DS3000.A9.Shreyas.Shukla

October 28, 2024

1 Module 09: *k*-Nearest Neighbors

Student Name: Shreyas Shukla

Date: 28 Oct 2024

1.1 Overview

The k-Nearest neighbor method is a type of instance based learning that uses existing data, with known labels, to predict the label for a new observation. The purpose of this exercise is to demonstrate the use of k-NN for classification using the iris dataset.

1.2 Instructions

Use the notebook: `k-NN.ipynb` to perform the following: Question 1. Load and Inspect the data

Question 2. Perform the k-nn analysis using different values of k. Recommend which value is more appropriate and justify your response.

Question 3. Evaluate the prediction accuracy of the k-nn model.

Question 4. Explore different metrics and data partitioning strategies.

1.2.1 Submission Instructions

The `ipynb` format stores outputs from the last time you ran the notebook. (When you open a notebook it has the figures and outputs of the last time you ran it too). To ensure that your submitted `ipynb` file represents your latest code, make sure to give a fresh run `Kernel > Restart & Run All` just before uploading the `ipynb` file.

1.2.2 Academic Integrity

Writing your homework is an individual effort. You may discuss general python problems with other students but under no circumstances should you observe another student's code which was written for this assignment, from this year or past years. Pop into office hours or DM us in MS Teams if you have a specific question about your work or if you would like another pair of eyes or talk through your code.

Don't forget to cite websites which helped you solve a problem in a unique way. You can do this in markdown near the code or with a simple one-line comment. You do not need to cite the official python documentation.

Documentation / style counts for credit Please refer to the Pep-8 style, to improve the readability and consistency of your Python code. For more information, read the following article How to Write Beautiful Python Code With PEP 8 or ask your TA's for tips.

NOTE: Write python expressions to answer ALL questions below and ensure that you use the `print()` function to display the output. Each question should be answered in a new code cell. For example, your solution for question 1.1 should be in a different code cell from your solution for question 1.2.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

1.3 Question 1: Inspect the data

In this question you will need to load the data (this was performed for you), inspect the distribution and pairwise relationships between the features.

[4]:
```
#load the data from sklearn.datasets
data     = load_iris()

#divide the data into the input 'X' and the labels 'y'
X        = data['data'] #the observations
y        = data['target'] #the label
```

[5]:
```
#load the data in a pandas dataframe
df          = pd.DataFrame(X, columns=['sepal length', 'sepal width', 'petal␣
↪length', 'petal width'])
df['class'] = [data['target_names'][idx] for idx in y]
```

1.3.1 Question 1.1 (5 pts)

Obtain a statistical summary of the iris flowers sepal and petal features. Use the statistical summary, explain your observation about the iris flowers. Note: You can use the `describe()` function to display the statistical summary.

[6]:
```
# Obtain a statistical summary of the iris flowers sepal and petal features
summary = df.describe()
print(summary)
```

```
       sepal length  sepal width  petal length  petal width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
```

```
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

Explain ^

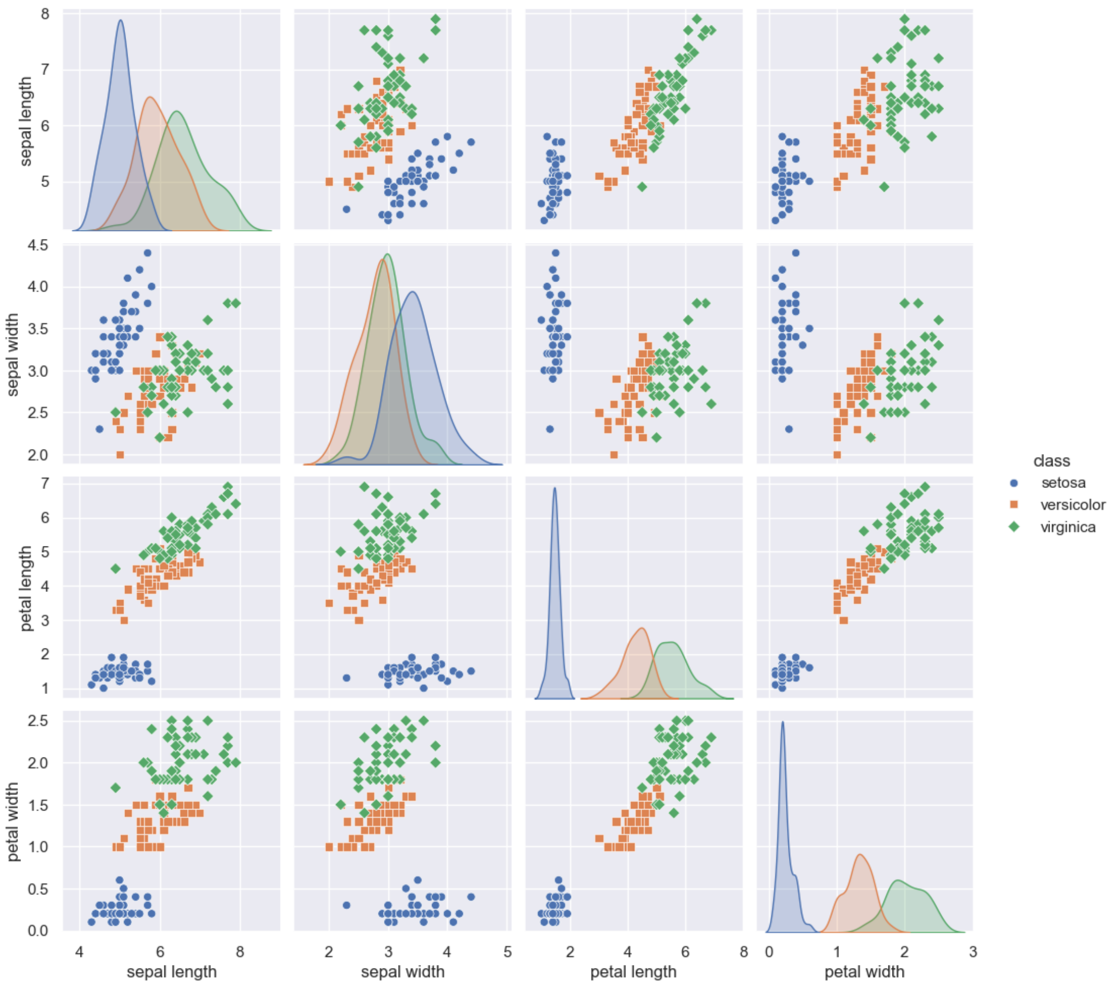
1.3.2 Question 1.2 (10 pts)

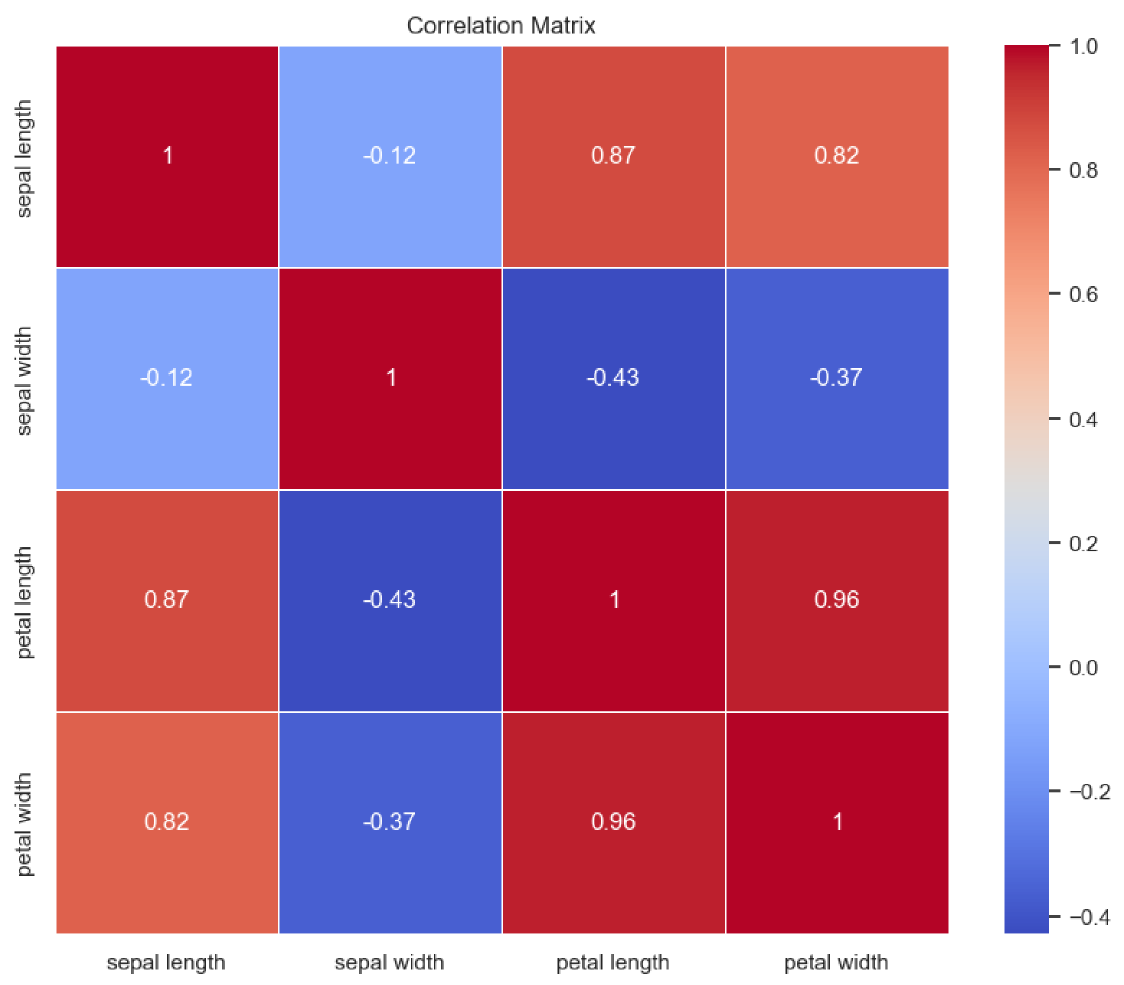
Visualize and discuss the **distribution of each feature** and the **pairwise relationships between features**. Note: Ensure that you discuss the strength of the **correlation** between the features.

- Tip 1: the seaborn library has a pairplot visualization which is useful to display both the distribution of each feature and pairwise relationships.
- Tip 2: another useful visualization is called a correlation matrix. It can be used to visually display the strength of the correlation.

[8]:
```python
# Visualize the distribution of each feature and the pairwise relationships␣
↪between features
sns.pairplot(df, hue='class', markers=["o", "s", "D"])
plt.show()

# Create a correlation matrix excluding the 'class' column
correlation_matrix = df.drop(columns=['class']).corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```


sepal length
sepal width
petal length
petal width
class
setosa
versicolor
virginica



Explain ⌒

1.4 Question 2: Choose *K* (25 pts)

Using the cell below, build a model to predict the target in the iris dataset using the k-NN algorithm. The program should try at least 20 different values of k and evaluate the accuracy of the predictions. After which, display a line graph showing the accuracy for each value of k. Evaluate the results from the graph and recommend the best value of *k*.

`Note:` If multiple values of *k* are found to be suitable, weigh the pros and cons of choosing a large versus a small value of *k*.

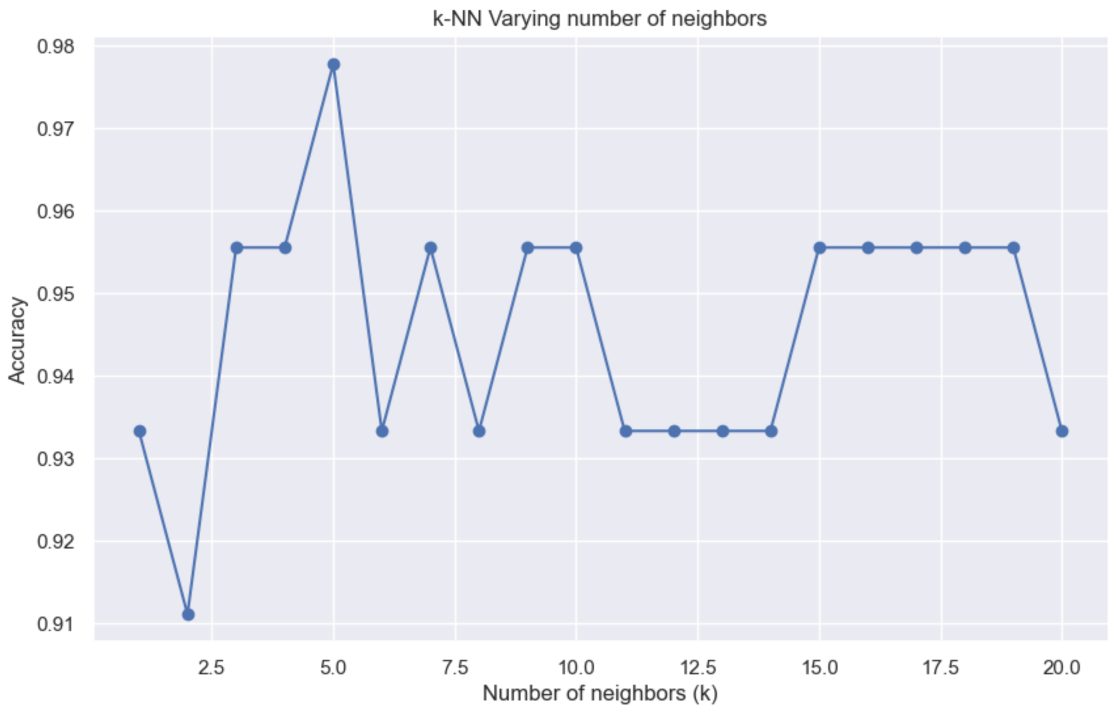
[9]:
```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

```
#build the k-nn model, experiment with different values of k and plot the␣
↪results
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42, stratify=y)

# Build the k-nn model, experiment with different values of k and plot the␣
↪results
k_values = range(1, 21)
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracies.append(accuracy_score(y_test, y_pred))

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(k_values, accuracies, marker='o')
plt.title('k-NN Varying number of neighbors')
plt.xlabel('Number of neighbors (k)')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```



⌢ Explain

1.5 Question 3: Evaluate the Results (5 pts)

Display the classification report for your recommended value of k. Evaluate the report and explain the results.

```
[ ]:
```

```
[ ]:
```

1.6 Question 4: Model Evaluation (5 pts)

Answer the following question using a markdown cell.

1. Research the difference between accuracy, precision, and recall. For each metric, provide an example that explains when you would choose one metric over the others.
2. In the lecture we discussed one method to partition your dataset using a train-test split. Identify another approach that can be used to partition your dataset and explain how it works.

'Note": you do not need to implement code for questions 4.1 and 4.2

1.7 Resources / References

Share any resources that were helpful in your response for question 4.