In [243…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder     # importing the necessary
```

In [245…
```
df = pd.read_csv('/Users/sagarbanjara/Downloads/Takeo projects/bda62_ Sagar/
```

In [247…
```
df
```

Out[247…

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIn
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	
...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	
610	LP002979	Male	Yes	3+	Graduate	No	
611	LP002983	Male	Yes	1	Graduate	No	
612	LP002984	Male	Yes	2	Graduate	No	
613	LP002990	Female	No	0	Graduate	Yes	

614 rows × 13 columns

In [249…
```
df.info()     #checking the data structure
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [251…
```
missing_values = df.isnull().sum()    #checking if there are missing values
```

In [253…
```
missing_values
```

Out[253…
```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

In [255…
```
df.dropna(inplace=True)   #removing the rows if thewre are any nan values
```

In [257…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 480 entries, 1 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            480 non-null    object 
 1   Gender             480 non-null    object 
 2   Married            480 non-null    object 
 3   Dependents         480 non-null    object 
 4   Education          480 non-null    object 
 5   Self_Employed      480 non-null    object 
 6   ApplicantIncome    480 non-null    int64  
 7   CoapplicantIncome  480 non-null    float64
 8   LoanAmount         480 non-null    float64
 9   Loan_Amount_Term   480 non-null    float64
 10  Credit_History     480 non-null    float64
 11  Property_Area      480 non-null    object 
 12  Loan_Status        480 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 52.5+ KB
```

In [259…

```
# df = pd.get_dummies(df, drop_first=True)
```

In [261…

```
label_encoders = {}       #encoding the categorial column
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

df.head()
```

Out[261…

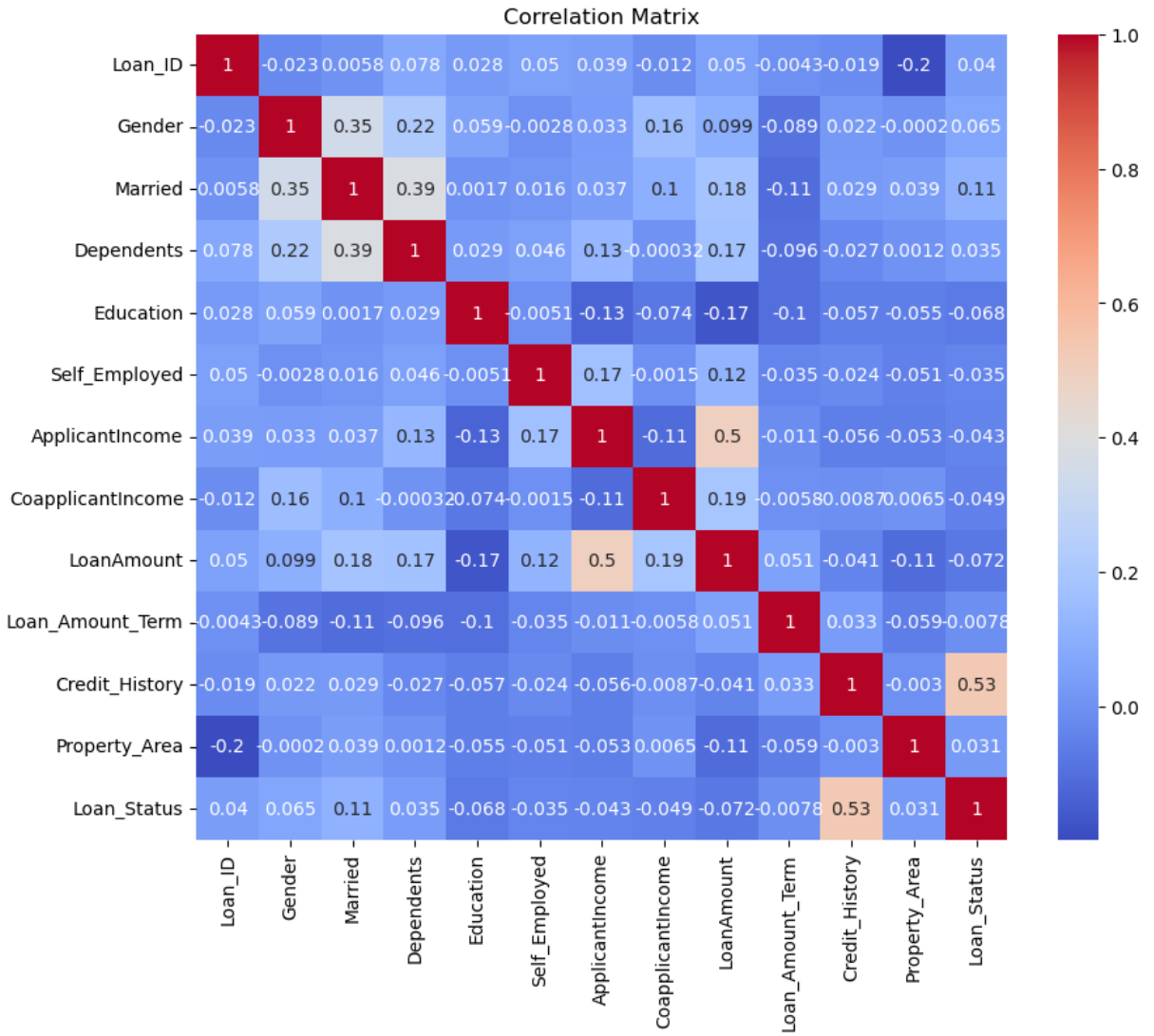
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
1	0	1	1	1	0	0	458
2	1	1	1	0	0	1	300
3	2	1	1	0	1	0	258
4	3	1	0	0	0	0	600
5	4	1	1	2	0	1	541

In [263…

```
plt.figure(figsize=(10, 8))   #coding for correlation matrix

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")

plt.title("Correlation Matrix")
plt.show()
```



In [265…
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report          #AGAIN IMPORTING t
from sklearn.metrics import accuracy_score, precision_score, recall_score, f
```

In [267…
```
X = df.drop("Loan_Status", axis = 1)      # defining the value for x and y
y = df["Loan_Status"]
```

In [269…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran
model = RandomForestClassifier(n_estimators=100, random_state=42)     #choos
model.fit(X_train, y_train)    #connecting the data
y_pred = model.predict(X_test)
```

In [271…
```
y_pred
```

Out[271…
```
array([1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1])
```

In [273…
```
print(x.dtypes.value_counts())
```

```
int64      8
float64    4
Name: count, dtype: int64
```

In [275…
```
baseline_model = LogisticRegression()   #choosing the next model
baseline_model.fit(X_train, y_train)
y_pred = baseline_model.predict(X_test)
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.p
y:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

In [277…
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Baseline Model Performance:")    #checking performance of accuracy,pr
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Baseline Model Performance:
Accuracy: 0.7777777777777778
Precision: 0.7615384615384615
Recall: 0.99
F1 Score: 0.8608695652173913
```

In [279…
```
from sklearn.feature_selection import RFE
```

In [281…
```
rfe = RFE(model, n_features_to_select=7)  # Selecting the next features
X_rfe = rfe.fit_transform(X, y)
selected_features_rfe = X.columns[rfe.support_]
rf_pred = rfe.predict(X_test)
```

In [283…
```
print("Selected Features using RFE:", selected_features_rfe.tolist())
```

```
Selected Features using RFE: ['Loan_ID', 'ApplicantIncome', 'CoapplicantInco
me', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Property_Area']
```

In [285…
```
rf_pred
```

Out[285…
```
array([1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1])
```

In [ ]:

In [ ]:

In [ ]: