assignment_10

December 11, 2023

1 Assignment 10

[1]:
```
# core
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# ml
from sklearn import datasets as ds
from sklearn import linear_model as lm
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.model_selection import train_test_split as tts

#plotly or other graphing library
```

[51]:
```
# Load datasets here once and assign to variables iris and boston
iris = ds.load_iris()
```

Q1

Data set: Iris

- Return the first 5 rows of the data including the feature names as column headings in a DataFrame and a separate Python list containing target names

[55]:
```
# Create iris dataframe and setting X-variable
X = pd.DataFrame(iris['data'])
# Using feature_names for column names of X
X.columns = iris['feature_names']
print("Iris Features:")
display(X.head())

# Assigning y-variable as target names
y = iris['target']
print("Iris Target Names:")
print(iris['target_names'])
```

```
Iris Features:
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
```

```
Iris Target Names:
['setosa' 'versicolor' 'virginica']
```

Q2

Data set: Iris

- Fit the Iris dataset into a kNN model with neighbors=5 and predict the category of observations passed in argument new_observations. Return back the target names of each prediction (and not their encoded values, i.e. return setosa instead of 0).

[59]:
```
knn = KNN(n_neighbors=5)
knn.fit(X, y)
```

[59]:
```
KNeighborsClassifier()
```

[61]:
```
# Predict new observations
new_observations = knn.predict(X)
print("New Observations:")
print(iris['target_names'][new_observations])
```

```
New Observations:
['setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'versicolor' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'versicolor' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
```

```
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica']
```

Q3 15 pts

Data set: Iris

- Split the Iris dataset into a train / test model with the split ratio between the two established by the function parameter split.
- Fit KNN with the training data with number of neighbors equal to the function parameter neighbors
- Generate and return back an accuracy score using the test data that was split out

[63]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=42, stratify=y)
knn = KNN(n_neighbors=5)
knn.fit(X_train, y_train)
accuracy = knn.score(X_test, y_test)
print("Accuracy:", accuracy)
Accuracy: 0.9736842105263158
```

```
Accuracy: 0.9736842105263158
```

Q4

Data set: Iris

- Generate an overfitting / underfitting curve of kNN each of the testing and training accuracy performance scores series for a range of neighbor (k) values from 1 to 30 and plot the curves (number of neighbors is x-axis, performance score is y-axis on the chart).

[65]:
```
neighbors = np.arange(1, 31)
train_accuracies = {}
test_accuracies = {}

for neighbor in neighbors:

        # Set up a KNN Classifier
        knn = KNN(n_neighbors=neighbor)

        # Fit the model
        knn.fit(X_train, y_train)

        # Compute accuracy
        train_accuracies[neighbor] = knn.score(X_train, y_train)
        test_accuracies[neighbor] = knn.score(X_test, y_test)
print(neighbors, '\n', train_accuracies, '\n', test_accuracies)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30]
 {1: 1.0, 2: 0.9821428571428571, 3: 0.9553571428571429, 4: 0.9642857142857143,
5: 0.9732142857142857, 6: 0.9732142857142857, 7: 0.9732142857142857, 8:
0.9732142857142857, 9: 0.9732142857142857, 10: 0.9732142857142857, 11:
0.9732142857142857, 12: 0.9642857142857143, 13: 0.9821428571428571, 14:
0.9821428571428571, 15: 0.9821428571428571, 16: 0.9732142857142857, 17:
0.9821428571428571, 18: 0.9821428571428571, 19: 0.9821428571428571, 20:
0.9821428571428571, 21: 0.9821428571428571, 22: 0.9642857142857143, 23:
0.9732142857142857, 24: 0.9642857142857143, 25: 0.9642857142857143, 26:
0.9553571428571429, 27: 0.9553571428571429, 28: 0.9642857142857143, 29:
0.9642857142857143, 30: 0.9642857142857143}
 {1: 0.9473684210526315, 2: 0.9210526315789473, 3: 0.9736842105263158, 4:
0.9736842105263158, 5: 0.9736842105263158, 6: 0.9473684210526315, 7:
0.9473684210526315, 8: 0.9473684210526315, 9: 0.9736842105263158, 10:
0.9736842105263158, 11: 0.9736842105263158, 12: 0.9736842105263158, 13:
0.9473684210526315, 14: 0.9473684210526315, 15: 0.9473684210526315, 16:
0.9473684210526315, 17: 0.9473684210526315, 18: 0.9473684210526315, 19:
0.9473684210526315, 20: 0.9473684210526315, 21: 0.9473684210526315, 22:
0.9473684210526315, 23: 0.9473684210526315, 24: 0.9210526315789473, 25:
0.9473684210526315, 26: 0.9210526315789473, 27: 0.9210526315789473, 28:
0.9210526315789473, 29: 0.9473684210526315, 30: 0.9736842105263158}
```

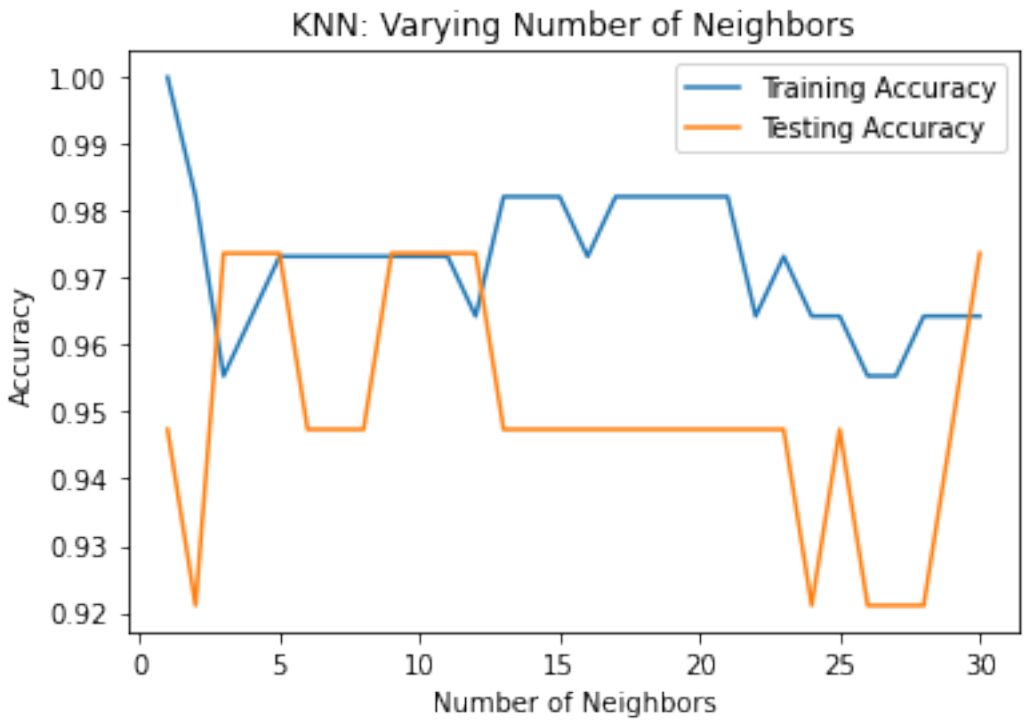
[67]:
```
plt.title("KNN: Varying Number of Neighbors")

# Plot training accuracies
plt.plot(neighbors, train_accuracies.values(), label="Training Accuracy")

# Plot test accuracies
plt.plot(neighbors, test_accuracies.values(), label="Testing Accuracy")

plt.legend()
plt.xlabel("Number of Neighbors")
plt.ylabel("Accuracy")

# Display the plot
plt.show()
```



Q5 10 pts

Data set: Boston

- Load sklearn's Boston data into a DataFrame (only the data and feature_name as column names)
- Load sklearn's Boston target values into a separate DataFrame
- Return back the average of AGE, average of the target (median value of homes or MEDV), and the target as NumPy values

[ ]:

Q6

Data set: Boston

- In the Boston dataset, the feature PTRATIO refers to pupil teacher ratio.
- Using a matplotlib scatter plot, plot MEDV median value of homes as y-axis and PTRATIO as x-axis
- Return back PTRATIO as a NumPy array

[ ]:

Q7

Data set: Boston

- Create a regression model for MEDV / PTRATIO and display a chart showing the regression line using matplotlib
- Use np.linspace() to generate prediction X values from min to max PTRATIO
- Return back the regression prediction space and regression predicted values
- Make sure to labels axes appropriately

[ ]: