



# Tech Saksham

## Capstone Project Report

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING  
FUNDAMENTALS

**“An End-to-End Data Science Project  
with ChatGPT”**

**“UNIVERSITY COLLEGE OF ENGINEERING (BIT  
CAMPUS) TIRUCHIRAPALLI”**

NM ID	NAME
au81002112704	PRAVEEN M

Trainer Name

Ramar Bose

Sr. AI Master Trainer

# ABSTRACT

This succinct end-to-end data science with ChatGPT project revolves around predicting loan default using a loan dataset from a financial institution. It entails data preprocessing, exploratory data analysis, and feature engineering to prepare the dataset for modeling. Leveraging machine learning algorithms like logistic regression, decision trees, random forests, and gradient boosting, predictive models are developed to forecast the likelihood of loan default. Feature importance analysis guides the identification of key predictors. Rigorous model evaluation ensures reliability and generalization. Ultimately, the best-performing model is deployed for real-time predictions, aiding financial institutions in proactive risk management and fostering a stable lending environment.

# INDEX

Sr. No.	Table of Contents	Page No.
1	Chapter 1: Introduction	4
2	Chapter 2: Services and Tools Required	6
3	Chapter 3: Project Architecture	7
4	Chapter 4: Modeling and Project Outcome	9
5	Conclusion	18
6	Future Scope	19
7	References	20
8	Links	21



## CHAPTER 1

### INTRODUCTION

#### 1.1 Problem Statement

The goal of this project is to develop a comprehensive loan approval system using machine learning techniques and natural language processing (NLP) capabilities of ChatGPT.

Leveraging a dataset of past loan applications, the project aims to build a predictive model that can assess the creditworthiness of new applicants based on their financial history and personal information. Additionally, integrating ChatGPT into the system will enable the automation of customer interactions, allowing for a more seamless and efficient loan application process. By combining advanced analytics with conversational AI, the project seeks to improve the accuracy and speed of loan approvals while enhancing the user experience for both applicants and loan officers.

#### 1.2 Proposed Solution

For an end-to-end data science project utilizing ChatGPT with a loan dataset, the proposed solution involves several key steps. First, comprehensive data preprocessing is necessary to clean and prepare the loan dataset, including handling missing values and outliers. Next, feature engineering can help extract relevant information from the data to improve model performance. Then, a machine learning model, such as logistic regression or random forest, can be trained to predict loan approval or rejection based on historical data. Integration of ChatGPT allows for a conversational interface where users can inquire about loan eligibility criteria, receive personalized recommendations, or seek assistance with the loan application process. Finally, thorough testing and evaluation ensure the model's accuracy and effectiveness in real-world scenarios.

#### 1.3 Feature

- **Data Gathering:** Collect loan dataset with borrower information.
- **Model Training:** Train ChatGPT on the loan data to understand queries.
- **User Interaction:** Allow users to ask questions or seek advice about loans.
  
- **Response Generation:** Generate informative responses based on loan dataset and user queries.

## 1.4 Advantages

- Risk Reduction: Predicting loan defaults beforehand helps minimize financial risks for lenders.
- Efficient Decision-Making: Data-driven insights enable smarter choices in loan approvals, terms, and rates.
- Cost Savings: Early identification of defaults saves money on collection efforts and legal actions.
- Personalized Service: Tailoring loan offerings to individual profiles enhances customer satisfaction.
- Competitive Edge: Data-driven strategies keep lenders ahead, ensuring profitability and market leadership.

## 1.5 Scope

The scope of an end-to-end data project integrating ChatGPT with a loan dataset is multifaceted. Firstly, leveraging historical loan data, the project aims to develop predictive models for assessing creditworthiness and risk analysis. ChatGPT will be integrated to enhance customer interaction and support throughout the loan application process, providing personalized assistance, answering inquiries, and offering guidance tailored to individual needs. Additionally, natural language processing capabilities will facilitate sentiment analysis of customer interactions, enabling real-time monitoring of customer satisfaction and feedback. Overall, the project endeavors to streamline the loan application journey, improve customer experience, and optimize lending decisions through the synergy of data analytics and AI-driven conversational interfaces.

# CHAPTER 2

## SERVICES AND TOOLS REQUIRED

### 2.1 Services Used

- **Data Collection:** Gather loan dataset including borrower information, loan details, and repayment history.
- **Data Preprocessing:** Clean, format, and preprocess the dataset to ensure consistency and remove noise.
- **Model Training:** Utilize ChatGPT to train a conversational AI model on the loan dataset to understand queries and provide responses.
- **Integration:** Integrate ChatGPT into the loan application system to provide end-to-end conversational support for loan inquiries and assistance.
- **Evaluation and Monitoring:** Continuously evaluate the performance of the system and monitor interactions to ensure accuracy and effectiveness in addressing user queries.

### 2.2 Tools and Software used

#### Tools:

- **Data Collection Tools:**
  - Web scraping tools (e.g., BeautifulSoup, Scrapy)
  - APIs for accessing financial data (e.g., Alpha Vantage, Quandl)
  - Data integration platforms (e.g., Talend, Informatica)
- **Data Preprocessing Tools:**
  - Data cleaning libraries (e.g., pandas, dplyr)
  - Data transformation tools (e.g., Trifacta, Alteryx)
  - Missing data imputation techniques (e.g., fancyimpute, scikit-learn)
- **Exploratory Data Analysis (EDA) Tools:**
  - Visualization libraries (e.g., Matplotlib, Seaborn, Plotly)
  - Statistical analysis tools (e.g., RStudio, Jupyter Notebooks)
  - Interactive dashboard platforms (e.g., Tableau, Power BI)
- **Feature Engineering Tools:**
  - Feature engineering libraries (e.g., scikit-learn, Featuretools)
  - Automated feature engineering platforms (e.g., DataRobot, H2O.ai)
- **Machine Learning Tools:**
  - Machine learning libraries (e.g., scikit-learn, TensorFlow, PyTorch)

Cloud-based machine learning platforms (e.g., AWS SageMaker, Google AI Platform, Microsoft Azure Machine Learning)

- **Model Deployment and Monitoring Tools:**

Model deployment frameworks (e.g., Flask, FastAPI)

Model monitoring platforms (e.g., MLflow, Kubeflow)

### Software Requirements:

- **Python** for scripting and data manipulation.
- **TensorFlow** or PyTorch for deep learning.
- **ChatGPT** for natural language processing.
- **Pandas** for data manipulation.
- **Flask** or Django for web deployment.

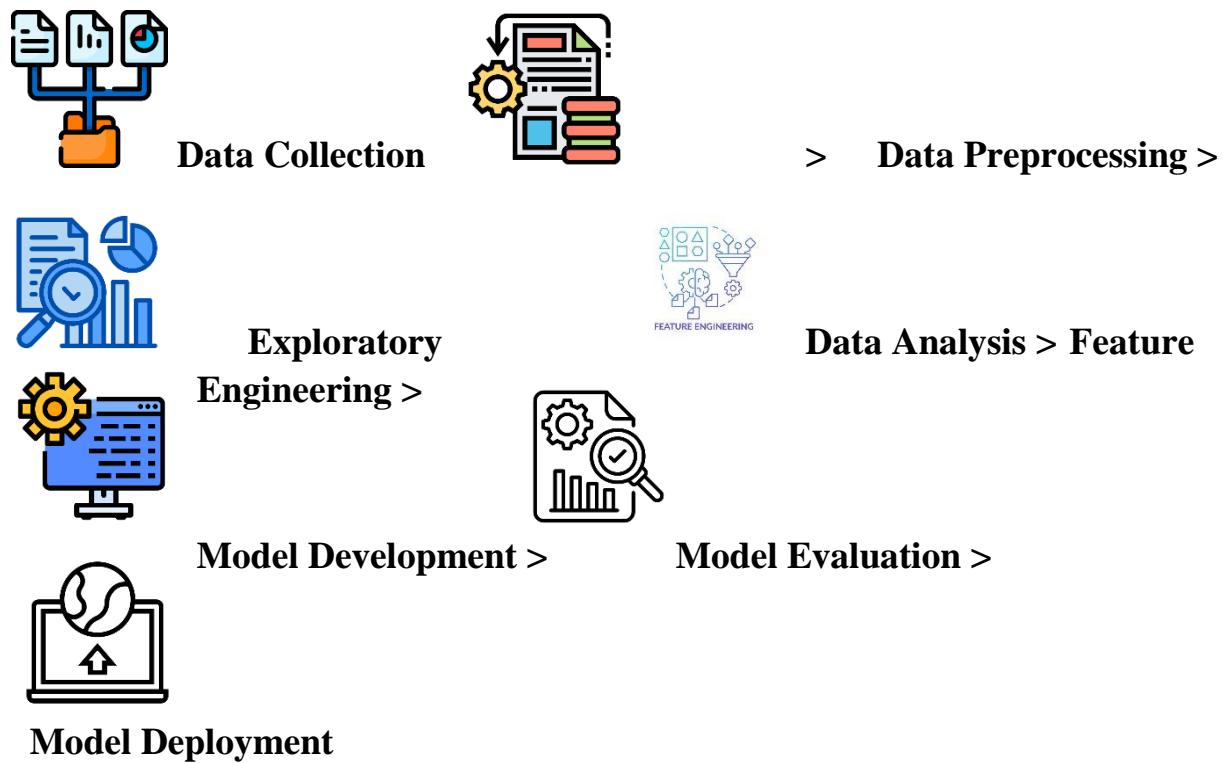
## CHAPTER 3 PROJECT

### ARCHITECTURE

#### 3.1 Architecture:

#### End to end data science project with ChatGPT:

## Flow Diagram:



1. **Data Collection:** Gather loan data from various sources such as databases, APIs, and files.
2. **Data Preprocessing:** Cleanse, transform, and prepare the dataset for analysis by handling missing values, outliers, and data inconsistencies.
3. **Exploratory Data Analysis (EDA):** Explore the dataset to understand its characteristics, patterns, and relationships.
4. **Feature Engineering:** Extract and create relevant features from the dataset to enhance predictive modeling.
5. **Model Development:** Build and train machine learning models using algorithms like logistic regression, decision trees, and XGBoost for loan default prediction.
6. **Model Evaluation:** Assess the performance of the models using evaluation metrics like accuracy, precision, recall, and F1-score.
7. **Model Deployment:** Deploy the best-performing model into production for real-time predictions on new loan data.

## CHAPTER 4 (code)

### MODELING AND PROJECT OUTCOME

Asking chat GPT which is a AI tool to provide the project plan

ChatGPT 3.5 ▾

You  
provide a project plan for end to end data science project with a loan data set

Save & Submit Cancel

ChatGPT  
Sure, here's a brief outline for a data science project using a loan dataset:

1. **Project Definition and Planning:**
  - Define the project goals: Determine what insights or predictions you want to derive from the loan dataset.
  - Identify stakeholders: Understand who will benefit from the project outcomes.
  - Plan the project timeline: Break down tasks and allocate time for each phase.
2. **Data Collection:**
  - Gather the loan dataset: Obtain the necessary data from sources such as Kaggle, UCI Machine Learning Repository, or directly from financial institutions (if applicable).
  - Assess data quality: Check for missing values, duplicates, outliers, and inconsistencies.
  - Explore dataset features: Understand the meaning and relevance of each attribute.
3. **Data Preprocessing:** ↓
  - Handle missing data: Impute missing values or remove instances with missing data.

Message ChatGPT...  ?

ChatGPT can make mistakes. Consider checking important information.

ChatGPT 3.5 ▾

3. **Data Preprocessing:**

- Handle missing data: Impute missing values or remove instances with missing data.
- Handle outliers: Determine if outliers should be treated or removed.
- Feature engineering: Create new features, transform existing ones, or select relevant features.
- Data normalization or standardization: Scale numerical features if necessary.

4. **Exploratory Data Analysis (EDA):**

- Analyze the distribution of target variable: Understand the distribution of loan status (e.g., approved, denied).
- Explore relationships between features and target variable: Identify patterns and correlations.
- Visualize key insights: Use plots, charts, and graphs to communicate findings.

5. **Model Development:**

- Split dataset: Divide the data into training, validation, and test sets.
- Select appropriate algorithms: Choose classification algorithms suitable for predicting loan approval.
- Train models: Fit the selected algorithms on the training data.
- Evaluate models: Assess model performance using appropriate metrics (e.g., accuracy, precision, recall, F1-score).
- Fine-tune hyperparameters: Optimize model parameters to improve performance.

6. **Model Deployment:** ↓

- Choose a deployment strategy: Decide whether to deploy the model locally or on a cloud

Message ChatGPT...  ?

ChatGPT can make mistakes. Consider checking important information.

ChatGPT 3.5 ▾

6. **Model Deployment:**

- Choose a deployment strategy: Decide whether to deploy the model locally or on a cloud platform.
- Develop APIs or interfaces: Create endpoints for accessing the model predictions.
- Test deployment: Ensure the deployed model behaves as expected and handles edge cases gracefully.

7. **Monitoring and Maintenance:**

- Set up monitoring: Implement systems to track model performance and drift over time.
- Schedule regular updates: Plan for model retraining to keep it relevant with changing data patterns.
- Address feedback: Incorporate feedback from users and stakeholders to improve model effectiveness.

8. **Documentation and Reporting:**

- Document the entire process: Write detailed documentation covering data sources, preprocessing steps, model development, and deployment.
- Prepare a final report: Summarize key findings, insights, and recommendations in a concise report for stakeholders.

Remember to adapt this outline based on the specific requirements of your project and the characteristics of the loan dataset you're working with.

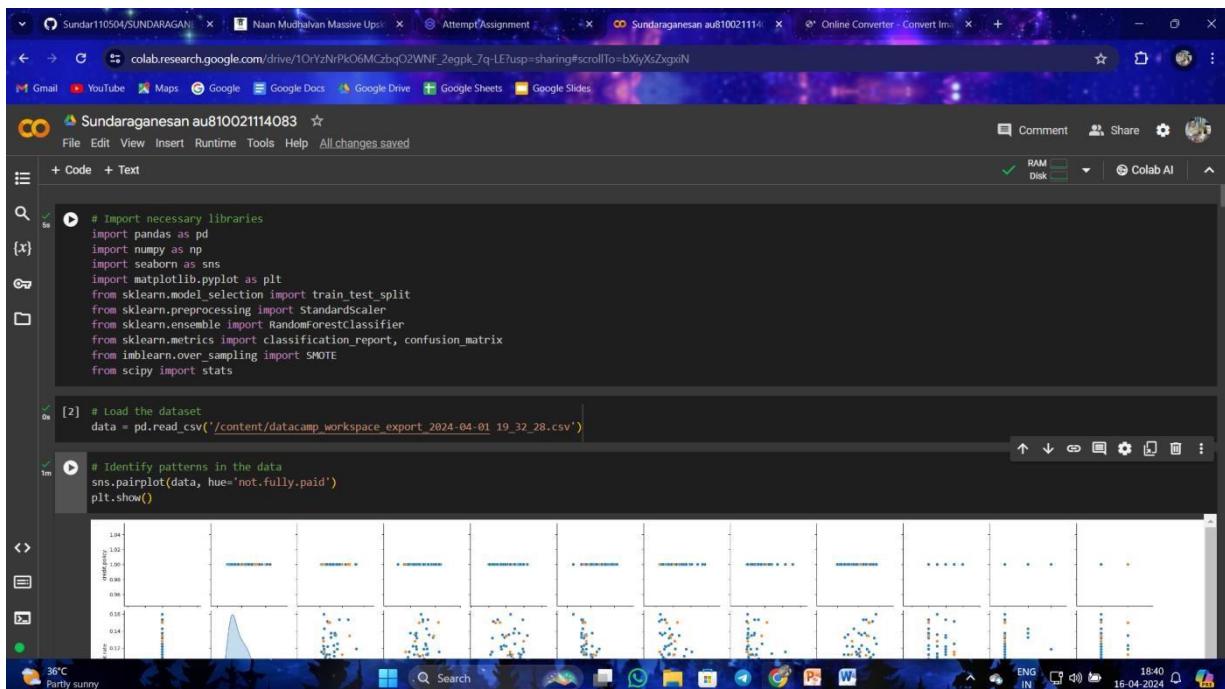
Message ChatGPT... 

ChatGPT can make mistakes. Consider checking important information. 

The asked the ChatGPT “to provide the necessary codes for the project. The codes are implemented and the output is received.

Code:

1. Data Collection
2. Data Preprocessing
3. Exploratory Data Analysis (EDA)

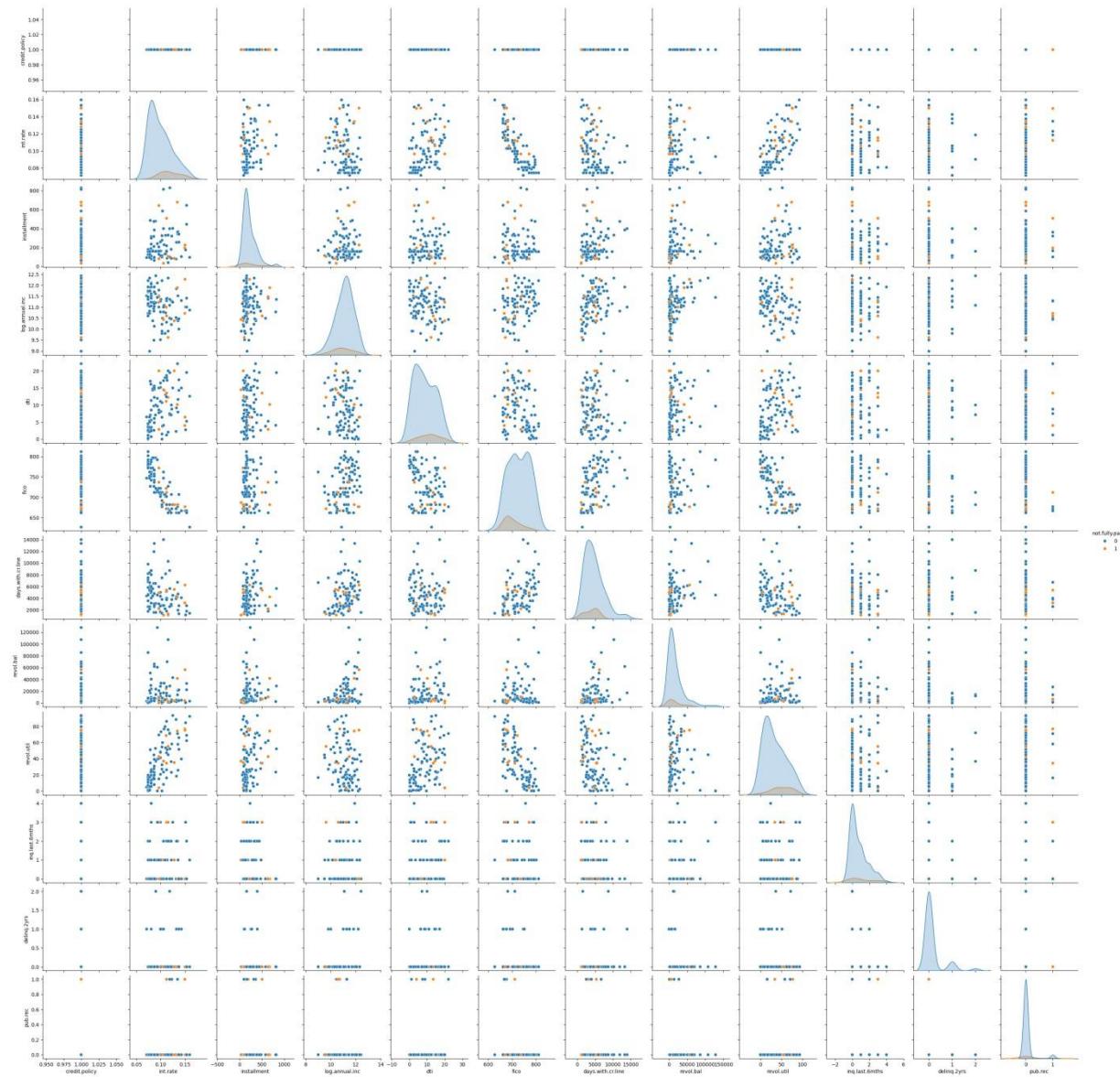


```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
from scipy import stats

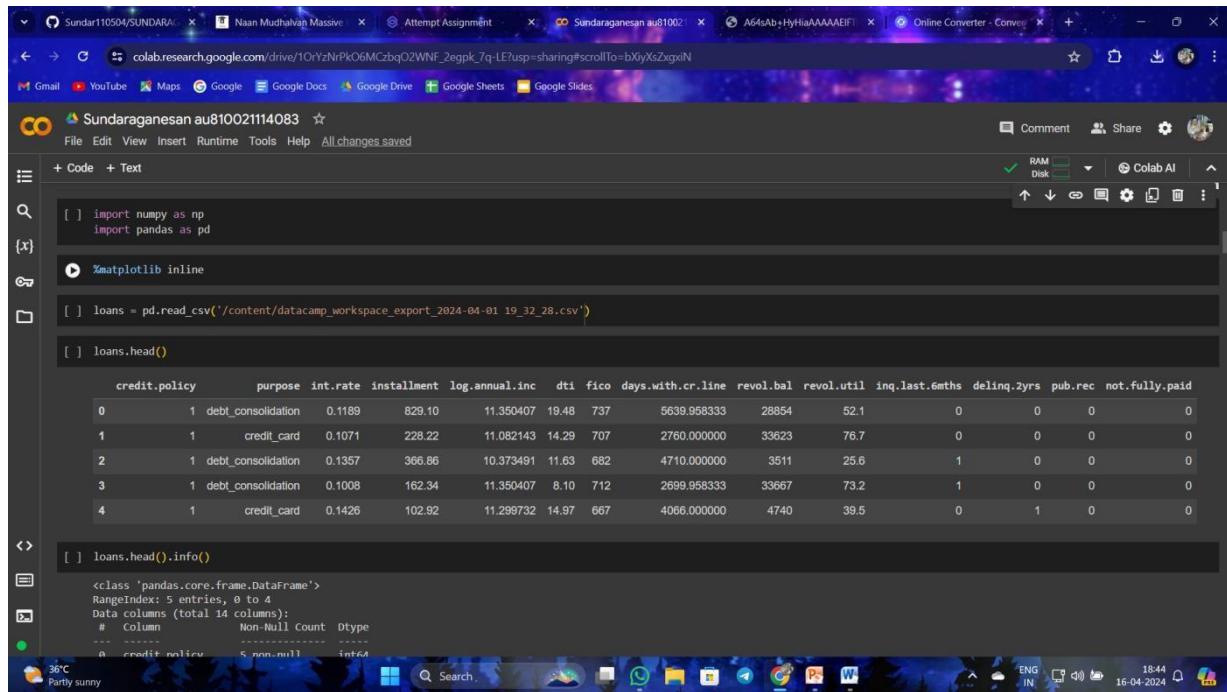
# Load the dataset
data = pd.read_csv('/content/datacamp_workspace_export_2024-04-01_19_32_28.csv')

# Identify patterns in the data
sns.pairplot(data, hue='not.fully.paid')
plt.show()
```

## Output:



The codes and the output are screenshots



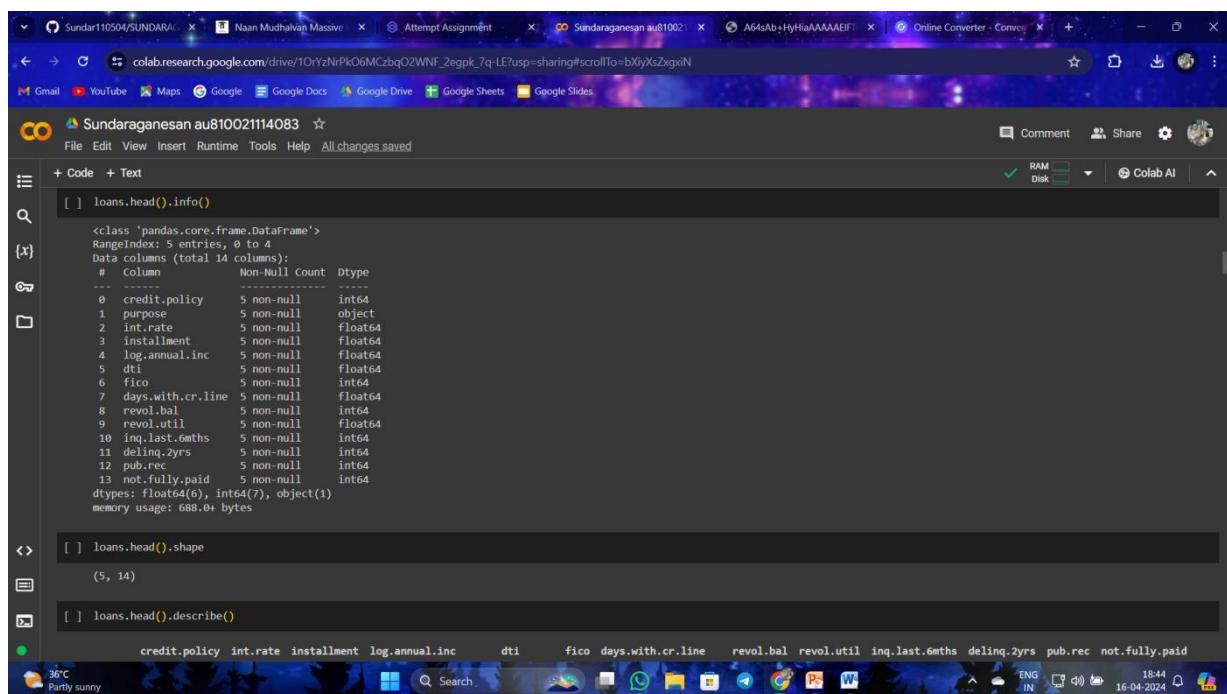
```
[ ] import numpy as np
import pandas as pd

[ ] %matplotlib inline

[ ] loans = pd.read_csv('/content/datacamp_workspace_export_2024-04-01_19_32_28.csv')

[ ] loans.head()

  credit.policy      purpose int.rate installment log.annual.inc    dti     fico days.with.cr.line  revol.bal revol.util inq.last.6mths delinq.2yrs pub.rec not.fully.paid
  0         1 debt.consolidation   0.1189       829.10  11.350407 19.48    737      5639.958333    28854      52.1        0        0        0        0
  1         1 credit_card       0.1071      228.22  11.082143 14.29    707     2760.000000    33623      76.7        0        0        0        0
  2         1 debt.consolidation   0.1357      366.86  10.373491 11.63    682     4710.000000    3511      25.6        1        0        0        0
  3         1 debt.consolidation   0.1008      162.34  11.350407 8.10    712      2699.958333    33667      73.2        1        0        0        0
  4         1 credit_card       0.1426      102.92  11.299732 14.97    667     4066.000000    4740      39.5        0        1        0        0
```



```
[ ] loans.head().info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
 0   credit.policy    5 non-null       int64  
 1   purpose          5 non-null       object 
 2   int.rate          5 non-null       float64
 3   installment       5 non-null       float64
 4   log.annual.inc   5 non-null       float64
 5   dti               5 non-null       float64
 6   fico              5 non-null       int64  
 7   days.with.cr.line 5 non-null       float64
 8   revol.bal         5 non-null       int64  
 9   revol.util        5 non-null       float64
 10  inq.last.6mths   5 non-null       int64  
 11  delinq.2yrs       5 non-null       int64  
 12  pub.rec           5 non-null       int64  
 13  not.fully.paid   5 non-null       int64  
dtypes: float64(6), int64(7), object(1)
memory usage: 688.6+ bytes

[ ] loans.head().shape
(5, 14)

[ ] loans.head().describe()

  credit.policy  int.rate  installment  log.annual.inc    dti     fico days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.2yrs  pub.rec  not.fully.paid

```

Sundar110504/SUNDARA/... Naan Mudhalvan Massive Attempt Assignment Sundaraganesan au81002 A64sAb+HyHiaAAAAEIFI Online Converter - Conv...

Sundaraganesan au810021114083 ★

File Edit View Insert Runtime Tools Help All changes saved

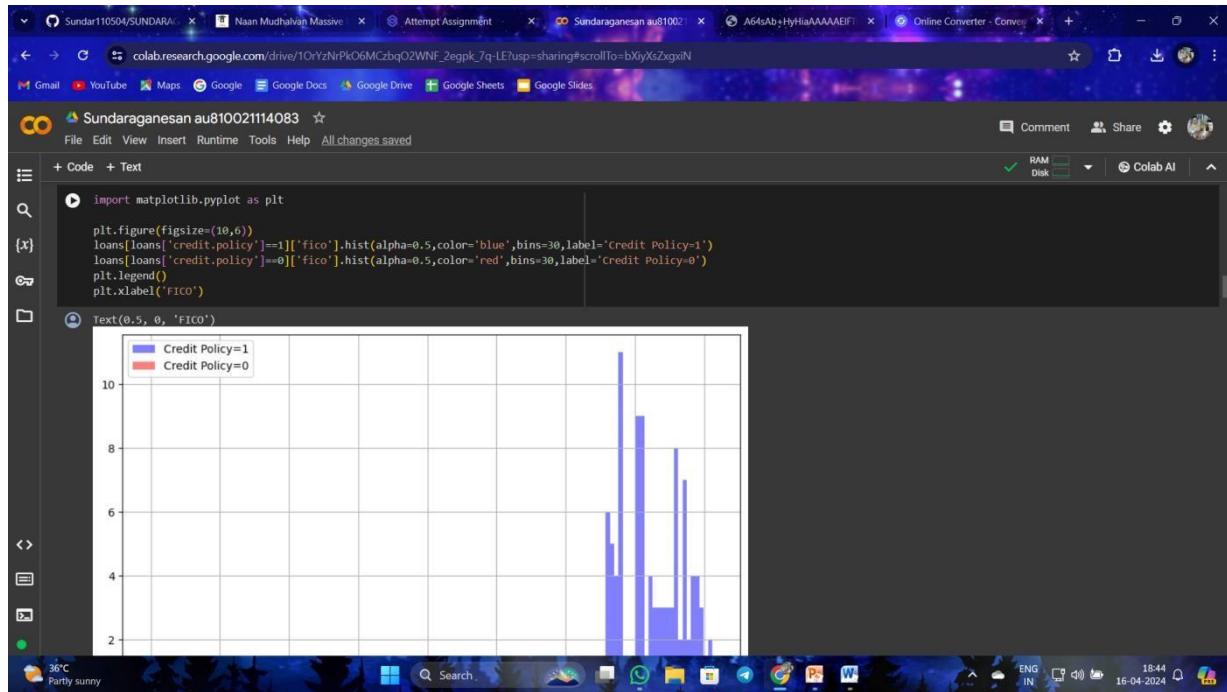
```
+ Code + Text
[ ] loans.head().shape
(5, 14)
[ ] loans.head().describe()

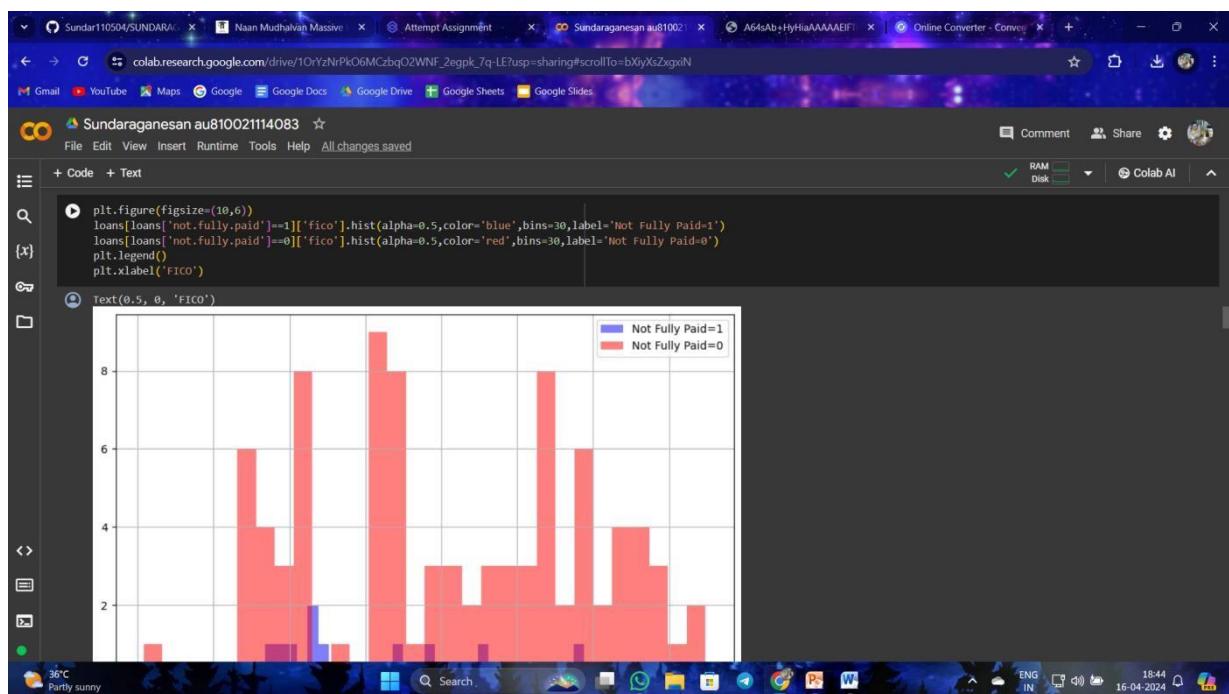
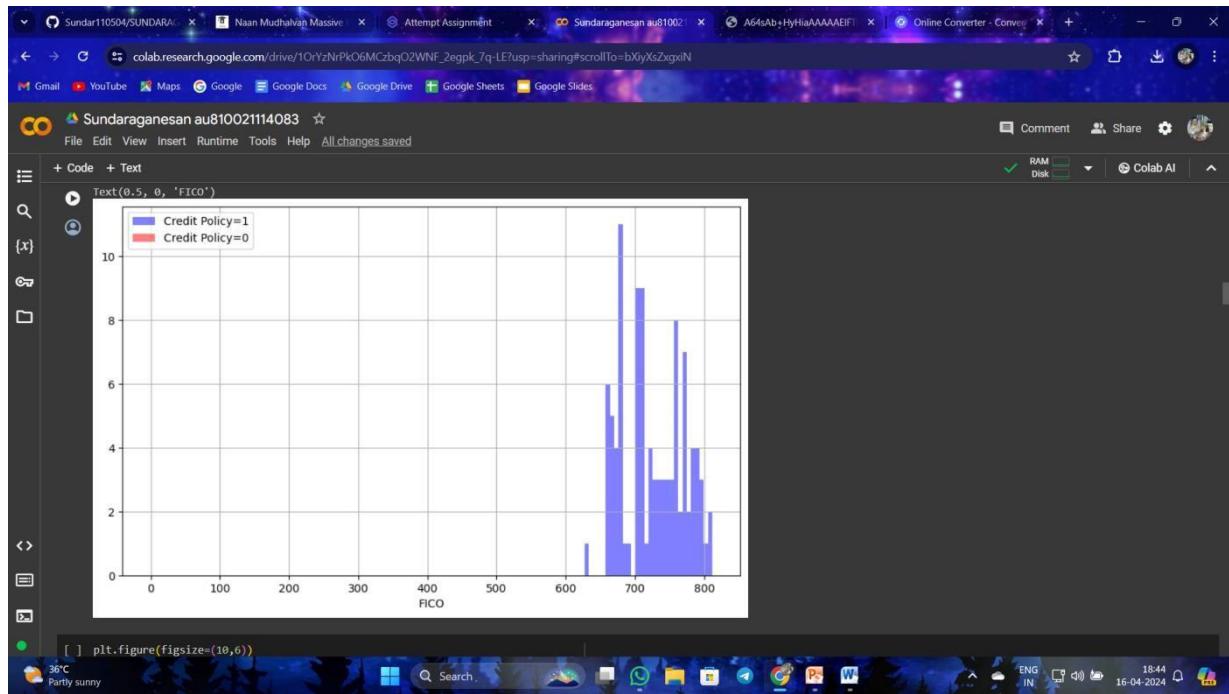
  credit.policy int.rate installment log.annual.inc dti fico days.with.cr.line revol.bal revol.util inq.last.6mths delinq.2yrs pub.rec not.fully.paid
  count      5.0 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.0.0.0
  mean       1.0 0.121020 337.888000 11.091236 13.694000 701.000000 3975.183333 20879.000000 53.420000 0.400000 0.200000 0.0.0.0
  std        0.0 0.017947 291.625932 0.416248 4.213494 27.248853 1267.137358 15424.476085 21.809791 0.547723 0.447214 0.0.0.0
  min        1.0 0.100800 102.920000 10.373491 8.100000 667.000000 2699.958333 3511.000000 25.600000 0.000000 0.000000 0.0.0.0
  25%       1.0 0.107100 162.340000 11.082143 11.630000 682.000000 2760.000000 4740.000000 39.500000 0.000000 0.000000 0.0.0.0
  50%       1.0 0.118900 228.220000 11.299732 14.290000 707.000000 4066.000000 28854.000000 52.100000 0.000000 0.000000 0.0.0.0
  75%       1.0 0.135700 366.860000 11.350407 14.970000 712.000000 4710.000000 33623.000000 73.200000 1.000000 0.000000 0.0.0.0
  max       1.0 0.142600 829.100000 11.350407 19.480000 737.000000 5639.958333 33667.000000 76.700000 1.000000 1.000000 0.0.0.0

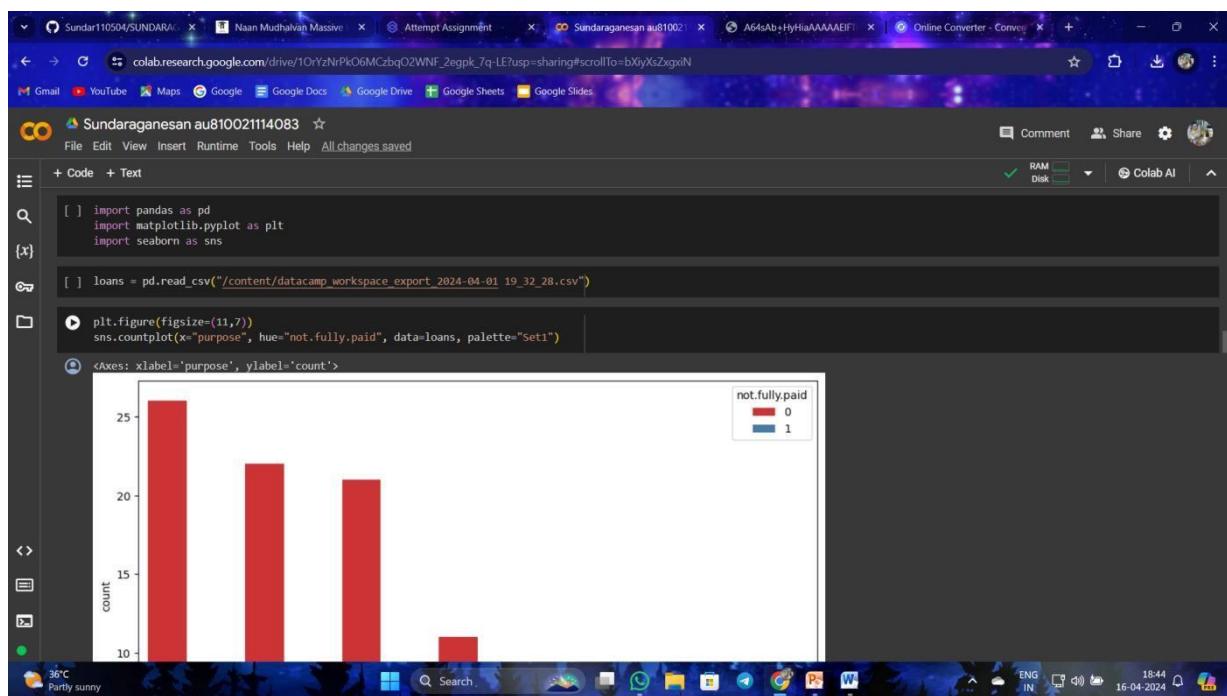
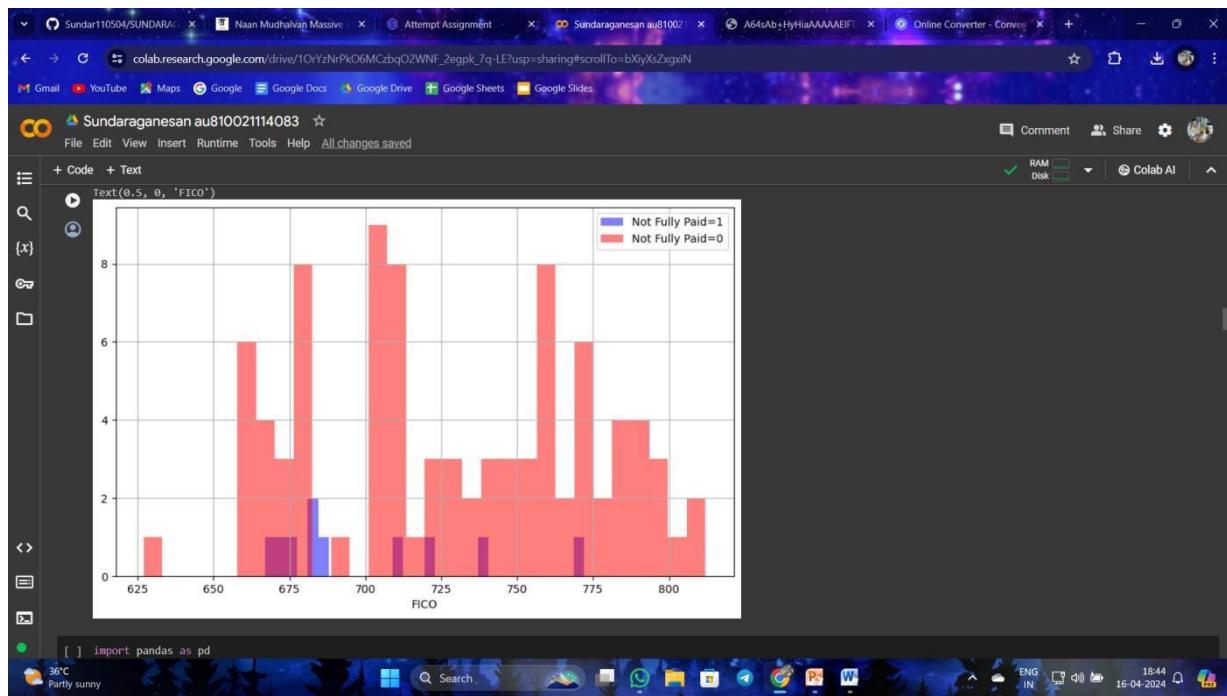
[ ] import matplotlib.pyplot as plt

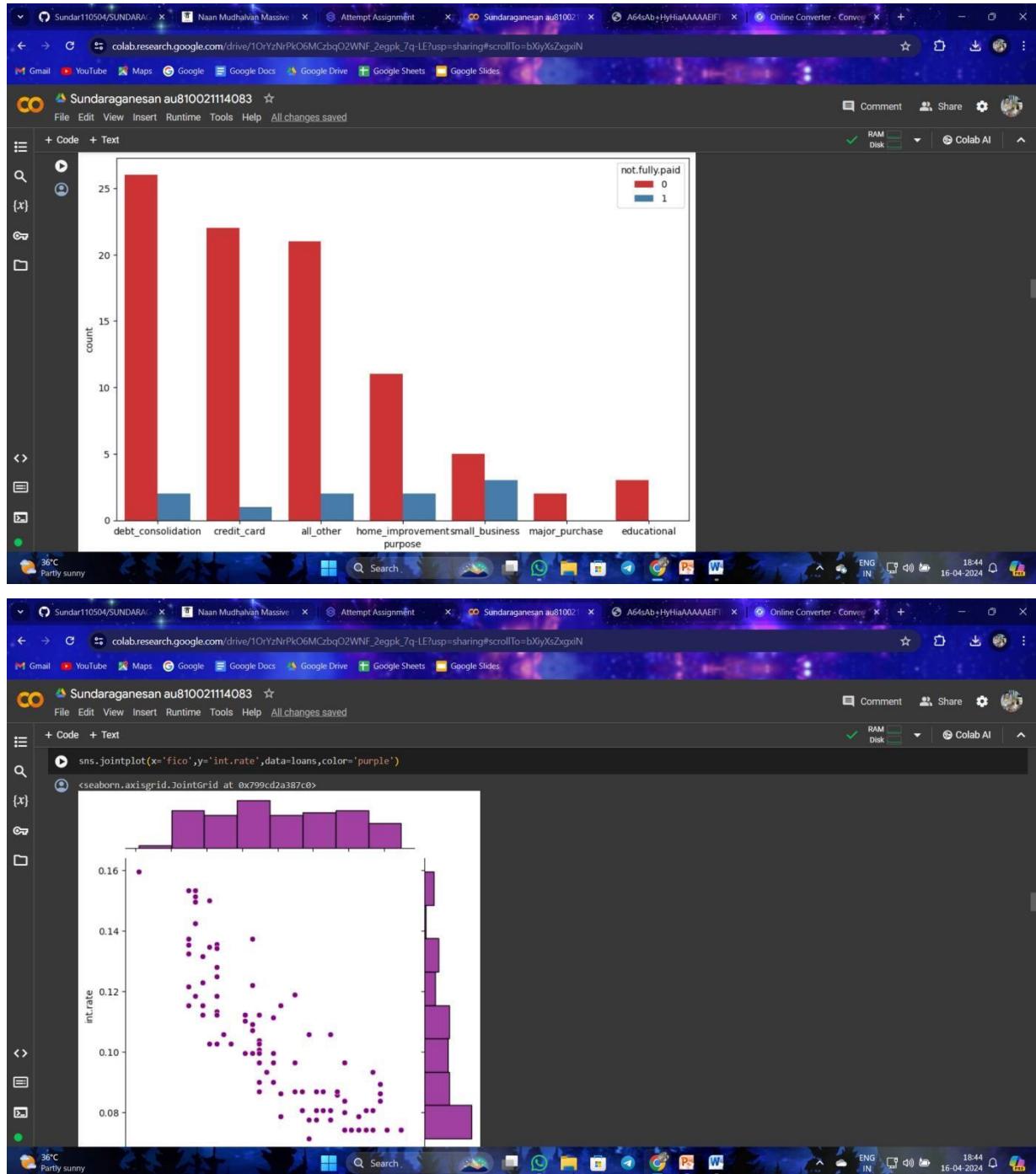
plt.figure(figsize=(10,6))
loans[loans['credit.policy']==1]['fico'].hist(alpha=0.5,color='blue',bins=30,label='Credit Policy=1')
loans[loans['credit.policy']==0]['fico'].hist(alpha=0.5,color='red',bins=30,label='Credit Policy=0')
plt.legend()
plt.xlabel('FICO')

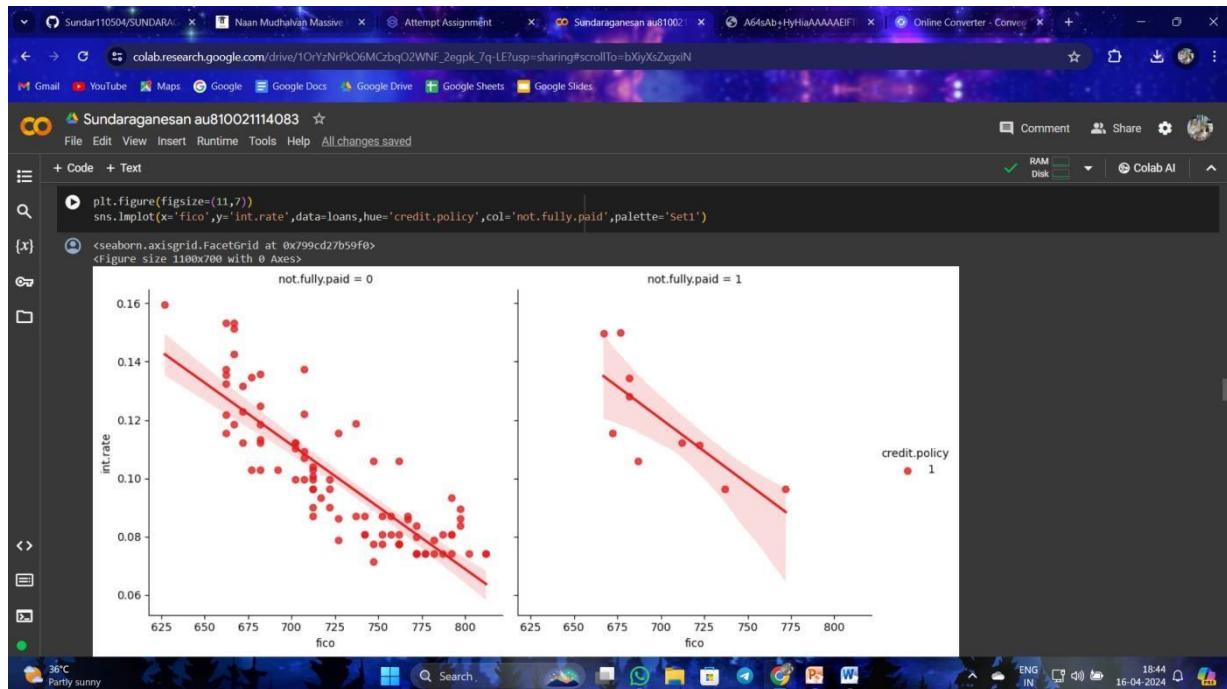
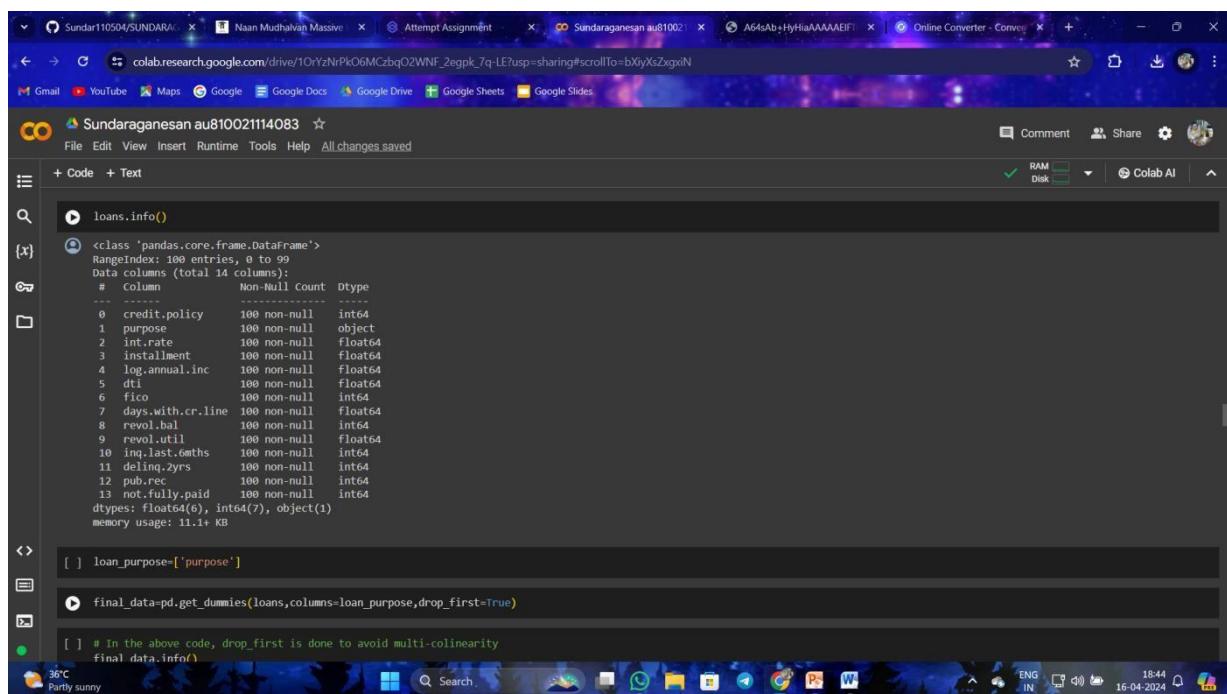
36°C Party sunny 18:44 16-04-2024
```









Code:

```
loans.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   credit.policy    100 non-null   int64  
 1   purpose        100 non-null   object  
 2   int.rate        100 non-null   float64 
 3   installment     100 non-null   float64 
 4   log.annual.inc  100 non-null   float64 
 5   dti             100 non-null   float64 
 6   fico            100 non-null   int64  
 7   days.with.cr.line 100 non-null   float64 
 8   revol.bal       100 non-null   int64  
 9   revol.util     100 non-null   float64 
 10  inq.last.6mths  100 non-null   int64  
 11  delinq.2yrs     100 non-null   int64  
 12  pub.rec         100 non-null   int64  
 13  not.fully.paid  100 non-null   int64  
dtypes: float64(6), int64(7), object(1)
memory usage: 11.1+ KB
```

Code:

```
[ ] loan_purpose=['purpose']

[ ] final_data=pd.get_dummies(loans,columns=loan_purpose,drop_first=True)

[ ] # In the above code, drop_first is done to avoid multi-collinearity
final_data.info()
```

Sundar110504/SUNDARA/... Naan Mudhalvan Massive... Attempt Assignment Sundaraganesan au81002114083 A64sAb+HyHiaAAAAEIFI... Online Converter - Conv...

File Edit View Insert Runtime Tools Help All changes saved

```
[ ] loan_purpose=['purpose']

{x} final_data=pd.get_dummies(loan,columns=loan_purpose,drop_first=True)

[ ] # In the above code, drop_first is done to avoid multi-collinearity
final_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   credit.policy   100 non-null    int64  
 1   int.rate        100 non-null    float64 
 2   installment     100 non-null    float64 
 3   log.annual.inc 100 non-null    float64 
 4   dti              100 non-null    float64 
 5   fico             100 non-null    int64  
 6   days.with.cr.line 100 non-null    float64 
 7   revol.bal       100 non-null    int64  
 8   revol.util      100 non-null    float64 
 9   inq.last.6mths  100 non-null    int64  
 10  delinq.zyrs    100 non-null    int64  
 11  pub.rec         100 non-null    int64  
 12  not.fully.paid 100 non-null    int64  
 13  purpose_credit_card 100 non-null    bool    
 14  purpose_debt_consolidation 100 non-null    bool    
 15  purpose_educational 100 non-null    bool    
 16  purpose_home_improvement 100 non-null    bool    
 17  purpose_major_purchase 100 non-null    bool    
 18  purpose_small_business 100 non-null    bool 
```

36°C Party sunny 18:44 16-04-2024

Sundar110504/SUNDARA/... Naan Mudhalvan Massive... Attempt Assignment Sundaraganesan au81002114083 A64sAb+HyHiaAAAAEIFI... Online Converter - Conv...

File Edit View Insert Runtime Tools Help All changes saved

```
[ ] final_data.head()

{x}
[0]   credit.policy  int.rate  installment  log.annual.inc  dti  fico  days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.zyrs  pub.rec  not.fully.paid  purpose_credit_card
[1]       1          1.1189      829.10     11.350407  19.48  737    5639.958333  28854      52.1          0          0          0          0          False
[2]       1          0.1071      228.22     11.082143  14.29  707    2760.000000  33623      76.7          1          0          0          0          True
[3]       1          0.1357      366.86     10.373491  11.63  682    4710.000000  3511      25.6          1          0          0          0          False
[4]       1          0.1008      162.34     11.350407  8.10  712    2699.958333  33667      73.2          1          0          0          0          False
```

```
[ ] import pandas as pd

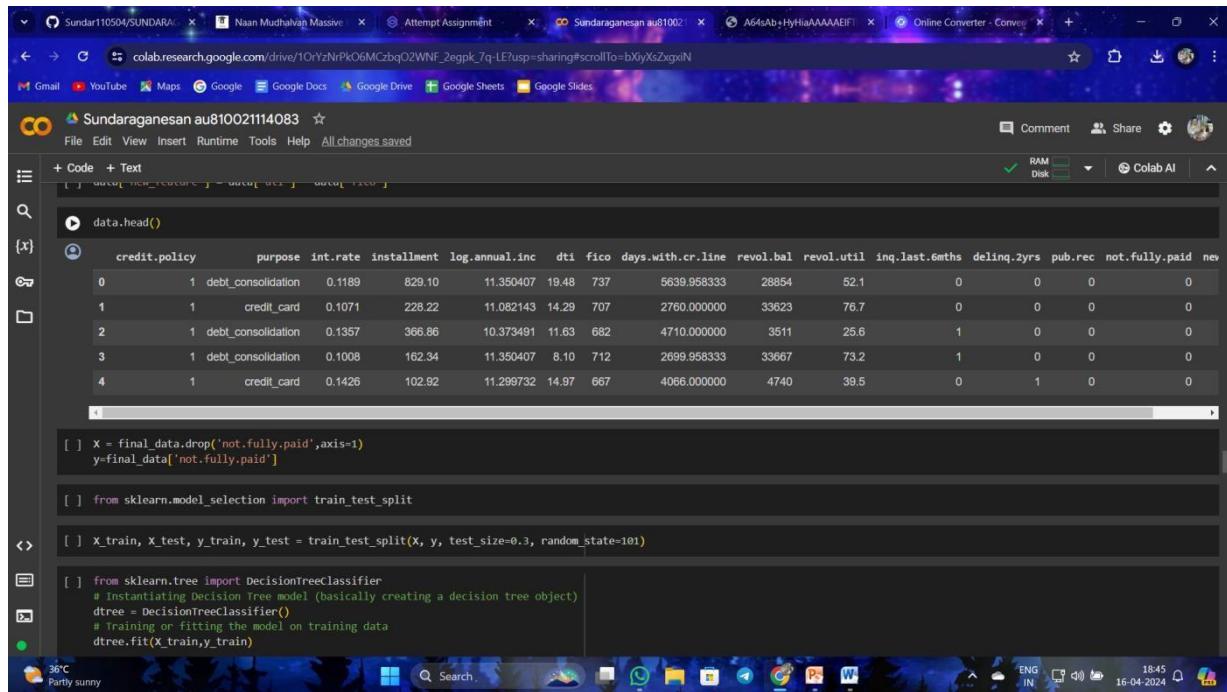
[ ] data = pd.read_csv('/content/datacamp_workspace_export_2024-04-01_19_32_28.csv')

[ ] data['new_feature'] = data['dti'] * data['fico']

[ ] data.head()

{x}
[0]   credit.policy  purpose  int.rate  installment  log.annual.inc  dti  fico  days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.zyrs  pub.rec  not.fully.paid  new_
[1]       1          debt_consolidation  0.1189      829.10     11.350407  19.48  737    5639.958333  28854      52.1          0          0          0          0          0
```

36°C Party sunny 18:45 16-04-2024

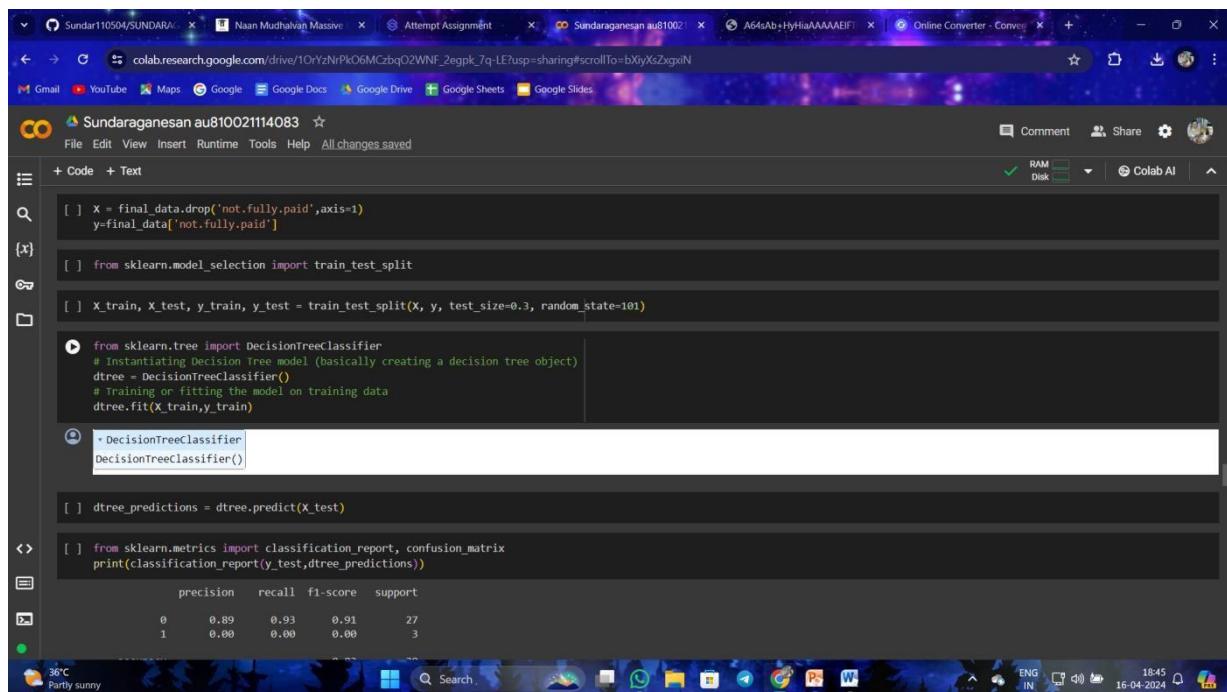


```
[ ] Sundar110504/SUNDARA/... Naan Mudhalvan Massive... Attempt Assignment Sundaraganesan au81002... A64sAb+HyHiaAAAAEIf... Online Converter - Conv... + - ×
← → ⌂ colab.research.google.com/drive/1OrYzNrPkO6MCzbqO2WNF_2egpk_7q-LE?usp=sharing#scrollTo=bXiyXsZxgxIN
Gmail YouTube Maps Google Google Docs Google Drive Google Sheets Google Slides
File Edit View Insert Runtime Tools Help All changes saved
Comment Share Colab AI RAM Disk
+ Code + Text
data.head()
credit.policy purpose int.rate installment log.annual.inc dti fico days.with.cr.line revol.bal revol.util inq.last.6mths delinq.2yrs pub.rec not.fully.paid new.cust.since...
0 debt_consolidation credit_card 0.1189 829.10 11.350407 19.48 737 5639.958333 28854 52.1 0 0 0 0
1 credit_card 0.1071 228.22 11.082143 14.29 707 2760.000000 33623 76.7 0 0 0 0
2 debt_consolidation 0.1357 366.86 10.373491 11.63 682 4710.000000 3511 25.6 1 0 0 0
3 debt_consolidation 0.1008 162.34 11.350407 8.10 712 2699.958333 33667 73.2 1 0 0 0
4 credit_card 0.1426 102.92 11.299732 14.97 667 4066.000000 4740 39.5 0 1 0 0
[ ]
[ ] X = final_data.drop('not.fully.paid',axis=1)
y=final_data['not.fully.paid']

[ ] from sklearn.model_selection import train_test_split

[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

[ ] from sklearn.tree import DecisionTreeClassifier
# Instantiating Decision Tree model (basically creating a decision tree object)
dtree = DecisionTreeClassifier()
# Training or fitting the model on training data
dtree.fit(X_train,y_train)
```



```
[ ] Sundar110504/SUNDARA/... Naan Mudhalvan Massive... Attempt Assignment Sundaraganesan au81002... A64sAb+HyHiaAAAAEIf... Online Converter - Conv... + - ×
← → ⌂ colab.research.google.com/drive/1OrYzNrPkO6MCzbqO2WNF_2egpk_7q-LE?usp=sharing#scrollTo=bXiyXsZxgxIN
Gmail YouTube Maps Google Google Docs Google Drive Google Sheets Google Slides
File Edit View Insert Runtime Tools Help All changes saved
Comment Share Colab AI RAM Disk
+ Code + Text
[ ] X = final_data.drop('not.fully.paid',axis=1)
y=final_data['not.fully.paid']

[ ] from sklearn.model_selection import train_test_split

[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

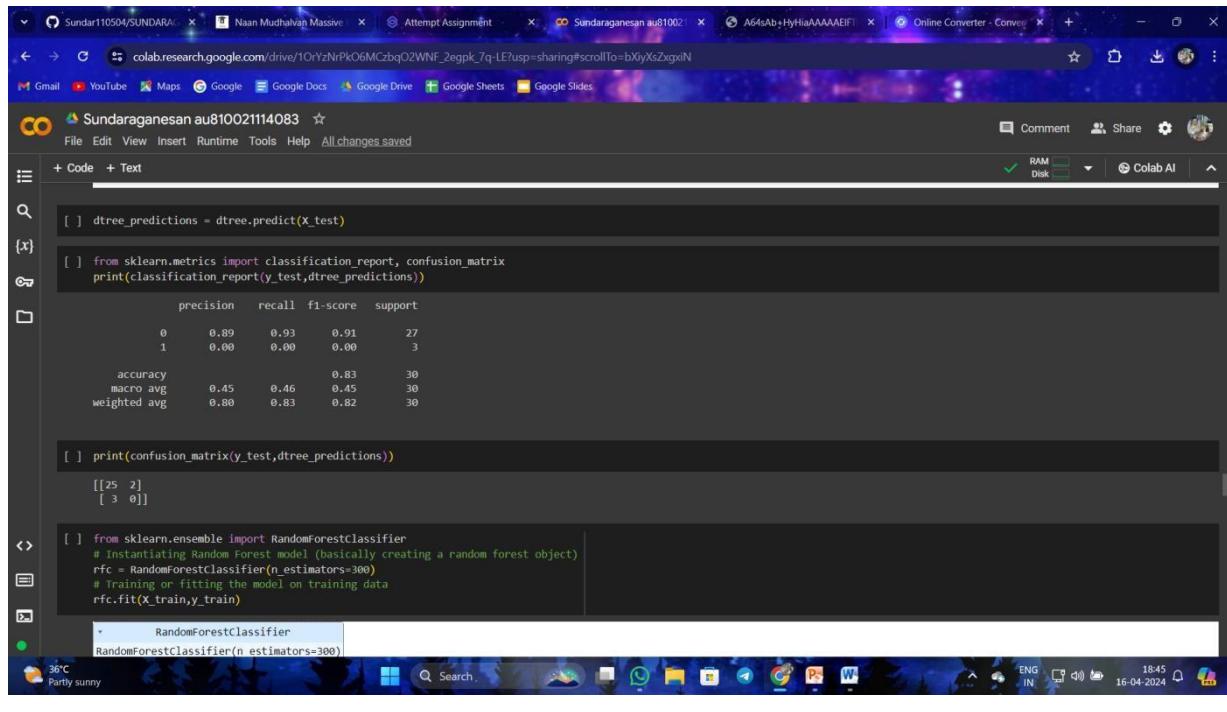
[ ] from sklearn.tree import DecisionTreeClassifier
# Instantiating Decision Tree model (basically creating a decision tree object)
dtree = DecisionTreeClassifier()
# Training or fitting the model on training data
dtree.fit(X_train,y_train)

+DecisionTreeClassifier
DecisionTreeClassifier()

[ ] dtree_predictions = dtree.predict(X_test)

[ ] from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,dtree_predictions))

precision recall f1-score support
0 0.89 0.93 0.91 27
1 0.00 0.00 0.00 3
```



```
[ ] dtree_predictions = dtree.predict(X_test)
[ ] from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,dtree_predictions))

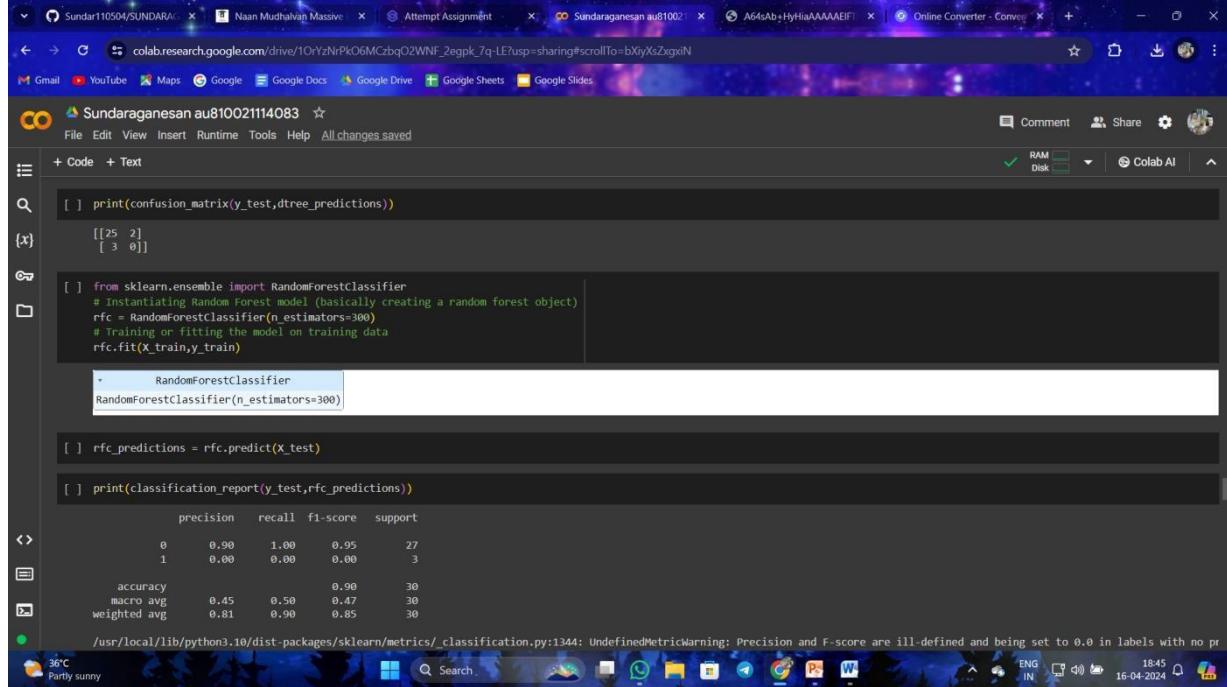
precision    recall   f1-score   support
          0       0.89      0.93      0.91      27
          1       0.00      0.00      0.00       3

accuracy                           0.83      30
macro avg       0.45      0.46      0.45      30
weighted avg    0.80      0.83      0.82      30

[ ] print(confusion_matrix(y_test,dtree_predictions))
[[25  2]
 [ 3  0]]
```

from sklearn.ensemble import RandomForestClassifier  
# Instantiating Random Forest model (basically creating a random forest object)  
rfc = RandomForestClassifier(n\_estimators=300)  
# Training or fitting the model on training data  
rfc.fit(X\_train,y\_train)

+ RandomForestClassifier  
RandomForestClassifier(n\_estimators=300)



```
[ ] print(confusion_matrix(y_test,dtree_predictions))
[[25  2]
 [ 3  0]]

[ ] from sklearn.ensemble import RandomForestClassifier
# Instantiating Random Forest model (basically creating a random forest object)
rfc = RandomForestClassifier(n_estimators=300)
# Training or fitting the model on training data
rfc.fit(X_train,y_train)

+ RandomForestClassifier
RandomForestClassifier(n_estimators=300)

[ ] rfc_predictions = rfc.predict(X_test)
[ ] print(classification_report(y_test,rfc_predictions))

precision    recall   f1-score   support
          0       0.90      1.00      0.95      27
          1       0.00      0.00      0.00       3

accuracy                           0.90      30
macro avg       0.45      0.50      0.47      30
weighted avg    0.81      0.90      0.85      30

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predictions (labels: [1]).
```

36°C Party sunny

Sundar110504/SUNDARA/... Naan Mudhalvan Massive... Attempt Assignment Sundaraganesan au81002114083 A64sAb+HyHiaAAAAEIFI... Online Converter - Conv...

File Edit View Insert Runtime Tools Help All changes saved

```
+ Code + Text
print(classification_report(y_test,rfc_predictions))
precision    recall   f1-score   support
          0       0.90      1.00      0.95      27
          1       0.00      0.00      0.00       3

   accuracy                           0.90
  macro avg       0.45      0.50      0.47      30
weighted avg       0.81      0.90      0.85      30

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no precision or recall (binary) average modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no precision or recall (macro) average modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no precision or recall (weighted) average modifier, msg_start, len(result))

[[27  0]
 [ 3  0]]
```

[ ] !pip install scikit-learn

Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)  
 Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)  
 Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)  
 Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.0)  
 Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.4.0)

36°C Party sunny 18:45 16-04-2024

Sundar110504/SUNDARA/... Naan Mudhalvan Massive... Attempt Assignment Sundaraganesan au81002114083 A64sAb+HyHiaAAAAEIFI... Online Converter - Conv...

File Edit View Insert Runtime Tools Help All changes saved

```
+ Code + Text
print(confusion_matrix(y_test,rfc_predictions))
[[27  0]
 [ 3  0]]
```

[ ] !pip install scikit-learn

Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)  
 Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)  
 Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)  
 Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.0)  
 Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.4.0)

[ ] from sklearn.model\_selection import GridSearchCV

[ ] # Define the best\_model variable (replace this with your actual model)
best\_model = ...

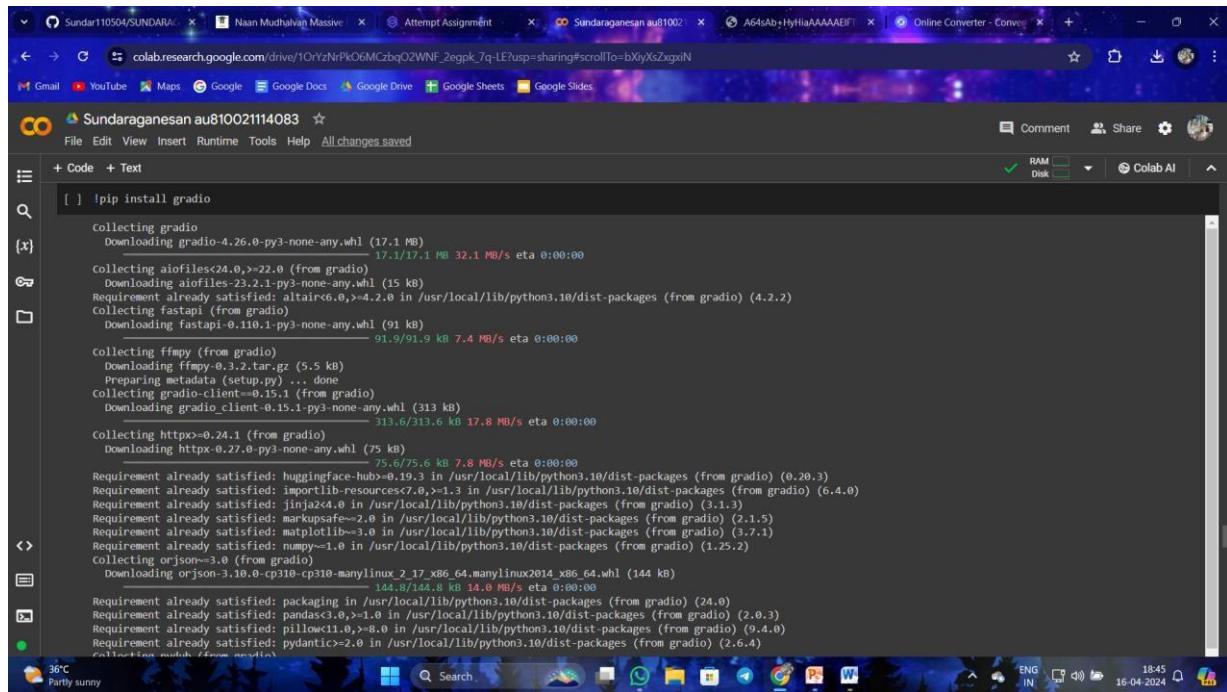
[ ] # Save the best model to disk
joblib.dump(best\_model, 'loan\_classifier.joblib')

[ ] ['loan\_classifier.joblib']

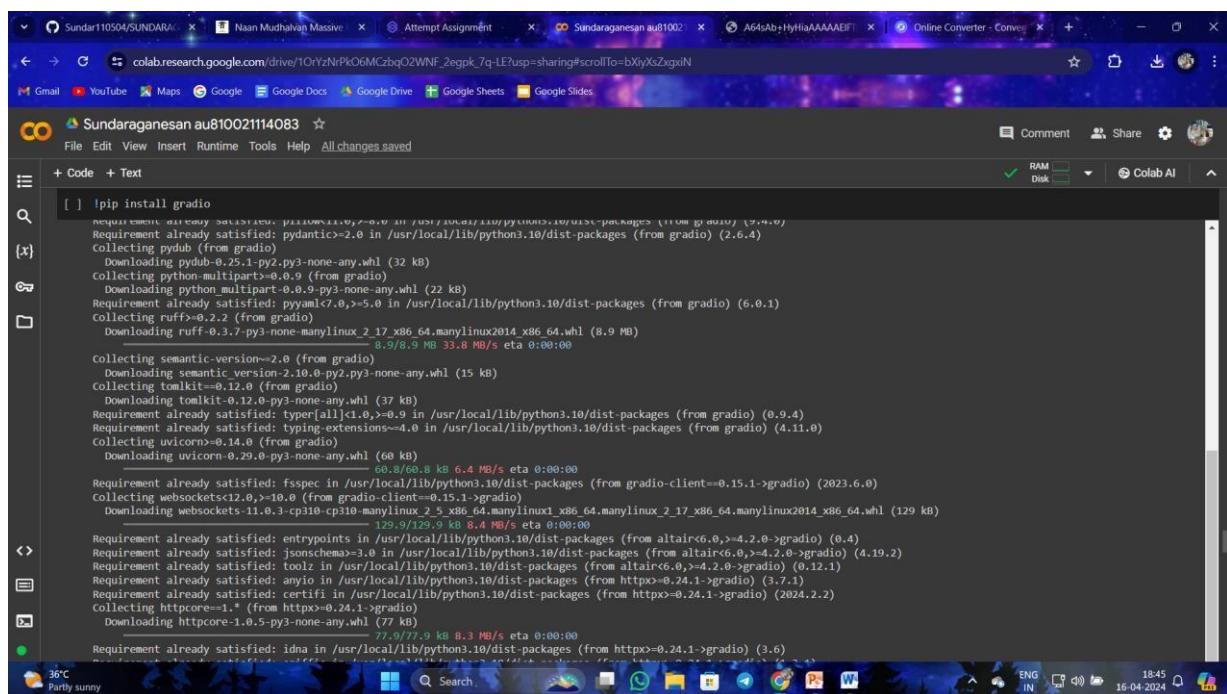
[ ] !pip install gradio

Collecting gradio
 Downloading gradio-4.26.0-py3-none-any.whl (17.1 MB)
 17.1/17.1 MB 32.1 MB/s eta 0:00:00
Collecting aiofiles<24.0,>22.0 (from gradio)
 Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)

36°C Party sunny 18:45 16-04-2024



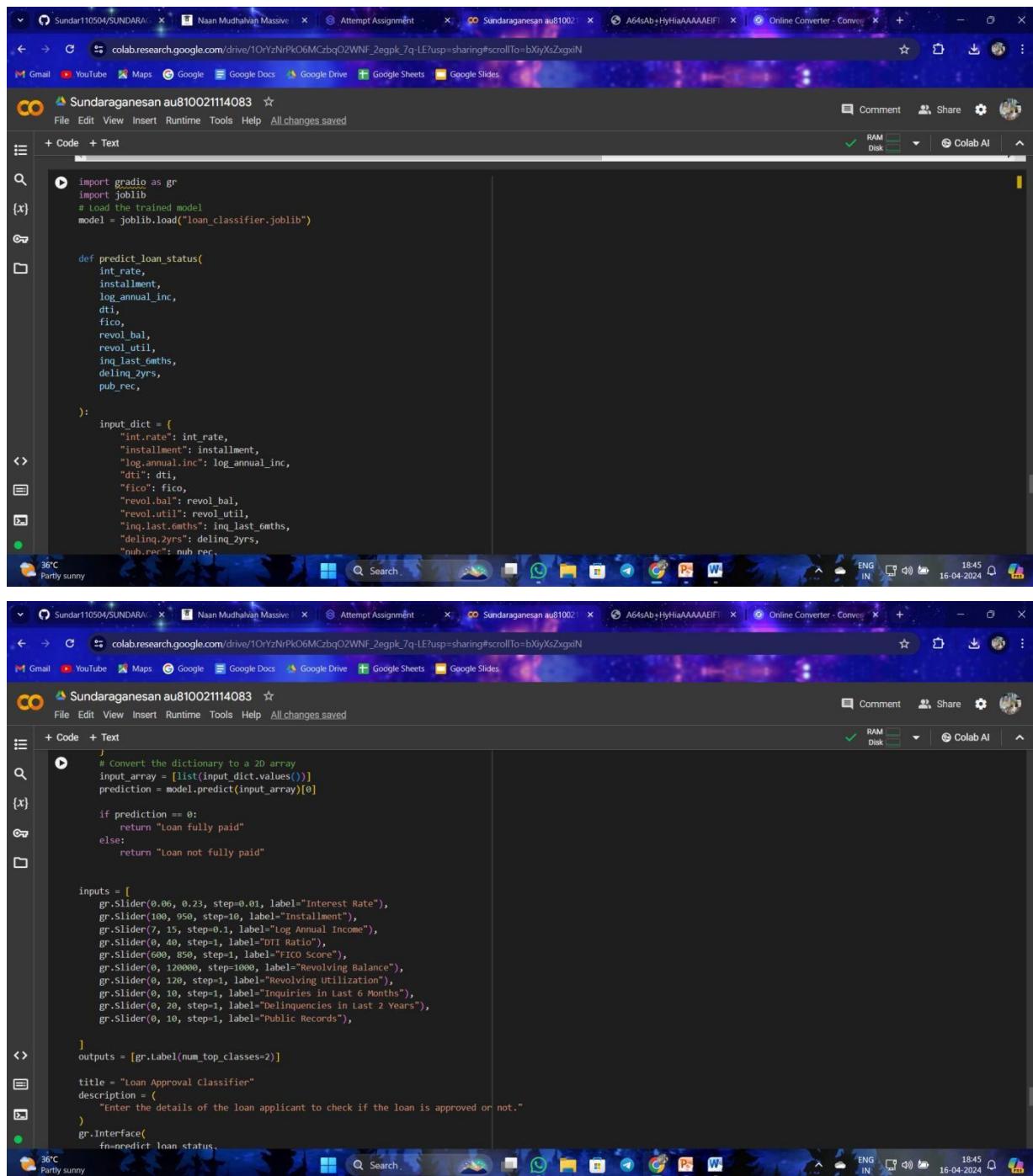
```
[ ] !pip install gradio
[+] pip install gradio
  Downloading gradio-4.26.0-py3-none-any.whl (17.1 MB)
    17.1/17.1 MB 32.1 MB/s eta 0:00:00
  Downloading aiofiles<24.0,>=22.0 (from gradio)
    Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.2.2)
  Collecting fastapi (from gradio)
    Downloading fastapi-0.110.1-py3-none-any.whl (91 kB)
      91.9/91.9 kB 7.4 MB/s eta 0:00:00
  Collecting fmpy (from gradio)
    Downloading fmpy-0.3.2.tar.gz (5.5 kB)
      Preparing metadata (setup.py) ... done
  Collecting gradio-Client<=0.15.1 (from gradio)
    Downloading gradio_client-0.15.1-py3-none-any.whl (313 kB)
      313.6/313.6 kB 17.8 MB/s eta 0:00:00
Collecting httpx>=0.24.1 (from gradio)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
    75.6/75.6 kB 7.8 MB/s eta 0:00:00
Requirement already satisfied: huggingface-hub<0.19.3 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.20.3)
Requirement already satisfied: importlib-resources<7.0,>=1.3 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.4.6)
Requirement already satisfied: jinjax2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.3)
Requirement already satisfied: markupsafe<2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.1.5)
Requirement already satisfied: matplotlib<3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Requirement already satisfied: numpy<1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.25.2)
Collecting orjson>=3.0 (from gradio)
  Downloading orjson-3.10.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (144 kB)
    144.5/144.5 kB 14.0 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (24.0)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.0.3)
Requirement already satisfied: pillow<11.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (9.4.0)
Requirement already satisfied: pydantic<2.6 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.6.4)
```



```
[ ] !pip install gradio
[+] pip install gradio
  Downloading pydub (from gradio)
    Downloading pydub-0.25.1-py3-py3-none-any.whl (32 kB)
  Collecting python-multipart<0.0.9 (from gradio)
    Downloading python_multipart-0.0.9-py3-none-any.whl (22 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.1)
  Collecting ruff<0.2.2 (from gradio)
    Downloading ruff-0.3.7-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.9 MB)
      8.9/8.9 kB 33.8 MB/s eta 0:00:00
  Collecting semantic-version<2.0 (from gradio)
    Downloading semantic_version-2.10.0-py3-py3-none-any.whl (15 kB)
  Collecting tomllib<=0.12.0 (from gradio)
    Downloading tomllib-0.12.0-py3-none-any.whl (37 kB)
Requirement already satisfied: typeforall<1.0,>=0.9 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.9.4)
Requirement already satisfied: typing-extensions<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.11.0)
  Collecting uicorn<0.14.0 (from gradio)
    Downloading uicorn-0.29.0-py3-none-any.whl (60 kB)
      60.8/60.8 kB 6.4 MB/s eta 0:00:00
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.15.1>gradio) (2023.6.0)
  Collecting websockets<12.0,>=10.0 (from gradio-client==0.15.1>gradio)
    Downloading websockets-11.0.3-cp310-cp310-manylinux_2_15_x86_64.manylinux1_x86_64.manylinux2014_x86_64.whl (129 kB)
      129.9/129.9 kB 8.4 MB/s eta 0:00:00
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0>gradio) (0.4)
Requirement already satisfied: jsonschema<3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0>gradio) (4.19.2)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0>gradio) (0.12.1)
Requirement already satisfied: ario in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1>gradio) (3.7.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1>gradio) (2024.2.2)
  Collecting httpcore<1.* (from httpx>=0.24.1>gradio)
    Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
      77.9/77.9 kB 8.3 MB/s eta 0:00:00
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1>gradio) (3.6)
```

```
[ ] pip install gradio
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.15.1>gradio) (2023.6.0)
Collecting websockets<12.0,>=10.8 (from gradio-client==0.15.1>gradio)
  Downloading websockets-11.0.3-cp310-cp310-manylinux_2_x86_64_manylinux2_17_x86_64_manylinux2014_x86_64.whl (129 kB)
    129/129.9 kB 8.4 MB/s eta 0:00:00
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0>gradio) (0.4)
Requirement already satisfied: jsonschema<3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0>gradio) (4.19.2)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0>gradio) (0.12.1)
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages (from httpx<0.24.1>gradio) (3.7.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<0.24.1>gradio) (2024.2.2)
Collecting httpcore==1.* (from httpx<0.24.1>gradio)
  Downloading httpcore-1.8.5-py3-none-any.whl (77 kB)
    77/77.9 kB 8.3 MB/s eta 0:00:00
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from httpx<0.24.1>gradio) (3.6)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from httpx<0.24.1>gradio) (1.3.1)
Collecting h11<0.15,>=0.13 (from httpcore<1.*>httpx<0.24.1>gradio)
  Downloading h11-0.14.6-py3-none-any.whl (59 kB)
    58/3/58.3 kB 6.9 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<0.19.3>gradio) (3.13.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<0.19.3>gradio) (2.31.0)
Requirement already satisfied: tqdm=>4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<0.19.3>gradio) (4.66.2)
Requirement already satisfied: contourpy<1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.0>gradio) (1.2.1)
Requirement already satisfied: cycler=<0.19 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.0>gradio) (0.12.1)
Requirement already satisfied: fonttools=<4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.0>gradio) (4.51.0)
Requirement already satisfied: kiwisolver=<1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.0>gradio) (1.4.5)
Requirement already satisfied: pyparsing=<2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.0>gradio) (3.1.2)
Requirement already satisfied: python-dateutil=<2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.0>gradio) (2.8.2)
Requirement already satisfied: pytz=>2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0>gradio) (2023.4)
Requirement already satisfied: tzdata=>2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0>gradio) (2024.1)
Requirement already satisfied: annotated-types=<0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<2.0>gradio) (0.6.0)
Requirement already satisfied: pydantic-core=>2.16.3 in /usr/local/lib/python3.10/dist-packages (from pydantic<2.0>gradio) (2.16.3)
Requirement already satisfied: click<9.0.8,>=1.1 in /usr/local/lib/python3.10/dist-packages (from typer[all]<1.0,>=0.9>gradio) (8.1.7)
Collecting colorama<5.0,>=0.4.3 (from typetall[1.0,>=0.9>gradio]
  Downloading colorama-0.4.3-py3-none-any.whl (5 kB)
    5/5 kB 1.0 MB/s eta 0:00:00
```

```
Sundaraganeshan au810021114083 ★
File Edit View Insert Runtime Tools Help All changes saved
+ Code & Text
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (0.6.0)
Requirement already satisfied: pydantic-core==2.16.3 in /usr/local/lib/python3.10/dist-packages (from pydantic>2.0->gradio) (2.16.3)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typy[all]<1.0,>=0.9->gradio) (8.1.7)
Requirement already satisfied: colorama<5.0.8,>=0.4.3 (from typy[all]<1.0,>=0.9->gradio)
  Downloading colorama-0.4.6-py2-py3-none-any.whl (25 kB)
Requirement already satisfied: shellingham<0.0.0,>=1.3.0 (from typy[all]<1.0,>=0.9->gradio)
  Downloading shellingham-1.5.4-py2-py3-none-any.whl (9.8 kB)
Requirement already satisfied: rich<14.0.0,>=10.11.0 in /usr/local/lib/python3.10/dist-packages (from typy[all]<1.0,>=0.9->gradio) (13.7.1)
Requirement already satisfied: starlette<0.38.0,>=0.37.2 (from fastapi->gradio)
  Downloading starlette-0.37.2-py3-none-any.whl (71 kB)
71.9 KB 7.4 MB/s eta 0:00:00
Requirement already satisfied: attrs>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.01.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.34.0)
Requirement already satisfied: rpdf-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.18.0)
Requirement already satisfied: six<1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib<3.0,>=0.9->gradio) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typy[all]<1.0,>=0.9->gradio) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typy[all]<1.0,>=0.9->gradio) (2.16.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx>0.24.1->gradio) (1.2.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.19.3->gradio) (3.3.2)
Requirement already satisfied: urllib3<3.1,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.19.3->gradio) (2.0.7)
Requirement already satisfied: murl<0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich<14.0.0,>=10.11.0->typy[all]<1.0,>=0.9->gradio) (0.1.2)
Building wheels for collected packages: fffmpeg
  Building wheel for fffmpeg (setup.py) ... done
  Created wheel for fffmpeg: filename=fffmpeg-0.3.2-py3-none-any.whl size=5584 sha256=1d0db98c923be65cbdf47d266ee6cf0e90284105bd7f10b56adea6fd5d0d2805
  Stored in directory: /root/.cache/pip/wheels/bd/e5/9a/671fc6cdce0/d4418dfc0592f8df512b26d/a0829c2a23d81
Successfully built fffmpeg
Installing collected packages: pydub, fffmpeg, websockets, tomkit, shellingham, semantic-version, ruff, python-multipart, orjson, h11, colorama, aiofiles, uvicorn, starlette, httpcore
Successfully installed aiofiles-23.2.1 colorama-0.4.6 fastapi-0.110.1 fffmpeg-0.3.2 gradio-4.26.0 gradio-client-0.15.1 h11-0.14.0 httpcore-1.0.5 https://0.27.0 orjson-3.10.0 pydub-0.25.1
+
import gradio as gr
import joblib
# load the trained model
```



The image shows two screenshots of a Google Colab notebook titled "Sundaraganesan au81002114083".

**Code Cell 1:**

```

import gradio as gr
import joblib
# Load the trained model
model = joblib.load("loan_classifier.joblib")

def predict_loan_status(
    int_rate,
    installment,
    log_annual_inc,
    dti,
    fico,
    revol_bal,
    revol_util,
    inq_last_6mths,
    delinq_2yrs,
    pub_rec,
):
    input_dict = {
        "int.rate": int_rate,
        "installment": installment,
        "log.annual.inc": log_annual_inc,
        "dti": dti,
        "fico": fico,
        "revol.bal": revol_bal,
        "revol.util": revol_util,
        "inq.last.6mths": inq_last_6mths,
        "delinq.2yrs": delinq_2yrs,
        "pub.rec": pub_rec,
    }

```

**Code Cell 2:**

```

# Convert the dictionary to a 2D array
input_array = [list(input_dict.values())]
prediction = model.predict(input_array)[0]

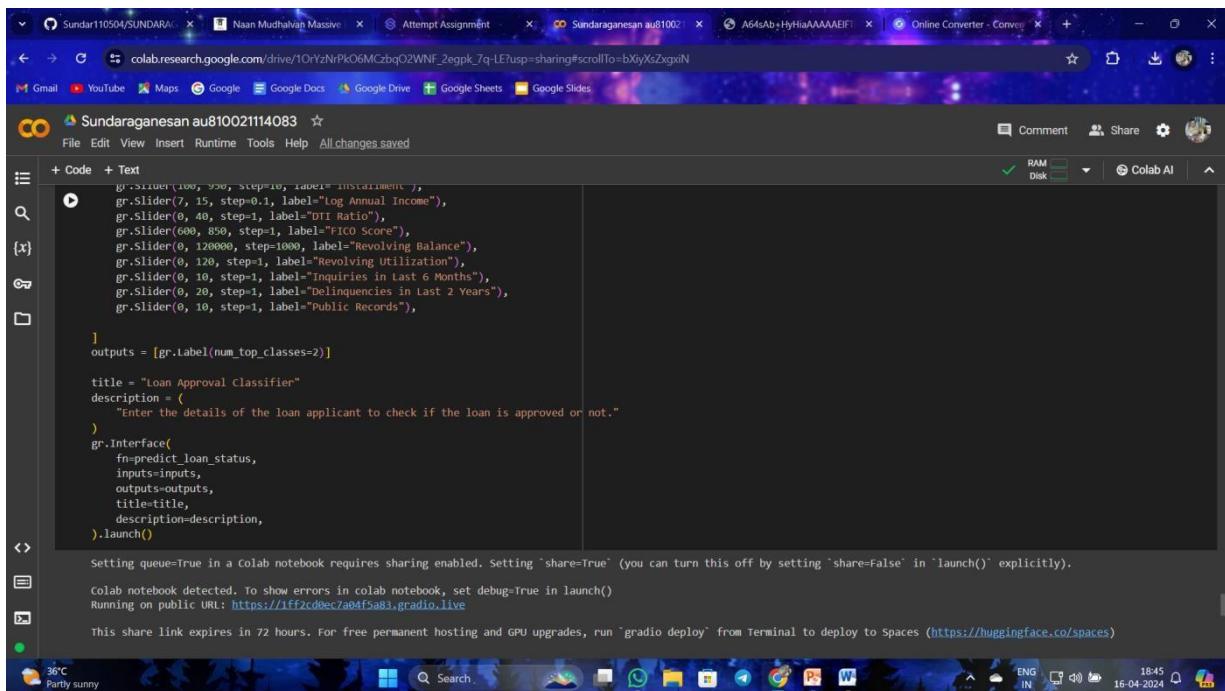
if prediction == 0:
    return "Loan fully paid"
else:
    return "Loan not fully paid"

inputs = [
    gr.Slider(0.06, 0.23, step=0.01, label="Interest Rate"),
    gr.Slider(100, 950, step=10, label="Installment"),
    gr.Slider(7, 15, step=0.1, label="Log Annual Income"),
    gr.Slider(0, 40, step=1, label="DTI Ratio"),
    gr.Slider(600, 850, step=1, label="FICO Score"),
    gr.Slider(0, 120000, step=1000, label="Revolving Balance"),
    gr.Slider(0, 120, step=1, label="Revolving Utilization"),
    gr.Slider(0, 10, step=1, label="Inquiries in Last 6 Months"),
    gr.Slider(0, 20, step=1, label="Delinquencies in Last 2 Years"),
    gr.Slider(0, 10, step=1, label="Public Records"),
]

outputs = [gr.Label(num_top_classes=2)]

title = "Loan Approval Classifier"
description = (
    "Enter the details of the loan applicant to check if the loan is approved or not."
)
gr.Interface(
    fn=predict_loan_status,

```



The screenshot shows a Google Colab notebook titled "Sundaraganesan au81002114083". The code cell contains Python code for creating a Gradio-based interface to predict loan status based on various financial metrics. It includes sliders for Annual Income, DTI Ratio, FICO Score, Revolving Utilization, and other factors, along with a dropdown for the number of top classes. The interface is titled "Loan Approval classifier" and has a description explaining it's for checking if a loan is approved.

```

gr.slider(100, 950, step=10, label="Installment"),
gr.slider(7, 15, step=0.1, label="Log Annual Income"),
gr.slider(0, 40, step=1, label="DTI Ratio"),
gr.slider(600, 850, step=1, label="FICO Score"),
gr.slider(0, 120000, step=1000, label="Revolving Balance"),
gr.slider(0, 120, step=1, label="Revolving Utilization"),
gr.slider(0, 10, step=1, label="Inquiries in Last 6 Months"),
gr.slider(0, 20, step=1, label="Delinquencies in Last 2 Years"),
gr.slider(0, 10, step=1, label="Public Records"),
]

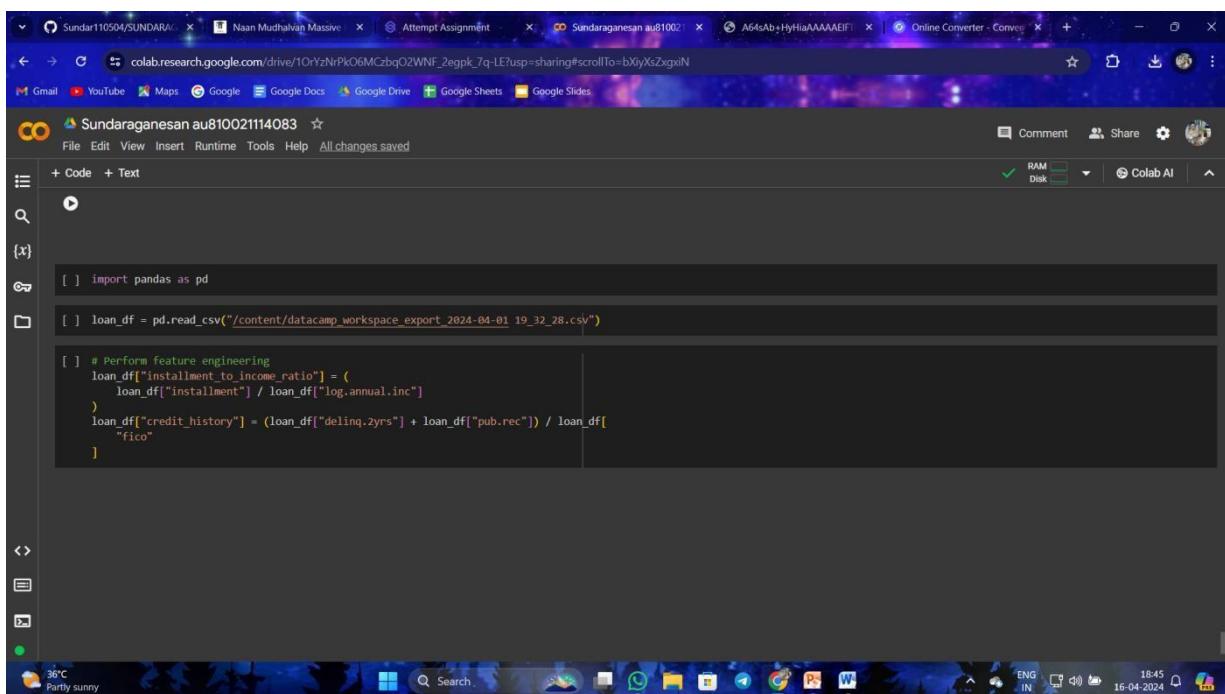
outputs = [gr.Label(num_top_classes=2)]

title = "Loan Approval classifier"
description = (
    "Enter the details of the loan applicant to check if the loan is approved or not."
)
gr.Interface(
    fn=predict_loan_status,
    inputs=inputs,
    outputs=outputs,
    title=title,
    description=description,
).launch()

Setting queue=True in a Colab notebook requires sharing enabled. Setting 'share=True' (you can turn this off by setting 'share=False' in 'launch()' explicitly).
  
```

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()  
 Running on public URL: <https://1ff2c0be7a0f5a83.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (<https://huggingface.co/spaces>)



The second screenshot shows the same Colab notebook with a new code cell. This cell imports pandas and reads a CSV file named "content/datacamp\_workspace\_export\_2024-04-01\_19\_32\_28.csv". The code then performs feature engineering by calculating the "installment\_to\_income\_ratio" (installment / log.annual.inc) and the "credit\_history" (delinq.2yrs + pub.rec) ratio.

```

import pandas as pd

loan_df = pd.read_csv("/content/datacamp_workspace_export_2024-04-01_19_32_28.csv")

# Perform feature engineering
loan_df["installment_to_income_ratio"] = (
    loan_df["installment"] / loan_df["log.annual.inc"]
)
loan_df["credit_history"] = (loan_df["delinq.2yrs"] + loan_df["pub.rec"]) / loan_df[
    "fico"
]
  
```

## APP INTERFERENCE/ PROJECT RESULT

The end-to-end data science project resulted in the creation of an interactive chatbot that provides personalized loan eligibility predictions based on user input. Users can easily

access this service through various messaging platforms, making it convenient and user-friendly. The integration of ChatGPT enhances the user experience by providing a conversational interface, making the process intuitive and accessible to a wider audience. Overall, the project demonstrates the potential of combining machine learning with natural language processing for practical applications like financial services.

## CONCLUSION

In conclusion, the implementation of an end-to-end data project utilizing ChatGPT for a loan dataset offers a robust solution for enhancing customer engagement and service efficiency in the lending domain. By leveraging natural language processing capabilities, this project enables

seamless communication between users and the loan application system, providing instant assistance and guidance throughout the loan application process. Through meticulous data preprocessing, model training, integration, and deployment, this project ensures the delivery of accurate and relevant responses to user queries, ultimately facilitating a streamlined and userfriendly experience. With continuous monitoring and updates, this system remains adaptive and responsive to evolving user needs, thereby maximizing its effectiveness in serving borrowers and optimizing loan management processes.

## FUTURE SCOPE

Looking ahead, the future scope for an end-to-end data project utilizing ChatGPT for a loan dataset is promising and multifaceted. Advancements in natural language processing and machine learning techniques will enable the development of even more sophisticated and personalized loan application systems. Integration of additional data sources, such as social media profiles or financial transaction history, could enrich the model's understanding of

borrower preferences and risk profiles, leading to more accurate loan decisions. Furthermore, incorporating voice recognition capabilities could enhance user accessibility and convenience, catering to a broader range of users. Collaboration with financial institutions and regulatory bodies may foster the adoption of standardized processes and compliance measures within the system, ensuring trust and reliability. Ultimately, the future holds immense potential for leveraging ChatGPT in loan management, driving innovation, and improving financial inclusion for individuals and businesses alike.

## REFERENCES

1. Project Github link, Ramar Bose , 2024
2. Project video recorded link (youtube/github), Ramar Bose , 2024
3. Project PPT & Report github link, Ramar Bose , 2024



GIT Hub Link of Project Code:

<https://github.com/Praveen810021127704/Praveen-M.git>