untitled1

November 16, 2023

[1]:
```
#20200604   Nour Ayman Abdullah
#20200183   Rawan Mostafa Mahmoud
#20200231   Salma Hany Gamal
#20200640   Yomna Sayed Mohamed
#20200056   Ahmed Hany Fathey

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error
```

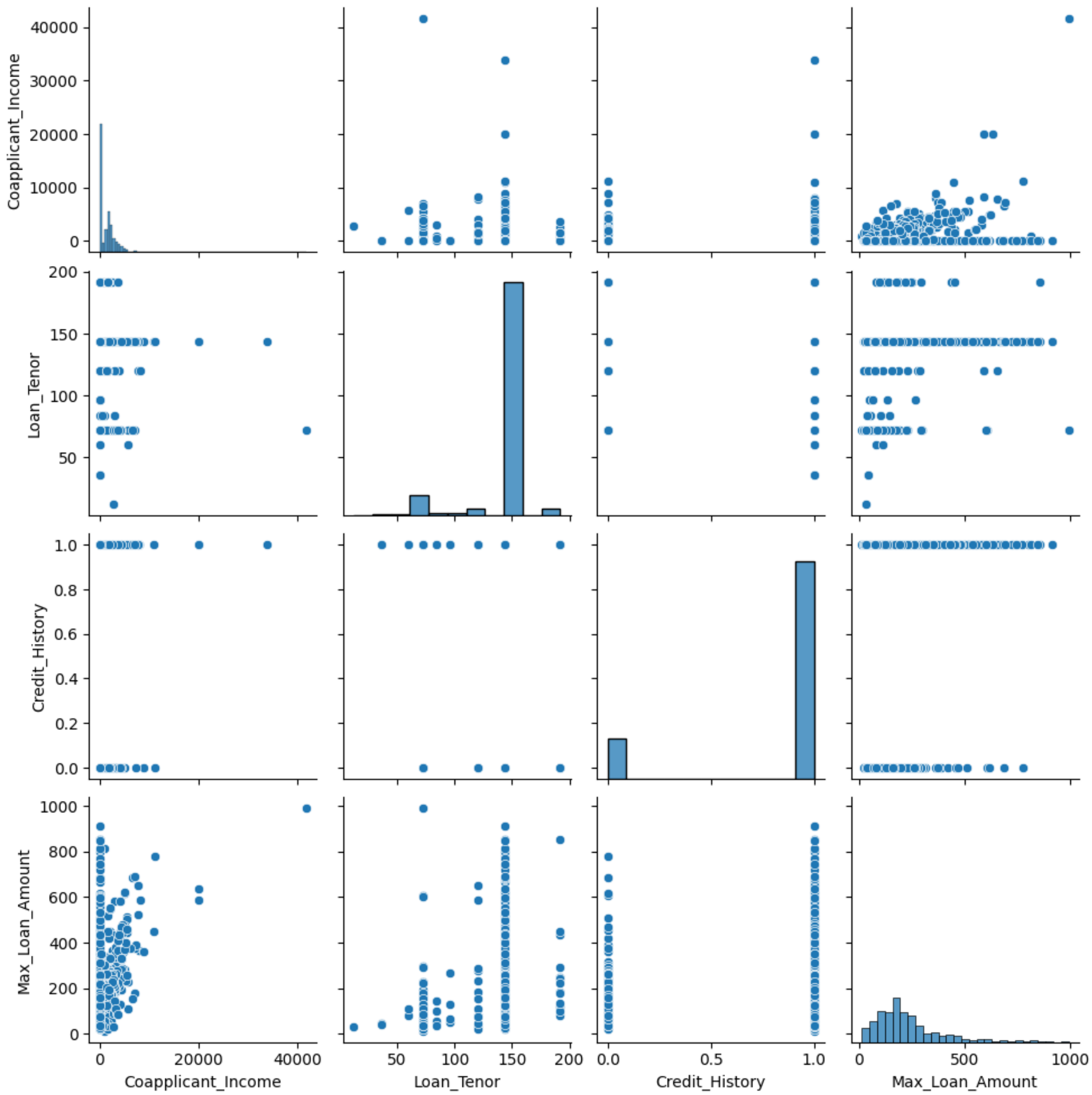
[2]:
```
# Load the dataset
loan_old = pd.read_csv("loan_old.csv")
print(loan_old)
# Display basic information about the dataset
print(loan_old.info())

# i) Check for missing values
print("Missing Values:\n", loan_old.isnull().sum())

# ii) Check the type of each feature
print("Data Types:\n", loan_old.dtypes)

# iii) Check if numerical features have the same scale
print("Statistical Summary:\n", loan_old.describe())

# iv) Visualize a pairplot between numerical columns
sns.pairplot(loan_old.select_dtypes(include=['float64']))
plt.show()
```

```
     Loan_ID  Gender Married Dependents     Education  Income  \
```

```
0    LP001002    Male      No          0      Graduate    5849
1    LP001003    Male     Yes          1      Graduate    4583
2    LP001005    Male     Yes          0      Graduate    3000
3    LP001006    Male     Yes          0  Not Graduate    2583
4    LP001008    Male      No          0      Graduate    6000
..        …       …     …          …            …     …
609  LP002978  Female      No          0      Graduate    2900
610  LP002979    Male     Yes         3+      Graduate    4106
611  LP002983    Male     Yes          1      Graduate    8072
612  LP002984    Male     Yes          2      Graduate    7583
613  LP002990  Female      No          0      Graduate    4583

     Coapplicant_Income  Loan_Tenor  Credit_History Property_Area  \
0                   0.0       144.0             1.0         Urban
1                1508.0       144.0             1.0         Rural
2                   0.0       144.0             1.0         Urban
3                2358.0       144.0             1.0         Urban
4                   0.0       144.0             1.0         Urban
..                  …           …           …            …
609                 0.0       144.0             1.0         Rural
610                 0.0        72.0             1.0         Rural
611               240.0       144.0             1.0         Urban
612                 0.0       144.0             1.0         Urban
613                 0.0       144.0             0.0     Semiurban

     Max_Loan_Amount Loan_Status
0                NaN           Y
1             236.99           N
2              81.20           Y
3             179.03           Y
4             232.40           Y
..               …          …
609            76.16           Y
610            33.47           Y
611           348.92           Y
612           312.18           Y
613           160.98           N

[614 rows x 12 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 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   Income              614 non-null    int64
 6   Coapplicant_Income  614 non-null    float64
 7   Loan_Tenor          599 non-null    float64
 8   Credit_History      564 non-null    float64
 9   Property_Area       614 non-null    object
 10  Max_Loan_Amount     589 non-null    float64
 11  Loan_Status         614 non-null    object
dtypes: float64(4), int64(1), object(7)
memory usage: 57.7+ KB
None
Missing Values:
 Loan_ID                0
Gender                13
Married                3
Dependents            15
Education              0
Income                 0
Coapplicant_Income     0
Loan_Tenor            15
Credit_History        50
Property_Area          0
Max_Loan_Amount       25
Loan_Status            0
dtype: int64
Data Types:
 Loan_ID                object
Gender                 object
Married                object
Dependents             object
Education              object
Income                  int64
Coapplicant_Income    float64
Loan_Tenor            float64
Credit_History        float64
Property_Area          object
Max_Loan_Amount       float64
Loan_Status            object
dtype: object
Statistical Summary:
              Income  Coapplicant_Income  Loan_Tenor  Credit_History  \
count    614.000000          614.000000  599.000000      564.000000
mean    5403.459283         1621.245798  137.689482        0.842199
std     6109.041673         2926.248369   23.366294        0.364878
min      150.000000            0.000000   12.000000        0.000000
25%     2877.500000            0.000000  144.000000        1.000000
50%     3812.500000         1188.500000  144.000000        1.000000
75%     5795.000000         2297.250000  144.000000        1.000000
```

```
max     81000.000000         41667.000000   192.000000          1.000000

       Max_Loan_Amount
count       589.000000
mean        230.499474
std         161.976967
min          12.830000
25%         123.990000
50%         190.370000
75%         276.500000
max         990.490000
```



[3]:
```python
# c) Preprocess the data

# i) Remove records containing missing values
loan_old_cleaned = loan_old.dropna()

# ii) Separate features and targets
# ii) Separate features and targets
X = loan_old_cleaned.drop(columns=['Loan_ID','Max_Loan_Amount', 'Loan_Status'])

y_amount = loan_old_cleaned['Max_Loan_Amount']
y_status = loan_old_cleaned['Loan_Status']

# iii) Shuffle and split into training and testing sets
X_train, X_test, y_amount_train, y_amount_test, y_status_train, y_status_test =␣
↪train_test_split(
    X, y_amount, y_status, test_size=0.2, random_state=90
)

# iv) Categorical feature encoding
encoder = LabelEncoder()
categorical_cols = X.select_dtypes(include=['object']).columns
for col in categorical_cols:
    X_train[col] = encoder.fit_transform(X_train[col])
    X_test[col] = encoder.transform(X_test[col])

# v) Categorical targets encoding
le_status = LabelEncoder()
y_status_train_encoded = le_status.fit_transform(y_status_train)
y_status_test_encoded = le_status.transform(y_status_test)

# vi) Numerical feature standardization
scaler = StandardScaler()
numerical_cols = X.select_dtypes(include=['float64']).columns
X_train[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])
```

[4]:
```python
# d) Fit a linear regression model to predict the loan amount
# -> Use sklearn's linear regression
linear_reg_model = LinearRegression()
linear_reg_model.fit(X_train, y_amount_train)
```

[4]:
```
LinearRegression()
```

[5]:
```python
# e) Evaluate the linear regression model using sklearn's R2 score and Mean␣
↪Squared Error
y_amount_pred = linear_reg_model.predict(X_test)
r2_score_result = r2_score(y_amount_test, y_amount_pred)
```

```
mse_result = mean_squared_error(y_amount_test, y_amount_pred)
print(f"R^2 Score for Linear Regression: {r2_score_result}")
print(f"Mean Squared Error for Linear Regression: {mse_result}")
```

```
R^2 Score for Linear Regression: 0.8277587164823796
Mean Squared Error for Linear Regression: 4174.342813848165
```

[6]:
```
# f) Fit a logistic regression model to predict the loan status
# -> Implement logistic regression from scratch using gradient descent

# Define the sigmoid function
def sigmoid(z):
    # Clip the input to the exponential function within a certain range
    z = np.clip(z, -700, 700)
    return 1 / (1 + np.exp(-z))


# Implement logistic regression from scratch using gradient descent
def logistic_regression(X, y, learning_rate=0.01, epochs=1000):
    m, n = X.shape
    X = np.c_[np.ones((m, 1)), X]  # add intercept term

    theta = np.zeros(n + 1)

    for epoch in range(epochs):
        z = np.dot(X, theta)
        h = sigmoid(z)
        gradient = np.dot(X.T, (h - y)) / m
        gradient = np.clip(gradient, -10, 10)
        theta -= learning_rate * gradient

    return theta

# Train logistic regression model
theta = logistic_regression(X_train, y_status_train_encoded)
print(theta)
```

```
[-0.30570428 -0.14272862  0.02294591 -0.00743804 -0.17261335  0.
  0.0238193   0.03760257  1.72572313 -0.18551576]
```

[7]:
```
# g) Write a function (from scratch) to calculate the accuracy of the model
def predict(X, theta):
    X = np.c_[np.ones((X.shape[0], 1)), X]  # add intercept term
    probabilities = sigmoid(np.dot(X, theta))
    #print(probabilities)
    predictions = (probabilities >= 0.5).astype(int)
    return predictions
```

```python
# Calculate accuracy on the test set
X_test_intercept = np.c_[np.ones((X_test.shape[0], 1)), X_test]
y_status_pred = predict(X_test, theta)
accuracy = np.mean(y_status_pred == y_status_test_encoded)
#print(accuracy)
print(f"Accuracy for Logistic Regression: {accuracy*100} %")
print(y_status_pred)
```

```
Accuracy for Logistic Regression: 67.96116504854369 %
[0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1
 0 1 1 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1
 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1]
```

[8]:
```python
#loan_new_cleaned=pd.read_csv("loan_new.csv")

#loan_new_cleaned.to_csv(output,index=false)
#x=loan_new_cleaned.iloc[:, 1:].values
#print(2)
loan_new= pd.read_csv("loan_new.csv") #h
# i) Remove records containing missing values
loan_new_cleaned = loan_new.dropna()
loan_new_cleaned = loan_new_cleaned.drop(columns=['Loan_ID'])

# iv) Categorical feature encoding
encoder = LabelEncoder()
categorical_cols = loan_new_cleaned.select_dtypes(include=['object']).columns
for col in categorical_cols:
    loan_new_cleaned[col] = encoder.fit_transform(loan_new_cleaned[col])

# v) Categorical targets encoding
le_status = LabelEncoder()
y_status_encoded = le_status.fit_transform(y_status)

# vi) Numerical feature standardization
scaler = StandardScaler()
numerical_cols = loan_new_cleaned.select_dtypes(include=['float64']).columns
loan_new_cleaned[numerical_cols] = scaler.
 ↪fit_transform(loan_new_cleaned[numerical_cols])
```

[9]:
```python
print(loan_new_cleaned)

new_loan_amount_pred = linear_reg_model.predict(loan_new_cleaned)
new_loan_status_pred = predict(loan_new_cleaned, theta)

# Display the predictions
print("Predicted Loan Amounts:")
```

```python
print(new_loan_amount_pred)

print("\nPredicted Loan Status:")
print(new_loan_status_pred)
```

```
     Gender  Married  Dependents  Education  Income  Coapplicant_Income  \
0         1        1           0          0    5720                   0
1         1        1           1          0    3076                1500
2         1        1           2          0    5000                1800
4         1        0           0          1    3276                   0
5         1        1           0          1    2165                3422
..      …      …         …        …     …                 …
361       1        1           1          0    2269                2167
362       1        1           3          1    4009                1777
363       1        1           0          0    4158                 709
365       1        1           0          0    5000                2393
366       1        0           0          0    9200                   0

     Loan_Tenor  Credit_History  Property_Area
0      0.251600         0.46082              2
1      0.251600         0.46082              2
2      0.251600         0.46082              2
4      0.251600         0.46082              2
5      0.251600         0.46082              2
..          …             …            …
361    0.251600         0.46082              1
362    0.251600         0.46082              2
363    0.251600         0.46082              2
365    0.251600         0.46082              0
366   -2.880653         0.46082              0

[314 rows x 9 columns]
Predicted Loan Amounts:
[ 2.50661505e+02  9.39085821e+04  1.12706591e+05  1.73008865e+02
  2.13956986e+05  1.45955261e+02  2.03943360e+02  3.53984209e+02
  1.50112832e+05  1.68247856e+02  9.48927790e+04  4.22849850e+02
  2.17559535e+02  2.51606695e+02  1.82424907e+05  2.10269468e+04
  4.94942264e+05  2.16789510e+05  1.01362818e+05 -1.97592764e+01
  1.74007040e+02  2.73910191e+05  1.49965632e+06  2.34601063e+05
  1.48844767e+05  1.48313238e+02  5.15071600e+04  1.05349069e+05
  1.69374234e+05  9.64575380e+04  1.80568551e+02  2.51874857e+05
  1.74609377e+05  2.51751446e+02  1.22846036e+05  5.13579615e+04
  1.83830793e+02  1.26604981e+05  1.62727529e+05  1.84396967e+02
  1.18996049e+05  1.59365155e+04  1.06825288e+05  1.92459813e+05
  6.25360000e+04  2.26734871e+02  1.62847095e+02  2.05255118e+02
  2.21170312e+05  2.29100715e+05  8.18739675e+01  1.07690284e+05
  1.85285319e+04  2.33586141e+02  9.38712505e+04  2.18451386e+02
```

2.24056182e+05	2.09338515e+02	1.89978566e+02	3.08169559e+02
3.19138658e+05	2.23921623e+02	1.10926829e+05	1.31054363e+05
2.24090416e+05	2.14816854e+05	1.48078585e+05	2.80065712e+02
2.53439249e+05	2.96464639e+02	6.91216063e+04	1.98238542e+03
3.19080235e+02	3.35747592e+02	5.31352357e+01	4.99932785e+05
1.19971864e+05	1.87969568e+02	1.26087304e+05	2.35075110e+02
4.78581120e+02	1.56430381e+05	1.03072950e+05	1.45992759e+05
2.72781427e+02	2.62614503e+05	2.03264937e+05	3.65958384e+02
2.31065918e+02	2.56521913e+02	2.07976492e+02	1.99167165e+01
8.06005707e+04	2.46835770e+02	2.26691960e+02	8.45565722e+04
2.19515409e+02	1.77988905e+02	2.12368989e+05	2.97261292e+05
1.10051497e+05	1.66700780e+05	2.19444734e+05	1.82060562e+02
2.37794449e+05	9.76727616e+04	4.00925844e+05	4.02178827e+02
2.29171601e+02	2.59354042e+02	3.04724716e+02	1.78607288e+01
7.95403871e+04	2.03760886e+02	9.49924890e+04	1.87340438e+02
6.88608468e+01	2.13036663e+02	2.42373922e+02	1.82931851e+05
2.28580322e+02	1.87482808e+05	3.32582692e+04	1.08242154e+02
3.68803615e+02	1.80436247e+02	3.17965711e+02	4.50360853e+04
2.53475703e+02	2.69412268e+05	9.11477405e+04	2.45168800e+02
1.47604159e+05	2.51509913e+02	1.54661541e+05	2.65516263e+05
1.43365260e+05	1.58278066e+05	3.08552589e+02	2.66143387e+05
1.75781807e+02	2.45317649e+02	1.74597423e+02	1.44932407e+02
1.26141386e+05	1.98059782e+05	1.83529548e+05	2.02572167e+02
2.06766405e+02	2.28998714e+02	2.52777352e+02	1.77765941e+05
1.13874202e+05	2.88541348e+05	2.08413033e+05	2.58535619e+02
1.64671582e+05	1.84229627e+05	2.23864762e+02	2.44952750e+05
7.42003194e+04	3.55052653e+02	4.42066734e+02	2.48192223e+05
8.66445221e+01	2.06714870e+02	1.82146799e+05	1.25013981e+05
1.68446857e+05	2.51751446e+02	2.28874093e+02	1.00126803e+05
4.39043728e+04	9.02651340e+04	4.50935937e+02	3.40411982e+04
1.48511022e+05	7.21256991e+04	1.51195250e+05	1.62693842e+05
6.78612873e+04	1.98474113e+02	1.50838284e+02	1.50199115e+05
2.61823735e+02	2.63389312e+02	1.33196232e+05	1.53503199e+05
1.36716331e+05	1.96903587e+05	1.33851353e+05	1.35654742e+05
2.81876613e+02	1.85018139e+02	2.29582169e+02	1.55547823e+02
1.10635964e+05	2.38201850e+05	1.72314340e+05	2.26645892e+02
9.06907931e+05	9.95807091e+04	2.97193860e+02	1.11863811e+05
1.73472408e+05	3.25736100e+05	8.73947277e+05	6.95250325e+04
3.21723069e+02	1.87626451e+02	9.49853715e+04	5.19356682e+04
1.06642097e+05	1.38765484e+02	1.35574720e+05	1.99419752e+02
2.89066855e+01	2.76025784e+04	2.05267776e+02	6.42581643e+04
9.92244476e+04	1.80481259e+02	2.05367642e+02	2.68108723e+05
2.32938259e+05	1.63820508e+05	9.70335243e+04	6.07289199e+02
7.01826142e+04	8.77544802e+04	2.23565658e+05	4.71280347e+04
2.06771903e+02	1.52837504e+05	2.44867158e+05	7.51348185e+02
2.77471556e+04	1.77989535e+02	9.37290226e+04	2.70734950e+02
8.30838412e+01	5.70522012e+04	1.28688687e+05	2.46986935e+02
3.35171834e+02	7.29263445e+05	3.32766409e+02	2.71773127e+05

```
  2.29876538e+02  2.50246934e+05  1.25167704e+05  1.77202498e+05
  1.87734954e+02  1.25102429e+05  1.60405105e+05  2.76481701e+02
  2.01554749e+02  1.90580671e+02  2.32594552e+02  3.00597426e+02
  2.60152430e+02  4.39146524e+04  3.19579480e+02  2.12939958e+02
  2.98080940e+02  2.93868066e+05  2.38394884e+02  3.29569632e+05
  2.55241515e+02  1.32709019e+02  5.50596851e+04  2.16948476e+02
  1.81320257e+05  1.71254573e+05  9.65371226e+04  1.10213236e+05
  1.49050542e+02  3.12931080e+05  2.55556752e+02  3.87823816e+04
  4.39782992e+04  8.84707664e+04  2.60086145e+02  3.65443680e+02
  1.69937734e+05  2.20484283e+02  1.22050768e+05  1.71657132e+02
  9.75081895e+04  2.09699108e+05  1.45975552e+05  3.01862322e+05
  2.67325949e+05  4.55328868e+01  2.28538905e+02  2.92984776e+02
  3.30315151e+02  1.23034104e+05  1.62508910e+02  8.64108262e+04
  1.86228043e+04  3.49612442e+02  5.71110724e+04  2.35741786e+05
  1.17587007e+04  7.61012284e+04  1.21168797e+05  1.65976071e+02
  1.66849433e+05  1.35570218e+05  1.11235209e+05  4.45096515e+04
  1.49764558e+05  1.90394977e+02]

Predicted Loan Status:
[1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1
 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 0
 0 1 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 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 1 1 0 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1
 0 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1]
```

[ ]:

[ ]:

[ ]:

[ ]: