Iris Species Detection

In this project, I created machine learning (ML) models among which one was being used to predict the species of flowers based on their features.

We will go through the steps taken, the challenges faced, etc. during the project.

<u>Index:</u>

Libraries used for the project:

- Numpy
- Pandas
- Matplotlib
- Seaborn
- Sklearn
- Streamlit
- Pickle

Data Collection

- The data has been collected from the following source:

  https://www.kaggle.com/datasets/uciml/iris

- This is an organized data loaded in the ".csv" format.

Data Understanding

- The data was then converted into a dataframe called "iris_df".
- The dimensions of the dataframe are 150 rows and 6 columns.
- These columns contain the names of the features and the target variable for the ML model.
- The six columns along with their datatypes are:

Column	***Datatypes***
1 Id	int64
2 SepalLengthCm	float64
3 SepalWidthCm	float64
4 PetalLengthCm	float64
5 PetalWidthCm	float64
6 Species	Object

- Here, the features for the ML model are the "*Length and Width of Sepal and Petal*" while the target variable is "*Species*" containing names of three iris species: *Setosa, Versicolor, Virginica*

Data Cleaning

- There were no missing values in the dataframe so therefore no cleaning was needed.
- The target column containing the names of iris species have prefix "*iris*" in every row so that prefix was removed.

Data Analysis

For the analysis, the visualisation of the data has been implemented through various plots which gave some meaningful insights as follows:

- According to the Heatmap of the numeric features,
    - The *length and width of petal* shows *high positive correlation* while the *width of sepal and petal* are *negatively correlated* to each other.
- According to their pair plot,
    - The *length and width of Petal* are directly proportional.
- According to the Scatter plot for the sizes of the Species,
    - Smallest - Setosa
    - Biggest - Virginca
    - Middle – Versicolor
- There is no imbalance in the count of the Species in the data.

Data Preprocessing

Following process have been performed before building the ML models:

- Encoding of the target column – *Label Encoding*
- Dropping Unnecessary column i.e., “*Id*” column.
- Train-Test Split of the dataframe – Ratio of 70:30
- Standard Scaling of the data.

Building ML models

Following models have been built along with their hyperparameter tuning separately in order to find the optimum one:

- Logistic Regression
- Logistic Regression with hyperparameter tuning
- Decision Trees
- Decision Trees with hyperparameter tuning
- Random Forest
- Random Forest with hyperparameter tuning
- Support Vector Machine (SVM)

- SVM with hyperparameter tuning

As the evaluation metrics of the above-mentioned models were mostly similar so each one could be the suitable fit, therefore the model of ***SVM with hyperparameter tuning*** was chosen for building the web application.

The Train and Test accuracies of the model are 96 and 100 percent respectively. At last, the model was also tested through User Input of the features and then predicted the species.

<u>**Building Web application**</u>

To build the web application, the SVM model , scaling code and encoding code were pickled in the jupyter notebook itself and were loaded into the Spyder Python IDE where the coding for the application was being done and then executed using streamlit library via Jupyter Terminal of the ML model environment which then opened the application on the browser.