**Hello World of Machine Learning**

The best small project to start with on a new tool is the classification of iris flowers (e.g. [the iris dataset](https://archive.ics.uci.edu/ml/datasets/Iris)).

* Attributes are numeric so you have to figure out how to load and handle data.
* It is a classification problem, allowing you to practice with perhaps an easier type of supervised learning algorithm.
* It is a multi-class classification problem (multi-nominal) that may require some specialized handling.
* It only has 4 attributes and 150 rows, meaning it is small and easily fits into memory (and a screen or A4 page).
* All of the numeric attributes are in the same units and the same scale, not requiring any special scaling or transforms to get started.

**Code:-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

# 2. Load the dataset

iris = load\_iris()

X = iris.data

y = iris.target

feature\_names = iris.feature\_names

target\_names = iris.target\_names

# Convert to pandas DataFrame for easier handling

data = pd.DataFrame(data=X, columns=feature\_names)

data['species'] = [target\_names[i] for i in y]

# 3. Summarize the dataset

print("Dimensions of the dataset:")

print(data.shape)

print("\nPeek at the data:")

print(data.head())

print("\nStatistical summary of all attributes:")

print(data.describe())

print("\nBreakdown of the data by the class variable:")

print(data['species'].value\_counts())

# 4. Visualize the dataset

# Univariate plots

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))

for i, feature in enumerate(feature\_names):

    sns.histplot(data[feature], kde=True, ax=axes[i // 2, i % 2])

    axes[i // 2, i % 2].set\_title(f'Distribution of {feature}')

plt.tight\_layout()

plt.show()

# Multivariate plots

sns.pairplot(data, hue='species')

plt.show()

# 5. Evaluate some algorithms

# Separate out a validation dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Set up models

models = {

    'Logistic Regression': LogisticRegression(max\_iter=200),

    'Linear Discriminant Analysis': LinearDiscriminantAnalysis(),

    'K-Nearest Neighbors': KNeighborsClassifier(),

    'Decision Tree': DecisionTreeClassifier(),

    'Gaussian Naive Bayes': GaussianNB(),

    'Support Vector Machine': SVC()

}

# Cross-validation and evaluation

results = {}

for name, model in models.items():

    # 10-fold cross-validation

    cv\_scores = cross\_val\_score(model, X\_train, y\_train, cv=10)

    results[name] = {

        'Mean Accuracy': np.mean(cv\_scores),

        'Standard Deviation': np.std(cv\_scores)

    }

# Print results

print("\nAlgorithm Performance:")

for name, metrics in results.items():

    print(f"{name}: Mean Accuracy = {metrics['Mean Accuracy']:.3f}, Std Dev = {metrics['Standard Deviation']:.3f}")

# 6. Making some predictions

# Fit all models and make predictions on the test set

for name, model in models.items():

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    accuracy = accuracy\_score(y\_test, y\_pred)

    print(f"\n{name} Test Accuracy: {accuracy:.3f}")

