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

In [2]:
```
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
df=pd.read_csv("C8 loan csv").dropna()

df
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
0	LP001015	Male	Yes	0	Graduate	No	5720	
1	LP001022	Male	Yes	1	Graduate	No	3076	
2	LP001031	Male	Yes	2	Graduate	No	5000	
4	LP001051	Male	No	0	Not Graduate	No	3276	
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	
...	...	...	...	...	...	...	...	
361	LP002969	Male	Yes	1	Graduate	No	2269	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	
363	LP002975	Male	Yes	0	Graduate	No	4158	

In [4]:
```
df.dropna(inplace=True)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 289 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            289 non-null    object 
 1   Gender             289 non-null    object 
 2   Married            289 non-null    object 
 3   Dependents         289 non-null    object 
 4   Education          289 non-null    object 
 5   Self_Employed      289 non-null    object 
 6   ApplicantIncome    289 non-null    int64  
 7   CoapplicantIncome  289 non-null    int64  
 8   LoanAmount         289 non-null    float64
 9   Loan_Amount_Term   289 non-null    float64
 10  Credit_History     289 non-null    float64
 11  Property_Area      289 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 29.4+ KB
```

In [6]:
```
feature_matrix = df[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_
target_vector = df['Property_Area']
```

In [7]:
```
feature_matrix.shape
```

Out[7]:
```
(289, 5)
```

In [8]:
```
target_vector.shape
```

Out[8]:
```
(289,)
```

In [9]:
```
from sklearn.preprocessing import StandardScaler
```

In [10]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [11]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

Out[11]:
```
LogisticRegression()
```

In [12]:
```
feature_matrix.shape
```

Out[12]:
```
(289, 5)
```

In [13]:
```
target_vector.shape
```

Out[13]:
```
(289,)
```

In [14]:
```
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [16]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

Out[16]:
```
LogisticRegression()
```

In [17]:
```
observation=df[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amoun
```

In [18]:
```
prediction = logr.predict(observation)
prediction
```

Out[18]:
```
array(['Urban', 'Urban', 'Urban', 'Urban', 'Rural', 'Urban', 'Urban',
       'Rural', 'Urban', 'Urban', 'Urban', 'Rural', 'Urban', 'Rural',
       'Rural', 'Rural', 'Urban', 'Urban', 'Rural', 'Rural', 'Urban',
       'Rural', 'Urban', 'Urban', 'Urban', 'Rural', 'Urban', 'Urban',
       'Rural', 'Urban', 'Urban', 'Urban', 'Urban', 'Rural', 'Urban',
       'Urban', 'Rural', 'Urban', 'Rural', 'Rural', 'Urban', 'Urban',
       'Urban', 'Urban', 'Rural', 'Rural', 'Urban', 'Rural', 'Urban',
       'Urban', 'Rural', 'Urban', 'Rural', 'Urban', 'Urban', 'Urban',
       'Rural', 'Urban', 'Urban', 'Rural', 'Rural', 'Rural', 'Urban',
       'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Rural',
       'Urban', 'Urban', 'Rural', 'Urban', 'Rural', 'Urban', 'Urban',
       'Urban', 'Rural', 'Rural', 'Urban', 'Urban', 'Urban', 'Urban',
       'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Rural',
       'Rural', 'Rural', 'Rural', 'Rural', 'Urban', 'Rural', 'Urban',
       'Rural', 'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Urban',
       'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Rural',
       'Urban', 'Rural', 'Urban', 'Urban', 'Urban', 'Urban', 'Urban',
       'Urban', 'Rural', 'Urban', 'Urban', 'Rural', 'Urban', 'Rural',
       'Rural', 'Urban', 'Urban', 'Urban', 'Rural', 'Urban', 'Urban',
```

In [19]:
```
logr.classes_
```

Out[19]:
```
array(['Rural', 'Semiurban', 'Urban'], dtype=object)
```

In [20]:
```
logr.predict_proba(observation)[0][1]
```

Out[20]:
```
0.0
```