

Assignment-1

Abdelrahman Sayed_20201114

Seif El-Din Mohamed_20200239

Shahd Fekry ali_20201101

Mariam Alaa Eldeen_20200525

Wessam Fawzy_20201215

Import Libraries

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer # Transform Column
from sklearn.preprocessing import OneHotEncoder # Kind of Encoding (Transformation)
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
```

Import Data From Dataset

```
In [ ]: dataset = pd.read_csv('loan_old.csv')
```

Analyze some data

Checking Missing Values

```
In [ ]: missing_values = dataset.isnull().sum()
print("Missing Values:\n", missing_values)
```

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
```

Type of Features

```
In [ ]: feature_types = dataset.dtypes
print("\nFeature Types:\n", feature_types)
```

```
Feature 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
```

Check The Scale

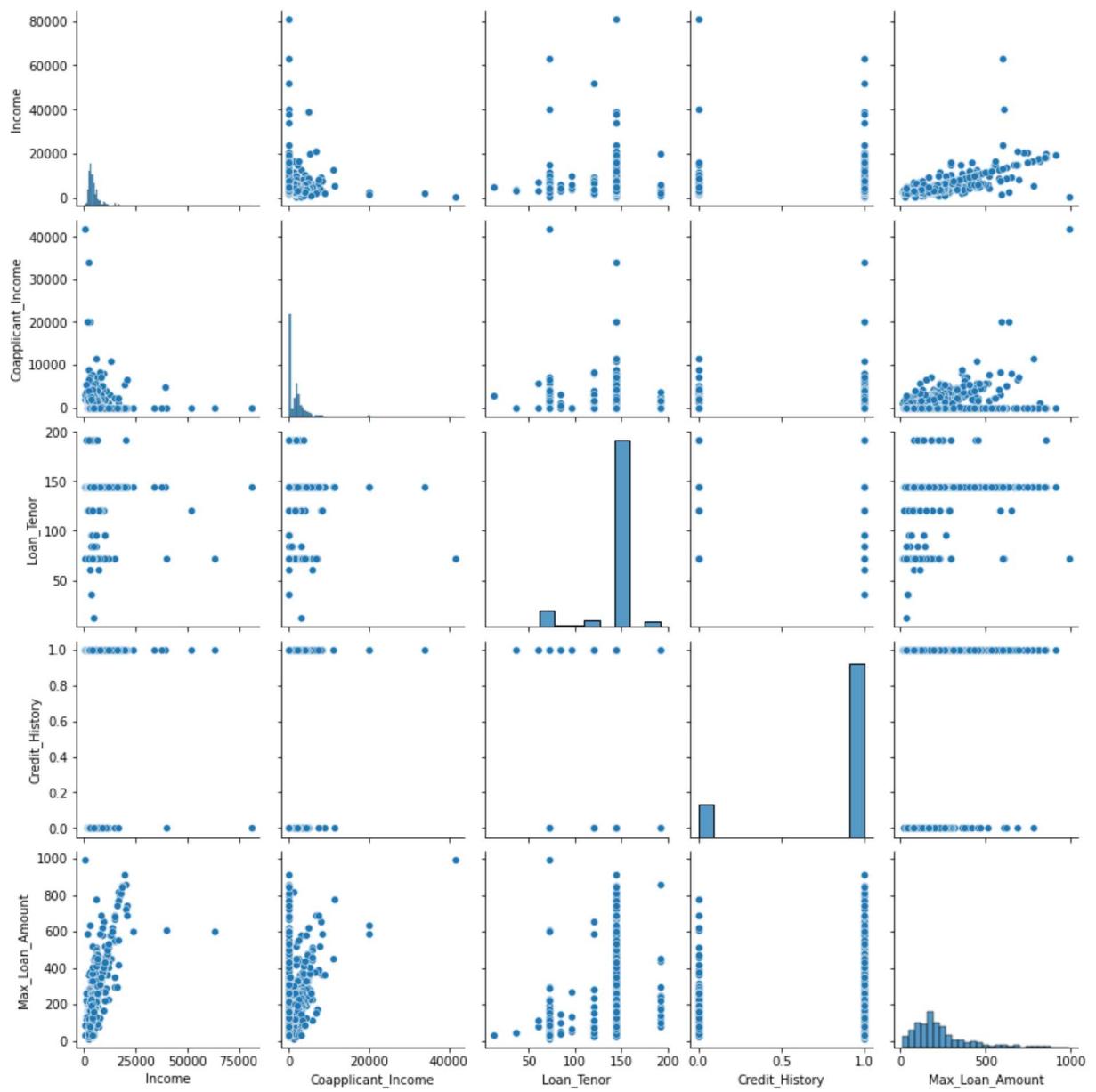
```
In [ ]: numerical_features = dataset.select_dtypes(include=['int64', 'float64'])
feature_scales_std = numerical_features.std()
print("\nFeature Scales (Standard Deviation):\n", feature_scales_std)
```

```
Feature Scales (Standard Deviation):
Income           6109.041673
Coapplicant_Income 2926.248369
Loan_Tenor        23.366294
Credit_History     0.364878
Max_Loan_Amount    161.976967
dtype: float64
```

Visualize pairplot between numerical values

```
In [ ]: sns.pairplot(numerical_features)
plt.show()
```

Linear Regression



Dataset Size before removing records

```
In [ ]: dataset.shape
```

```
Out[ ]: (614, 12)
```

Drop Records with empty values

```
In [ ]: dataset.dropna(inplace=True)
dataset
```

	Loan_ID	Gender	Married	Dependents	Education	Income	Coapplicant_Income	Loan_Tenor
1	LP001003	Male	Yes	1	Graduate	4583	1508.0	144.0
2	LP001005	Male	Yes	0	Graduate	3000	0.0	144.0
3	LP001006	Male	Yes	0	Not Graduate	2583	2358.0	144.0
4	LP001008	Male	No	0	Graduate	6000	0.0	144.0
5	LP001011	Male	Yes	2	Graduate	5417	4196.0	144.0

	Loan_ID	Gender	Married	Dependents	Education	Income	Coapplicant_Income	Loan_Tenor
...
609	LP002978	Female	No	0	Graduate	2900	0.0	144.0
610	LP002979	Male	Yes	3+	Graduate	4106	0.0	72.0
611	LP002983	Male	Yes	1	Graduate	8072	240.0	144.0
612	LP002984	Male	Yes	2	Graduate	7583	0.0	144.0
613	LP002990	Female	No	0	Graduate	4583	0.0	144.0

513 rows × 12 columns

Check Missing Values after Deleting empty value records

```
In [ ]: missing_values = dataset.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
Loan_ID          0
Gender           0
Married          0
Dependents       0
Education         0
Income            0
Coapplicant_Income 0
Loan_Tenor        0
Credit_History    0
Property_Area     0
Max_Loan_Amount   0
Loan_Status        0
dtype: int64
```

Dataset Size after removing records

```
In [ ]: dataset.shape
```

```
Out[ ]: (513, 12)
```

Separate Dataset

```
In [ ]: X = dataset.iloc[:, 1:-2].values #dataset.iloc[:, [0]].values pandas series
Y1 = dataset.iloc[:, -2].values
Y2 = dataset.iloc[:, -1].values
```

Train & Test Split

```
In [ ]: X_train, X_test, Y1_train, Y1_test, Y2_train, Y2_test = train_test_split(X, Y1, Y2,
```

Encoding Features

```
In [ ]:
le1 = LabelEncoder()
le2 = LabelEncoder()
le3 = LabelEncoder()
le4 = LabelEncoder()

X_train[:,0] = np.array(le1.fit_transform(X_train[:,0])) # 1/0 -- Male/Female
X_test[:,0] = np.array(le1.transform(X_test[:,0])) # 1/0 -- Male/Female
X_train[:,1] = np.array(le2.fit_transform(X_train[:,1])) # 1/0 Yes/No
X_test[:,1] = np.array(le2.transform(X_test[:,1])) # 1/0 Yes/No
X_train[:,2] = np.array(le3.fit_transform(X_train[:,2]))
X_test[:,2] = np.array(le3.transform(X_test[:,2]))
X_train[:,3] = np.array(le4.fit_transform(X_train[:,3])) # 1/0 Not Graduate/Graduate
X_test[:,3] = np.array(le4.transform(X_test[:,3])) # 1/0 Not Graduate/Graduate
# X_train[:,8] = le.fit_transform(X_train[:,8]) # 1/0 Not Graduate/Graduate

# X_test[:,8] = le.fit_transform(X_test[:,8]) # 1/0 Not Graduate/Graduate

# kind of transformation, Encoder algo, idx of column, remainder of columns passthrough
ct = ColumnTransformer( transformers = [('encoder', OneHotEncoder(), [8])], remainder='passthrough')
X_train = np.array(ct.fit_transform(X_train)) # fit_transform doesn't return Numpy array
X_test = np.array(ct.transform(X_test)) # fit_transform doesn't return Numpy array
```

Encoding Targets

```
In [ ]:
le5 = LabelEncoder()
Y2_train = np.array(le5.fit_transform(Y2_train)) # 1/0 Y/N
Y2_test = np.array(le5.transform(Y2_test)) # 1/0 Y/N
```

numerical features are standardized

```
In [ ]:
nf1 = X_train[:, 7:10]
nf2 = X_test[:, 7:10]

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit and transform the numerical features
nf_standardized1 = np.array(scaler.fit_transform(nf1))
nf_standardized2 = np.array(scaler.transform(nf2))

# Replace the original numerical features with the standardized ones in X
X_train[:, 7:10] = np.array(nf_standardized1)
X_test[:, 7:10] = np.array(nf_standardized2)
X_train
```

```
Out[ ]: array([[0.0, 1.0, 0.0, ..., -0.7085273663857746, 0.26497531728793927,
   1.0],
   [0.0, 1.0, 0.0, ..., -0.7085273663857746, 0.26497531728793927,
   1.0],
   [0.0, 1.0, 0.0, ..., 0.6858482876904199, 0.26497531728793927, 1.0],
   ...,
   [0.0, 1.0, 0.0, ..., -0.7085273663857746, 0.26497531728793927,
   1.0],
   [1.0, 0.0, 0.0, ..., -0.7085273663857746, 0.26497531728793927,
   1.0],
```

```
[0.0, 0.0, 1.0, ..., 0.020384879869032937, 0.26497531728793927,
1.0]], dtype=object)
```

Linear Regression

Train & Fit The Model

```
In [ ]: linear_model = LinearRegression()
linear_model.fit(X_train, Y1_train)

Out[ ]: LinearRegression()
```

Predict New Values

```
In [ ]: Y_predict = linear_model.predict(X_test)
print(Y_predict)

[318.23891098 250.91209729 199.49452399 168.20908719 121.63492447
187.92768686 223.98992686 195.94348614 105.28783772 169.33625118
247.94236999 189.6287291 125.68894144 194.07277289 300.38835984
196.44663472 63.01566148 46.40615272 279.59679854 240.7187483
93.21733945 230.42905317 246.75705866 224.47001968 185.09927516
274.93670558 15.10760739 189.01083886 255.27774307 163.13669682
341.92211078 81.93411975 131.33027718 91.01216678 156.18061778
84.42772628 255.91974953 135.59792066 213.31826757 110.83754159
44.2652818 513.6445081 235.11470282 198.67870064 340.79410586
221.10950568 189.63990832 168.36935341 208.07287281 521.08609863
199.34114997 748.67544074 217.63133647 230.75501991 164.94382457
196.86277744 388.00028742 389.38401018 211.31054841 141.50856395
195.43525337 168.1897227 192.06074912 205.02029368 210.69511328
16.26184451 184.06796461 297.55911372 202.22067235 237.00082482
139.33205728 138.72294039 277.23371448 209.93575456 162.1880175
175.75583641 270.86692624 288.16212446 150.65921383 273.9581264
229.43726167 246.85525063 454.03348179 151.67481348 234.71803595
198.14111145 210.88640142 579.53207833 231.96664186 35.04441162
227.89106382 26.85448874 208.14378617 621.9149476 55.06880689
379.23598538 340.16540938 220.19444511 143.92909433 402.00560777
236.30117173 176.35352409 258.15719526]
```

R-Squared Error Evaluation

```
In [ ]: r2 = r2_score(Y1_test, Y_predict)
print(f'R-squared score: {r2}')
```

R-squared score: 0.8774662963560383

Logistic Regression From Scratch

```
In [ ]: def sigmoid(x):
    return 1/(1+np.exp(-x))

class LogisticRegression():

    def __init__(self, lr=0.001, n_iters=1000):
        self.lr = lr
        self.n_iters = n_iters
        self.weights = None
```

```

        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        for _ in range(self.n_iters):
            linear_pred = np.array(np.dot(X, self.weights), dtype=np.float32) + self.bias
            predictions = sigmoid(linear_pred)

            dw = (1/n_samples) * np.array(np.dot(X.T, (predictions - y)), dtype=np.float32)
            db = (1/n_samples) * np.sum(predictions-y)

            self.weights = self.weights - self.lr*dw
            self.bias = self.bias - self.lr*db

    def predict(self, X):
        linear_pred = np.array(np.dot(X, self.weights), dtype=np.float32) + self.bias
        y_pred = sigmoid(linear_pred)
        class_pred = [0 if y<=0.5 else 1 for y in y_pred]
        return class_pred

```

Train Logistic Regression Model

```

In [ ]:
clf = LogisticRegression(lr=0.07)
clf.fit(X_train,Y2_train)
y_pred = clf.predict(X_test)

def accuracy(y_pred, y_test):
    return np.sum(y_pred==y_test)/len(y_test)

acc = accuracy(y_pred, Y2_test)
print(acc)

```

0.8640776699029126

Load Newloan Data

```

In [ ]:
new_dataset = pd.read_csv('loan_new.csv')
new_dataset.dropna(inplace=True)

```

Preprocessing New Data

```

In [ ]:
new_X = new_dataset.iloc[:, 1: ].values #dataset.iloc[:, [0]].values pandas seri

```

Encoding

```

In [ ]:
new_X[:,0] = np.array(le1.transform(new_X[:,0])) # 1/0 -- Male/Female
new_X[:,1] = np.array(le2.transform(new_X[:,1])) # 1/0 Yes/No
new_X[:,2] = np.array(le3.transform(new_X[:,2]))
new_X[:,3] = np.array(le4.transform(new_X[:,3])) # 1/0 Not Graduate/Graduate

new_X = np.array(ct.transform(new_X)) # fit_transform doesn't return Numpy array

```

Standardization

```
In [ ]: new_nf = new_X[:, 7:10]

# Fit and transform the numerical features
new_nf_standardized = np.array(scaler.transform(new_nf))

# Replace the original numerical features with the standardized ones in X
new_X[:, 7:10] = np.array(new_nf_standardized)
```

Predict New Values

```
In [ ]: new_Y2_pred = clf.predict(new_X)  
new_Y1_pred = linear_model.predict(new_X)
```

```
In [ ]: print(new_Y2_pred)
```

```
In [ ]: print(new_Y1_pred)
```

[2.08219470e+02	1.91029745e+02	2.57080058e+02	1.26175970e+02
2.07204758e+02	1.03126606e+02	1.71752917e+02	3.23596190e+02	
1.84069793e+02	1.21290997e+02	1.74267498e+02	3.91692730e+02	
1.75390648e+02	2.08949745e+02	2.80831712e+02	1.91496259e+02	
5.47374368e+02	4.40332466e+01	1.49661213e+02	-5.31697464e+01	
1.31596055e+02	3.36203830e+02	8.15019272e+02	3.72030802e+02	
4.72337678e+01	1.03878936e+02	2.59560295e+02	1.98392495e+02	
2.15019593e+02	1.77532905e+02	1.37862437e+02	2.30506724e+02	
2.04011023e+02	2.07431223e+02	2.10749828e+02	2.31296739e+02	
1.42138022e+02	1.58966269e+02	3.12298226e+02	1.42718937e+02	
1.67395598e+02	2.86445628e+02	1.80129400e+02	2.57614511e+02	
5.07608082e+01	1.83207422e+02	1.21033882e+02	1.67432262e+02	
1.22833549e+02	2.50280757e+02	4.77285058e+01	1.67087522e+02	
2.51536290e+02	1.95168627e+02	1.62144825e+02	1.79709384e+02	
2.52080008e+02	1.68825629e+02	1.49424030e+02	2.65870556e+02	
2.95128948e+02	1.77522467e+02	4.17854428e+01	2.44404639e+02	
2.86743847e+02	2.55017319e+02	2.15118209e+02	2.40257230e+02	
2.85718995e+02	2.59611113e+02	2.05360291e+02	1.99003387e+03	
2.83857952e+02	2.99295481e+02	1.80622076e+01	3.01526469e+02	
2.11187822e+02	1.40634319e+02	2.05929285e+02	1.98491115e+02	
4.50849130e+02	2.62455495e+02	2.38633679e+02	2.58732923e+02	
2.38958060e+02	2.81919075e+02	2.57101549e+02	3.31519491e+02	
1.89981615e+02	2.16066214e+02	1.66090097e+02	-1.28425800e+01	
1.86271701e+02	2.13069763e+02	1.82271503e+02	2.04926960e+02	
1.77397447e+02	1.36144028e+02	2.50698614e+02	3.34312048e+02	

9.32695136e+01	1.70497499e+02	2.11462644e+02	1.35925779e+02
2.56261793e+02	2.11937337e+02	3.33240394e+02	3.70993081e+02
1.82909140e+02	2.20053608e+02	2.67024424e+02	-2.35048352e+01
1.84749757e+02	1.61965634e+02	2.17625544e+02	1.47406091e+02
2.73157643e+01	1.69614980e+02	2.06824699e+02	2.26391203e+02
1.92253137e+02	1.63513805e+02	2.52004023e+02	8.57821535e+01
3.34941877e+02	1.32442421e+02	2.83503449e+02	2.16897437e+02
2.09200375e+02	2.68722336e+02	1.76020639e+02	2.04451697e+02
1.82904814e+02	2.16198567e+02	1.12538383e+02	3.13361741e+02
2.53193559e+02	2.98112059e+02	2.70929964e+02	3.14578753e+02
1.32951058e+02	2.00829908e+02	1.36438754e+02	1.00835995e+02
1.39442596e+02	2.36156096e+02	2.06303682e+02	1.68715494e+02
1.69445329e+02	1.92638807e+02	2.14164922e+02	5.86463760e+01
1.78353471e+02	3.61426652e+02	2.25081821e+02	2.23407203e+02
2.61748466e+02	2.69622992e+02	1.84059097e+02	3.02240093e+02
2.17773402e+02	3.17308442e+02	4.07920453e+02	4.19485419e+02
5.73065215e+01	1.61222018e+02	2.59924846e+02	2.07491810e+02
4.17193913e+02	2.07431223e+02	1.90999332e+02	1.65109876e+02
1.52868938e+02	1.71284759e+02	4.11453326e+02	1.53419718e+02
2.13318460e+02	1.39214715e+02	2.07732696e+02	2.81307963e+02
1.84254117e+02	1.57162615e+02	1.08286478e+02	2.83013105e+02
2.23319139e+02	2.23146621e+02	1.34051523e+02	-3.82751901e+01
3.50949601e+02	2.72760556e+02	2.28652094e+02	2.22410583e+02
2.37089297e+02	1.51812685e+02	1.95882239e+02	1.13118641e+02
1.79666018e+02	2.91475486e+02	2.21931274e+02	1.92939089e+02
9.05287881e+02	-1.23470774e+00	2.58765476e+02	1.62340435e+02
1.93737780e+02	2.69558474e+02	6.72639871e+02	1.62446590e+02
2.73195026e+02	1.41174158e+02	2.08348681e+02	1.81921937e+02
1.91328920e+02	9.40825846e+01	2.68480711e+02	1.53826771e+02
-8.39675781e+00	2.74486647e+02	1.64445887e+02	2.06798124e+02
2.03163933e+02	1.42901706e+02	1.69916713e+02	2.78758420e+02
2.71629291e+02	2.22719784e+02	1.90469772e+02	5.79574333e+02
2.01972027e+02	2.87109085e+02	2.70222811e+02	2.60394981e+02
1.65055049e+02	1.72959229e+02	2.74922798e+02	7.20348820e+02
2.33605811e+02	1.39612191e+02	2.01005544e+02	2.28815573e+02
4.64581089e+01	1.86728893e+02	1.80943746e+02	2.08223739e+02
2.96335735e+02	6.88933512e+02	2.94129203e+02	2.59902685e+02
1.84986752e+02	4.07715510e+02	2.22574571e+02	2.40830064e+02
1.45060000e+02	1.66172210e+02	1.76562420e+02	2.31723863e+02
1.59239625e+02	1.52487549e+02	1.96790701e+02	2.61122628e+02
2.24446393e+02	1.53906037e+02	2.86040479e+02	1.76161925e+02
2.62315134e+02	3.07760365e+02	1.93726893e+02	3.04342858e+02
2.20027329e+02	9.27000217e+01	1.32797401e+02	1.81799080e+02
1.73676441e+02	2.20103610e+02	2.00554322e+02	1.34326867e+02
1.11978806e+02	6.43808372e+02	2.17016690e+02	1.43516585e+02
2.23898417e+02	3.21582321e+02	2.21664016e+02	3.30991386e+02
2.22100251e+02	1.80298115e+02	1.64250540e+02	1.28585556e+02
1.66232887e+02	1.43549939e+02	2.47028753e+02	1.25133679e+02
3.10470454e+02	1.51076074e+01	1.87187900e+02	2.53512670e+02
2.94948206e+02	2.21987325e+02	1.15402624e+02	1.88487218e+02
9.87449421e+01	3.16855418e+02	2.39126666e+02	3.28767771e+02
5.05640383e+01	3.13430092e+02	2.23490612e+02	1.19422491e+02
2.48372081e+02	1.96763751e+02	2.22360804e+02	1.89059325e+02
2.82764073e+02	1.57296249e+02		

Write Predicted Values on new CSV File

In []:

```
df = pd.DataFrame(new_dataset)
df['Max Loan Amount'] = new_Y1_pred
df['Loan Status'] = new_Y2_pred
df.to_csv('D:\\4th_1st_Sem\\Machine Learning\\Assignment 1\\PREDICTED_LOAN.csv', index=False)
print(df)
```

	Loan_ID	Gender	Married	Dependents	Education	Income	\
0	LP001015	Male	Yes	0	Graduate	5720	
1	LP001022	Male	Yes	1	Graduate	3076	
2	LP001031	Male	Yes	2	Graduate	5000	
4	LP001051	Male	No	0	Not Graduate	3276	
5	LP001054	Male	Yes	0	Not Graduate	2165	
..
361	LP002969	Male	Yes	1	Graduate	2269	
362	LP002971	Male	Yes	3+	Not Graduate	4009	
363	LP002975	Male	Yes	0	Graduate	4158	
365	LP002986	Male	Yes	0	Graduate	5000	
366	LP002989	Male	No	0	Graduate	9200	
	Coapplicant_Income	Loan_Tenor	Credit_History	Property_Area	\		
0	0	144.0	1.0	Urban			
1	1500	144.0	1.0	Urban			
2	1800	144.0	1.0	Urban			
4	0	144.0	1.0	Urban			
5	3422	144.0	1.0	Urban			
..		
361	2167	144.0	1.0	Semiurban			
362	1777	144.0	1.0	Urban			
363	709	144.0	1.0	Urban			
365	2393	144.0	1.0	Rural			
366	0	72.0	1.0	Rural			
	Max_Loan_Amount	Loan_Status					
0	208.219470	1					
1	191.029745	1					
2	257.080058	1					
4	126.175970	1					
5	207.204758	1					
..					
361	196.763751	1					
362	222.360804	1					
363	189.059325	1					
365	282.764073	1					
366	157.296249	1					

[314 rows x 12 columns]