```
import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, StandardScaler, PCA, StringIndexer
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.stat import Correlation
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import matplotlib.pyplot as plt
import pandas as pd

# ✅ Start Spark Session
spark = SparkSession.builder.appName("Multivariate_Analysis_Iris").getOrCreate()

# ✅ Load Dataset
df = spark.read.csv("/content/Iris.csv", header=True, inferSchema=True)

# ✅ Clean column names and rename label
for old_name in df.columns:
    new_name = old_name.replace(".", "_").replace(" ", "_")
    df = df.withColumnRenamed(old_name, new_name)
df = df.withColumnRenamed(df.columns[-1], "label")  # Assuming last column is target

# ✅ Assemble features into vector
feature_cols = df.columns[:-1]  # All except label
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
df = assembler.transform(df)

# ✅ Show Correlation Matrix (Multivariate insight)
correlation_matrix = Correlation.corr(df, "features", method="pearson").head()[0]
print("Correlation Matrix:\n", correlation_matrix)
```

✅ Standardize Features

```
scaler = StandardScaler(inputCol="features", outputCol="scaled_features", withMean=True, withStd=True)

scaler_model = scaler.fit(df)

df = scaler_model.transform(df)
```

✅ Apply PCA (Multivariate Reduction)

```
pca = PCA(k=2, inputCol="scaled_features", outputCol="pca_features")

pca_model = pca.fit(df)

df = pca_model.transform(df)
```

✅ Index string labels

```
indexer = StringIndexer(inputCol="label", outputCol="indexedLabel")

df = indexer.fit(df).transform(df)
```

✅ Split data

```
train_df, test_df = df.randomSplit([0.8, 0.2], seed=1)
```

✅ Logistic Regression

```
lr = LogisticRegression(featuresCol="pca_features", labelCol="indexedLabel", maxIter=100)

lr_model = lr.fit(train_df)
```

✅ Predictions

```
predictions = lr_model.transform(test_df)
```

✅ Evaluation

```
evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")

accuracy = evaluator.evaluate(predictions)

print(f"Model Accuracy: {accuracy:.2f}")
```

```
# ✅ Optional: Confusion Matrix
predictions.groupBy("indexedLabel", "prediction").count().show()

# ✅ Visualize PCA Result
pandas_df = predictions.select("pca_features", "prediction").toPandas()
pandas_df["PCA1"] = pandas_df["pca_features"].apply(lambda x: x[0])
pandas_df["PCA2"] = pandas_df["pca_features"].apply(lambda x: x[1])

plt.figure(figsize=(8, 6))
scatter = plt.scatter(pandas_df["PCA1"], pandas_df["PCA2"], c=pandas_df["prediction"],
cmap="Set1", alpha=0.7)
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.title("Multivariate PCA Projection of Iris Classification")
plt.grid(True)
plt.colorbar(scatter)
plt.show()

# ✅ Stop Spark
spark.stop()
```