ARTIFICAL NEURAL NETWORK (ANN) WITH PYTHON

SCIKIT LEARN / PYSPARK

DR. ALVIN ANG

CONTENTS

I. Deep Learning = Neural Network (NN) 3

II. ANN with SciKit Learn (Classifying the Iris Dataset) 4

A. Import Libraries 4

B. Load Iris Dataset 5

C. Train Test Split (80-20) 7

D. Standard Scaling 7

E. MLP Classifier 9

F. Prediction using X_testscaled 10

G. Prediction Score 10

H. Confusion Matrix 11

III. ANN with PySpark (Classifying the Iris Dataset) 12

A. Start A Spark Session 12

B. Import the Iris Dataset 13

C. Import Vector Assembler and String Indexer 14

D. Train Test Split 15

1. Training Set 15
2. Testing Set 15

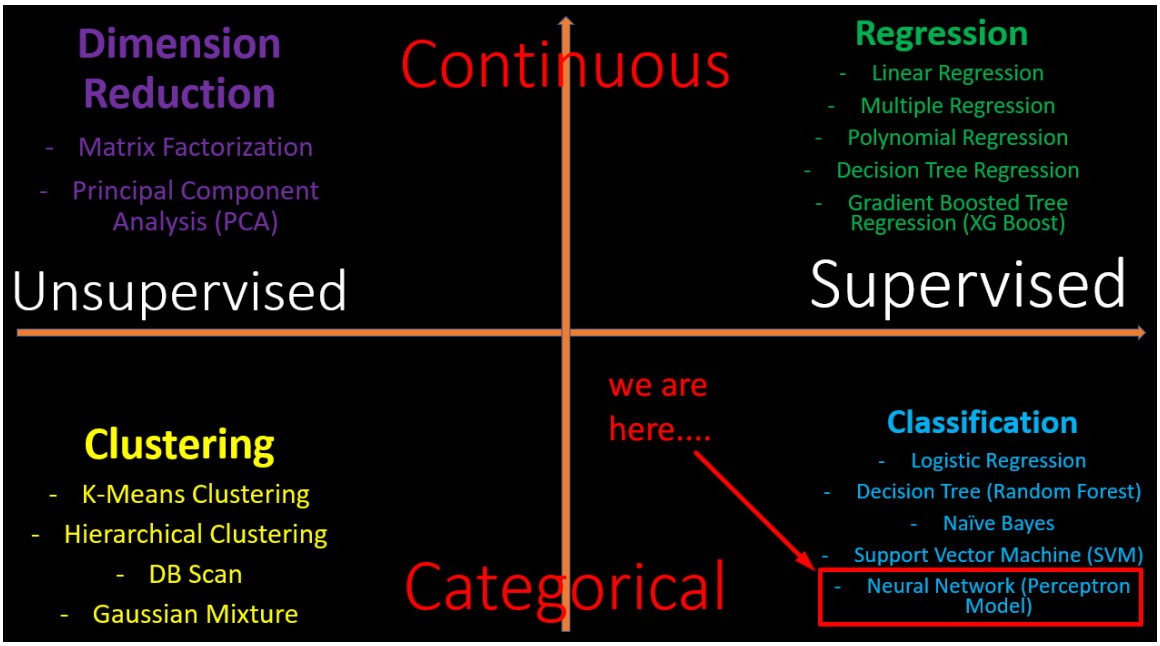
E. Multi Layer Perceptron (MLP) 16

F. Fitting MLP to Training DAta 17

G. Predicting Using the Test Dataset 18

H. Evaluation 19

About Dr. Alvin Ang 20



- Above is a table categorizing the different Machine Learning algorithms.
- Objective of Neural Network is to predict a CATEGORY.

(actually, it can also be used to predicut Regression….but most literature use it for classifying images like cats vs dogs….so we mainly use it for Classification….)

II. ANN WITH SCIKIT LEARN (CLASSIFYING THE IRIS DATASET)

Reference:

https://towardsdatascience.com/deep-neural-multilayer-perceptron-mlp-with-scikit-learn-2698e77155e

IPYNB:

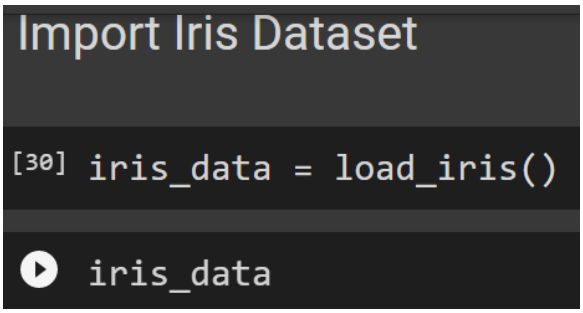
https://www.alvinang.sg/s/NN_Python_Dr_Alvin_Ang.ipynb

A. IMPORT LIBRARIES

Import Libraries

```
from sklearn.datasets import load_iris
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import pandas as pd
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
```

B. LOAD IRIS DATASET




```
{'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n-------
 'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
```

```
        [5.9, 3. , 5.1, 1.8]]),
 'data_module': 'sklearn.datasets.data',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'iris.csv',
 'frame': None,
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10')}
```

- As can be seen, Iris Dataset is a Dictionary with a lot of features.
- The 'target' is the flower type (setosa / veriscolor / virginica → 0 / 1 / 2).

```
X = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
X
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0

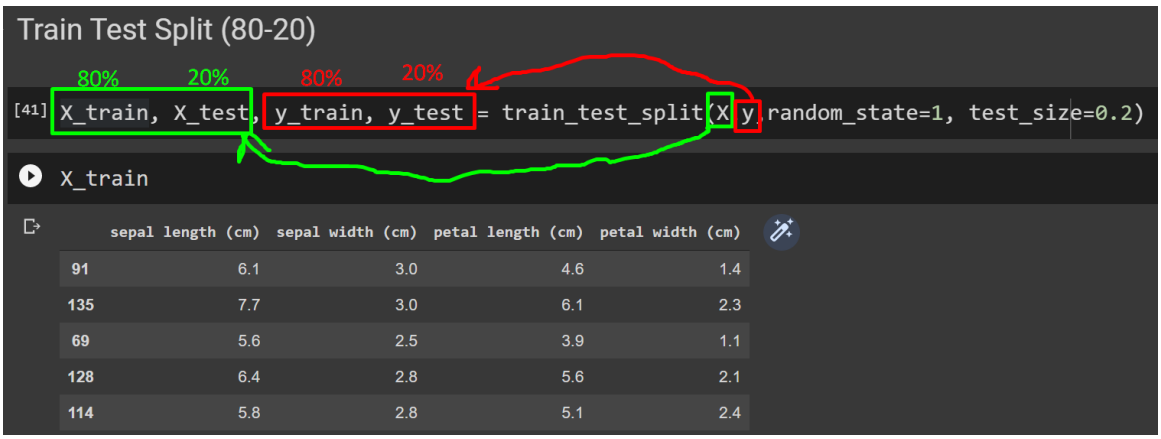
- We extract out from the Iris Data dictionary the data values as well as the column names (feature_names).

```
y = iris_data.target
pd.DataFrame(y)
```

	0
0	0
1	0
2	0
3	0
4	0

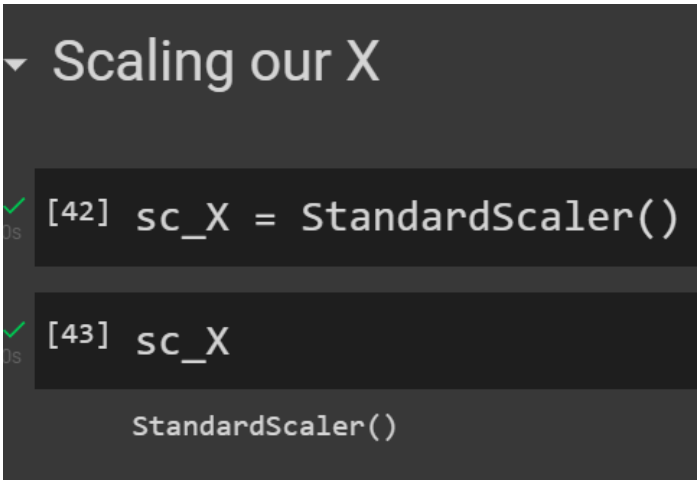
- Y is supposed to be the predicted column (called target)

C. TRAIN TEST SPLIT (80-20)



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
91	6.1	3.0	4.6	1.4
135	7.7	3.0	6.1	2.3
69	5.6	2.5	3.9	1.1
128	6.4	2.8	5.6	2.1
114	5.8	2.8	5.1	2.4

D. STANDARD SCALING



- ML algorithms are sensitive to the scale of the input data, hence, its critical to scale.

```
X_trainscaled=sc_X.fit_transform(X_train)

#Independent train and test dataset are further scaled
#to make sure that the input data is
#standard normally distributed are centred around zero
#and have variance in the same order.
```

X_train

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
91	6.1	3.0	4.6	1.4
135	7.7	3.0	6.1	2.3
69	5.6	2.5	3.9	1.1
128	6.4	2.8	5.6	2.1
114	5.8	2.8	5.1	2.4
...	...	...	...	...
133	6.3	2.8	5.1	1.5
137	6.4	3.1	5.5	1.8

Notice that after scaling, its centered Around 0…purpose so that the values Don't vary too much….

pd.DataFrame(X_trainscaled)

	0	1	2	3
0	0.315537	-0.045789	0.447675	0.233803
1	2.244933	-0.045789	1.297692	1.397429
2	-0.287400	-1.240281	0.051001	-0.154073
3	0.677298	-0.523586	1.014353	1.138845
4	-0.046225	-0.523586	0.731014	1.526721
...	...	...	...	...
115	0.556711	-0.523586	0.731014	0.363094
116	0.677298	0.193110	0.957685	0.750970

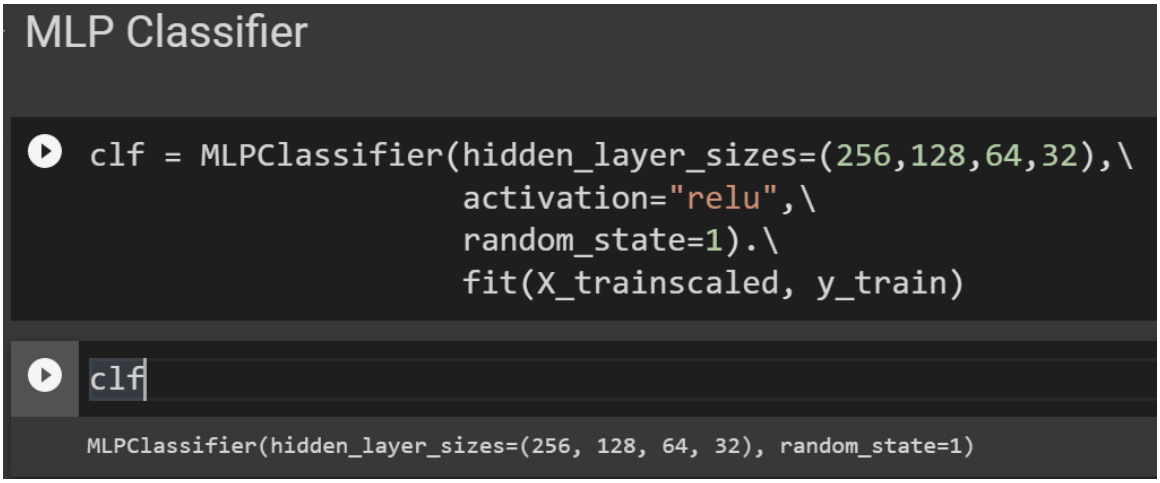
```
[48] X_testscaled=sc_X.transform(X_test)
```

```
pd.DataFrame(X_testscaled)
```

	0	1	2	3
0	-0.046225	2.343195	-1.479029	-1.317699
1	-0.890336	-1.240281	-0.459009	-0.154073
2	0.918473	-0.045789	0.334340	0.233803

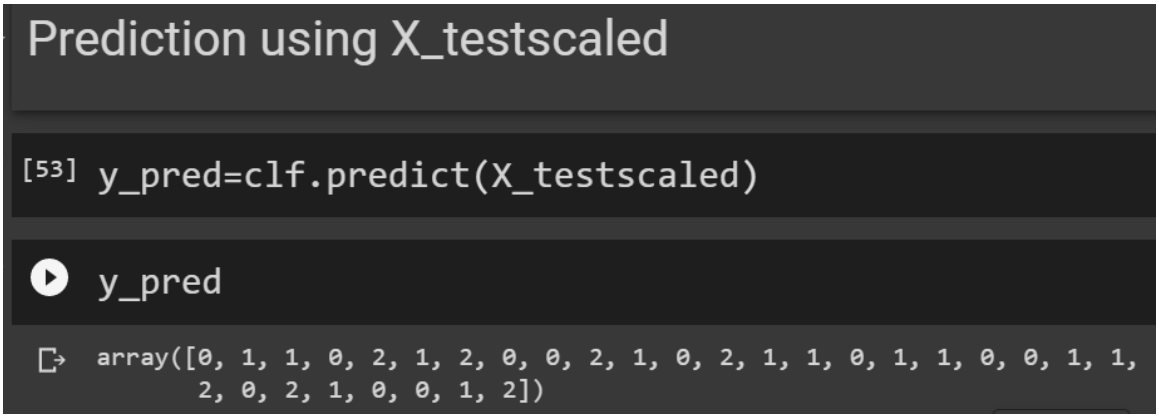
- Likewise, we scale X_test as well….

E. MLP CLASSIFIER



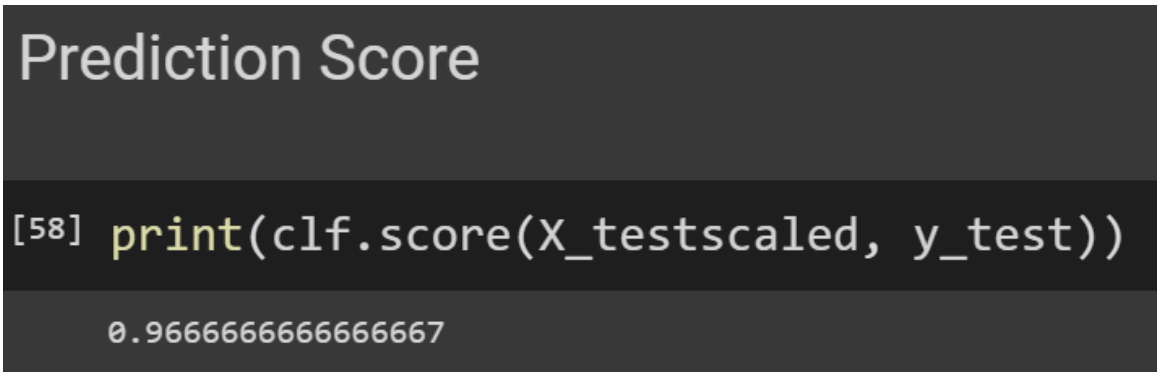
- MLP Classifier stands for Multi Layered Perceptrons…..
- Inner Layer = 256 nodes
- Hidden Layer 1 = 128 nodes
- Hidden Layer 2 = 64 nodes
- Outer Layer = 32 nodes
- Activation Unit: "ReLU" selected.
- We fit the X_trainscaled and y_train to the MLP Classifier model.

F. PREDICTION USING X_TESTSCALED



- Since the MLP model has already been trained, we make use of it to predict values inside the X_testscaled dataset.

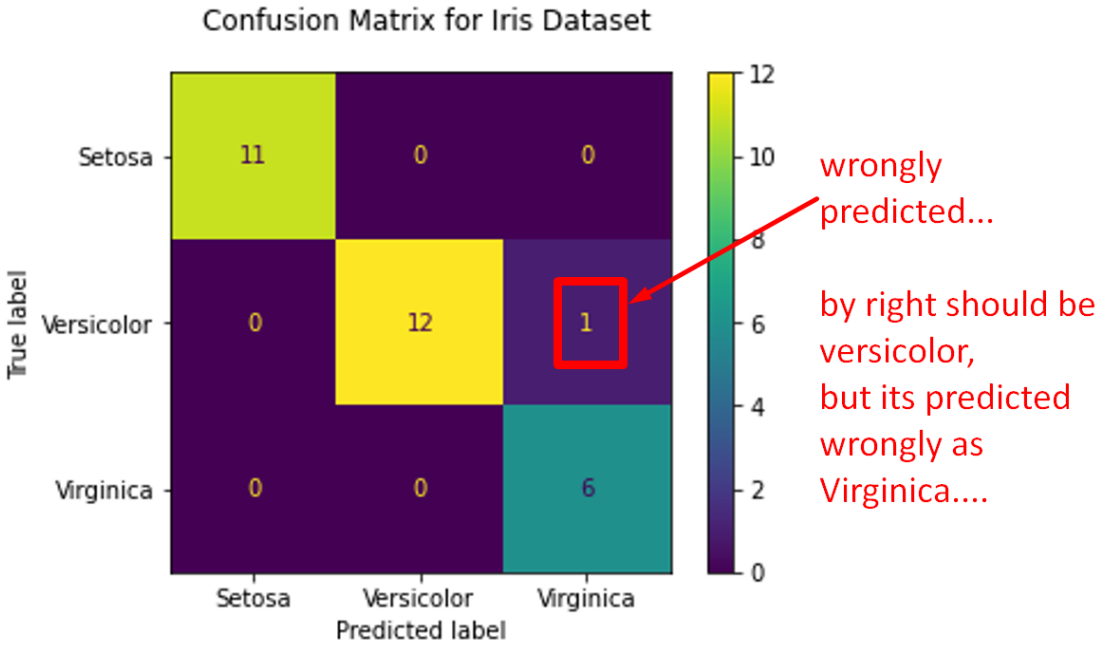
G. PREDICTION SCORE



- The prediction score is 97%!

Plotting the Confusion Matrix

```
fig=plot_confusion_matrix(clf, X_testscaled, \
                          y_test,\
                          display_labels=["Setosa","Versicolor","Virginica"])
fig.figure_.suptitle("Confusion Matrix for Iris Dataset")
plt.show()
```



III. ANN WITH PYSPARK (CLASSIFYING THE IRIS DATASET)

Reference:

https://medium.com/swlh/pysparks-multi-layer-perceptron-classifier-on-iris-dataset-dcf70d553cd8

IPYNB:

https://www.alvinang.sg/s/NN_Pyspark_Dr_Alvin_Ang.ipynb

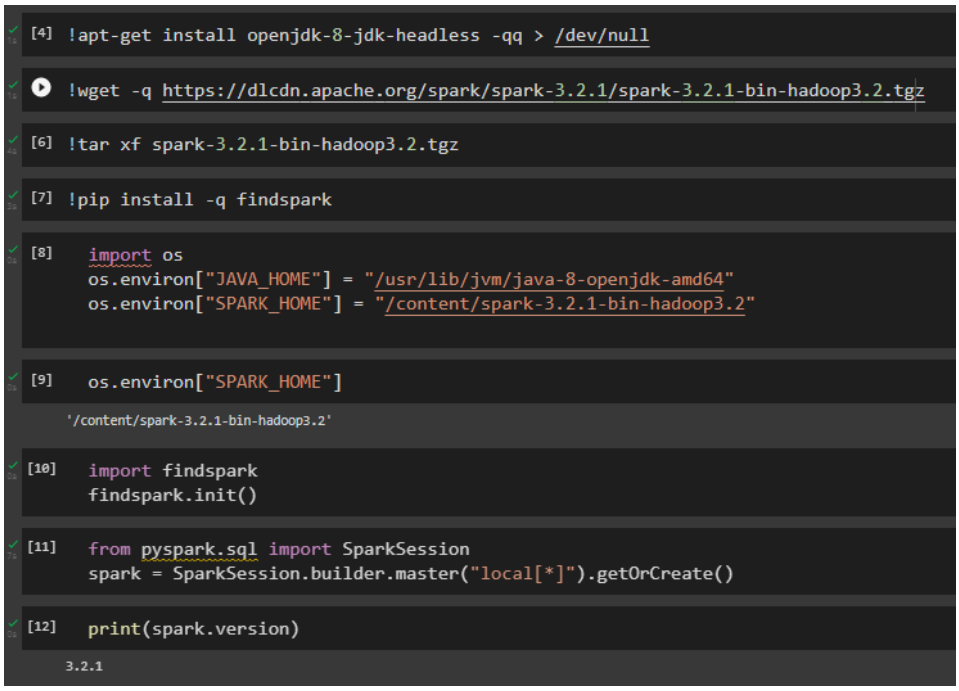
A. START A SPARK SESSION

First, you need to install PySpark into Google Colab.

Follow the steps here:

- https://tatwan.github.io/blog/colab/python/spark/2020/01/06/Colab-Spark-Instructions.html

Or….



```
import findspark
findspark.init()
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('pyspark-ml').getOrCreate()
```

B. IMPORT THE IRIS DATASET

```
from pyspark import SparkFiles

url = 'https://www.alvinang.sg/s/iris_dataset.csv'
spark.sparkContext.addFile(url)
iris_df = spark.read.csv(SparkFiles.get("iris_dataset.csv"), header=True, inferSchema=True)
```

```
iris_df.show(5)
```

```
+------------+-----------+------------+-----------+-------+
|sepal_length|sepal_width|petal_length|petal_width|species|
+------------+-----------+------------+-----------+-------+
|         5.1|        3.5|         1.4|        0.2| setosa|
|         4.9|        3.0|         1.4|        0.2| setosa|
|         4.7|        3.2|         1.3|        0.2| setosa|
|         4.6|        3.1|         1.5|        0.2| setosa|
|         5.0|        3.6|         1.4|        0.2| setosa|
+------------+-----------+------------+-----------+-------+
only showing top 5 rows
```

```
iris_df.printSchema()
```

```
root
 |-- sepal_length: double (nullable = true)
 |-- sepal_width: double (nullable = true)
 |-- petal_length: double (nullable = true)
 |-- petal_width: double (nullable = true)
 |-- species: string (nullable = true)
```

C. IMPORT VECTOR ASSEMBLER AND STRING INDEXER

```
from pyspark.sql.functions import *
from pyspark.ml.feature import VectorAssembler, StringIndexer
```

```
vectorAssembler = VectorAssembler(inputCols = ['sepal_length',\
                                               'sepal_width',\
                                               'petal_length',\
                                               'petal_width'], \
                                  outputCol = 'features')

v_iris_df = vectorAssembler.transform(iris_df)

v_iris_df.show(5)
```

```
+------------+-----------+------------+-----------+-------+-----------------+
|sepal_length|sepal_width|petal_length|petal_width|species|         features|
+------------+-----------+------------+-----------+-------+-----------------+
|         5.1|        3.5|         1.4|        0.2| setosa|[5.1,3.5,1.4,0.2]|
|         4.9|        3.0|         1.4|        0.2| setosa|[4.9,3.0,1.4,0.2]|
|         4.7|        3.2|         1.3|        0.2| setosa|[4.7,3.2,1.3,0.2]|
|         4.6|        3.1|         1.5|        0.2| setosa|[4.6,3.1,1.5,0.2]|
|         5.0|        3.6|         1.4|        0.2| setosa|[5.0,3.6,1.4,0.2]|
+------------+-----------+------------+-----------+-------+-----------------+
only showing top 5 rows
```

features column has been created

its just a combination of the first 4 columns

```
indexer = StringIndexer(inputCol = 'species', outputCol = 'label')

i_v_iris_df = indexer.fit(v_iris_df).transform(v_iris_df)

i_v_iris_df.show(5)
```

```
+------------+-----------+------------+-----------+-------+-----------------+-----+
|sepal_length|sepal_width|petal_length|petal_width|species|         features|label|
+------------+-----------+------------+-----------+-------+-----------------+-----+
|         5.1|        3.5|         1.4|        0.2| setosa|[5.1,3.5,1.4,0.2]|  0.0|
|         4.9|        3.0|         1.4|        0.2| setosa|[4.9,3.0,1.4,0.2]|  0.0|
|         4.7|        3.2|         1.3|        0.2| setosa|[4.7,3.2,1.3,0.2]|  0.0|
|         4.6|        3.1|         1.5|        0.2| setosa|[4.6,3.1,1.5,0.2]|  0.0|
|         5.0|        3.6|         1.4|        0.2| setosa|[5.0,3.6,1.4,0.2]|  0.0|
+------------+-----------+------------+-----------+-------+-----------------+-----+
only showing top 5 rows
```

StringIndexer creates a new 'label' column...using the 'species' column

```
i_v_iris_df.select('species','label').groupBy('species','label').count().show()
```

```
+----------+-----+-----+
|   species|label|count|
+----------+-----+-----+
|    setosa|  0.0|   50|
| virginica|  2.0|   50|
|versicolor|  1.0|   50|
+----------+-----+-----+
```

- **We have 50 of each type.**

D. TRAIN TEST SPLIT

```
Train Test Split

[ ] splits = i_v_iris_df.randomSplit([0.6,0.4],1)

    (98, 52, 150)
```

- We split (randomly), the data 60% into "Training" set.
- 40% into "Testing" set.

1. TRAINING SET

```
train_df = splits[0]

train_df.show()
```

```
+------------+-----------+------------+-----------+----------+-----------------+-----+
|sepal_length|sepal_width|petal_length|petal_width|   species|         features|label|
+------------+-----------+------------+-----------+----------+-----------------+-----+
|         4.4|        2.9|         1.4|        0.2|    setosa|[4.4,2.9,1.4,0.2]|  0.0|
|         4.4|        3.0|         1.3|        0.2|    setosa|[4.4,3.0,1.3,0.2]|  0.0|
|         4.4|        3.2|         1.3|        0.2|    setosa|[4.4,3.2,1.3,0.2]|  0.0|
|         4.6|        3.2|         1.4|        0.2|    setosa|[4.6,3.2,1.4,0.2]|  0.0|
|         4.6|        3.4|         1.4|        0.3|    setosa|[4.6,3.4,1.4,0.3]|  0.0|
|         4.6|        3.6|         1.0|        0.2|    setosa|[4.6,3.6,1.0,0.2]|  0.0|
|         4.7|        3.2|         1.6|        0.2|    setosa|[4.7,3.2,1.6,0.2]|  0.0|
|         4.8|        3.0|         1.4|        0.1|    setosa|[4.8,3.0,1.4,0.1]|  0.0|
|         4.8|        3.0|         1.4|        0.3|    setosa|[4.8,3.0,1.4,0.3]|  0.0|
|         4.9|        3.0|         1.4|        0.2|    setosa|[4.9,3.0,1.4,0.2]|  0.0|
|         4.9|        3.1|         1.5|        0.1|    setosa|[4.9,3.1,1.5,0.1]|  0.0|
|         4.9|        3.1|         1.5|        0.1|    setosa|[4.9,3.1,1.5,0.1]|  0.0|
|         4.9|        3.1|         1.5|        0.1|    setosa|[4.9,3.1,1.5,0.1]|  0.0|
|         5.0|        2.0|         3.5|        1.0|versicolor|[5.0,2.0,3.5,1.0]|  1.0|
|         5.0|        2.3|         3.3|        1.0|versicolor|[5.0,2.3,3.3,1.0]|  1.0|
```

2. TESTING SET

```
[21] test_df = splits[1]

     test_df.show()
```

```
+------------+-----------+------------+-----------+----------+-----------------+-----+
|sepal_length|sepal_width|petal_length|petal_width|   species|         features|label|
+------------+-----------+------------+-----------+----------+-----------------+-----+
|         4.3|        3.0|         1.1|        0.1|    setosa|[4.3,3.0,1.1,0.1]|  0.0|
|         4.5|        2.3|         1.3|        0.3|    setosa|[4.5,2.3,1.3,0.3]|  0.0|
|         4.6|        3.1|         1.5|        0.2|    setosa|[4.6,3.1,1.5,0.2]|  0.0|
|         4.7|        3.2|         1.3|        0.2|    setosa|[4.7,3.2,1.3,0.2]|  0.0|
|         4.8|        3.1|         1.6|        0.2|    setosa|[4.8,3.1,1.6,0.2]|  0.0|
|         4.8|        3.4|         1.6|        0.2|    setosa|[4.8,3.4,1.6,0.2]|  0.0|
|         4.8|        3.4|         1.9|        0.2|    setosa|[4.8,3.4,1.9,0.2]|  0.0|
|         4.9|        2.4|         3.3|        1.0|versicolor|[4.9,2.4,3.3,1.0]|  1.0|
|         4.9|        2.5|         4.5|        1.7| virginica|[4.9,2.5,4.5,1.7]|  2.0|
|         5.0|        3.5|         1.3|        0.3|    setosa|[5.0,3.5,1.3,0.3]|  0.0|
|         5.0|        3.6|         1.4|        0.2|    setosa|[5.0,3.6,1.4,0.2]|  0.0|
|         5.1|        2.5|         3.0|        1.1|versicolor|[5.1,2.5,3.0,1.1]|  1.0|
```

```
train_df.count(), test_df.count(), i_v_iris_df.count()
```

```
(98, 52, 150)
```

- 98 rows of Training data
- 52 rows of Testing data
- Total 150 rows.

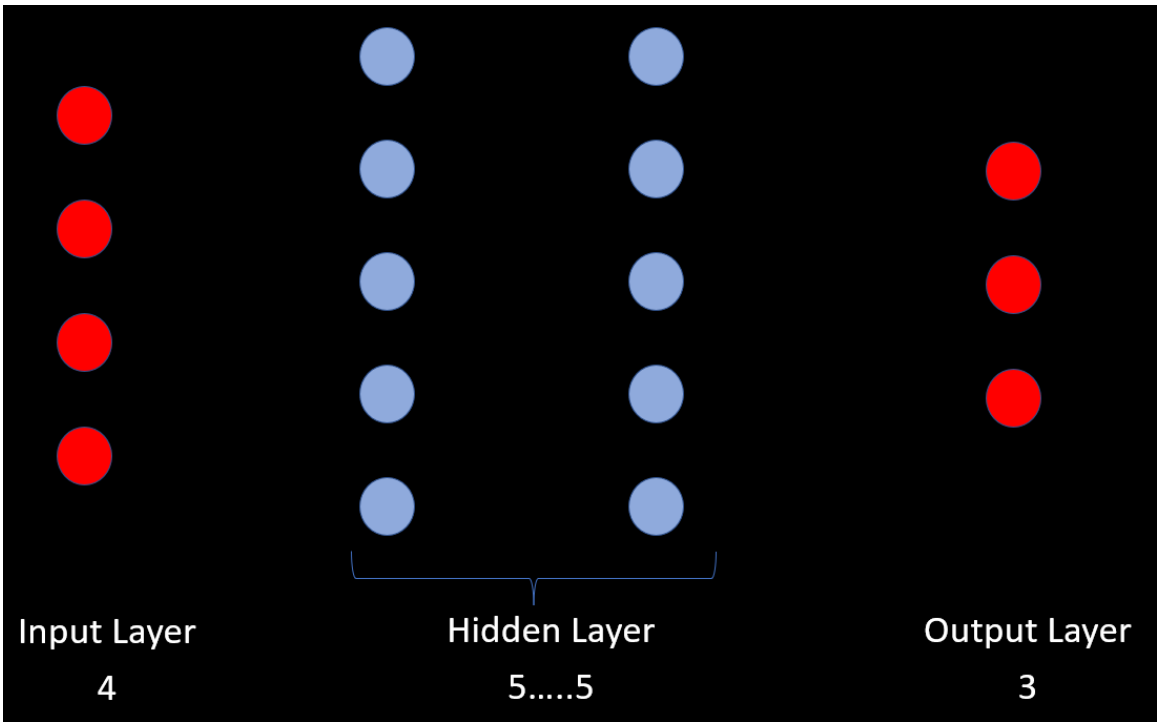
E. MULTI LAYER PERCEPTRON (MLP)

Import MultilayerPerceptronClassifier

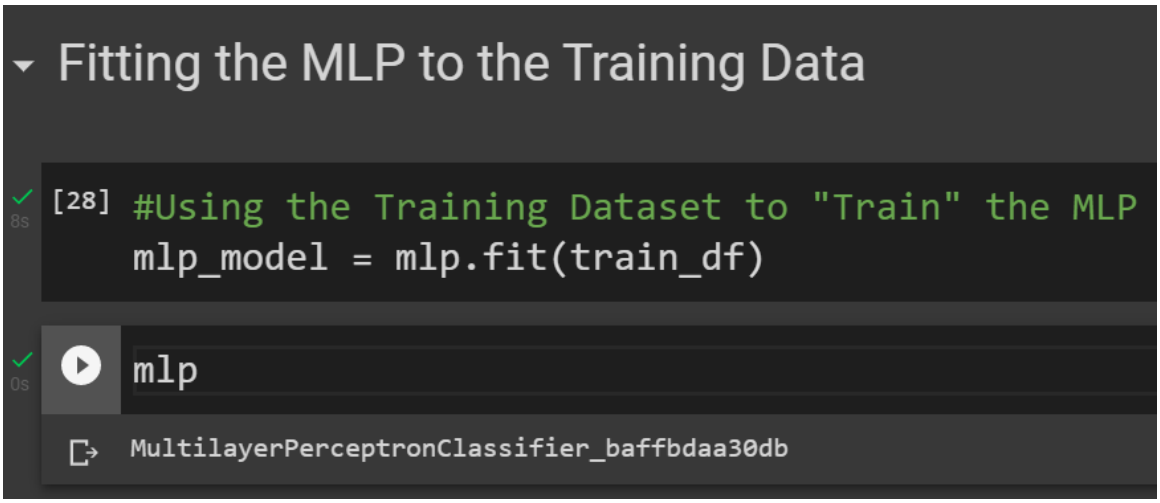
```
[24] from pyspark.ml.classification import MultilayerPerceptronClassifier
```

```
[26] layers = [4,5,5,3]

    #We are using two hidden layers of 5 nodes each
    #and hence our layers array is [4,5,5,3]
    #(input-4, 2 x hidden-5, output nodes-3)
```

```
mlp = MultilayerPerceptronClassifier(layers = layers, seed = 1)

#We are passing this along with a seed value of 1,
#just to replicate the same results in different runs.
```

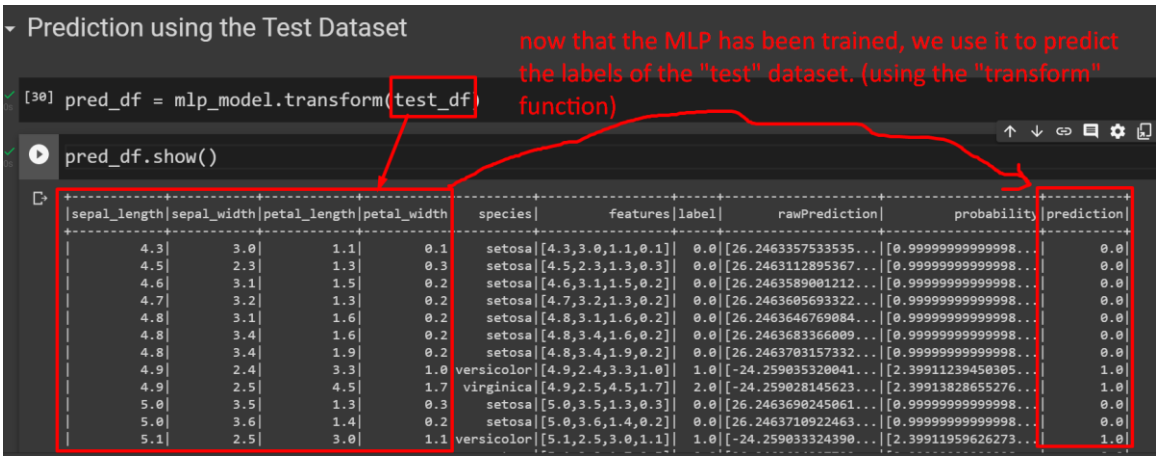


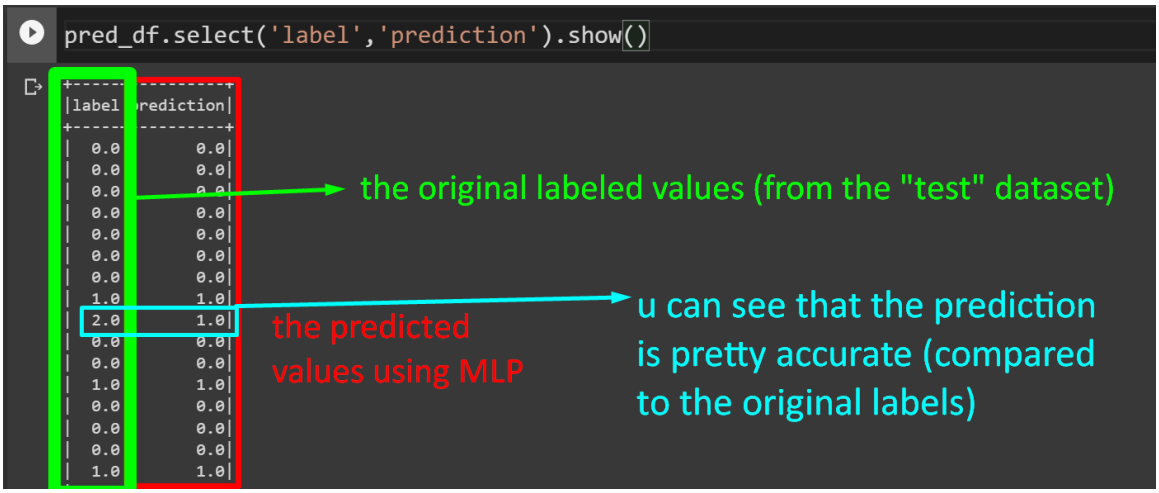
F. FITTING MLP TO TRAINING DATA



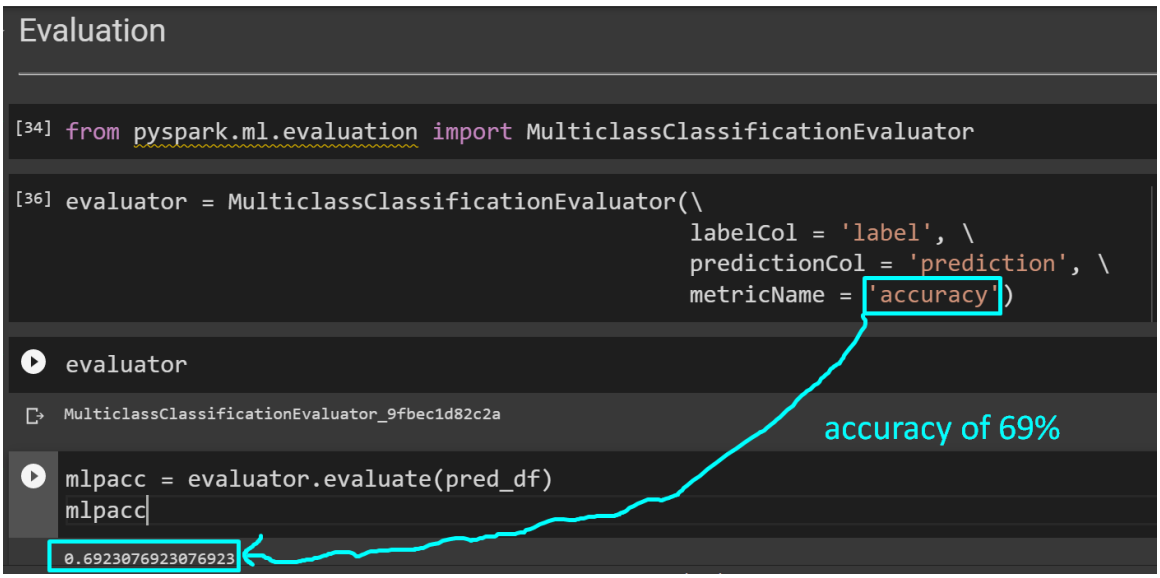
- **We make use of the Training Data to train the MLP**

G. PREDICTING USING THE TEST DATASET





H. EVALUATION



ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.