Major Project

NAME:-ATUKURI LAKSHMI SAI VENKATESH

COLLEGE:- NIT-TIRUCHIRAPALLI

BRANCH:- CHEMICAL ENGINEERING (2nd Year)

EMAIL:-atukurivenkatesh2@gmail.com

CONTACT:- 9177537099

PROJECT -1:-

Take a dataset of your choice and do EDA (Exploratory Data Analysis)-atleast 5 to 8 different Analysis and Apply a suitable algoritm and calculate the accuracy

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('heart.csv')
df
```

Out[1]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina
0	40	M	ATA	140	289	0	Normal	172	N
1	49	F	NAP	160	180	0	Normal	156	N
2	37	M	ATA	130	283	0	ST	98	N
3	48	F	ASY	138	214	0	Normal	108	Y
4	54	M	NAP	150	195	0	Normal	122	N
...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N
914	68	M	ASY	144	193	1	Normal	141	N
915	57	M	ASY	130	131	0	Normal	115	Y
916	57	F	ATA	130	236	0	LVH	174	N
917	38	M	NAP	138	175	0	Normal	173	N

918 rows × 12 columns

In [2]:
```
df.nunique()
```

Out[2]:
```
Age                 50
Sex                  2
ChestPainType        4
RestingBP           67
Cholesterol        222
FastingBS            2
RestingECG           3
MaxHR              119
ExerciseAngina       2
Oldpeak             53
ST_Slope             3
HeartDisease         2
dtype: int64
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
```

```
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

In [4]:
```
print(df.describe())
```

```
              Age   RestingBP  Cholesterol   FastingBS       MaxHR  \
count  918.000000  918.000000   918.000000  918.000000  918.000000
mean    53.510893  132.396514   198.799564    0.233115  136.809368
std      9.432617   18.514154   109.384145    0.423046   25.460334
min     28.000000    0.000000     0.000000    0.000000   60.000000
25%     47.000000  120.000000   173.250000    0.000000  120.000000
50%     54.000000  130.000000   223.000000    0.000000  138.000000
75%     60.000000  140.000000   267.000000    0.000000  156.000000
max     77.000000  200.000000   603.000000    1.000000  202.000000

          Oldpeak  HeartDisease
count  918.000000    918.000000
mean     0.887364      0.553377
std      1.066570      0.497414
min     -2.600000      0.000000
25%      0.000000      0.000000
50%      0.600000      1.000000
75%      1.500000      1.000000
max      6.200000      1.000000
```

In [5]:
```
df.groupby('HeartDisease').size()
```

Out[5]:
```
HeartDisease
0    410
1    508
dtype: int64
```

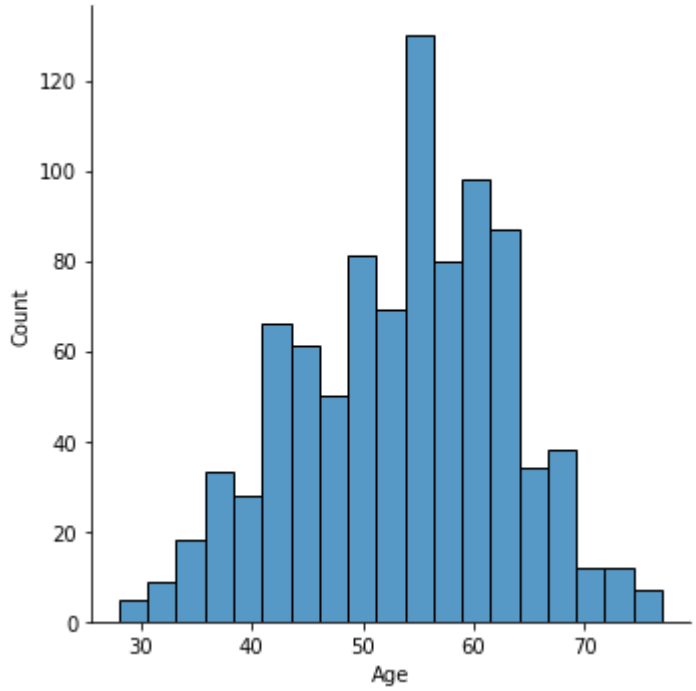
In [6]:
```
df.groupby('Sex').count()
```

Out[6]:

	Age	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldp
Sex									
F	193	193	193	193	193	193	193	193	
M	725	725	725	725	725	725	725	725	

In [7]:
```
sns.displot(df['Age'])
```

Out[7]:
```
<seaborn.axisgrid.FacetGrid at 0x225a7a6daf0>
```

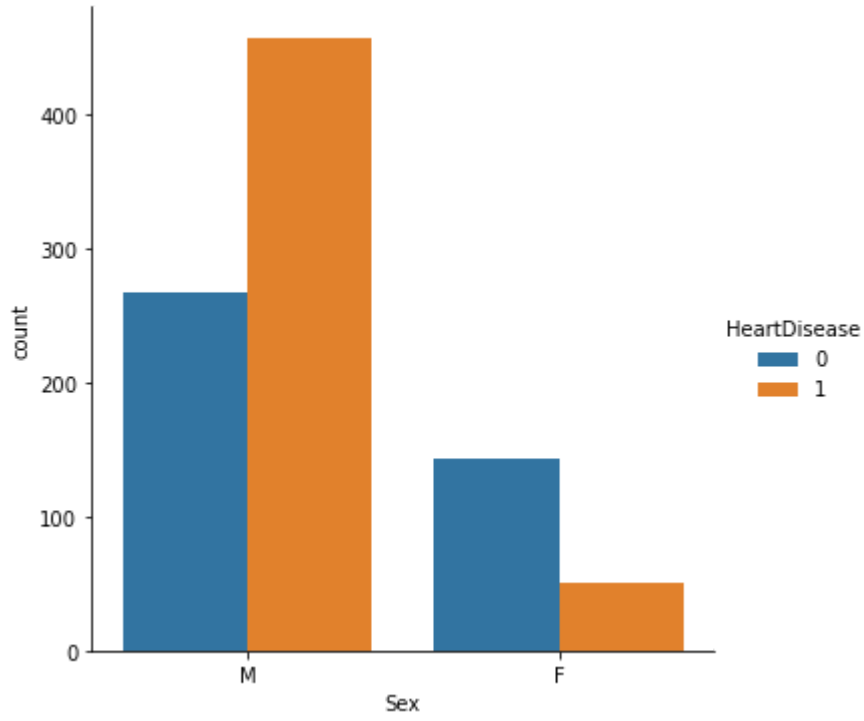


In [8]:
```
young = np.sum((df['Age']>=0) & (df['Age']<20))
adult = np.sum((df['Age']>=20) & (df['Age']<40))
midage = np.sum((df['Age']>=40) & (df['Age']<60))
old = np.sum((df['Age']>=60))
print("\n young =",young,"\n","adult =",adult,"\n","midage =",midage,"\n","old =",old)
```

```
young = 0
adult = 80
midage = 585
old = 253
```

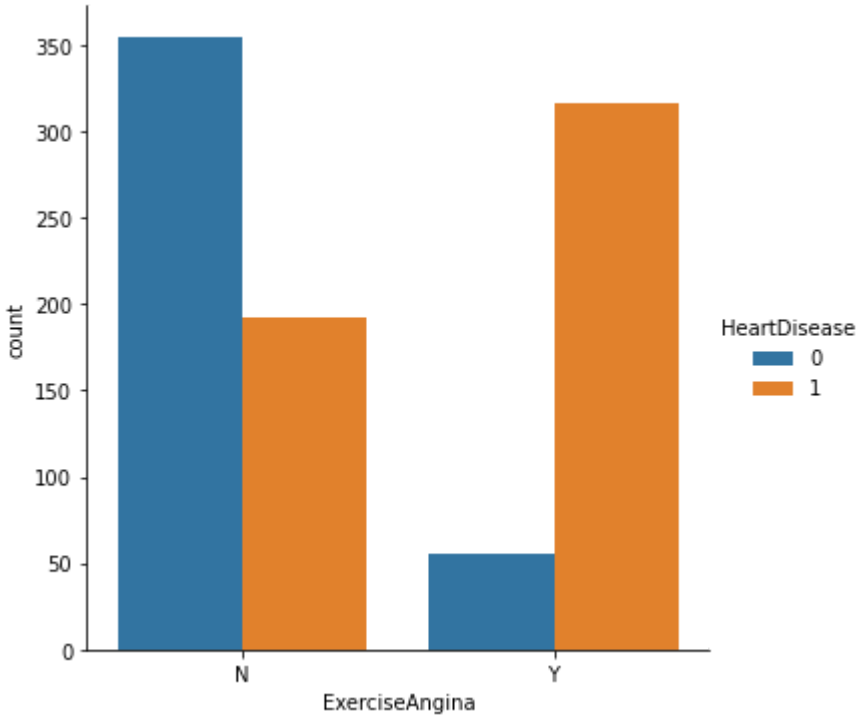
In [9]:
```
sns.catplot(x='Sex',hue='HeartDisease',kind='count',data=df)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x225a7a6d160>



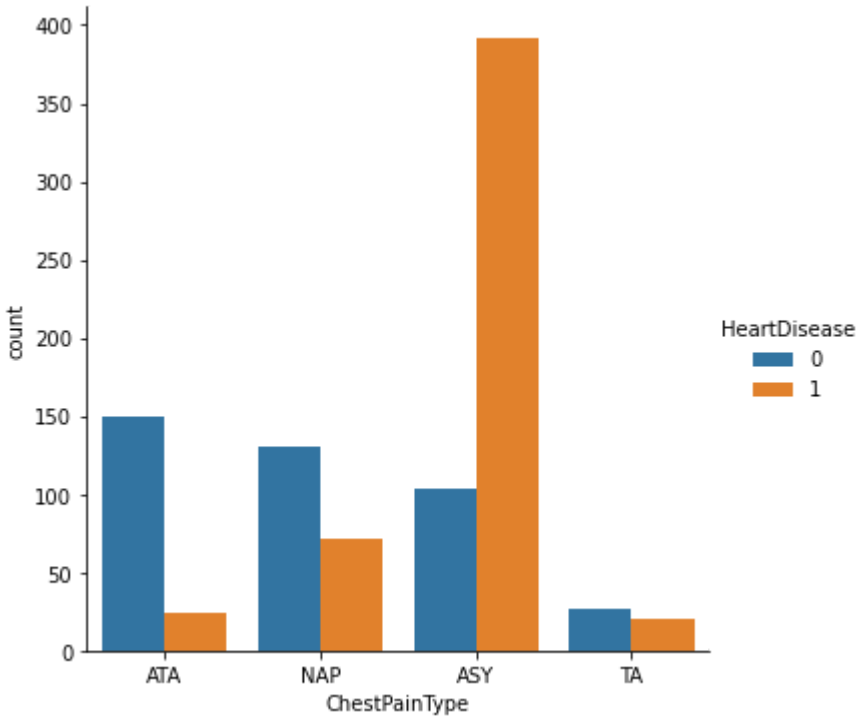
In [10]:
```
sns.catplot(x='ExerciseAngina',hue='HeartDisease',kind='count',data=df)
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x225a7e94670>



In [11]:
```
sns.catplot(x='ChestPainType',hue='HeartDisease',kind='count',data=df)
```

Out[11]: <seaborn.axisgrid.FacetGrid at 0x225a7f1dcd0>



In [17]:
```
# input
# output - survived or not

x = df.iloc[:,0:11].values
y = df.iloc[:,11].values
```

```
print(x)
print(y)
```

```
[[40 'M' 'ATA' ... 'N' 0.0 'Up']
 [49 'F' 'NAP' ... 'N' 1.0 'Flat']
 [37 'M' 'ATA' ... 'N' 0.0 'Up']
 ...
 [57 'M' 'ASY' ... 'Y' 1.2 'Flat']
 [57 'F' 'ATA' ... 'N' 0.0 'Flat']
 [38 'M' 'NAP' ... 'N' 0.0 'Up']]
[0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1
 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0
 1 0 1 0 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0
 1 0 0 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 0
 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0
 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 0 1 1
 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0
 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1 1
 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1
 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 1 1 0
 1 0 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0
 1 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 1 1 0 0 0 1
 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1
 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0
 0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0
 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 1
 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0
 1 0 1 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1
 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1
 1 1 0 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 0]
```

In [18]:

```
#train_test_split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 0)
```

In [19]:

```
print(x.size)
print(x_train.size)
print(x_test.size)
print("\n")
print(y.size)
print(y_train.size)
print(y_test.size)
```

```
10098
8074
2024


918
734
184
```

In [25]:

```
for i in range(0,918):
    if df['ExerciseAngina'][i] == 'N':
        df['ExerciseAngina'][i] = int(0)
    else:
```

```
        df['ExerciseAngina'][i] = int(1)
df
```

```
<ipython-input-25-4cb67c7c975d>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df['ExerciseAngina'][i] = int(0)
<ipython-input-25-4cb67c7c975d>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df['ExerciseAngina'][i] = int(1)
```

Out[25]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina
0	40	M	ATA	140	289	0	Normal	172	0
1	49	F	NAP	160	180	0	Normal	156	0
2	37	M	ATA	130	283	0	ST	98	0
3	48	F	ASY	138	214	0	Normal	108	1
4	54	M	NAP	150	195	0	Normal	122	0
...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	0
914	68	M	ASY	144	193	1	Normal	141	0
915	57	M	ASY	130	131	0	Normal	115	1
916	57	F	ATA	130	236	0	LVH	174	0
917	38	M	NAP	138	175	0	Normal	173	0

918 rows × 12 columns

In [27]:

```
df['ExerciseAngina']=df['ExerciseAngina'].astype('int64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    int64
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(7), object(4)
memory usage: 86.2+ KB
```

In [28]:
```
df_num = df.select_dtypes(include=['float64','int64'])
df_num
```

Out[28]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	HeartDisease
0	40	140	289	0	172	0	0.0	0
1	49	160	180	0	156	0	1.0	1
2	37	130	283	0	98	0	0.0	0
3	48	138	214	0	108	1	1.5	1
4	54	150	195	0	122	0	0.0	0
...	...	...	...	...	...	...	...	...
913	45	110	264	0	132	0	1.2	1
914	68	144	193	1	141	0	3.4	1
915	57	130	131	0	115	1	1.2	1
916	57	130	236	0	174	0	0.0	1
917	38	138	175	0	173	0	0.0	0

918 rows × 8 columns

In [29]:
```
df_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   RestingBP       918 non-null    int64
 2   Cholesterol     918 non-null    int64
 3   FastingBS       918 non-null    int64
 4   MaxHR           918 non-null    int64
 5   ExerciseAngina  918 non-null    int64
 6   Oldpeak         918 non-null    float64
 7   HeartDisease    918 non-null    int64
dtypes: float64(1), int64(7)
memory usage: 57.5 KB
```

In [30]:
```
# input
# output - survived or not

x = df_num.iloc[:,0:7].values
y = df_num.iloc[:,7].values

print(x)
print(y)
```

```
[[ 40.  140.  289.  ... 172.    0.    0. ]
 [ 49.  160.  180.  ... 156.    0.    1. ]
 [ 37.  130.  283.  ...  98.    0.    0. ]
 ...
 [ 57.  130.  131.  ... 115.    1.    1.2]
 [ 57.  130.  236.  ... 174.    0.    0. ]
 [ 38.  138.  175.  ... 173.    0.    0. ]]
```

```
[0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1
 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0
 1 0 1 0 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0
 1 0 0 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 0
 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0
 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 0 1 1
 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0
 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1 1
 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1
 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 1 1 0
 1 0 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0
 1 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 1 1 0 0 0 1
 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1
 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0
 0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0
 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 1
 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0
 1 0 1 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1
 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1
 1 1 0 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 0]
```

In [31]:

```python
#train_test_split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 0)
```

In [32]:

```python
print(x.size)
print(x_train.size)
print(x_test.size)
print("\n")
print(y.size)
print(y_train.size)
print(y_test.size)
```

```
6426
5138
1288


918
734
184
```

In [33]:

```python
#scaling or normalisation
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

In [37]:

```python
#applying regresser or classifier
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()

model.fit(x_train,y_train)
```

Out[37]:

```
LogisticRegression()
```

In [38]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[38]:
```
array([1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

In [39]:
```
y_test
```

Out[39]:
```
array([1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 1], dtype=int64)
```

In [40]:
```
#to check the accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)*100
```

Out[40]:
```
82.06521739130434
```

PROJECT -2

You will be given a dataset [IRIS FLOWER DATASET] , create a model and deploy it using STREAMLIT app and permanently.

USE HEROKU FOR DEPLOYMENT.

URL:- https://iris-data-set.herokuapp.com/

https://github.com/ALSVenky/iris-data-set

Procedure:-

- CREATE a github repo
- Create a procfile and make sure you have setup.sh and app.py mentioned
- Create app.py file and write streamlit code in it.
- Create a requirements.txt file and add the required libraries.
- Create setup.sh file.
- Now go to heroku website and create a new app by linking to current github repository
- Make some negligible changes in the repository and commit the changes.
- After the changes are commited, Streamlink URL is generated.

Search or jump to... / Pull requests Issues Marketplace Explore

ALSVenky / iris-data-set Public

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main iris-data-set / Procfile Go to file ...

ALSVenky Create Procfile Latest commit e04969c 26 minutes ago History

1 contributor

1 lines (1 sloc) | 41 Bytes Raw Blame

```
web: sh setup.sh && streamlit run app.py
```


Search or jump to... / Pull requests Issues Marketplace Explore

ALSVenky / iris-data-set Public

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main iris-data-set / app.py / Jump to Go to file ...

ALSVenky Create app.py Latest commit 5db11c3 25 minutes ago History

1 contributor

18 lines (14 sloc) | 492 Bytes Raw Blame

```
import streamlit as st
st.title("Iris Classifier- API")

sl = st.slider('Sepal Length', 4.3, 7.9, 0.5)
sw = st.slider('Sepal Width', 2.0, 4.4, 0.5)
pl = st.slider('Petal Length', 1.0, 6.9, 0.5)
pw = st.slider('Petal Width', 0.1,2.5, 0.5)

from sklearn.datasets import load_iris
iris = load_iris()

```

Iris Classifier- API

Sepal Length

3.50

0.50 7.90

Sepal Width

2.43

0.50 4.40

Petal Length

2.75

0.50 6.90

Petal Width

1.28

0.10 2.50

versicolor

Search or jump to... / Pull requests Issues Marketplace Explore

ALSVenky / iris-data-set Public

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

ALSVenky Create setup.sh ec66af0 23 minutes ago 4 commits

Procfile	Create Procfile	25 minutes ago
app.py	Create app.py	24 minutes ago
requirements.txt	Create requirements.txt	23 minutes ago
setup.sh	Create setup.sh	23 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package