02_ML_101_Machine_learning

July 29, 2022

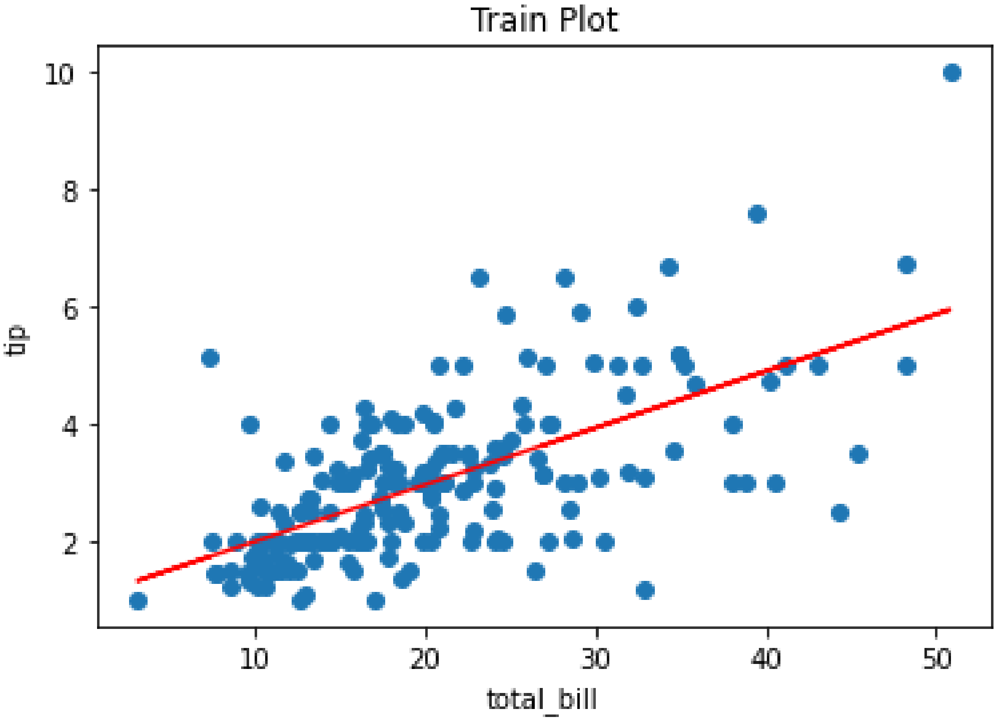
1 Machine Learning Crash Course (ML-101)

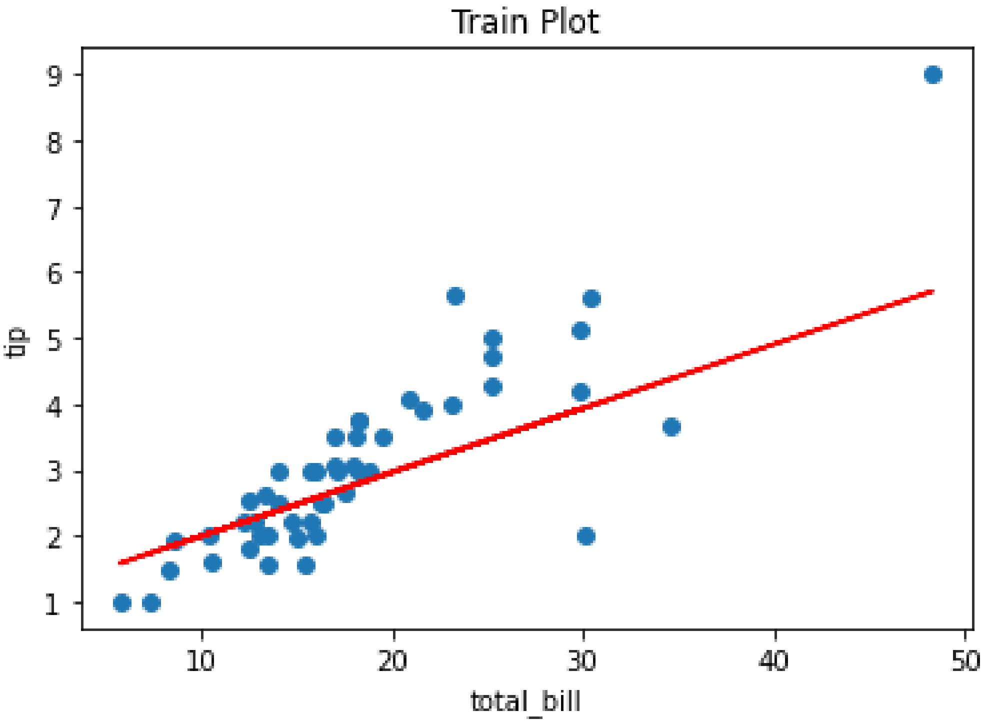
- **Name: Muhammad Waleed Anjum**
- **Email: waleedanjum_2009@yahoo.com****

1.1 Simple Linear Regression

[ ]:

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

# Step-1 Importing 'tips' dataset from seaborn example datasets
data = sns.load_dataset('tips')
data.head()

# Finding number of rows and columns
data.shape

# Step-2 Splitting dataset into training and testing data
x = data[['total_bill']]  # Features/ input
y = data["tip"]           # Labels/ output

x.head()
y.head()

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)

# Step-3 Fit Linear Regression Model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model = model.fit(x_train, y_train)
model

# Step-4 Plotting
# Training Data Plot
plt.scatter(x_train, y_train)
plt.plot(x_train, model.predict(x_train), color='red')
plt.xlabel('total_bill')
plt.ylabel('tip')
plt.title('Train Plot')
plt.show()

# Test Data Plot
plt.scatter(x_test, y_test)
plt.plot(x_test, model.predict(x_test), color='red')
plt.xlabel('total_bill')
plt.ylabel('tip')
plt.title('Train Plot')
plt.show()

# Step-5 Testing or Evaluating Model
# Model Fitness
print('Score for training data = ', model.score(x_train, y_train))
print('Score for testing data = ', model.score(x_test, y_test))

# Step-6 Prediction of unknown values
model.predict([[10]])       # Single value
model.predict([[10], [5],[20]])     # list of values (multiple)
```





```
Score for training data =  0.4150096506457662
Score for testing data =  0.5906895098589039
```

[ ]:
```
array([1.99707797, 1.51281096, 2.965612  ])
```

1.2 Multi Linear Regression

1.2.1 Assignemnt 1:

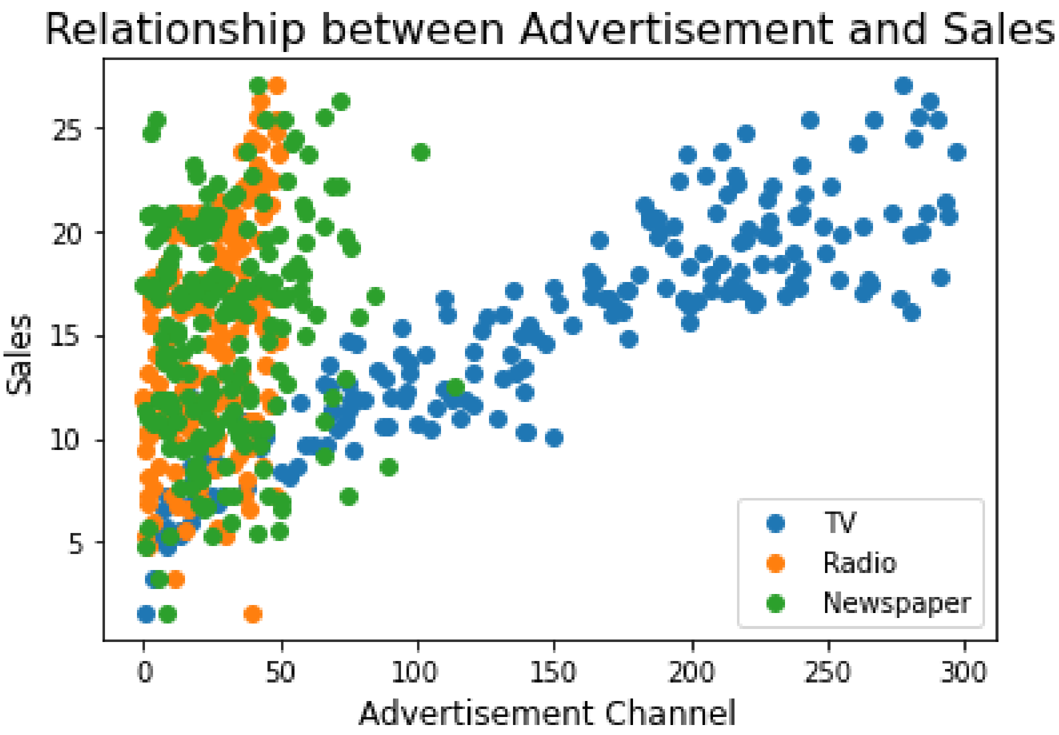
1. Finding Accuracy of MultiLinear Regression model
2. How to plot multiple linear reg model?

[ ]:
```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# import dataset
mlr = pd.read_csv("advertising.csv")
mlr.head()

# Splitting data into features and labels
```

```
x = mlr[['TV', 'Radio', 'Newspaper']]
y = mlr['Sales']

# Splitting data into train and test data
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=40)

# Fitting model on trained data

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model = model.fit(x_train, y_train)
model

# Predicting Results
print('Predicted Value is: ', model.predict([[230, 37.5, 70]]))

# Finding values of coefficents
print('values of coefficents are: ', model.coef_)

# Finding value of intercept
print('Value of intercept: ', model.intercept_)

print('Score for training data = ', model.score(x_train, y_train))
print('Score for testing data = ', model.score(x_test, y_test))

plt.scatter(mlr['TV'], y)
plt.scatter(mlr['Radio'], y)
plt.scatter(mlr['Newspaper'], y)
# legend, title and labels.
plt.legend(labels=['TV', 'Radio', 'Newspaper'])
plt.title('Relationship between Advertisement and Sales', size=16)
plt.xlabel('Advertisement Channel', size=12)
plt.ylabel('Sales', size=12)
```

```
Predicted Value is:  [21.12556982]
values of coefficents are:  [ 0.053514    0.11565901 -0.00517342]
Value of intercept:  4.842276855109864
Score for training data =  0.9149197048447624
Score for testing data =  0.8477283412895041
```

```
[ ]: Text(0, 0.5, 'Sales')
```



1.3 Decision Tree Classifiers on Biryani

[ ]:

```
# import dataset
import pandas as pd
df = pd.read_csv('mldata.csv')
df.head()

# Replacing 'Male' and 'Female' with '1' and '0'
df['gender'] = df['gender'].replace('Male', 1)
df['gender'] = df['gender'].replace('Female', 0)
df.head()

# Selection of input and output variable
x = df[['weight', 'gender']]
y = df['likeness']

# Machine learning algorithm
from sklearn.tree import DecisionTreeClassifier

# Create and fit model
model = DecisionTreeClassifier().fit(x,y)
model
```

```python
# Prediction
model.predict([[80, 1]])

# How to measure the accuracy of our model
# split data into test and train (80/20)
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)

# Create a model
model = DecisionTreeClassifier().fit(x_train, y_train)

# Prediction
predicted_values = model.predict(x_test)
print('Predicted values are: ', predicted_values)

# Checking Score
from sklearn.metrics import accuracy_score
score = accuracy_score(y_test, predicted_values)
print('Accuracy Score is: ', score)


# How to train and save your model

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import joblib

model = DecisionTreeClassifier().fit(x,y)

joblib.dump(model, 'foodie.joblib')


# Plotting
from sklearn import tree

model = DecisionTreeClassifier().fit(x,y)
tree.export_graphviz(model,
                    out_file = 'foodie.dot',
                    feature_names= ['age', 'gender'],
                    class_names= sorted(y.unique()),
                    label='all',
                    rounded=True,
                    filled=True)
```

```
Predicted values are:  ['Biryani' 'Biryani' 'Pakora' 'Biryani' 'Samosa'
'Biryani' 'Pakora'
 'Biryani' 'Biryani' 'Biryani' 'Samosa' 'Samosa' 'Samosa' 'Pakora'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Pakora' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Samosa'
 'Biryani' 'Samosa' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Samosa' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani']
Accuracy Score is:  0.6122448979591837
```

1.3.1 Assignment 2: instead of two use all available input variables to repeat above task?

[ ]:

```python
import pandas as pd
df = pd.read_csv('mldata.csv')
df.head()

df['gender'] = df['gender'].replace('Male', 1)
df['gender'] = df['gender'].replace('Female', 0)
df.head()

# Selection of input and output variable
x = df[['age','height','weight', 'gender']]
y = df['likeness']
x.head()

# Machine learning algorithm
from sklearn.tree import DecisionTreeClassifier

# Create and fit model
model = DecisionTreeClassifier().fit(x,y)
model

# Prediction
model.predict([[41, 165, 70, 1]])

# How to measure the accuracy of our model
# split data into test and train (80/20)
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=1)

# Create a model
model = DecisionTreeClassifier().fit(x_train, y_train)

# Prediction
predicted_values = model.predict(x_test)
predicted_values
```

```
# Checking Score
score = accuracy_score(y_test, predicted_values)
score

# Plotting
from sklearn import tree

model = DecisionTreeClassifier().fit(x,y)
tree.export_graphviz(model,
                    out_file = 'foodee.dot',
                    feature_names= ['age','height','weight', 'gender'],
                    class_names= sorted(y.unique()),
                    label='all',
                    rounded=True,
                    filled=True)
```

1.3.2 Assignmnet 3: Find accuracy score of 'iris' dataset on three different splitting ((10,90), (20,80), (30,70)) by using Decision Tree Classifier Model

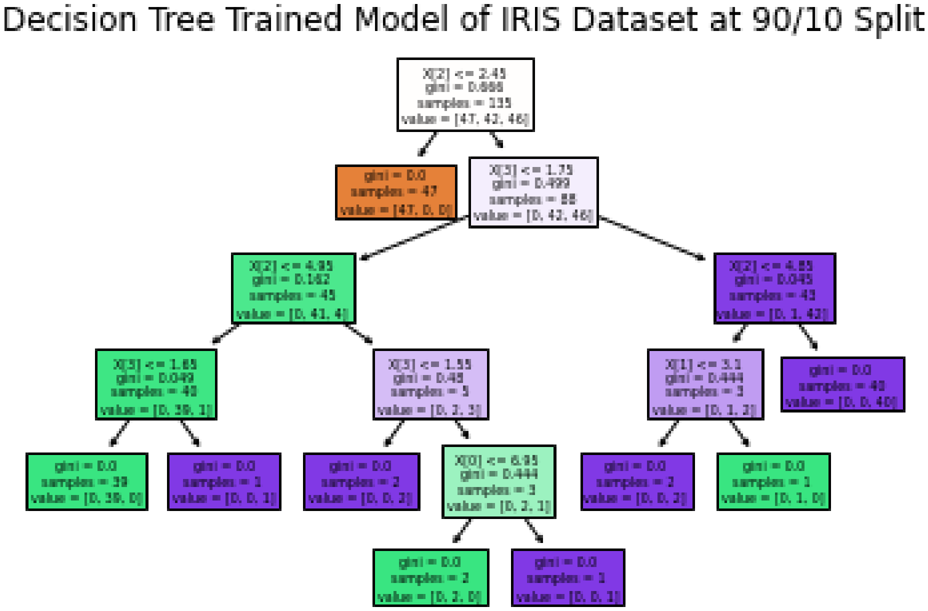
1. Solving 10/90 Split data

[ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

df = sns.load_dataset('iris')
df.head()

x = df.iloc[ : ,:-1]
y = df.iloc[ : ,-1:]

x.head()
y.head()

# 1. Splitting Data into Test and Train (90/10)
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1,␣
 ↪random_state=0)

# Create a model
model = DecisionTreeClassifier().fit(x_train, y_train)

plot_tree(model, filled=True)
```

```
plt.title('Decision Tree Trained Model of IRIS Dataset at 90/10 Split')

# Prediction
predicted_values = model.predict(x_test)
predicted_values

from sklearn.metrics import accuracy_score
# Checking Score
score = accuracy_score(y_test, predicted_values)
print('Accuracy Score of Model at 90/10 split is: ', score)

print('Prediction of unknown values: ', model.predict([[5, 3, 1.5, 0.2]]))
```

```
Accuracy Score of Model at 90/10 split is:  1.0
Prediction of unknown values:  ['setosa']
```



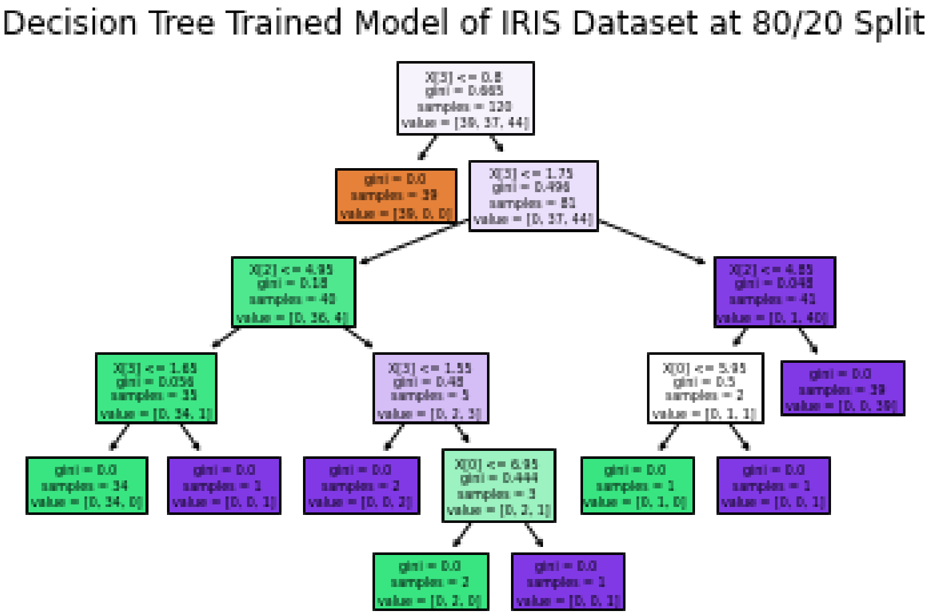
2. Solving 20/80 Split data

[ ]:

```
df = sns.load_dataset('iris')
df.head()

x = df.iloc[ : ,:-1]
y = df.iloc[ : ,-1:]

x.head()
y.head()
```

```
# 1. Splitting Data into Test and Train (80/20)
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)

# Create a model
model = DecisionTreeClassifier().fit(x_train, y_train)

plot_tree(model, filled=True)
plt.title('Decision Tree Trained Model of IRIS Dataset at 80/20 Split')

# Prediction
predicted_values = model.predict(x_test)
predicted_values

from sklearn.metrics import accuracy_score
# Checking Score
score = accuracy_score(y_test, predicted_values)
print('Accuracy Score of Model at 80/20 split is: ', score)

print('Prediction of unknown values: ', model.predict([[6, 4, 2.5, 0.1]]))
```

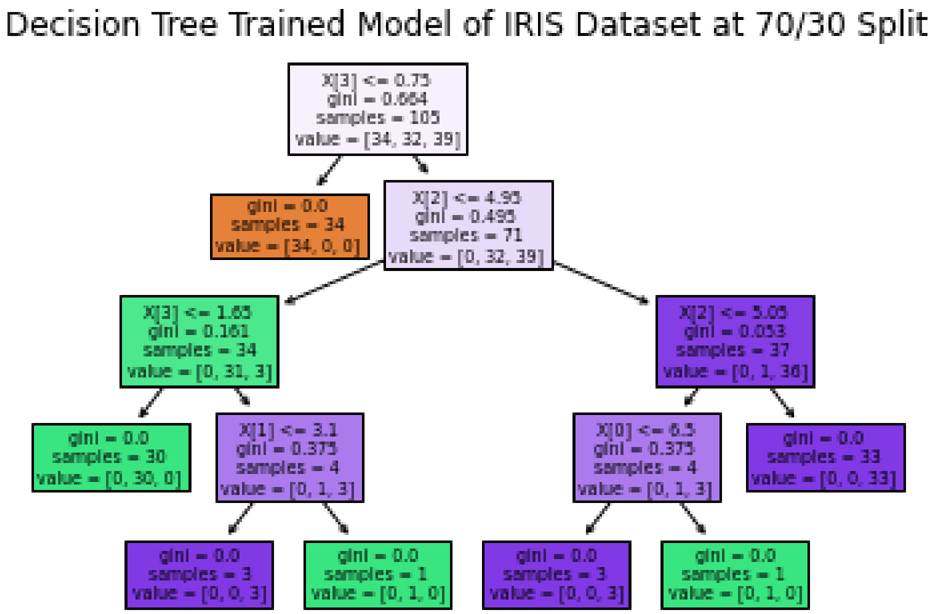
```
Accuracy Score of Model at 80/20 split is:  1.0
Prediction of unknown values:  ['setosa']
```



3. Solving 30/70 Split data

```
[ ]: df = sns.load_dataset('iris')
     df.head()

     x = df.iloc[ : ,:-1]
     y = df.iloc[ : ,-1:]

     x.head()
     y.head()

     # 1. Splitting Data into Test and Train (70/30)
     from sklearn.model_selection import train_test_split

     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
      ↪random_state=0)

     # Create a model
     model = DecisionTreeClassifier().fit(x_train, y_train)

     plot_tree(model, filled=True)
     plt.title('Decision Tree Trained Model of IRIS Dataset at 70/30 Split')

     # Prediction
     predicted_values = model.predict(x_test)
     predicted_values

     from sklearn.metrics import accuracy_score
     # Checking Score
     score = accuracy_score(y_test, predicted_values)
     print('Accuracy Score of Model at 70/30 split is: ', score)

     print('Prediction of unknown values: ', model.predict([[7, 4, 1.8, 0.1]]))
```

```
Accuracy Score of Model at 70/30 split is:  0.9777777777777777
Prediction of unknown values:  ['setosa']
```



1.4 Random Forest Model

- Random forest Classifier

[ ]:

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

df = sns.load_dataset('iris')
df.head()
```

[ ]:

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

[ ]:

```
x = df.iloc[ : ,:-1]
y = df.iloc[ : ,-1:]

from sklearn.ensemble import RandomForestClassifier

# Assignment: Do same for RandomForestRegressor
```

```
model = RandomForestClassifier(n_estimators=100)
model.fit(x,y)
model.predict([[5,4,2,6]])
```

```
C:\Users\Waleed\AppData\Local\Temp/ipykernel_16252/3020383993.py:9:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  model.fit(x,y)
```

```
[ ]: array(['setosa'], dtype=object)
```

```
[ ]: 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)

     predicted_values = model.predict(x_test)
     predicted_values
```

```
[ ]: array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
            'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
            'virginica', 'versicolor', 'versicolor', 'versicolor',
            'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
            'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
            'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor',
            'setosa'], dtype=object)
```

```
[ ]: # Accuracy Test
     score = model.score(x_test, y_test)
     print("The Accuracy Score is: ", score)
```

```
The Accuracy Score is:  1.0
```

```
[ ]: from sklearn.metrics import accuracy_score
     # Checking Score
     score = accuracy_score(y_test, predicted_values)
     print('Accuracy Score of Model at 80/20 split is: ', score)
```

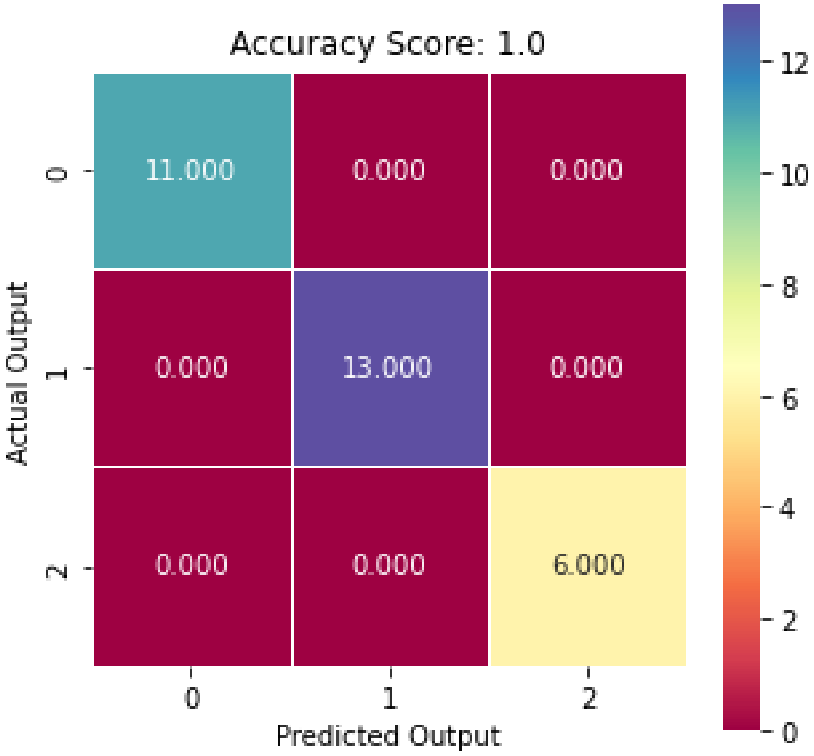
```
Accuracy Score of Model at 80/20 split is:  1.0
```

```
[ ]: from sklearn import metrics

     cm = metrics.confusion_matrix(y_test, predicted_values)
     cm
```

```
[ ]: array([[11,  0,  0],
            [ 0, 13,  0],
            [ 0,  0,  6]], dtype=int64)
```

[ ]:

```
import seaborn as sns
plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, fmt='.3f', linewidths=.5, square=True,␣
 ↪cmap='Spectral');
plt.ylabel('Actual Output');
plt.xlabel('Predicted Output');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 12);
```



1.4.1 Assignment 4: Use Random Forest Regressor

[ ]:

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

# Loading ml_salary Dataset
df = pd.read_csv('ml_data_salary.csv')
df.head()

x = df.iloc[ : ,:-1]
y = df.iloc[ : ,-1:]
```

```
x.head()

from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=10) # n_estimator: The number of␣
 ↪trees in the forest
model.fit(x,y)
model.predict([[30,77,1.1]])

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)
predicted_values = model.predict(x_test)
predicted_values

# Accuracy Test
score = model.score(x_test, y_test)
print("The Accuracy Score is: ", score)
```

```
The Accuracy Score is:  0.994593633784487
```

```
C:\Users\Waleed\AppData\Local\Temp/ipykernel_16252/3920625703.py:16:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  model.fit(x,y)
```

1.5 K-nearest neighbour

[ ]:

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

# Loading Dataset
df = pd.read_csv('mldata.csv')
df['gender'] = df['gender'].replace('Male', 1)
df['gender'] = df['gender'].replace('Female', 0)

# Selection of input and output variable
x = df[['weight', 'gender']]
y = df['likeness']

# model and prediction
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)

# Train model using the training sets
model.fit(x,y)
```

```
# Predict output
predicted = model.predict([[70,1]])
predicted
```

```
[ ]: array(['Biryani'], dtype=object)
```

[ ]:

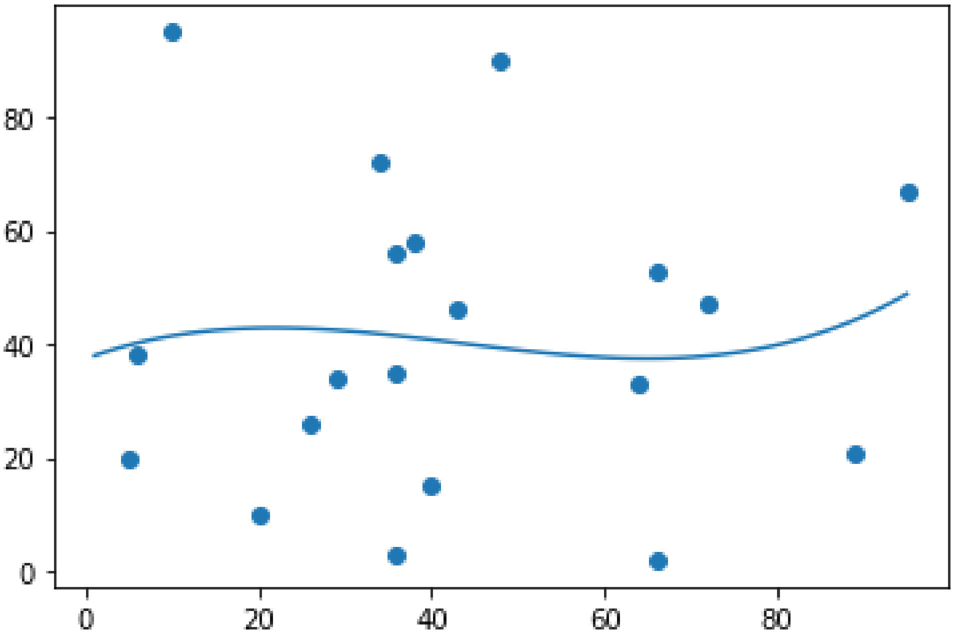
```
# Metrices for evaluation
# Split data

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

model = KNeighborsClassifier(n_neighbors=5).fit(x_train, y_train)

predicted_values = model.predict(x_test)
predicted_values

# Check score
# y_test = actual_values

score = accuracy_score(y_test, predicted_values)
print('The Accuracy score for our model is: ', score)
```

```
The Accuracy score for our model is:  0.7346938775510204
```

1.6 Polynomial Regression

[ ]:

```
# Bad fit
import numpy as np
import matplotlib.pyplot as plt

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

mymodel = np.poly1d(np.polyfit(x,y,3))
myline = np.linspace(1,95,100)

plt.scatter(x,y)
plt.plot(myline, mymodel(myline))
plt.show()
```



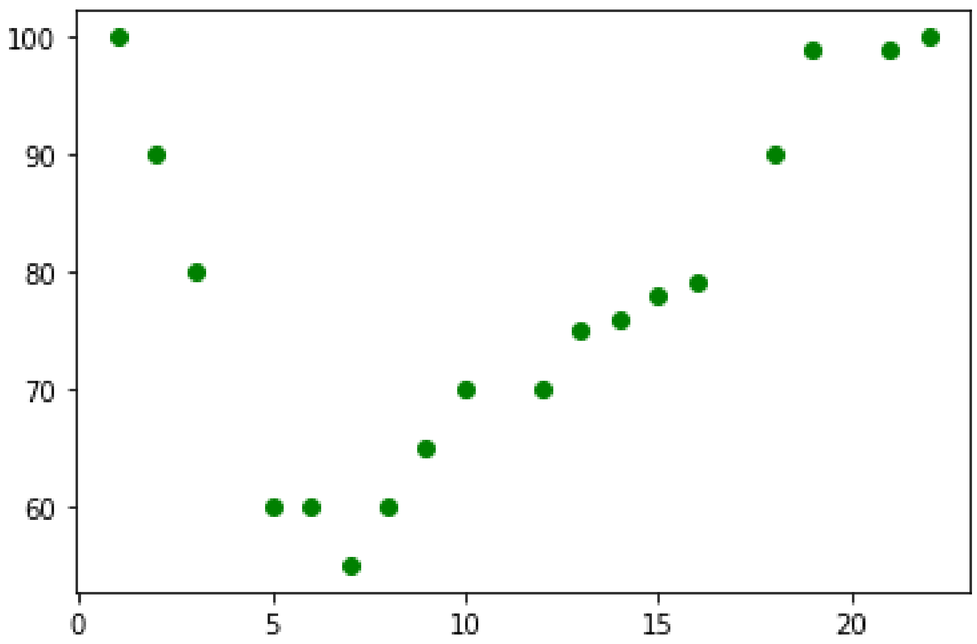
[ ]:

```
# R-squared for bad fit
from sklearn.metrics import r2_score

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

model = np.poly1d(np.polyfit(x,y,3))

print(r2_score(y, model(x)))
```

```
0.009952707566680652
```

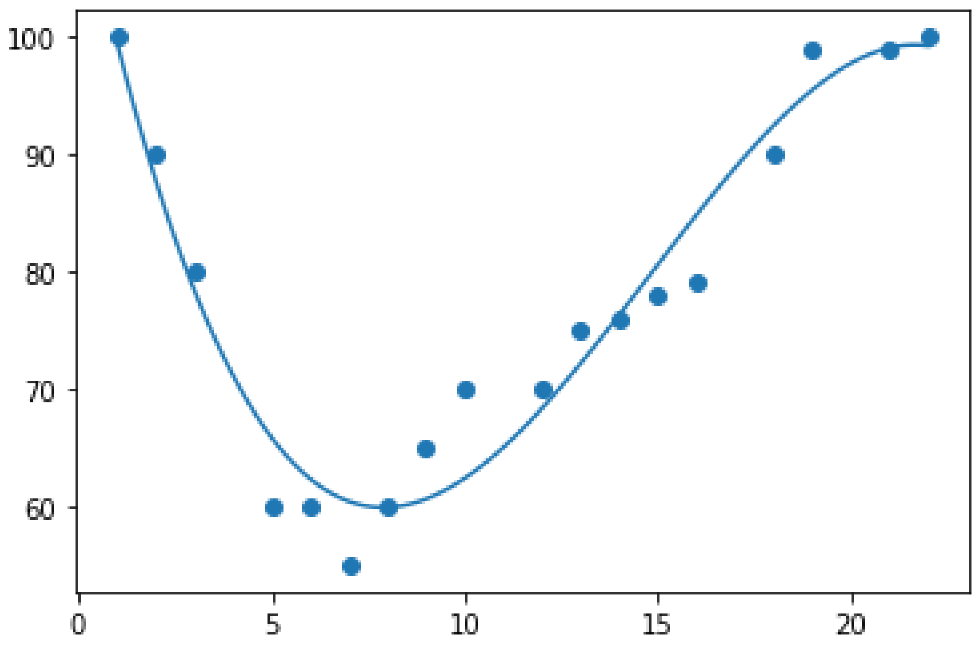
[ ]:

```
# Step 1: Data

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

plt.scatter(x, y, color='green')
plt.show()
```



[ ]:

```
# Step2: Draw a line

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = np.poly1d(np.polyfit(x,y,3))
myline = np.linspace(1,22,100)

plt.scatter(x,y)
plt.plot(myline, mymodel(myline))
plt.show()

print(r2_score(y, mymodel(x)))
```



```
0.9432150416451027
```

[ ]:

```
# Prediction
speed = mymodel(30)
print(speed)
```

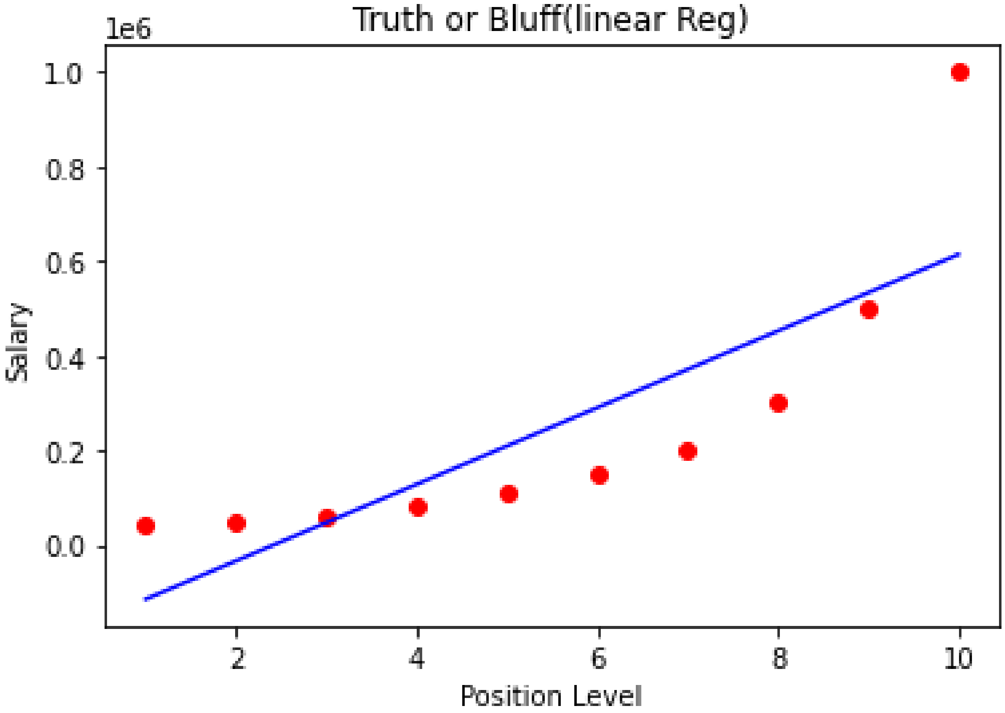
```
37.95389510181887
```

1.6.1 Hands on example

[ ]:

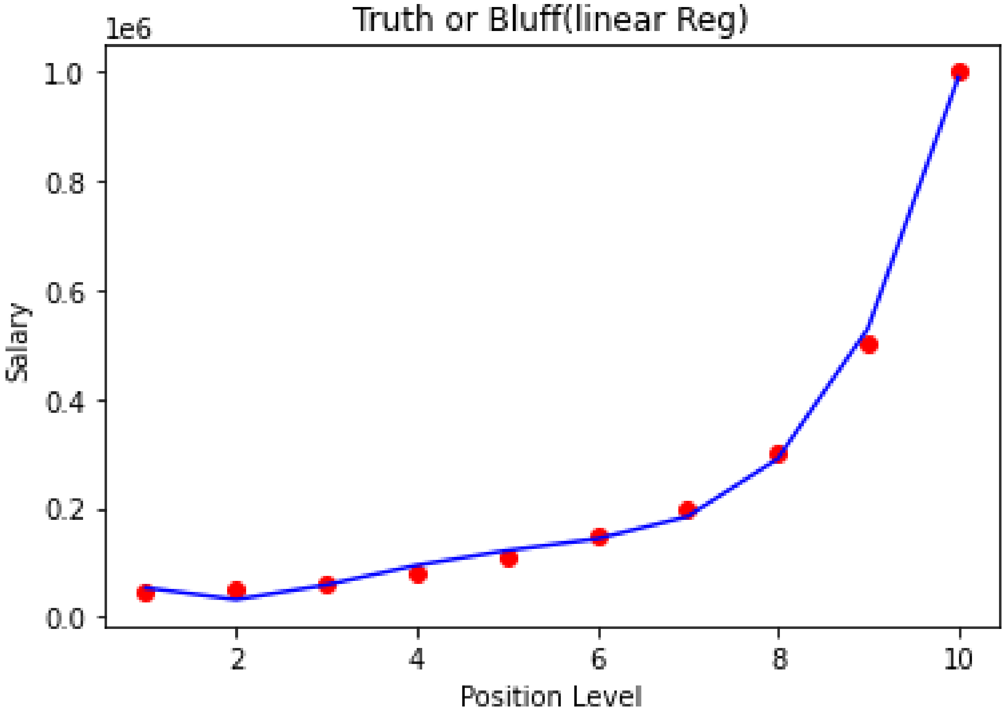
```
# Important example
import pandas as pd

dataset = pd.read_csv('https://s3.us-west-2.amazonaws.com/public.gamelab.fun/
 ↪dataset/position_salaries.csv')
dataset.head()

x = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values

# Splitting data
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)

# Fitting Linear Regression
```

```python
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression().fit(x,y)

# Visualize results
def viz_linear():
    plt.scatter(x,y, color='red')
    plt.plot(x, lin_reg.predict(x), color='blue')
    plt.title('Truth or Bluff(linear Reg)')
    plt.xlabel('Position Level')
    plt.ylabel('Salary')
    plt.show()
    return
viz_linear()
```



[ ]:

```python
# Fitting Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
x_poly = poly_reg.fit_transform(x)
pol_reg = LinearRegression().fit(x_poly,y)

# Visualize results
def viz_polynomial():
    plt.scatter(x,y, color='red')
    plt.plot(x, pol_reg.predict(poly_reg.fit_transform(x)), color='blue')
    plt.title('Truth or Bluff(linear Reg)')
    plt.xlabel('Position Level')
    plt.ylabel('Salary')
    plt.show()
    return
viz_polynomial()
```



[ ]:

```
# Prediction result with linearregression
pred_linear = lin_reg.predict([[11]])

# Prediction result with Polynomial regression
pred_polynomial = pol_reg.predict(poly_reg.fit_transform([[11]]))

print('Linear Regression Results= ', pred_linear)
print('Polynomial Regression Result= ', pred_polynomial)

print('Difference is = ', pred_polynomial - pred_linear)
```

```
Linear Regression Results=  [694333.33333333]
Polynomial Regression Result=  [1780833.33333322]
Difference is =  [1086499.99999989]
```

1.7 Naive Bayes

[ ]:

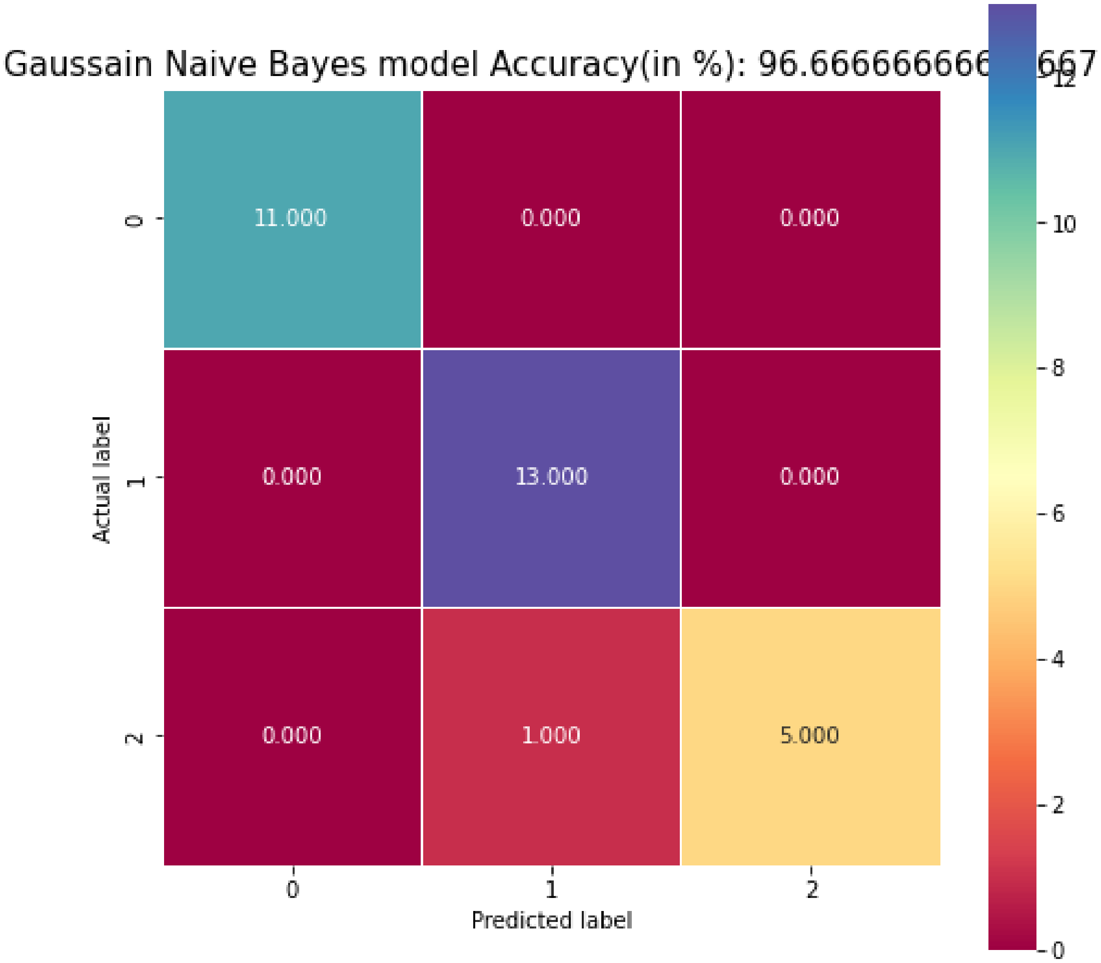
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# load dataset
phool = sns.load_dataset('iris')
phool.head()

# input and output
x = phool.iloc[:, :-1] # fearture
y = phool.iloc[:, -1:] # labels

from sklearn.naive_bayes import GaussianNB
model = GaussianNB().fit(x,y)
model

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)

# training model on training set
from sklearn.naive_bayes import GaussianNB
model = GaussianNB().fit(x_train, y_train)

# Making prediction
y_pred = model.predict(x_test)
y_pred

from sklearn import metrics
score = metrics.accuracy_score(y_test, y_pred)
print('Gaussain Naive Bayes Model Accuracy(in %): ', metrics.
 ↪accuracy_score(y_test, y_pred)*100)

# Confusion Matrix
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
cm

plt.figure(figsize=(8,8))
sns.heatmap(cm, annot=True, fmt='.3f', linewidths=.5, square=True, cmap=␣
 ↪'Spectral')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
all_sample_title = 'Gaussain Naive Bayes model Accuracy(in %): {0}'.
↪format(score*100)
plt.title(all_sample_title, size=15)
```

```
c:\Users\Waleed\anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
c:\Users\Waleed\anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

```
Gaussain Naive Bayes Model Accuracy(in %):  96.66666666666667
```

```
[ ]: Text(0.5, 1.0, 'Gaussain Naive Bayes model Accuracy(in %): 96.66666666666667')
```



1.7.1 Assignment 5: do same as above on different dataset

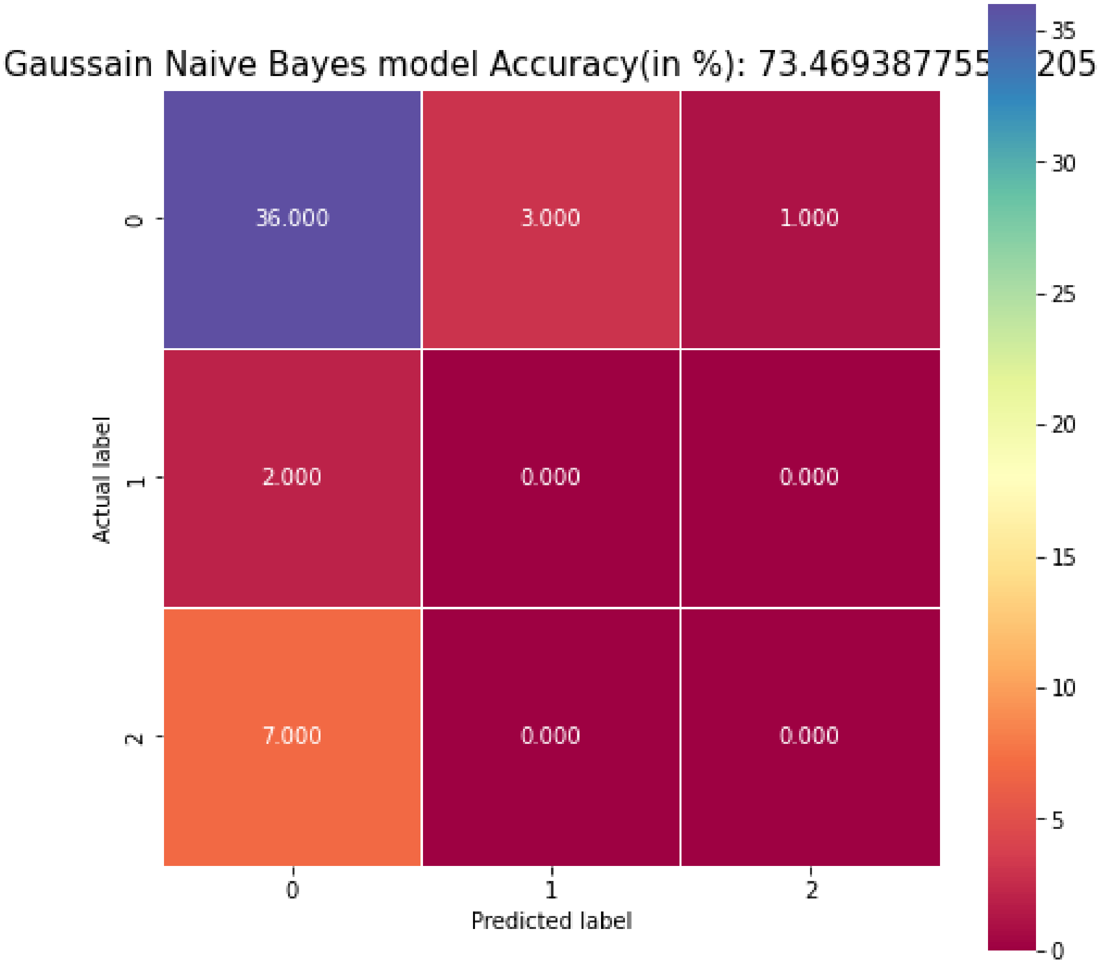
[ ]:

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

# Loading Dataset
df = pd.read_csv('mldata.csv')
df['gender'] = df['gender'].replace('Male', 1)
df['gender'] = df['gender'].replace('Female', 0)

# input and output
x = df.iloc[:, :-1] # fearture
y = df.iloc[:, -1:] # labels

from sklearn.naive_bayes import GaussianNB
model = GaussianNB().fit(x,y)
model

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=50)

# training model on training set
from sklearn.naive_bayes import GaussianNB
model = GaussianNB().fit(x_train, y_train)

# Making prediction
y_pred = model.predict(x_test)
y_pred

from sklearn import metrics
score = metrics.accuracy_score(y_test, y_pred)
print('Gaussain Naive Bayes Model Accuracy(in %): ', metrics.
 ↪accuracy_score(y_test, y_pred)*100)

# Confusion Matrix
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
cm

plt.figure(figsize=(8,8))
sns.heatmap(cm, annot=True, fmt='.3f', linewidths=.5, square=True, cmap=␣
 ↪'Spectral')
```

```
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Gaussain Naive Bayes model Accuracy(in %): {0}'.
↪format(score*100)
plt.title(all_sample_title, size=15)
```

```
c:\Users\Waleed\anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
c:\Users\Waleed\anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)

Gaussain Naive Bayes Model Accuracy(in %):  73.46938775510205
```

```
[ ]: Text(0.5, 1.0, 'Gaussain Naive Bayes model Accuracy(in %): 73.46938775510205')
```



1.8 SVM

[ ]:

```
# import scikit-learn dataset library
from sklearn import datasets
# load dataset
cancer = datasets.load_breast_cancer()

# print the names of the 30 features
print('Features: ', cancer.feature_names)

# print the label type of cancer('malignant' 'benign')
print('Labels: ', cancer.target_names)
```

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
```

[ ]:

```
# print data shape
cancer.data.shape
```

```
[ ]: (569, 30)
```

[ ]:

```
# print the cancer data features (top 5)
print(cancer.data[0:5])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```

[ ]:
```
# print cancer labels (0: malignant, 1: benign)
print(cancer.target)
```

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

[ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(cancer.data,cancer.target,␣
 ↪test_size=0.2, random_state=0)

# import svm model
from sklearn import svm

# create a svm classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

# Train model
clf.fit(x_train, y_train)

# Predict the response for test dataset
y_pred = clf.predict(x_test)
```

```
from sklearn import metrics
score = metrics.accuracy_score(y_test, y_pred)

print('Accuracy: ', score)

print('Precision: ', metrics.precision_score(y_test, y_pred))

print('Recall: ', metrics.recall_score(y_test, y_pred))
```

```
Accuracy:  0.956140350877193
Precision:  0.984375
Recall:  0.9402985074626866
```

[ ]:

```
# confusion matrix
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
cm
```

[ ]:

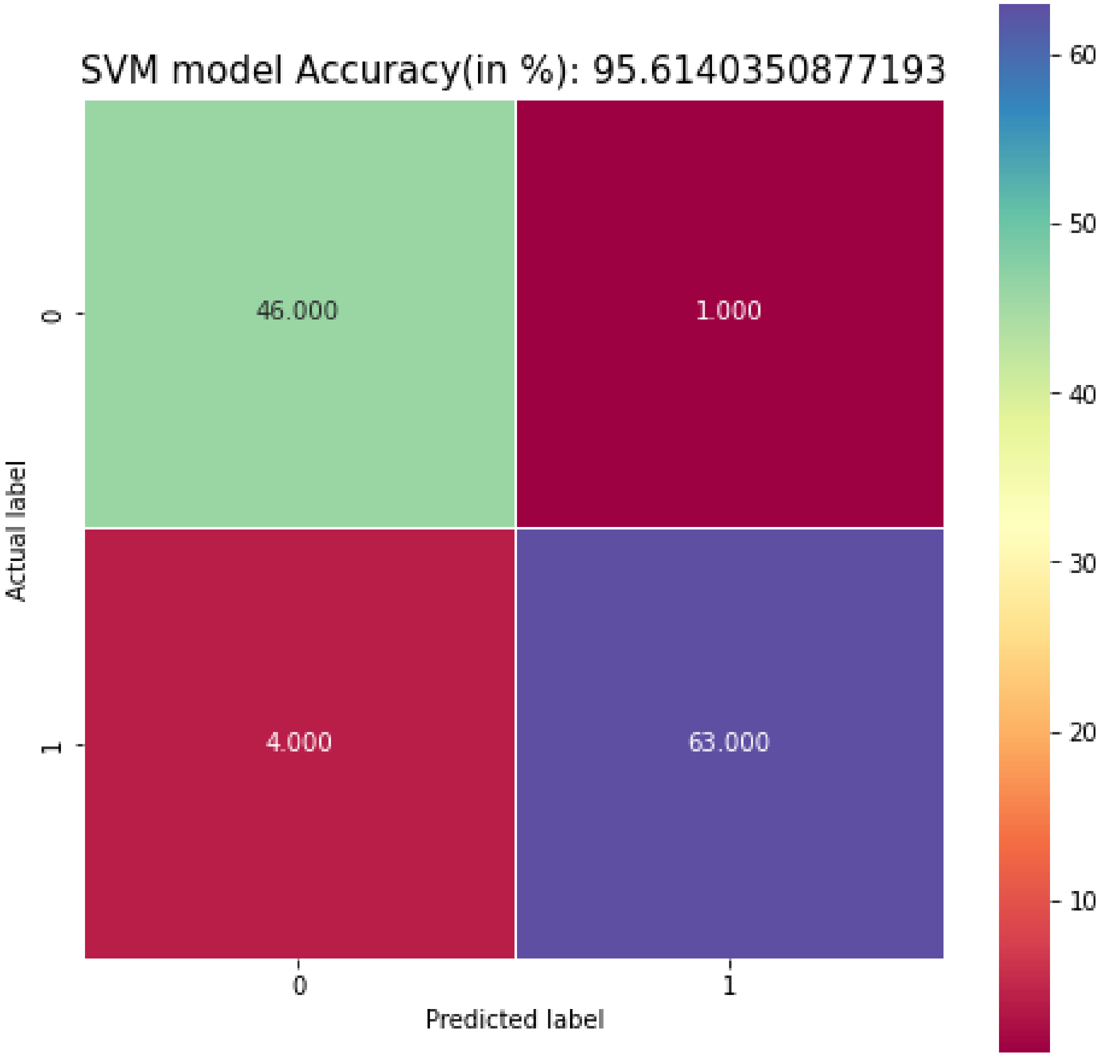
```
array([[46,  1],
       [ 4, 63]], dtype=int64)
```

[ ]:

```
plt.figure(figsize=(8,8))
sns.heatmap(cm, annot=True, fmt='.3f', linewidths=.5, square=True, cmap=␣
 ↪'Spectral')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'SVM model Accuracy(in %): {0}'.format(score*100)
plt.title(all_sample_title, size=15)
```

[ ]:

```
Text(0.5, 1.0, 'SVM model Accuracy(in %): 95.6140350877193')
```



1.9 Timestamp Assignments:

- @ 00:04:40: Machine Learning with Codanics is very easy
- @ 00:29:37: Data-Driven-Decision-Making-baba g k sath
- @ 00:34:45: Spelling mistake (MROE - MORE)
- @ 00:35:32: Types of machine learning
  1. Supervised ML (Task-Driven)
  2. Unsupervised/ Clustring ML (Data-Driven)
  3. SemiSupervised ML
  4. Reinforcement ML (Learn from errors)
- @ 00:39:07: Amber
- @ 00:55:48: Reinforcement learning example is self driving cars
- @ 01:05:07: 'x' should be independent variable and 'y' dependent
- @ 01:31:33:

  - Input variables:
    - Features
    - input data
    - independent variable
  - Output variables:
    - Prediction
    - Dependent variable
- @ 01:38:00: Just put double square brakets around
- @ 01:54:28: Least Square Method: A statistical method to find out the best fit for a set of data points by minimizing the sum of the offsets or residuals of points from the plotted curve
- @ 02:04:57: put double square brakets around. x = df[['weight', 'gender']]
- @ 02:07:24: yahan tak clear hai
- @ 02:50:30: Decision tree classifier is clear now
- @ 02:55:40: Random forest model with codenics
- @ 02:58:38: Training data change karny say decision tree change ho jata hai (sensitive)
- @ 03:13:44: khamoshi
- @ 03:17:55: yahan tak random forest clear hai
- @ 03:25:18: feature selection (scalling feature)
- @ 03:28:21: n_estimator: The number of trees in the forest
- @ 03:39:17:
  1. precision_score:
  2. recall_score:
  3. f1_score:
- @ 03:48:40: k-nearest neighbor is clear
- @ 03:52:24: b1 k sath x pe square aana hai
- @ 03:52:36: multiple linear regression bola
- @ 03:54:50: list hai
- @ 03:55:26: 3 means equation ke power 3 hai
- @ 03:58:37: 100 kiya hai
- @ 03:59:36: daal dekhnay gey
- @ 04:01:17: clear yahan tak
- @ 04:08:50: polynomial say zaida salary banti hai
- @ 04:09:27: numpy and sklearn libraries are used
- @ 04:13:19: probability % is 16.67
- @ 04:22:55: GaussainNB: it is a normal distribution designed by mean in the middle with optimally screwed data like a bell curve
- @ 04:32:00: uper +ve ho ga
- @ 04:37:54: parabola
- @ 04:54:31 Theory is clear for Support Vector Machine
- @ 04:58:55 Accuracy: it is simply a ratio of correctly predicted observation to the total observations. Precision: it is the ratio of correctly predicted positive observations to the total predicted positive observations.
- @ 05:13:21 K-Mediod Clustering
- @ 05:15:29 clustering with Codanics is very eas