

```
In [1]: import pandas as pd
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, recall_score, precision_score, confusion_matrix
from sklearn import tree
from sklearn.metrics import roc_auc_score
```

```
In [2]: loan_data=pd.read_csv("loan_data.csv")
```

```
In [3]: loan_data.head()
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
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	



```
In [4]: loan_data.shape
```

Out[4]: (614, 13)

```
In [5]: loan_data.isnull().sum()
```

Out[5]:

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 [6]: loan_data.dtypes
```

```
Out[6]: 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
```

```
In [7]: loan_data.nunique()
```

```
Out[7]: Loan_ID          614
Gender           2
Married          2
Dependents       4
Education         2
Self_Employed    2
ApplicantIncome  505
CoapplicantIncome 287
LoanAmount       203
Loan_Amount_Term 10
Credit_History   2
Property_Area    3
Loan_Status       2
dtype: int64
```

```
In [8]: loan_data.isnull().sum()
```

```
Out[8]: 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 [9]: loan_data['Loan_Status'].value_counts()
```

```
Out[9]: Y      422  
N      192  
Name: Loan_Status, dtype: int64
```

```
In [10]: loan_data['Credit_History'].value_counts()
```

```
Out[10]: 1.0      475  
0.0      89  
Name: Credit_History, dtype: int64
```

```
In [11]: loan_data['Dependents'].value_counts()
```

```
Out[11]: 0      345  
1      102  
2      101  
3+      51  
Name: Dependents, dtype: int64
```

```
In [12]: loan_data['Gender'].value_counts()
```

```
Out[12]: Male      489  
Female     112  
Name: Gender, dtype: int64
```

```
In [13]: loan_data['Loan_Amount_Term'].value_counts()
```

```
Out[13]: 360.0      512  
180.0       44  
480.0       15  
300.0       13  
240.0        4  
84.0        4  
120.0        3  
60.0        2  
36.0        2  
12.0        1  
Name: Loan_Amount_Term, dtype: int64
```

```
In [14]: loan_data['Self_Employed'].value_counts()
```

```
Out[14]: No      500  
Yes     82  
Name: Self_Employed, dtype: int64
```

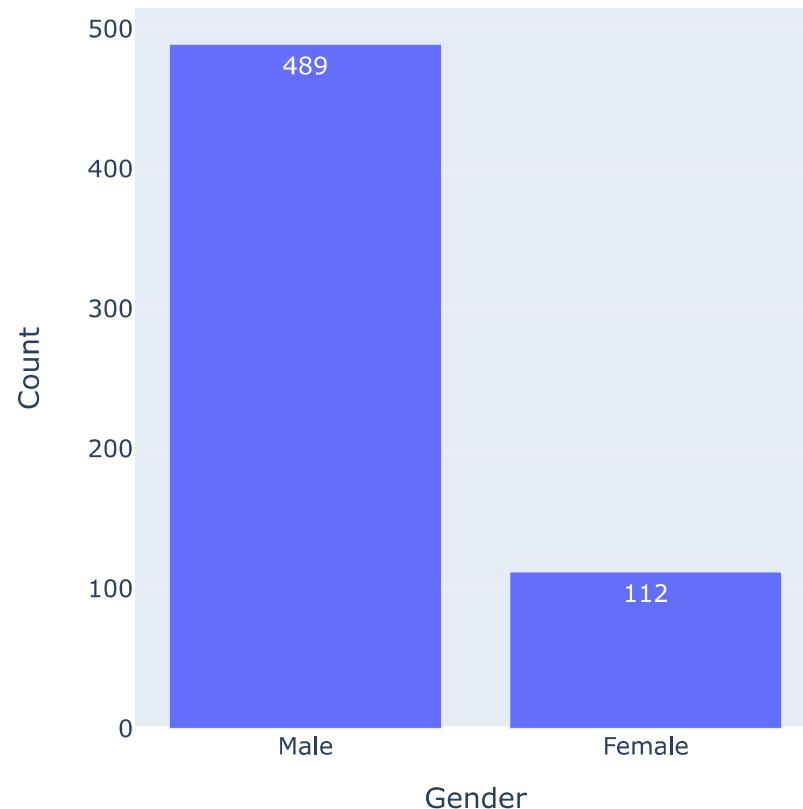
```
In [15]: loan_data['Gender'].value_counts()
```

```
Out[15]: Male      489  
Female     112  
Name: Gender, dtype: int64
```

In []:

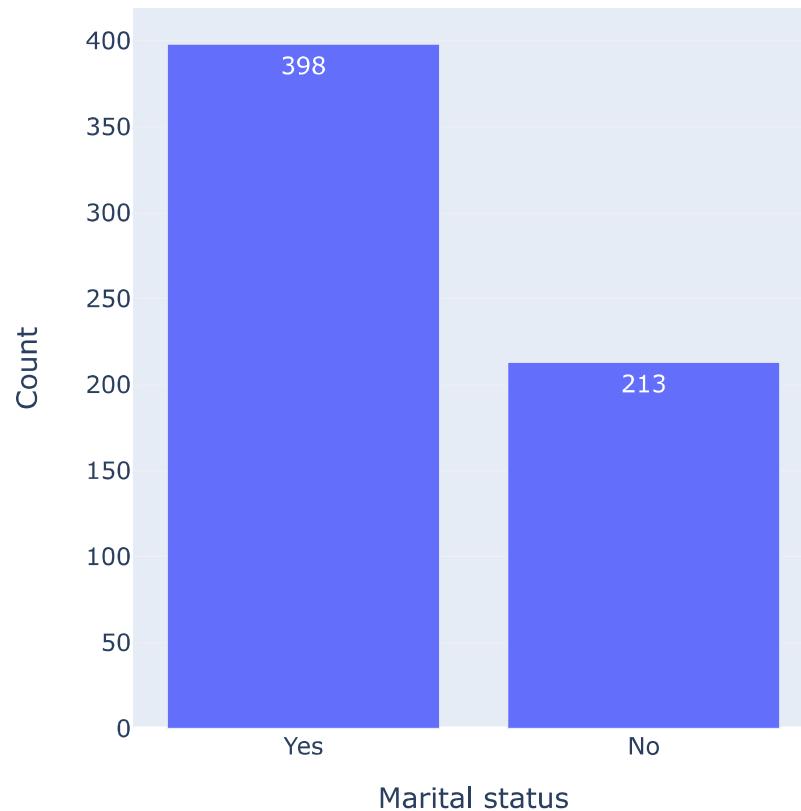
```
In [16]: fig = px.bar(data_frame=loan_data, x=loan_data['Gender'].value_counts().index, y=fig.update_layout(title='Number of Males and Females',xaxis_title='Gender',yaxis_ fig.show()
```

Number of Males and Females



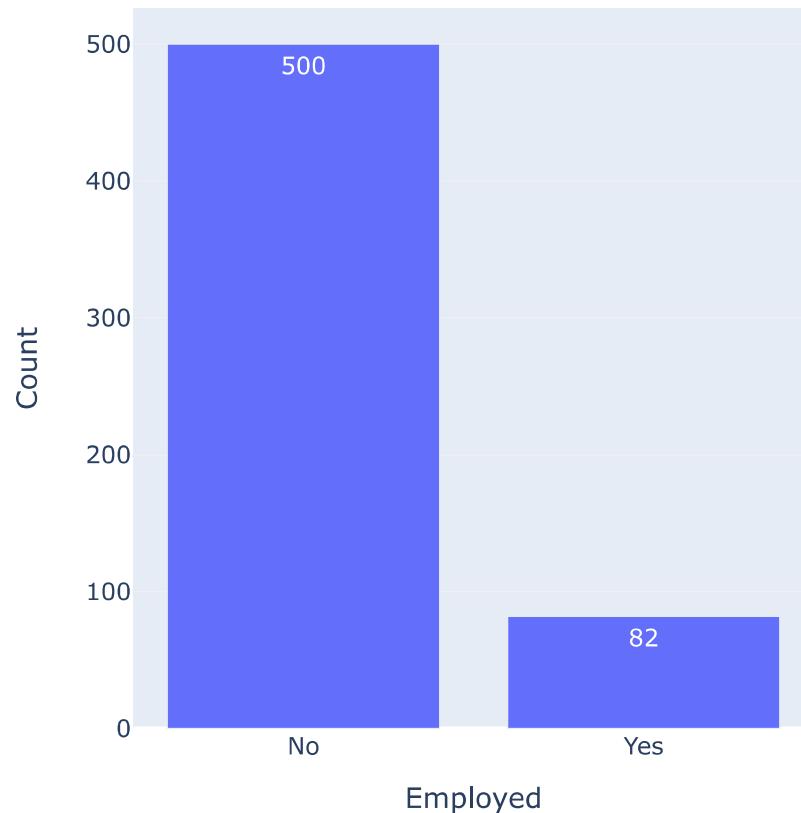
```
In [17]: fig = px.bar(data_frame=loan_data, x=loan_data['Married'].value_counts().index, y  
fig.update_layout(title='Number of Married and Unmarried',xaxis_title='Marital st  
fig.show()
```

Number of Married and Unmarried



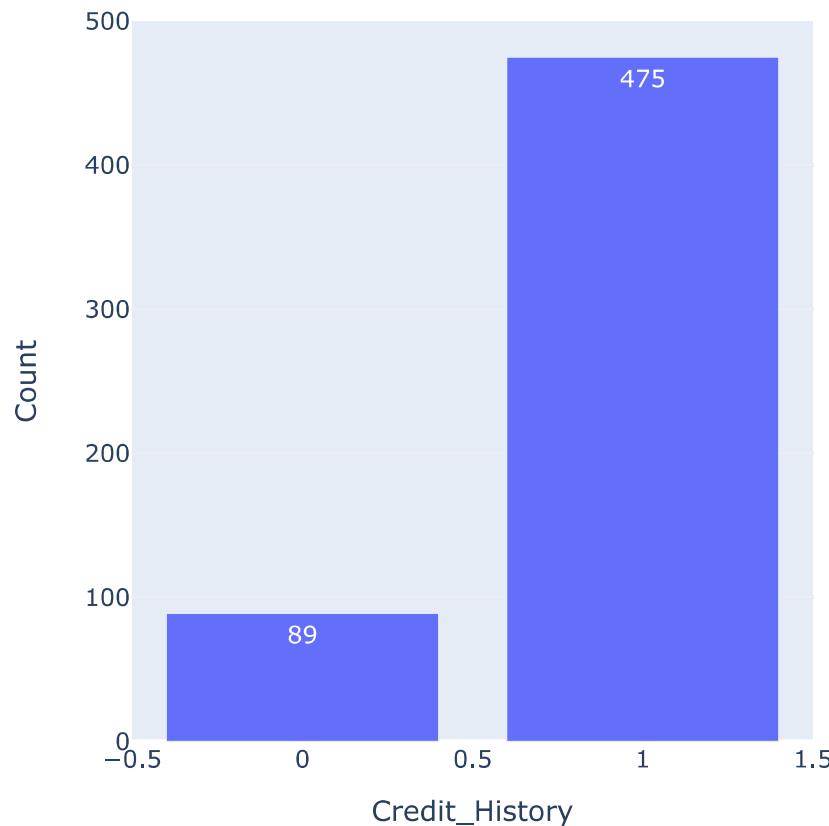
```
In [18]: fig = px.bar(data_frame=loan_data, x=loan_data['Self_Employed'].value_counts().index, y=loan_data['Self_Employed'].value_counts().values)
fig.update_layout(title='Number of Self_Employed or Not', xaxis_title='Employed', yaxis_title='Count')
fig.show()
```

Number of Self_Employed or Not



```
In [19]: fig = px.bar(data_frame=loan_data, x=loan_data['Credit_History'].value_counts().index, y=loan_data['Credit_History'].value_counts().values)
fig.update_layout(title='Number of Credit_History', xaxis_title='Credit_History', yaxis_title='Count')
fig.show()
```

Number of Credit_History

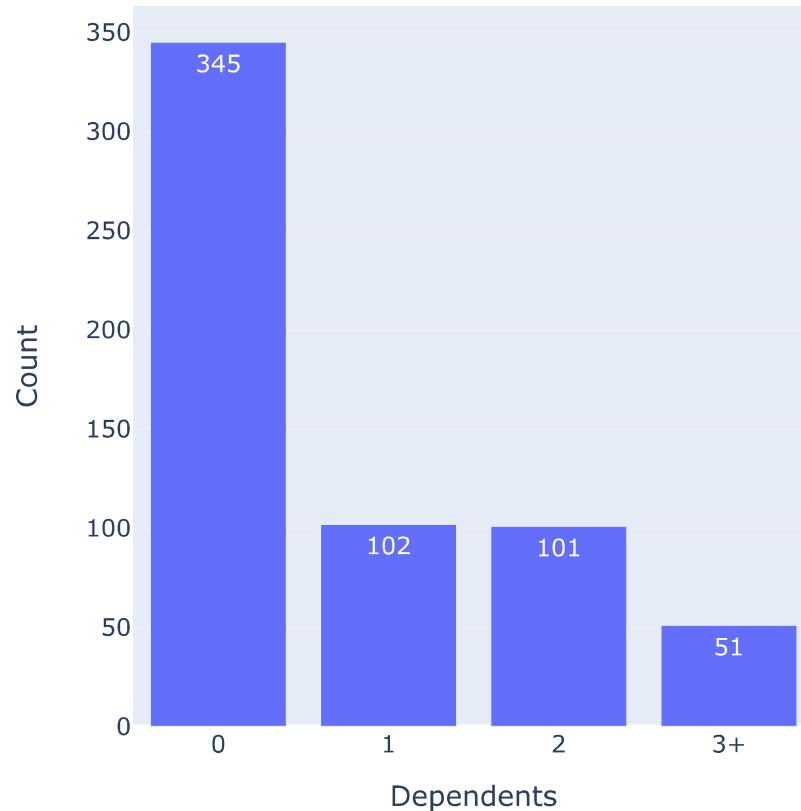


```
In [20]: loan_data.dtypes
```

```
Out[20]: 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
```

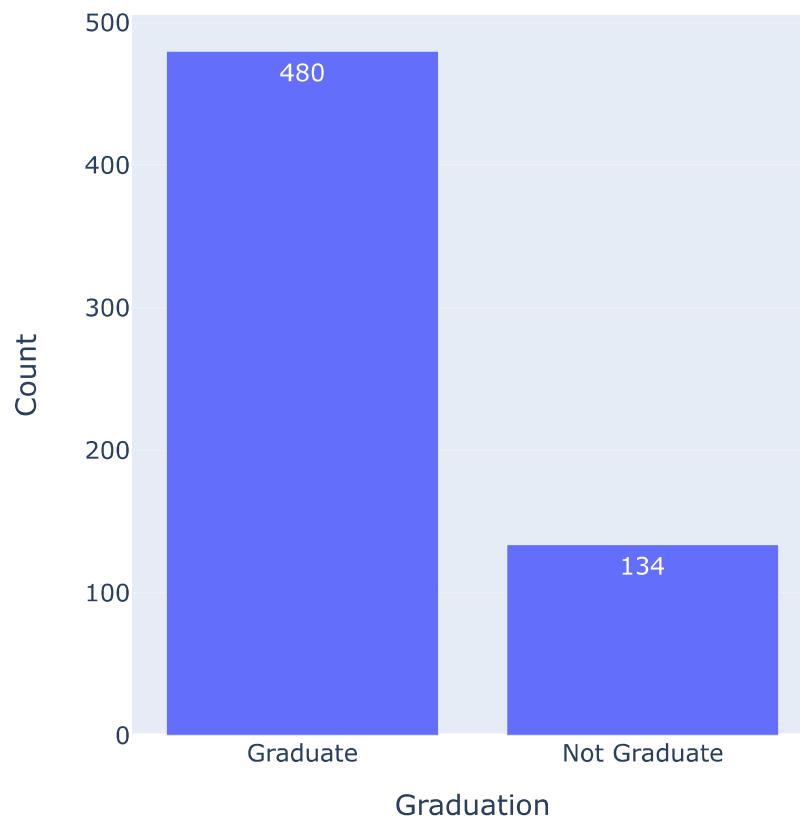
```
In [21]: fig = px.bar(data_frame=loan_data, x=loan_data['Dependents'].value_counts().index
fig.update_layout(title='Number of Dependents',xaxis_title='Dependents',yaxis_tit
fig.show()
```

Number of Dependents



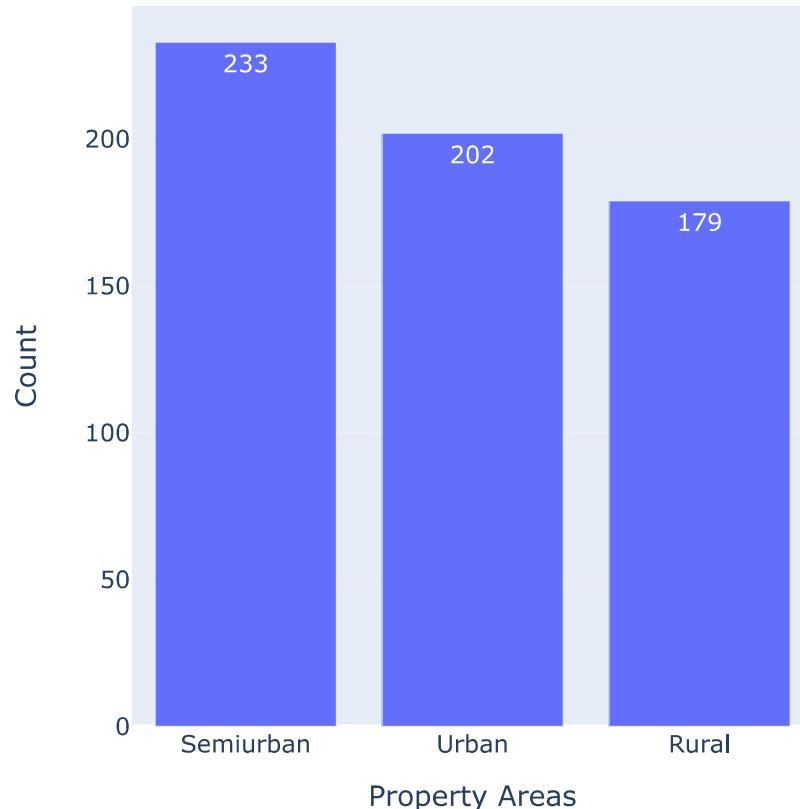
```
In [22]: fig = px.bar(data_frame=loan_data, x=loan_data['Education'].value_counts().index, fig.update_layout(title='Number of Graduate and Not Graduate',xaxis_title='Gradua fig.show()
```

Number of Graduate and Not Graduate

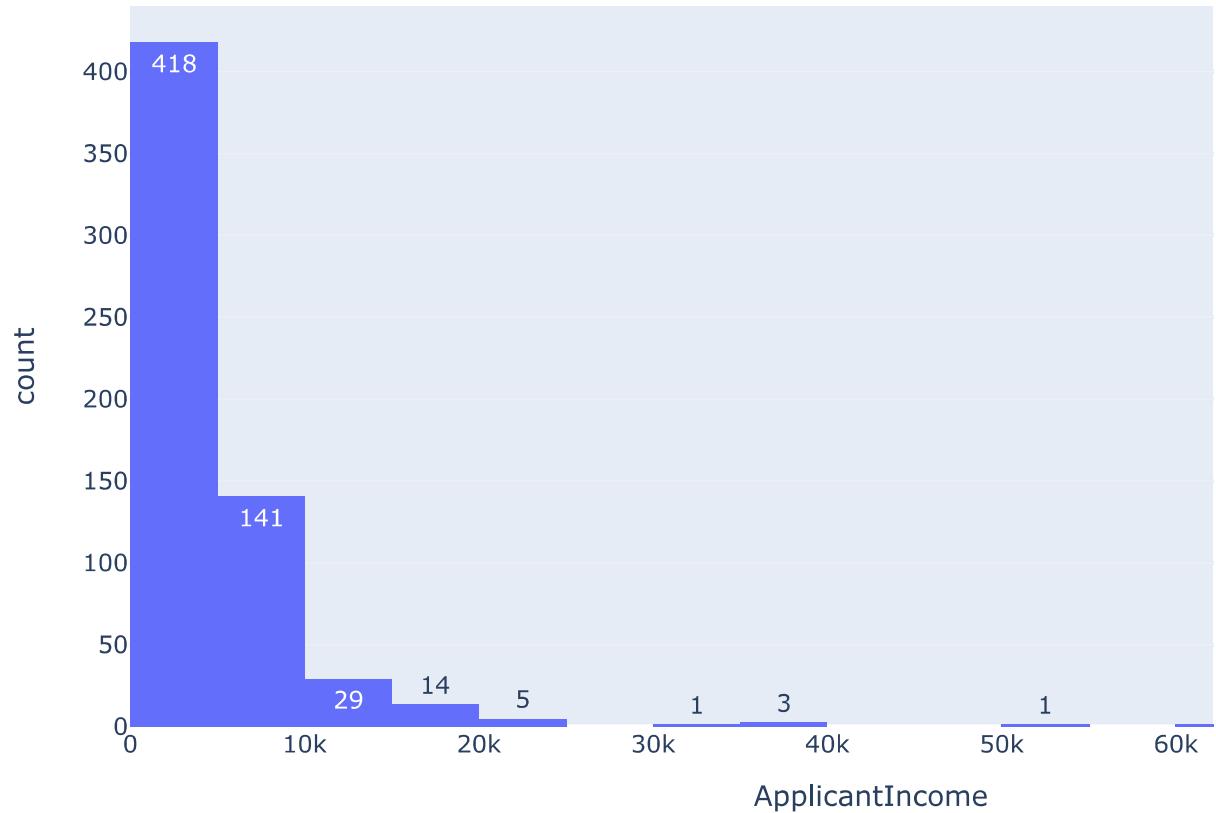


```
In [23]: fig = px.bar(data_frame=loan_data, x=loan_data['Property_Area'].value_counts().index, y=loan_data['Property_Area'].value_counts().values)
fig.update_layout(title='Number of Property Areas', xaxis_title='Property Areas', yaxis_title='Count')
fig.show()
```

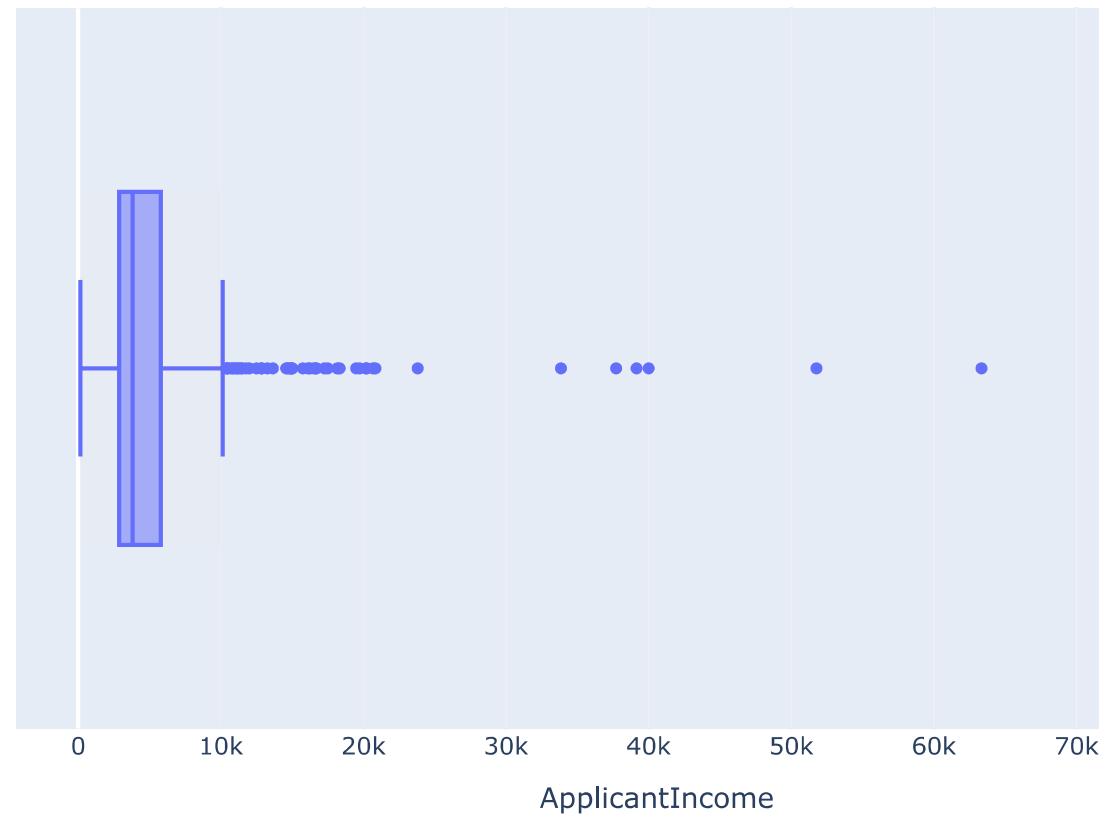
Number of Property Areas



```
In [24]: fig=px.histogram(data_frame=loan_data,x='ApplicantIncome',text_auto=True,nbins=20)
fig.update_layout(width=900,height=500)
fig.show()
```



```
In [25]: fig=px.box(data_frame=loan_data,x='ApplicantIncome')
fig.update_layout(width=800,height=500)
fig.show()
```

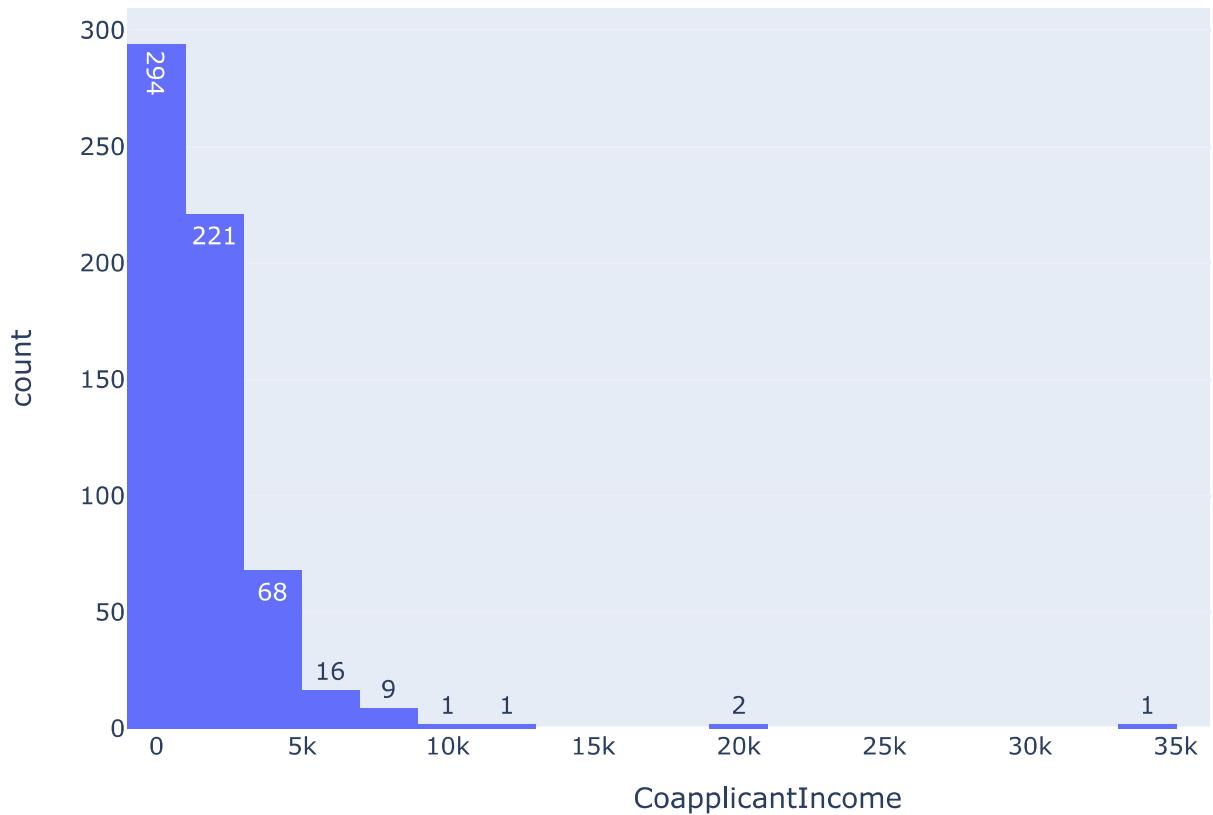


```
In [26]: fig=px.box(data_frame=loan_data,x='ApplicantIncome',y='Education',orientation='h'  
fig.update_layout(title='Applicant Income',width=800,height=500)  
fig.show()
```

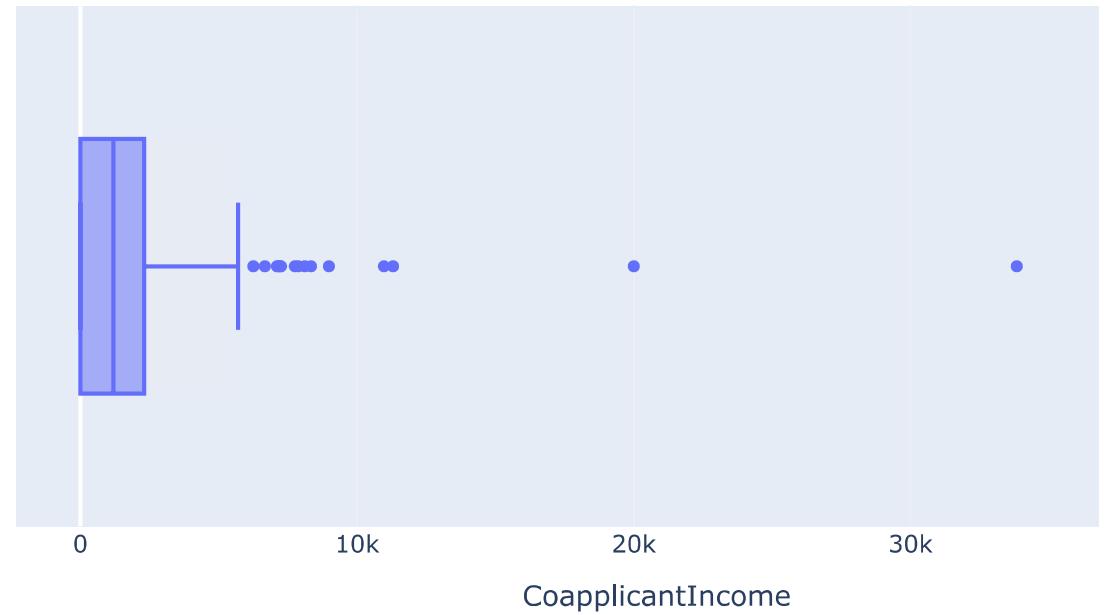


```
In [27]: fig=px.histogram(data_frame=loan_data,x='CoapplicantIncome',text_auto=True,nbins=10)
fig.update_layout(title='Coapplicant Income',width=800,height=500)
fig.show()
```

Coapplicant Income

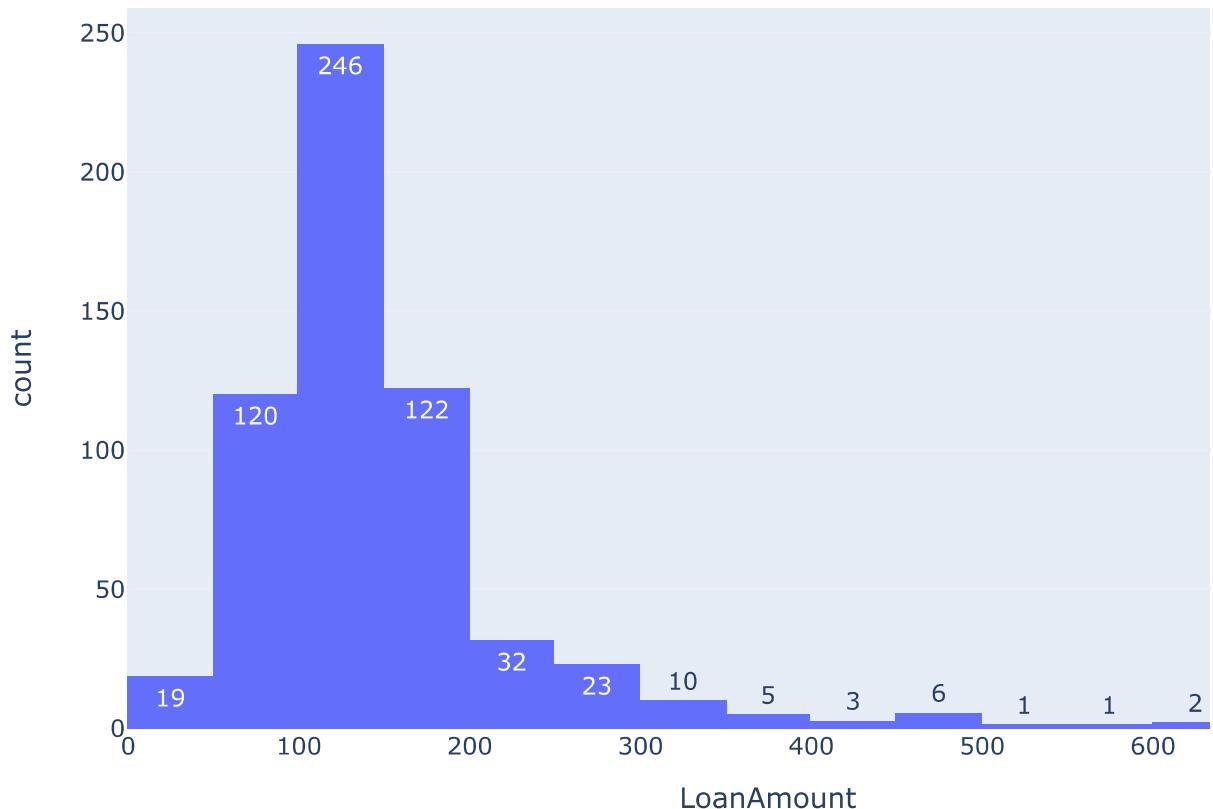


```
In [28]: fig=px.box(data_frame=loan_data,x='CoapplicantIncome')
fig.update_layout(width=800,height=400)
fig.show()
```



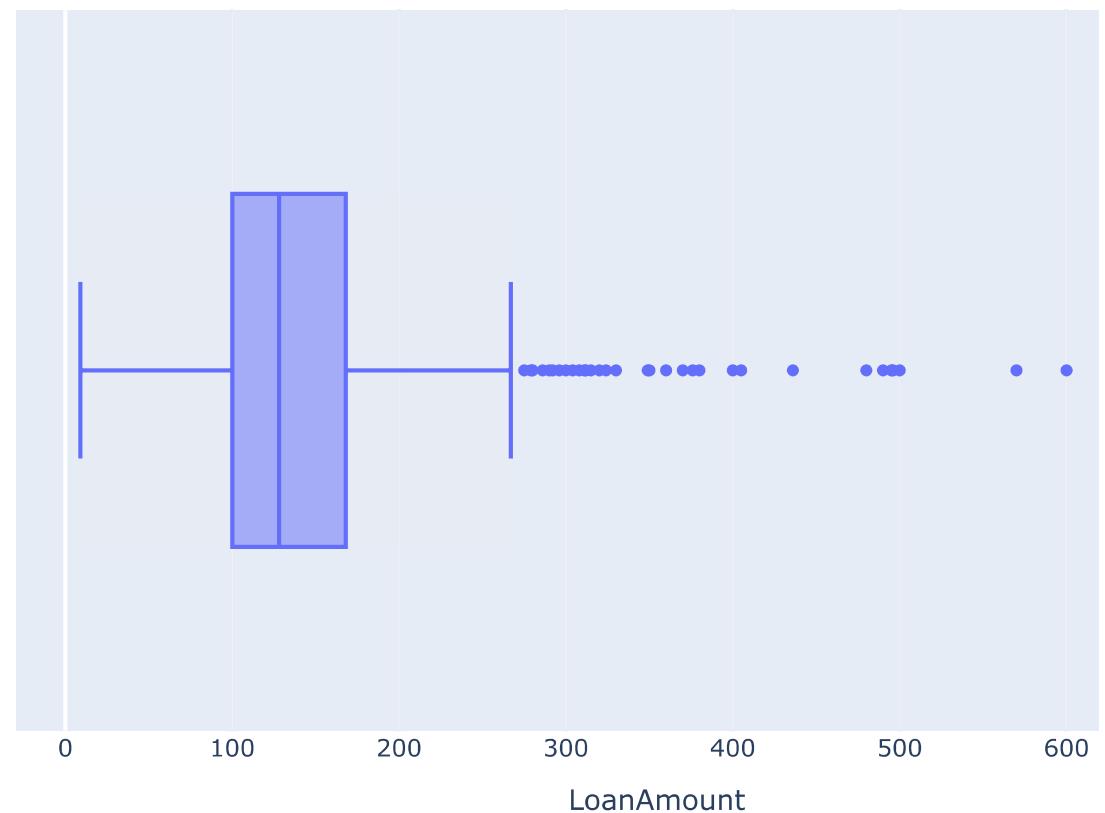
```
In [29]: fig=px.histogram(data_frame=loan_data,x='LoanAmount',text_auto=True,nbins=20)
fig.update_layout(title='Loan Amount',width=800,height=500)
fig.show()
```

Loan Amount

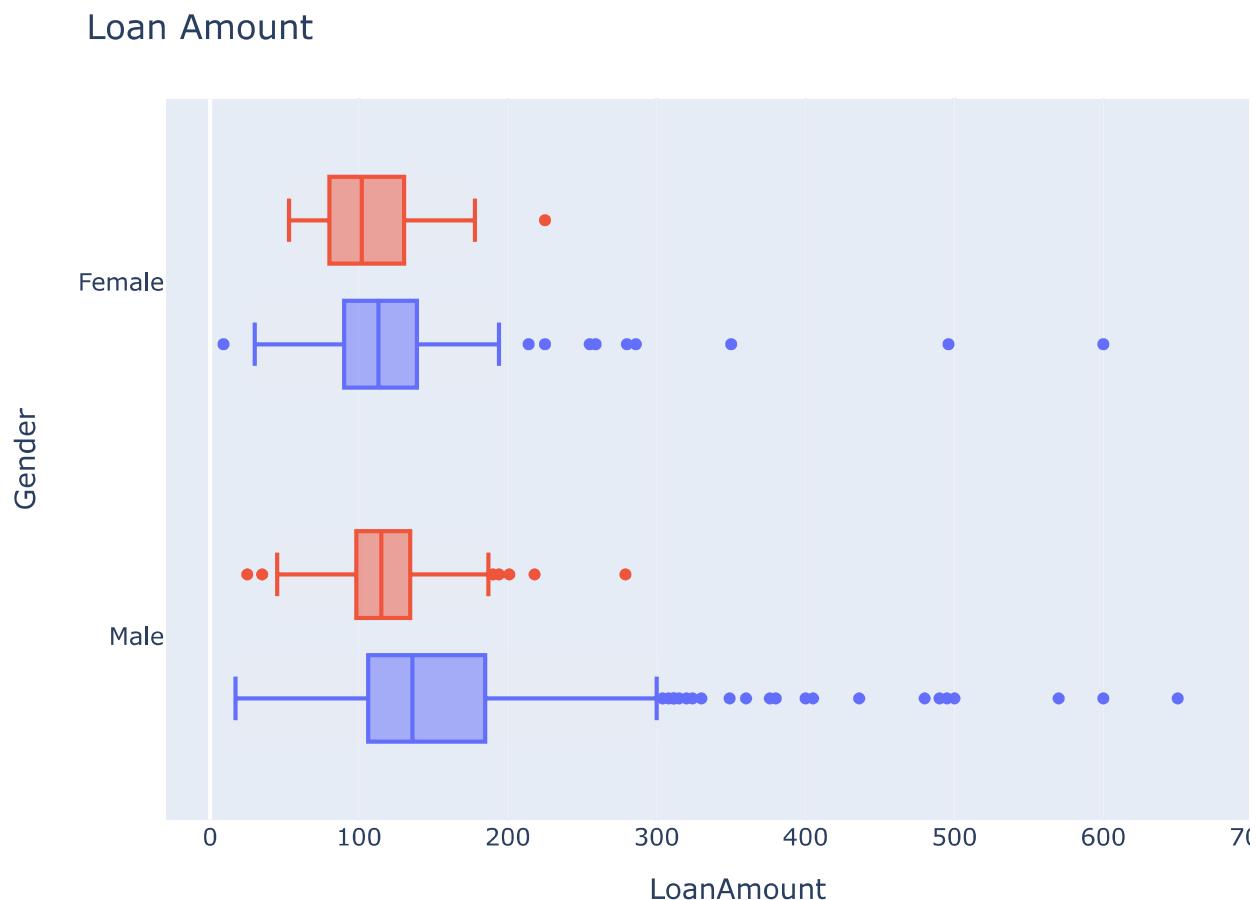


```
In [30]: fig=px.box(data_frame=loan_data,x='LoanAmount',orientation='h', )
fig.update_layout(title='Loan Amount',width=800,height=500)
fig.show()
```

Loan Amount



```
In [31]: fig=px.box(data_frame=loan_data,x='LoanAmount',y='Gender',orientation='h', color=fig.update_layout(title='Loan Amount',width=800,height=500)fig.show()
```



```
In [32]: Gender_vise_Loan_Status= loan_data.groupby(['Gender', 'Loan_Status']).size().reset_index()Gender_vise_Loan_Status
```

Out[32]:

	Gender	Loan_Status	Count
0	Female	N	37
1	Female	Y	75
2	Male	N	150
3	Male	Y	339

In [33]: `Married_vise_Loan_Status= loan_data.groupby(['Married', 'Loan_Status']).size().reset_index()`

Out[33]:

	Married	Loan_Status	Count
0	No	N	79
1	No	Y	134
2	Yes	N	113
3	Yes	Y	285

In [34]: `loan_data.dtypes`

Out[34]:

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

In [35]: `Dependents_vise_Loan_Status= loan_data.groupby(['Dependents', 'Loan_Status']).size().reset_index()`

Out[35]:

	Dependents	Loan_Status	Count
0	0	N	107
1	0	Y	238
2	1	N	36
3	1	Y	66
4	2	N	25
5	2	Y	76
6	3+	N	18
7	3+	Y	33

In [36]: `Education_vise_Loan_Status= loan_data.groupby(['Education', 'Loan_Status']).size()`
`Education_vise_Loan_Status`

Out[36]:

	Education	Loan_Status	Count
0	Graduate	N	140
1	Graduate	Y	340
2	Not Graduate	N	52
3	Not Graduate	Y	82

In [37]: `Self_Employed_vise_Loan_Status= loan_data.groupby(['Self_Employed', 'Loan_Status'])`
`Self_Employed_vise_Loan_Status`

Out[37]:

	Self_Employed	Loan_Status	Count
0	No	N	157
1	No	Y	343
2	Yes	N	26
3	Yes	Y	56

In [38]: `Credit_History_vise_Loan_Status= loan_data.groupby(['Credit_History', 'Loan_Status'])`
`Credit_History_vise_Loan_Status`

Out[38]:

	Credit_History	Loan_Status	Count
0	0.0	N	82
1	0.0	Y	7
2	1.0	N	97
3	1.0	Y	378

In [39]: `Property_Area_vise_Loan_Status= loan_data.groupby(['Property_Area', 'Loan_Status'])`
`Property_Area_vise_Loan_Status`

Out[39]:

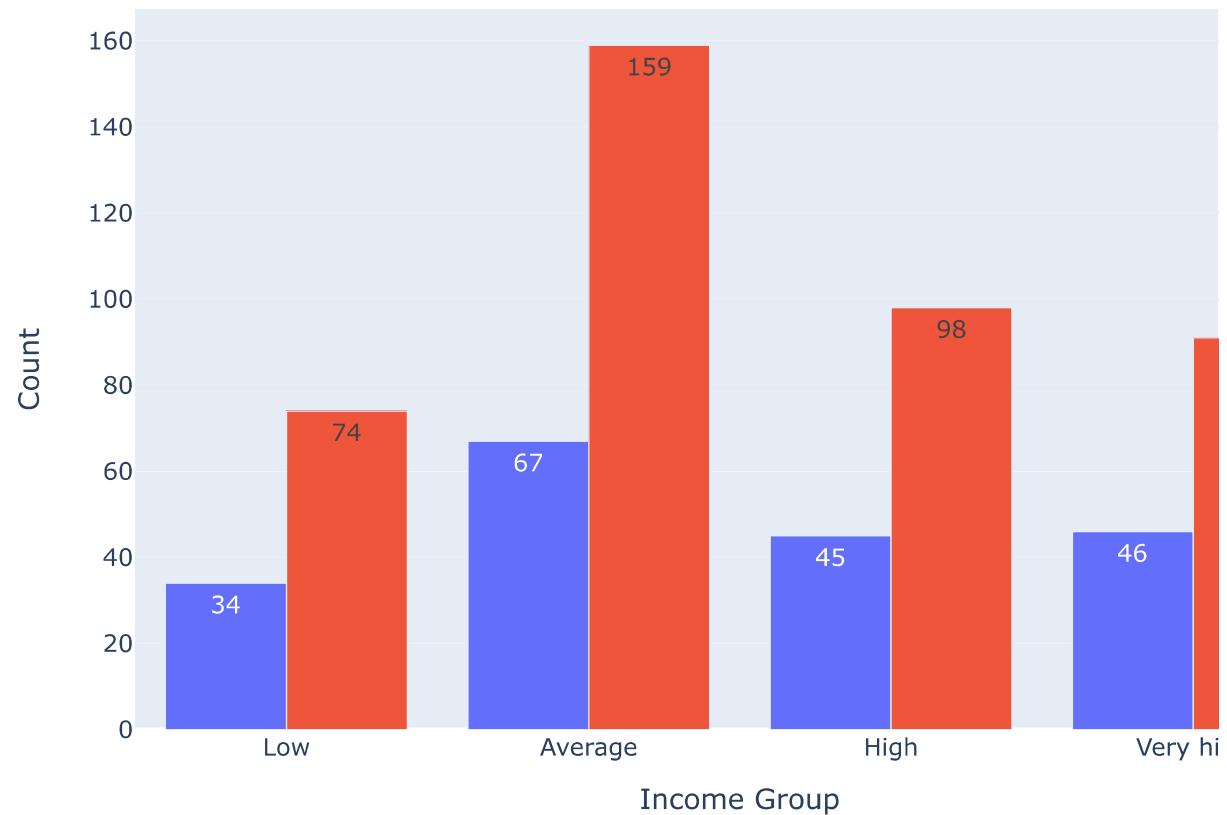
	Property_Area	Loan_Status	Count
0	Rural	N	69
1	Rural	Y	110
2	Semiurban	N	54
3	Semiurban	Y	179
4	Urban	N	69
5	Urban	Y	133

```
In [40]: bins=[0,2500,4000,6000,81000]
group=['Low','Average','High','Very high']
loan_data['Income_Group']=pd.cut(loan_data['ApplicantIncome'],bins=bins,labels=group)
Income_Group_vise_Loan_Status= loan_data.groupby(['Income_Group', 'Loan_Status'])
Income_Group_vise_Loan_Status
```

Out[40]:

	Income_Group	Loan_Status	Count
0	Low	N	34
1	Low	Y	74
2	Average	N	67
3	Average	Y	159
4	High	N	45
5	High	Y	98
6	Very high	N	46
7	Very high	Y	91

```
In [41]: fig = px.bar(Income_Group_vise_Loan_Status,x='Income_Group',y='Count',color='Loan_Status',color_discrete_map={0:'blue',1:'red'},width=800,height=600)
fig.update_layout(xaxis_title='Income Group',yaxis_title='Count',width=800,height=600)
fig.show()
```

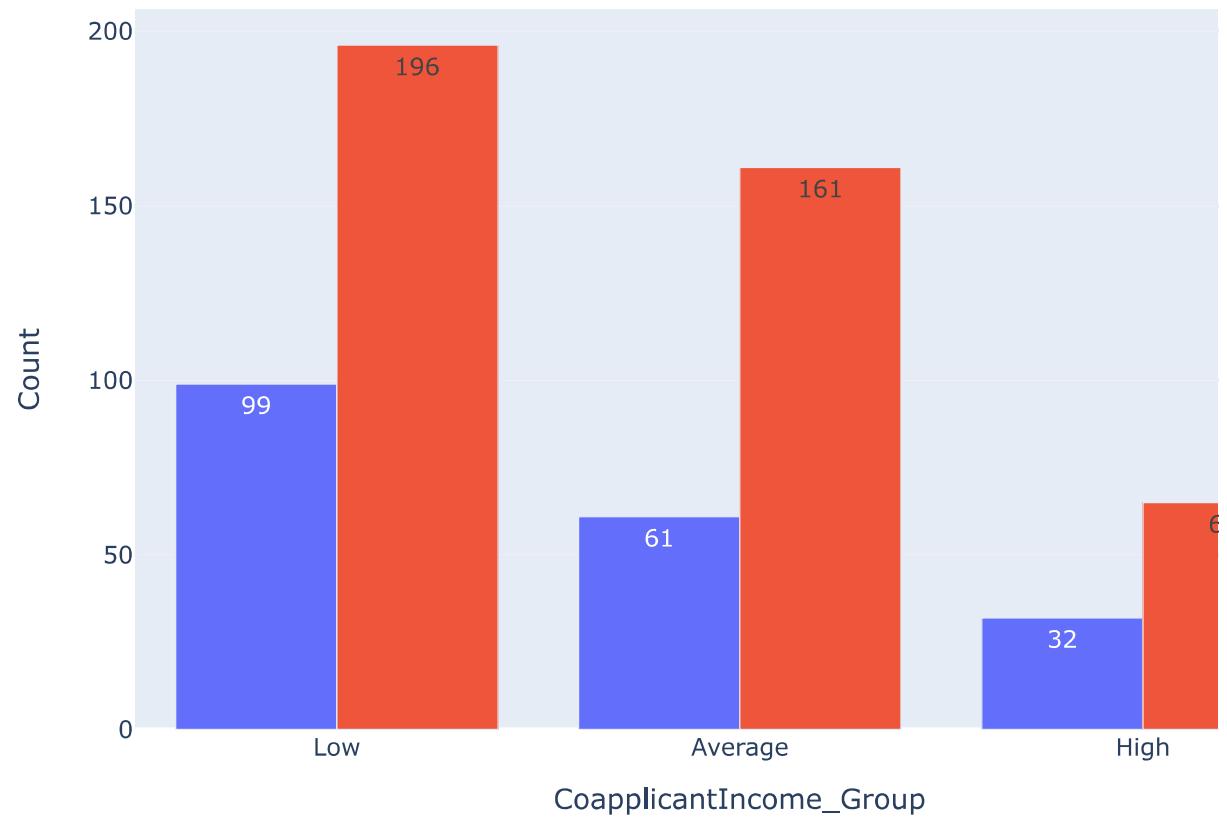


```
In [42]: bins=[0,1000,3000,42000]
group=['Low','Average','High']
loan_data['CoapplicantIncome_Group']=pd.cut(loan_data['CoapplicantIncome'],bins=bins)
CoapplicantIncome_Group_vise_Loan_Status= loan_data.groupby(['CoapplicantIncome_Group'])
CoapplicantIncome_Group_vise_Loan_Status
```

Out[42]:

	CoapplicantIncome_Group	Loan_Status	Count
0	Low	N	99
1	Low	Y	196
2	Average	N	61
3	Average	Y	161
4	High	N	32
5	High	Y	65

```
In [43]: fig = px.bar(CoapplicantIncome_Group_vise_Loan_Status,x='CoapplicantIncome_Group'  
fig.update_layout(xaxis_title='CoapplicantIncome_Group',yaxis_title='Count',width=1000)  
fig.show()
```

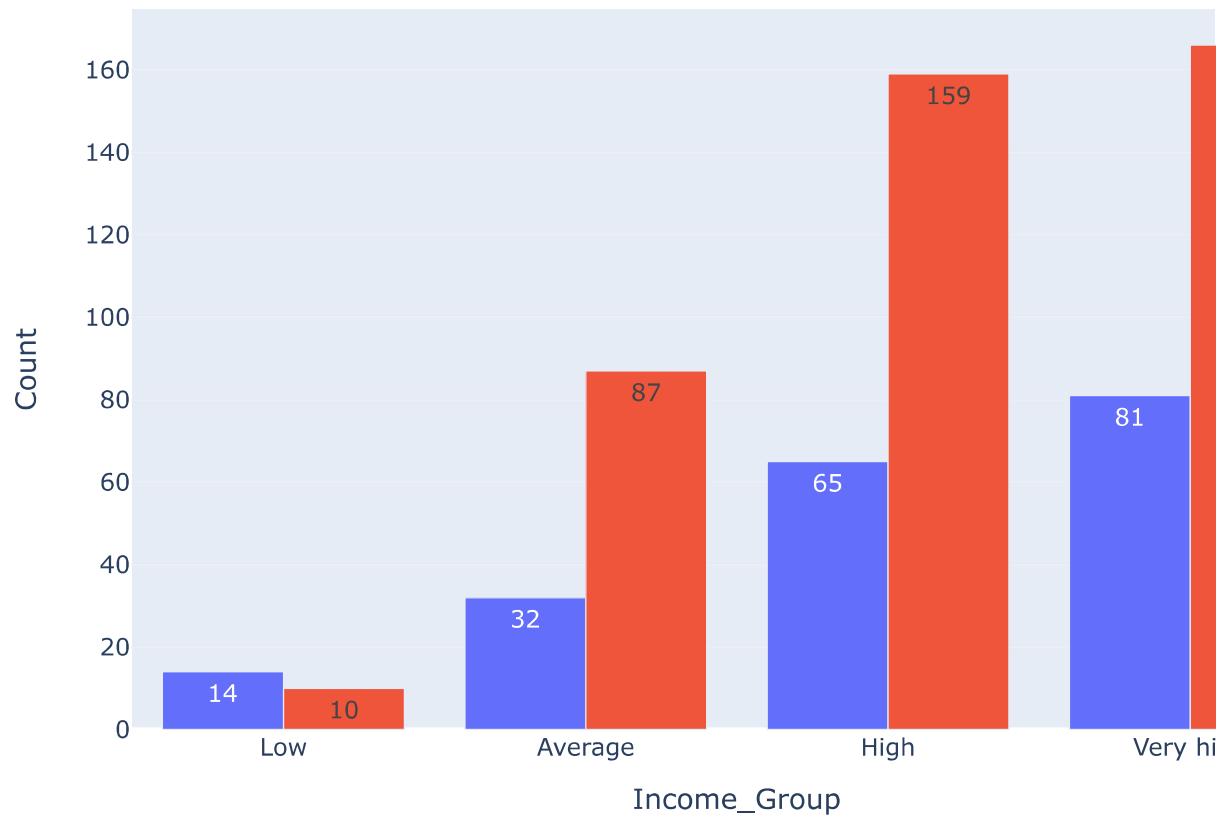


```
In [44]: loan_data['Total_Income']=loan_data['ApplicantIncome']+loan_data['CoapplicantIncome']
bins=[0, 2500, 4000, 6000, 81000]
group=['Low', 'Average', 'High', 'Very high']
loan_data['Total_Income_Group']=pd.cut(loan_data['Total_Income'], bins=bins, labels=group)
Total_Income_Group_vise_Loan_Status= loan_data.groupby(['Total_Income_Group', 'Loan_Status']).size().reset_index()
Total_Income_Group_vise_Loan_Status
```

Out[44]:

	Total_Income_Group	Loan_Status	Count
0	Low	N	14
1	Low	Y	10
2	Average	N	32
3	Average	Y	87
4	High	N	65
5	High	Y	159
6	Very high	N	81
7	Very high	Y	166

```
In [45]: fig = px.bar(Total_Income_Group_vise_Loan_Status,x='Total_Income_Group',y='Count'  
fig.update_layout(xaxis_title='Income_Group',yaxis_title='Count',width=800,height=600)  
fig.show()
```

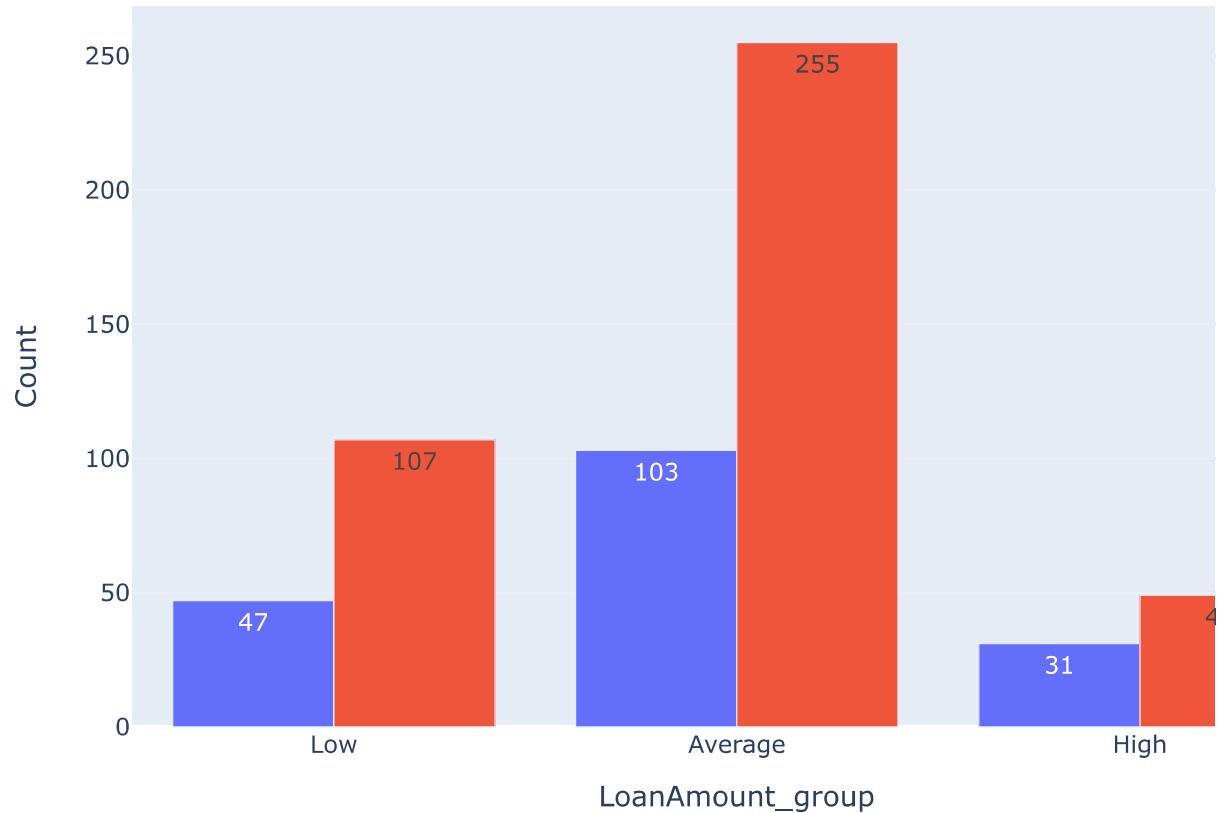


```
In [46]: bins=[0,100,200,700]
group=['Low','Average','High']
loan_data['LoanAmount_group']=pd.cut(loan_data['LoanAmount'],bins,labels=group)
LoanAmount_group_vise_Loan_Status= loan_data.groupby(['LoanAmount_group', 'Loan_Status'])
LoanAmount_group_vise_Loan_Status
```

Out[46]:

	LoanAmount_group	Loan_Status	Count
0	Low	N	47
1	Low	Y	107
2	Average	N	103
3	Average	Y	255
4	High	N	31
5	High	Y	49

```
In [47]: fig = px.bar(LoanAmount_group_vise_Loan_Status,x='LoanAmount_group',y='Count',color='Loan_Status')
fig.update_layout(xaxis_title='LoanAmount_group',yaxis_title='Count',width=800,he
fig.show()
```



```
In [48]: loan_data=loan_data.drop(['Income_Group', 'CoapplicantIncome_Group', 'Total_Inco
```

```
In [49]: loan_data['Dependents'].replace('3+', 3,inplace=True)
loan_data['Loan_Status'].replace('N', 0,inplace=True)
loan_data['Loan_Status'].replace('Y', 1,inplace=True)
```

```
In [50]: loan_data.head()
```

Out[50]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplican
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	

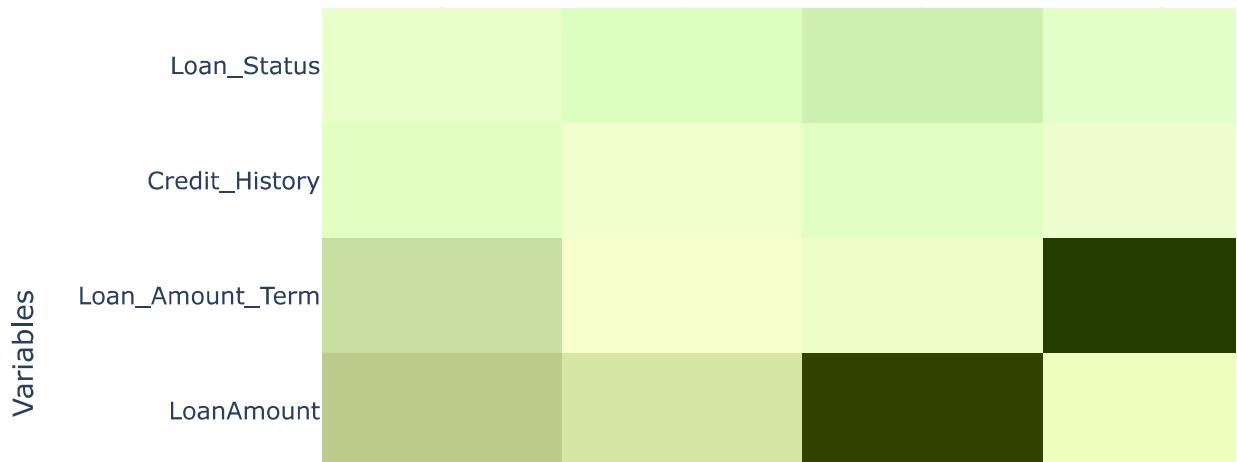
```
In [51]: Correlation=loan_data.corr(method='pearson')
print(Correlation)
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	\
ApplicantIncome	1.000000	-0.116605	0.570909	
CoapplicantIncome	-0.116605	1.000000	0.188619	
LoanAmount	0.570909	0.188619	1.000000	
Loan_Amount_Term	-0.045306	-0.059878	0.039447	
Credit_History	-0.014715	-0.002056	-0.008433	
Loan_Status	-0.004710	-0.059187	-0.037318	
	Loan_Amount_Term	Credit_History	Loan_Status	
ApplicantIncome	-0.045306	-0.014715	-0.004710	
CoapplicantIncome	-0.059878	-0.002056	-0.059187	
LoanAmount	0.039447	-0.008433	-0.037318	
Loan_Amount_Term	1.000000	0.001470	-0.021268	
Credit_History	0.001470	1.000000	0.561678	
Loan_Status	-0.021268	0.561678	1.000000	

C:\Users\Suyash\AppData\Local\Temp\ipykernel_9496\4188836588.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

```
In [52]: fig=go.Figure(go.Heatmap(x=Correlation.columns,y=Correlation.columns,z=Correlation))
fig.update_layout(title='Correlation Heatmap',xaxis_title='Variables',yaxis_title='Variables')
fig.show()
```



```
In [53]: loan_data.isnull().sum()
```

```
Out[53]:
```

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 [54]: loan_data.dtypes
```

```
Out[54]:
```

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	int64
dtype:	object

```
In [55]: loan_data['Gender'].fillna(method='ffill', inplace=True)
```

```
In [56]: loan_data['Dependents'].fillna(loan_data['Dependents'].mode()[0], inplace=True)
```

```
In [57]: loan_data['Married'].fillna(loan_data['Married'].mode()[0], inplace=True)
```

```
In [58]: loan_data['Self_Employed'].fillna(method='ffill', inplace=True)
```

```
In [59]: loan_data['Credit_History'].fillna(method='bfill', inplace=True)
```

```
In [60]: loan_data['LoanAmount'].fillna(loan_data['LoanAmount'].median(), inplace=True)
```

```
In [61]: loan_data['Loan_Amount_Term'].fillna(loan_data['Loan_Amount_Term'].mode()[0], in
```

```
In [62]: loan_data.isnull().sum()
```

```
Out[62]:
```

Loan_ID	0
Gender	0
Married	0
Dependents	0
Education	0
Self_Employed	0
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	0
Loan_Amount_Term	0
Credit_History	0
Property_Area	0
Loan_Status	0
dtype:	int64

In [63]: `loan_data.describe()`

Out[63]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan...
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.
mean	5403.459283	1621.245798	145.752443	342.410423	0.84202	0.
std	6109.041673	2926.248369	84.107233	64.428629	0.36502	0.
min	150.000000	0.000000	9.000000	12.000000	0.00000	0.
25%	2877.500000	0.000000	100.250000	360.000000	1.00000	0.
50%	3812.500000	1188.500000	128.000000	360.000000	1.00000	1.
75%	5795.000000	2297.250000	164.750000	360.000000	1.00000	1.
max	81000.000000	41667.000000	700.000000	480.000000	1.00000	1.

◀ ▶

In [64]: `loan_data`

Out[64]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic...
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	
...
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

◀ ▶

In [65]: `loan_data['NormLoanAmount'] = np.log(loan_data['LoanAmount'])`

In [66]: loan_data

Out[66]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
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	
...
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 14 columns



In [67]: loan_data=loan_data.drop('Loan_ID',axis=1)

In [68]: X = loan_data.drop(labels='Loan_Status',axis=1)
Y = loan_data['Loan_Status']

In [69]: X

Out[69]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.0
1	Male	Yes	1	Graduate	No	4583	1508.0
2	Male	Yes	0	Graduate	Yes	3000	0.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0
4	Male	No	0	Graduate	No	6000	0.0
...
609	Female	No	0	Graduate	No	2900	0.0
610	Male	Yes	3	Graduate	No	4106	0.0
611	Male	Yes	1	Graduate	No	8072	240.0
612	Male	Yes	2	Graduate	No	7583	0.0
613	Female	No	0	Graduate	Yes	4583	0.0

614 rows × 12 columns



In [70]: Y

Out[70]:

```

0      1
1      0
2      1
3      1
4      1
 ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 614, dtype: int64

```

In [71]: columns = X.columns

```

cat_col= [col for col in X.columns if X[col].dtypes=='O']
cat_col

```

Out[71]:

```

['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area']

```

```
In [72]: dummy = pd.get_dummies(X[cat_col])
dummy.shape
```

Out[72]: (614, 15)

```
In [73]: dummy
```

Out[73]:

	Gender_Female	Gender_Male	Married_No	Married_Yes	Dependents_3	Dependents_0	Dependents_2
0	0	1	1	0	0	0	1
1	0	1	0	1	0	0	0
2	0	1	0	1	0	0	1
3	0	1	0	1	0	0	1
4	0	1	1	0	0	0	1
...
609	1	0	1	0	0	0	1
610	0	1	0	1	1	1	0
611	0	1	0	1	0	0	0
612	0	1	0	1	0	0	0
613	1	0	1	0	0	0	1

614 rows × 15 columns

◀ ▶

```
In [74]: final = pd.concat([X,dummy],axis=1)
final.shape
```

Out[74]: (614, 27)

```
In [75]: final.drop(cat_col,inplace=True,axis=1)
```

```
In [76]: final.shape
```

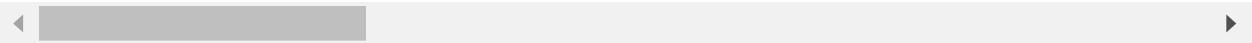
Out[76]: (614, 21)

In [77]: final

Out[77]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	NormLc
0	5849	0.0	128.0	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	
4	6000	0.0	141.0	360.0	1.0	
...
609	2900	0.0	71.0	360.0	1.0	
610	4106	0.0	40.0	180.0	1.0	
611	8072	240.0	253.0	360.0	1.0	
612	7583	0.0	187.0	360.0	1.0	
613	4583	0.0	133.0	360.0	0.0	

614 rows × 21 columns



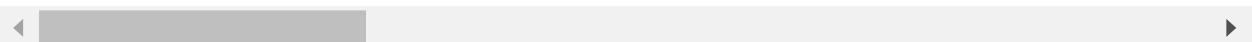
In [78]: X=final

X

Out[78]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	NormLc
0	5849	0.0	128.0	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	
4	6000	0.0	141.0	360.0	1.0	
...
609	2900	0.0	71.0	360.0	1.0	
610	4106	0.0	40.0	180.0	1.0	
611	8072	240.0	253.0	360.0	1.0	
612	7583	0.0	187.0	360.0	1.0	
613	4583	0.0	133.0	360.0	0.0	

614 rows × 21 columns



In [79]: Y

```
Out[79]: 0      1
         1      0
         2      1
         3      1
         4      1
         ..
        609     1
        610     1
        611     1
        612     1
        613     0
Name: Loan_Status, Length: 614, dtype: int64
```

In [80]: x_train,x_test,y_train,y_test = train_test_split(X,Y, test_size = 0.2)

In []:

LogisticRegression

In [81]: clf = LogisticRegression()

In [82]: clf.fit(x_train,y_train)

```
Out[82]: LogisticRegression
          LogisticRegression()
```

In [83]: pred = clf.predict(x_test)

In [84]: accuracy_score(y_test,pred)

Out[84]: 0.7804878048780488

In [85]: f1_score(y_test,pred)

Out[85]: 0.8556149732620322

In [86]: precision_score(y_test,pred)

Out[86]: 0.7692307692307693

In [87]: recall_score(y_test,pred)

Out[87]: 0.963855421686747

```
In [88]: confusion_matrix(y_test,pred)
```

```
Out[88]: array([[16, 24],  
 [ 3, 80]], dtype=int64)
```

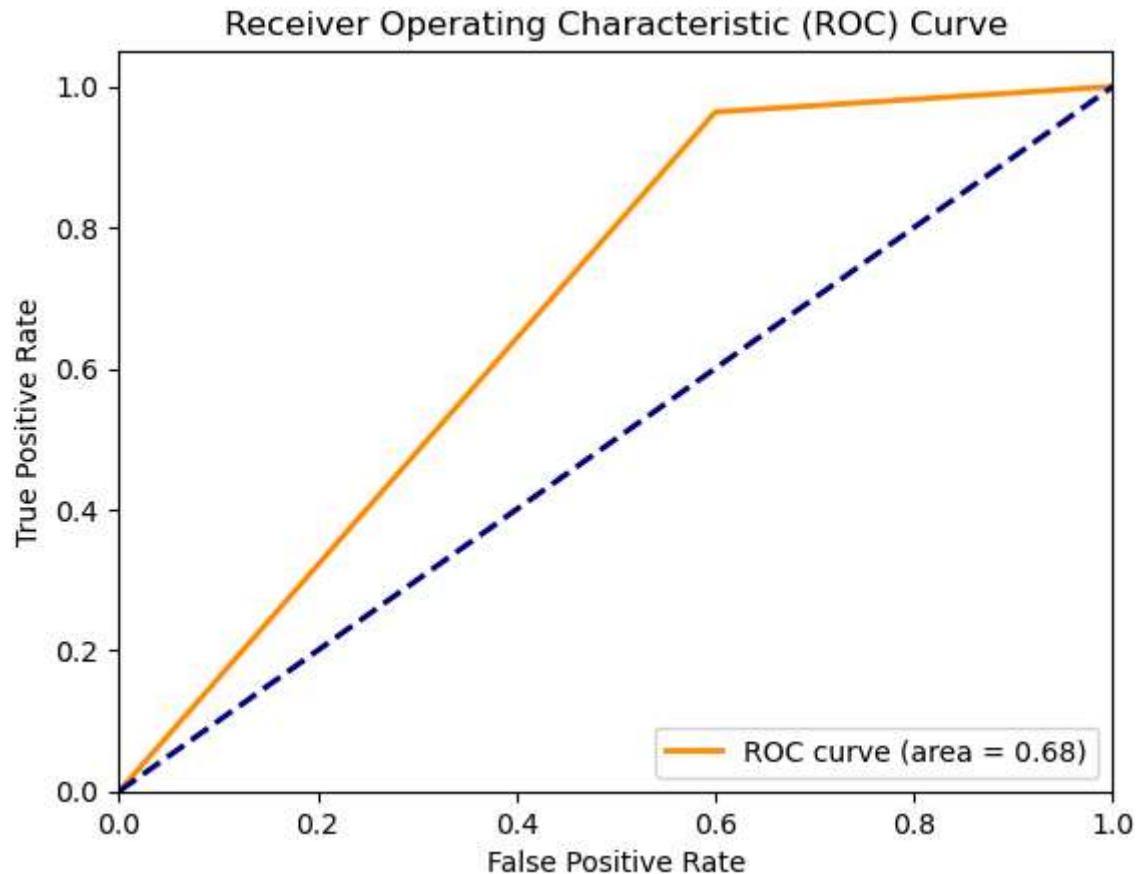
```
In [89]: from sklearn.metrics import roc_curve, auc  
import matplotlib.pyplot as plt
```

```
In [90]: fpr, tpr, thresholds = roc_curve(y_test,pred)
```

```
In [91]: roc_auc = auc(fpr, tpr)  
print("ROC AUC:", roc_auc)
```

```
ROC AUC: 0.6819277108433734
```

```
In [92]: plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
          lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



Decision_Tree

```
In [93]: dct=tree.DecisionTreeClassifier()
```

```
In [94]: dct.fit(x_train,y_train)
```

```
Out[94]:
```

DecisionTreeClassifier
DecisionTreeClassifier()

```
In [95]: pred = dct.predict(x_test)
```

In [96]: pred

```
In [97]: accuracy_score(y_test,pred)
```

Out[97]: 0.6585365853658537

```
In [98]: f1_score(y_test, pred)
```

Out[98]: 0.7439024390243902

```
In [99]: precision_score(y_test, pred)
```

Out[99]: 0.7530864197530864

```
In [100]: recall_score(y_test, pred)
```

Out[100]: 0.7349397590361446

```
In [101]: confusion_matrix(y_test,pred)
```

```
Out[101]: array([[20, 20],  
                  [22, 61]], dtype=int64)
```

```
In [102]: roc_auc_score(y_test,pred)
```

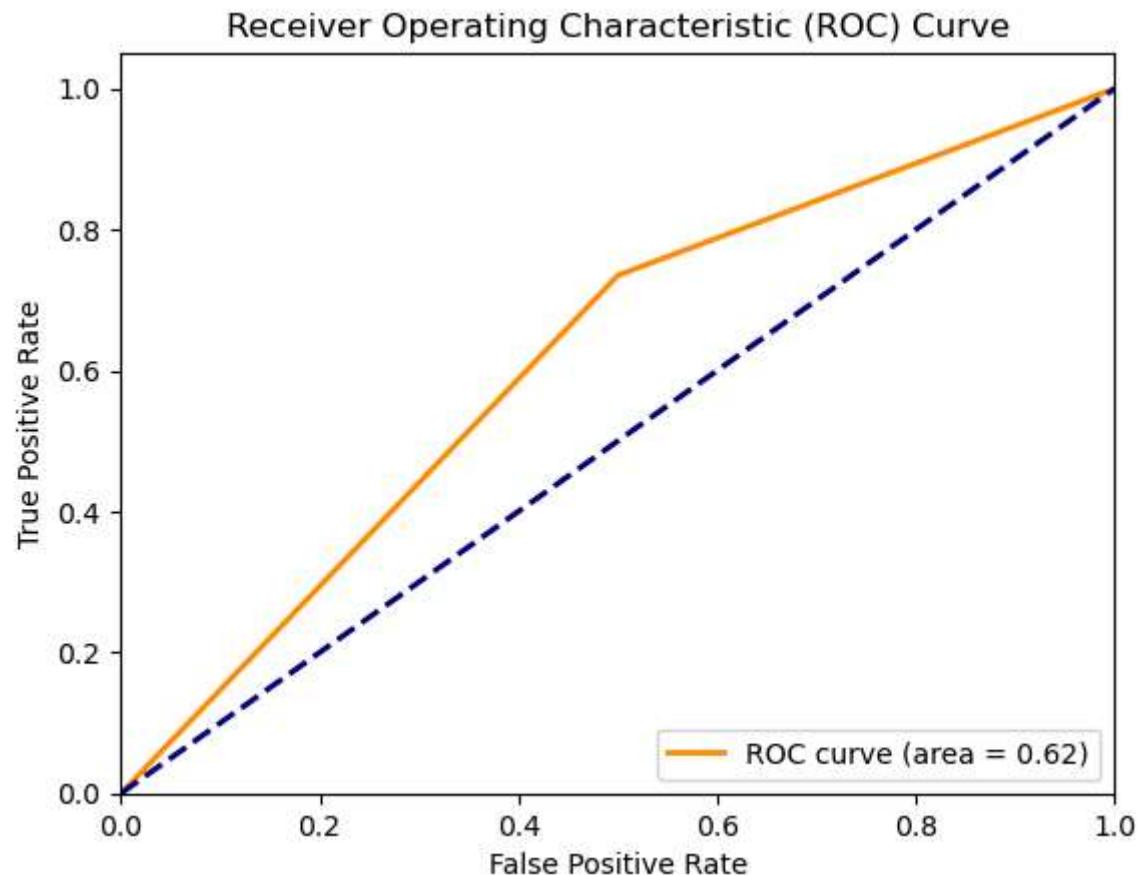
Out[102]: 0.6174698795180723

```
In [103]: import matplotlib.pyplot as plt
```

```
from sklearn.metrics import RocCurveDisplay
```

```
In [104]: fpr, tpr, thresholds = roc_curve(y_test,pred)
roc_auc = auc(fpr, tpr)
print("ROC AUC:", roc_auc)
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
          lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

ROC AUC: 0.6174698795180723



Random Forest

```
In [105]: rfc=RandomForestClassifier()
```

```
In [106]: rfc.fit(x_train,y_train)
```

```
Out[106]: RandomForestClassifier()
           |____ RandomForestClassifier()
```

```
In [107]: pred = rfc.predict(x_test)
```

```
In [108]: accuracy_score(y_test,pred)
```

```
Out[108]: 0.7642276422764228
```

```
In [109]: precision_score(y_test,pred)
```

```
Out[109]: 0.7755102040816326
```

```
In [110]: recall_score(y_test,pred)
```

```
Out[110]: 0.9156626506024096
```

```
In [111]: f1_score(y_test,pred)
```

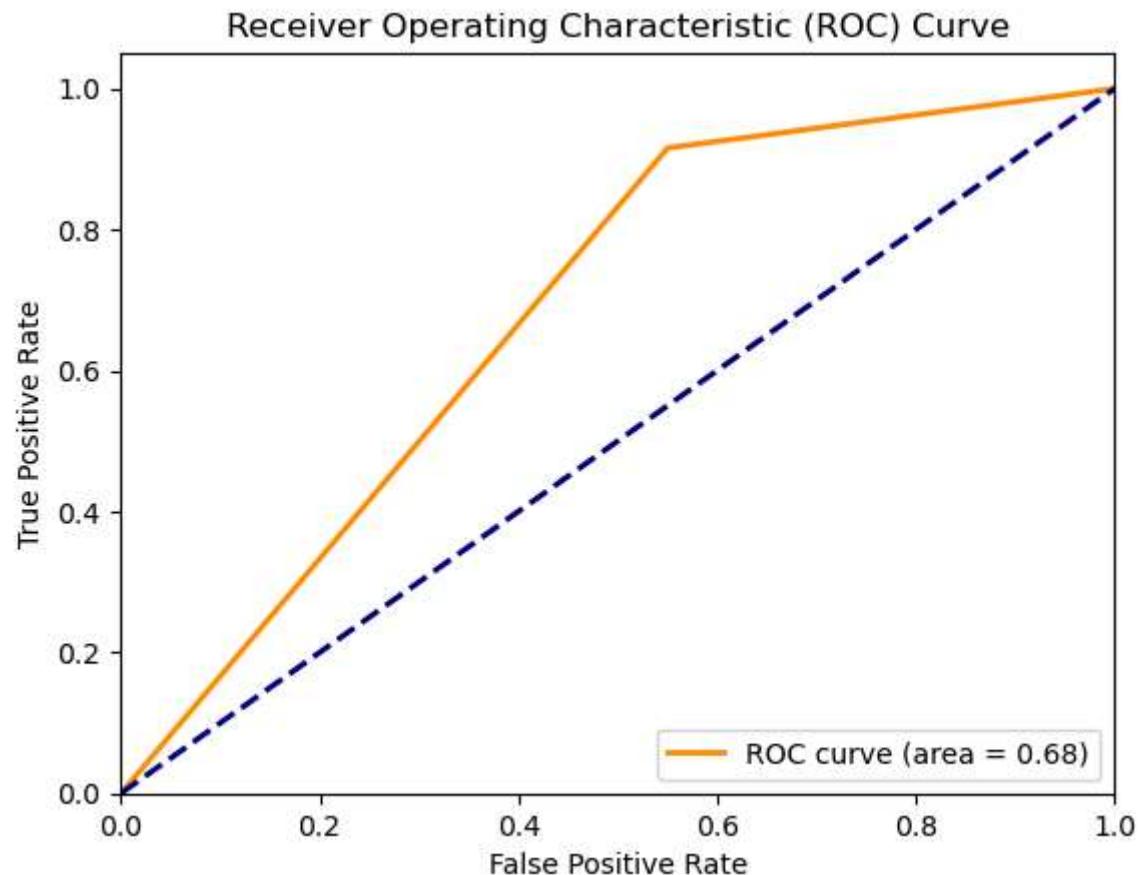
```
Out[111]: 0.839779005524862
```

```
In [112]: confusion_matrix(y_test,pred)
```

```
Out[112]: array([[18, 22],
                  [ 7, 76]], dtype=int64)
```

```
In [113]: fpr, tpr, thresholds = roc_curve(y_test,pred)
roc_auc = auc(fpr, tpr)
print("ROC AUC:", roc_auc)
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
          lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

ROC AUC: 0.6828313253012047



```
In [ ]:
```