In [115...
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [141...
```
data=pd.read_csv("C:\\Users\\reshma_koduri\\OneDrive\\Documents\\archive (3)\\medica
data
```

Out[141...

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate	Bl Press
0	1.0	18.0	Female	161.777924	72.354947	O	27.645835	NaN	95.0	1
1	2.0	NaN	Male	152.069157	47.630941	B	NaN	98.714977	93.0	1
2	3.0	32.0	Female	182.537664	55.741083	A	16.729017	98.260293	76.0	1
3	NaN	30.0	Male	182.112867	63.332207	B	19.096042	98.839605	99.0	1
4	5.0	23.0	Female	NaN	46.234173	O	NaN	98.480008	95.0	N
...	...	...	...	...	...	...	...	...	...	
199995	NaN	24.0	Male	176.503260	95.756997	B	30.737254	99.170685	65.0	1
199996	99997.0	29.0	Female	163.917675	45.225194	NaN	16.831734	97.865785	62.0	1
199997	99998.0	34.0	Female	NaN	99.648914	NaN	33.189303	98.768210	60.0	
199998	99999.0	30.0	Female	156.446944	50.142824	A	20.486823	98.994212	61.0	1
199999	100000.0	20.0	Female	153.927409	99.928405	O	42.175189	98.595817	95.0	1

200000 rows × 13 columns

In [142...
```
a=data.tail(1800)
a
```

Out[142...

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate	Bl Press
198200	98201.0	18.0	Male	NaN	NaN	A	15.165705	99.296993	94.0	1
198201	98202.0	18.0	Female	193.034462	71.347347	B	NaN	97.878775	77.0	1
198202	98203.0	28.0	Male	184.871975	79.299260	O	23.202083	98.959156	63.0	1
198203	NaN	28.0	NaN	198.473003	61.309481	B	15.564127	99.679238	71.0	1
198204	98205.0	18.0	Female	159.772722	43.143915	A	16.901073	99.305172	85.0	1
...	...	...	...	...	...	...	...	...	...	
199995	NaN	24.0	Male	176.503260	95.756997	B	30.737254	99.170685	65.0	1
199996	99997.0	29.0	Female	163.917675	45.225194	NaN	16.831734	97.865785	62.0	1
199997	99998.0	34.0	Female	NaN	99.648914	NaN	33.189303	98.768210	60.0	9
199998	99999.0	30.0	Female	156.446944	50.142824	A	20.486823	98.994212	61.0	1
199999	100000.0	20.0	Female	153.927409	99.928405	O	42.175189	98.595817	95.0	1

1800 rows × 13 columns

In [143...
```
data.tail(10)
```

Out[143...

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate	Bl Press
199990	99991.0	21.0	Female	183.735110	51.172076	AB	15.158238	97.998790	67.0	9
199991	99992.0	28.0	Male	183.499177	NaN	A	26.527962	97.321680	70.0	1
199992	99993.0	34.0	Male	161.590030	90.877589	B	34.803881	98.728836	70.0	9
199993	99994.0	22.0	Male	NaN	46.155224	A	NaN	98.331019	93.0	1(
199994	99995.0	22.0	Male	159.486907	NaN	A	27.631082	98.971976	86.0	1
199995	NaN	24.0	Male	176.503260	95.756997	B	30.737254	99.170685	65.0	1
199996	99997.0	29.0	Female	163.917675	45.225194	NaN	16.831734	97.865785	62.0	1
199997	99998.0	34.0	Female	NaN	99.648914	NaN	33.189303	98.768210	60.0	9
199998	99999.0	30.0	Female	156.446944	50.142824	A	20.486823	98.994212	61.0	1(
199999	100000.0	20.0	Female	153.927409	99.928405	O	42.175189	98.595817	95.0	1

In [144...
```
a.describe()
```

Out[144...

	Student ID	Age	Height	Weight	BMI	Temperature	Heart Rate
count	1633.000000	1627.000000	1604.000000	1614.000000	1626.000000	1618.000000	1619.000000
mean	99098.347214	26.249539	174.607063	69.947989	23.390458	98.600257	79.924027
std	518.285621	4.976123	14.791578	17.338499	7.259284	0.508875	11.491052
min	98201.000000	18.000000	150.000041	40.009476	10.458955	96.843803	60.000000
25%	98654.000000	22.000000	161.589501	54.486845	17.756278	98.254190	70.000000
50%	99097.000000	27.000000	174.619626	69.434012	22.626262	98.596611	80.000000
75%	99542.000000	31.000000	187.507729	84.790252	28.099608	98.950415	90.000000
max	100000.000000	34.000000	199.980696	99.980893	43.694294	100.587479	99.000000

In [145...
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1800 entries, 198200 to 199999
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Student ID      1633 non-null   float64
 1   Age             1627 non-null   float64
 2   Gender          1611 non-null   object 
 3   Height          1604 non-null   float64
```

```
 4   Weight          1614 non-null   float64
 5   Blood Type      1650 non-null   object
 6   BMI             1626 non-null   float64
 7   Temperature     1618 non-null   float64
 8   Heart Rate      1619 non-null   float64
 9   Blood Pressure  1614 non-null   float64
 10  Cholesterol     1603 non-null   float64
 11  Diabetes        1625 non-null   object
 12  Smoking         1637 non-null   object
dtypes: float64(9), object(4)
memory usage: 182.9+ KB
```

In [146…

```
print('Gender:',a['Gender'].unique())
print('Blood Type:',a['Blood Type'].unique())
print('Diabetes:',a['Diabetes'].unique())
print('Smoking:',a['Smoking'].unique())
```

```
Gender: ['Male' 'Female' nan]
Blood Type: ['A' 'B' 'O' 'AB' nan]
Diabetes: ['No' nan 'Yes']
Smoking: [nan 'No' 'Yes']
```

In [147…

```
a.isna().sum()
```

Out[147…

```
Student ID        167
Age               173
Gender            189
Height            196
Weight            186
Blood Type        150
BMI               174
Temperature       182
Heart Rate        181
Blood Pressure    186
Cholesterol       197
Diabetes          175
Smoking           163
dtype: int64
```

In [148…

```
a['Gender'].unique()
```

Out[148…

```
array(['Male', 'Female', nan], dtype=object)
```

In [149…

```
a['Student ID']=a['Student ID'].fillna(a['Student ID'].mean())
a['Age']=a['Age'].fillna(a['Age'].mean())
a['Gender']=a['Gender'].fillna(a['Gender'].mode()[0])
a['Height']=a['Height'].fillna(a['Height'].mean())
a['Weight']=a['Weight'].fillna(a['Weight'].mean())
a['Blood Type']=a['Blood Type'].fillna(a['Blood Type'].mode()[0])
a['BMI']=a['BMI'].fillna(a['BMI'].mean())
a['Temperature']=a['Temperature'].fillna(a['Temperature'].mean())
a['Heart Rate']=a['Heart Rate'].fillna(a['Heart Rate'].mean())
a['Blood Pressure']=a['Blood Pressure'].fillna(a['Blood Pressure'].median())
a['Cholesterol']=a['Cholesterol'].fillna(a['Cholesterol'].mean())
a['Diabetes']=a['Diabetes'].fillna(a['Diabetes'].mode()[0])
a['Smoking']=a['Smoking'].fillna(a['Smoking'].mode()[0])
```

In [150…

```
#a.fillna('unknown',inplace=True)
```

In [151…
```
a.isna().sum()
```

Out[151…
```
Student ID        0
Age               0
Gender            0
Height            0
Weight            0
Blood Type        0
BMI               0
Temperature       0
Heart Rate        0
Blood Pressure    0
Cholesterol       0
Diabetes          0
Smoking           0
dtype: int64
```

In [152…
```
a['Gender']=a['Gender'].map({'Male':1,'Female':0})
a
```

Out[152…

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate
198200	98201.000000	18.0	1	174.607063	69.947989	A	15.165705	99.296993	94.0
198201	98202.000000	18.0	0	193.034462	71.347347	B	23.390458	97.878775	77.0
198202	98203.000000	28.0	1	184.871975	79.299260	O	23.202083	98.959156	63.0
198203	99098.347214	28.0	1	198.473003	61.309481	B	15.564127	99.679238	71.0
198204	98205.000000	18.0	0	159.772722	43.143915	A	16.901073	99.305172	85.0
…	…	…	…	…	…	…	…	…	…
199995	99098.347214	24.0	1	176.503260	95.756997	B	30.737254	99.170685	65.0
199996	99997.000000	29.0	0	163.917675	45.225194	B	16.831734	97.865785	62.0
199997	99998.000000	34.0	0	174.607063	99.648914	B	33.189303	98.768210	60.0
199998	99999.000000	30.0	0	156.446944	50.142824	A	20.486823	98.994212	61.0
199999	100000.000000	20.0	0	153.927409	99.928405	O	42.175189	98.595817	95.0

1800 rows × 13 columns

In [153…
```
a['Diabetes']=a['Diabetes'].map({'Yes':1,'No':0})
a
```

Out[153…

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate
198200	98201.000000	18.0	1	174.607063	69.947989	A	15.165705	99.296993	94.0
198201	98202.000000	18.0	0	193.034462	71.347347	B	23.390458	97.878775	77.0
198202	98203.000000	28.0	1	184.871975	79.299260	O	23.202083	98.959156	63.0
198203	99098.347214	28.0	1	198.473003	61.309481	B	15.564127	99.679238	71.0
198204	98205.000000	18.0	0	159.772722	43.143915	A	16.901073	99.305172	85.0

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate
...	...	...	...	...	...	...	...	...	...
199995	99098.347214	24.0	1	176.503260	95.756997	B	30.737254	99.170685	65.0
199996	99997.000000	29.0	0	163.917675	45.225194	B	16.831734	97.865785	62.0
199997	99998.000000	34.0	0	174.607063	99.648914	B	33.189303	98.768210	60.0
199998	99999.000000	30.0	0	156.446944	50.142824	A	20.486823	98.994212	61.0
199999	100000.000000	20.0	0	153.927409	99.928405	O	42.175189	98.595817	95.0

1800 rows × 13 columns

In [154...

```
a['Smoking']=a['Smoking'].map({'Yes':1,'No':0})
a
```

Out[154...

	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate
198200	98201.000000	18.0	1	174.607063	69.947989	A	15.165705	99.296993	94.0
198201	98202.000000	18.0	0	193.034462	71.347347	B	23.390458	97.878775	77.0
198202	98203.000000	28.0	1	184.871975	79.299260	O	23.202083	98.959156	63.0
198203	99098.347214	28.0	1	198.473003	61.309481	B	15.564127	99.679238	71.0
198204	98205.000000	18.0	0	159.772722	43.143915	A	16.901073	99.305172	85.0
...	...	...	...	...	...	...	...	...	...
199995	99098.347214	24.0	1	176.503260	95.756997	B	30.737254	99.170685	65.0
199996	99997.000000	29.0	0	163.917675	45.225194	B	16.831734	97.865785	62.0
199997	99998.000000	34.0	0	174.607063	99.648914	B	33.189303	98.768210	60.0
199998	99999.000000	30.0	0	156.446944	50.142824	A	20.486823	98.994212	61.0
199999	100000.000000	20.0	0	153.927409	99.928405	O	42.175189	98.595817	95.0

1800 rows × 13 columns

In [155...

```
#dummies = pd.get_dummies(a[['Gender', 'Blood Type' , 'Smoking']], drop_first = True
#data_dummies = pd.concat([a, dummies], axis = 1)
#data_dummies.head()
#data_dummies = data_dummies.drop(['Gender', 'Blood Type',  'Smoking'], axis=1)
#data_dummies.head()
```

In [156...

```
data.groupby(['Diabetes']).count()
```

Out[156...

Diabetes	Student ID	Age	Gender	Height	Weight	Blood Type	BMI	Temperature	Heart Rate	Blood Pressure
No	146184	146123	146114	145984	146166	146136	146118	146146	146071	146059
Yes	16223	16227	16214	16268	16197	16220	16192	16239	16240	16229

In [157...

```
a['Gender'].unique()
```

Out[157...

```
array([1, 0], dtype=int64)
```

In [158...

```
a.groupby(['Gender']).count()
```

Out[158...

Gender	Student ID	Age	Height	Weight	Blood Type	BMI	Temperature	Heart Rate	Blood Pressure	Cholesterol	Di
0	786	786	786	786	786	786	786	786	786	786	
1	1014	1014	1014	1014	1014	1014	1014	1014	1014	1014	

In [164...

```
b=a.drop(['Student ID'],axis=1)
```

In [165...

```
c=pd.get_dummies(b,dtype=int)
c
```

Out[165...

	Age	Gender	Height	Weight	BMI	Temperature	Heart Rate	Blood Pressure	Cholesterol
198200	18.0	1	174.607063	69.947989	15.165705	99.296993	94.0	105.0	196.0
198201	18.0	0	193.034462	71.347347	23.390458	97.878775	77.0	106.0	157.0
198202	28.0	1	184.871975	79.299260	23.202083	98.959156	63.0	109.0	204.0
198203	28.0	1	198.473003	61.309481	15.564127	99.679238	71.0	122.0	224.0
198204	18.0	0	159.772722	43.143915	16.901073	99.305172	85.0	118.0	166.0
...	...	...	...	...	...	...	...	...	...
199995	24.0	1	176.503260	95.756997	30.737254	99.170685	65.0	121.0	130.0
199996	29.0	0	163.917675	45.225194	16.831734	97.865785	62.0	125.0	198.0
199997	34.0	0	174.607063	99.648914	33.189303	98.768210	60.0	90.0	154.0
199998	30.0	0	156.446944	50.142824	20.486823	98.994212	61.0	106.0	225.0
199999	20.0	0	153.927409	99.928405	42.175189	98.595817	95.0	133.0	132.0

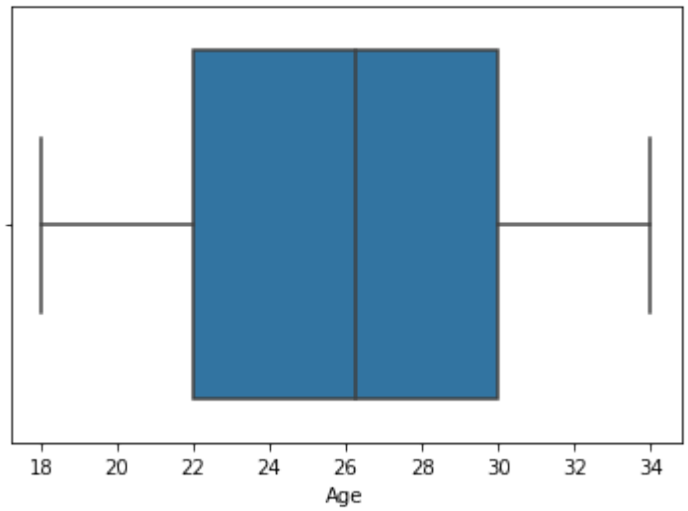
1800 rows × 15 columns

In [166…

```
import seaborn as sb
import matplotlib.pyplot as plt
sb.boxplot(b.Age)
```

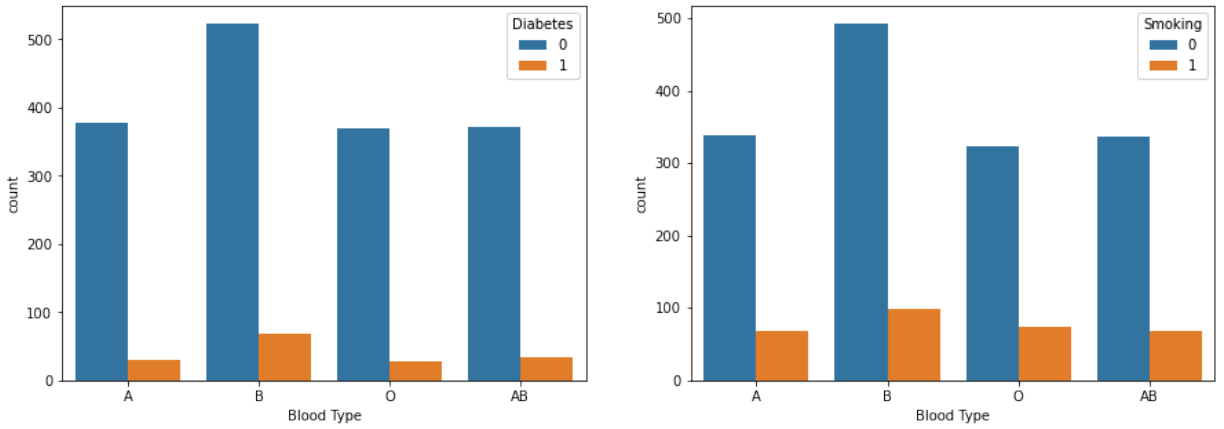
Out[166…

```
<AxesSubplot:xlabel='Age'>
```



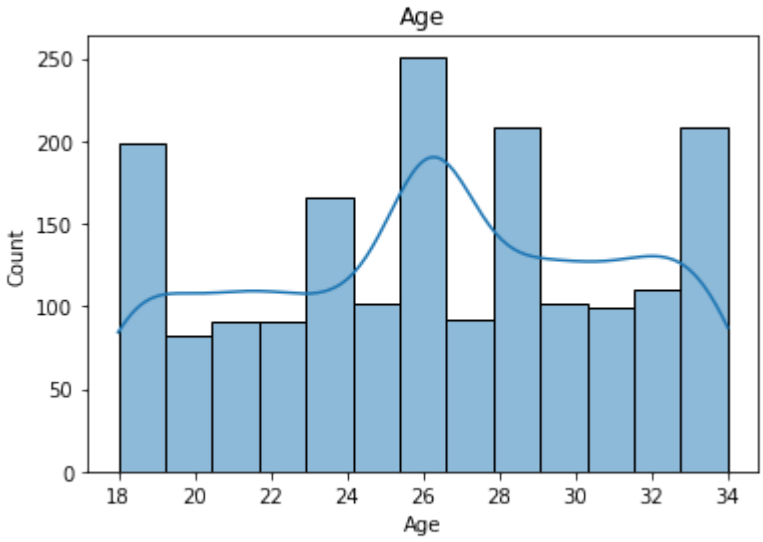
In [167…

```
fig, ax = plt.subplots(1,2, figsize=(15,5))
sb.countplot(x='Blood Type', hue='Diabetes', data=b, ax=ax[0])
sb.countplot(x='Blood Type', hue='Smoking', data=b, ax=ax[1])
fig.show()
```



In [168…

```
sb.histplot(a, x='Age', kde=True)
plt.title('Age')
plt.show()
```



In [171…

```
y=c['Diabetes']
y
```

Out[171…

```
198200    0
198201    0
198202    0
198203    0
198204    0
         ..
199995    0
199996    0
199997    0
199998    0
199999    0
Name: Diabetes, Length: 1800, dtype: int64
```

In [172…

```
x=c.drop(['Diabetes'],axis=1)
x
```

Out[172…

	Age	Gender	Height	Weight	BMI	Temperature	Heart Rate	Blood Pressure	Cholesterol
198200	18.0	1	174.607063	69.947989	15.165705	99.296993	94.0	105.0	196.0
198201	18.0	0	193.034462	71.347347	23.390458	97.878775	77.0	106.0	157.0
198202	28.0	1	184.871975	79.299260	23.202083	98.959156	63.0	109.0	204.0
198203	28.0	1	198.473003	61.309481	15.564127	99.679238	71.0	122.0	224.0
198204	18.0	0	159.772722	43.143915	16.901073	99.305172	85.0	118.0	166.0
…	…	…	…	…	…	…	…	…	…
199995	24.0	1	176.503260	95.756997	30.737254	99.170685	65.0	121.0	130.0
199996	29.0	0	163.917675	45.225194	16.831734	97.865785	62.0	125.0	198.0
199997	34.0	0	174.607063	99.648914	33.189303	98.768210	60.0	90.0	154.0
199998	30.0	0	156.446944	50.142824	20.486823	98.994212	61.0	106.0	225.0
199999	20.0	0	153.927409	99.928405	42.175189	98.595817	95.0	133.0	132.0

1800 rows × 14 columns

In [173…

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [174…

```
print(x_train.shape, y_train.shape)
```

```
(1206, 14) (1206,)
```

In [175…

```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
classifier.fit(x_train, y_train)
```

Out[175…

```
▾ LogisticRegression
LogisticRegression()
```

In [176…

```
ypred=classifier.predict(x_test)
ypred
```

Out[176…

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [177…

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,ypred)
```

Out[177…

```
array([[545,   0],
       [ 49,   0]], dtype=int64)
```

In [178…

```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test,ypred)
```

Out[178…

```
0.9175084175084175
```

In [179…

```python
ypred
```

Out[179…

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [181…

```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
reg=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['gini','entropy']
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth
rfc_reg = GridSearchCV(reg, parameters)
rfc_reg.fit(x_train,y_train)
```

Out[181…

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [182…

```python
rfc_reg.best_params_
```

Out[182…

```
{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 25}
```

In [184…

```
reg=RandomForestClassifier(n_estimators=25,criterion='gini',max_depth=10)
reg.fit(x_train,y_train)
```

Out[184…

```
RandomForestClassifier

RandomForestClassifier(max_depth=10, n_estimators=25)
```

In [185…

```
ypred=reg.predict(x_test)
ypred
```

Out[185…

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [186…

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,ypred)
```

Out[186…

```
0.9158249158249159
```

In [ ]: