ASSIGNMENT/ TASK 11

GO_STP_12574
SHWETA JHA

Predict Loan Eligibility for Dream Housing Finance company Dream Housing Finance company deals in all kinds of home loans. They have presence across all urban, semi urban and rural areas. Customer first applies for home loan and after that company validates the customer eligibility for loan.

Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have provided a dataset to identify the customers segments that are eligible for loan amount so that they can specifically target these customers.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as matplot
import seaborn as sns
```

```
df=pd.read_csv("/content/loan.csv")
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000

```
df.tail()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
609	LP002978	Female	No	0	Graduate	No	290
610	LP002979	Male	Yes	3+	Graduate	No	410

```
df.columns
```

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

```
df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hist
count	614.000000	614.000000	592.000000	600.00000	564.000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842
std	6109.041673	2926.248369	85.587325	65.12041	0.364
min	150.000000	0.000000	9.000000	12.00000	0.000
25%	2877.500000	0.000000	100.000000	360.00000	1.000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000
max	81000.000000	41667.000000	700.000000	480.00000	1.000

```
df.shape
```

```
(614, 13)
```

```
df.dtypes
```

```
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
```

```
df.isna().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
```

```
df.Loan_Status.value_counts()
```

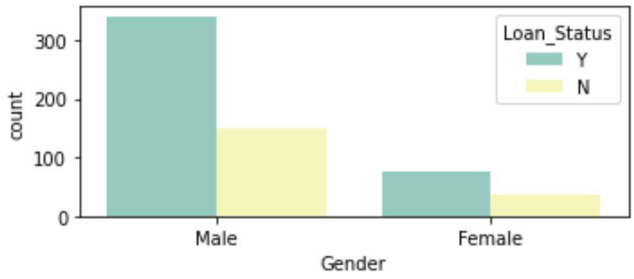
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

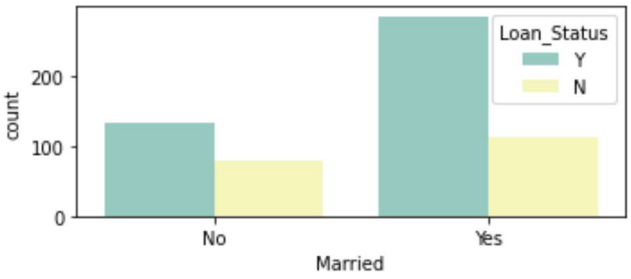
```
df.groupby('Loan_Status').count()
```

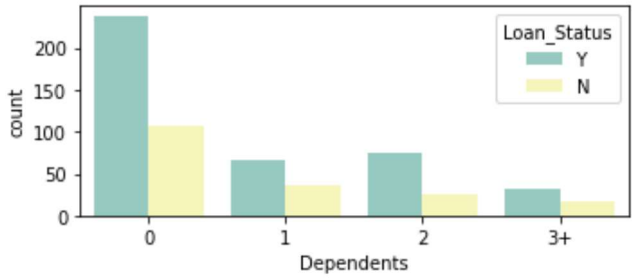
Loan_Status	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applica
N	192	187	192	186	192	183	
Y	422	414	419	413	422	399	

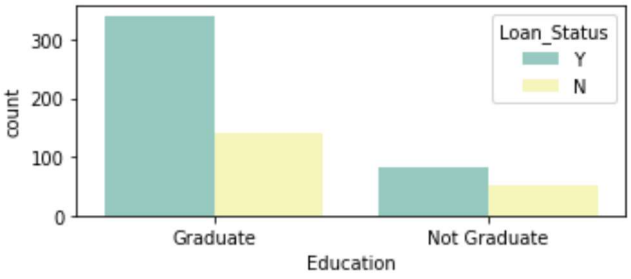
```
categorical_columns=['Gender','Married','Dependents','Education','Self_Employed','Property
```

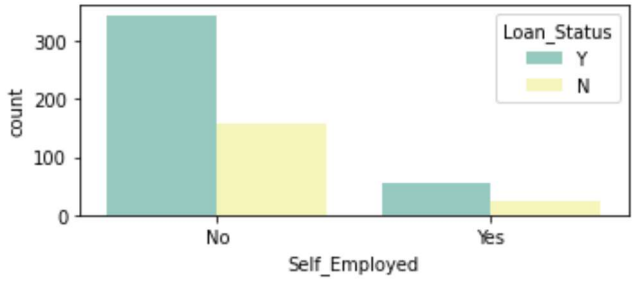
```
fig,axes=plt.subplots(4,2,figsize=(12,15))
for idx,cat_col in enumerate(categorical_columns):
  row,col=idx//2,idx%2
  sns.countplot(x=cat_col,data=df,hue='Loan_Status',ax=axes[row,col],palette="Set3")
plt.subplots_adjust(hspace=1)
```

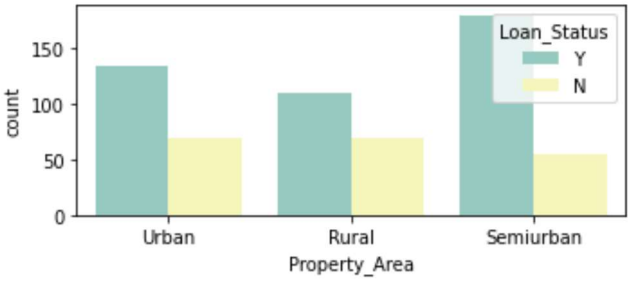






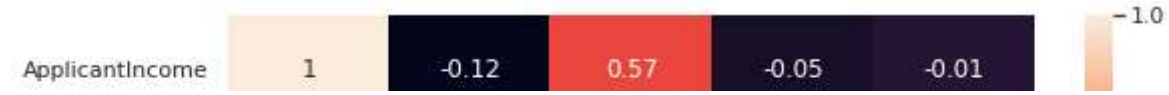






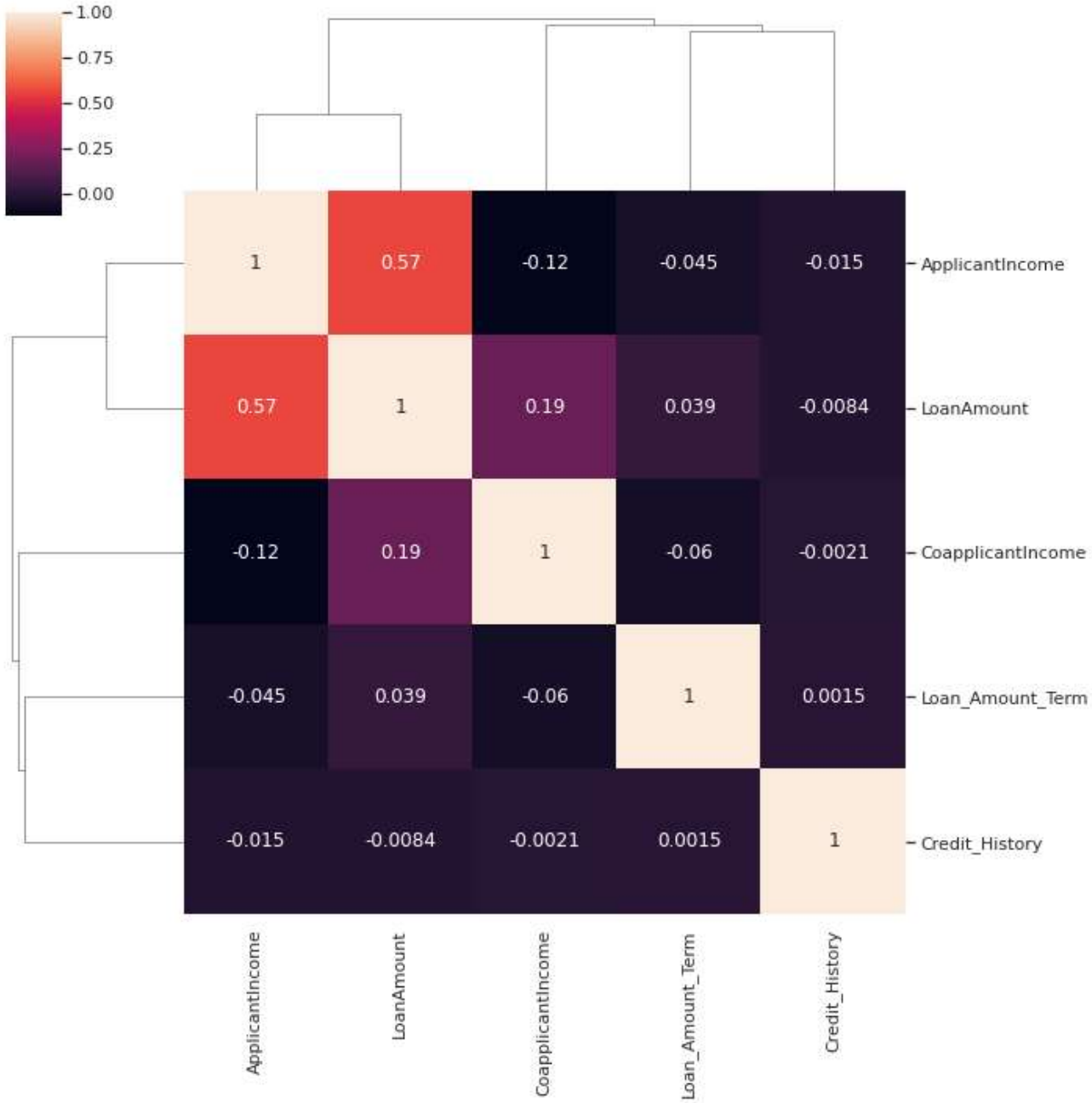

```
sns.set(rc={'figure.figsize':(9,5)})
correlation_matrix = df.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbefb254fd0>
```




```
sns.clustermap(df.corr(),annot=True)
```

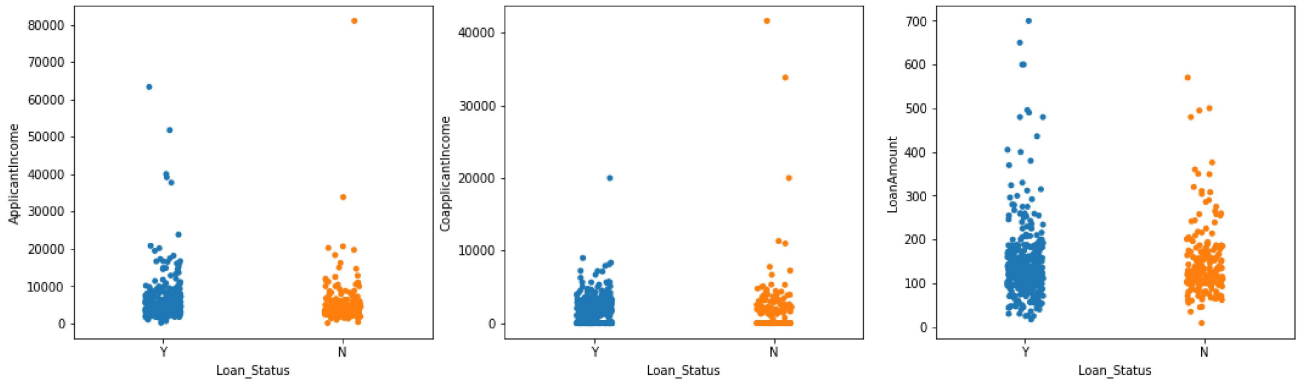
```
<seaborn.matrix.ClusterGrid at 0x7fbefb0d79d0>
```




```
sns.countplot(x='LoanAmount',data=df).set_title("Loan Amount Details");
```


```
numerical_columns=['ApplicantIncome','CoapplicantIncome','LoanAmount']
```

```
fig,axes=plt.subplots(1,3,figsize=(18,5))
for idx,cat_col in enumerate(numerical_columns):
  row,col=idx//2,idx%2
  sns.stripplot(y=cat_col,data=df,x='Loan_Status',ax=axes[idx])
plt.subplots_adjust(hspace=1)
```




```
sns.pairplot(df,hue='Loan_Status',corner=True,height=9.5,kind='hist');
```

Data Processing

```
df_encoded=pd.get_dummies(df,drop_first=True)
df_encoded.head()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	5849	0.0	NaN	360.0	1.0
1	4583	1508.0	128.0	360.0	1.0
2	3000	0.0	66.0	360.0	1.0
3	2583	2358.0	120.0	360.0	1.0

```
from sklearn.model_selection import train_test_split
x=df_encoded.drop(columns='Loan_Status_Y')
y=df_encoded['Loan_Status_Y']
Xtrain,Xtest,Ytrain,Ytest=train_test_split(x,y,train_size=0.85,random_state=7)


from sklearn.impute import SimpleImputer
imp=SimpleImputer(strategy='mean')
imp_train=imp.fit(Xtrain)
Xtrain=imp_train.transform(Xtrain)
Xtrain_imp=imp_train.transform(Xtest)


from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score,f1_score


tree_clf=DecisionTreeClassifier()
tree_clf.fit(Xtrain,Ytrain)
y_pred=tree_clf.predict(Xtrain)
y_pred
```

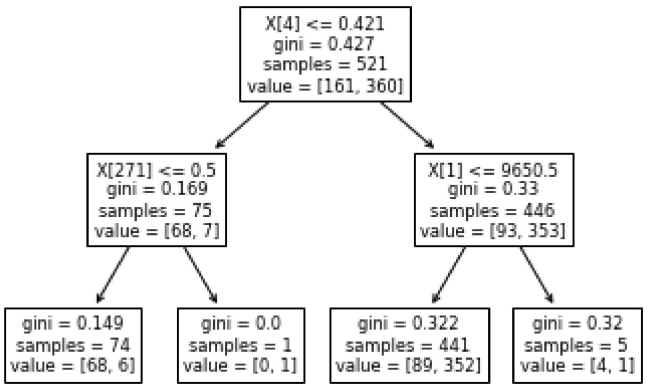
```
array([0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1], dtype=uint8)
```

```
print("Accuray is:",accuracy_score(Ytrain,y_pred))
```

```
Accuray is: 1.0
```

```
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=2)
clf.fit(Xtrain,Ytrain)
tree.plot_tree(clf)
```

```
[Text(167.4, 181.2, 'X[4] <= 0.421\ngini = 0.427\nsamples = 521\nvalue = [161, 360]')
 Text(83.7, 108.72, 'X[271] <= 0.5\ngini = 0.169\nsamples = 75\nvalue = [68, 7]'),
 Text(41.85, 36.23999999999998, 'gini = 0.149\nsamples = 74\nvalue = [68, 6]'),
 Text(125.55000000000001, 36.23999999999998, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]
 Text(251.10000000000002, 108.72, 'X[1] <= 9650.5\ngini = 0.33\nsamples = 446\nvalue
 Text(209.25, 36.23999999999998, 'gini = 0.322\nsamples = 441\nvalue = [89, 352]'),
 Text(292.95, 36.23999999999998, 'gini = 0.32\nsamples = 5\nvalue = [4, 1]')]
```




```
text_represntation=tree.export_text(clf)
print(text_represntation)
```

```
|--- feature_4 <= 0.42
|   |--- feature_271 <= 0.50
|   |   |--- class: 0
|   |--- feature_271 >  0.50
|   |   |--- class: 1
|--- feature_4 >  0.42
|   |--- feature_1 <= 9650.50
|   |   |--- class: 1
|   |--- feature_1 >  9650.50
|   |   |--- class: 0
```

✓ 0s completed at 11:30 PM