In [1]:

```python
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
from numpy import percentile
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```python
data =pd.read_csv("loan_prediction.csv")
data.head(10)
```

Out[3]:

	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
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	Semiurban	N
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	1.0	Urban	Y
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0	1.0	Semiurban	N

In [4]:

```python
data.isnull().sum()
```

Out[4]:

```
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
```

In [7]:

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 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   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [6]:

```python
data.shape
```

Out[6]:

```
(614, 13)
```

In [8]:

```python
data.drop("Loan_ID" , axis =1 , inplace =True)
```

In [9]:

```python
data.head(10)
```

Out[9]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
6	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
7	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	Semiurban	N
8	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	1.0	Urban	Y
9	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0	1.0	Semiurban	N

In [11]:

```python
data.value_counts()
```

Out[11]:

```
Gender  Married  Dependents  Education     Self_Employed  ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  Property_Area  Loan_Status
Female  No       0           Graduate      No             645              3683.0             113.0       480.0             1.0             Rural          Y              1
                                                          1500             1800.0             103.0       360.0             0.0             Semiurban      N              1
Male    Yes      1           Graduate      No             4283             3000.0             172.0       84.0              1.0             Rural          N              1
                                                          3988             0.0                50.0        240.0             1.0             Urban          Y              1
                                                          3875             0.0                67.0        360.0             1.0             Urban          N              1
                                                                                                                                                                           ..
        No       0           Graduate      Yes            10416            0.0                187.0       360.0             0.0             Urban          N              1
                                                          7167             0.0                128.0       360.0             1.0             Urban          Y              1
                                                          7085             0.0                84.0        360.0             1.0             Semiurban      Y              1
                                                          6950             0.0                175.0       180.0             1.0             Semiurban      Y              1
        Yes      3+          Not Graduate  Yes            7100             0.0                125.0       60.0              1.0             Urban          Y              1
Length: 480, dtype: int64
```

In [12]:

```python
# dropping all null values
data1 = data.dropna()
```

In [13]:

```python
data1
```

Out[13]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

480 rows × 12 columns

In [14]:

```python
data1.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':0,'Female':1},'Self_Employed':{'No':0,'Yes':1},'Loan_Status':{'N':0,'Y':1},'Dependents':{"0":0,"1":1,"2":2,"3+":3},
              'Property_Area':{'Rural':1,'Semiurban':2,'Urban':0},'Education':{'Graduate':1,'Not Graduate':0}}, inplace =True)
```

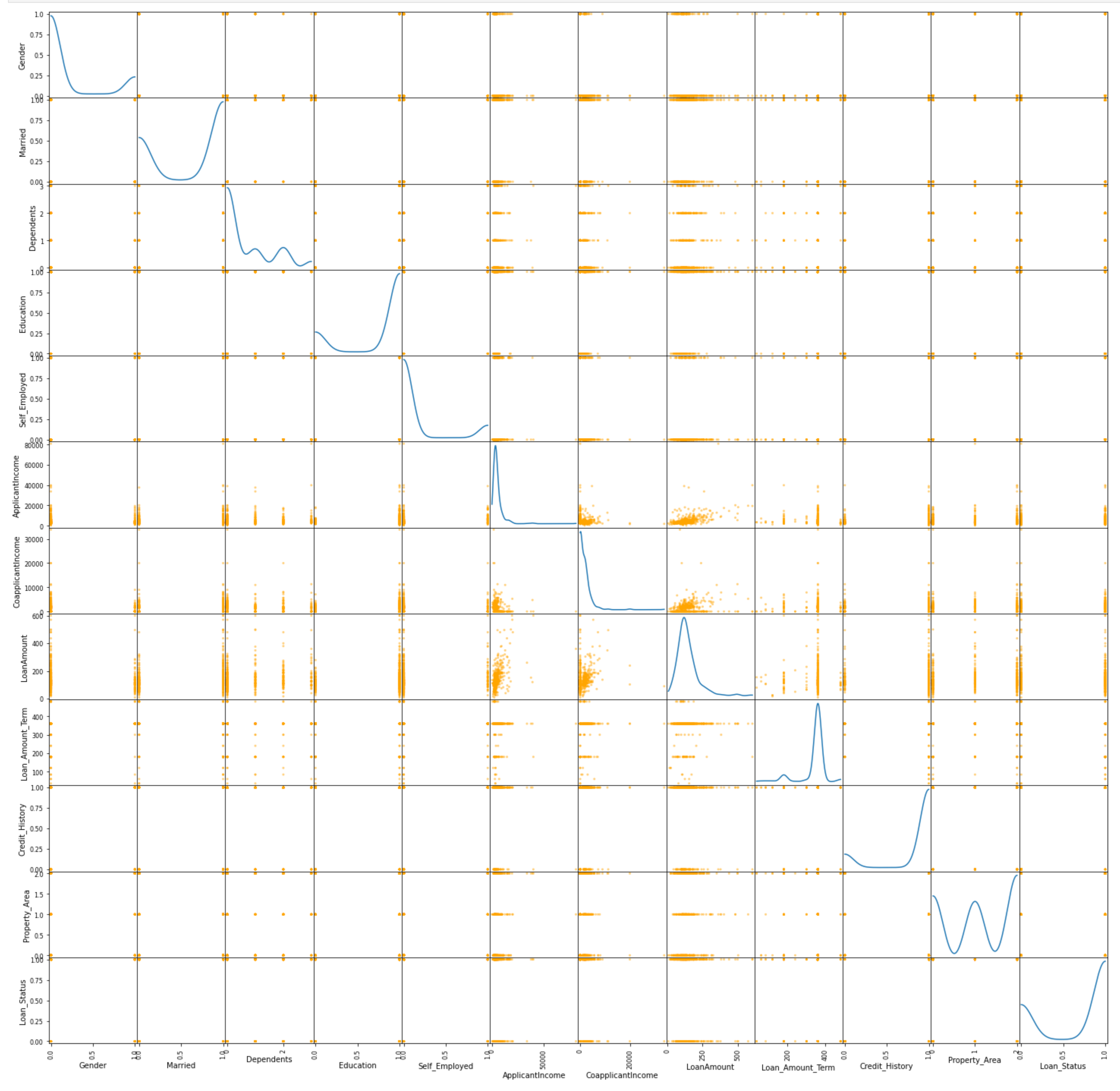
In [15]:

```python
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 480 entries, 1 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             480 non-null    int64
 1   Married            480 non-null    int64
 2   Dependents         480 non-null    int64
 3   Education          480 non-null    int64
 4   Self_Employed      480 non-null    int64
 5   ApplicantIncome    480 non-null    int64
 6   CoapplicantIncome  480 non-null    float64
 7   LoanAmount         480 non-null    float64
 8   Loan_Amount_Term   480 non-null    float64
 9   Credit_History     480 non-null    float64
 10  Property_Area      480 non-null    int64
 11  Loan_Status        480 non-null    int64
dtypes: float64(4), int64(8)
memory usage: 48.8 KB
```

In [17]:

```python
scatter_matrix(data1 , figsize=(25,25) ,diagonal="kde" ,color="orange")
plt.show()
```



In [18]:

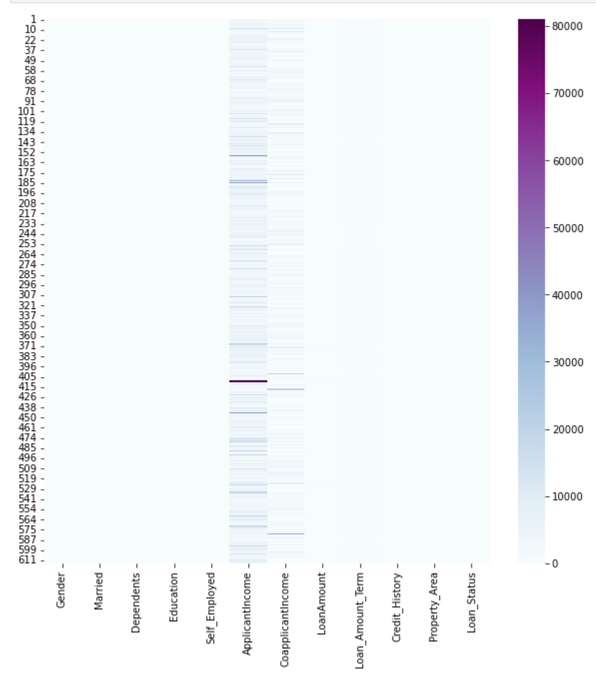
```python
data1.describe()
```

Out[18]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
count	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000
mean	0.179167	0.647917	0.777083	0.797917	0.137500	5364.231250	1581.093583	144.735417	342.050000	0.854167	1.085417	0.691667
std	0.383892	0.478118	1.020815	0.401973	0.344734	5668.251251	2617.692267	80.508164	65.212401	0.353307	0.839398	0.462287
min	0.000000	0.000000	0.000000	0.000000	0.000000	150.000000	0.000000	9.000000	36.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000	0.000000	2898.750000	0.000000	100.000000	360.000000	1.000000	0.000000	0.000000
50%	0.000000	1.000000	0.000000	1.000000	0.000000	3859.000000	1084.500000	128.000000	360.000000	1.000000	1.000000	1.000000
75%	0.000000	1.000000	2.000000	1.000000	0.000000	5852.500000	2253.250000	170.000000	360.000000	1.000000	2.000000	1.000000
max	1.000000	1.000000	3.000000	1.000000	1.000000	81000.000000	33837.000000	600.000000	480.000000	1.000000	2.000000	1.000000

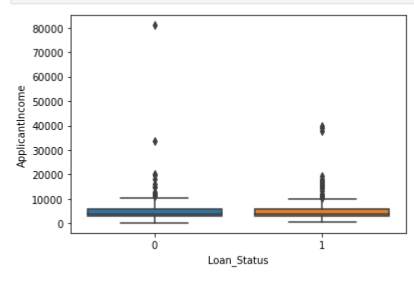
In [21]:

```python
plt.figure(figsize =(10,10))
sns.heatmap(data1 ,cmap="BuPu")
plt.show()
```



In [23]:

```python
sns.boxplot(x ='Loan_Status' , y ="ApplicantIncome" ,data =data1)
plt.show()
```



In [27]:

```python
# building the module
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

In [25]:

```python
x =data1.drop("Loan_Status" ,axis =1)
y =data1["Loan_Status"]

print(x)
print(y)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
1         0        1           1          1              0             4583
2         0        1           0          1              1             3000
3         0        1           0          0              0             2583
4         0        0           0          1              0             6000
5         0        1           2          1              1             5417
..      ...      ...         ...        ...            ...              ...
609       1        0           0          1              0             2900
610       0        1           3          1              0             4106
611       0        1           1          1              0             8072
612       0        1           2          1              0             7583
613       1        0           0          1              1             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
5               4196.0       267.0             360.0             1.0
..                 ...         ...               ...             ...
609                0.0        71.0             360.0             1.0
610                0.0        40.0             180.0             1.0
611              240.0       253.0             360.0             1.0
612                0.0       187.0             360.0             1.0
613                0.0       133.0             360.0             0.0

     Property_Area
1                1
2                0
3                0
4                0
5                0
..             ...
609              1
610              1
611              0
612              0
613              2

[480 rows x 11 columns]
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 480, dtype: int64
```

In [28]:

```python
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=2)
print(x.shape,x_train.shape,x_test.shape)
```

```
(480, 11) (432, 11) (48, 11)
```

In [29]:

```python
classi =svm.SVC(kernel='linear')
```

In [30]:

```python
classi.fit(x_train,y_train)
```

Out[30]:

```
SVC(kernel='linear')
```

In [32]:

```python
# pradicting train data
x_train_predic = classi.predict(x_train)
x_train_accuracy =accuracy_score(x_train_predic ,y_train)

print("Accuracy of x_data train :",x_train_accuracy)
```

```
Accuracy of x_data train : 0.7731481481481481
```

In [33]:

```python
# predicting test data
x_test_predic = classi.predict(x_test)
x_test_accuracy =accuracy_score(x_test_predic ,y_test)

print("Accuracy of test :",x_test_accuracy)
```

```
Accuracy of test : 0.7916666666666666
```

In [ ]: