Problem Statement: Loan Approval Prediction Problem

Type: Binary Classification Loan approval prediction is classic problem to learn and apply lots of data analysis techniques to create best Classification model.

Given with the dataset consisting of details of applicants for loan and status whether the loan application is approved or not. Basis on the a binary classification model is to be created with maximum accuracy.

In [11]:

```
### Step 1
### Import the packages numpy,pandas,
```

In [12]:

```
### Step 2:Load the Dataset
```

In [13]:

```
### Step 3:Explore the data-shape,vis
```

In [14]:

```
### Step 4:X,y-->train data test data
```

In [15]:

```
#Basic and most important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#Classifiers
from sklearn.ensemble import AdaBoost
from sklearn.linear_model import Logi
from sklearn.neighbors import KNeighb
from sklearn.tree import DecisionTree

#Model evaluation tools
from sklearn.metrics import classific
from sklearn.metrics import f1_score
from sklearn.model_selection import c

#Data processing functions

from sklearn.model_selection import t
from sklearn.preprocessing import Lab
le = LabelEncoder()

import warnings
warnings.filterwarnings("ignore")
```

In [16]:

```
data = pd.read_csv("loan_prediction.c
data.head(5)
```

Out[16]:

	Loan_ID	Gender	Married	Dependents
0	LP001002	Male	No	0
1	LP001003	Male	Yes	1
2	LP001005	Male	Yes	0

3	LP001006	Male	Yes	0
4	LP001008	Male	No	0

In [17]:

```
data.shape
```

Out[17]:

```
(614, 13)
```

In [18]:

```
data.dtypes
```

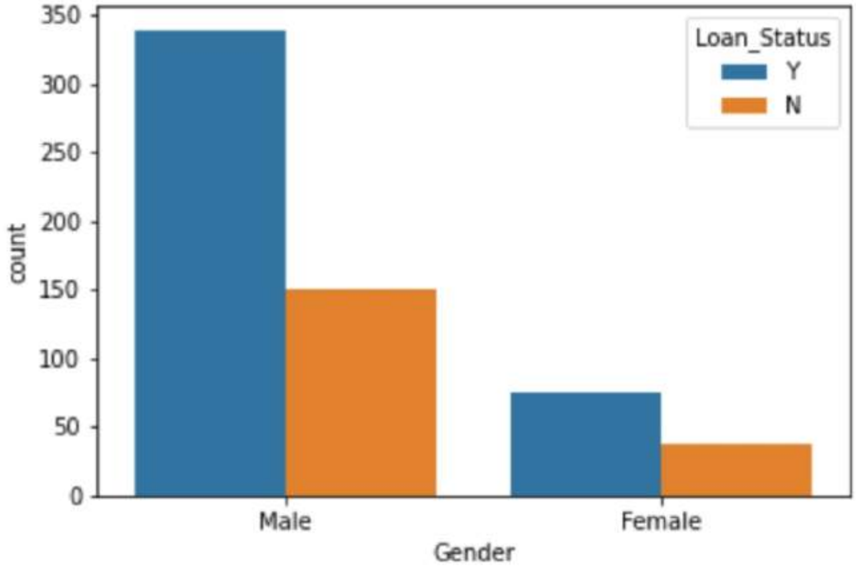
Out[18]:

```
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 [19]:

```
sns.countplot(x="Gender",hue="Loan_St
```

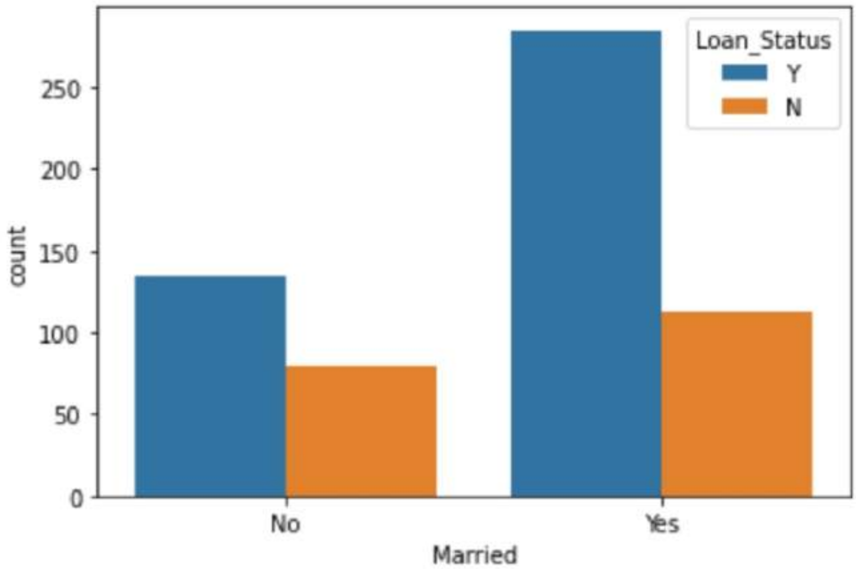
Out[19]:
<matplotlib.axes._subplots.AxesSubplot at 0x1d2296125e0>



In [20]:

```
sns.countplot(x="Married",hue="Loan_S
```

Out[20]:
<matplotlib.axes._subplots.AxesSubplot at 0x1d22b6dbd30>



In [21]:

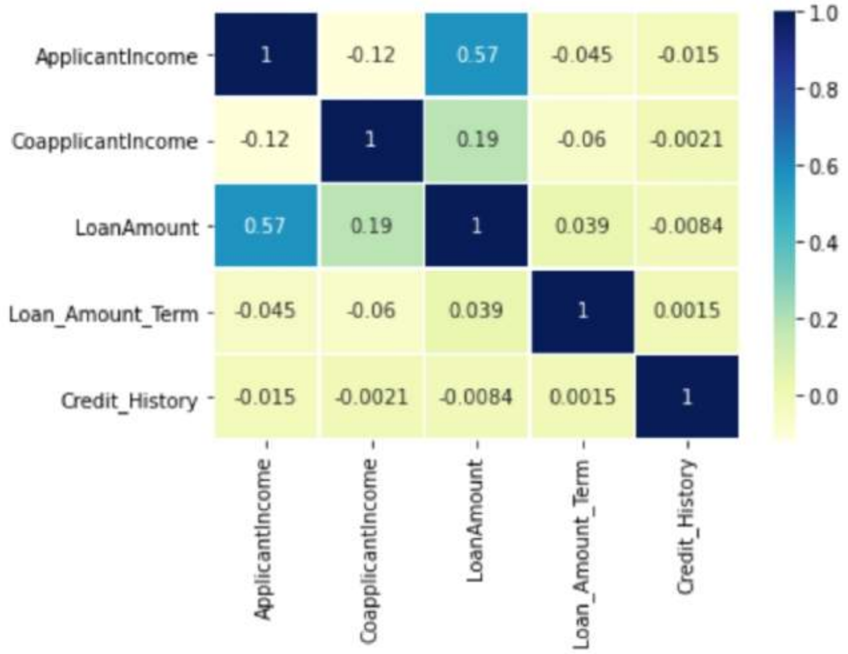
```
correlation_mat = data.corr()
```

In [22]:

```
sns.heatmap(correlation_mat,annot=Tru
```

Out[22]:

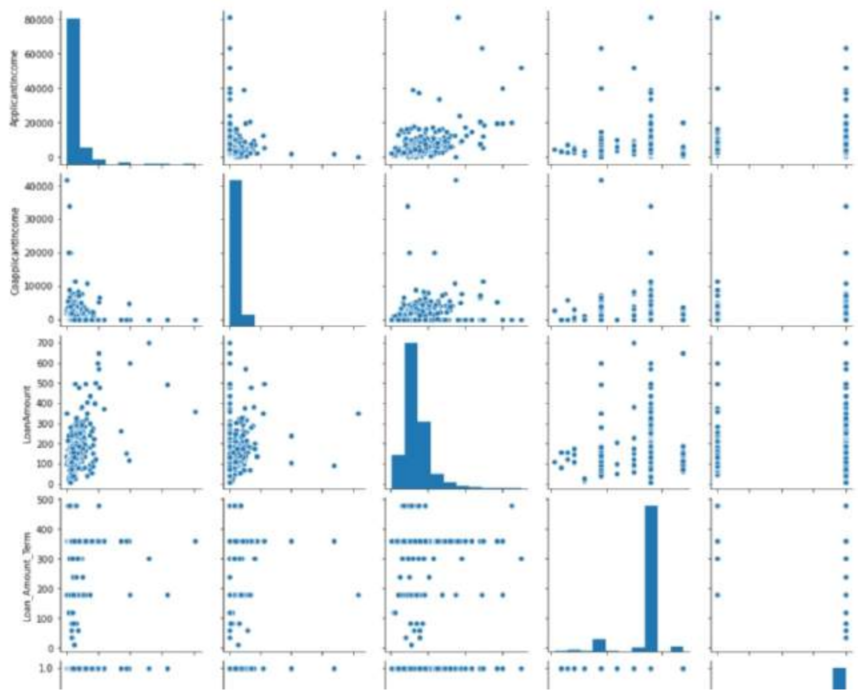
<matplotlib.axes._subplots.AxesSubplot at 0x1d22b756490>

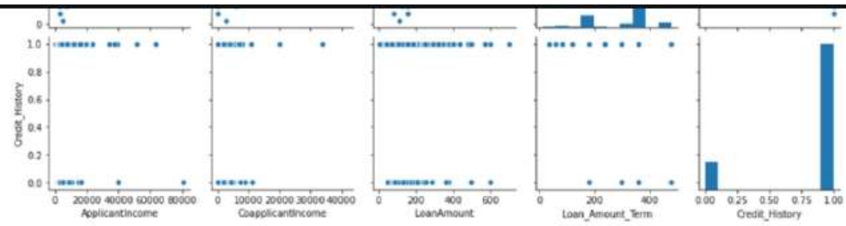


There is a positive correlation between ApplicantIncome and LoanAmount, CoapplicantIncome and LoanAmount.

In [23]:

```
sns.pairplot(data)
plt.show()
```





In [24]:

```
data.describe()
```

Out[24]:

	ApplicantIncome	CoapplicantIncome	L
count	614.000000	614.000000	
mean	5403.459283	1621.245798	
std	6109.041673	2926.248369	
min	150.000000	0.000000	
25%	2877.500000	0.000000	
50%	3812.500000	1188.500000	
75%	5795.000000	2297.250000	
max	81000.000000	41667.000000	

In [25]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count
Dtype
---  ------             --------------
-----
 0   Loan_ID            614 non-null
object
 1   Gender             601 non-null
object
 2   Married            611 non-null
object
 3   Dependents         599 non-null
object
```

```
object
 5   Self_Employed      582 non-null
object
 6   ApplicantIncome    614 non-null
int64
 7   CoapplicantIncome  614 non-null
float64
 8   LoanAmount         592 non-null
float64
 9   Loan_Amount_Term   600 non-null
float64
 10  Credit_History     564 non-null
float64
 11  Property_Area      614 non-null
object
 12  Loan_Status        614 non-null
object
dtypes: float64(4), int64(1), object
(8)
memory usage: 62.5+ KB
```

In [26]:

```
data.isnull().sum()
```

Out[26]:

```
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 [27]:

```
plt.figure(figsize=(10,6))
sns.heatmap(data.isnull(),yticklabels
```

In [28]:

```
print(data["Gender"].value_counts())
print(data["Married"].value_counts())
print(data["Self_Employed"].value_cou
print(data["Dependents"].value_counts
print(data["Credit_History"].value_co
print(data["Loan_Amount_Term"].value_
```

```
Male      489
Female    112
Name: Gender, dtype: int64
Yes    398
No     213
Name: Married, dtype: int64
No     500
Yes     82
Name: Self_Employed, dtype: int64
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
1.0    475
0.0     89
Name: Credit_History, dtype: int64
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 [29]:

```
#Filling all Nan values with mode of
data["Gender"].fillna(data["Gender"].
data["Married"].fillna(data["Married"
data["Self_Employed"].fillna(data["Se
data["Loan_Amount_Term"].fillna(data[
```

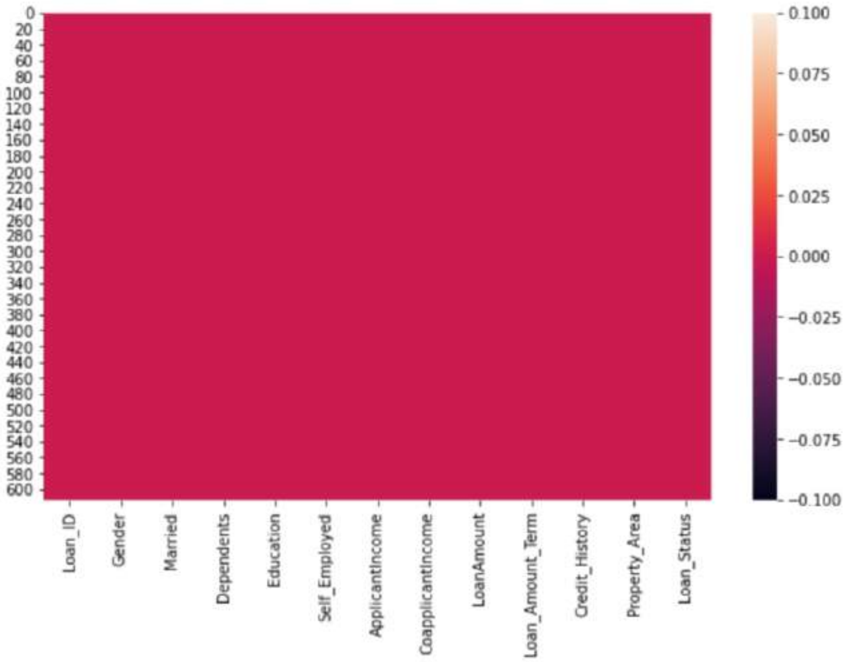
```
#All values of "Dependents" columns w
data["Dependents"] = data["Dependents
data["Dependents"] = data["Dependents
data["Dependents"] = data["Dependents
data["Dependents"] = data["Dependents

data["LoanAmount"].fillna(data["LoanA

print(data.isnull().sum())

#Heat map for null values
plt.figure(figsize=(10,6))
sns.heatmap(data.isnull())
```

```
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
Out[29]:
<matplotlib.axes._subplots.AxesSubplot
at 0x1d22d80aca0>
```



In [30]:

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

Out[30]:

	Loan_ID	Gender	Married	Dependents
0	LP001002	Male	No	0
1	LP001003	Male	Yes	1
2	LP001005	Male	Yes	0
3	LP001006	Male	Yes	0
4	LP001008	Male	No	0

In [31]:

```
data["Gender"] = le.fit_transform(dat
data["Married"] = le.fit_transform(da
data["Education"] = le.fit_transform(
data["Self_Employed"] = le.fit_transf
data["Property_Area"] = le.fit_transf
data["Loan_Status"] = le.fit_transfor

#data = pd.get_dummies(data)
data.head(5)
```

Out[31]:

	Loan_ID	Gender	Married	Dependents
0	LP001002	1	0	0
1	LP001003	1	1	1
2	LP001005	1	1	0
3	LP001006	1	1	0
4	LP001008	1	0	0

In [32]:

```
#Dividing data into Input X variables
X = data.drop(["Loan_Status","Loan_ID
y = data["Loan_Status"]
```

In [33]:

```
X_train,X_test,y_train,y_test=train_t
```

In [49]:

```
model=LogisticRegression(solver="libl
```

In [50]:

```
model.fit(X_train,y_train)
```

Out[50]:

```
LogisticRegression(solver='liblinear')
```

In [51]:

```
model.score(X_train,y_train)
```

Out[51]:

```
0.8018648018648019
```

In [52]:

```
model.score(X_test,y_test)
```

Out[52]:

```
0.8324324324324325
```

In [38]:

```
dtree=DecisionTreeClassifier(criterio
dtree.fit(X_train,y_train)
```

Out[38]:

```
DecisionTreeClassifier()
```

In [39]:

```
dtree.score(X_train,y_train)
```

Out[39]:

```
1.0
```

In [40]:

```
dtree.score(X_test,y_test)
```

Out[40]:

```
0.7621621621621621
```

In [55]:

```
dTreeR = DecisionTreeClassifier(crite
dTreeR.fit(X_train, y_train)
print(dTreeR.score(X_train, y_train))
```

```
0.8181818181818182
```

In [56]:

```
y_predict = dTreeR.predict(X_test)
```

In [58]:

```
print(dTreeR.score(X_test, y_test))
```

```
0.8108108108108109
```

In [53]:

```
from sklearn import metrics
```

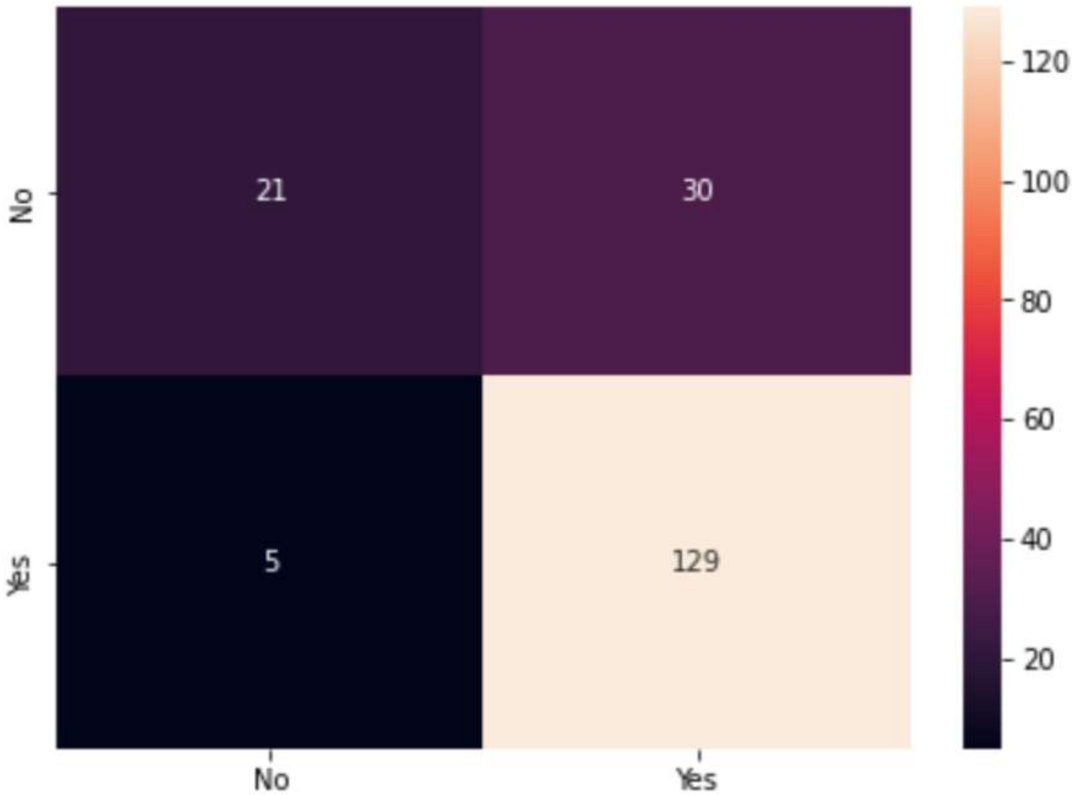
In [54]:

```
cm=metrics.confusion_matrix(y_test, y

df_cm = pd.DataFrame(cm, index = [i f
                     columns = [i for i
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True ,fmt='g
```

Out[54]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d22c0d7c10>



In [59]:

```
from sklearn.ensemble import BaggingC
bgcl = BaggingClassifier( n_estimator
bgcl = bgcl.fit(X_train,y_train)
y_predict = bgcl.predict(X_test)
print(bgcl.score(X_test,y_test))
```

0.827027027027027

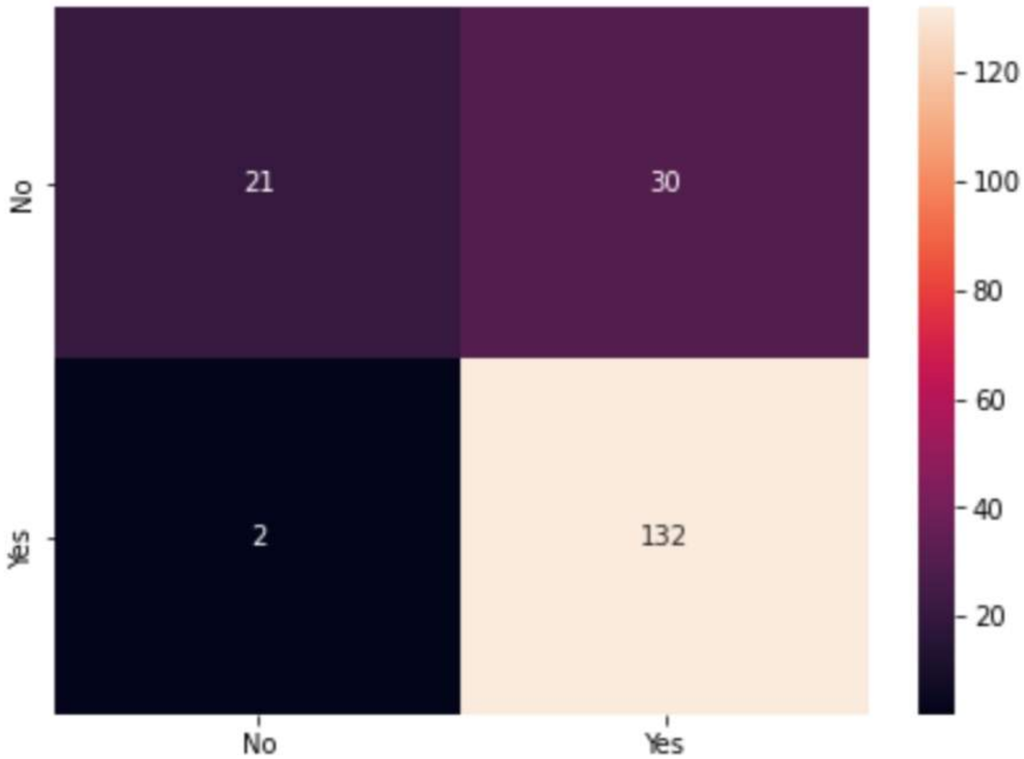
In [60]:

```
from sklearn import metrics
cm=metrics.confusion_matrix(y_test, y

df_cm = pd.DataFrame(cm, index = [i f
                     columns = [i for i
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True ,fmt='g
```

Out[60]:

```
<matplotlib.axes._subplots.AxesSubplot
at 0x1d277faa790>
```



In [71]:

```
from sklearn.ensemble import AdaBoost
abcl = AdaBoostClassifier(n_estimator
abcl = abcl.fit(X_train, y_train)
y_predict = abcl.predict(X_test)
print(abcl.score(X_test, y_test))
```

```
0.8216216216216217
```

In [82]:

```
from sklearn.ensemble import Gradient
gbcl = GradientBoostingClassifier(n_e
gbcl = gbcl.fit(X_train, y_train)
y_predict = gbcl.predict(X_test)
print(gbcl.score(X_test, y_test))
```

0.8108108108108109

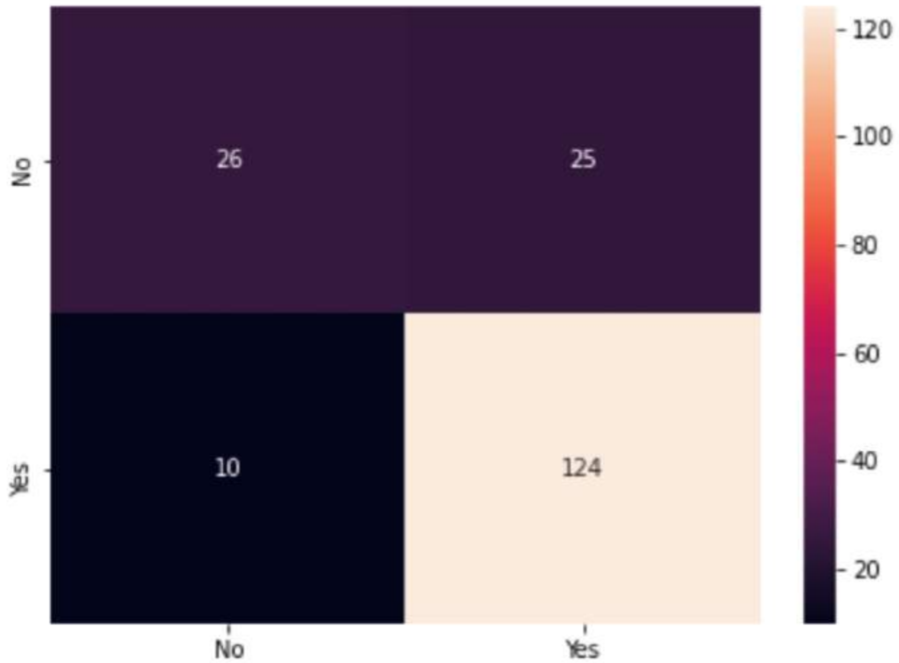
In [83]:

```
cm=metrics.confusion_matrix(y_test, y

df_cm = pd.DataFrame(cm, index = [i f
                     columns = [i for i
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True ,fmt='g
```

Out[83]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d22e18b2b0>



In [96]:

```
from sklearn.ensemble import RandomFo
rfcl = RandomForestClassifier(n_estim
rfcl = rfcl.fit(X_train, y_train)
```

In [97]:

```
y_predict = rfcl.predict(X_test)
print(rfcl.score(X_test, y_test))
cm=metrics.confusion_matrix(y_test, y

df_cm = pd.DataFrame(cm, index = [i f
                  columns = [i for i
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True ,fmt='g
```

```
0.8
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot
at 0x1d22d940f10>
```