code

September 12, 2021

[8]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[9]:
```
train = pd.read_csv('./loan_train.csv')
train.head()
```

[9]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

[10]:
```
test = pd.read_csv('./loan_test.csv')
test.head()
```

[10]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed
0	LP001015	Male	Yes	0	Graduate	No
1	LP001022	Male	Yes	1	Graduate	No
2	LP001031	Male	Yes	2	Graduate	No
3	LP001035	Male	Yes	2	Graduate	No

```
4  LP001051    Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

[13]:
```
train_original = train.copy()
test_original = test.copy()
```

[17]:
```
train.dtypes
```

[17]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

[19]:
```
train.shape
```

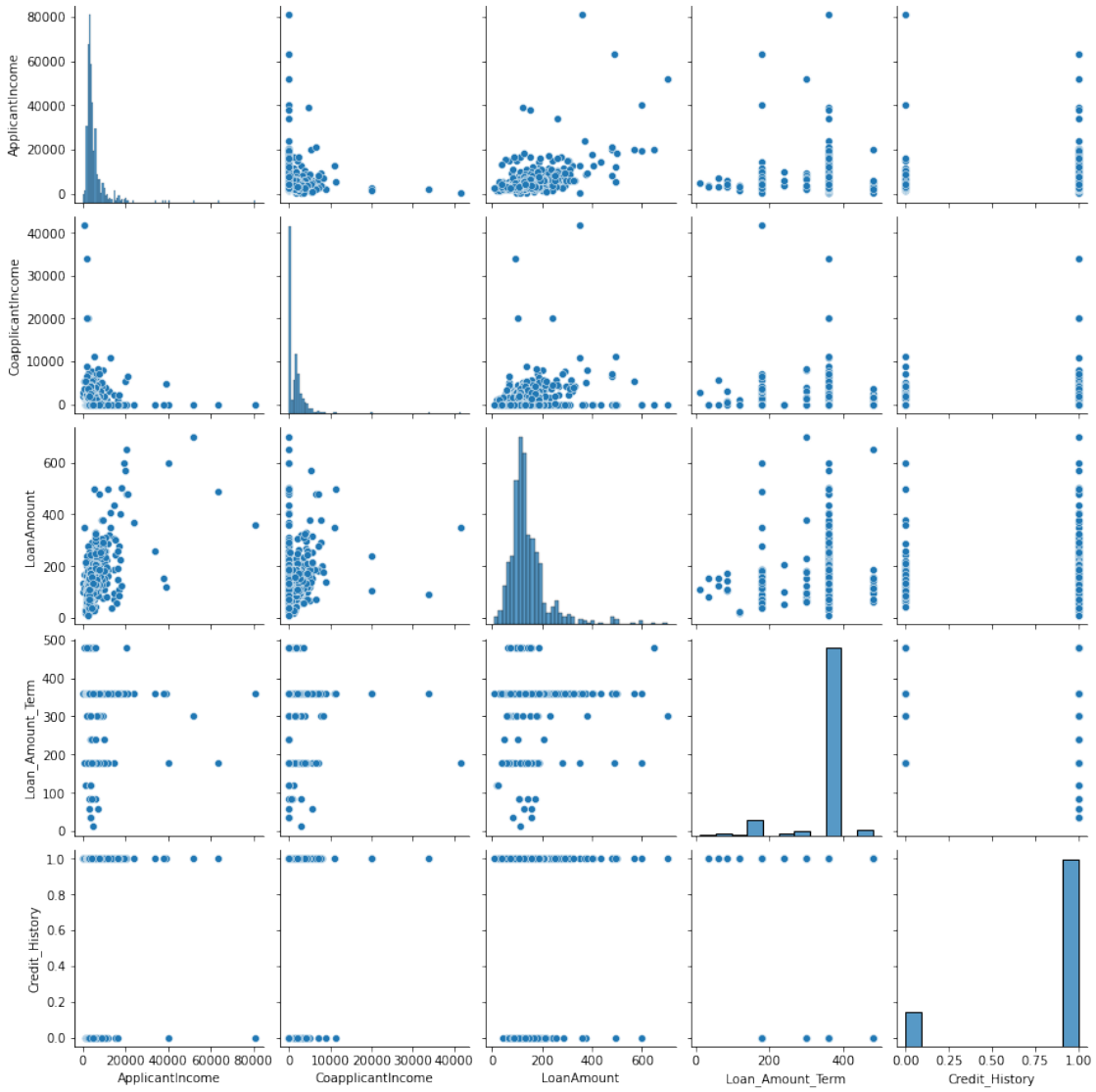
[19]:
```
(614, 13)
```

[21]:
```
test.shape
```

[21]:
```
(367, 12)
```

[22]:
```
sns.pairplot(train)
```

[22]:
```
<seaborn.axisgrid.PairGrid at 0x7f2ccbfaad30>
```



[23]:
```
train.isnull().sum()
```

[23]:
```
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
```

[26]:
```
train['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
train['Married'].fillna(train['Married'].mode()[0], inplace=True)
train['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
train['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
```

[28]:
```
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0],␣
↪inplace=True)
```

[30]:
```
train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
```

[31]:
```
train.isnull().sum()
```

[31]:
```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

[33]:
```
test['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
test['Married'].fillna(train['Married'].mode()[0], inplace=True)
test['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
test['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
test['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
test['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0],␣
↪inplace=True)
test['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
```

[35]:
```
train['LoanAmount_log'] = np.log(train['LoanAmount'])
test['LoanAmount_log'] = np.log(test['LoanAmount'])
```

[36]:
```
train = train.drop('Loan_ID', axis=1)
test = test.drop('Loan_ID', axis=1)
```

[38]:
```
X = train.drop('Loan_Status', 1)
y = train.Loan_Status
```

[41]:
```
X = pd.get_dummies(X)
train = pd.get_dummies(train)
test = pd.get_dummies(test)
```

[42]:
```
from sklearn.model_selection import train_test_split
x_train, x_cv, y_train, y_cv = train_test_split(X, y, test_size=0.3)
```

[44]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
model = LogisticRegression(solver="liblinear")
model.fit(x_train, y_train)
```

[44]:
```
LogisticRegression(solver='liblinear')
```

[45]:
```
pred_cv = model.predict(x_cv)
accuracy_score(y_cv, pred_cv)
```

[45]:
```
0.8432432432432433
```

[46]:
```
pred_test = model.predict(test)
```

[47]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

k = 5
```

[48]:
```
kf = KFold(n_splits=k, random_state=None)
```

[52]:
```
model = LogisticRegression(solver='liblinear')
result = cross_val_score(model, X, y, cv=kf)

print("Avg accuracy before scaling: {}".format(result.mean()))
```

```
Avg accuracy before scaling: 0.809462881514061
```

[ ]:
```

```