Loan Approval Prediction

Importing Libraries

In [78]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

Importing Dataset

In [46]:
```
data = pd.read_csv(r'C:\Users\User\Downloads\LoanApprovalPrediction.csv')
```

In [47]:
```
data.head(10)
```

Out[47]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loa
0	LP001002	Male	No	0.0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0.0	Graduate	No	6000	0.0	
5	LP001011	Male	Yes	2.0	Graduate	Yes	5417	4196.0	
6	LP001013	Male	Yes	0.0	Not Graduate	No	2333	1516.0	
7	LP001014	Male	Yes	3.0	Graduate	No	3036	2504.0	
8	LP001018	Male	Yes	2.0	Graduate	No	4006	1526.0	
9	LP001020	Male	Yes	1.0	Graduate	No	12841	10968.0	

Data Preprocessing and Visualization

As Loan_ID is completely unique and not correlated with any of the other column

In [48]:
```
data.drop(['Loan_ID'],axis=1,inplace=True)
```

In [49]:
```
data
```

Out[49]:

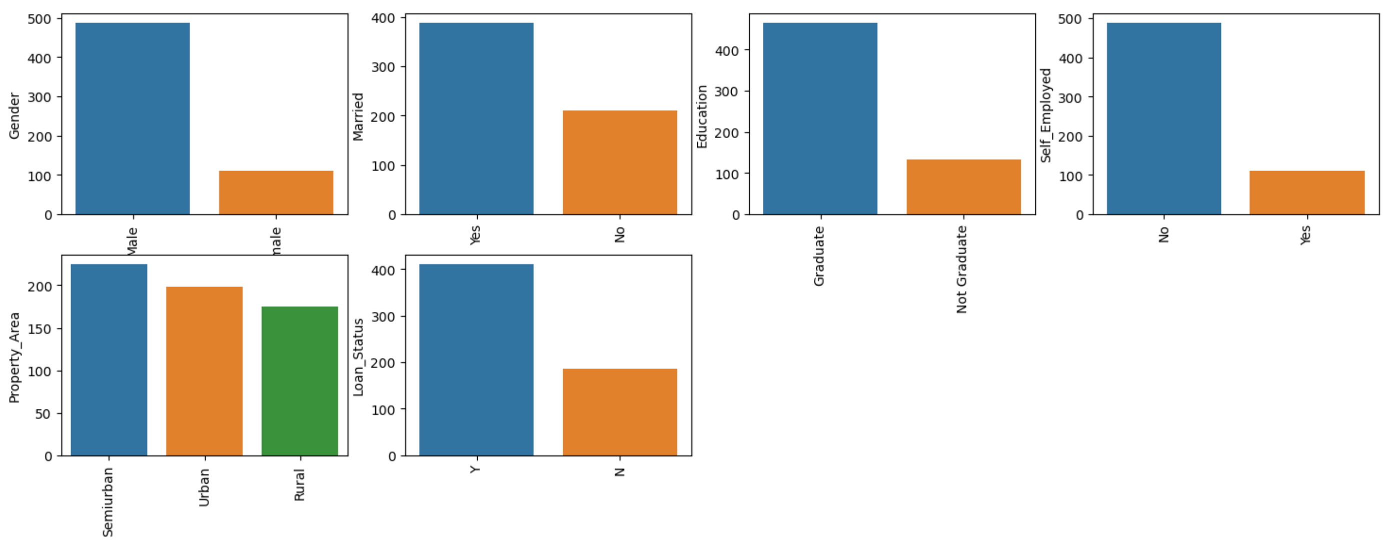
Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount

0	Male	No	0.0	Graduate	No	5849	0.0	NaN
1	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0
2	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0
3	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0
4	Male	No	0.0	Graduate	No	6000	0.0	141.0
...	...	...	...	...	...	...	...	...
593	Female	No	0.0	Graduate	No	2900	0.0	71.0
594	Male	Yes	3.0	Graduate	No	4106	0.0	40.0
595	Male	Yes	1.0	Graduate	No	8072	240.0	253.0
596	Male	Yes	2.0	Graduate	No	7583	0.0	187.0
597	Female	No	0.0	Graduate	Yes	4583	0.0	133.0

598 rows × 12 columns

In [50]:
```
obj = (data.dtypes == 'object')
object_cols = list(obj[obj].index)
plt.figure(figsize=(18,36))
index = 1

for col in object_cols:
  y = data[col].value_counts()
  plt.subplot(11,4,index)
  plt.xticks(rotation=90)
  sns.barplot(x=list(y.index), y=y)
  index +=1
```



In [51]:
```
label_encoder = preprocessing.LabelEncoder()
obj = (data.dtypes == 'object')
for col in list(obj[obj].index):
  data[col] = label_encoder.fit_transform(data[col])
```

check the object datatype columns

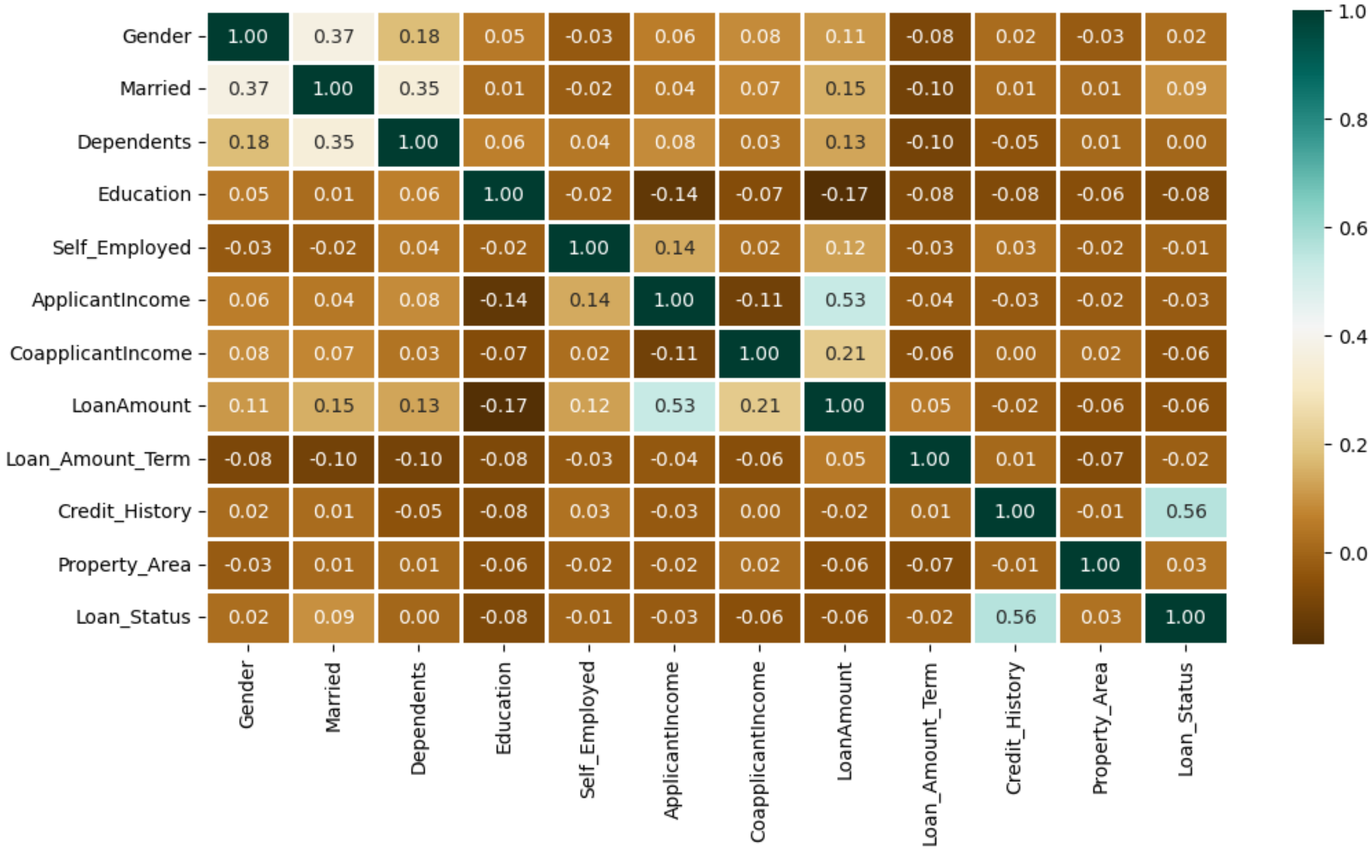
In [53]:
```
obj = (data.dtypes == 'object')
print("Categori values:",len(list(obj[obj].index)))
```

```
Categori values: 0
```

In [54]:
```python
plt.figure(figsize=(12,6))

sns.heatmap(data.corr(),cmap='BrBG',fmt='.2f',linewidths=2,annot=True)
```

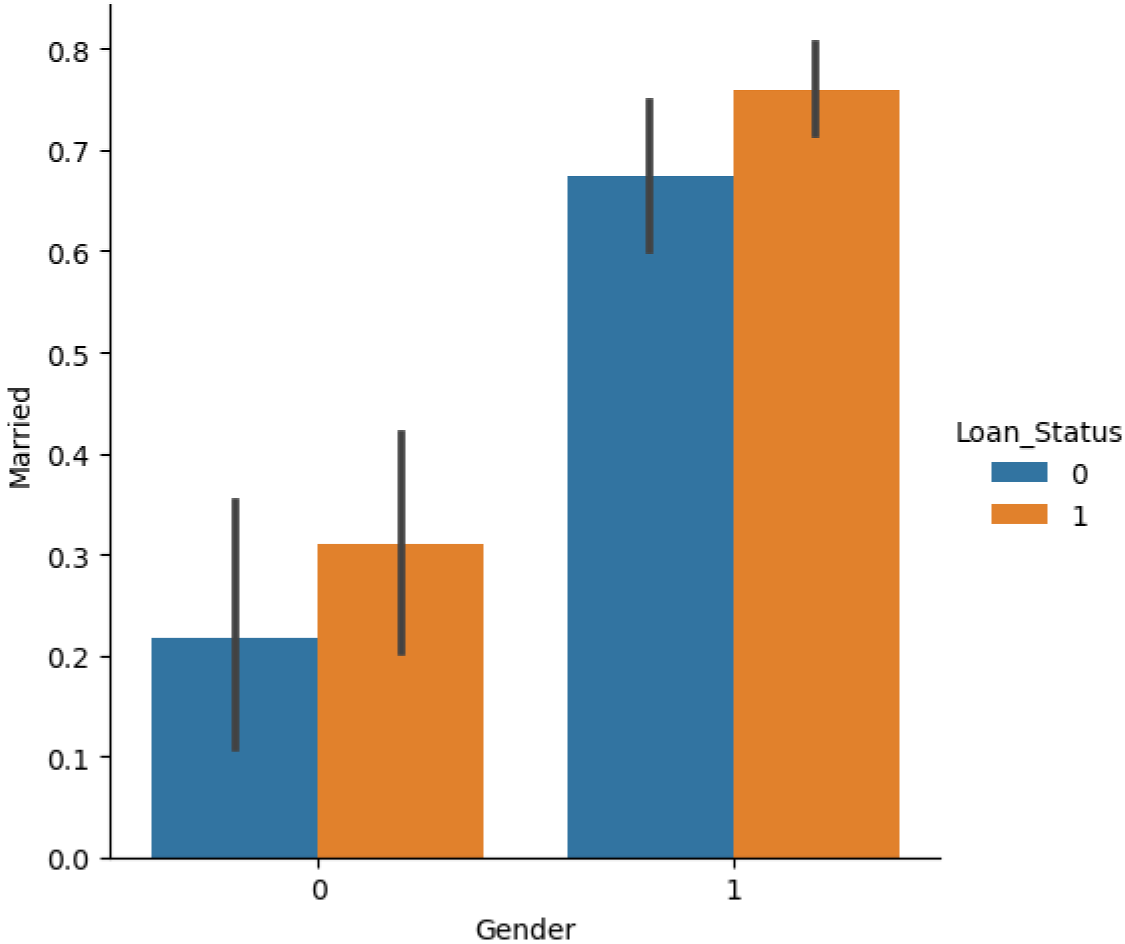
Out[54]: <AxesSubplot:>



The above heatmap shows the correlation between loan amount and applicant income. It also demonstrates how strongly credit history influences loan status.

Now we will use Catplot to visualize the plot for the gender and marital status of the applicant.

In [79]:
```python
sns.catplot(x="Gender", y="Married", hue="Loan_Status", kind="bar", data=data)
plt.show()
```



In [56]:
```
for col in data.columns:
  data[col] = data[col].fillna(data[col].mean())

data.isna().sum()
```

Out[56]:
```
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
```

As there is no missing value then we must proceed to model training.

In [59]:
```
from sklearn.model_selection import train_test_split

X = data.drop(['Loan_Status'],axis=1)
Y = data['Loan_Status']
X.shape,Y.shape

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,test_size=0.4,random_state=1)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[59]:
```
((358, 11), (240, 11), (358,), (240,))
```

In [68]:
```
knn = KNeighborsClassifier(n_neighbors=3)
```

```python
rfc = RandomForestClassifier(n_estimators=7, criterion='entropy', random_state=7)
svc = SVC()
lc = LogisticRegression(max_iter=1000)

for clf in (rfc, knn, svc, lc):
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_train)
    print("Accuracy score of",clf.__class__.__name__,"=", 100 * metrics.accuracy_score(Y
```

```
Accuracy score of RandomForestClassifier = 98.04469273743017
Accuracy score of KNeighborsClassifier = 78.49162011173185
Accuracy score of SVC = 68.71508379888269
Accuracy score of LogisticRegression = 79.60893854748603
```

In [69]:
```python
for clf in (rfc, knn, svc,lc):
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_test)
    print("Accuracy score of ",
          clf.__class__.__name__,"=",
          100*metrics.accuracy_score(Y_test,Y_pred))
```

```
Accuracy score of  RandomForestClassifier = 82.5
Accuracy score of  KNeighborsClassifier = 63.74999999999999
Accuracy score of  SVC = 69.16666666666667
Accuracy score of  LogisticRegression = 80.41666666666667
```

With an accuracy score of 82% for the testing dataset, the Random Forest Classifier provides the best results. Additionally, ensemble learning strategies like bagging and boosting can be applied to obtain far better outcomes.

Don't pass up the opportunity to benefit from the data revolution! By leveraging data, every industry is reaching new heights. Develop your abilities and join the most popular movement of the twenty-first century.