## Build a Classification Model with Spark with iris dataset

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import StringIndexer, VectorAssembler, IndexToString
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import urllib.request

# Initialize Spark session
spark = SparkSession.builder.appName("IrisClassification").getOrCreate()

# Download the dataset
data_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
iris_path = "/tmp/iris.csv"
urllib.request.urlretrieve(data_url, iris_path)

# Load dataset into DataFrame
df = spark.read.csv(iris_path, inferSchema=True)\
    .toDF("sepal_length", "sepal_width", "petal_length", "petal_width", "species")

# Step 1: Encode species as numerical label
indexer = StringIndexer(inputCol="species", outputCol="label")
indexer_model = indexer.fit(df)
df = indexer_model.transform(df)

# Step 2: Assemble features into a single vector
assembler = VectorAssembler(
    inputCols=["sepal_length", "sepal_width", "petal_length", "petal_width"],
    outputCol="features"
)
df = assembler.transform(df)

# Step 3: Train/test split
train, test = df.randomSplit([0.7, 0.3], seed=42)

# Step 4: Train logistic regression model
lr = LogisticRegression(featuresCol="features", labelCol="label")
model = lr.fit(train)

# Step 5: Make predictions
predictions = model.transform(test)

# Step 6: Convert prediction index back to species name
label_converter = IndexToString(
    inputCol="prediction",
    outputCol="predicted_species",
    labels=indexer_model.labels
)
predictions = label_converter.transform(predictions)

# Step 7: Display predictions
predictions.select("features", "label", "prediction", "species", "predicted_species")\
    .show(20, truncate=False)

# Step 8: Evaluate accuracy
evaluator = MulticlassClassificationEvaluator(
    labelCol="label", predictionCol="prediction", metricName="accuracy"
)
accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy: {accuracy:.2f}")
```

```
+-----------------+-----+----------+---------------+-----------------+
|features         |label|prediction|species        |predicted_species|
+-----------------+-----+----------+---------------+-----------------+
|[4.4,3.0,1.3,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.6,3.2,1.4,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.6,3.6,1.0,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.7,3.2,1.3,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.8,3.1,1.6,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.8,3.4,1.6,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.8,3.4,1.9,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.9,3.1,1.5,0.1]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[4.9,3.1,1.5,0.1]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.0,2.3,3.3,1.0]|1.0  |1.0       |Iris-versicolor|Iris-versicolor  |
|[5.0,3.0,1.6,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.0,3.4,1.6,0.4]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.0,3.5,1.3,0.3]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.0,3.5,1.6,0.6]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.1,2.5,3.0,1.1]|1.0  |1.0       |Iris-versicolor|Iris-versicolor  |
|[5.1,3.4,1.5,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.1,3.5,1.4,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
```

```
|[5.1,3.8,1.6,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.2,3.4,1.4,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
|[5.2,3.5,1.5,0.2]|0.0  |0.0       |Iris-setosa    |Iris-setosa      |
+-----------------+-----+----------+---------------+-----------------+
only showing top 20 rows

Test Accuracy: 0.98
```

## Build a Clustering Model with Spark with a dataset of your choice

```
# Install PySpark
!pip install -q pyspark

# Import libraries
from pyspark.sql import SparkSession
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import ClusteringEvaluator
import urllib.request

# Download the Iris dataset locally
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
file_path = "/content/iris.csv"
urllib.request.urlretrieve(url, file_path)

# Start Spark session
spark = SparkSession.builder.appName("IrisKMeans").getOrCreate()

# Load the local CSV into Spark
df = spark.read.csv(file_path, inferSchema=True)
columns = ["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]
df = df.toDF(*columns)

# Drop label column for unsupervised clustering
df_unlabeled = df.drop("class")

# Assemble features
assembler = VectorAssembler(inputCols=df_unlabeled.columns, outputCol="features")
data = assembler.transform(df_unlabeled)

# Apply KMeans clustering
kmeans = KMeans(featuresCol="features", k=3, seed=42)
model = kmeans.fit(data)
predictions = model.transform(data)

# Evaluate clustering
evaluator = ClusteringEvaluator()
silhouette = evaluator.evaluate(predictions)
print(f"Silhouette Score = {silhouette:.4f}")

# Show cluster centers
print("\nCluster Centers:")
for i, center in enumerate(model.clusterCenters()):
    print(f"Cluster {i+1}: {center}")
```

```
Silhouette Score = 0.7342

Cluster Centers:
Cluster 1: [5.006 3.418 1.464 0.244]
Cluster 2: [6.85384615 3.07692308 5.71538462 2.05384615]
Cluster 3: [5.88360656 2.74098361 4.38852459 1.43442623]
```

## Build a Recommendation Engine with Spark with a custom dataset

```
!pip install pyspark ipywidgets --quiet

import pandas as pd
from pyspark.sql import SparkSession
from pyspark.ml.recommendation import ALS
from pyspark.sql.functions import col
import ipywidgets as widgets
from IPython.display import display

# Initialize Spark
spark = SparkSession.builder.master("local[*]").appName("MovieRecommender").getOrCreate()

# Sample MovieLens-like data
data = [
    (0, 100, 4.0),
    (0, 101, 3.5),
```

```
    (1, 100, 5.0),
    (1, 102, 4.0),
    (2, 101, 2.5),
    (2, 103, 4.5),
    (3, 100, 4.0),
    (3, 102, 5.0)
]
ratings_df = spark.createDataFrame(data, ["userId", "movieId", "rating"])

# Train ALS model
als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating", coldStartStrategy="drop", nonnegative=True)
model = als.fit(ratings_df)

# Widget setup
user_input = widgets.BoundedIntText(value=0, min=0, max=10, description='User ID:')
output = widgets.Output()

def on_click_submit(b):
    output.clear_output()
    with output:
        try:
            user_id = user_input.value
            user_df = spark.createDataFrame([(user_id,)], ["userId"])
            recommendations = model.recommendForUserSubset(user_df, 3).collect()
            if recommendations:
                recs = recommendations[0]['recommendations']
                for rec in recs:
                    print(f"Movie ID: {rec['movieId']} | Predicted Rating: {rec['rating']:.2f}")
            else:
                print("No recommendations found.")
        except Exception as e:
            print("Error:", e)

submit_button = widgets.Button(description="Get Recommendations")
submit_button.on_click(on_click_submit)

# Display everything
display(widgets.VBox([user_input, submit_button, output]))

```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 31.6 MB/s eta 0:00:00
```

User ID: 1

Get Recommendati…

```
Movie ID: 100 | Predicted Rating: 4.82
Movie ID: 103 | Predicted Rating: 4.27
Movie ID: 102 | Predicted Rating: 4.06
```