

Machine Learning

Assignment 1

Name	ID
Eman Fathy Abo Alhassan	20200105
Sara Ahmed Sayed	20200214
Fatma Mahmoud Ramadan	20201134
Basmala Magdy Mohamed	20201045
Esraa Osama Mohamed	20201014

a-load the dataset

.b-Data analysis

The screenshot shows two Jupyter Notebook sessions side-by-side. Both sessions have the same project structure:

- Project folder: ass1 C:\User\...
- Files: main.py, ass1ml.py, ass1mledit.py, Lab 4 (complete code).ipynb, Untitled.ipynb

The code in both sessions is identical, demonstrating data analysis steps:

```
# a) Load the "loan_old.csv" dataset
data = pd.read_csv('loan_old.csv')

# b) Data Analysis
# i) Check for missing values
missing_values = data.isnull().sum()
print(missing_values[missing_values > 0])

Gender      13
Married      3
Dependents   15
Loan_Tenor   15
Credit_History  50
Max_Loan_Amount  25
dtype: int64

# ii) Check the type of each feature
print("\nData Types:\n")
for column in data.columns:
    if data[column].dtype == 'int64' or data[column].dtype == 'float64':
        print(f"{column: <20}: numerical")
    else:
        print(f"{column: <20}: categorical")

# iii) Check whether numerical features have the same scale
numerical_features = ['Income', 'Coapplicant_Income', 'Loan_Tenor', 'Credit_History', 'Max_Loan_Amount']
```

The output from the second session shows the data types for each column:

Column	Type
Loan_ID	categorical
Gender	categorical
Married	categorical
Dependents	categorical
Education	categorical
Income	numerical
Coapplicant_Income	numerical
Loan_Tenor	numerical
Credit_History	numerical
Property_Area	categorical
Max_Loan_Amount	numerical
Loan_Status	categorical

```
# iii) Check whether numerical features have the same scale
numerical_features = ['Income', 'Coapplicant_Income', 'Loan_Tenor', 'Credit_History', 'Max_Loan_Amount']

# Calculate the ranges
ranges = data[numerical_features].max() - data[numerical_features].min()

# Calculate the standard deviations
std_devs = data[numerical_features].std()

for feature in numerical_features:
    print(f"Range of {feature}: {ranges[feature]}")
    print(f"Standard deviation of {feature}: {std_devs[feature]}")
    print()

# histograms for each feature
data[numerical_features].hist(bins=10, figsize=(12, 6))
plt.show()

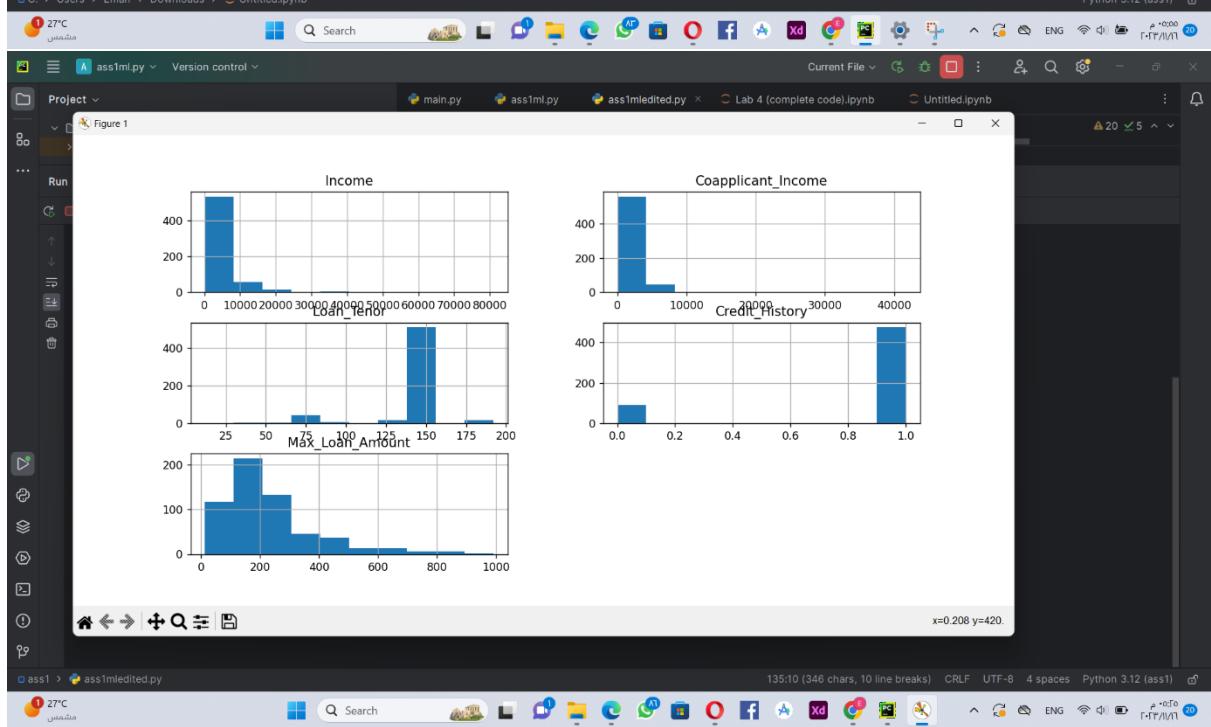
Range of Income: 80850.0
Standard deviation of Income: 6109.041673387174

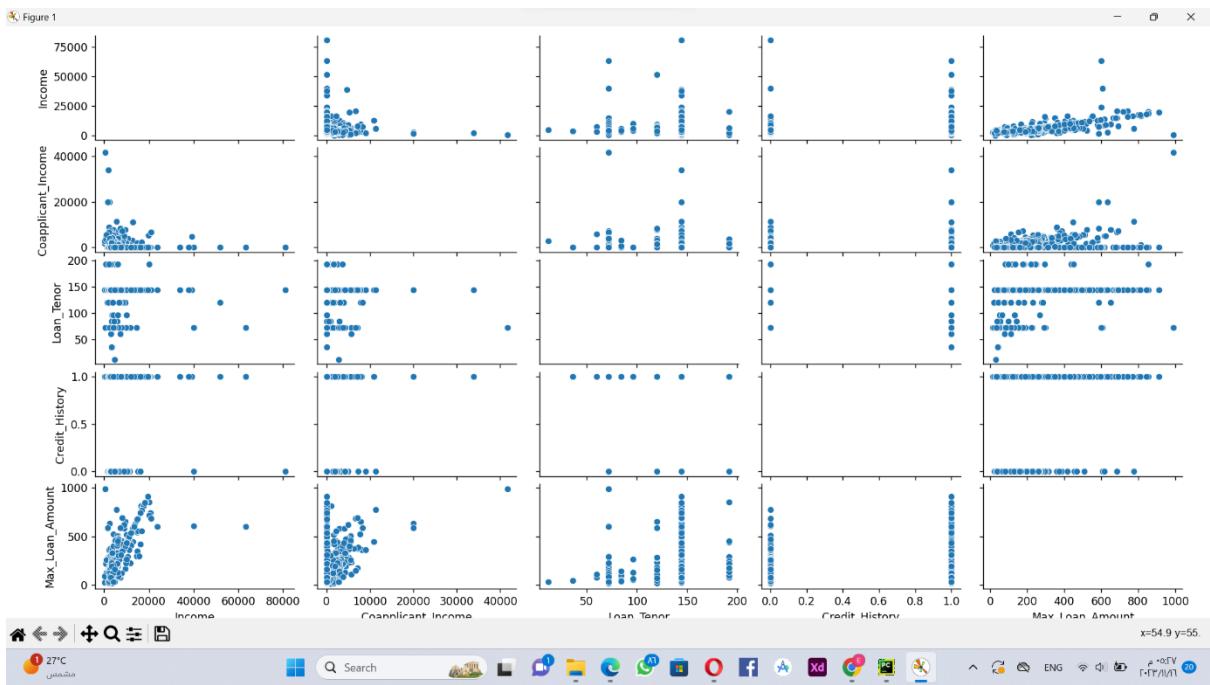
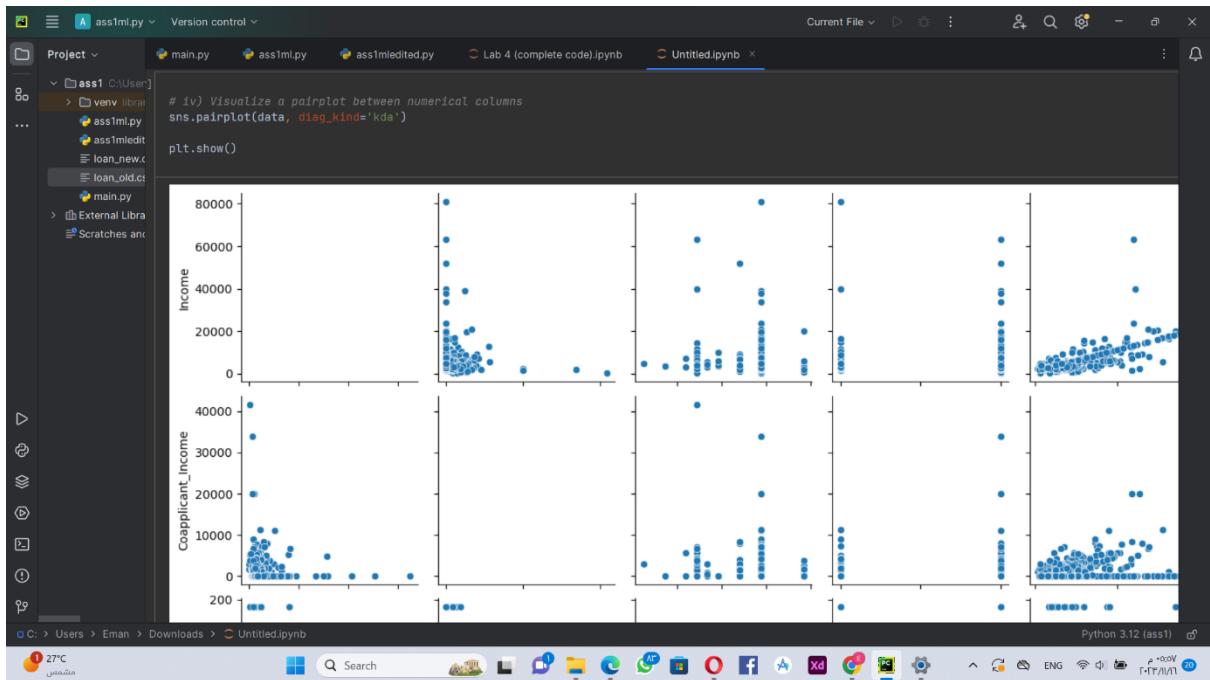
Range of Coapplicant_Income: 41667.0
Standard deviation of Coapplicant_Income: 2926.2483692241917

Range of Loan_Tenor: 180.0
Standard deviation of Loan_Tenor: 25.366293785656753

Range of Credit_History: 1.0
Standard deviation of Credit_History: 0.3648783192364049

Range of Max_Loan_Amount: 977.66
Standard deviation of Max_Loan_Amount: 161.97696676527076
```





c-data preprocessing

The screenshot shows a Jupyter Notebook interface with the following code in the cell:

```
# c) Data Preprocessing
# i) Remove records containing missing values
data = data.dropna()

# ii) Separate features and targets
features = data.drop(columns=['Max_Loan_Amount', 'Loan_Status'])
target = data[['Max_Loan_Amount', 'Loan_Status']]

print("\n----- features -----")
print(features)
print("\n----- target -----")
print(target)
```

Below the code, the output displays the 'features' DataFrame:

	Loan_ID	Gender	Married	Dependents	Education	Income
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
5	LP001011	Male	Yes	2	Graduate	5417
6	LP001013	Male	Yes	0	Not Graduate	2333
7	LP001014	Male	Yes	3+	Graduate	3036
8	LP001018	Male	Yes	2	Graduate	4006
9	LP001020	Male	Yes	1	Graduate	12841
10	LP001024	Male	Yes	2	Graduate	3200
11	LP001027	Male	Yes	2	Graduate	2500

The screenshot shows a Jupyter Notebook interface with the following code in the cell:

```
# c) Data Preprocessing
# i) Remove records containing missing values
data = data.dropna()

# ii) Separate features and targets
features = data.drop(columns=['Max_Loan_Amount', 'Loan_Status'])
target = data[['Max_Loan_Amount', 'Loan_Status']]

print("\n----- features -----")
print(features)
print("\n----- target -----")
print(target)
```

Below the code, the output displays the 'features' DataFrame:

	Loan_ID	Gender	Married	Dependents	Education	Income	Property_Area
1	LP001003	Male	Yes	...	144.0	1.0	Rural
2	LP001005	Male	Yes	...	144.0	1.0	Urban
3	LP001006	Male	Yes	...	144.0	1.0	Urban
4	LP001008	Male	No	...	144.0	1.0	Urban
5	LP001011	Male	Yes	...	144.0	1.0	Urban
...
609	LP002978	Female	No	...	144.0	1.0	Rural
610	LP002979	Male	Yes	...	72.0	1.0	Rural
611	LP002983	Male	Yes	...	144.0	1.0	Urban
612	LP002984	Male	Yes	...	144.0	1.0	Urban
613	LP002990	Female	No	...	144.0	0.0	Semiurban

[613 rows x 10 columns]

----- target -----

	Max_Loan_Amount	Loan_Status
1	236.99	N
2	81.20	Y
3	179.83	Y
4	232.40	Y
5	414.50	Y
...
609	76.16	Y
610	35.47	Y
611	348.92	Y

```
...  
..... target .....
```

	Max_Loan_Amount	Loan_Status
1	236.99	N
2	81.20	Y
3	179.03	Y
4	232.40	Y
5	414.50	Y
..
609	76.16	Y
610	33.47	Y
611	348.92	Y
612	312.18	Y
613	160.98	N

[513 rows x 2 columns]

```
Loan_ID      384  
Gender       384  
Married      384  
Dependents   384  
Education     384  
Income        384  
Coapplicant_Income  384  
Loan_Tenor    384  
Credit_History 384  
Property_Area 384  
dtype: int64
```

```
Max_Loan_Amount Loan_Status  
336           229.28      Y
```

ass1 > ass1medited.py 13510 (346 chars, 10 line breaks) CRLF UTF-8 4 spaces Python 3.12 (ass1)

```
# iii) Shuffle and split the data into training and testing sets  
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.25, random_state=42)  
  
#print(x_train.count())  
#print(y_train)  
#print(x_test.count())  
#print(y_test)  
  
# iv) Encode categorical features  
label_encoder = LabelEncoder()  
categorical_columns = features.select_dtypes(include='object').columns  
  
# Exclude 'Loan_ID' from encoding  
categorical_columns = categorical_columns.drop('Loan_ID', errors='ignore')  
  
for column in categorical_columns:  
    x_train[column] = label_encoder.fit_transform(x_train[column])  
    x_test[column] = label_encoder.transform(x_test[column])  
  
    print("\n----- Features (encoded) -----")  
    print("\nx_train :")  
    print(x_train.head())  
    print("\nx_test :")  
    print(x_test.head())  
  
----- Features (encoded) -----
```

C:\ > Users > Eman > Downloads > Untitled.ipynb Python 3.12 (ass1)

The screenshot shows a Jupyter Notebook interface with the file `Untitled.ipynb` open. The code cell displays the beginning of a data encoding process:

```
----- Features (encoded) -----  
x_train :  
    Loan_ID Gender Married Dependents Education Income \\\n336 LP002110 1 1 1 0 5250  
579 LP002888 1 0 0 0 3182  
521 LP002690 1 0 0 0 2500  
186 LP001641 1 1 1 0 2178  
31 LP001095 1 0 0 0 3167  
  
Coapplicant_Income Loan_Tenor Credit_History Property_Area  
336 688.0 144.0 1.0 0  
579 2917.0 144.0 1.0 2  
521 0.0 144.0 1.0 1  
186 0.0 128.0 0.0 0  
31 0.0 144.0 1.0 2  
  
x_test :  
    Loan_ID Gender Married Dependents Education Income \\\n366 LP002187 1 0 0 0 2500  
595 LP002940 1 0 0 1 3833  
527 LP002706 1 1 1 1 5285  
184 LP001639 0 1 0 0 3625  
598 LP002945 1 1 0 0 9963  
  
Coapplicant_Income Loan_Tenor Credit_History Property_Area  
366 0.0 192.0 1.0 1  
595 0.0 144.0 1.0 0  
527 1430.0 144.0 0.0 1  
184 0.0 144.0 1.0 1  
598 0.0 144.0 1.0 0
```

The screenshot shows the continuation of the Jupyter Notebook session. A new code cell has been added at the bottom:

```
# v) Encode categorical targets  
encoder = LabelEncoder()
```

Project ass1ml.py Version control

ass1 C:\User... venv libra... ass1mlited.py Lab 4 (complete code).ipynb Untitled.ipynb

... ass1ml.py ass1mlited loan_newx loan_old.cs main.py External Libra... Scratches and

v) Encode categorical targets
encoder = LabelEncoder()

y_train['Loan_Status'] = encoder.fit_transform(y_train['Loan_Status'])
y_test['Loan_Status'] = encoder.fit_transform(y_test['Loan_Status'])

print(y_train.head())
print(y_test.head())

	Max_Loan_Amount	Loan_Status
336	229.28	1
579	237.39	1
521	56.00	1
186	21.48	0
31	89.62	0

	Max_Loan_Amount	Loan_Status
366	98.00	0
595	123.18	1
527	268.44	1
184	112.70	1
598	432.14	1

[72] # vi) Scale numerical features to the range [0, 1]
scaler = MinMaxScaler()

C: > Users > Eman > Downloads > Untitled.ipynb Python 3.12 (ass1)

27°C John Doe

Search

File Explorer Task View Taskbar

Project ass1ml.py Version control

ass1 C:\User... venv libra... ass1mlited.py Lab 4 (complete code).ipynb Untitled.ipynb

... ass1ml.py ass1mlited loan_newx loan_old.cs main.py External Libra... Scratches and

vi) Scale numerical features to the range [0, 1]
scaler = MinMaxScaler()
numerical_features = features.select_dtypes(include=['float64', 'int64']).columns

for i in numerical_features:
 x_train[i] = scaler.fit_transform(x_train[[i]])
 x_test[i] = scaler.transform(x_test[[i]])

print("----- Numerical features (Scaled) -----")
print(x_train)
print(x_train.head())
print("-----")
print(x_test)
print(x_test.head())

----- Numerical features (Scaled) -----

x_train :
 Loan_ID Gender Married Dependents Education Income \\\n336 LP002110 1 1 1 0 0.127983
579 LP002888 1 0 0 0 0.076087
521 LP002690 1 0 0 0 0.058973
186 LP001641 1 1 1 0 0.050892
31 LP001095 1 0 0 0 0.075711

Coapplicant_Income Loan_Tenor Credit_History Property_Area
336 0.03440 0.692308 1.0 0
579 0.14585 0.692308 1.0 2
521 0.00000 0.692308 1.0 1
186 0.00000 0.538462 0.0 0
31 0.00000 0.692308 1.0 2

C: > Users > Eman > Downloads > Untitled.ipynb Python 3.12 (ass1)

27°C John Doe

Search

File Explorer Task View Taskbar

The screenshot shows a Jupyter Notebook interface with several files listed in the sidebar: ass1ml.py, ass1mledit.py, ass1mledit.py, ass1mlitedit.py, Lab 4 (complete code).ipynb, and Untitled.ipynb. The main area displays two tables of data:

x_train :

	Loan_ID	Gender	Married	Dependents	Education	Income
336	LP002110	1	1	1	0	0.127983
579	LP002888	1	0	0	0	0.076087
521	LP002690	1	0	0	0	0.058973
186	LP001641	1	1	1	0	0.050892
31	LP001095	1	0	0	0	0.075711

x_test :

	Loan_ID	Gender	Married	Dependents	Education	Income
366	LP002187	1	0	0	0	0.058973
595	LP002940	1	0	0	1	0.092424
527	LP002706	1	1	1	1	0.128861
184	LP001639	0	1	0	0	0.087264
598	LP002945	1	1	0	0	0.246255

D-Linear Regression

```

x_train = x_train.drop(columns=['Loan_ID'])
x_test = x_test.drop(columns=['Loan_ID'])

# d) Fit a linear regression model to the data to predict the loan amount.

linear_model = LinearRegression()
linear_model.fit(x_train, y_train['Max_Loan_Amount'])

#predict the loan amount.
y_pred = linear_model.predict(x_test)
print("Predicted Loan Amounts:")
print(y_pred)

# e) Evaluate the linear regression model using sklearn's R2 score.
r2 = r2_score(y_test['Max_Loan_Amount'], y_pred)
print("R2 Score: ")
print(r2)
print("\n*****\n")

```

```

Predicted Loan Amounts:
[ 208.40992332 155.74344976 246.28980727 146.48274023 386.90426757
 193.53528974 286.64360771 207.19572733 339.74198426 211.08607306
 187.31219152 142.4814755 322.36046009 118.78724003 164.28441308
 226.13203884 165.29392969 392.83406954 212.4512585 141.50790217
 133.71580913 189.1439686 157.06041787 179.86223883 236.17490298
 373.47897449 270.83858527 237.38800085 141.20710153 183.14627799
 167.78764178 211.29132409 189.70074641 153.11980317 182.93512589
 172.79445538 254.13217675 422.78139726 321.24584082 232.56499199
 188.29675835 108.45487638 211.11482271 35.73588276 318.54964984
 302.44504833 189.35336414 289.19297189 219.85309435 133.23879302
 257.59263374 230.82878935 160.84318343 250.20555065 -94.10185371
 60.38841956 188.16299331 193.72958591 260.15212389 204.30843648
 177.52863063 144.63017698 139.60598029 21.04993512 74.95529273
 270.9836853 116.80139155 304.143284 190.59106853 238.4129221
 384.46856552 771.47103725 198.725880514 356.99874055 158.31377601
 125.40501216 93.78502343 149.20314413 130.98342505 412.2215406
 222.04783765 175.22921606 444.33210485 369.48374636 135.9966917
 173.27170945 219.97349246 232.39621957 2144.88032264 479.40545634
 195.99297647 34.28496401 101.43943701 148.75129509 564.59259222
 232.49515183 116.07656351 190.55264262 264.63261197 157.80943158
 224.87169611 74.54422719 115.10551852 -10.98014487 109.07806205
 366.87527726 303.47945333 89.26765117 92.67079705 200.41779818
 238.38321471 187.37296915 212.5827069 109.37108201 177.85086141
 166.41830071 171.07067888 306.91669179 251.42557426 224.299675
 96.80782628 120.93403869 587.85013782 216.09185322 648.47870053
 168.38803426 219.78777408 88.33783633 405.31327357]
R2 Score:
0.04232570340284625

```

F-Logistic regression

```

##(f) Fit a logistic regression model to the data to predict the loan status.
5 usages
def sigmoid(z):
    return 1 / (1 + np.exp(-z))
1 usage
def compute_cost(X, y, theta):
    m = len(y)
    h = sigmoid(np.dot(X, theta))
    cost = (-1/m) * np.sum(y * np.log(h) + (1 - y) * np.log(1 - h))
    return cost
2 usages
def gradient_descent(X, y, theta, learning_rate, iterations):
    m = len(y)
    cost_history = np.zeros(iterations)
    for i in range(iterations):
        h = sigmoid(np.dot(X, theta))
        gradient = np.dot(X.T, (h - y)) / m
        theta = theta - learning_rate * gradient
        cost_history[i] = compute_cost(X, y, theta)
    return theta, cost_history
159
# Convert target variable to binary (0 or 1)
y_train_binary = y_train['Loan_Status']
160 x_train_logistic = np.c_[np.ones([len(x_train), 1]), x_train]
161 theta_initial = np.zeros(x_train_logistic.shape[1])
162 # Set hyperparameters
learning_rate = 0.01
iterations = 1000

```

```

def gradient_descent(X, y, theta, learning_rate, iterations):
    m = len(y)
    cost_history = np.zeros(iterations)

    for i in range(iterations):
        h = sigmoid(np.dot(X, theta))
        gradient = np.dot(X.T, (h - y)) / m
        theta = theta - learning_rate * gradient
        cost_history[i] = compute_cost(X, y, theta)

    return theta, cost_history

# Convert target variable to binary (0 or 1)
y_train_binary = y_train['Loan_Status']
x_train_logistic = np.c_[np.ones((len(x_train), 1)), x_train]
theta_initial = np.zeros(x_train_logistic.shape[1])
# Set hyperparameters
learning_rate = 0.01
iterations = 1000
theta_final, cost_history = gradient_descent(x_train_logistic, y_train_binary, theta_initial, learning_rate, iterations)
x_test_logistic = np.c_[np.ones((len(x_test), 1)), x_test]
probabilities = sigmoid(np.dot(x_test_logistic, theta_final))

```

Accuracy of prediction

Accuracy of Logistic Regression (from scratch): 71.31782945736434%

Data analysis of new data

```

print(missing_values[missing_values > 0])

Gender          11
Dependents      10
Loan_Tenor       7
Credit_History   29
dtype: int64

```

Data Types of new data:

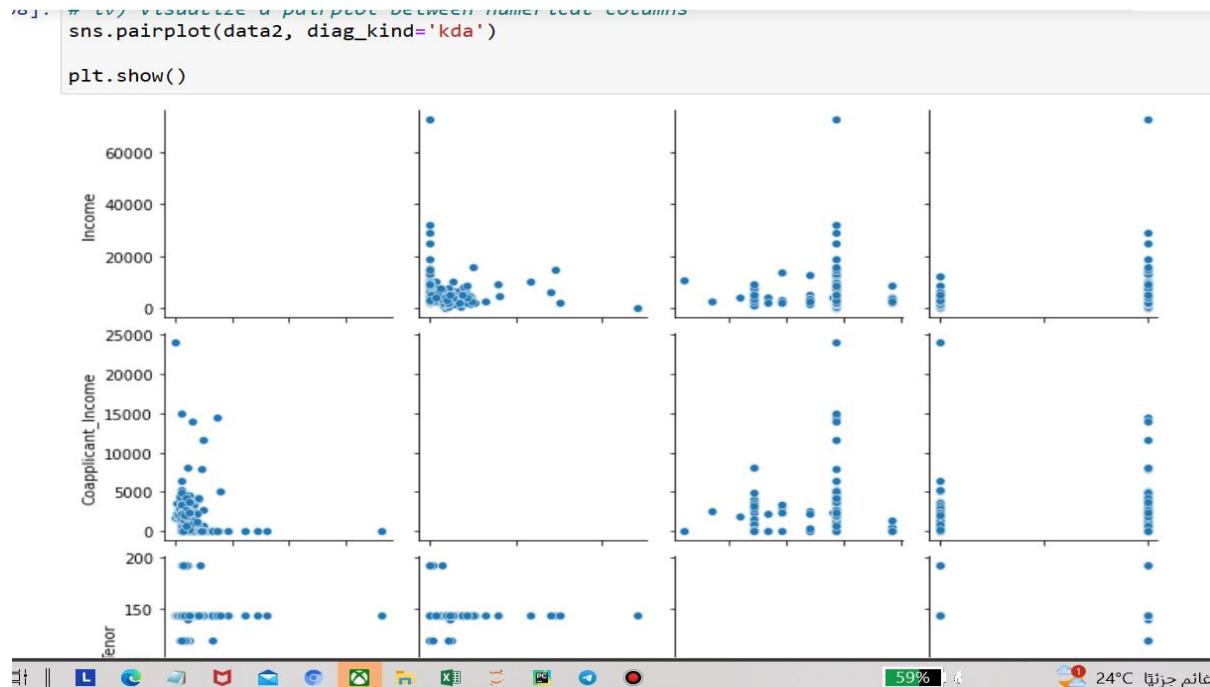
Gender	: categorical
Married	: categorical
Dependents	: categorical
Education	: categorical
Income	: numerical
Coapplicant_Income	: numerical
Loan_Tenor	: numerical
Credit_History	: numerical
Property_Area	: categorical

```
New data Ranges & standard deviations
Range of Income: 72529.0
Standard deviation of Income: 0.06541198928270776

Range of Coapplicant_Income: 24000.0
Standard deviation of Coapplicant_Income: 0.0973469898089432

Range of Loan_Tenor: 180.0
Standard deviation of Loan_Tenor: 0.12790746528146

Range of Credit_History: 1.0
Standard deviation of Credit_History: 0.3807099557532439
```





Data processing of new data

```

----- data2 (encoded) -----
   Loan_ID  Gender  Married  Dependents  Education  Income \
0  LP001015      1        1          0          0     5720
1  LP001022      1        1          1          0     3076
2  LP001031      1        1          2          0      5000
4  LP001051      1        0          0          1     3276
5  LP001054      1        1          0          1     2165

   Coapplicant_Income  Loan_Tenor  Credit_History  Property_Area
0                  0       144.0           1.0             2
1                 1500       144.0           1.0             2
2                 1800       144.0           1.0             2
4                  0       144.0           1.0             2
5                3422       144.0           1.0             2

-----
----- Numerical features (Scaled) -----
data2 :
   Loan_ID  Gender  Married  Dependents  Education  Income \
0  LP001015      1        1          0          0  0.078865
1  LP001022      1        1          1          0  0.042411
2  LP001031      1        1          2          0  0.068938
4  LP001051      1        0          0          1  0.045168
5  LP001054      1        1          0          1  0.029850

   Coapplicant_Income  Loan_Tenor  Credit_History  Property_Area
0      0.000000    0.733333           1.0             2
1      0.062500    0.733333           1.0             2
2      0.075000    0.733333           1.0             2
4      0.000000    0.733333           1.0             2
5      0.142583    0.733333           1.0             2

*****

```

```

----- data2 (encoded) -----
      Loan_ID  Gender  Married  Dependents  Education  Income \
0  LP001015      1       1        0          0     5720
1  LP001022      1       1        1          0     3076
2  LP001031      1       1        2          0      5000
4  LP001051      1       0        0          1     3276
5  LP001054      1       1        0          1     2165

   Coapplicant_Income  Loan_Tenor  Credit_History  Property_Area
0                  0      144.0           1.0            2
1                 1500      144.0           1.0            2
2                 1800      144.0           1.0            2
4                  0      144.0           1.0            2
5                3422      144.0           1.0            2

-----
----- Numerical features (Scaled) -----
data2 :
      Loan_ID  Gender  Married  Dependents  Education  Income \
0  LP001015      1       1        0          0  0.078865
1  LP001022      1       1        1          0  0.042411
2  LP001031      1       1        2          0  0.068938
4  LP001051      1       0        0          1  0.045168
5  LP001054      1       1        0          1  0.029850

   Coapplicant_Income  Loan_Tenor  Credit_History  Property_Area
0      0.000000    0.733333           1.0            2
1      0.062500    0.733333           1.0            2
2      0.075000    0.733333           1.0            2
4      0.000000    0.733333           1.0            2
5      0.142583    0.733333           1.0            2

*****

```

Predictions of max loan amount in new data

```
In [118]: print(y_pred2)

[ 1. 5.42322016e+02  1. 38618269e+02  1. 81319022e+02  9. 50603875e+01
 1. 72810751e+02  0. 52831244e+01  1. 06530491e+02  2. 15559154e+02
 1. 44796498e+02  9. 14603639e+01  1. 24618022e+02  2. 68089181e+02
 1. 30333737e+02  1. 18101111e+02  1. 25762760e+02  1. 94424563e+02
 1. 33568333e+02  1. 09956576e+01  1. 16267836e+02  1. 99955508e+01
 9. 55093008e+01  2. 64182870e+02  6. 66728553e+02  2. 88991175e+02
 1. 28289565e+02  1. 34182349e+01  1. 79978833e+02  1. 449728691e+02
 1. 68142882e+02  1. 36315204e+02  8. 85119974e+01  1. 54294403e+02
 1. 60186450e+02  1. 47619168e+02  1. 56498327e+02  1. 74149615e+02
 9. 52865543e+02  1. 30822363e+02  3. 33115771e+02  9. 56449316e+01
 1. 44796498e+02  1. 18101111e+02  1. 25762760e+02  1. 94424563e+02
 1. 00528841e+01  1. 43717699e+01  7. 73020499e+01  1. 14709909e+02
 7. 08335753e+01  1. 71607811e+02  4. 49723833e+02  1. 35957673e+02
 1. 87565154e+02  1. 18172702e+02  1. 01733025e+02  1. 18144274e+02
 1. 97327440e+02  1. 25865409e+02  1. 03178431e+02  1. 95957838e+02
 2. 22521898e+02  1. 23898965e+02  1. 11186047e+01  1. 89518826e+02
 2. 31023343e+02  1. 94384811e+02  1. 67131433e+02  1. 46036651e+02
 1. 66383984e+02  1. 18101111e+02  1. 25762760e+02  1. 94424563e+02
 1. 99951082e+02  1. 47262692e+02  4. 49459638e+00  1. 12707376e+02
 1. 62699494e+02  9. 63142290e+01  1. 44570862e+02  1. 27494450e+02
 1. 05062204e+02  1. 93908777e+02  1. 73838794e+02  1. 84845275e+02
 1. 49278148e+02  1. 25362919e+02  1. 99995876e+02  2. 23446416e+02
 1. 26986189e+02  1. 59810533e+02  1. 15085459e+02  4. 01584168e+01
 1. 37846612e+02  1. 27318720e+02  1. 33463596e+02  1. 64353462e+02
 1. 10193135e+02  1. 92975762e+01  1. 70709496e+02  2. 56082101e+02
 1. 74107747e+02  1. 18101111e+02  1. 25762760e+02  1. 94424563e+02
 1. 14953048e+02  1. 39137656e+02  4. 2954254963e+02  2. 6865564268e+02
 1. 27915658e+02  1. 51324879e+02  1. 82885855e+02  3. 94416260e+01
 1. 40882682e+02  1. 06842838e+02  1. 509666447e+02  9. 14918623e+01
 1. 53375163e+02  1. 25373333e+02  1. 3829582e+02  1. 62188729e+02
 1. 22954613e+02  9. 04236815e+01  1. 68178828e+02  1. 33639586e+01
 1. 27417777e+02  1. 18101111e+02  1. 25762760e+02  1. 94424563e+02
 1. 48041693e+02  1. 07482242e+02  1. 34907658e+02  1. 37808180e+02
 1. 37880882e+02  1. 37302471e+02  8. 675018113e+01  2. 31364913e+02
 1. 86929678e+02  2. 14626674e+02  2. 017057791e+02  2. 34006686e+02
 8. 48405097e+02  1. 42684373e+02  8. 33574380e+01  6. 98163452e+01
 0. 95164091e+01  1. 85438715e+02  1. 287038389e+02  1. 05856594e+02
 1. 08031467e+02  1. 27813801e+02  1. 51687870e+02  2. 15131376e+02
 1. 17001111e+02  1. 25373333e+02  1. 3829582e+02  1. 62188729e+02
 1. 66149966e+02  1. 08312198e+02  1. 20865348e+02  1. 18430993e+02
 1. 56739874e+02  2. 19104445e+02  3. 04112663e+02  3. 82199339e+02
 1. 05521516e+01  1. 13075601e+01  1. 81934764e+02  1. 32106608e+02
 3. 10194364e+01  1. 47619168e+02  2. 266051113e+02  1. 27815425e+02
 9. 41087485e+01  1. 23733081e+02  3. 08187908e+02  1. 00285139e+02
 1. 66887909e+01  9. 54501111e+01  1. 38897909e+02  2. 08810654e+02
 1. 08224002e+02  1. 62381801e+02  6. 90009042e+01  1. 36242165e+02
 4. 41109171e+02  1. 51778521e+02  1. 09185262e+02  6. 94880879e+01
 2. 60456319e+02  1. 97498877e+02  1. 68825558e+02  1. 72501794e+02
 1. 78308070e+02  1. 23889622e+01  1. 12976042e+02  7. 35131943e+01
 1. 38649797e+02  1. 23164368e+02  1. 59849838e+02  1. 22303986e+02
 1. 15214113e+02  -1. 91613182e+01  1. 73367178e+02  1. 22166347e+02
 4. 08601333e+02  2. 00312171e+02  3. 12274511e+02  1. 15139263e+02
 2. 08070000e+02  1. 25373333e+02  1. 38285855e+02  1. 22166347e+02
 1. 33849092e+02  1. 609051136e+02  1. 91034517e+02  1. 26382173e+02
 3. 57326292e+01  1. 86122441e+02  1. 025858823e+02  1. 41204846e+02
 1. 44363347e+02  1. 606151317e+02  1. 144073785e+02  2. 20939837e+02
 2. 06186741e+02  1. 57462849e+02  1. 44644367e+02  4. 16595871e+02
 1. 30818609e+02  1. 28583337e+02  1. 95550650e+02  1. 67085586e+02
 1. 05521516e+02  1. 17461506e+02  1. 95550650e+02  5. 22804243e+02
 1. 22468522e+02  1. 18101111e+02  1. 25762760e+02  1. 94424563e+02
 2. 88504699e+01  1. 19256598e+02  1. 33121516e+02  6. 64645566e+02
 1. 24674211e+02  3. 27085610e+02  2. 10806783e+02  1. 93424345e+02
 1. 30848864e+02  3. 087498118e+02  1. 78119286e+02  1. 72538418e+02
 1. 01558442e+02  1. 19958669e+02  1. 39457962e+02  1. 76445485e+02
 1. 2118382e+02  1. 00473537e+02  1. 22794173e+02  1. 86126807e+02
 1. 25979467e+02  1. 14246093e+02  1. 89513085e+02  1. 15556562e+02
 1. 77401111e+02  1. 25373333e+02  1. 38285855e+02  1. 22166347e+02
 4. 01646562e+02  3. 54499241e+01  1. 31616418e+01  1. 23290917e+02
 1. 40315184e+02  1. 66018613e+02  1. 38286587e+02  1. 09317498e+02
 5. 45287111e+01  4. 81105180e+02  1. 53219703e+02  1. 07037351e+02
 1. 52928755e+02  2. 13028107e+02  1. 56693799e+02  2. 23051363e+02
 1. 70392229e+02  1. 28385533e+02  1. 26351299e+02  7. 33726558e+01
 1. 27933317e+02  9. 19657656e+01  1. 67919291e+02  9. 26179387e+01
 1. 05521516e+02  -2. 10806783e+02  1. 025858823e+02  1. 41204846e+02
 9. 96167381e+01  1. 57611492e+02  8. 70068015e+01  1. 68870515e+02
 6. 096223932e+01  1. 25248230e+02  1. 88302727e+02  2. 49084375e+02
 6. 85166367e-01  1. 32877295e+02  1. 64305433e+02  8. 18284774e+01
 1. 85546317e+02  1. 43707685e+02  1. 55820311e+02  1. 42219867e+02
 2. 10168981e+02  8. 8120926e+01]
```

Predictions of loan status in new data