assignment_10

April 12, 2025

1 Assignment 9

1.0.1 John Ferrara

[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
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

#plotly or other graphing library
import plotly.express as px
```

[2]:
```
# Load datasets here once and assign to variables iris and boston
from sklearn.datasets import load_iris#, load_boston
iris = load_iris()
#boston = load_boston()
## Boston data set removed from sklearn in post 1.2 versions
```

1.0.2 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

[3]:
```
# Seeing Iris Contents
[k for k,v in iris.items()]
```

[3]:
```
['data',
 'target',
 'frame',
 'target_names',
 'DESCR',
 'feature_names',
 'filename',
 'data_module']
```

[4]:

```
df = pd.DataFrame(iris.data, columns=iris.feature_names)
#Takign a look at IRIS in df
print(df.head(n=5))
print(df.shape) #150 rows
print('---------')
target_list = list(iris.target_names)
print("Columns: ", target_list)
```

```
   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
(150, 4)
---------
Columns:  ['setosa', 'versicolor', 'virginica']
```

1.0.3 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).

[6]:

```
## Need to add the target information to my df
#print(iris.target)
# Seesm to be index of list of targ. names; this is the"encoded vals"
## Cofirming len
print(len(iris.target))
df['target'] = iris.target
df['target_name']=pd.Series([iris.target_names[i] for i in iris.target])
#Checkign results
df.head()
#Setting vars
x = df[[i for i in df.columns if i not in ["target","target_name"]]]
y = df['target']
## Splitting btwn  train and test
x_train, x_test, y_train, y_test = tts(x, y, test_size=0.3, random_state=100,␣
 ↪shuffle=True)


# KNN start
```

```
knn = KNN(n_neighbors=5)
knn.fit(x_train, y_train)
```

```
150
```

[6]:
```
KNeighborsClassifier()
```

Q3

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

[7]:
```
## Did the first point above with the tts function
## Ran the fit for the KNN model above as well.

predictions = knn.predict(x_test)
print("Raw Predicitons: ", [iris.target_names[i] for i in predictions])

print('Classification Accuracy Score:')
print(round(accuracy_score(y_test, predictions)*100,2)," percent accurate.")
```

```
Raw Predicitons:  ['virginica', 'setosa', 'virginica', 'setosa', 'virginica',
'virginica', 'setosa', 'setosa', 'virginica', 'setosa', 'setosa', 'virginica',
'setosa', 'setosa', 'virginica', 'versicolor', 'versicolor', 'versicolor',
'virginica', 'virginica', 'virginica', 'setosa', 'virginica', 'setosa',
'versicolor', 'virginica', 'versicolor', 'setosa', 'versicolor', 'virginica',
'versicolor', 'versicolor', 'versicolor', 'setosa', 'setosa', 'versicolor',
'setosa', 'versicolor', 'virginica', 'virginica', 'setosa', 'versicolor',
'virginica', 'virginica', 'setosa']
Classification Accuracy Score:
97.78  percent accurate.
```

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).

[11]:
```
## Creating a fro loop for range 1, 30
accuracy_scores_list = []
for k in range(1, 31):
    knn = KNN(n_neighbors=k)
    knn.fit(x_train, y_train)
    predictions = knn.predict(x_test)
    accuracy_scores_list.append(round(accuracy_score(y_test,␣
 ↪predictions)*100,2))
```

[17]:

```
## Plotting using plotly
x_list = [k for k in range(1, 31)]
y_list = accuracy_scores_list
results = pd.DataFrame({"K Values":x_list,"Accuracy Scores (%)":y_list})
fig = px.line(results, x='K Values', y='Accuracy Scores (%)',
              title='KNN Accuracy vs Number of Neighbors (1-30)')

fig.show()
```

[ ]:

[ ]: