**Table of Content**

1. **The potential of defining new or existing iris species using KNN model**

6510004 – Sai Kyal Sin Tun

1. **Introduction**

* Role of Classification in Machine Learning
* Iris Dataset and Research Objective

1. **Dataset Overview**

* Dataset Description (features, targets, samples)
* Statistics Description of Iris dataset (Highlighted features: Correlation)
* Relevance of the Iris Dataset

1. **Methodology**

* Research Process and ML Workflow

1. **Results**

* Dataset Exploration
* Model Predictions
* Outliers and Observations
* Accuracy Score

1. **Discussion**

* Key Insights and Interpretation
* Model Strengths and Limitations
* Summary of Findings

1. **Conclusion**

* Learning Outcomes and Final Remarks

1. **The potential of defining new or existing iris species using KNN model**

6510004 – Sai Kyal Sin Tun

Platform used: Jupyter Notebook

1. **Introduction**

Role of Classification in Machine Learning

In this case study, I will be defining the undefined species of the iris flower types based on their sepal and petal dataset. My research will be mainly based on the existing dataset and will try to solved the problems or get the appropriate result. During my research, I will be determining the species types which is iris family types. In this dataset, we mainly have sample data like the length and width of both sepal and the petal. So, to summarize, I will research along with the dataset based on existing model which is KNN model (i.e. k-Nearest Neighbors). The way KNN model work in my study will mainly be based on historical data or existing data from iris dataset to clearly differentiate the type of plants just by their average length and width of petal and sepal.

Iris Dataset and Research Objective

The procedure in which I will be importing the existing iris dataset and reached to the end of research which is the result of the type of iris. But does it really define what kind of iris will be shown as result or the accuracy of the KNN model. At the end of this research, I will be defining the potential of accuracy of KNN model and varieties of result.

Summary Objective

* **To understand how classification works in machine learning** by applying the KNN algorithm to the Iris dataset.
* **To explore the relationship between sepal/petal measurements and iris species**, and identify which features contribute most to species separation.
* **To analyze the strengths and limitations of KNN** in classifying flower species, including its handling of outliers and different values of *k*.

1. **Dataset Overview**

**Dataset Description (features, targets, samples)**

Let first take a look at the dataset overview and their normality alongside their class correlation. The dataset contains length and width of sepal and petal ranging from Minimum, Maximum, Mean and SD which will then result with Class correlation. In Class correlation of iris data, we have positive correlation and negative correlation and high value correlation. In this dataset, we have 150 instances and 4 numeric, predictive attributes and class.

**Class of Iris**: Iris-Setosa, Iris-Versicolour, Iris-Virginica

**Statistic Description of Iris Dataset**

Let say that we have +1, 0 and -1 in relation with class correlation. Below I will be describing and analyzing of how the correlation is affected by sepal and petal width and length. But the correlation of ones’ values can be change depending on the iris class type because not every type of iris is the same width and length so the problem with defining positive and negative correlation can be just a saying in the case of different iris but within the same type it can be define normally and assume with correct precision. So, to summarize, if ones’ correlation is high, we can assume that it is some type of iris evidently.

* **Positive correlation**: Higher feature values tend to belong to higher species labels.
* **Negative correlation**: Higher feature values tend to belong to lower species labels.
* **High correlation (near ±1): F**eature is very predictive of class.
* **Low correlation (near 0)**: Feature is not very helpful for classification.

**Relevance of the Iris Dataset and its benefit**

Now come the part where we imagine as a machine learning engineer. Learning from one dataset that is clearly big which is big-data or large data but doesn’t have information about classification or relevance can be a big challenge for us all. But in this dataset, we can see much more than what we need as a machine learning engineer or data analytics. Why Iris dataset is relevance to us all is it can be used as basic model for most of the machine learning tasks which is mostly famous for making prediction and training model. Iris dataset clearly has the basic functions that is not hard to maintain and easy to understand. In this iris dataset, we have different type of classification and clear statistical result of how ones’ class correlation of different class can’t be taken normally.

1. **Methodology**

**Research Process and ML Workflow**

Let get started for our research by loading the dataset and to the end of training the KNN model.

**Loading the dataset**

I will be using a well-known labeled dataset (supervised learning). Each row is a flower; features are sepal/petal length/width; the target is species.

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

For the next step after loading dataset, we can define the keys of iris dataset. In this case, we will be using iris\_dataset.keys() command where it will show data, target, features as dict\_keys as a result.

Output: Keys of iris dataset: dict\_keys(['data', 'target', 'frame', 'target\_names', 'DESCR', 'feature\_names', 'filename', 'data\_module'])

After that, we can figure out the features and target separately using format() function with their index name. (i.e. format(iris\_dataset[‘target\_names’]) and format(iris\_dataset[‘feature\_names’]))

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

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

So, in target we have three class namely setosa, versicolor and virginica whereas in feature we have sepal and petal length and width of each types.

**Splitting data into training and testing sets**

In this step, the Iris dataset is divided into training and testing sets using the train\_test\_split function from scikit-learn. The feature matrix (iris\_dataset['data']) and the corresponding target labels (iris\_dataset['target']) are randomly separated into two parts. The **training set** (X\_train, y\_train) contains the majority of samples and is used to teach the model the relationship between flower measurements and species. The **testing set** (X\_test, y\_test) contains the remaining samples and is reserved for evaluating the model’s performance on unseen data. The printed shapes confirm the split: 112 samples in the training set and 38 samples in the testing set. By specifying random\_state=0, the split is reproducible, meaning the same training and testing sets can be generated each time the code is run. This process ensures that the model can be trained effectively while providing a fair assessment of its ability to generalize.

**Building the model with k-Nearest Neighbors**

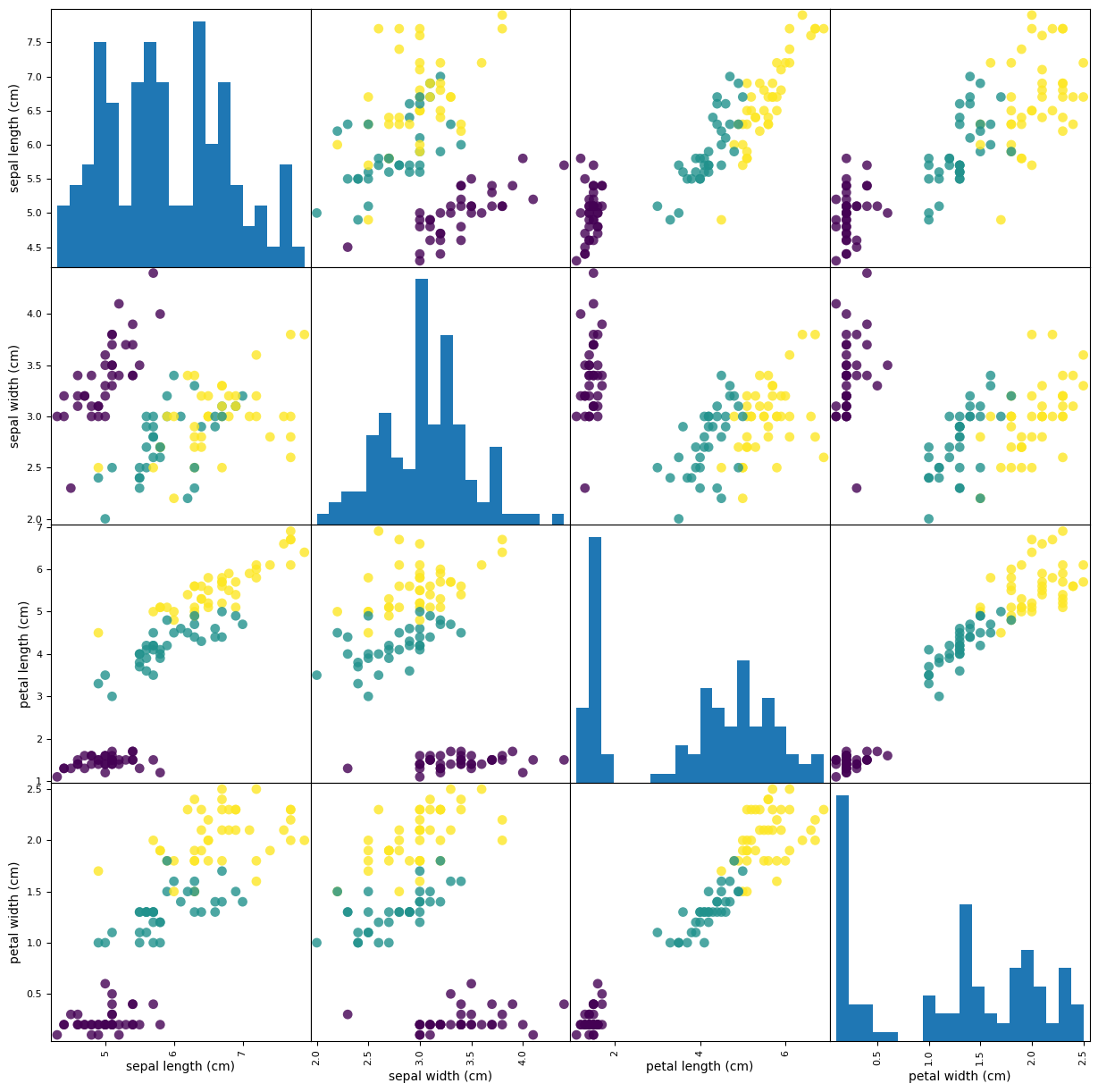
In this step, a k-Nearest Neighbors (k-NN) classifier is created using scikit-learn’s KNeighborsClassifier. The code knn = KNeighborsClassifier(n\_neighbors=1) initializes the model with k=1, meaning that the class of a new sample will be predicted based on the label of its **single nearest neighbor** in the feature space. This algorithm is an instance-based, distance-driven classifier: it stores the training data and makes predictions by measuring distances (usually Euclidean) between the new sample and all points in the training set. The output KNeighborsClassifier(n\_neighbors=1) confirms that the model has been successfully created with the specified parameter. Building this model is a preparatory step before fitting it to the training data and using it to predict the species of unseen Iris flowers.

1. **Results**

**Dataset Exploration**

Exploring the dataset is an essential first step in any machine learning workflow. By printing iris\_dataset['target'], we can observe the numeric labels representing the three Iris species: 0 for Setosa, 1 for Versicolor, and 2 for Virginica. The output shows that the dataset is **ordered by species**, with the first 50 samples labeled 0, the next 50 labeled 1, and the last 50 labeled 2. This reveals that the dataset is **balanced**, with an equal number of samples per class, which is important for training a fair classifier. Dataset exploration like this helps confirm the integrity of the data, provides insights into class distribution, and guides subsequent steps such as splitting into training and testing sets, feature selection, and model evaluation.

**Scatter matrix plots**



**Model Predictions**

After the k-Nearest Neighbors model is trained, it can be used to predict the class of both individual new samples and the entire test set. For a single new sample X\_new, the code prediction = knn.predict(X\_new) returns a numeric label [0], which corresponds to the species ['setosa'] when mapped using iris\_dataset['target\_names'][prediction]. This demonstrates how the model classifies an unseen example based on the nearest neighbor in the feature space.

To evaluate the model more broadly, predictions are also made on the **testing set** using y\_pred = knn.predict(X\_test). The output shows the predicted class labels for all test samples, such as [2 1 0 2 0 2 ...]. Comparing these predicted labels with the true test labels allows assessment of the model’s overall performance and generalization ability. Together, these steps illustrate how the k-NN algorithm applies distance-based classification to both individual and multiple samples, providing practical insights into the model’s accuracy and reliability.

**Accuracy Score**

The accuracy of the k-Nearest Neighbors model on the testing set is calculated to evaluate its performance. Using the code np.mean(y\_pred == y\_test), the proportion of correctly predicted labels (y\_pred) that match the true labels (y\_test) is computed. The output Test set score: 0.97 indicates that the model correctly predicted **97% of the test samples**. This high accuracy demonstrates that the k-NN classifier has successfully learned the patterns in the training data and can generalize well to unseen examples. Accuracy is a straightforward and widely used metric for classification tasks, providing a clear measure of how reliably the model can classify new instances.

1. **Discussion**

**Scatter Plots and Feature Separation**  
The scatter plots of the Iris dataset reveal that petal length and petal width provide the clearest separation between species, whereas sepal length and sepal width show more overlap. Specifically, Setosa forms a distinct cluster, while Versicolor and Virginica overlap slightly. This visualization explains why the k-NN model achieves high accuracy: features with well-separated clusters allow distance-based classifiers to assign the correct labels more reliably.

**Strengths and Limitations of k-NN**  
The k-Nearest Neighbors algorithm has several strengths. It is simple and intuitive, requires no explicit training or model assumptions, and can achieve high accuracy when classes are well-separated in feature space. However, k-NN also has limitations: it can be computationally expensive on large datasets because it calculates distances to all training points, it is sensitive to irrelevant or differently scaled features, and its performance depends on the choice of k and the distance metric.

**Relation to Machine Learning Concepts**  
The high test accuracy (97%) confirms that the k-NN model can generalize well when features provide clear separation, illustrating key machine learning concepts such as **generalization**, **overfitting**, and **instance-based learning**. The model’s success also emphasizes the importance of **feature selection** and **data preprocessing**, as features that separate classes effectively improve predictive performance. Overall, this experiment demonstrates how a simple, non-parametric algorithm can be highly effective when the dataset’s structure aligns with the method’s assumptions.

1. **Conclusion**

**Learning Outcomes and Final Remarks**

This research demonstrated the effectiveness of the k-Nearest Neighbors (k-NN) algorithm in classifying Iris flower species using sepal and petal measurements. By applying the model to the Iris dataset, the study achieved a high accuracy score of 97%, confirming that k-NN can generalize well when the features provide clear class separation. The analysis also revealed that petal length and petal width are the most influential features for species differentiation, while sepal-based features contribute less distinctly.

The experiment highlighted both the strengths and limitations of the k-NN model. Its simplicity and instance-based approach made it effective for this dataset, but its sensitivity to feature scaling, computational cost for large datasets, and dependency on the choice of *k* were also evident. These insights reinforced key machine learning concepts such as generalization, feature importance, and the trade-offs involved in model selection.

Overall, this study not only validated the predictive power of k-NN for well-structured datasets but also provided practical experience in applying a supervised learning workflow—from dataset exploration and preprocessing to training, testing, and evaluation. The findings underscore the importance of selecting appropriate features and models in classification tasks, offering a foundation for extending similar approaches to more complex, real-world datasets.