PHASE 3

START BUILDING THE MACHINE LEARNING MODEL USING IBM CLOUD WASTON STUDIO

# **DEPLOYING MODELS WITH WASTON MACHINE LEARNING:**

You can use IBM Watson Machine Learning to deploy models, scripts, and functions, manage deployments, and prepare assets for distribution to provide predictions and insights.

**IBM WASTON MACHINE LEARNING ARCHITECTURE AND SERVICES:**

Watson Machine Learning is an IBM Cloud service that allows you to train and deploy machine learning models and neural networks. Watson Machine Learning is a scalable, open source platform built on Kubernetes and Docker components that allows you to develop, train, deploy, and managemachine learning and deep learning models.

**IRIS FLOWER DATASET:**

The Iris flower data set, often known as Fisher's Iris data set, is a multivariate data set that was developed and popularized by British statistician and biologist Ronald Fisher in his 1936 paper The use of many measurements in taxonomic issues as an example of linear discriminant analysis. It is frequently dubbed Anderson's Iris data set since Edgar Anderson collected the data to assess the morphologic variance of Iris blossoms of three related species. Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, picked on the same day, and measured by the same person with the same apparatus."

The data set includes 50 samples from each of the three Iris species (Iris setosa, Iris virginica, and Iris versicolor). Each sample had four characteristics measured: the length and width of the sepals and petals in centimeters. Fisher created a linear discriminant model to distinguish the species based on the combination of these four traits. Fisher's study was published in the Annals of Eugenics (formerly the Annals of Human Genetics) and contains a discussion of the techniques' applications to phrenology.

The Iris Dataset is known as the "Hello World" of data science. It is divided into five columns: Petal Length, Petal Width, Sepal Length, Sepal Width, and Species Type. Iris is a flowering plant, and the researchers measured and documented many characteristics of the various iris blossoms.

Obtain the iris.csv file. We will now load this CSV file into the Pandas library and transform it to a dataframe. To read CSV files, use the read\_csv() function.

Lets we see the example for iris flower data set

# PYTHON PROGRAM:

EXAMPLE:

**import** pandas as pd

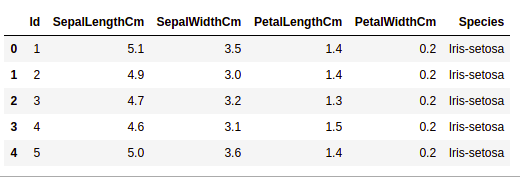
# Reading the CSV file

df **=** pd.read\_csv("Iris.csv")

# Printing top 5 rows

df.head()

OUTPUT:



**GETTING INFORMATION ABOUT THE DATASET:**

The shape parameter will be used to determine the shape of the dataset.

EXAMPLE:

df.shape

OUTPUT:

150,6

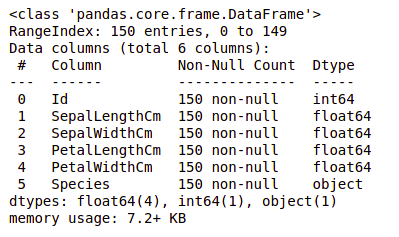
We can observe that the dataframe comprises 6 columns and 150 rows.

Now consider the columns and their data types. We shall use the info() method for this.

EXAMPLE:

df.info()

OUTPUT:



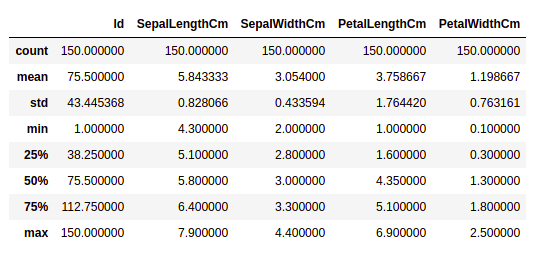
As we can see, just one column has categorical information, while the others are all numeric columns with non-Null entries.

Let's use the describe() method to acquire a quick statistical summary of the dataset. The describe() method performs fundamental statistical computations on the dataset, such as extreme values, data point count, standard deviation, and so on. Any missing or NaN value is skipped automatically. The describe() function provides an accurate representation of data distribution.

EXAMPLE:

df.describe()

OUTPUT:



Each column's count is shown, as well as its mean, standard deviation, minimum and maximum values.

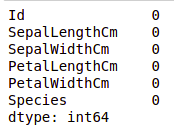
**CHECKING MISSING VALUES:**

We'll see if there are any missing values in our data. When no information is provided for one or more elements, or for the entire unit, missing values can occur. The isnull() method will be used.

EXAMPLE:

df.isnull().sum()

OUTPUT:



We can observe that there is no missing value in any column.

CHECKING DUPLICATES:

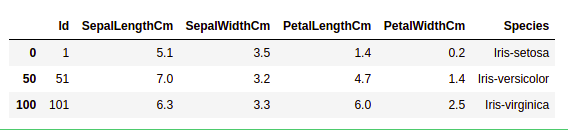
Let's see whether there are any duplicates in our dataset. The drop\_duplicates() method in Pandas assists in deleting duplicates from a data frame.

EXAMPLE:

data **=** df.drop\_duplicates(subset **=**"Species",)

data

OUTPUT:

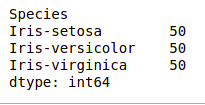


As we can see, there are just three distinct species. Let's see if the dataset is balanced, that is, if all of the species have an equal number of rows. The Series.value\_counts() function will be used. This function returns a Series with unique value counts.

EXAMPLE:

df.value\_counts("Species")

OUTPUT:



We can see that each species has an equal number of rows, hence no entries need be deleted.

DATA VISUALISATION:

VISUALISING THE TARGET COLUMN:

Our target column will be the Species column because we will only require the results based on the species. Let's look at a species countplot.

Lets we see example

EXAMPLE:

# importing packages

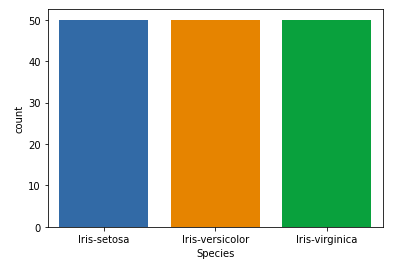
**import** seaborn as sns

**import** matplotlib.pyplot as plt

sns.countplot(x**=**'Species', data**=**df, )

plt.show()

OUTPUT:



RELATION BETWEEN VARIABLES:

We'll look at the relationship between sepal length and sepal breadth, as well as petal length and petal width.

EXAMPLE 1: Comparing sepal length and sepal width:

# importing packages

**import** seaborn as sns

**import** matplotlib.pyplot as plt

sns.scatterplot(x**=**'SepalLengthCm', y**=**'SepalWidthCm',

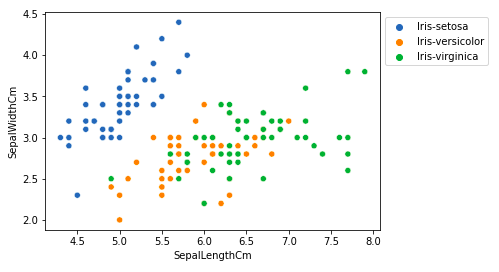
                hue**=**'Species', data**=**df, )

# Placing Legend outside the Figure

plt.legend(bbox\_to\_anchor**=**(1, 1), loc**=**2)

plt.show()

OUTPUT:



From the above plot, we can infer that

* Setosa has shorter sepal lengths but wider sepal widths.
* In terms of sepal length and width, the Versicolor Species falls in the middle of the other two species.
* Species Virginica has longer sepals but narrower sepals.

EXAMPLE 2: Comparing petal length and petal width:

# importing packages

**import** seaborn as sns

**import** matplotlib.pyplot as plt

sns.scatterplot(x**=**'PetalLengthCm', y**=**'PetalWidthCm',

                hue**=**'Species', data**=**df, )

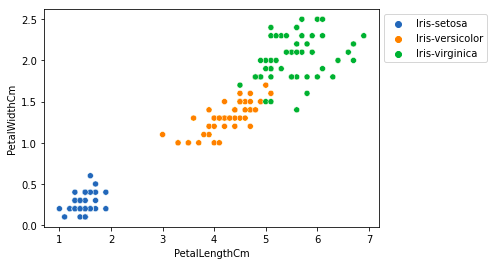
# Placing Legend outside the Figure

plt.legend(bbox\_to\_anchor**=**(1, 1), loc**=**2)

plt.show()

Lets we see the output for the comparison of the petal length and petal width

OUTPUT:



From the above plot, we can infer that

* Species Setosa has shorter and narrower petal lengths and widths.
* In terms of petal length and width, the Versicolor Species falls in the middle of the other two species.
* The petal lengths and widths of Species Virginica are the longest and widest

Let's use a pairplot to visualize all of the column's relationships. It is suitable for multivariate analysis.

EXAMPLE:

# importing packages

**import** seaborn as sns

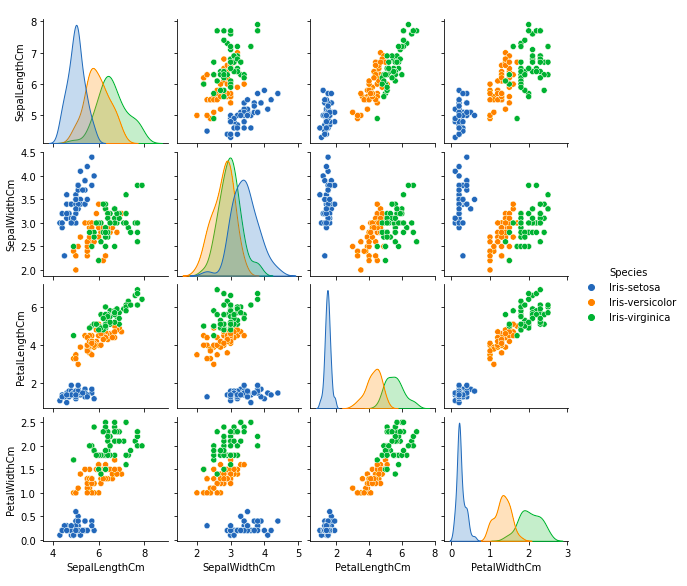
**import** matplotlib.pyplot as plt

sns.pairplot(df.drop(['Id'], axis **=** 1),

             hue**=**'Species', height**=**2)

Lets we see the output for above example…

OUTPUT:



This plot reveals a variety of correlations, such as the species Setosa having the shortest petals widths and lengths. It also has the shortest sepal length yet the widest sepals. Such data can be acquired for any other species.

HISTOGRAMS:

A Histogram is a variation of a bar chart in which data values are grouped together and put into different classes. This grouping enables you to see how frequently data in each class occur in the dataset. The histogram graphically shows the following

* Frequency of different data points in the dataset.
* Location of the center of data.
* The spread of dataset.
* Skewness/variance of dataset.
* Presence of outliers in the dataset.

Histograms allow seeing the distribution of data for various columns. It can be used for uni as well as bi-variate analysis.

EXAMPLE:

# importing packages

**import** seaborn as sns

**import** matplotlib.pyplot as plt

fig, axes **=** plt.subplots(2, 2, figsize**=**(10,10))

axes[0,0].set\_title("Sepal Length")

axes[0,0].hist(df['SepalLengthCm'], bins**=**7)

axes[0,1].set\_title("Sepal Width")

axes[0,1].hist(df['SepalWidthCm'], bins**=**5);

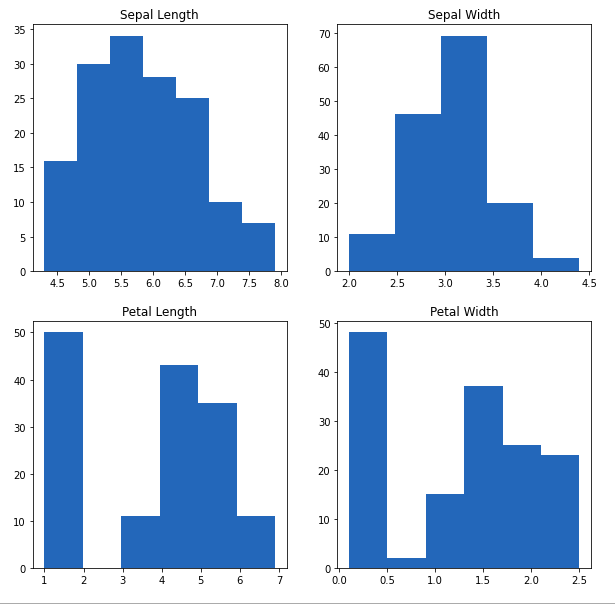
axes[1,0].set\_title("Petal Length")

axes[1,0].hist(df['PetalLengthCm'], bins**=**6);

axes[1,1].set\_title("Petal Width")

axes[1,1].hist(df['PetalWidthCm'], bins**=**6);

OUTPUT:



From the above plot, we can see that

* The largest frequency of sepal length is between 30 and 35, or 5.5 and 6
* The maximum frequency of sepal width is roughly 70, which is between 3.0 and 3.5.
* The petal length has a frequency of roughly 50, which is between 1 and 2.
* The petal width has the largest frequency between 40 and 50, which is between 0.0 and 0.5.

HISTOGRAM WITH DISPLOT PLOT:

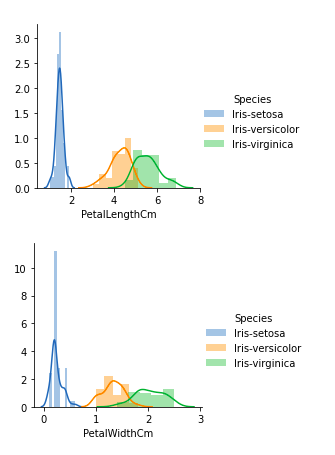
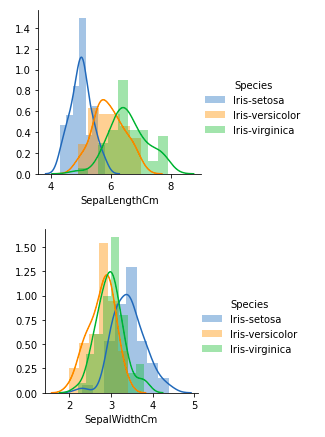
Distplot is mostly used to depict a univariant set of observations using a histogram, i.e. only one observation and hence one specific column of the dataset.

EXAMPLE:

|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt    plot **=** sns.FacetGrid(df, hue**=**"Species")  plot.map(sns.distplot, "SepalLengthCm").add\_legend()    plot **=** sns.FacetGrid(df, hue**=**"Species")  plot.map(sns.distplot, "SepalWidthCm").add\_legend()    plot **=** sns.FacetGrid(df, hue**=**"Species")  plot.map(sns.distplot, "PetalLengthCm").add\_legend()    plot **=** sns.FacetGrid(df, hue**=**"Species")  plot.map(sns.distplot, "PetalWidthCm").add\_legend()    plt.show() |

Lets we see the output for above program…

OUTPUT:



From the above plots, we can see that

* There is a lot of overlapping in the case of Sepal Length.
* There is also a great deal of overlapping in the case of Sepal Width.
* There is relatively little overlapping in the case of Petal Length.
* There is likewise very little overlapping in the case of Petal Width.

So we can use Petal Length and Petal Width as the classification feature.

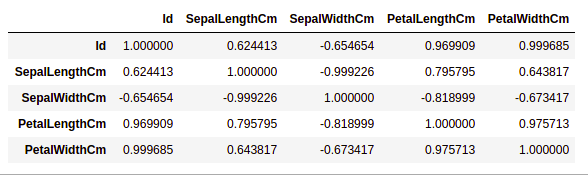
HANDLING CORELATION:

Pandas dataframe.corr() is used to find the pairwise correlation of all columns in the dataframe. Any NA values are automatically excluded. For any non-numeric data type columns in the dataframe it is ignored.

EXAMPLE:

data.corr(method**=**'pearson')

OUTPUT:



HEATMAPS:

Heatmap is a way to show some sort of matrix plot. To use a heatmap the data should be in a matrix form. By matrix we mean that the index name and the column name must match in some way so that the data that we fill inside the cells are relevant.

The heatmap is a data visualization approach that uses colors in two dimensions to examine a dataset. Essentially, it demonstrates a relationship between all numerical variables in the dataset. In simplest terms, we may use the heatmaps to visualize the previously discovered association.

Lets we see the example

EXAMPLE:

# importing packages

**import** seaborn as sns

**import** matplotlib.pyplot as plt

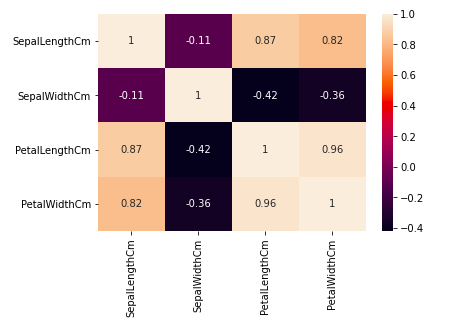
sns.heatmap(df.corr(method**=**'pearson').drop(

  ['Id'], axis**=**1).drop(['Id'], axis**=**0),

            annot **=** True);

 plt.show()

OUTPUT:



From the above graph, we can see that

* Petal breadth and length have a strong relationship.
* Petal length and sepal breadth have a strong relationship.
* Petal width and Sepal length have a strong relationship.

BOX PLOTS:

Boxplots can be used to see how the categorical value is distributed in relation to other numerical values.

Lets we see the example for the box plots

EXAMPLE:

# importing packages

**import** seaborn as sns

**import** matplotlib.pyplot as plt

**def** graph(y):

    sns.boxplot(x**=**"Species", y**=**y, data**=**df)

plt.figure(figsize**=**(10,10))

# Adding the subplot at the specified

# grid position

plt.subplot(221)

graph('SepalLengthCm')

plt.subplot(222)

graph('SepalWidthCm')

plt.subplot(223)

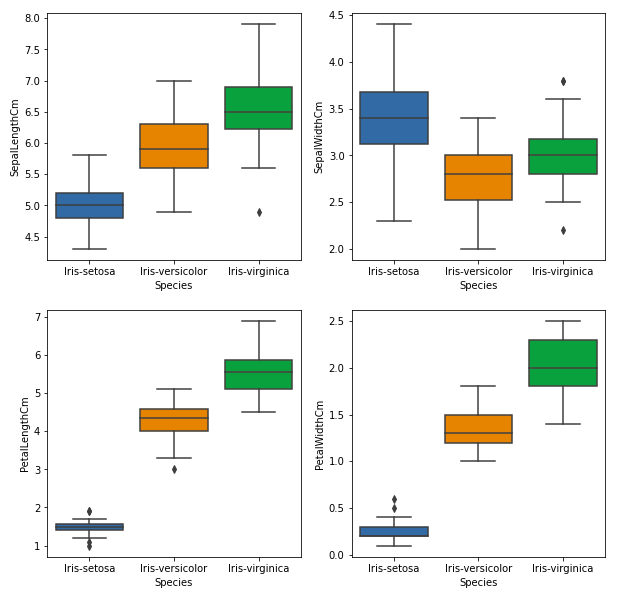
graph('PetalLengthCm')

plt.subplot(224)

graph('PetalWidthCm')

plt.show()

OUTPUT:



From the above graph, we can see that

* Setosa has the smallest characteristics and is less evenly dispersed, with several outliers.
* Species Versicolor has mediocre characteristics.
* The Virginica species possesses the best characteristics.

HANDLING OUTLIERS:

Outliers are data items or objects that differ dramatically from the rest of the (so-called normal)objects. Errors in measurement or execution can cause them. Outlier mining refers to the process of detecting outliers. There are numerous methods for detecting outliers, and the removal process is the same as removing a data item from the Panda's dataframe.

Consider the iris dataset and create a boxplot for the SepalWidthCm column.

EXAMPLE:

# importing packages

**import** seaborn as sns

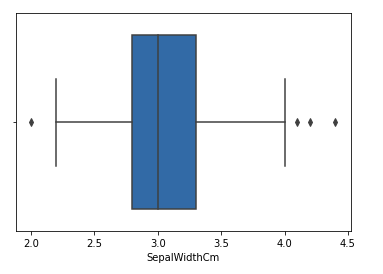
**import** matplotlib.pyplot as plt

# Load the dataset

df **=** pd.read\_csv('Iris.csv')

sns.boxplot(x**=**'SepalWidthCm', data**=**df)

OUTPUT:



In the above graph, the values above 4 and below 2 are acting as outliers.

To remove the outlier, follow the same procedure as removing an entry from the dataset using its exact position in the dataset, because the end result of all of the above methods of detecting outliers is a list of all data items that satisfy the outlier definition according to the method used.

For example, we will use IQR to locate outliers and subsequently eliminate them. We will also create a boxplot to see whether or not the outliers are removed.

Lets we see in the program

# Importing

**import** sklearn

**from** sklearn.datasets **import** load\_boston

**import** pandas as pd

**import** seaborn as sns

# Load the dataset

df **=** pd.read\_csv('Iris.csv')

# IQR

Q1 **=** np.percentile(df['SepalWidthCm'], 25,

                interpolation **=** 'midpoint')

Q3 **=** np.percentile(df['SepalWidthCm'], 75,

                interpolation **=** 'midpoint')

IQR **=** Q3 **-** Q1

print("Old Shape: ", df.shape)

# Upper bound

upper **=** np.where(df['SepalWidthCm'] >**=** (Q3**+**1.5**\***IQR))

# Lower bound

lower **=** np.where(df['SepalWidthCm'] <**=** (Q1**-**1.5**\***IQR))

# Removing the Outliers

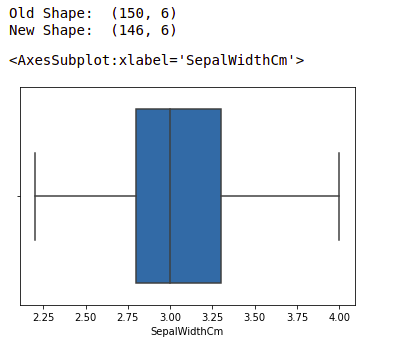
df.drop(upper[0], inplace **=** True)

df.drop(lower[0], inplace **=** True)

**print**("New Shape: ", df.shape)

sns.boxplot(x**=**'SepalWidthCm', data**=**df)

OUTPUT:



CONCLUSION:

Watson Studio, Watson Machine Learning, and Cloud Object Storage are highly powerful, well-integrated, incredibly flexible, and simple-to-use tools for data scientists and AI engineers, as demonstrated in this lesson. Watson Studio provides a collaborative environment and tools for you to work on data to solve business challenges. You can select the tools you require for data analysis and visualization, data cleansing and shaping, streaming data intake, and machine learning model creation and training.