Predicting Iris Flower Species with Machine Learning: A KNN Approach

- **Introduction:** You are a data scientist working on a project to classify iris flowers based on their features. The **Iris dataset** (also known as Fisher's Iris dataset) will be the primary data source. It is a classic machine learning dataset containing **150 samples** of iris flowers from three different species: **Setosa, Versicolor, and Virginica**. Each flower is described by four key features: **sepal length, sepal width, petal length, and petal width**.

The problem being solved is a **multi-class classification task**, where the goal is to build a machine learning model that can accurately predict the species of an iris flower based on these physical measurements.

Classification plays an important role in machine learning because it teaches a model to **sort data into categories by learning from labeled examples**. For instance, just as emails can be categorized as "spam" or "not spam," iris flowers can be classified into their respective species. By analyzing labeled data, the model learns patterns that distinguish between categories and can then **automatically predict the category of new, unseen data**. This process is widely used in real-world applications such as email filtering, medical diagnosis, fraud detection, and more, making it a fundamental technique in machine learning (Ref: *Getting Started with Classification - GeeksforGeeks*).

In this scenario, you are collaborating with a team of **botanists** who want to automate the process of identifying iris species. They have collected measurements of sepal and petal dimensions from various flowers, and your team's goal is to develop a **reliable classification model** that can assist botanists in quickly and accurately identifying the correct species.

- **Dataset Overview**

Describe the Iris dataset: features, target labels, number of samples.

The Iris dataset originated from a seminal paper by British statistician and biologist Ronald Fisher titled "The Use of Multiple Measurements in Taxonomic Problems," published in 1936. Fisher collected and measured samples of iris flowers from three different species: Setosa, Versicolor, and Virginica.

The dataset comprises **150 samples**, with each sample having four features measured:

1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width

Additionally, each sample is labeled with its corresponding species, making it a supervised learning problem with three target variables:

1. Setosa
2. Versicolor
3. Virginica

Provide summary statistics or a short description from the dataset.

```
:Summary Statistics:

============== ==== ==== ======= ===== ====================
                Min  Max   Mean    SD   Class Correlation
============== ==== ==== ======= ===== ====================
sepal length:   4.3  7.9   5.84   0.83    0.7826
sepal width:    2.0  4.4   3.05   0.43   -0.4194
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
============== ==== ==== ======= ===== ====================
```

Mention why this dataset is suitable for learning classification.

The Iris dataset is often used as a benchmark in machine learning and pattern recognition for tasks like **classification and clustering**. Its simplicity, clarity, and well-structured data make it an excellent starting point for learning classification algorithms. The dataset is small, fully labeled, balanced across classes, and contains numerical features that are easy to visualize, which allows learners to focus on understanding classification concepts without worrying about data cleaning or preprocessing.

Methodology

The process of building the classifier involves several key steps, from loading and understanding the data to training and evaluating the model.

- ### Loading the Dataset

  The first step is to load the necessary libraries and the dataset. Using scikit-learn, a popular machine learning library in Python, which includes the Iris dataset.

  Code Snippet:

  Import Libraries:

```python
import numpy as np
import pandas as pd
```

  Loading Dataset:

```python
from sklearn.datasets import load_iris
iris_dataset = load_iris()
```

- **Exploring dataset keys, features, and targets**

  Before building a model, it's crucial to understand the data's structure and content. The loaded iris_dataset object is a dictionary-like object containing the data and its metadata**.**

  Dataset:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
..	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

[150 rows x 5 columns]

To view dataset keys:

```
print(iris_dataset.keys())
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

Dataset Information:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Columns 0–3 → Features.

Column 4 (species) → Target variable.

- **Data Splitting**

  We cannot use the same data to train and evaluate our model. If we use the same data, the model might memorize the answers and always give correct predictions. Memorizing does not mean the model will work well with new data.

  To success this:

  Code snippet:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris_dataset['data'],
iris_dataset['target'], random_state=0)
```

- **Visualizing the data with scatter plots:**

  Data visualization is essential for identifying patterns and relationships between features. A **scatter matrix** is an excellent tool for this, as it plots every feature against every other feature. The points are colored according to their species, which helps us visually assess if the classes are separable.

  Code Snippet:

```
iris_dataframe=pd.DataFrame(X_train, columns=iris_dataset.feature_names)
grr=pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15,15),
marker='o', hist_kwds={'bins':20}, s=60, alpha=.8)
```

- **Model Building**

  Initialize the KNN classifier with desired parameters.

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=1)
```

  Model Parameters:

KNeighborsClassifier

▼ Parameters

n_neighbors	1
weights	'uniform'
algorithm	'auto'
leaf_size	30
p	2
metric	'minkowski'
metric_params	None
n_jobs	None

  Model Training:

```
knn.fit(X_train, y_train)
```

  The .fit() method trains the k-NN algorithm by storing the training data points

and their corresponding labels. The model learns the relationship between features and target classes.

- **Results**

  This section presents the outputs and findings from executing the methodology described above.

  Dataset Exploration Output

  The initial exploration provided the following information about the Iris dataset:

  Data Shape: The dataset contains 150 samples (rows) and 4 features (columns).

  Shape of data:

```
#Print the type and shape of the data.

print("Type of data: {}".format(type(iris_dataset['data'])))
print("Shape of data: {}".format(iris_dataset['data'].shape))
```

```
Type of data: <class 'numpy.ndarray'>
Shape of data: (150, 4)
```

  Target Shape: The target vector has 150 entries, one for each sample.

  Shape of target:

```
#Print the type and shape of the target data

print("Type of target: {}".format(type(iris_dataset['target'])))
print("Shape of target: {}".format(iris_dataset['target'].shape))
```

```
Type of target: <class 'numpy.ndarray'>
Shape of target: (150,)
```

  Feature Names:

```
print("Feature names: {}".format(iris_dataset['feature_names']))
```

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

  Target Names:

```
print("Target names: {}".format(iris_dataset['target_names']))
```

```
Target names: ['setosa' 'versicolor' 'virginica']
```

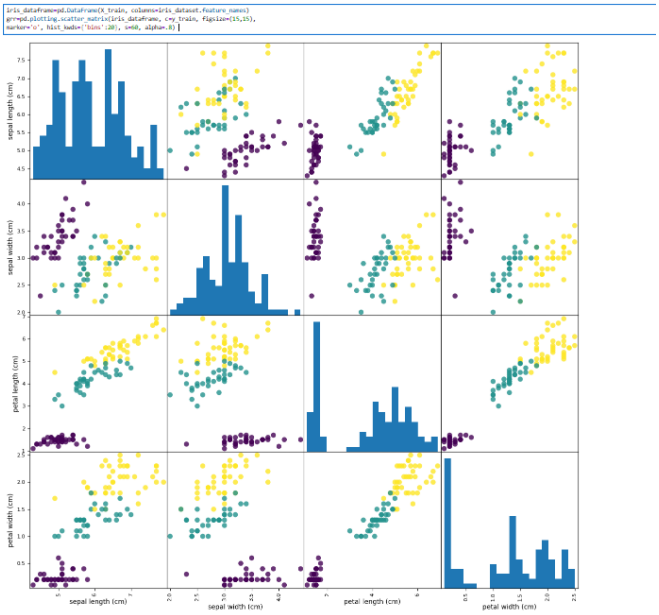
First Five Rows: The feature values for the first five flower samples are:

```
print("First five columns of data:\n{}".format(iris_dataset['data'][:5]))
```

```
First five columns of data:
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

Scatter Matrix Plots

The generated scatter matrix plot (screenshot below) reveals important patterns:



Interpretation:

- The three species are represented by three different colors.
- The plots for petal width vs. petal length show clear separation between the three classes.
- The setosa species (the cluster at the bottom-left of many plots) is distinctly separate from versicolor and virginica.
- versicolor and virginica show some overlap, but can still be distinguished, particularly using petal measurements.
- This visual inspection suggests that a machine learning model should be able to learn to separate these classes effectively.

Model Predictions for New Data

Tested the trained model on a hypothetical new flower with the following measurements: sepal length = 5 cm, sepal width = 2.9 cm, petal length = 1 cm, and petal width = 0.2 cm.

Code & Output:

```
#Making Predictions
```

```
X_new = np.array([[5,2.9,1,0.2]])
print("X_new.shape: {}".format(X_new.shape))
```

```
X_new.shape: (1, 4)
```

```
prediction = knn.predict(X_new)
print("Prediction: {}".format(prediction))
print("Predicted target name: {}".format(iris_dataset['target_names'][prediction]))
```

Result:

```
Prediction: [0]
Predicted target name: ['setosa']
```

The model correctly predicts that this new flower is of the setosa species, which aligns with our visual analysis (low petal length and width).

Accuracy Score on the Test Set

Finally, evaluated the model's performance on the test set, which contains data the model has never seen before. The model's predictions (y_pred) are compared to the actual labels (y_test).

Code & Output:

```
print("Test set score: {:.2f}".format(np.mean(y_pred == y_test)))
```

Result:

```
Test set score: 0.97
```

This final score means the model achieved an accuracy of 97%. Out of the 38 samples in the test set, the model correctly predicted the species for approximately 37 of them. This high accuracy indicates that our model is very effective at classifying iris species based on their measurements.

E. Discussion

Feature Separation and Data Distribution

The scatter plot matrix is a crucial tool for visual analysis. The plots reveal that petal length and petal width are the most discriminative features for classifying the iris species. These features create clear, well-separated clusters with minimal overlap between the three species. The setosa species, in particular, is distinctly separate from versicolor and virginica. While sepal measurements are informative, they show more overlap between the versicolor and virginica species.

The histograms on the diagonal of the matrix show the distribution of each feature. Notably, the histograms for petal length and petal width appear bimodal, showing two distinct groups of data. This visualizes the clear separation between setosa and the other two species, suggesting that simple thresholds on these features could be effective classifiers.

Analysis of the k-NN Model

The k-Nearest Neighbors (k-NN) algorithm was used to build the classification model. In simple terms, this algorithm classifies a new data point by finding the most similar data points in the training set and making a prediction based on their labels. For this exercise, with the parameter

n_neighbors=1, the model finds the single closest data point in the training data and assigns its label to the new flower.

This choice of k=1 has specific implications:

- High Variance: The model has high variance but low bias because its prediction is entirely dependent on the single nearest neighbor. This makes it sensitive to noise and outliers; a single mislabeled point in the training data could cause an incorrect prediction.
- Effect of a Larger k: Increasing k to 5 or 10 would force the model to consider the "votes" of more neighbors. This would smooth out the decision boundaries and make the model more robust to individual outliers, potentially improving its ability to generalize to new data.

When training the model, the

.fit() method for k-NN is straightforward: it simply stores the training data and their labels. This is different from algorithms like linear regression, which learn specific model parameters (weights) during the fitting process.

Prediction and Evaluation

For a new iris sample with measurements

[5, 2.9, 1, 0.2], the model predicted the species as setosa. This prediction aligns perfectly with the visual evidence from the scatter plots, where flowers with a petal length of 1.0 cm and petal width of 0.2 cm fall squarely within the distinct cluster identified as setosa.

The model achieved an accuracy of 97% on the test set. In real-world terms, this means the model correctly identified the species for approximately 37 out of the 38 flowers in the test set—data the model had never seen before. This high accuracy indicates that the

model is very effective and could be reliably used by botanists to automate species identification for new iris samples.

Strengths, Limitations, and Broader Challenges

The k-NN approach demonstrated several strengths in this exercise:

- Simplicity: The algorithm is easy to understand and implement.
- No Assumptions: It is a non-parametric model that makes no assumptions about the underlying data distribution.
- High Accuracy: It achieved excellent performance on this particular dataset.

However, if this approach were applied to a much larger and noisier dataset, several challenges and limitations would become more prominent:

- Computational Cost: As a dataset grows, prediction time increases because the algorithm must calculate the distance to all training points for each new prediction.
- Sensitivity to Noise: The model's performance can be degraded by noisy data or outliers, especially with a small k value. To address this, one might increase

k or apply data cleaning techniques beforehand.

- Curse of Dimensionality: The model's effectiveness can decrease as the number of features (dimensions) grows.

Machine Learning Concepts Demonstrated

This exercise illustrates several core machine learning concepts:

- Supervised Learning: Using a dataset with labeled examples (features and their corresponding species) to train a predictive model.
- Train-Test Split: The practice of splitting data into a training set (75%) and a testing set (25%) is essential for proper model evaluation. It ensures that we can get an unbiased estimate of how the model will perform on new, unseen data, rather than just testing its ability to "memorize" the training data.
- Feature Importance: The visual analysis showed that some features (petal measurements) are more informative for prediction than others.

F. Conclusion

This laboratory exercise successfully demonstrated the complete machine learning workflow for classification tasks. Key learning outcomes include:

1. **Data understanding:** The importance of exploring dataset characteristics, feature distributions, and class balance before modeling.
2. **Proper evaluation:** The necessity of splitting data into training and testing sets to obtain unbiased performance estimates.
3. **Algorithm selection:** k-Nearest Neighbors proved effective for this dataset due to the clear clustering of species in the feature space.
4. **Visual analysis:** Scatter plots provide valuable insights into feature relationships and class separability that inform model choice and interpretation.

   The 97% accuracy achieved demonstrates that iris species can be reliably identified using simple measurements, validating the utility of machine learning for botanical classification tasks. This exercise provides a solid foundation for understanding more complex classification scenarios and advanced machine learning algorithms.

   The workflow demonstrated here—from data loading and exploration through model building and evaluation—represents the standard approach to supervised learning problems and serves as a template for tackling similar classification challenges in various domains.