**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.

To do

1. Installing the Python and SciPy platform.

pip install numpy pandas matplotlib seaborn scikit-learn

1. Loading the dataset.
2. Summarizing the dataset.
   * Dimensions of the dataset.
   * Peek at the data itself.
   * Statistical summary of all attributes.
   * Breakdown of the data by the class variable.
3. Visualizing the dataset.
   * Univariate plots to better understand each attribute.
   * Multivariate plots to better understand the relationships between attributes.

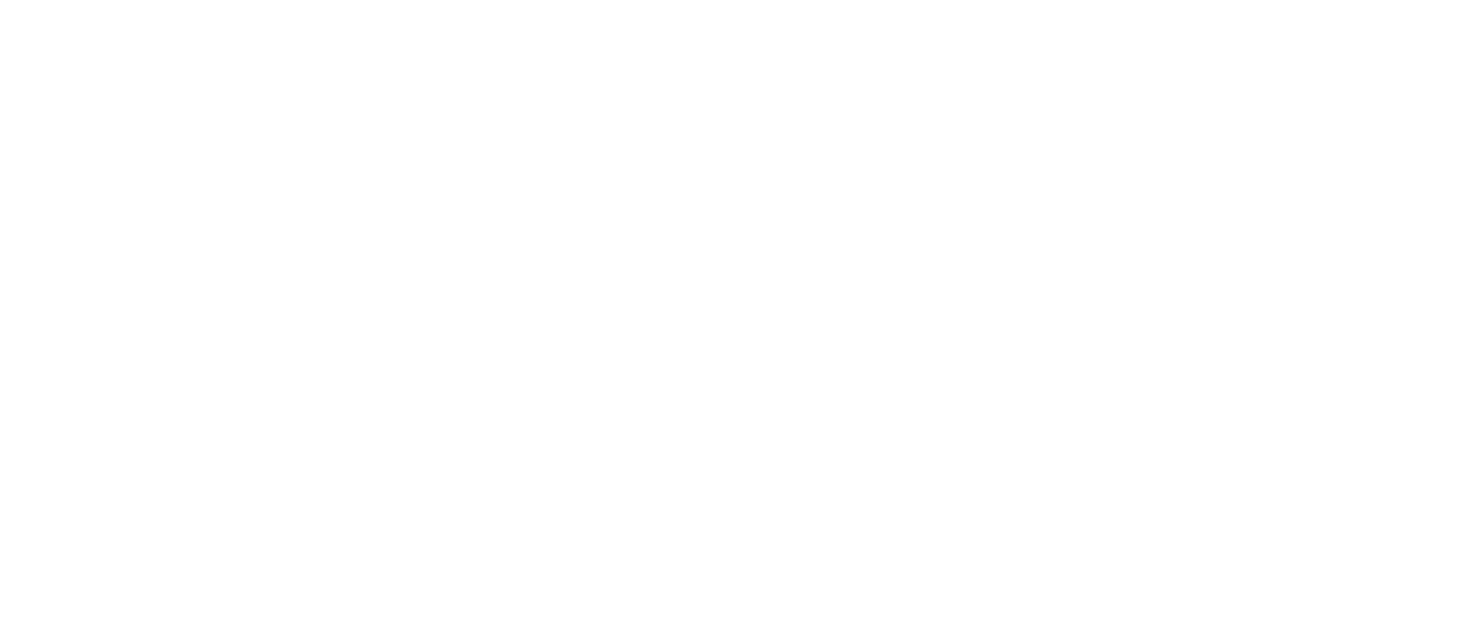
1. Evaluating some algorithms.
   * Separate out a validation dataset.
   * Set-up the test harness to use 10-fold cross validation.
   * Build multiple different models to predict species from flower measurements  Select the best model. test 6 different algorithms:

* + - Logistic Regression (LR) o Linear Discriminant Analysis (LDA) o K-Nearest Neighbors (KNN).
    - Classification and Regression Trees (CART).
    - Gaussian Naive Bayes (NB). o Support Vector Machines (SVM).

1. Making some predictions.

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.datasets import load\_iris from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score 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 classification\_report, accuracy\_score  # Step 2: Load the Iris Dataset iris = load\_iris() data = pd.DataFrame(data=iris.data, columns=iris.feature\_names) data['target'] = iris.target  # Step 3: Summarize the Dataset  # Check dimensions print("Dataset Dimensions:", data.shape)  # Peek at the data print("\nFirst 5 Rows of the Dataset:") print(data.head())  # Statistical summary of all attributes print("\nStatistical Summary:") print(data.describe())  # Breakdown of the data by class variable print("\nClass Distribution:") print(data['target'].value\_counts())  # Step 4: Visualize the Dataset # Univariate Plots (Histograms) data.hist(figsize=(10, 8)) plt.suptitle('Histogram of Each Feature') plt.show()  # Multivariate Plots (Pairplot) sns.pairplot(data, hue='target', markers=["o", "s", "D"]) plt.suptitle('Pairplot of Features', y=1.02) plt.show()  # Step 5: Evaluate Algorithms X = data.drop('target', axis=1) y = data['target']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) kfold = KFold(n\_splits=10, random\_state=42, shuffle=True) models = {  'Logistic Regression (LR)': LogisticRegression(max\_iter=200),  'Linear Discriminant Analysis (LDA)': LinearDiscriminantAnalysis(),  'K-Nearest Neighbors (KNN)': KNeighborsClassifier(),  'Classification and Regression Trees (CART)': DecisionTreeClassifier(),  'Gaussian Naive Bayes (NB)': GaussianNB(),  'Support Vector Machines (SVM)': SVC()  }  results = {}  print("\nCross-Validation Results:") for name, model in models.items(): |



cv\_results

=

cross\_val\_score

(

model

,

X\_train

,

y\_train

,

cv

=

kfold

,

scoring

=

'accuracy'

)

results

[

name

]

=

cv\_results

.

mean

()

print

(

f

"

{

name

}

:

{

cv\_results

.

mean

()

:.4f}

"

)

#

Step

6:

Make

Predictions

best\_model\_name

=

max

(

results

,

key

=

results

.

get

)

best\_model

=

models

[

best\_model\_name

]

print

(

f

"

\

n

Best Model:

{

best\_model\_name

}

"

)

best\_model

.fit(

X\_train

,

y\_train

)

predictions

=

best\_model

.predict(

X\_test

)

print

(

"

\

n

Accuracy Score on Test Set:"

)

print

(

accuracy\_score

(

y\_test

,

predictions

))

print

(

"

\

n

Classification Report:"

)

print

(

classification\_report

(

y\_test

,

predictions

))

