Project #1

CIS 2541 - Prof. John P. Baugh – Oakland Community College – OR

Objectives

- To practice and learn more about supervised learning
  - In particular, classification and classifiers
- To apply the fundamental principles and steps involved with making predictions using machine learning

Instructions

Overview of Technologies

We learned about some fundamental libraries involved with machine learning and related activities (E.g., NumPy, Pandas, Scikit-Learn, Matplotlib) and also learned about the fundamentals of **predicting categories** using **classification**. We used models such as SGDClassifier and analyzed the MNIST data set, containing handwriting samples. We covered how to use **binary classifiers** (that classify using two target classes) and also **multiclass classifiers** (that can use more than two target classes).

For this project, you'll use **kNN** (**k-Nearest Neighbors**) and **Naïve Bayes** classifiers for a multiclass classification task: namely, identifying subtypes of irises (an iris is a type of flower).

The kNN algorithm classifies an instance based on the ***closest*** of a given instance's neighbors. That is in turn, based on the shortest ***distance*** between a given instance and its neighbors. The other, **Naïve Bayes** is based on probabilities of each class given the feature values and assumes that features are independent.

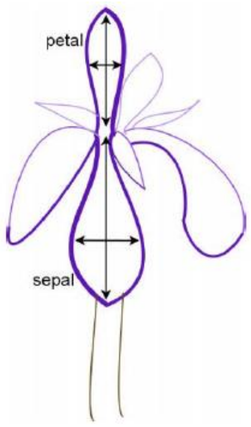
Before beginning the actual project, I do recommend reading up on kNN and Naïve Bayes a bit. Consider using the following resources:

- kNN with Scikit-Learn:
  - https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn
- Naïve Bayes with Scikit-Learn:
  - https://www.datacamp.com/tutorial/naive-bayes-scikit-learn

Overview of Dataset (The Iris Dataset)

The Iris dataset is a classic and widely-used dataset in the field of machine learning and statistics. It was first introduced by the British biologist and statistician Ronald A. Fisher in 1936. The dataset is often used for testing and demonstrating various machine learning algorithms, particularly classification techniques.

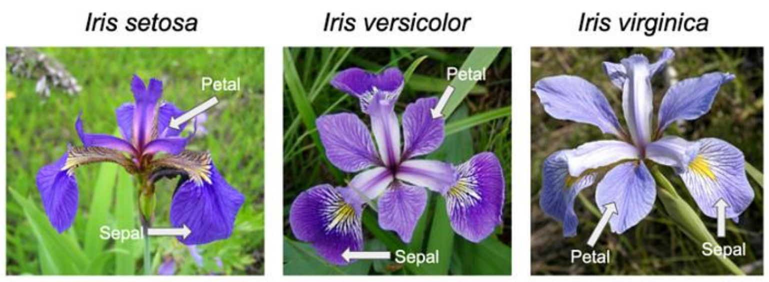
The dataset contains 150 samples of Iris flowers, with each sample having four features:



- Sepal length (cm)
- Sepal width (cm)
- Petal length (cm)
- Petal width (cm)

These features are numerical measurements that describe the physical characteristics of the Iris flowers. The dataset is divided into three classes, each representing a different species of Iris flower:

- ***Iris setosa***
  - Generally shorter and wider sepals compared to other species
  - Significantly shorter and narrower petals compared to other species
- ***Iris versicolor***
  - Generally very intermediate length and width of sepals
  - Longer and wider petals than *Iris setosa* but shorter and narrower petals than *Iris virginica*
- ***Iris virginica***
  - Generally the longest sepal of the three species, but a narrower sepal width than *Iris setosa*, but similar sepal width to *Iris versicolor*
  - The longest and widest petals of the three species



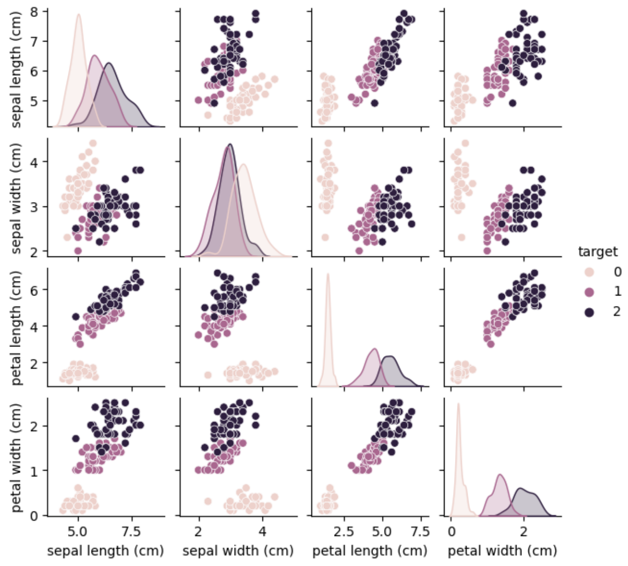
Each class contains 50 samples within the dataset.

What you should do

As a very general summary, you will be using the iris dataset, which can be obtained directly from the sklearn datasets, similarly to how MNIST was.

1. Load the Iris dataset
    a. Hint: datasets.load_iris()
    b. Obtain a Pandas DataFrame from the iris data portion, and use columns=iris.feature_names as the second parameter to the DataFrame constructor
    c. Display the first and last few rows of the dataset

2. Visualize the Data
    a. You may use Matplotlib, or look into using Seaborn
    b. Hint: With Seaborn, the **pairplot** function may come in handly
    c. Seaborn produced the following with the pairplot function:



3. Split the Data into Training and Testing Sets
    a. Hint: train_test_split is a handy function from sklearn.model_selection
    b. **Print out** the train features shape, and the test features shape

4. Build and Evaluate the kNN model
    a. Hints:
        i. KNeighborsClassifier with 3 neighbors as the hyperparameter should be good
        ii. GaussianNB is a good Naïve Bayes model to use
    b. Make sure to **fit** the model to the **training data**, and then **predict** on the **test data**
    c. You can use the **f1 score** as the metric to evaluate predictions

5. Build and Evaluate the Naïve Bayes Model
    a. Again, make sure to **fit** the model to the **training data** and to **predict** on the **test data**
    b. You can use the **f1 score** as the metric to evaluate predictions

6. Compare the Model Performance using the F1 Score

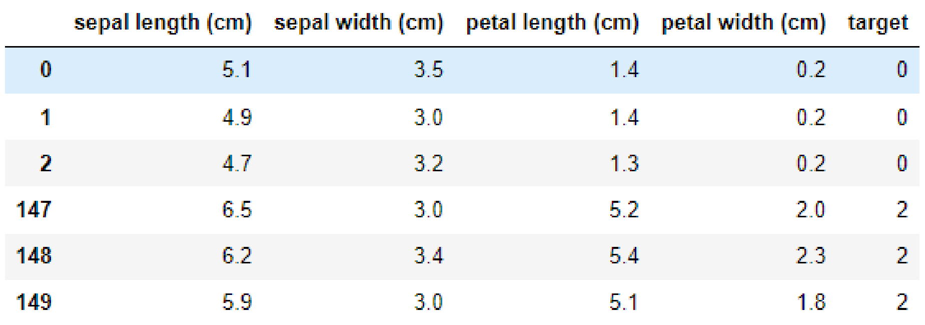
Handy code / module hints

- `from sklearn import datasets`
- `from pandas as pd`
- `import matplotlib.pyplot as plt`
- `import numpy as np`
- `from sklearn.model_selection import train_test_split`
- `from sklearn import neighbors, metrics`
- `from sklearn import naive_bayes`

Sample Outputs of some of the tasks

These are examples of the type of outputs you might expect from many of the tasks.

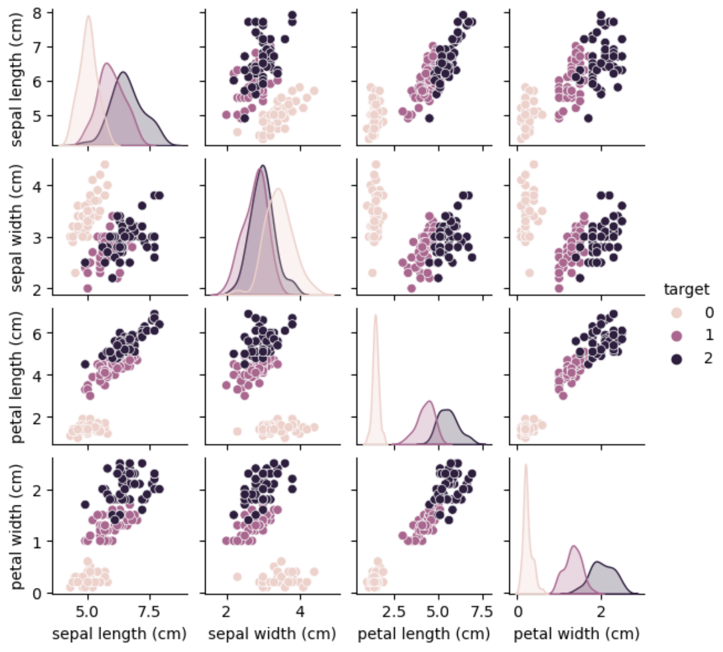
Displaying the first and last few rows of the dataset



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

Displaying the plot (e.g., pairplot) of the Iris Dataset

We saw this diagram earlier, using the Seaborn pairplot function:



Displaying the train and test feature shapes

```
Train features shape: (112, 4)
Test features shape: (38, 4)
```

Displaying the F1 score / Evaluation

Yes, it is entirely possible that you will get a perfect score with this very small dataset and the distinct features of the three Iris species.

```
kNN F1 Score: 1.00
NB F1 Score: 1.00
```

Deliverables

- Turn in a zip including your source code and screenshots of the program functioning, as follows:
    - An **ipynb** file for Jupyter Notebooks
        - Alternatively, a **py** source file is acceptable as well
    - **Include screenshots of your program working,** placed inside the zip file that you turn in
        - This should include screenshots of the outputs including diagrams and printing of the shapes, evaluation metrics, etc.