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

In [41]:
```
train = pd.read_csv("C:\\Users\\aarus\\Downloads\\archive (4)\\train_u6lujuX_CVtuZ9i.csv")
test = pd.read_csv("C:\\Users\\aarus\\Downloads\\archive (4)\\test_Y3wMUE5_7gLdaTN.csv")
```

In [42]:
```
train
```

Out[42]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Lo
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	
...	...	...	...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	

614 rows × 13 columns

EXPLORATORY DATA ANALYSIS ON DATASET

In [6]:
```
train.head()  #Displays list of first 5 columns
```

Out[6]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

In [43]:
```
train.info()   #Info on training set
```

```
<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 [44]:
```
test.head()  #Displays 5 columns of test set
```

Out[44]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

In [45]:
```
test.info()  #info on test set
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             356 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         357 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      344 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

In [47]:
```
train['Loan_Status'].value_counts()
```

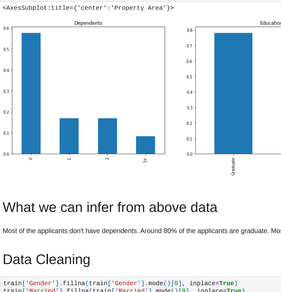
Out[47]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [48]:
```
train['Property_Area'].value_counts()  #counts values of types of property
```

Out[48]:
```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

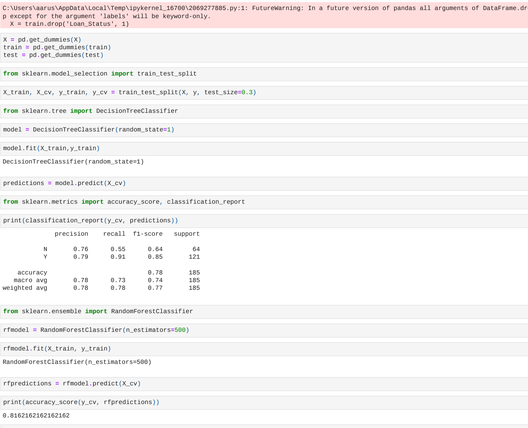
In [17]:
```
train['Property_Area'].value_counts().plot.bar(figsize=(10,6), title='Property Area')  #bar chart of property area
```

Out[17]:
```
<AxesSubplot:title={'center':'Property Area'}>
```



In [18]:
```
plt.figure(1)
plt.subplot(221)
train['Gender'].value_counts(normalize=True).plot.bar(figsize=(20,10), title= 'Gender')
plt.subplot(222)
train['Married'].value_counts(normalize=True).plot.bar(title= 'Married')
plt.subplot(223)
train['Self_Employed'].value_counts(normalize=True).plot.bar(title= 'Self Employed')
plt.subplot(224)
train['Credit_History'].value_counts(normalize=True).plot.bar(title= 'Credit_History')
```

Out[18]:
```
<AxesSubplot:title={'center':'Credit_History'}>
```



In [49]:
```
plt.figure(1)
plt.subplot(131)
train['Dependents'].value_counts(normalize=True).plot.bar(figsize=(24,6), title='Dependents')
plt.subplot(132)
train['Education'].value_counts(normalize=True).plot.bar(title='Education')
plt.subplot(133)
train['Property_Area'].value_counts(normalize=True).plot.bar(title='Property Area')
```

Out[49]:
```
<AxesSubplot:title={'center':'Property Area'}>
```


What we can infer from above data

Most of the applicants don't have dependents. Around 80% of the applicants are graduate. Most of the applicants are from Semiurban area.

Data Cleaning

In [50]:
```
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['Dependents'].mode()[0], inplace=True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)
train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
```

In [51]:
```
test['Gender'].fillna(test['Gender'].mode()[0], inplace=True)
test['Married'].fillna(test['Married'].mode()[0], inplace=True)
test['Dependents'].fillna(test['Dependents'].mode()[0], inplace=True)
test['Self_Employed'].fillna(test['Dependents'].mode()[0], inplace=True)
test['Credit_History'].fillna(test['Credit_History'].mode()[0], inplace=True)
test['Loan_Amount_Term'].fillna(test['Loan_Amount_Term'].mode()[0], inplace=True)
test['LoanAmount'].fillna(test['LoanAmount'].median(), inplace=True)
```

Moving to applying Random Forest Classifer Algorithm

In [31]:
```
df=pd.DataFrame(train)
df2=pd.DataFrame(test)
```

In [52]:
```
X = train.drop('Loan_Status', 1)
y = train['Loan_Status']
```

```
C:\Users\aarus\AppData\Local\Temp\ipykernel_16708\2069277885.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.dro
p except for the argument 'labels' will be keyword-only.
  X = train.drop('Loan_Status', 1)
```

In [53]:
```
X = pd.get_dummies(X)
train = pd.get_dummies(train)
test = pd.get_dummies(test)
```

In [40]:
```
from sklearn.model_selection import train_test_split
```

In [54]:
```
X_train, X_cv, y_train, y_cv = train_test_split(X, y, test_size=0.3)
```

In [55]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [56]:
```
model = DecisionTreeClassifier(random_state=1)
```

In [57]:
```
model.fit(X_train,y_train)
```

Out[57]:
```
DecisionTreeClassifier(random_state=1)
```

In [58]:
```
predictions = model.predict(X_cv)
```

In [59]:
```
from sklearn.metrics import accuracy_score, classification_report
```

In [60]:
```
print(classification_report(y_cv, predictions))
```

```
              precision    recall  f1-score   support

           N       0.76      0.55      0.64        64
           Y       0.79      0.91      0.85       121

    accuracy                           0.78       185
   macro avg       0.78      0.73      0.74       185
weighted avg       0.78      0.78      0.77       185
```

In [61]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [62]:
```
rfmodel = RandomForestClassifier(n_estimators=500)
```

In [63]:
```
rfmodel.fit(X_train, y_train)
```

Out[63]:
```
RandomForestClassifier(n_estimators=500)
```

In [64]:
```
rfpredictions = rfmodel.predict(X_cv)
```

In [65]:
```
print(accuracy_score(y_cv, rfpredictions))
```

```
0.8162162162162162
```

In [ ]: