In [1]:
```python
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn import svm
from sklearn.linear_model import LogisticRegression
```

In [2]:
```python
loan_dataset = pd.read_csv("D:\Machine learning\Loan Status\Loan_status.csv")
```

In [3]:
```python
loan_dataset
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Ter
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360
...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360

614 rows × 13 columns

In [4]:
```python
loan_dataset.shape
```

Out[4]: (614, 13)

In [5]:
```python
loan_dataset.describe()
```

Out[5]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [6]:
```python
loan_dataset.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]:
```python
loan_dataset.isnull().sum()
```

Out[7]:
```
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 [8]:
```python
#Missing Value
```

In [9]:
```python
loan_dataset['Gender'].fillna(loan_dataset['Gender'].mode()[0], inplace=True)
loan_dataset['Married'].fillna(loan_dataset['Married'].mode()[0],inplace=True)
loan_dataset['Self_Employed'].fillna(loan_dataset['Self_Employed'].mode()[0],inplace=True)
loan_dataset['Credit_History'].fillna(loan_dataset['Credit_History'].mode()[0],inplace=True)
loan_dataset['Dependents'].fillna(loan_dataset['Dependents'].mode()[0],inplace=True)
```

In [10]:
```python
loan_dataset.isnull().sum()
```

Out[10]:
```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

In [11]:
```python
loan_dataset['Loan_Amount_Term'].value_counts()
```

Out[11]:
```
360.0    512
180.0     44
480.0     15
300.0     13
84.0       4
240.0      4
120.0      3
36.0       2
60.0       2
12.0       1
Name: Loan_Amount_Term, dtype: int64
```

In [12]:
```python
loan_dataset['Loan_Amount_Term'].fillna(loan_dataset['Loan_Amount_Term'].mode()[0],inplace=True)
```

In [13]:
```python
loan_dataset['Dependents'].value_counts()
```

Out[13]:
```
0     360
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [14]:
```python
loan_dataset['LoanAmount'].fillna(loan_dataset['LoanAmount'].median(), inplace=True)
```

In [15]:
```python
loan_dataset.isnull().sum()
```

Out[15]:
```
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
```

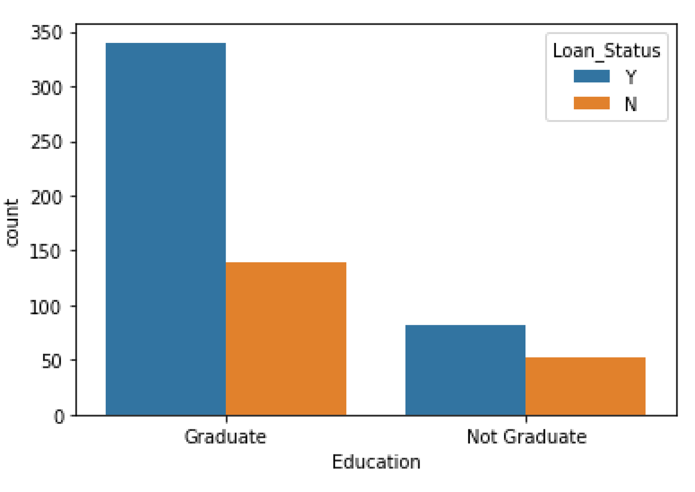
In [16]:
```python
loan_dataset = loan_dataset.replace(to_replace='3+', value=4)
```

In [17]:
```python
loan_dataset['Dependents'].value_counts()
```

Out[17]:
```
0    360
1    102
2    101
4     51
Name: Dependents, dtype: int64
```

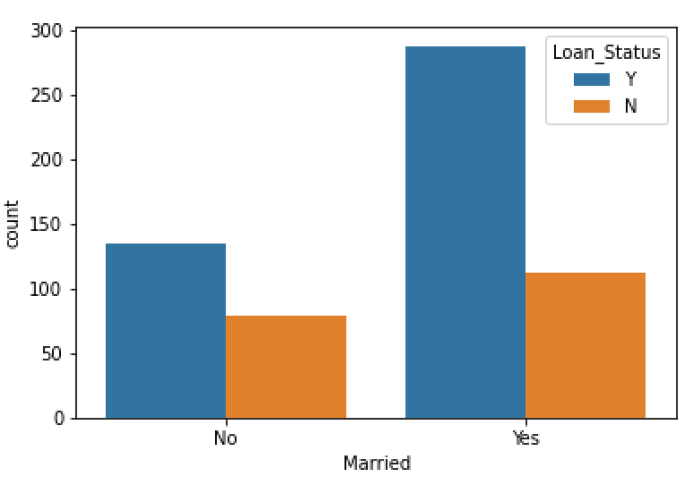
In [18]:
```python
sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)
```

Out[18]: <AxesSubplot:xlabel='Education', ylabel='count'>



In [19]:
```python
sns.countplot(x='Married',hue='Loan_Status',data=loan_dataset)
```

Out[19]: <AxesSubplot:xlabel='Married', ylabel='count'>



In [20]:
```python
loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self_Employed':{'No':0,'Yes':1
                      'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Graduate':1,'Not Gradua
```

In [21]:
```python
loan_dataset.head()
```

Out[21]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	1	0	0	1	0	5849	0.0	128.0	360.0
1	LP001003	1	1	1	1	0	4583	1508.0	128.0	360.0
2	LP001005	1	1	0	1	1	3000	0.0	66.0	360.0
3	LP001006	1	1	0	0	0	2583	2358.0	120.0	360.0
4	LP001008	1	0	0	1	0	6000	0.0	141.0	360.0

In [22]:
```python
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = loan_dataset['Loan_Status']
```

In [23]:
```python
X
```

Out[23]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_I
0	1	0	0	1	0	5849	0.0	128.0	360.0	
1	1	1	1	1	0	4583	1508.0	128.0	360.0	
2	1	1	0	1	1	3000	0.0	66.0	360.0	
3	1	1	0	0	0	2583	2358.0	120.0	360.0	
4	1	0	0	1	0	6000	0.0	141.0	360.0	
...	...	...	...	...	...	...	...	...	...	
609	0	0	0	1	0	2900	0.0	71.0	360.0	
610	1	1	4	1	0	4106	0.0	40.0	180.0	
611	1	1	1	1	0	8072	240.0	253.0	360.0	
612	1	1	2	1	0	7583	0.0	187.0	360.0	
613	0	0	0	1	1	4583	0.0	133.0	360.0	

614 rows × 11 columns

In [24]:
```python
Y
```

Out[24]:
```
0      Y
1      N
2      Y
3      Y
4      Y
      ..
609    Y
610    Y
611    Y
612    Y
613    N
Name: Loan_Status, Length: 614, dtype: object
```

In [25]:
```python
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

In [26]:
```python
X.shape
```

Out[26]: (614, 11)

In [27]:
```python
X_train.shape
```

Out[27]: (552, 11)

In [28]:
```python
X_test.shape
```

Out[28]: (62, 11)

In [29]:
```python
#classifier=svm.SVC(kernel='linear')
```

In [30]:
```python
model = LogisticRegression()
```

In [31]:
```python
model.fit(X_train,Y_train)
```

```
C:\Users\GAURAV RATHOD\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[31]: LogisticRegression()

In [32]:
```python
# accuracy score on training data
X_train_prediction = model.predict(X_train)
training_data_accuray = accuracy_score(X_train_prediction,Y_train)
```

In [33]:
```python
print('Accuracy on training data : ', training_data_accuray)
```

```
Accuracy on training data :  0.8079710144927537
```

In [34]:
```python
X_test_prediction = model.predict(X_test)
test_data_accuray = accuracy_score(X_test_prediction,Y_test)
```

In [35]:
```python
print('Accuracy on training data : ', test_data_accuray)
```

```
Accuracy on training data :  0.7741935483870968
```

In [37]:
```python
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV


model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto'),
        'params' : {
            'C': [1,10,20],
            'kernel': ['rbf','linear']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10,15,20]
        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='auto'),
        'params': {
            'C': [1,15,30]
        }
    },
    'DecisionTreeClassifier': {
        'model':DecisionTreeClassifier(random_state=0),
        'params':{
            'max_depth':[3,4,5,6,7,8,9]
            }
    },
    'KNN':{
        'model':KNeighborsClassifier(),
        'params':{
            'n_neighbors':[3,5,7,9,11,13,15]
        }
    }
}
scores = []

for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=False)
    clf.fit(X_test, Y_test)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
df = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df
```

Out[37]:

	model	best_score	best_params
0	svm	0.743590	{'C': 20, 'kernel': 'linear'}
1	random_forest	0.710256	{'n_estimators': 20}
2	logistic_regression	0.743590	{'C': 1}
3	DecisionTreeClassifier	0.666667	{'max_depth': 4}
4	KNN	0.693590	{'n_neighbors': 13}

In [ ]: