In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("train_u6lujuX_CVtuZ9i.csv", encoding='cp1252')
```

In [3]:
```
df
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849	0.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0
...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0

614 rows × 13 columns

In [4]:
```
df.dropna(axis = 0, inplace=True)
```

In [5]:
```
df
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0
...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0

480 rows × 13 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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 [7]:
```
df.columns
```

Out[7]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [11]:
```
df[["ApplicantIncome","CoapplicantIncome", "LoanAmount", "Loan_Amount_Term","Credit_History"]]
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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 [13]:
```
df.describe()
```

Out[13]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	480.000000	480.000000	480.000000	480.000000	480.000000
mean	5364.231250	1581.093583	144.735417	342.050000	0.854167
std	5668.251251	2617.692267	80.508164	65.212401	0.353307
min	150.000000	0.000000	9.000000	36.000000	0.000000
25%	2898.750000	0.000000	100.000000	360.000000	1.000000
50%	3859.000000	1084.500000	128.000000	360.000000	1.000000
75%	5852.500000	2253.250000	170.000000	360.000000	1.000000
max	81000.000000	33837.000000	600.000000	480.000000	1.000000

In [14]:
```
encode = pd.get_dummies(df[['Gender','Married','Dependents','Education','Self_Employed']])
```

In [20]:
```
X = pd.concat([df[["ApplicantIncome","CoapplicantIncome", "Loan_Amount_Term","Credit_History"]
```

In [21]:
```
X
```

Out[21]:

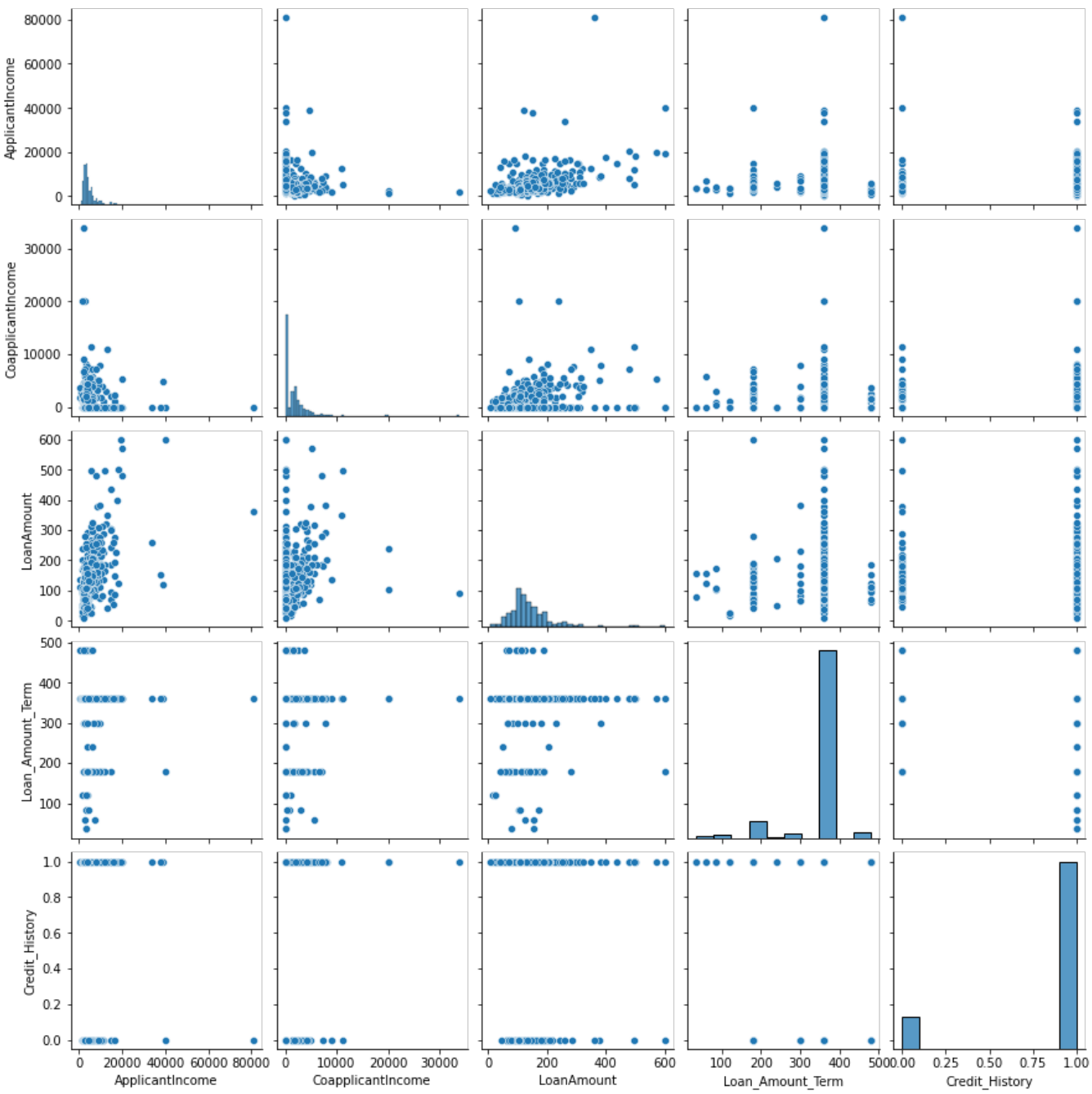
	ApplicantIncome	CoapplicantIncome	Loan_Amount_Term	Credit_History	Gender_Female	Gender_Male	M
1	4583	1508.0	360.0	1.0	0	1	
2	3000	0.0	360.0	1.0	0	1	
3	2583	2358.0	360.0	1.0	0	1	
4	6000	0.0	360.0	1.0	0	1	
5	5417	4196.0	360.0	1.0	0	1	
...	...	...	...	...	...	...	
609	2900	0.0	360.0	1.0	1	0	
610	4106	0.0	180.0	1.0	0	1	
611	8072	240.0	360.0	1.0	0	1	
612	7583	0.0	360.0	1.0	0	1	
613	4583	0.0	360.0	0.0	1	0	

480 rows × 16 columns

In [23]:
```
Y = df[['LoanAmount']]
```

In [13]:
```
sns.pairplot(df)
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x13c2de61a90>
```



In [51]:
```
X.shape
```

Out[51]:
```
(480, 16)
```

In [25]:
```
from sklearn.linear_model import LinearRegression
```

In [26]:
```
reg = LinearRegression()
```

In [27]:
```
model = reg.fit(X,Y)
```

In [28]:
```
model.coef_
```

Out[28]:
```
array([[ 6.93863914e-03,  6.99002669e-03,  9.03727323e-02,
        -4.50538671e+00, -4.72792103e-01,  4.72792103e-01,
        -9.84921922e+00,  9.84921922e+00, -9.80073887e+00,
         3.18261918e+00,  1.32553271e+00,  5.29258698e+00,
         8.71170948e+00, -8.71170948e+00, -3.62295692e+00,
         3.62295692e+00]])
```

In [29]:
```
model.intercept_
```

Out[29]:
```
array([67.99558506])
```

In [37]:
```
# importing and handling test datacases
test_df = pd.read_csv('test_Y3wMUE5_7gLdaTN.csv')
test_df.dropna(axis = 0, inplace=True)
test_df[["ApplicantIncome","CoapplicantIncome", "Loan_Amount_Term","Credit_History"]] = test_
test_encode = pd.get_dummies(test_df[['Gender','Married','Dependents','Education','Self_Emplo
X_test = pd.concat([test_df[["ApplicantIncome","CoapplicantIncome", "Loan_Amount_Term","Credi
Y_test = test_df['LoanAmount']
```

In [39]:
```
Y_pred = model.predict(X_test)
```

In [32]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [40]:
```
mean_absolute_error(Y_test, Y_pred)
```

Out[40]:
```
35.406487444601176
```

In [50]:
```
# sample input
X_test.iloc[10]
```

Out[50]:
```
ApplicantIncome           5667.0
CoapplicantIncome            0.0
Loan_Amount_Term           360.0
Credit_History               1.0
Gender_Female                0.0
Gender_Male                  1.0
Married_No                   1.0
Married_Yes                  0.0
Dependents_0                 0.0
Dependents_1                 1.0
Dependents_2                 0.0
Dependents_3+                0.0
Education_Graduate           1.0
Education_Not Graduate       0.0
Self_Employed_No             1.0
Self_Employed_Yes            0.0
Name: 15, dtype: float64
```

In [49]:
```
Y_test.iloc[10] #actual output
```

Out[49]:
```
131.0
```

In [48]:
```
Y_pred[10][0] #predicted output
```

Out[48]:
```
134.24059463797886
```