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

data = pd.read_csv("/content/LoanApprovalPrediction.csv")
```

```
data.head(5)
```

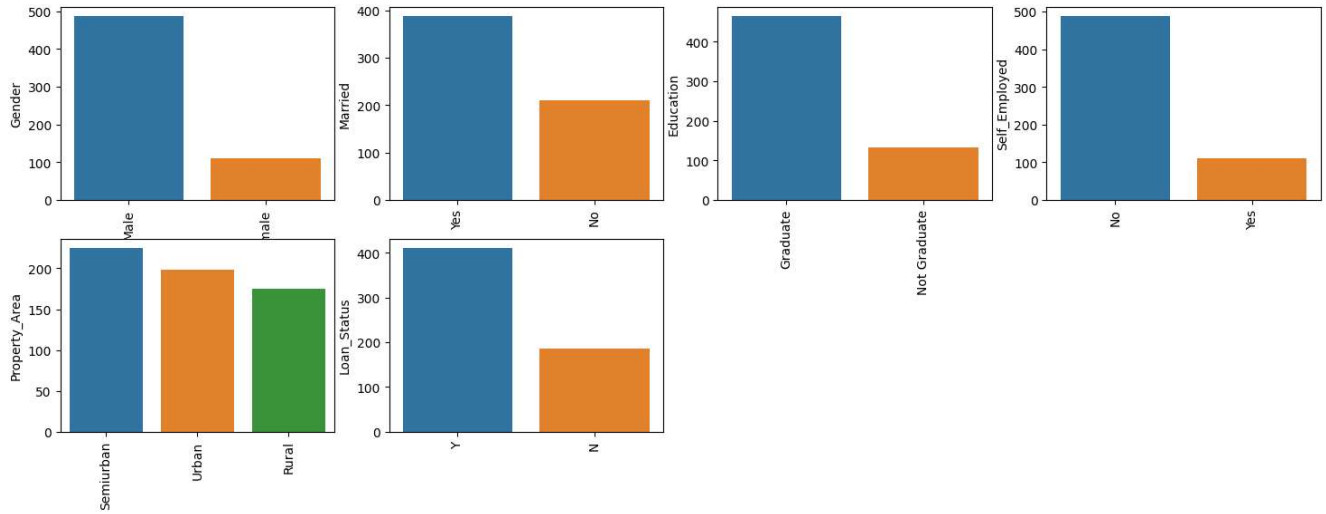
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
0	LP001002	Male	No	0.0	Graduate	No	5849	
1	LP001003	Male	Yes	1.0	Graduate	No	4583	
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	
4	LP001008	Male	No	0.0	Graduate	No	6000	

```
obj = (data.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

```
Categorical variables: 7
```

```
# Dropping Loan_ID column
data.drop(['Loan_ID'],axis=1,inplace=True)
```

```
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
```

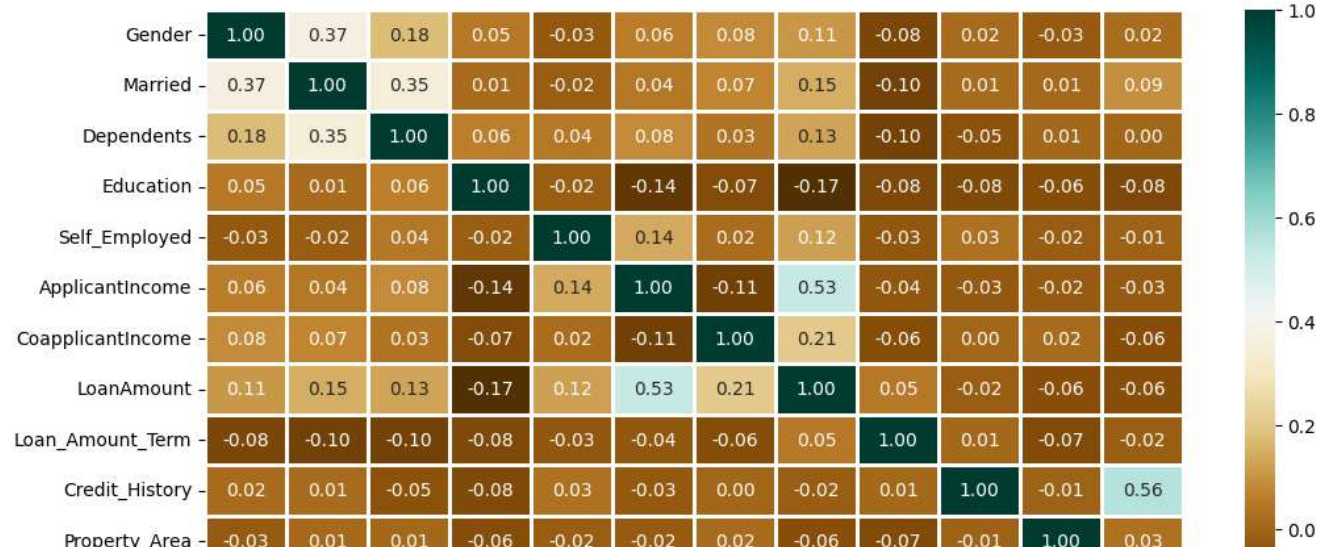



```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how
# to understand word labels.
label_encoder = preprocessing.LabelEncoder()
obj = (data.dtypes == 'object')
for col in list(obj[obj].index):
  data[col] = label_encoder.fit_transform(data[col])



# To find the number of columns with
# datatype==object
obj = (data.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

```
     Categorical variables: 0
```

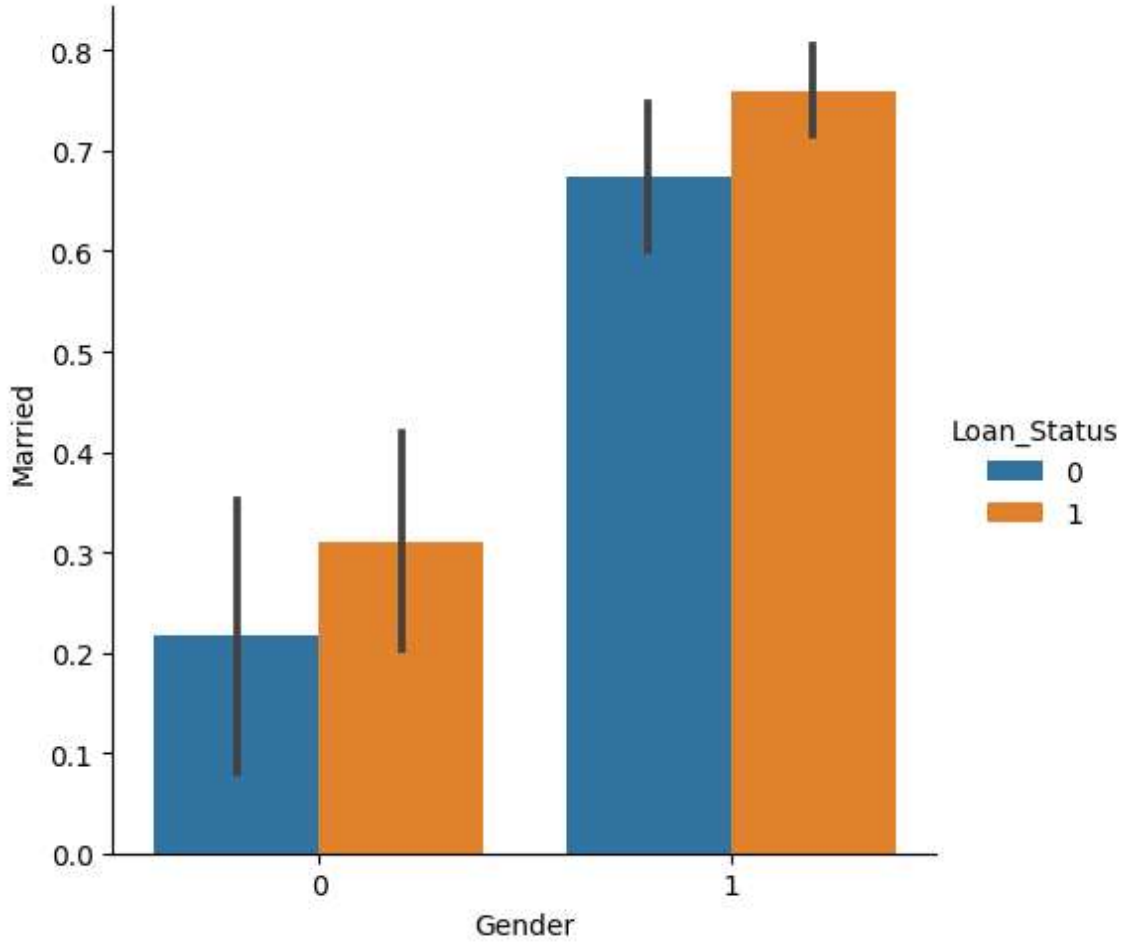
```
plt.figure(figsize=(12,6))

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

```
<Axes: >
```




```
sns.catplot(x="Gender", y="Married",
            hue="Loan_Status",
            kind="bar",
            data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7ba8049d7040>
```



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

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

```
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
```

```
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
```

```
((358, 11), (240, 11), (358,), (240,))
```

```
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

knn = KNeighborsClassifier(n_neighbors=3)
rfc = RandomForestClassifier(n_estimators = 7,
                             criterion = 'entropy',
                             random_state =7)
svc = SVC()
lc = LogisticRegression()

# making predictions on the training set
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_train,
                                       Y_pred))
```

```
 Accuracy score of  RandomForestClassifier = 98.04469273743017
 Accuracy score of  KNeighborsClassifier = 78.49162011173185
 Accuracy score of  SVC = 68.71508379888269
 Accuracy score of  LogisticRegression = 80.44692737430168
```

```
# making predictions on the testing set
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 = 80.83333333333333
```