Assignment 2

Aim: Write a program for Cluster Analysis of Big Data using Clustering techniques.

Objective: Implement cluster analysis on the hack_data.csv dataset.

Theory:
Cluster analysis is a machine learning technique used to group similar data points based on specific features. It helps uncover hidden patterns in data without predefined labels. This technique is widely used in market segmentation, anomaly detection, and cybersecurity.

K-Means Clustering:
K-Means is a popular clustering algorithm that partitions data into K clusters based on feature similarity. It works as follows:

1. **Initialize:** Select K random points as initial cluster centers (centroids).
2. **Assign:** Assign each data point to the nearest centroid based on Euclidean distance.
3. **Update:** Compute new centroids by averaging the assigned data points.
4. **Repeat:** Continue steps 2 and 3 until the centroids stop changing or a stopping condition is met.

Dataset Used:
hack_data.csv
The dataset contains various network and user activity metrics, including:

- **Session_Connection_Time**
- **Bytes Transferred**
- **Kali_Trace_Used**
- **Servers_Corrupted**
- **Pages_Corrupted**
- **Location**
- **WPM_Typing_Speed**

Libraries Used:

- **PySpark:** For handling big data processing.
- **MLlib (PySpark ML):** For clustering and feature scaling.

Code Implementation:

1. **Data Preprocessing:** Loaded hack_data.csv, selected key features, and handled missing values. Converted categorical data to numerical format where needed.
2. **K-Means Clustering:**
    - Defined K=3 clusters based on data exploration.
    - Applied PySpark MLlib's KMeans algorithm.

- Predicted cluster labels for each data point.
- Evaluated cluster distribution and characteristics.

Silhouette Scores for K-Means Clustering:
A silhouette score was calculated to evaluate clustering performance for different values of K:

K	Silhouette Score
2	0.82
3	0.76
4	0.65
5	0.63
6	0.56
7	0.47
8	0.43
9	0.39
10	0.37

Result Interpretation:

- The dataset was segmented into three clusters, grouping similar network activity patterns.
- Clusters helped identify suspicious activities based on features like Kali_Trace_Used, Servers_Corrupted, and WPM_Typing_Speed.
- Clustering highlighted variations in user behavior and potential security threats.

clustering-analysis

March 18, 2025

[1]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Cluster").getOrCreate()
```

[2]:
```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("soheiltehranipour/sample-hack-data")

print("Path to dataset files:", path)
```

```
C:\Users\Harshal\OneDrive\Desktop\py_spark project\myenv\Lib\site-
packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

Path to dataset files:
C:\Users\Harshal\.cache\kagglehub\datasets\soheiltehranipour\sample-hack-
data\versions\1
```

[3]:
```
df = spark.read.csv("C:/Users/Harshal/.cache/kagglehub/datasets/
↪soheiltehranipour/sample-hack-data/versions/1/hack_data.csv", header=True,␣
↪inferSchema=True)
```

[4]:
```
df.show(10)
```

```
+----------------------+-----------------+--------------+-----------------+---
-----------+--------------------+----------------+
|Session_Connection_Time|Bytes
Transferred|Kali_Trace_Used|Servers_Corrupted|Pages_Corrupted|
Location|WPM_Typing_Speed|
+----------------------+-----------------+--------------+-----------------+---
-----------+--------------------+----------------+
|                      8|           391.09|             1|             2.96|
7|            Slovenia|           72.37|
|                     20|           720.99|             0|             3.04|
9|British Virgin Is…|          69.08|
```

```
|                    31|           356.32|              1|              3.71|
8|             Tokelau|           70.58|
|                     2|           228.08|              1|              2.48|
8|             Bolivia|            70.8|
|                    20|            408.5|              0|              3.57|
8|                Iraq|           71.28|
|                     1|           390.69|              1|              2.79|
9|    Marshall Islands|           71.57|
|                    18|           342.97|              1|               5.1|
7|             Georgia|           72.32|
|                    22|           101.61|              1|              3.03|
7|         Timor-Leste|           72.03|
|                    15|           275.53|              1|              3.53|
8|Palestinian Terri…|           70.17|
|                    12|           424.83|              1|              2.53|
8|          Bangladesh|           69.99|
+----------------------+-----------------+---------------+-----------------+---
------------+--------------------+---------------+
only showing top 10 rows
```

[5]:
```
from pyspark.ml.feature import VectorAssembler

feat_cols = ['Session_Connection_Time',
 'Bytes Transferred',
 'Kali_Trace_Used',
 'Servers_Corrupted',
 'Pages_Corrupted',
 'WPM_Typing_Speed']

assembler = VectorAssembler(inputCols=feat_cols, outputCol='features')


final_df = assembler.transform(df)
```

[6]:
```
from pyspark.ml.feature import StandardScaler

scaler = StandardScaler(inputCol='features',
                        outputCol='scaled_feat',
                        withStd = True,
                        withMean = False)

scaled_model = scaler.fit(final_df)

cluster_df = scaled_model.transform(final_df)
```

[7]:
```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

eval = ClusteringEvaluator(predictionCol="prediction",
                           featuresCol="scaled_feat",
                           metricName="silhouette",
                           distanceMeasure="squaredEuclidean")
```

[16]:
```
silhouette_score = []
print("""
Silhouette Scores for K-Means Clustering
=======================================
Model\tScore\t
=====\t=====\t
""")

for k in range(2, 11):
    kmeans_algo = KMeans(featuresCol='scaled_feat', k=k)
    kmeans_fit = kmeans_algo.fit(cluster_df)
    output = kmeans_fit.transform(cluster_df)

    # Evaluate silhouette score
    score = eval.evaluate(output)
    silhouette_score.append(score)

    print(f"K{k}\t{round(score, 2)}\t")


output.select("*", "prediction").show(5)
```

```
Silhouette Scores for K-Means Clustering
=======================================
Model   Score
=====   =====

K2      0.82
K3      0.76
K4      0.65
K5      0.63
K6      0.56
K7      0.47
K8      0.43
K9      0.39
K10     0.37
+----------------------+-----------------+--------------+-----------------+---
-----------+--------------------+---------------+--------------------+--------
```

```
-----------+---------+----------+
|Session_Connection_Time|Bytes
Transferred|Kali_Trace_Used|Servers_Corrupted|Pages_Corrupted|
Location|WPM_Typing_Speed|            features|
scaled_feat|prediction|prediction|
+----------------------+-----------------+--------------+-----------------+---
-----------+--------------------+---------------+--------------------+--------
-----------+---------+----------+
|                      8|           391.09|             1|             2.96|
7|            Slovenia|
72.37|[8.0,391.09,1.0,2…|[0.56785108466505…|         0|         0|
|                     20|           720.99|             0|             3.04|
9|British Virgin Is…|
69.08|[20.0,720.99,0.0,…|[1.41962771166263…|         8|         8|
|                     31|           356.32|             1|             3.71|
8|             Tokelau|
70.58|[31.0,356.32,1.0,…|[2.20042295307707…|         7|         7|
|                      2|           228.08|             1|             2.48|
8|             Bolivia|
70.8|[2.0,228.08,1.0,2…|[0.14196277116626…|         0|         0|
|                     20|            408.5|             0|             3.57|
8|                Iraq|
71.28|[20.0,408.5,0.0,3…|[1.41962771166263…|         9|         9|
+----------------------+-----------------+--------------+-----------------+---
-----------+--------------------+---------------+--------------------+--------
-----------+---------+----------+
only showing top 5 rows
```

[15]:
```
output.printSchema()
```

```
root
 |-- Session_Connection_Time: integer (nullable = true)
 |-- Bytes Transferred: double (nullable = true)
 |-- Kali_Trace_Used: integer (nullable = true)
 |-- Servers_Corrupted: double (nullable = true)
 |-- Pages_Corrupted: integer (nullable = true)
 |-- Location: string (nullable = true)
 |-- WPM_Typing_Speed: double (nullable = true)
 |-- features: vector (nullable = true)
 |-- scaled_feat: vector (nullable = true)
 |-- prediction: integer (nullable = false)
```

[ ]:
```

```