```
# Apache Spark machine learning techniques for developing big data processing applications. using Seeds Dataset give me pyt
```

```
pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.12/dist-packages (4.0.1)
Requirement already satisfied: py4j==0.10.9.9 in /usr/local/lib/python3.12/dist-packages (from pyspark) (0.10.9.9)
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.ml.classification import LogisticRegression, DecisionTreeClassifier
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, ClusteringEvaluator
```

```
# 1. Create Spark Session
spark = SparkSession.builder \
    .appName("Seeds Dataset ML with Spark") \
    .getOrCreate()
```

```
# 2. Load Seeds Dataset
# --------------------------------------------------
# Download seeds_dataset.txt from UCI repository
# https://archive.ics.uci.edu/ml/datasets/seeds

from pyspark.sql.functions import split

df = spark.read.text(data_path)

df = df.select(
    split(df.value, r"\s+").alias("data")
)

df = df.select(
    col("data")[0].cast("double").alias("area"),
    col("data")[1].cast("double").alias("perimeter"),
    col("data")[2].cast("double").alias("compactness"),
    col("data")[3].cast("double").alias("kernel_length"),
    col("data")[4].cast("double").alias("kernel_width"),
    col("data")[5].cast("double").alias("asymmetry"),
    col("data")[6].cast("double").alias("groove_length"),
    col("data")[7].cast("int").alias("label")
)

df.show(5)
df.printSchema()
```

```
+-----+---------+-----------+-------------+------------+---------+-------------+-----+
| area|perimeter|compactness|kernel_length|kernel_width|asymmetry|groove_length|label|
+-----+---------+-----------+-------------+------------+---------+-------------+-----+
|15.26|    14.84|      0.871|        5.763|       3.312|    2.221|         5.22|    1|
|14.88|    14.57|     0.8811|        5.554|       3.333|    1.018|        4.956|    1|
|14.29|    14.09|      0.905|        5.291|       3.337|    2.699|        4.825|    1|
|13.84|    13.94|     0.8955|        5.324|       3.379|    2.259|        4.805|    1|
|16.14|    14.99|     0.9034|        5.658|       3.562|    1.355|        5.175|    1|
+-----+---------+-----------+-------------+------------+---------+-------------+-----+
only showing top 5 rows
root
 |-- area: double (nullable = true)
 |-- perimeter: double (nullable = true)
 |-- compactness: double (nullable = true)
 |-- kernel_length: double (nullable = true)
 |-- kernel_width: double (nullable = true)
 |-- asymmetry: double (nullable = true)
 |-- groove_length: double (nullable = true)
 |-- label: integer (nullable = true)
```

```
# 3. Feature Vectorization
# --------------------------------------------------
```

```
feature_cols = columns[:-1]  # exclude label

assembler = VectorAssembler(
    inputCols=feature_cols,
    outputCol="features"
)

df_vector = assembler.transform(df)
```

```
# 4. Feature Scaling
# ---------------------------------------------------
scaler = StandardScaler(
    inputCol="features",
    outputCol="scaled_features",
    withMean=True,
    withStd=True
)

scaler_model = scaler.fit(df_vector)
df_scaled = scaler_model.transform(df_vector)
```

```
# 5. Train-Test Split
# ---------------------------------------------------
train_df, test_df = df_scaled.randomSplit([0.8, 0.2], seed=42)
```

```
# 6. Logistic Regression Classification
# ---------------------------------------------------
lr = LogisticRegression(
    featuresCol="scaled_features",
    labelCol="label",
    maxIter=100
)

lr_model = lr.fit(train_df)
lr_predictions = lr_model.transform(test_df)

lr_evaluator = MulticlassClassificationEvaluator(
    labelCol="label",
    predictionCol="prediction",
    metricName="accuracy"
)

lr_accuracy = lr_evaluator.evaluate(lr_predictions)
print("Logistic Regression Accuracy:", lr_accuracy)
```

```
Logistic Regression Accuracy: 1.0
```

```
# 7. Decision Tree Classification
# ---------------------------------------------------
dt = DecisionTreeClassifier(
    featuresCol="scaled_features",
    labelCol="label",
    maxDepth=5
)

dt_model = dt.fit(train_df)
dt_predictions = dt_model.transform(test_df)

dt_accuracy = lr_evaluator.evaluate(dt_predictions)
print("Decision Tree Accuracy:", dt_accuracy)
```

```
Decision Tree Accuracy: 0.96875
```

```
# 8. K-Means Clustering (Unsupervised Learning)
# ---------------------------------------------------
kmeans = KMeans(
    featuresCol="scaled_features",
    k=3,
    seed=42
)
```

```
kmeans_model = kmeans.fit(df_scaled)
kmeans_predictions = kmeans_model.transform(df_scaled)

cluster_evaluator = ClusteringEvaluator(
    featuresCol="scaled_features",
    metricName="silhouette"
)

silhouette_score = cluster_evaluator.evaluate(kmeans_predictions)
print("K-Means Silhouette Score:", silhouette_score)
```

```
K-Means Silhouette Score: 0.5928460025426404
```

```
# 9. Stop Spark Session
# ---------------------------------------------------
spark.stop()
```