Loan_Prediction

April 9, 2023

CONTENTS

1 Intorduction

In this project we have create a Machine Learning Model based on given information to predict whether or not loan will get approved.

What we will do?

- Visualize and compare the data.
- Pre-processing of data.
- Handling Missing Value.
- Analyze Categorical and Numerical Data.
- Outliers Detection
- Different Machine Learning Algorithms and Evaluation Matrices for evaluation.

What we will Use?

- Different Python Libraries such as `sklearn`,`matplotlib`,`numpy`,`seaborn`.
- Different Machine Learning Algorithm for Prediction Model and select best of them–
  - Logisctic Regression
  - KNeighbors Classifier
  - Support Vecort Machine(SVC)
  - DecisionTreeClassifier

NOTE: I, Ukant doing this project under my Data Science Internship at CodeClause. Currently studying and learning about this field, so if there is any mistake I have made, please feel free comment below.

[1]:
```
# Importing some libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

2 Data Disscussion and Collection

[2]:
```
df_train = pd.read_csv("../Data/train_u6lujuX_CVtuZ9i.csv")
df_test = pd.read_csv("../Data/test_Y3wMUE5_7gLdaTN.csv")
```

[3]:
```
# Shape of data
print(df_train.shape)
print(df_test.shape)
```

```
(614, 13)
(367, 12)
```

[4]:
```
df_train.head()
```

[4]:

	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]:
```
df_test.head()
```

[5]:

	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

[6]:
```
# Stastical Summary of Continous data
df_train.describe()
```

[6]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

- In above summary feature `Credit_History` only contain `0` and `1`, so we need to change its type

[7]:
```
# print(df_train.info())
print(df_test.info())
```

```
<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
None
```

- we can see there is some values are missing in both object and float64 datatype.

3 Data Preprocessing

[8]:
```
# Statical summary of categorical Data
df_train['Credit_History'] = df_train['Credit_History'].astype('O')
df_train.describe(include='O')
```

[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	\
count	614	601	611	599	614	582	
unique	614	2	2	4	2	2	
top	LP001002	Male	Yes	0	Graduate	No	
freq	1	489	398	345	480	500	

	Credit_History	Property_Area	Loan_Status
count	564.0	614	614
unique	2.0	3	2
top	1.0	Semiurban	Y
freq	475.0	233	422

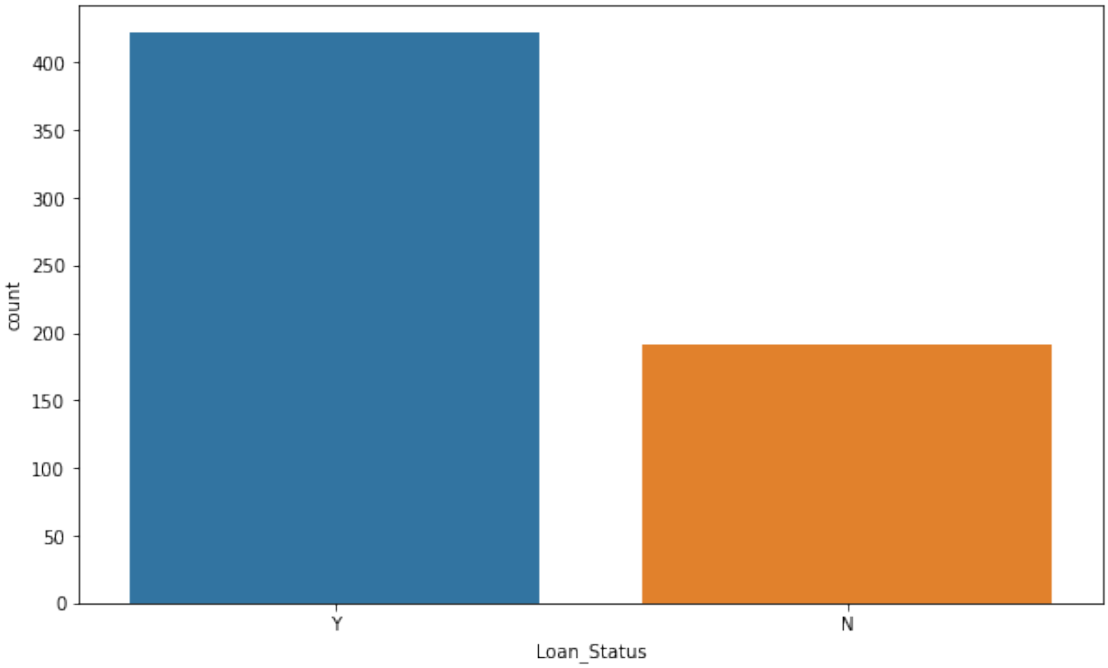
[9]:
```
# Checking for duplicate values
print(df_train.duplicated().sum())
print(df_test.duplicated().sum())
```

```
0
0
```

[10]:
```
## Checking for null values
print(df_train.isnull().sum())
# print(df_test.isnull().sum())
```

```
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
```

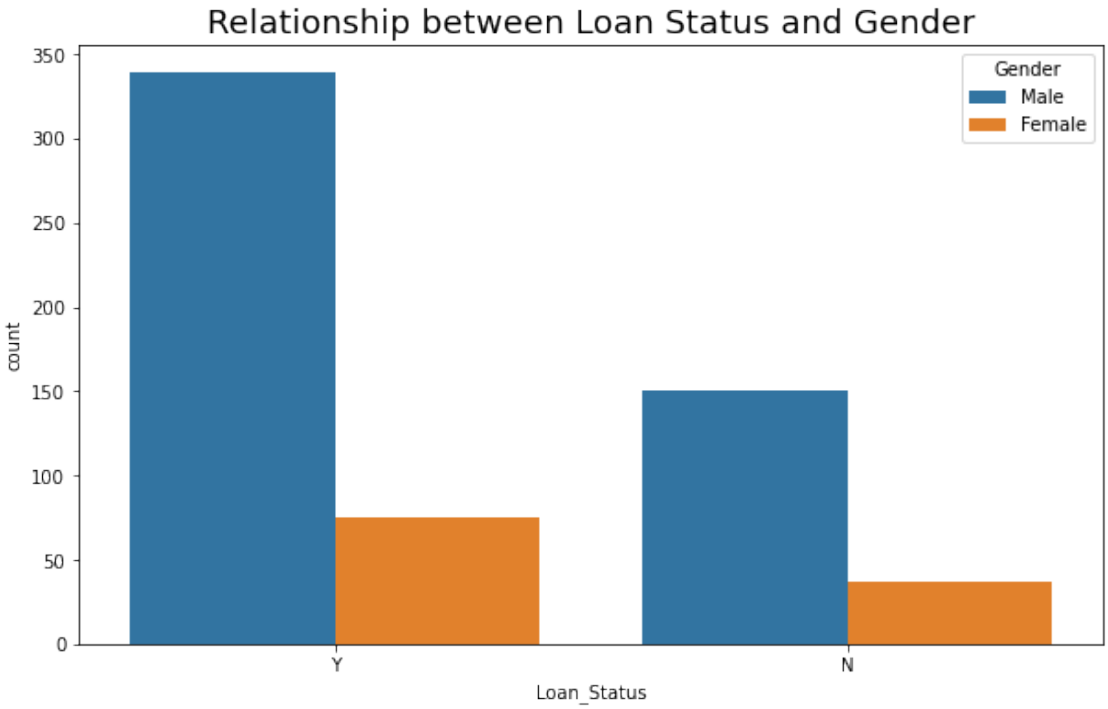
[11]:
```
## Let's analyze our traget feature
plt.figure(figsize=(10,6))
sns.countplot(df_train['Loan_Status'])
plt.show()

print("The weight of Y class : %.2f" % (df_train['Loan_Status'].
 ↪value_counts()[0] / len(df_train)*100))
print("The weight of N class : %.2f" % (df_train['Loan_Status'].
 ↪value_counts()[1] / len(df_train)*100))
```




```
The weight of Y class : 68.73
The weight of N class : 31.27
```

Bivariate Analysis

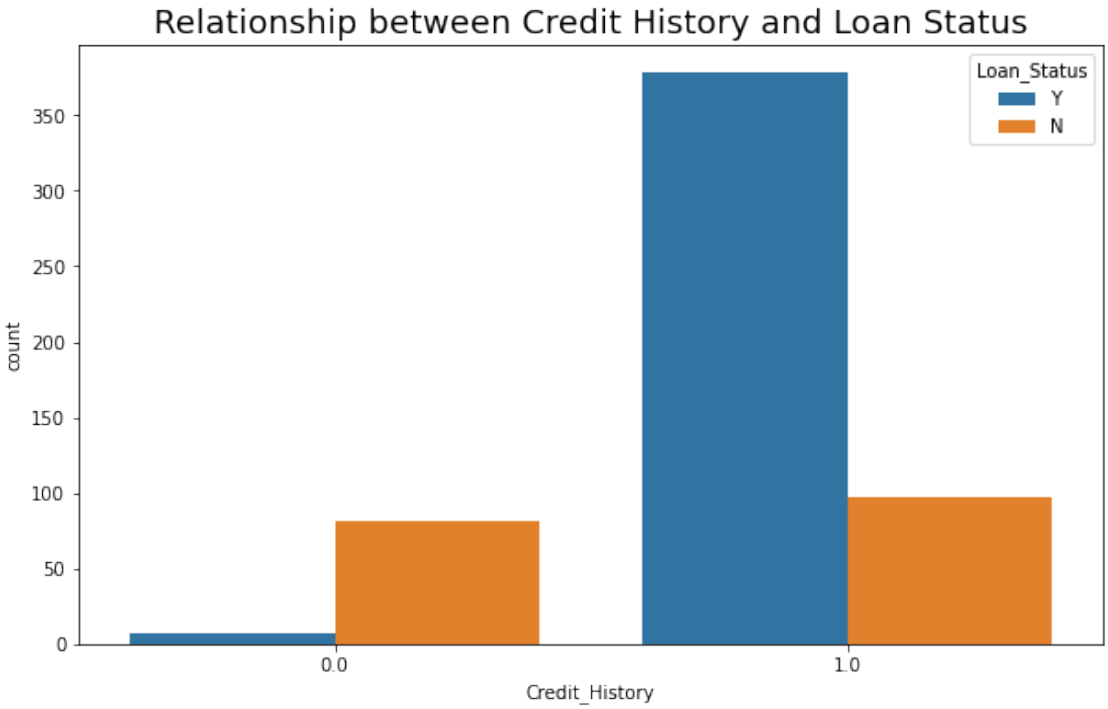
[12]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='Loan_Status',hue='Gender',data=df_train)
plt.title("Relationship between Loan Status and Gender",fontsize=18)
plt.show()
```



Observation

- Most males got the more loans in comparision to females

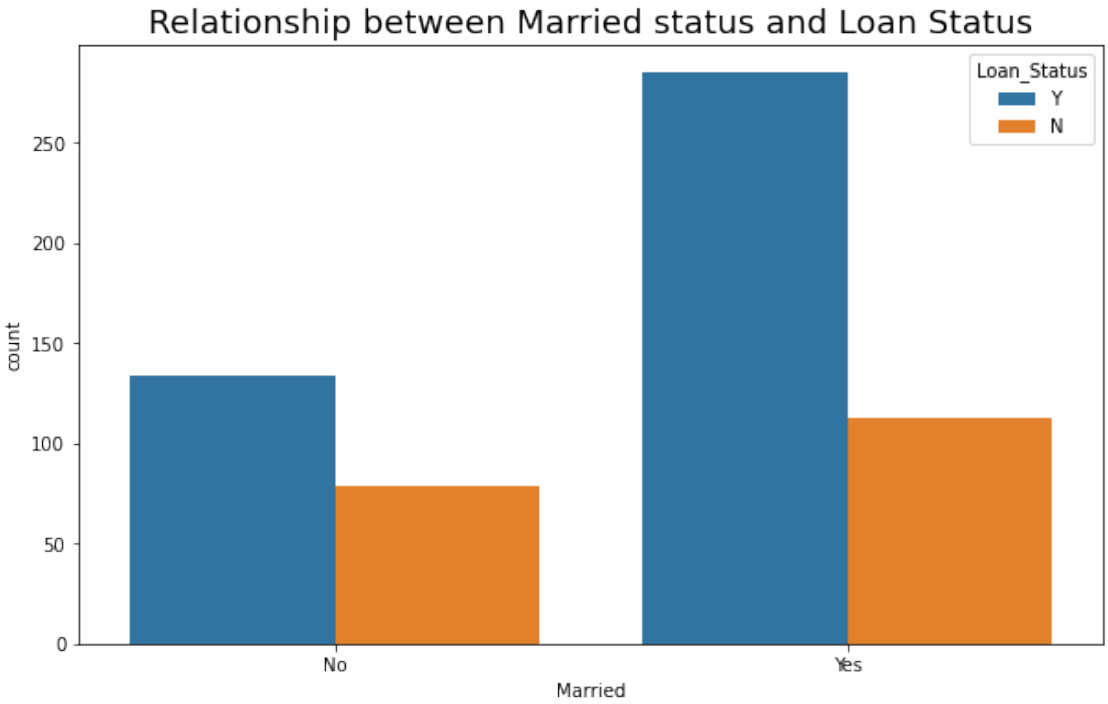
[13]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='Credit_History',hue='Loan_Status',data=df_train)
plt.title("Relationship between Credit History and Loan Status",fontsize=18)
plt.show()
```



Observation

- The more clear Credit History(1) more chance to get loan
- Not approving loan with credit history(0)

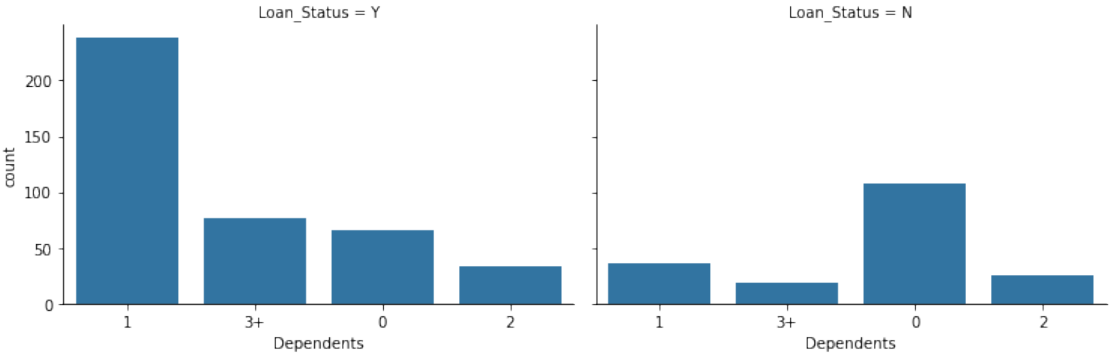
[14]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='Married',hue='Loan_Status',data=df_train)
plt.title("Relationship between Married status and Loan Status",fontsize=18)
plt.show()
```



Observation

- Married people have better chance to get loan

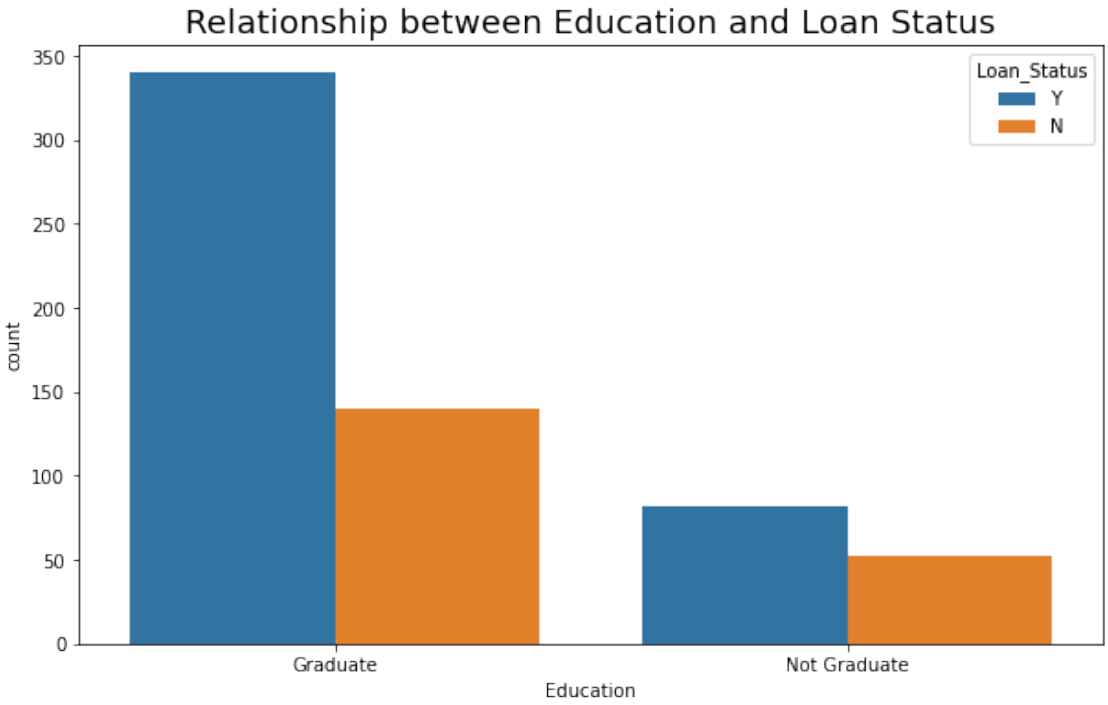
[15]:
```
grid = sns.FacetGrid(col='Loan_Status',data=df_train,size=3.5,aspect=1.5)
grid.map(sns.countplot,'Dependents')
plt.show()
```



Observation

- Dependents with 1 have more chances to get loan

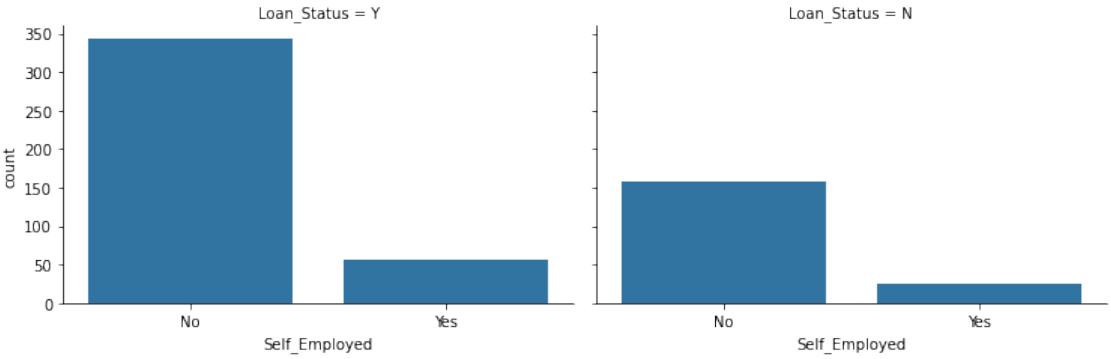
[16]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='Education',hue='Loan_Status',data=df_train)
plt.title("Relationship between Education and Loan Status",fontsize=18)
plt.show()
```



Observation

- From above plot Graduate's have better chance of getting a loan

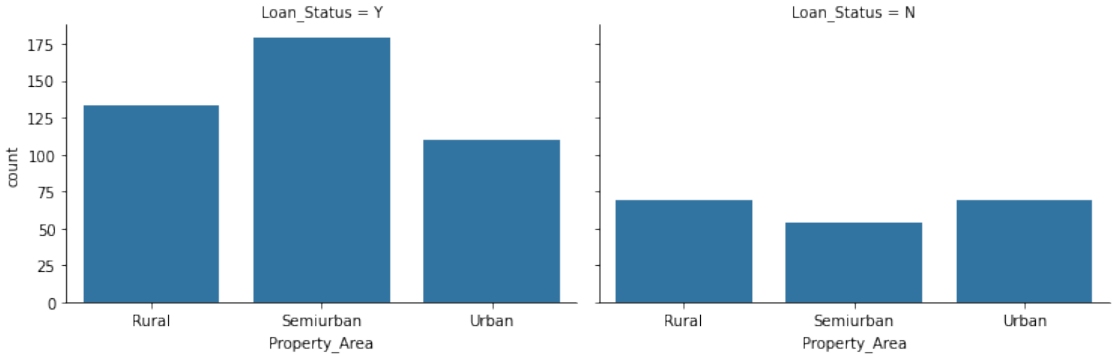
[17]:
```
grid = sns.FacetGrid(col='Loan_Status',data=df_train,size=3.5,aspect=1.5)
grid.map(sns.countplot,'Self_Employed')
plt.show()
```



Observation

- We can say, Self Employed people got more loan than others

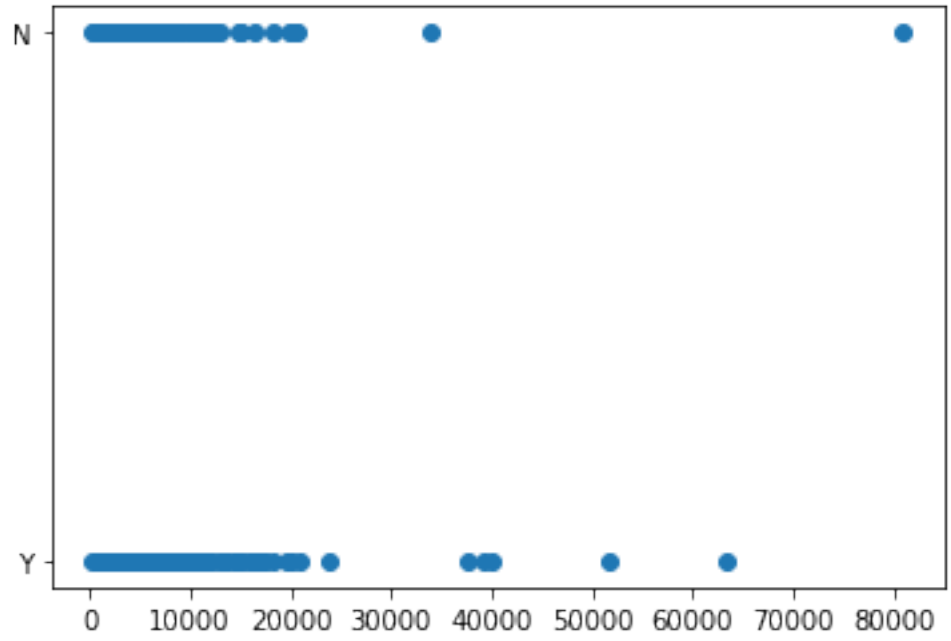
[18]:
```
grid = sns.FacetGrid(col='Loan_Status',data=df_train,size=3.5,aspect=1.5)
grid.map(sns.countplot,'Property_Area')
plt.show()
```



Observation

- Here Semiurban Property Area get more loans in comparision to other area

[19]:
```
plt.scatter(df_train['ApplicantIncome'],df_train['Loan_Status'])
plt.show()

# No Pattern
```



Univariate Analysis

[20]:
```
df_train.isnull().sum().sort_values(ascending=False)
```

[20]:
```
Credit_History       50
Self_Employed        32
LoanAmount           22
Dependents           15
Loan_Amount_Term     14
Gender               13
Married               3
Loan_ID               0
Education             0
ApplicantIncome       0
CoapplicantIncome     0
Property_Area         0
Loan_Status           0
dtype: int64
```

[21]:
```
## Dropping Loan Id
df_train.drop('Loan_ID',axis=1,inplace=True)
```

[22]:
```
## Separating the categorical and numerical data
cat_data = []
num_data = []
```

```
for name,dtype in enumerate(df_train.dtypes):
    if dtype == object:
        cat_data.append(df_train.iloc[:,name])
    else:
        num_data.append(df_train.iloc[:,name])
```

[23]:
```
cat_data = pd.DataFrame(cat_data).T
num_data = pd.DataFrame(num_data).T
```

[24]:
```
num_data
```

[24]:
```
     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term
0             5849.0                0.0         NaN             360.0
1             4583.0             1508.0       128.0             360.0
2             3000.0                0.0        66.0             360.0
3             2583.0             2358.0       120.0             360.0
4             6000.0                0.0       141.0             360.0
..               …                  …           …               …
609           2900.0                0.0        71.0             360.0
610           4106.0                0.0        40.0             180.0
611           8072.0              240.0       253.0             360.0
612           7583.0                0.0       187.0             360.0
613           4583.0                0.0       133.0             360.0

[614 rows x 4 columns]
```

[25]:
```
cat_data
```

[25]:
```
     Gender Married Dependents     Education Self_Employed Credit_History  \
0      Male      No          0      Graduate            No             1.0
1      Male     Yes          1      Graduate            No             1.0
2      Male     Yes          0      Graduate           Yes             1.0
3      Male     Yes          0  Not Graduate            No             1.0
4      Male      No          0      Graduate            No             1.0
..      …     …         …           …           …            …
609  Female      No          0      Graduate            No             1.0
610    Male     Yes         3+      Graduate            No             1.0
611    Male     Yes          1      Graduate            No             1.0
612    Male     Yes          2      Graduate            No             1.0
613  Female      No          0      Graduate           Yes             0.0

    Property_Area Loan_Status
0           Urban           Y
1           Rural           N
2           Urban           Y
3           Urban           Y
4           Urban           Y
```

```
..             …          …
609         Rural          Y
610         Rural          Y
611         Urban          Y
612         Urban          Y
613     Semiurban          N

[614 rows x 8 columns]
```

[26]:
```
## Handling missing values in categorical data
cat_data = cat_data.apply(lambda x: x.fillna(x.value_counts().index[0]))
cat_data.isnull().sum().sort_values(ascending=False)
```

[26]:
```
Gender            0
Married           0
Dependents        0
Education         0
Self_Employed     0
Credit_History    0
Property_Area     0
Loan_Status       0
dtype: int64
```

[27]:
```
## Handling missing values in numerical data
num_data.fillna(method='bfill',inplace=True)
num_data.isnull().sum().sort_values(ascending=False)
```

[27]:
```
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
dtype: int64
```

[28]:
```
## Categorical Data Preprocessing

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[29]:
```
target_values = {'Y':0,'N':1}
target = cat_data['Loan_Status']
cat_data.drop('Loan_Status',axis=1,inplace=True)
target = target.map(target_values)
```

[30]:
```
for i in cat_data:
    cat_data[i] = le.fit_transform(cat_data[i])
```

[31]:
```
cat_data
```

[31]:
```
     Gender  Married  Dependents  Education  Self_Employed  Credit_History  \
0         1        0           0          0              0               1
1         1        1           1          0              0               1
2         1        1           0          0              1               1
3         1        1           0          1              0               1
4         1        0           0          0              0               1
..      ...      ...         ...        ...            ...             ...
609       0        0           0          0              0               1
610       1        1           3          0              0               1
611       1        1           1          0              0               1
612       1        1           2          0              0               1
613       0        0           0          0              1               0

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

[614 rows x 7 columns]
```

[32]:
```
df = pd.concat([cat_data,num_data,target],axis=1)
df
```

[32]:
```
     Gender  Married  Dependents  Education  Self_Employed  Credit_History  \
0         1        0           0          0              0               1
1         1        1           1          0              0               1
2         1        1           0          0              1               1
3         1        1           0          1              0               1
4         1        0           0          0              0               1
..      ...      ...         ...        ...            ...             ...
609       0        0           0          0              0               1
610       1        1           3          0              0               1
611       1        1           1          0              0               1
612       1        1           2          0              0               1
613       0        0           0          0              1               0

     Property_Area  ApplicantIncome  CoapplicantIncome  LoanAmount  \
0                2           5849.0                0.0       128.0
1                0           4583.0             1508.0       128.0
```

```
2                2           3000.0                0.0        66.0
3                2           2583.0             2358.0       120.0
4                2           6000.0                0.0       141.0
..             ...              ...              ...         ...
609              0           2900.0                0.0        71.0
610              0           4106.0                0.0        40.0
611              2           8072.0              240.0       253.0
612              2           7583.0                0.0       187.0
613              1           4583.0                0.0       133.0

     Loan_Amount_Term  Loan_Status
0               360.0            0
1               360.0            1
2               360.0            0
3               360.0            0
4               360.0            0
..                ...          ...
609             360.0            0
610             180.0            0
611             360.0            0
612             360.0            0
613             360.0            1

[614 rows x 12 columns]
```

4 Data Splitting

[33]:
```
X = pd.concat([num_data,cat_data],axis=1)
y = target
```

[34]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

[35]:
```
print('X_test shape',X_test.shape)
print('X_train shape',X_train.shape)
print('y_test shape',y_test.shape)
print('y_train shape',y_train.shape)
```

```
X_test shape (123, 11)
X_train shape (491, 11)
y_test shape (123,)
y_train shape (491,)
```

5 Model Implementation and Evaluation

[36]:
```
## Various Machine Learning Algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

models = {
    'LogisticRegression': LogisticRegression(random_state=42),
    'KNeighborsClassifier': KNeighborsClassifier(),
    'SVC': SVC(random_state=42),
    'DecisionTreeClassifier':DecisionTreeClassifier(max_depth=1,random_state=42)
}
```

[37]:
```
from sklearn.metrics import precision_score , recall_score, f1_score, log_loss,␣
 ↪accuracy_score
def loss(y_true,y_pred,retu=False):
    pre = precision_score(y_true,y_pred)
    rec = recall_score(y_true,y_pred)
    f1 = f1_score(y_true,y_pred)
    loss = log_loss(y_true,y_pred)
    acc = accuracy_score(y_true,y_pred)

    if retu:
        return pre, rec, f1, loss, acc
    else:
        print(' pre: %.3f\n rec: %.3f\n f1: %.3f\n loss: %.3f\n acc: %.3f'%␣
 ↪(pre,rec,f1,loss,acc))
```

[38]:
```
def train_eval(models,X,y):
    for name, model in models.items():
        print(name, ":")
        model.fit(X,y)
        loss(y,model.predict(X))
        print('-'*10)


train_eval(models,X_train,y_train)
```

```
LogisticRegression :
 pre: 0.918
 rec: 0.438
 f1: 0.593
 loss: 6.472
 acc: 0.813
----------
KNeighborsClassifier :
```

```
 pre: 0.675
 rec: 0.340
 f1: 0.452
 loss: 8.863
 acc: 0.743
----------
SVC :
 pre: 1.000
 rec: 0.007
 f1: 0.013
 loss: 10.692
 acc: 0.690
----------
DecisionTreeClassifier :
 pre: 0.929
 rec: 0.425
 f1: 0.583
 loss: 6.542
 acc: 0.811
----------
```

[39]:
```
df_test
```

[39]:

	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	
..	…	…	…	…	…	…	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	
363	LP002975	Male	Yes	0	Graduate	No	
364	LP002980	Male	No	0	Graduate	No	
365	LP002986	Male	Yes	0	Graduate	No	
366	LP002989	Male	No	0	Graduate	Yes	

	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	
..	…	…	…	…	
362	4009	1777	113.0	360.0	
363	4158	709	115.0	360.0	
364	3250	1993	126.0	360.0	
365	5000	2393	158.0	360.0	

```
366             9200                  0        98.0             180.0

     Credit_History Property_Area
0               1.0         Urban
1               1.0         Urban
2               1.0         Urban
3               NaN         Urban
4               1.0         Urban
..              …           …
362             1.0         Urban
363             1.0         Urban
364             NaN     Semiurban
365             1.0         Rural
366             1.0         Rural

[367 rows x 12 columns]
```

6 For validation of program

We have done the training and testing of our model with training data `df_train`. Now we have process the validation data and user input for prediction.

[40]:
```
df_test.drop('Loan_ID',axis=1,inplace=True)
df_test
```

[40]:
```
    Gender Married Dependents     Education Self_Employed  ApplicantIncome  \
0     Male     Yes          0      Graduate            No             5720
1     Male     Yes          1      Graduate            No             3076
2     Male     Yes          2      Graduate            No             5000
3     Male     Yes          2      Graduate            No             2340
4     Male      No          0  Not Graduate            No             3276
..     …     …        …           …           …              …
362   Male     Yes         3+  Not Graduate           Yes             4009
363   Male     Yes          0      Graduate            No             4158
364   Male      No          0      Graduate            No             3250
365   Male     Yes          0      Graduate            No             5000
366   Male      No          0      Graduate           Yes             9200

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                    0       110.0             360.0             1.0
1                 1500       126.0             360.0             1.0
2                 1800       208.0             360.0             1.0
3                 2546       100.0             360.0             NaN
4                    0        78.0             360.0             1.0
..                 …         …                 …              …
362               1777       113.0             360.0             1.0
363                709       115.0             360.0             1.0
```

```
364                1993       126.0             360.0              NaN
365                2393       158.0             360.0              1.0
366                   0        98.0             180.0              1.0

    Property_Area
0           Urban
1           Urban
2           Urban
3           Urban
4           Urban
..            …
362         Urban
363         Urban
364     Semiurban
365         Rural
366         Rural

[367 rows x 11 columns]
```

list of preprocessing we have used - remove duplicate - seprate handle the missing value - transform the cat_data - –:transform target data - concat them

[41]:
```
# Handling duplicate values
df_test.duplicated().sum()
```

[41]:
```
1
```

[42]:
```
## Changing the data type of `Credit History`
df_test['Credit_History'] = df_test['Credit_History'].astype('O')
```

[43]:
```
## Seprating categorical and numerical data
Tcat_data = []
Tnum_data = []

for name, dtype in enumerate(df_test.dtypes):
    if dtype == object:
        Tcat_data.append(df_test.iloc[:,name])
    else:
        Tnum_data.append(df_test.iloc[:,name])

Tcat_data = pd.DataFrame(Tcat_data).T
Tnum_data = pd.DataFrame(Tnum_data).T
```

[44]:
```
## Handling missing value in categorical data
Tcat_data = Tcat_data.apply(lambda x: x.fillna(x.value_counts().index[0]))
Tcat_data.isnull().sum()
```

[44]:
```
Gender            0
Married           0
Dependents        0
Education         0
Self_Employed     0
Credit_History    0
Property_Area     0
dtype: int64
```

[45]:
```
## Handling missing value in numerical data
Tnum_data.fillna(method='bfill',inplace=True)
Tnum_data.isnull().sum()
```

[45]:
```
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
dtype: int64
```

[46]:
```
Tcat_data
```

[46]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	\
0	Male	Yes	0	Graduate	No	1.0	
1	Male	Yes	1	Graduate	No	1.0	
2	Male	Yes	2	Graduate	No	1.0	
3	Male	Yes	2	Graduate	No	1.0	
4	Male	No	0	Not Graduate	No	1.0	
..	…	…	…	…	…	…	
362	Male	Yes	3+	Not Graduate	Yes	1.0	
363	Male	Yes	0	Graduate	No	1.0	
364	Male	No	0	Graduate	No	1.0	
365	Male	Yes	0	Graduate	No	1.0	
366	Male	No	0	Graduate	Yes	1.0	

	Property_Area
0	Urban
1	Urban
2	Urban
3	Urban
4	Urban
..	…
362	Urban
363	Urban
364	Semiurban
365	Rural
366	Rural

```
[367 rows x 7 columns]
```

[47]:
```
## Transforming the categorical data storing into other dataframe
Transform_cat_data = pd.DataFrame()
for data in Tcat_data:
    Transform_cat_data[data] = le.fit_transform(Tcat_data[data])
```

[48]:
```
## Createing validation dataframe
X_valid = pd.concat([Tnum_data,Transform_cat_data],axis=1)
```

[49]:
```
## Predicting target with logisticRegression
predict = models['LogisticRegression'].predict(X_valid)
```

[50]:
```
output = pd.concat([Tnum_data,Tcat_data],axis=1)
```

[51]:
```
## Collecting all validating data into one dataframe
predict = pd.DataFrame(predict)
output = pd.concat([output,predict],axis=1)
output = output.rename({0:'Predicted'},axis='columns')
```

[52]:
```
output
```

[52]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Gender	\
0	5720.0	0.0	110.0	360.0	Male	
1	3076.0	1500.0	126.0	360.0	Male	
2	5000.0	1800.0	208.0	360.0	Male	
3	2340.0	2546.0	100.0	360.0	Male	
4	3276.0	0.0	78.0	360.0	Male	
..	…	…	…	…	…	
362	4009.0	1777.0	113.0	360.0	Male	
363	4158.0	709.0	115.0	360.0	Male	
364	3250.0	1993.0	126.0	360.0	Male	
365	5000.0	2393.0	158.0	360.0	Male	
366	9200.0	0.0	98.0	180.0	Male	

	Married	Dependents	Education	Self_Employed	Credit_History	\
0	Yes	0	Graduate	No	1.0	
1	Yes	1	Graduate	No	1.0	
2	Yes	2	Graduate	No	1.0	
3	Yes	2	Graduate	No	1.0	
4	No	0	Not Graduate	No	1.0	
..	…	…	…	…	…	
362	Yes	3+	Not Graduate	Yes	1.0	
363	Yes	0	Graduate	No	1.0	
364	No	0	Graduate	No	1.0	
365	Yes	0	Graduate	No	1.0	
366	No	0	Graduate	Yes	1.0	

```
    Property_Area  Predicted
0           Urban          0
1           Urban          0
2           Urban          0
3           Urban          0
4           Urban          0
..            …        …
362         Urban          0
363         Urban          0
364     Semiurban          0
365         Rural          0
366         Rural          0

[367 rows x 12 columns]
```

[53]:
```
## Saving validation file as output.csv
output.to_csv('../Data/output.csv')
```

6.0.1 Thank You :)

- By Ukant Jadia https://ukantjadia.me/linkedin