AI_OUTPUTS_N>VIJAY

ex_8a:

In [5]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
# importing machine learning models for prediction
import xgboost as xgb
# importing bagging module
from sklearn.ensemble import BaggingRegressor
# loading Iris dataset
from sklearn.datasets import load_iris
iris = load_iris()

# getting target data from the dataset
target = iris.target
# getting train data from the dataset
train = pd.DataFrame(iris.data, columns=iris.feature_names)
# Splitting between train data into training and validation dataset
X_train, X_test, y_train, y_test = train_test_split(
train, target, test_size=0.20)
# initializing the bagging model using XGBoost as base model with defaul
model = BaggingRegressor(estimator=xgb.XGBRegressor())
# training model
model.fit(X_train, y_train)
# predicting the output on the test dataset
pred = model.predict(X_test)
# printing the mean squared error between real value and predicted value
print(mean_squared_error(y_test, pred))
```

```
0.03864110131248733
```

ex_8b:

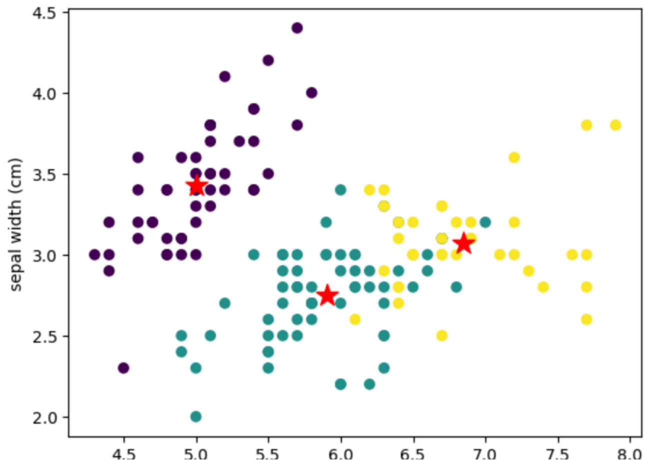
In [2]:
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# importing machine learning models for prediction
from sklearn.ensemble import GradientBoostingClassifier
# loading iris dataset
iris = load_iris()
# getting feature data from the iris dataset
features = iris.data
# getting target data from the iris dataset
target = iris.target

# Splitting between train data into training and validation dataset
X_train, X_test, y_train, y_test = train_test_split(features, target, te
# initializing the boosting module with default parameters
model = GradientBoostingClassifier()
# training the model on the train dataset
model.fit(X_train, y_train)
# predicting the output on the test dataset
pred_final = model.predict(X_test)
# printing the accuracy score between real value and predicted value
print(accuracy_score(y_test, pred_final))
```

```
0.9
```

ex_9:

```
In [2]: from sklearn.cluster import KMeans
        from sklearn.datasets import load_iris
        import matplotlib.pyplot as plt
        # Load the Iris dataset
        iris = load_iris()
        # Extract the data and target values
        X = iris.data
        y = iris.target
        # Create a KMeans object with 3 clusters
        kmeans = KMeans(n_clusters=3, n_init=10)
        # Fit the KMeans object to the data
        kmeans.fit(X)
        # Get the predicted cluster labels
        labels = kmeans.predict(X)
        # Plot the data points and centroids
        plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
        plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], marker='*', s=200, c='red')
        plt.xlabel(iris.feature_names[0])
        plt.ylabel(iris.feature_names[1])
        plt.show()
```



ex_10:

```
Iteration 1...
Iteration 2...
Iteration 3...
Iteration 4...
Iteration 5...
Iteration 6...
Iteration 7...
Iteration 8...
Iteration 9...
Iteration 10...
Learned Parameters:
CPT for Burglary:
[0.5 0.5]

CPT for Earthquake:
[0.5 0.5]

CPT for Alarm given Burglary and Earthquake:
[[[0.00000000e+00 1.00000000e+00]
  [9.66383974e-04 9.99033616e-01]]

 [[0.00000000e+00 1.00000000e+00]
  [6.40576290e-03 9.93594237e-01]]]

CPT for JohnCalls given Alarm:
[[0.06051854 0.93948146]
 [0.27711179 0.72288821]]

CPT for MaryCalls given Alarm:
[[0.06051854 0.93948146]
 [0.27711179 0.72288821]]
```

ex_11:

```
x = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0], [1], [1], [0]])
# Define the network model and its arguments.
# Set the number of neurons/nodes for each layer:
model = Sequential()
model.add(Dense(2, input_shape=(2,)))
model.add(Activation('sigmoid'))
model.add(Dense(1))
model.add(Activation('sigmoid'))
# Compile the model and calculate its accuracy:
model.compile(loss='mean_squared_error', optimizer='sgd', metrics=['accuracy']) # Print
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 2)                 6

 activation (Activation)     (None, 2)                 0

 dense_5 (Dense)             (None, 1)                 3

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
```

Ex_12:

```
model.evaluate(x_test, y_test)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/5
1875/1875 [==============================] - 19s 9ms/step - loss: 0.2940 - accuracy: 0.9140
Epoch 2/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1421 - accuracy: 0.9575
Epoch 3/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.1086 - accuracy: 0.9672
Epoch 4/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0907 - accuracy: 0.9715
Epoch 5/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0752 - accuracy: 0.9762
313/313 [==============================] - 1s 2ms/step - loss: 0.0755 - accuracy: 0.9761
[0.07545579224824905, 0.9761000275611877]
```