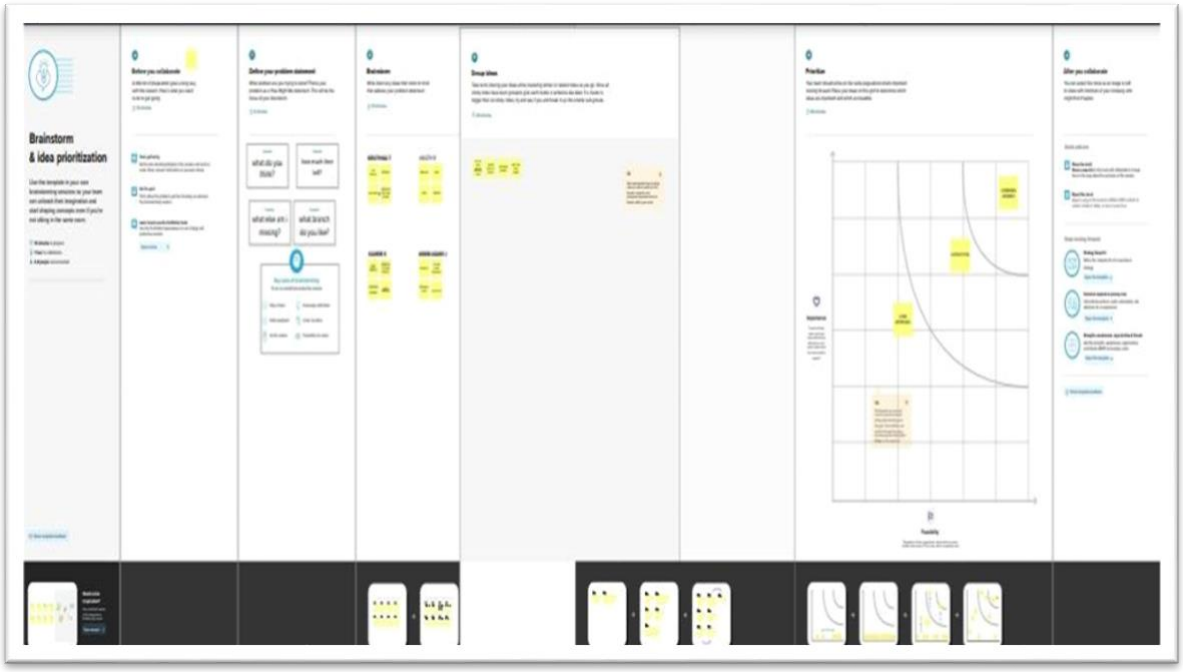
THE PURPOSE OF PREDICTING PERSONAL LOAN USING MACHINE LEARNING IS TO ACCURATELY IDENTIFY INDIVIDUALS WHO ARE MORE LIKELY TO APPLY AND OBTAIN A PERSONAL LOAN, BASED ON THEIR PAST BEHAVIORS AND CHARACTERISTICS. BY ANALYZING HISTORICAL DATA AND IDENTIFYING PATTERNS AND CORRELATIONS, MACHINE LEARNING MODELS CAN MAKE PREDICTIONS ABOUT WHICH INDIVIDUALS ARE MORE LIKELY TO APPLY AND BE APPROVED FOR LOANS, WHICH CAN PROVIDE VALUABLE INSIGHTS FOR LENDERS AND FINANCIAL INSTITUTIONS. THIS CAN ALSO HELP TO STREAMLINE THE LOAN APPLICATION PROCESS, REDUCE DEFAULTS, AND ULTIMATELY IMPROVE THE OVERALL DECISION-MAKING PROCESS IN THE LENDING INDUSTRY.

2.PROBLEM DEFINITION & DESING THINKING

2.1 EMPATHY MAP

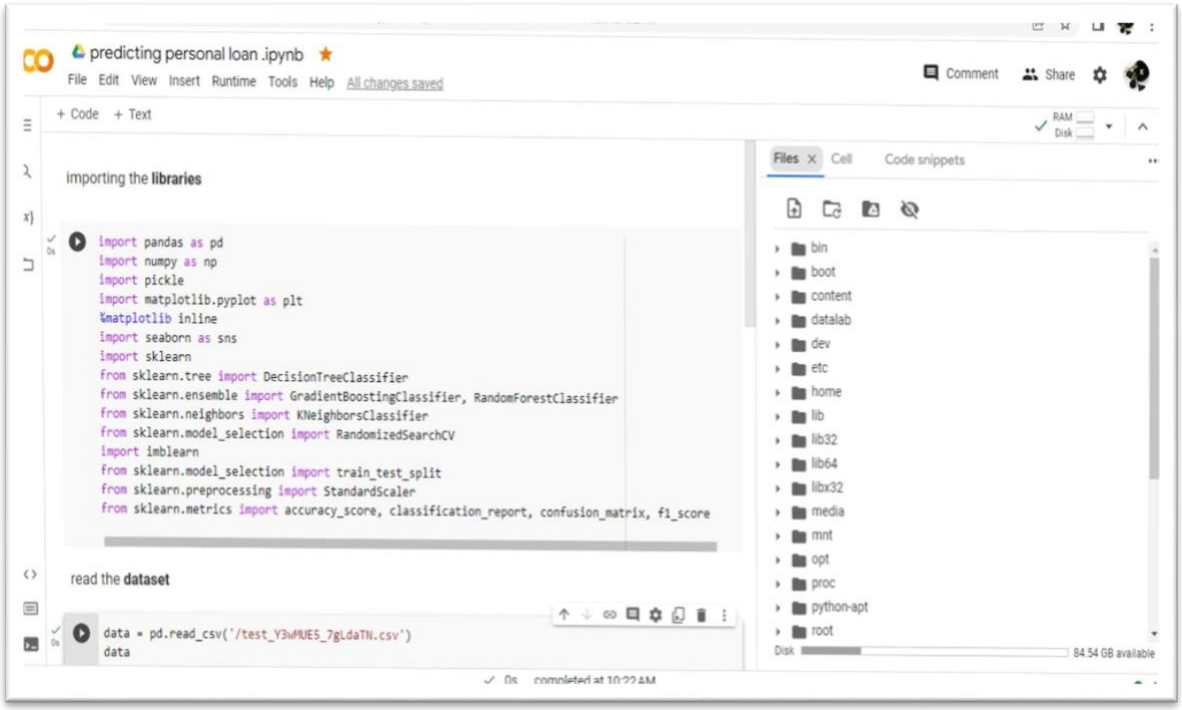


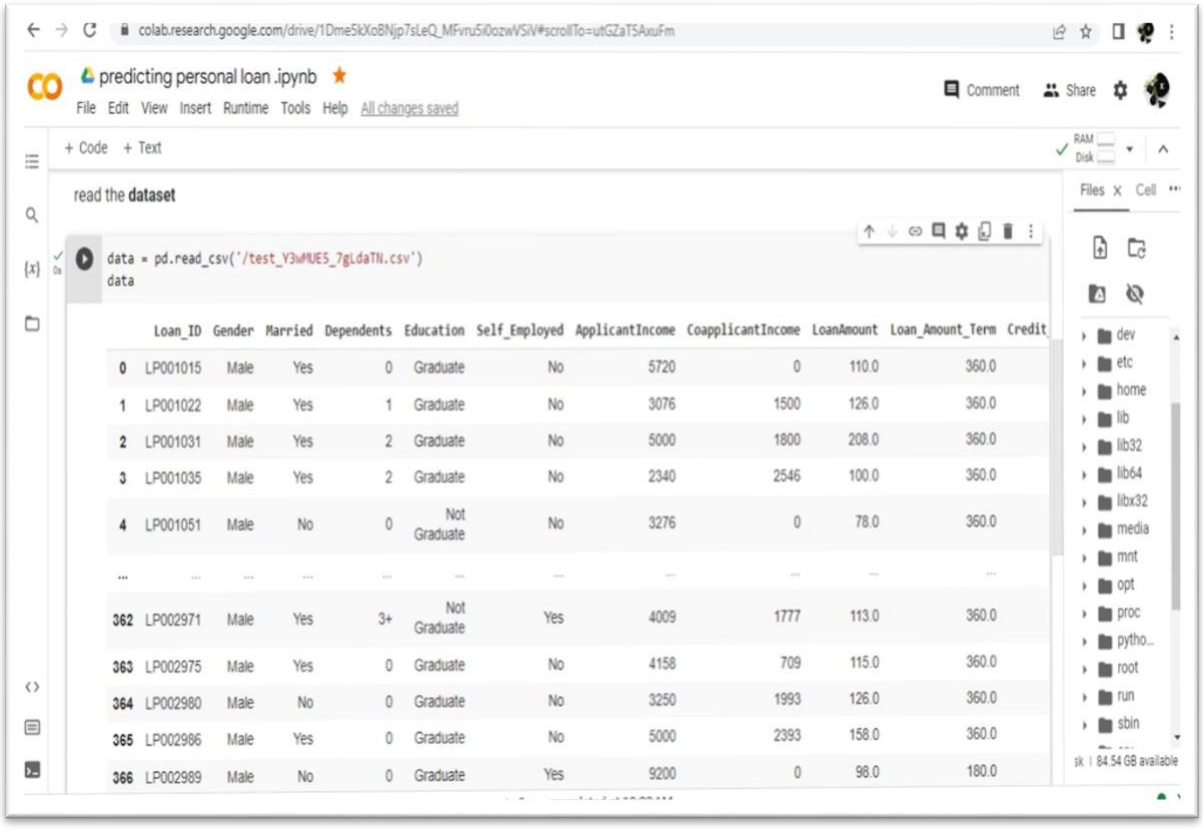
3.2idealation & brainstorming



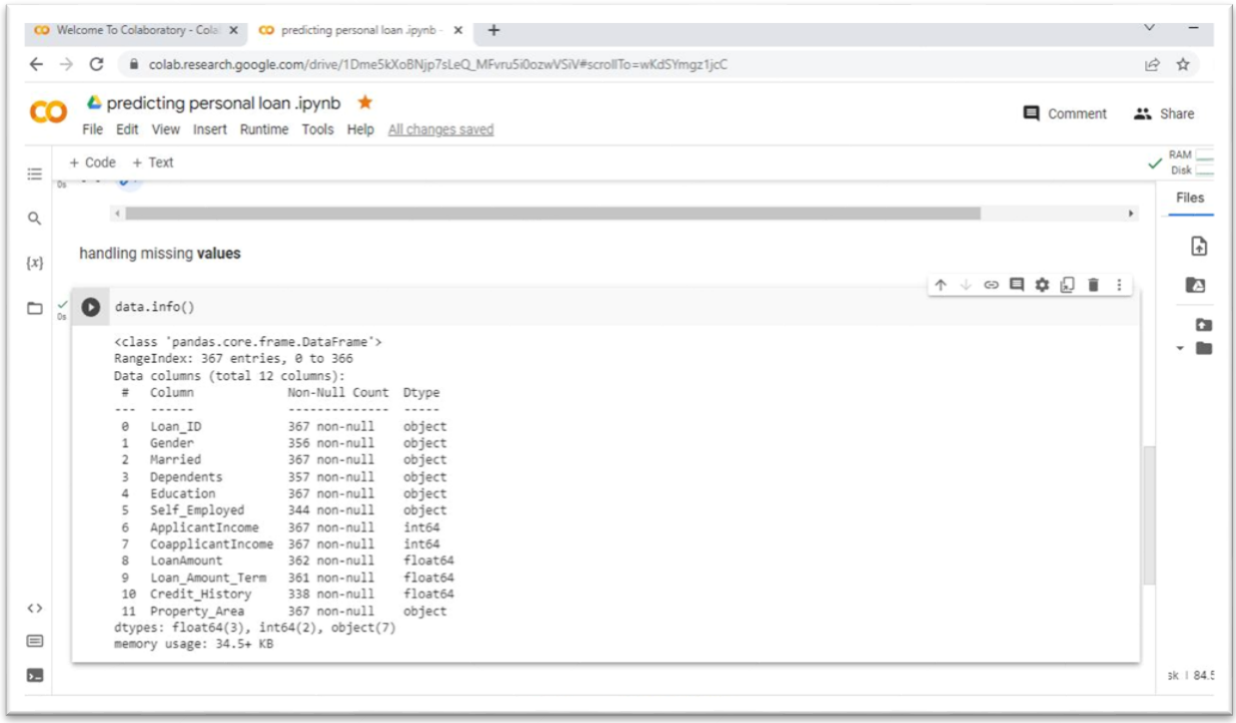
3.RESULT

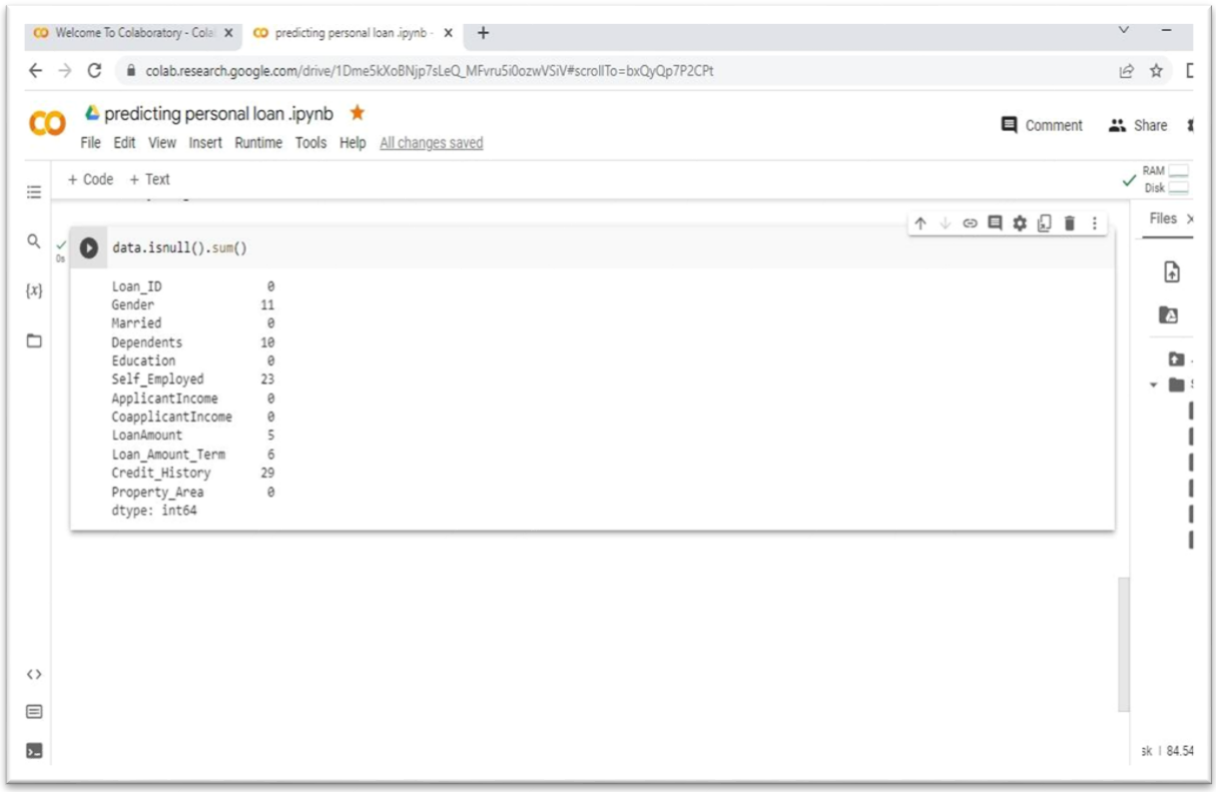
3.1COLAB RESULT

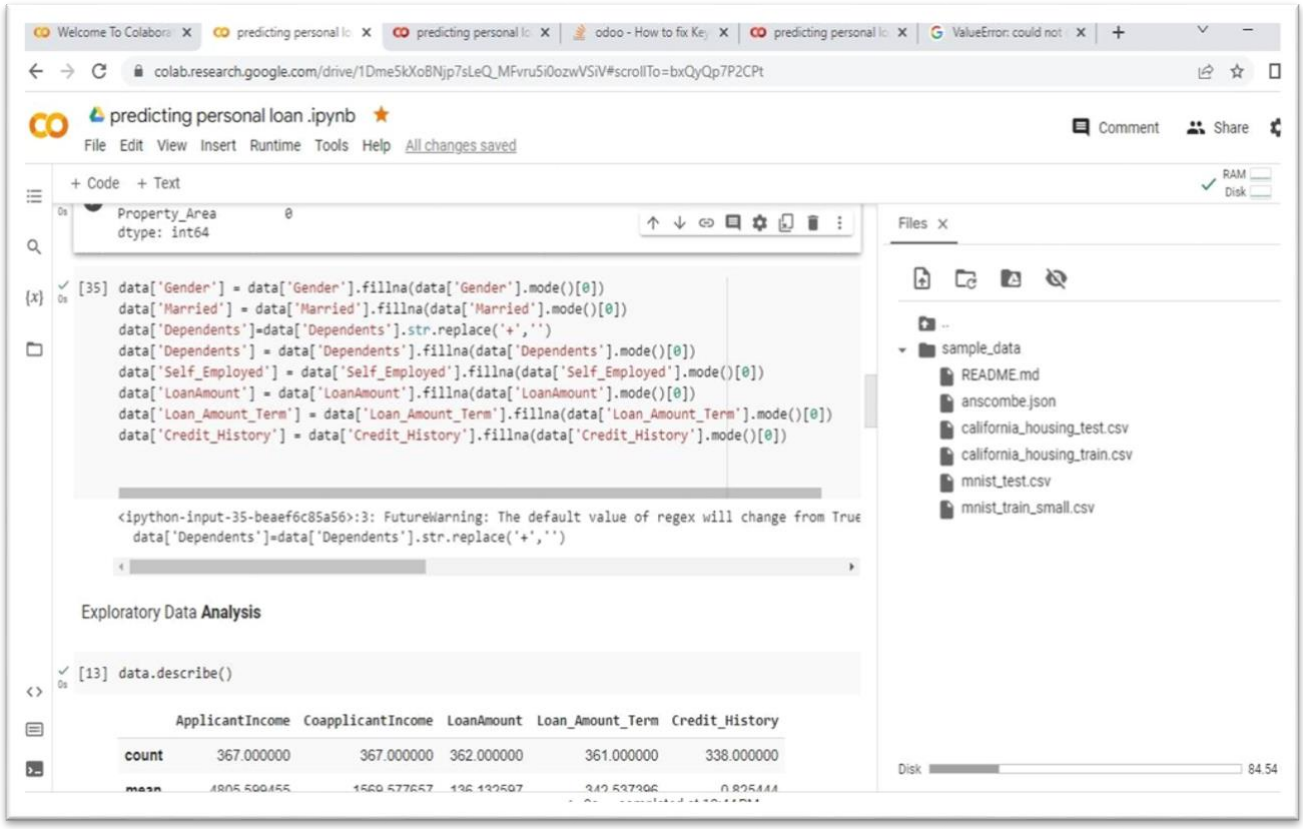


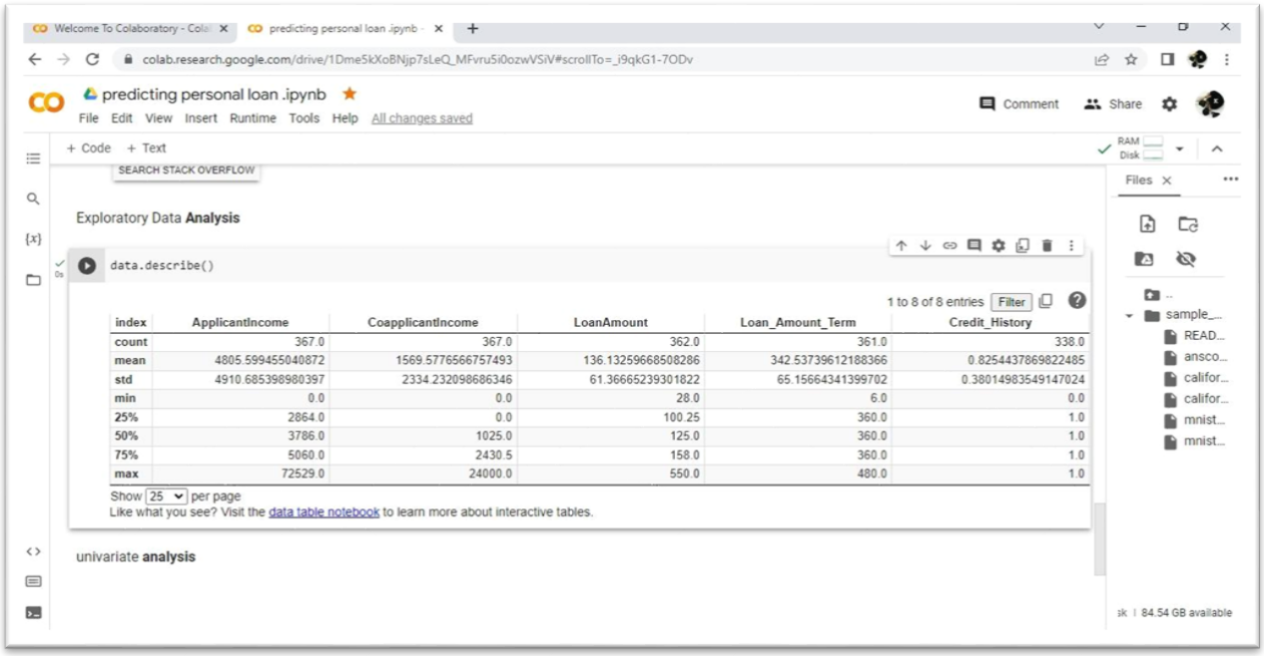


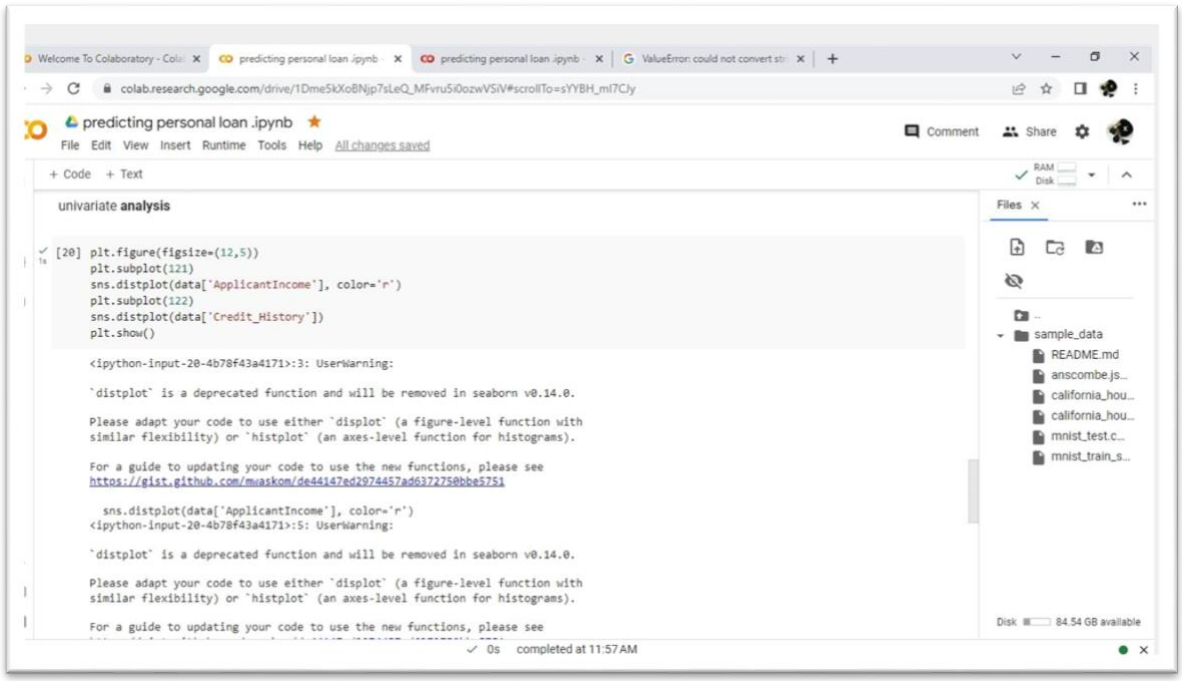
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	
...	...	...	...	...	...	...	...	...	...	...	...
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	1777	113.0	360.0	
363	LP002975	Male	Yes	0	Graduate	No	4158	709	115.0	360.0	
364	LP002980	Male	No	0	Graduate	No	3250	1993	126.0	360.0	
365	LP002986	Male	Yes	0	Graduate	No	5000	2393	158.0	360.0	
366	LP002989	Male	No	0	Graduate	Yes	9200	0	98.0	180.0	

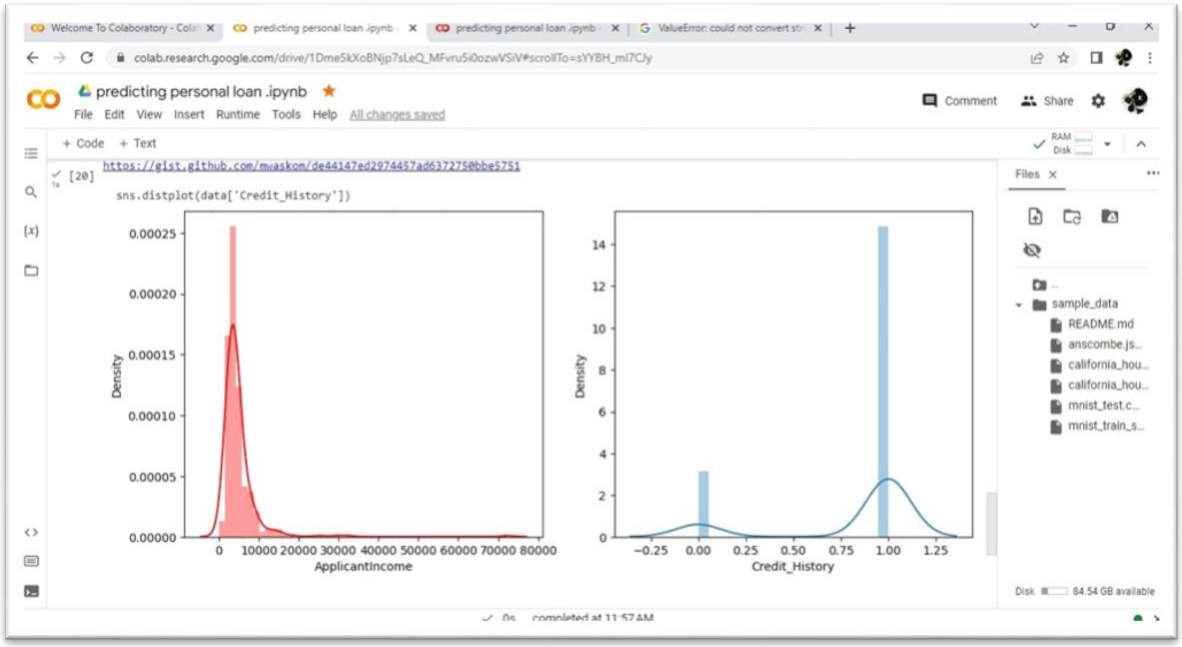

Welcome To Colaboratory - Cola
predicting personal loan .ipynb
colab.research.google.com/drive/1Dme5kXoBNjp7sLeQ_MFvru5i0ozwVSiV#scrollTo=wKdSYmgz1jcC
predicting personal loan .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Files
handling missing values
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 Loan_ID 367 non-null object
1 Gender 356 non-null object
2 Married 367 non-null object
3 Dependents 357 non-null object
4 Education 367 non-null object
5 Self_Employed 344 non-null object
6 ApplicantIncome 367 non-null int64
7 CoapplicantIncome 367 non-null int64
8 LoanAmount 362 non-null float64
9 Loan_Amount_Term 361 non-null float64
10 Credit_History 338 non-null float64
11 Property_Area 367 non-null object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB


Welcome To Colaboratory - Cola
predicting personal loan .ipynb
colab.research.google.com/drive/1Dme5kXoBNjp7sLeQ_MFvru5i0ozwVSiV#scrollTo=bxQyQp7P2CPt
predicting personal loan .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Files
data.isnull().sum()
Loan_ID 0
Gender 11
Married 0
Dependents 10
Education 0
Self_Employed 23
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount 5
Loan_Amount_Term 6
Credit_History 29
Property_Area 0
dtype: int64


predicting personal loan .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Property_Area 0
dtype: int64
[35] data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])
data['Married'] = data['Married'].fillna(data['Married'].mode()[0])
data['Dependents']=data['Dependents'].str.replace('+','')
data['Dependents'] = data['Dependents'].fillna(data['Dependents'].mode()[0])
data['Self_Employed'] = data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])
data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount'].mode()[0])
data['Loan_Amount_Term'] = data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0])
data['Credit_History'] = data['Credit_History'].fillna(data['Credit_History'].mode()[0])
<ipython-input-35-beaef6c85a56>:3: FutureWarning: The default value of regex will change from True
data['Dependents']=data['Dependents'].str.replace('+','')
Exploratory Data Analysis
[13] data.describe()
ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History
count 367.000000 367.000000 362.000000 361.000000 338.000000
Files
sample_data
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 84.54


predicting personal loan .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
SEARCH STACK OVERFLOW
Exploratory Data Analysis
data.describe()
1 to 8 of 8 entries Filter
index ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History
count 367.0 367.0 362.0 361.0 338.0
mean 4805.599455040872 1569.5776566757493 136.13259668508286 342.53739612188366 0.8254437869822485
std 4910.685398980397 2334.232098686346 61.36665239301822 65.15664341399702 0.38014983549147024
min 0.0 0.0 28.0 6.0 0.0
25% 2864.0 0.0 100.25 360.0 1.0
50% 3786.0 1025.0 125.0 360.0 1.0
75% 5060.0 2430.5 158.0 360.0 1.0
max 72529.0 24000.0 550.0 480.0 1.0
Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.
univariate analysis
Files
sample_...
READ...
ansco...
califor...
califor...
mnist...
mnist...
84.54 GB available

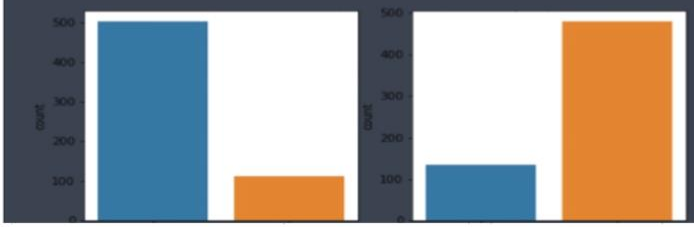

predicting personal loan .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
univariate analysis
[20] plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'], color='r')
plt.subplot(122)
sns.distplot(data['Credit_History'])
plt.show()
<ipython-input-20-4b78f43a4171>:3: UserWarning:
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
sns.distplot(data['ApplicantIncome'], color='r')
<ipython-input-20-4b78f43a4171>:5: UserWarning:
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
For a guide to updating your code to use the new functions, please see
Files
sample_data
README.md
anscombe.js...
california_hou...
california_hou...
mnist_test.c...
mnist_train_s...
Disk 84.54 GB available
0s completed at 11:57 AM


predicting personal loan .ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
[20]
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
sns.distplot(data['Credit_History'])
Density
ApplicantIncome
Credit_History
Files
sample_data
README.md
anscombe.js...
california_hou...
california_hou...
mnist_test.c...
mnist_train_s...
Disk 84.54 GB available

Activity 2.2: Bivariate analysis

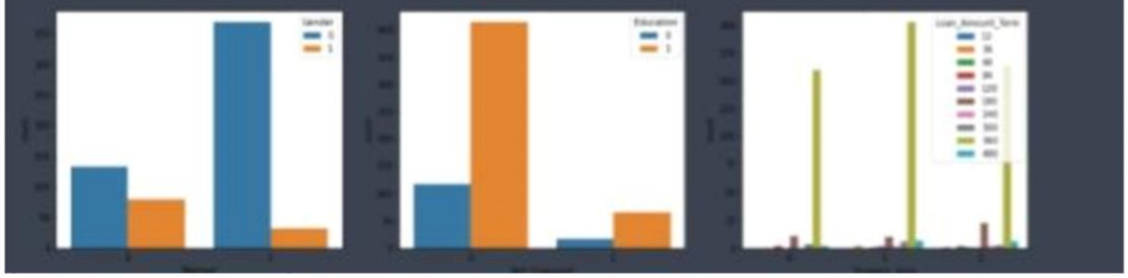
```
#plotting the count plot
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(data['Gender'])
plt.subplot(1,4,2)
sns.countplot(data['Education'])
plt.show()
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From ve
rsion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an e
rror or misinterpretation.
  warnings.warn(
```



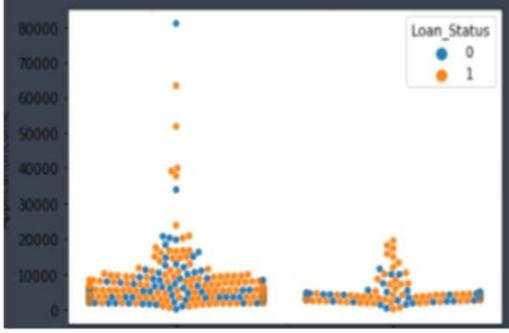

```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(data['Married'], hue=data['Gender'])
plt.subplot(132)
sns.countplot(data['Self_Employed'], hue=data['Education'])
plt.subplot(133)
sns.countplot(data['Property_Area'], hue=data['Loan_Amount_Term'])
```

```
error or misinterpretation.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(


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



```
isulaized based gender and income what would be the appplication status
s.swarmplot(data['Gender'],data['ApplicantIncome'], hue = data['Loan_Status'])
```

```
:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an
rror or misinterpretation.
 warnings.warn(
:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 67.1% of the points cannot be placed; you may want to de
rease the size of the markers or use stripplot.
 warnings.warn(msg, UserWarning)
:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 33.0% of the points cannot be placed; you may want to de
rease the size of the markers or use stripplot.
 warnings.warn(msg, UserWarning)


AxesSubplot:xlabel='Gender', ylabel='ApplicantIncome'>
```



ANN

```
# Importing the Keras libraries and packages
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
[226] # Initialising the ANN
      classifier = Sequential()
```

```
[227] # Adding the input layer and the first hidden layer
      classifier.add(Dense(units=100, activation='relu', input_dim=11))
```

```
      # Adding the second hidden layer
      classifier.add(Dense(units=50, activation='relu'))
```

```
[229] # Adding the output layer
      classifier.add(Dense(units=1, activation='sigmoid'))
```

```
[230] # Compiling the ANN
      classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

3.2 WEBSITE RESULT

4. ADVANTAGESS & DISADVANTAGES

ADVANTAGES OF PREDICTING PERSONAL LOAN:

1.ACCURACY: MACHINE LEARNING MODELS CAN ANALYZE VAST AMOUNTS OF HISTORICAL DATA TO MAKE HIGHLY ACCURATE PREDICTIONS ABOUT PERSONAL LOAN APPLICATIONS.

2. IMPROVED EFFICIENCY: MACHINE LEARNING MODELS CAN QUICKLY PROCESS DATA AND GENERATE PREDICTIONS, IMPROVING THE SPEED OF THE LOAN APPROVAL PROCESS.

3. REDUCED RISK: BY ACCURATELY PREDICTING WHO IS LIKELY TO APPLY AND RECEIVE A LOAN, MACHINE LEARNING CAN HELP LENDERS REDUCE THE RISK OF DEFAULT AND IMPROVE OVERALL FINANCIAL PERFORMANCE.

4. PERSONALIZATION: MACHINE LEARNING MODELS CAN BE PERSONALIZED TO FIT THE UNIQUE NEEDS AND CHARACTERISTICS OF A PARTICULAR LENDER, IMPROVING OUTCOMES FOR BOTH LENDERS AND BORROWERS.

DISADVANTAGES OF PREDICTING PERSONAL LOAN:

1. COMPLEXITY: DEVELOPING AND MAINTAINING MACHINE LEARNING MODELS CAN BE COMPLEX AND REQUIRE SPECIALIZED SKILLS AND RESOURCES.

2. DATA ISSUES: MACHINE LEARNING MODELS RELY ON HIGH-QUALITY DATA, WHICH CAN BE DIFFICULT TO OBTAIN AND MAINTAIN, LEADING TO LOWER ACCURACY AND LESS USEFUL PREDICTIONS.

3. BIAS: MACHINE LEARNING MODELS CAN BE BIASED BASED ON THE DATA USED TO TRAIN THEM, LEADING TO UNFAIR OUTCOMES OR DISCRIMINATION AGAINST CERTAIN INDIVIDUALS OR GROUPS.

4. INTERPRETABILITY: MACHINE LEARNING MODELS CAN BE DIFFICULT TO INTERPRET, MAKING IT CHALLENGING FOR LENDERS TO EXPLAIN THEIR PREDICTIONS AND DECISIONS TO BORROWERS, REGULATORS, OR OTHER STAKEHOLDERS.

5.APPLICATIONS

THERE ARE SEVERAL APPLICATIONS OF PREDICTING PERSONAL LOAN USING MACHINE LEARNING, INCLUDING:

1. *Credit risk assessment: Machine learning models can analyze credit history and other financial data to assess the risk of loan default and make more accurate lending decisions.*

2. *Customer segmentation: Machine learning can be used to segment customers based on their financial behavior, allowing lenders to target their marketing efforts more effectively.*

3. *Fraud detection: Machine learning algorithms can detect fraudulent loan applications by analyzing patterns of behavior or data anomalies.*

OVERALL, PREDICTING PERSONAL LOAN USING MACHINE LEARNING HAS THE POTENTIAL TO STREAMLINE LENDING PROCESSES, REDUCE RISK, AND IMPROVE CUSTOMER SATISFACTION.

6.CONCLUSION

- PREDICTING PERSONAL LOAN USING MACHINE LEARNING HAS SEVERAL POTENTIAL APPLICATIONS IN THE FINANCIAL INDUSTRY.

- BY LEVERAGING MACHINE LEARNING ALGORITHMS, LENDERS CAN IMPROVE CREDIT RISK ASSESSMENTS, CUSTOMER SEGMENTATION, FRAUD DETECTION, LOAN APPROVAL AUTOMATION, RISK-BASED PRICING, AND UPSELLING EFFORTS. THESE APPLICATIONS CAN STREAMLINE LENDING PROCESSES, REDUCE RISK, INCREASE PROFITABILITY, AND IMPROVE CUSTOMER SATISFACTION.

- HOWEVER, THERE ARE ALSO POTENTIAL CHALLENGES ASSOCIATED WITH IMPLEMENTING MACHINE LEARNING MODELS IN THE FINANCIAL SECTOR, INCLUDING PRIVACY CONCERNS, DATA QUALITY ISSUES, AND REGULATORY COMPLIANCE. DESPITE THESE CHALLENGES, THE BENEFITS OF USING MACHINE LEARNING TO PREDICT PERSONAL LOAN ARE CONSIDERABLE, AND THE TECHNOLOGY HAS THE POTENTIAL TO REVOLUTIONIZE THE WAY FINANCIAL INSTITUTIONS OPERATE IN THE LENDING INDUSTRY.

7.FUTURE SCOPE

THERE ARE SEVERAL POTENTIAL ENHANCEMENTS TO THE FUTURE SCOPE OF PREDICTING PERSONAL LOANS USING MACHINE LEARNING, INCLUDING:

1. Improved accuracy: As machine learning algorithms are trained on more data, they can become more accurate in predicting whether a loan will be approved or not. This could help lenders reduce their risk and identify more qualified borrowers, which could ultimately result in more loans being approved.

2. Personalized loan offerings: By analyzing individual borrower data, machine learning algorithms can help lenders customize loan offerings based on the borrower's financial profile. This could include offering lower interest rates to borrowers with good credit scores or extending repayment terms for borrowers with low incomes.

3. Increased efficiency: Machine learning can help lenders streamline their loan approval processes by automating many of the steps involved in evaluating loan applications. This could reduce the time it takes for borrowers to receive loan decisions and reduce the workload for loan officers.

4. Improved fraud detection: By analyzing borrower data and detecting patterns indicative of fraud, Machine Learning algorithms can help lenders identify fraudulent loan application Before they are approved. This cloud save lenders millions of dollars in losses and improve their overall financial stability.

5. Broader Market penetration: As machine learning algorithms become more sophisticated, they could enable lenders to penetrate new market and reach borrowers who were previously considered too risky or difficult to work with. This cloud expand the availability of personal loans to individuals who may not have access to traditional lending options.

8. APPENDIX

A. SOURCE CODE

IMPORTING THE LIBRARIES:-.

```
import pandas as pd

import numpy as np

import pickle

import matplotlib.pyplot as plt

matplotlib inline

import seaborn as sns

import sklearn

from sklearn.tree import decisiontreeclassifier

from sklearn.ensemble import gradientboostingclassifier, randomforestclassifier
```

```
from sklearn.neighbors import kneighborsclassifier

from sklearn.model_selection import randomizedsearchcv

import imblearn

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import standardscaler

from sklearn.metrics import accuracy score, classification_report, confusion_matr,f1_score
```

READ THE DATASET:-

pd.read_csv('loan_prediction.csv') data

data

HANDLING MISSING VALUES:-

data.info()

the sum of null values in each column

22 data.isnull().sum()

data] Gender'] = data['Gender].fillne(data[Gender].ode()[*])

datal Married'] datal Harried').fillna(datal 'Married').mode()[0])

#replacing + with space for filling the ran values

data['Dependents']-data['Dependents'].str.replace('+','')

data['Dependents'] data['Dependents'].fillna(data['Dependents'].mode()[0])

data['Self Employed'] data['Self Employed'].fillna(data['Self Employed"].mode()[0])

data['LoanAmount'] data['LoanAmount'].fillna(data["LoanAmount"].mode()[0])

data['Loan Amount Term'] data['Loan Amount Ters'].fillna(data["Loan Amount Ters'].mode()[0])

data['Credit History'] data['Credit History').fillna(data['Credit History'].mode()[0])

HANDLING CATEGORICAL VALUES:-

#changing the dotype of each flout column to int

datal Gender']-data['Gender'].astype('int64')

data['Harried]-data['Married'].astype('int64')

data[‘Dependents”]-data[‘Dependents’].astype(‘int64”)

data[‘Self Employed’]-data[‘Self Employed’].astype(“int64’)

data[‘Coapplicant Income]-data[‘Coapplicant Income].astype(“Int64”)

data[‘LoanAmount”]-data[“LoanAmount’].astype(‘int64”)

datal Loan Amount Tera’]-data[‘Loan Amount Tera’].astype(“Int64”)

datal Credit History”]-data[‘Credit History’).astype(“int64”)

HANDLING IMBALANCE DATA:-

#Balancing the dataset by using smote

from imblearn.combine import SMOTETomek

smote = SMOTETomek(0.90)

C\Users\HP\AppData\Roaming\Python\Python39\site-packages\imblearn\utils_validation.py:587: Futurewarning: Pass sampling keyword args. From version 9.9 passing these as positional arguments will result in an error warnings.warn(

#dividing the dataset into dependent and independent y and x respectively

y =data[‘Loan_Status’]

X=data.drop(columns-[‘Loan_Status’),axis-1)

#creating a new x and y vartalbles for the balanced set

x_bal,y_bal smote.fit_resample(x,y)

#printing the values of y before balancing the dots and ofter

print(y.value_counts())

Print(y_bal.value_counts())

EXPLORATORY DATA ANALYSIS:-

data.describe()

UNIVARIATE ANALYSIA:-

plotting the using distplot plt.figure(figsize=(12,5))

plt.subplot(121)

sns.distplot(data[‘Applicant Income”], color=’r’)

Plt.subplot(177)

plt.subplot (122) sns.distplot(data['Credit_History"]}

Plt.show()

BIVARIATE ANALYSIS:-

#plotting the count plot plt.figure(figsize=(18,4))

plt.subplot(1,4,1)

sns.countplot(data['Gender'])

plt.subplot(1,4,2)

sns.countplot(data['Education'])

plt.show()

plt.figure(figsize=(20,5))

plt.subplot(131)

sns.countplot(data['Married'], hue-data['Gender'])

plt.subplot(132)

sns.countplot(data['Self_Employed'], hue-data["Education'])

sts.countplot{data['Property Area'), hue-data['Loan_Amount_Term`]}

MULTIVARIATE ANALYSIS:-

sns.swarmplot(data['Gender"],data["ApplicantIncome'], hue = data["Loan_Status'])

sc-StandardScaler()

x_bal-sc.fit_transform(x_bal)

X_bal=pd.DataFrame(x_bal,columns-names)

#splitting the dataset in train and test on balanced datasew

X_train, X_test, y_train, y_test = train_test_split(

x_bal, y_bal, test_size=0.33, random_state=42)

DECISION TREE MODEL :-

def decisionTree(x_train, x_test, y_train, y_test)

dt= DecisionTreeClassifier()

```
    dt.fit(x_train,y_train)

    dt.predict(x_test)

    print(***DecisionTreeClassifier****)

    print('Confusion matrix')

    print(confusion_matrix(y_test,yPred))

    Print('Classification report") print(classification_report(y_test,yPred))
```

RANDOM FOREST MODEL:-

```
def randomForest (x_train, x_test, y_train, y_test):

    rf = RandomForestClassifier()

    rf.fit(x_train,y_train)

    yPred = rf.predict(x_test)

    print(****RandomForestClassifier****)

    print('Classification report')

    print(classification_report (y_test,yPred))

    print('Confusion matrix')

    print(confusion_matrix(y_test,yPred))
```

KNN MODEL:-

```
def KNN(x_train, x_test, y_train, y_test):

    knn = KNeighborsClassifier()

    knn.fit(x_train,y_train)

    yPred = knn.predict(x_test)

    print(***KNeighborsClassifier***')

    print('Confusion matrix')

    print(confusion_matrix(y_test, yPred))

    print('Classification report')

    print(classification_report (y_test,yPred))
```

XGBOOST MODEL:-

```
def xgboost (x_train, x_test, y_train, y_test):
    xg = Gradient BoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print(***Gradient BoostingClassifier****)
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report")
    print(classification_report(y_test,yPred))
```

ANN MODEL:-

```
#Importing the Keras libraries and packages
import tensor flow
from tensorflow.keras.models import Sequential
 from tensorflow.keras.. layers import Dense
#Initializing the ANN
[226] # Initialising the A NN
 classifier Sequential()
 [227] # Adding the input layer and the first hidden layer
 classifier.add(Dense (units-100, activation-'relu', input_dis-11))
# Adding the second hidden layer
 classifier.add(Dense(units-50, activation-'relu"))
 [229] Adding the output layer
classifier.add(Dense(units-1, activation-sigmoid'))
[230] Compiling the ANN
 classifier.compile(optimizer-adam', loss-binary crossentropy', metrics-["accuracy'])
#Fitting the ANN to the training set
modell history classifier.fit(x_train, y train, batch_size-100, validation_split-0.2, epochs-100)
```

TESTING THE MODEL:-

#Gender Married Dependents Education Self Employed Applicant is one Coapplicant Income Loan _Amount _Term _ history_Area,

dtr.predict([[1,1, 0, 1, 1, 4276, 1542,145, 248, 0,1]])

/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature warnings.warn(array([0])

[149] #Gender Married Dependents Education Self Employed Applicant Income Coapplicant income Loanmount Loan Amount Term Credit History Property Area

rfr.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: Userlarning: X does not have valid feature names, but RandomforestClassifier was fitted with feature names warnings.warn(array([1])

#Gender Married Dependents Education Self Employed Applicant Income Coapplicant Income Loan amount Loan Amount Term Credit History Property _Area

knn.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

/usr/local/lib/python3.6/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but neighbors Classifier was fitted with feature names warnings.warn(array([1])

[153] #Genderr Married Dependents Education Self Employed Applicant Income Compplicant income loan amount Loan_ Amount Term Credit History Property _Ares

xgb.predict([[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]])

/usr/local/lib/python3.5/dist-packages/sklearn/base.py:450: userwarning: X does not have valid feature a warnings.warn(GradientestingClassifier was fitted with feature name, but GradientBoostingClassifier was fitted with features name's warning.warn(

array([1])

classifier.save("loan.h5")

#Predicting the Test set results y_pred = classifier.predict(x_test)

[237] y pred

[238] y pred Y pred (y_pred > 0.5)

y_pred

def predict_exit(sample_value):

#Predictions

#Value order 'CreditScore', 'Age', 'Tenure', 'Balance', 'NumofProducts", "HasCrCard, 'IsActiveRember", 'EstimatedSalary", "France", "Germany", "Spain", "Female Male".

sample value [[1,1, 0, 1, 1, 4276, 1542,145, 240, 0,1]]

if predict exit(sample_value)>0.5:

print("Prediction: High chance of Loan Approvall')

Else:

print("Prediction: Low chance Loan Approval.")

#Predictions

#value order 'CreditScore', 'Age", "Tenure", "Balance", "NumofProducts", "HasCrCard", "IsActiveMember", "Estimatedsalary", "France", "Germany", "Spain",'Female','Male'.

Sample_value [[1,8, 1, 1, 1, 45, 14,45, 240, 1,1]]

if predict_exit(sample_value)>0.5:

print("Prediction: High chance of Loan Approval!')

Else:

print('Prediction: Low chance of Loan Approval.")

COMPARE THE MODEL:-

def compareModel(x_train,x_test,y_train,y_test):

decisionTree(x_train,x_test,y_train,y_test)

print('-'*100)

RandomForest (x_train,x_test,y_train,y_test)

print('-'*100)

XGB (X_train,x_test,y_train,y_test)

print('-'*100) KNN(X_train,x_test,y_train,y_test)

print('-100)**

compareModel(x_train,x_test,y_train,y_test)

yPred classifier.predict(x_test)

print(accuracy_score (y_pred,y_test))

```
print("ANN Model")

print("Confusion_Matrix")

print(confusion_matrix(y_test, y_pred))

print("classification Report")

print(classification_report (y_test,y_pred))
```

COMPARING MODEL ACCURACY BEFORE & AFTER APPLYING HYPERPARAMETER TUNING:-

```
from sklearn.model_selection import cross_val_score

#Random forest model is selected

rf = RandomForestClassifier()

rf.fit(x_train,y_train)

yPred rf.predict(x_test)

f1_score (yPred,y_test, average='weighted')

CV=Cross_val_score(rf,x,y,CV=5)

np.mean(CV)
```

MODEL DEPLOYMENT:-

```
#saving the model by using pickle function

Pickle.dump(model,open('rdf.pkl','wb'))
```