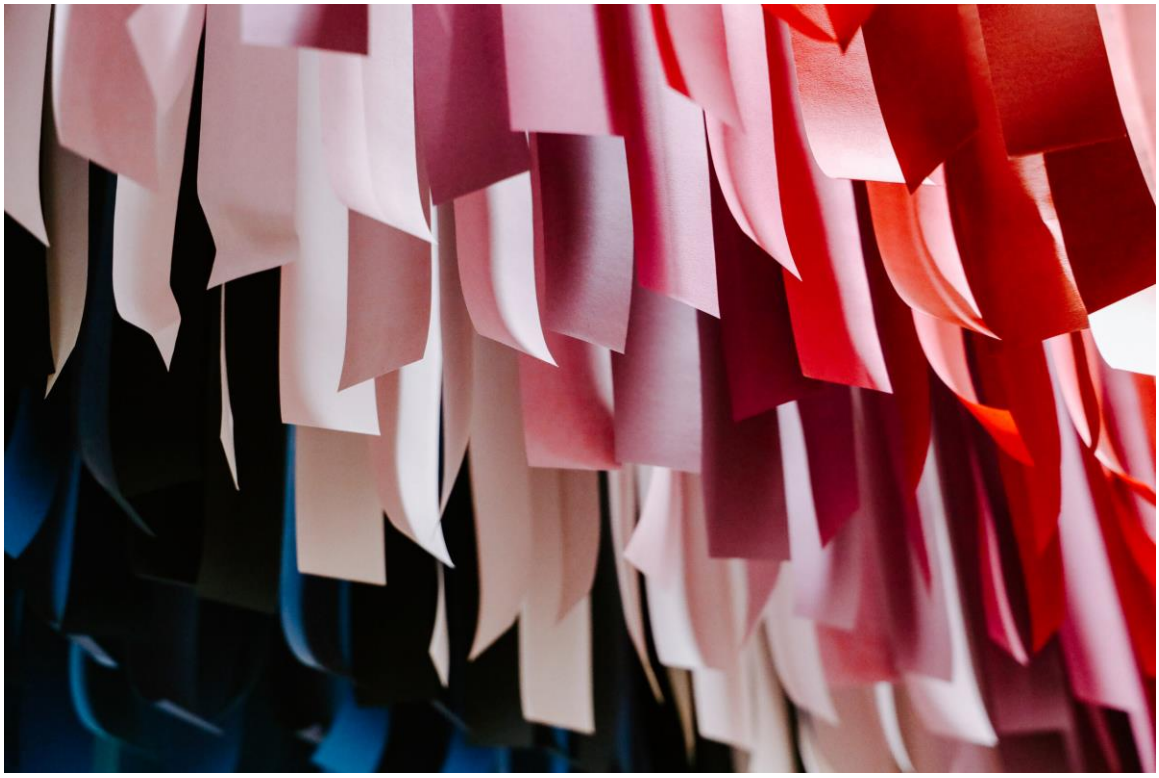
K MEANS CLUSTERING

USING PYSPARK

DR. ALVIN ANG



CONTENTS

I. K MEANS USING PYSPARK – CASE I (SALES DATASET)

- File can be found here: https://www.alvinang.sg/s/sales.csv
- IPYNB: https://www.alvinang.sg/s/KMEANS_PySpark_by_Dr_Alvin.ipynb
- Go here to learn about K Means first: https://www.alvinang.sg/s/K-Means-Clustering-with-Python-by-Dr-Alvin-Ang.pdf

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….

```
[4] !apt-get install openjdk-8-jdk-headless -qq > /dev/null

!wget -q https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

[6] !tar xf spark-3.2.1-bin-hadoop3.2.tgz

[7] !pip install -q findspark

[8] import os
    os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
    os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"

[9] os.environ["SPARK_HOME"]

'/content/spark-3.2.1-bin-hadoop3.2'

[10] import findspark
     findspark.init()

[11] from pyspark.sql import SparkSession
     spark = SparkSession.builder.master("local[*]").getOrCreate()

[12] print(spark.version)

3.2.1
```

B. IMPORT THE DATA

```
from pyspark import SparkFiles

url = 'https://www.alvinang.sg/s/sales.csv'
spark.sparkContext.addFile(url)

df_sales = spark.read.csv(SparkFiles.get("sales.csv"), \
                          header=True, inferSchema=True)
df_sales

DataFrame[RowID: int, OrderID: int, OrderDate: string, OrderMonthYear: string, Qua
```

C. TRANSFORMING THE DATA

```
[10] from pyspark.ml.feature import VectorAssembler

     vectorAssembler = VectorAssembler(
         inputCols = ['Quantity', 'Quote', 'HourlyWage'],
         outputCol = 'features'
     )
```

```
df_sales1 = vectorAssembler.transform(df_sales)

df_sales1.select(['Quantity','Quote','HourlyWage','features']).show()
```

```
+--------+-----+----------+-----------------+
|Quantity|Quote|HourlyWage|         features|
+--------+-----+----------+-----------------+
|       6| 1200|        60| [6.0,1200.0,60.0]|
|       2|  280|        75|  [2.0,280.0,75.0]|
|      26| 3250|        75|[26.0,3250.0,75.0]|
|      24| 3000|        53|[24.0,3000.0,53.0]|
|      23| 3450|        50|[23.0,3450.0,50.0]|
|      15| 2250|        62|[15.0,2250.0,62.0]|
|      30| 3600|        62|[30.0,3600.0,62.0]|
|      14| 2100|        67|[14.0,2100.0,67.0]|
|      46| 6900|        51|[46.0,6900.0,51.0]|
|      32| 4800|        68|[32.0,4800.0,68.0]|
|      41| 4920|        70|[41.0,4920.0,70.0]|
```

D. FITTING THE DATA TO KMEANS

```
from pyspark.ml.clustering import KMeans

kmeans = KMeans().setK(3) # set 3 groups

model_sales = kmeans.fit(df_sales1)
#this training / learning / fitting will only use vectors in 'features' column
```

E. GETTING THE CENTROIDS

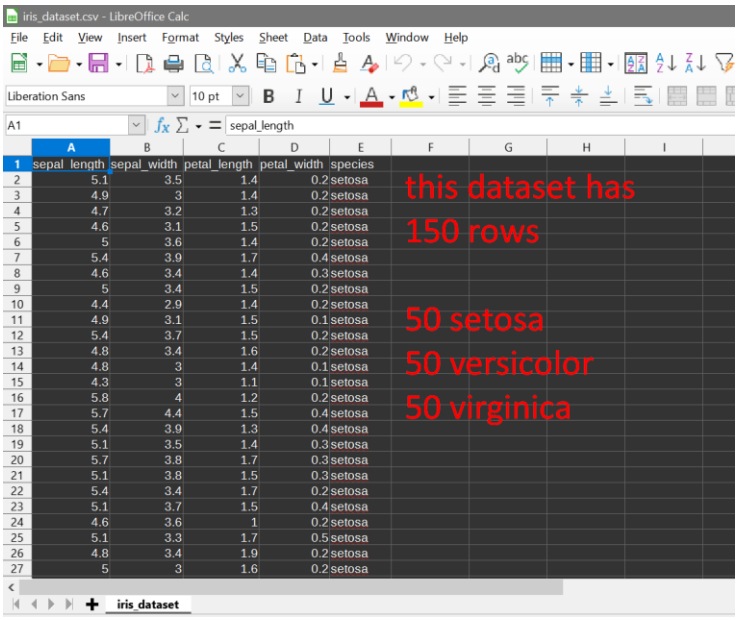
```
[14] centers = model_sales.clusterCenters()

     import numpy as np
     centers.sort(key=np.linalg.norm)
     #sort by increasing distance of centroid from origin

     display(centers)
```

```
[array([   9.85691004, 1323.95697523,   61.1059322 ]),
 array([  29.30902232, 3912.68783401,   61.10091166]),
 array([  42.46697674, 6515.16744186,   60.62697674])]
```

II. K MEANS USING PYSPARK – CASE II (IRIS DATASET)

- Dataset can be found here: https://www.alvinang.sg/s/iris_dataset.csv
- IPYNB: https://www.alvinang.sg/s/KMEANS_PySpark_by_Dr_Alvin.ipynb

A. READING IN THE DATA



	sepal_length	sepal_width	petal_length	petal_width	species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa
11	4.9	3.1	1.5	0.1	setosa
12	5.4	3.7	1.5	0.2	setosa
13	4.8	3.4	1.6	0.2	setosa
14	4.8	3	1.4	0.1	setosa
15	4.3	3	1.1	0.1	setosa
16	5.8	4	1.2	0.2	setosa
17	5.7	4.4	1.5	0.4	setosa
18	5.4	3.9	1.3	0.4	setosa
19	5.1	3.5	1.4	0.3	setosa
20	5.7	3.8	1.7	0.3	setosa
21	5.1	3.8	1.5	0.3	setosa
22	5.4	3.4	1.7	0.2	setosa
23	5.1	3.7	1.5	0.4	setosa
24	4.6	3.6	1	0.2	setosa
25	5.1	3.3	1.7	0.5	setosa
26	4.8	3.4	1.9	0.2	setosa
27	5	3	1.6	0.2	setosa

```
[15] from pyspark import SparkFiles

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

```
print((df.count(), len(df.columns)))
```

```
(150, 5)
```

B. GLANCE THE DATA

```
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)
```

```
[18] from pyspark.sql.functions import rand
```

```
df.orderBy(rand()).show(10,False)
```

```
+------------+-----------+------------+-----------+----------+
|sepal_length|sepal_width|petal_length|petal_width|species   |
+------------+-----------+------------+-----------+----------+
|5.5         |2.5        |4.0         |1.3        |versicolor|
|7.7         |3.8        |6.7         |2.2        |virginica |
|6.0         |2.9        |4.5         |1.5        |versicolor|
|5.0         |3.6        |1.4         |0.2        |setosa    |
|5.7         |2.8        |4.1         |1.3        |versicolor|
|7.2         |3.2        |6.0         |1.8        |virginica |
|5.5         |2.4        |3.8         |1.1        |versicolor|
|4.8         |3.4        |1.9         |0.2        |setosa    |
|6.3         |3.3        |6.0         |2.5        |virginica |
|6.2         |2.9        |4.3         |1.3        |versicolor|
+------------+-----------+------------+-----------+----------+
```

```
[20] df.groupBy('species').count().orderBy('count').show(10,False)
```

```
+----------+-----+
|species   |count|
+----------+-----+
|virginica |50   |
|versicolor|50   |
|setosa    |50   |
+----------+-----+
```

C. TRANSFORM THE DATA

```
[21] from pyspark.ml.linalg import Vector
     from pyspark.ml.feature import VectorAssembler

[22] input_cols=['sepal_length', 'sepal_width','petal_length', 'petal_width']
     input_cols

     ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']

     vec_assembler = VectorAssembler(inputCols = input_cols, outputCol="features")
     vec_assembler

     VectorAssembler_259df7e3820f
```

```
[24] final_data = vec_assembler.transform(df)
     final_data.show()
```

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]
5.4	3.9	1.7	0.4	setosa	[5.4,3.9,1.7,0.4]
4.6	3.4	1.4	0.3	setosa	[4.6,3.4,1.4,0.3]
5.0	3.4	1.5	0.2	setosa	[5.0,3.4,1.5,0.2]
4.4	2.9	1.4	0.2	setosa	[4.4,2.9,1.4,0.2]
4.9	3.1	1.5	0.1	setosa	[4.9,3.1,1.5,0.1]
5.4	3.7	1.5	0.2	setosa	[5.4,3.7,1.5,0.2]
4.8	3.4	1.6	0.2	setosa	[4.8,3.4,1.6,0.2]
4.8	3.0	1.4	0.1	setosa	[4.8,3.0,1.4,0.1]
4.3	3.0	1.1	0.1	setosa	[4.3,3.0,1.1,0.1]

we simply put all 4 columns into 1 array and call it "features"

D. ELBOW METHOD : FINDNG APPROPRIATE NUMBER OF K'S

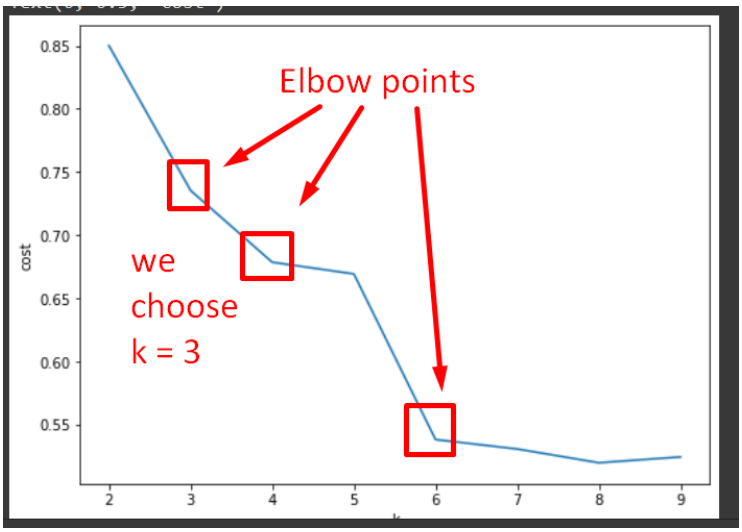
```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
silhouette_score=[]
evaluator = ClusteringEvaluator()

for k in range(2,10):

    kmeans=KMeans(featuresCol='features', k=k)
    model=kmeans.fit(final_data)
    intra_distance=model.transform(final_data)

    score=evaluator.evaluate(intra_distance)
    silhouette_score.append(score)

    print("Silhouette Score:",score)
```

```
Silhouette Score: 0.8501515983265806
Silhouette Score: 0.7354567373091194
Silhouette Score: 0.6786090104505119
Silhouette Score: 0.6693502650358714
Silhouette Score: 0.5382053692938615
Silhouette Score: 0.5307297724052582
Silhouette Score: 0.5198141093898501
Silhouette Score: 0.5244134122188032
```

- You can see that the score is decreasing as the k increases.
- Because the more number of k (centroids / clusters), the lower the score (or error).
- Error being defined as the distance from the Centroid to the other points.

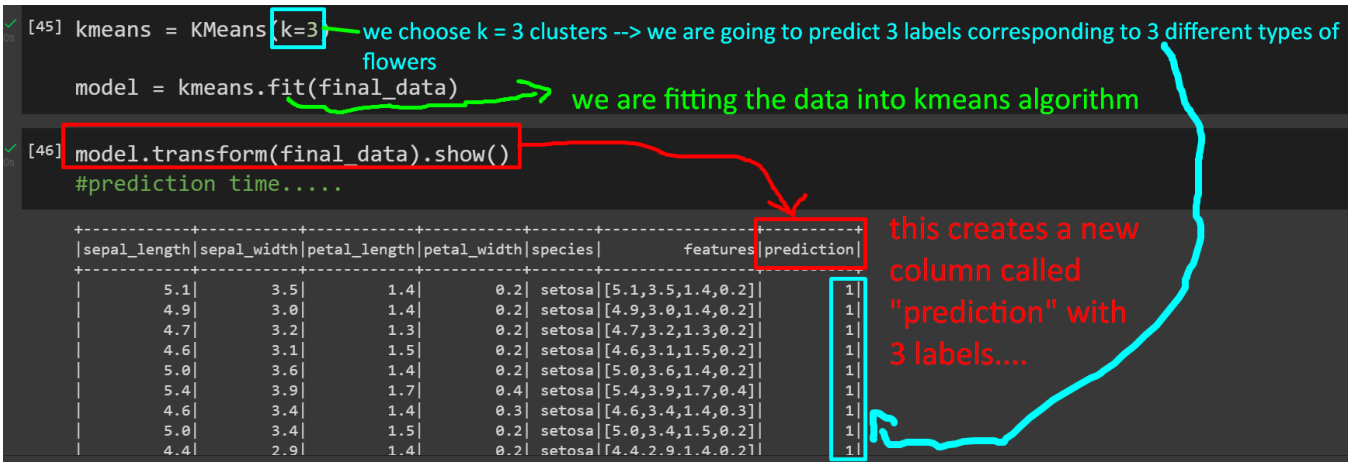
E. PLOTTING THE SILHOUETTE AND FINDING THE ELBOW

```
#Visualizing the silhouette scores in a plot
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,10),silhouette_score)
ax.set_xlabel('k')
ax.set_ylabel('cost')
```



- There seems to be an Elbow formation at k = 3 (means we choose 3 clusters).
- There are more significant elbows at 4 and 6 but we stick to 3 because we already know beforehand that there are 3 types of flowers.
- And also because we don't want too many clusters.

F. FITTING THE DATA TO KMEANS THEN PREDICTION USING BACK SAME DATASET



sepal_length	sepal_width	petal_length	petal_width	species	features	prediction
5.1	3.5	1.4	0.2	setosa	[5.1,3.5,1.4,0.2]	1
4.9	3.0	1.4	0.2	setosa	[4.9,3.0,1.4,0.2]	1
4.7	3.2	1.3	0.2	setosa	[4.7,3.2,1.3,0.2]	1
4.6	3.1	1.5	0.2	setosa	[4.6,3.1,1.5,0.2]	1
5.0	3.6	1.4	0.2	setosa	[5.0,3.6,1.4,0.2]	1
5.4	3.9	1.7	0.4	setosa	[5.4,3.9,1.7,0.4]	1
4.6	3.4	1.4	0.3	setosa	[4.6,3.4,1.4,0.3]	1
5.0	3.4	1.5	0.2	setosa	[5.0,3.4,1.5,0.2]	1
4.4	2.9	1.4	0.2	setosa	[4.4,2.9,1.4,0.2]	1

- It might be a little confusing because you might ask "what's the point of predicting back an old dataset when the species / labels / categories are already known?"
- That's because we are using it to tally the species to the predicted labels.
    - E.g. 1 refers to setosa…. 2 refers to virginica…etc…
- Later in the next section, we will use a new dataset to test the prediction.

```
[47] model.transform(final_data).groupBy('prediction').count().show()

+----------+-----+
|prediction|count|
+----------+-----+
|         1|   50|
|         2|   38|
|         0|   62|
+----------+-----+
```

the kmean prediction algorithm doesn't work too well.... there's supposed to be 50 of each flower type... end up 50 / 38 / 62.....

```
predictions=model.transform(final_data)

predictions.groupBy('species','prediction').count().show()

#K-means can produce different results every time as it chooses the starting point
#(centroid) randomly every time. Hence, the results that you might get
#in you K-means clustering might be totally different from these results

+----------+----------+-----+
|   species|prediction|count|
+----------+----------+-----+
| virginica|         2|   36|
| virginica|         0|   14|
|versicolor|         0|   48|
|    setosa|         1|   50|
|versicolor|         2|    2|
+----------+----------+-----+
```

Type 0 = virginica / versi = 14+48 = 62 (wrong)
Type 1 = setosa = 50 (correct prediction)
Type 2 = virginica / versi = 36+2=38 (wrong)

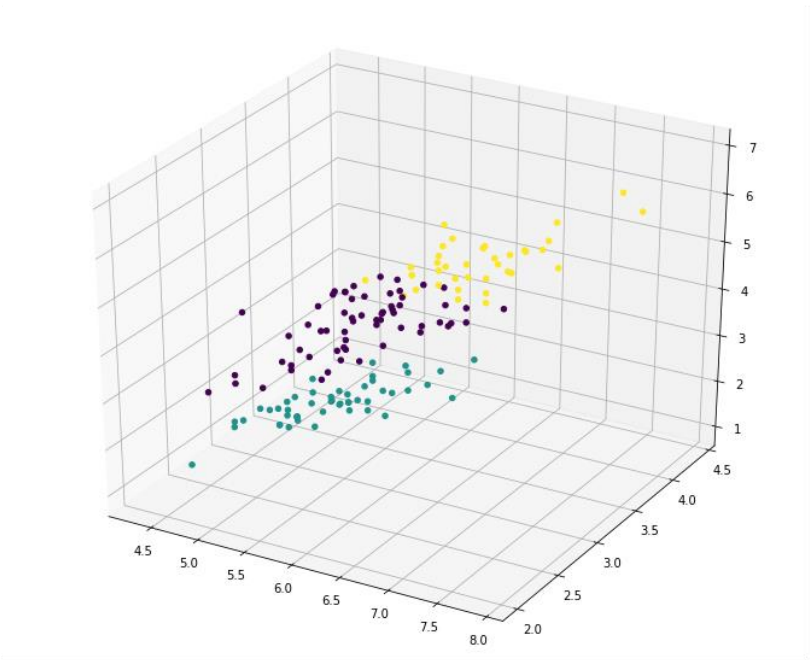
```
pandas_df = predictions.toPandas()
pandas_df
```

convert to pandas dataframe...

	sepal_length	sepal_width	petal_length	petal_width	species	features	prediction
0	5.1	3.5	1.4	0.2	setosa	[5.1, 3.5, 1.4, 0.2]	1
1	4.9	3.0	1.4	0.2	setosa	[4.9, 3.0, 1.4, 0.2]	1
2	4.7	3.2	1.3	0.2	setosa	[4.7, 3.2, 1.3, 0.2]	1
3	4.6	3.1	1.5	0.2	setosa	[4.6, 3.1, 1.5, 0.2]	1
4	5.0	3.6	1.4	0.2	setosa	[5.0, 3.6, 1.4, 0.2]	1
...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	[6.7, 3.0, 5.2, 2.3]	2
146	6.3	2.5	5.0	1.9	virginica	[6.3, 2.5, 5.0, 1.9]	0
147	6.5	3.0	5.2	2.0	virginica	[6.5, 3.0, 5.2, 2.0]	2

G. PLOTTING

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

cluster_vis = plt.figure(figsize=(12,10)).gca(projection='3d')

cluster_vis.scatter(pandas_df.sepal_length,pandas_df.sepal_width, pandas_df.petal_length, \
                    c=pandas_df.prediction,depthshade=False)

plt.show()
```



- You see 3 clusters.

H. PREDICTING USING FAKE DATA

Dataset is here: https://www.alvinang.sg/s/iris_dataset_prediction.csv

	A	B	C	D	E
1	sepal_length	sepal_width	petal_length	petal_width	species
2	5.1	3.5	1.4	0.2	???

1. IMPORT AND TRANSFORM THE DATA

```
[30] from pyspark import SparkFiles

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

```
input_cols_pred=['sepal_length', 'sepal_width','petal_length', 'petal_width']
vec_assembler_pred = VectorAssembler(inputCols = input_cols_pred, outputCol="features")
final_data_pred = vec_assembler_pred.transform(df_pred)
final_data_pred.show()
```

```
+------------+-----------+------------+-----------+-------+-----------------+
|sepal_length|sepal_width|petal_length|petal_width|species|         features|
+------------+-----------+------------+-----------+-------+-----------------+
|         5.1|        3.5|         1.4|        0.2|    ???|[5.1,3.5,1.4,0.2]|
+------------+-----------+------------+-----------+-------+-----------------+
```

2. PREDICT

```
[32] prediction_pred=model.transform(final_data_pred)
```

```
pandas_df_pred = prediction_pred.toPandas()
pandas_df_pred
```

	sepal_length	sepal_width	petal_length	petal_width	species	features	prediction
0	5.1	3.5	1.4	0.2	???	[5.1, 3.5, 1.4, 0.2]	1

we don't know the species
but the predicted label is 1

which tallies with "setosa"

- Given the data set above, Kmeans predicted the species to be Setosa.

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.