Nama : Syava Aprilia Puspitasari

NIM : 2241760129 / 18

Kelas : SIB 3E

Unsupervised Learning dengan Mllib Spark

Jobsheet 14 BIG DATA

Mari Praktik

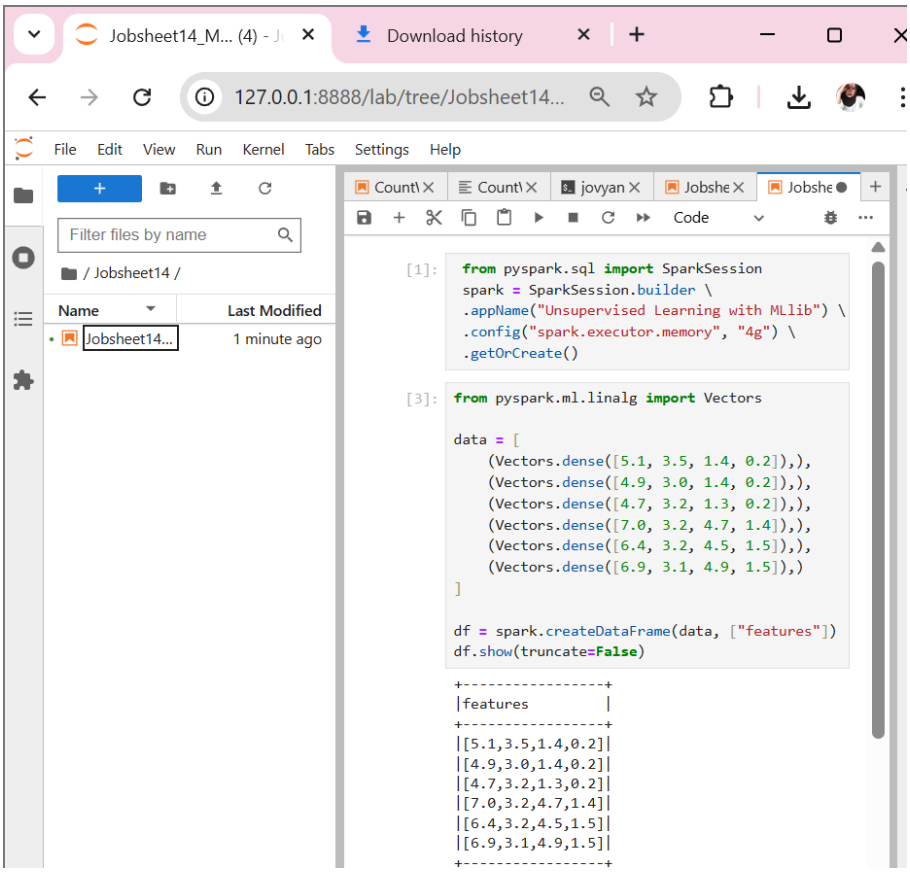
Inisialisasi Spark Session

from pyspark.sql import SparkSession spark = SparkSession.builder \ .appName("Unsupervised Learning with MLlib") \ .config("spark.executor.memory", "4g") \ .getOrCreate(

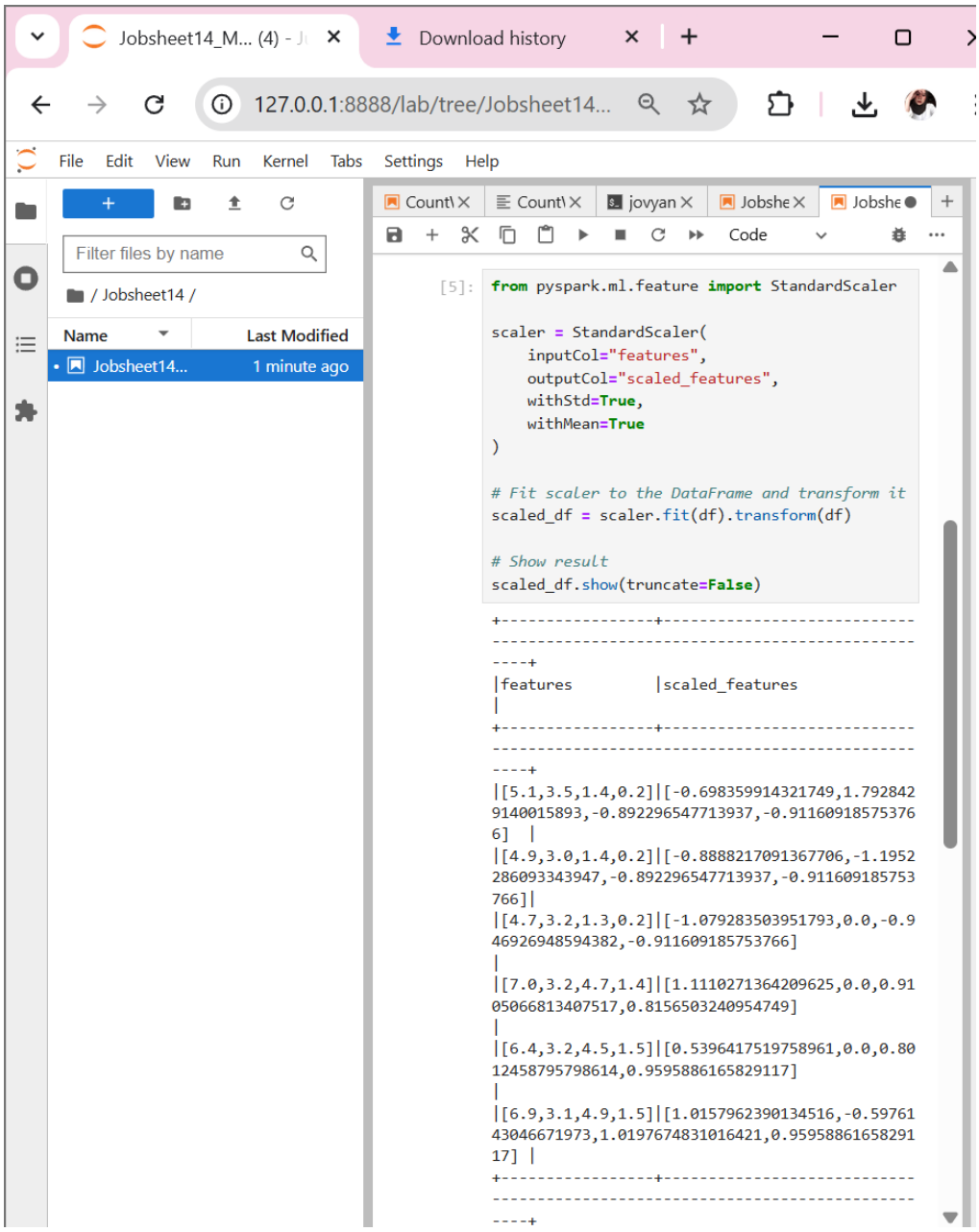
Clustering dengan K-Means

3.1 Persiapan Data

Dataset: Sample dari dataset Iris (sepal length, sepal width, petal length, petal width)



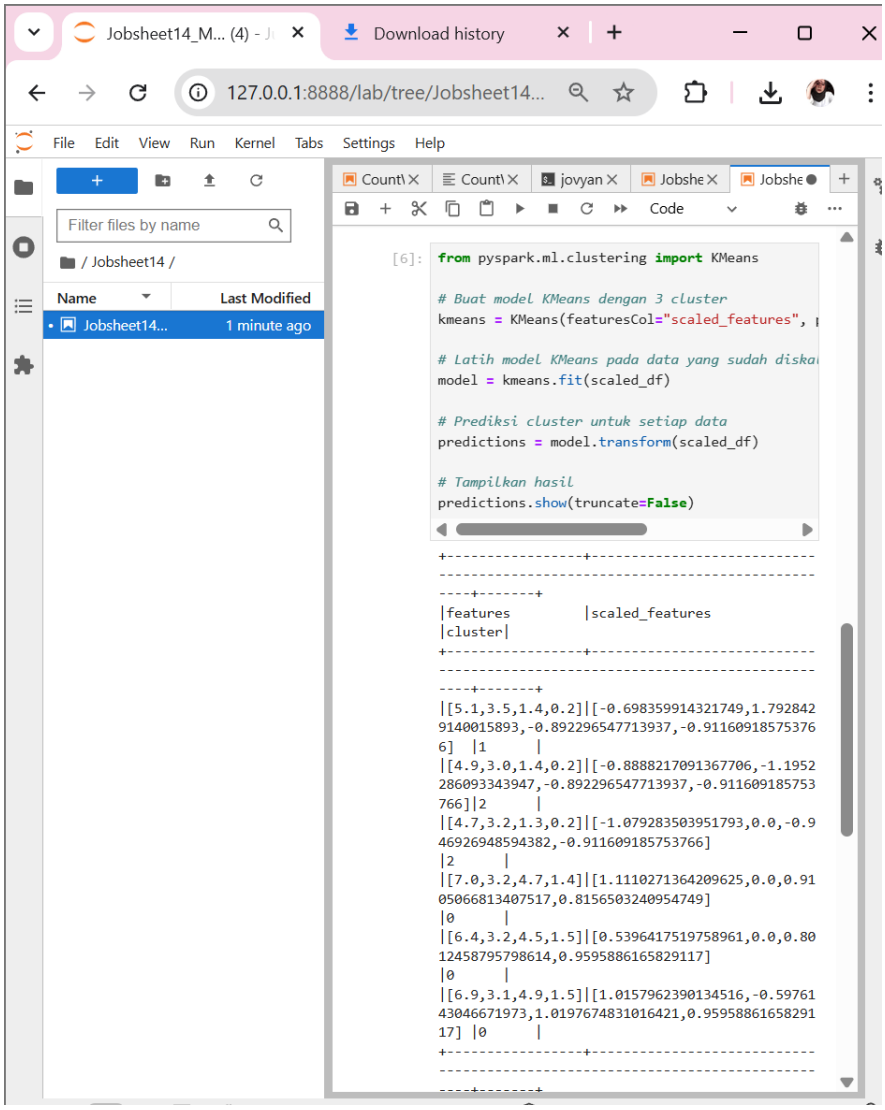
Feature Scaling



Pelatihan Model K-Means

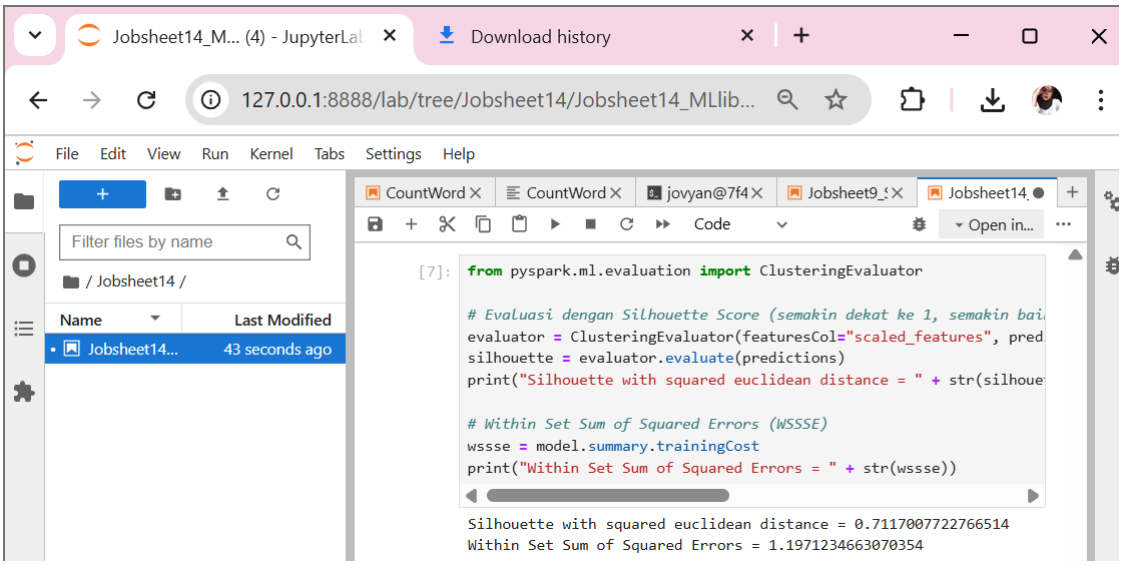
```
from pyspark.ml.clustering import KMeans

kmeans = KMeans().setK(3).setSeed(1)
model = kmeans.fit(scaled_df)
# Prediksi cluster
```

predictions = model.transform(scaled_df)
predictions.show()



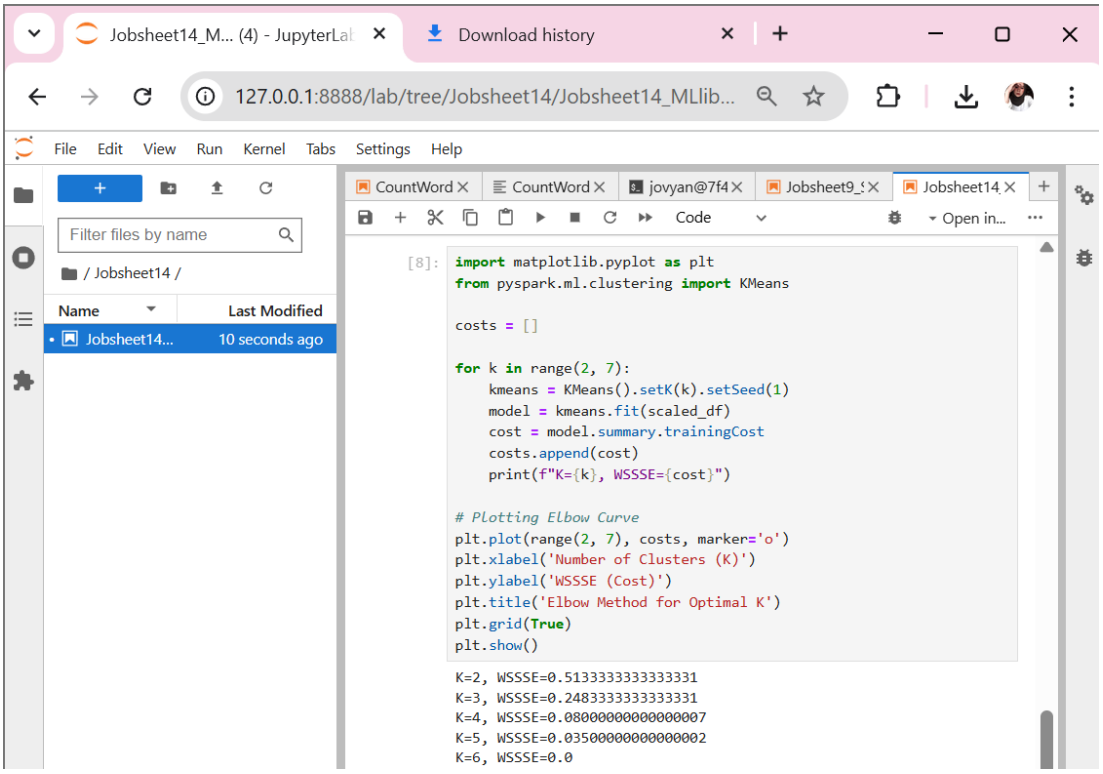
Evaluasi Model from

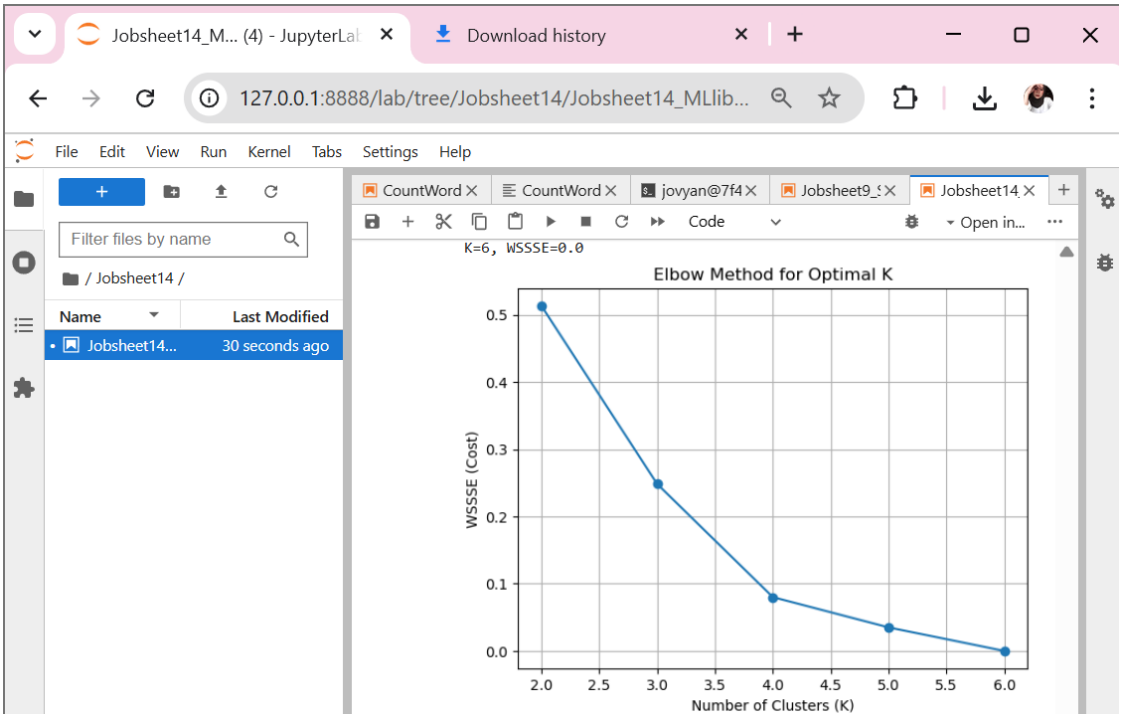
pyspark.ml.evaluation import ClusteringEvaluator
evaluator = ClusteringEvaluator()
silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))
# Within Set Sum of Squared Errors
wssse = model.summary.trainingCost
print("Within Set Sum of Squared Errors = " + str(wssse))



Menentukan Jumlah Cluster Optimal (Elbow Method)

```
import matplotlib.pyplot as plt
costs = []
for k in range(2, 7):
kmeans = KMeans().setK(k).setSeed(1)
model = kmeans.fit(scaled_df)
cost = model.summary.trainingCost
costs.append(cost) print(f"K={k}, WSSSE={costs[-1]}")
plt.plot(range(2, 7), costs)
plt.xlabel('Number of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method')
plt.show()
```

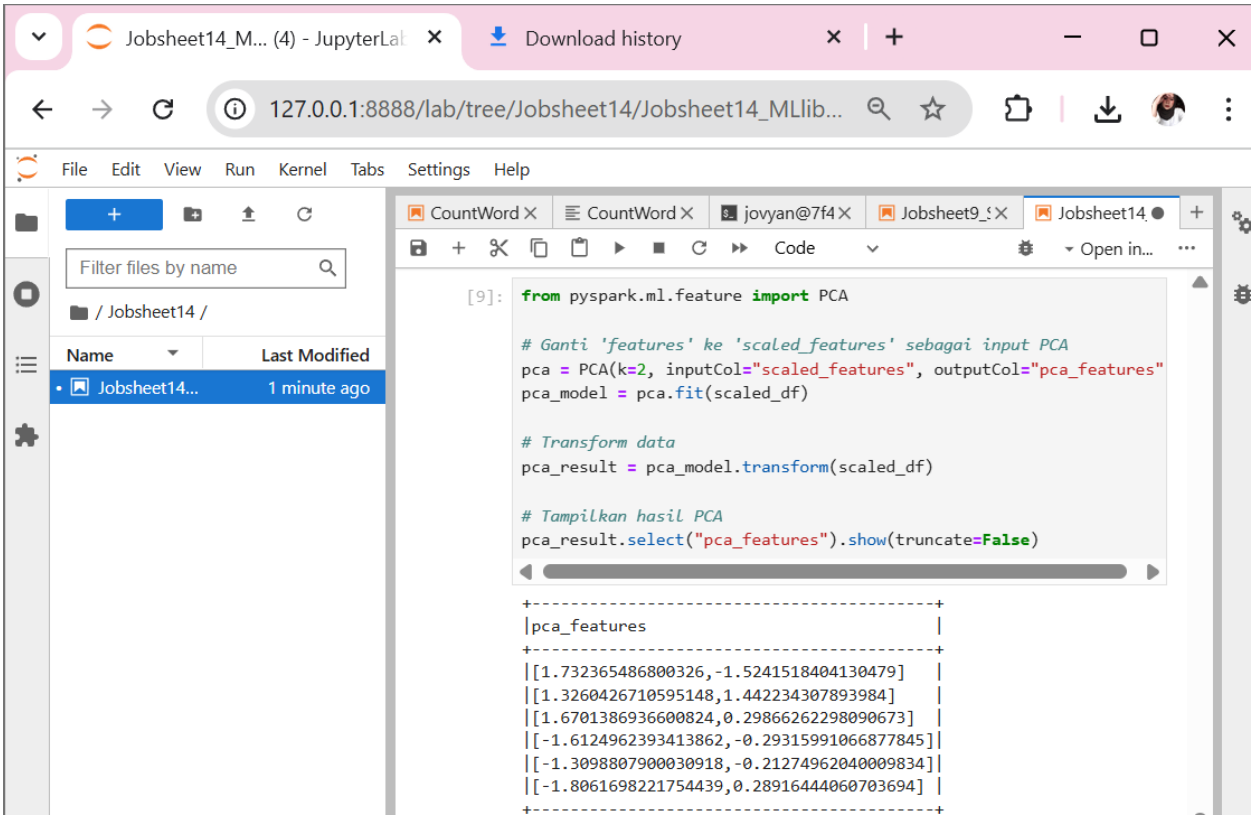




Dimensionality Reduction dengan PCA

from pyspark.ml.feature import PCA
pca = PCA(k=2, inputCol="features", outputCol="pca_features")

pca_model = pca.fit(scaled_df)
# Transform data pca_result = pca_model.transform(scaled_df)
pca_result.select("pca_features").show(truncate=False)



Visualisasi Hasil PCA

import numpy as np
# Ekstrak komponen utama
pca_data = pca_result.select("pca_features").rdd.map(lambda x: x[0]).collect()
x = [float(d[0]) for d in pca_data]
y = [float(d[1]) for d in pca_data]
plt.scatter(x, y)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Result')
plt.show()

Jobsheet14_M... (4) - JupyterLab

Download history

127.0.0.1:8888/lab/tree/Jobsheet14/Jobsheet14_MLlib...

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ Jobsheet14 /

Name	Last Modified
Jobsheet14...	1 minute ago

CountWord | CountWord | jovyan@7f4 | Jobsheet9_S | Jobsheet14

Code | Open in...

[12]:

```
# Gabungkan hasil PCA dan hasil cluster (dalam kolom 'cluster')
pca_result_with_cluster = pca_result.join(predictions.select("featu

# Ambil kolom PCA dan cluster
pca_and_cluster = pca_result_with_cluster.select("pca_features", "c

# Ekstrak nilai-nilai untuk plotting
x = [float(row["pca_features"][0]) for row in pca_and_cluster]
y = [float(row["pca_features"][1]) for row in pca_and_cluster]
clusters = [row["cluster"] for row in pca_and_cluster]

# Visualisasi
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
scatter = plt.scatter(x, y, c=clusters, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Result with Clustering')
plt.colorbar(scatter, label='Cluster')
plt.grid(True)
plt.show()
```

Jobsheet14_M... (4) - JupyterLab

Download history

127.0.0.1:8888/lab/tree/Jobsheet14/Jobsheet14_MLlib...

File Edit View Run Kernel Tabs Settings Help

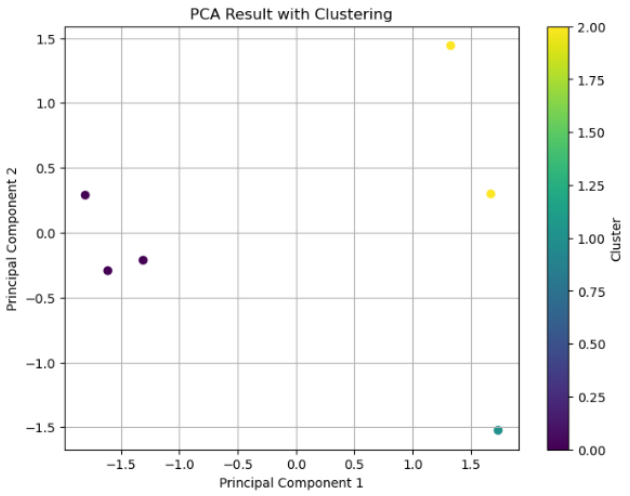
Filter files by name

/ Jobsheet14 /

Name	Last Modified
Jobsheet14...	now

CountWord | CountWord | jovyan@7f4 | Jobsheet9_S | Jobsheet14

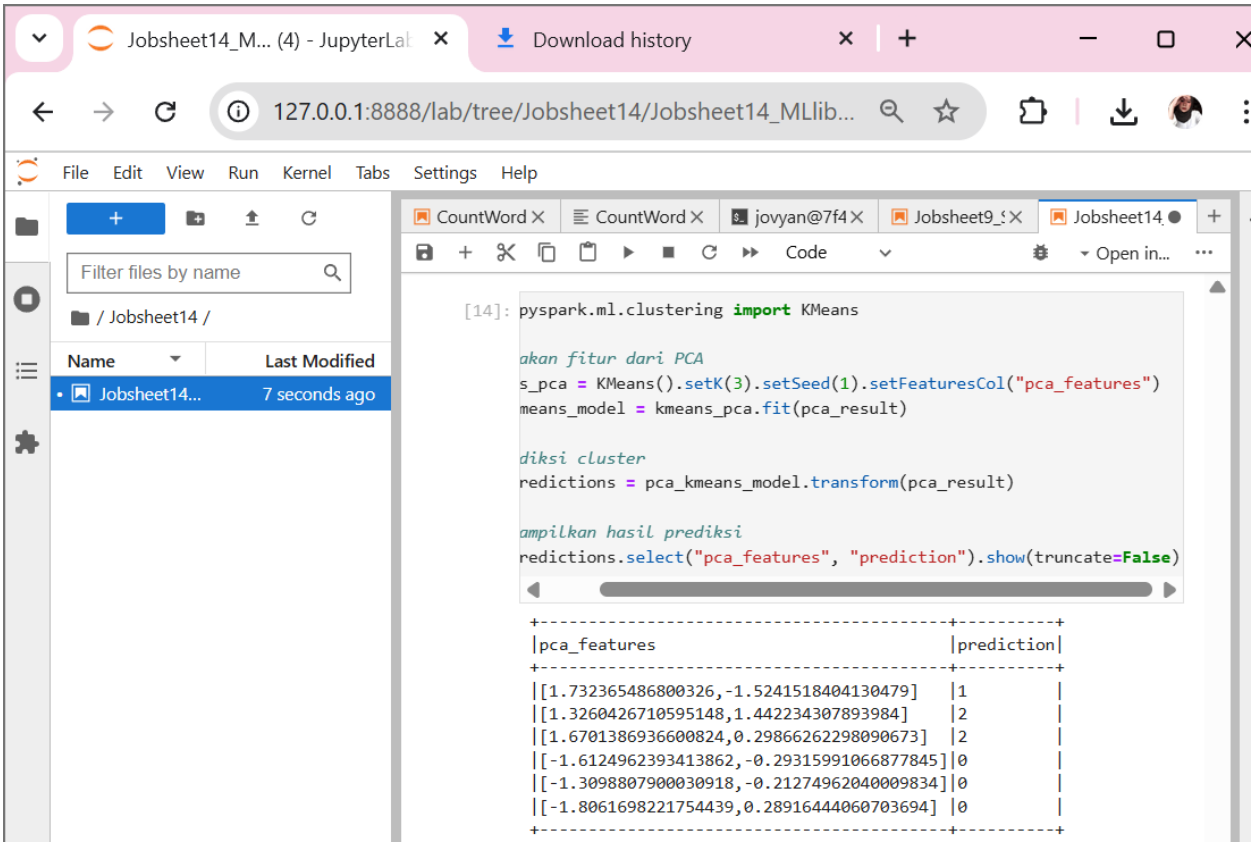
Code | Open in...



Integrasi PCA dan K-Means

Clustering pada Data yang Direduksi

```
kmeans_pca = KMeans().setK(3).setSeed(1).setFeaturesCol("pca_features")
pca_kmeans_model = kmeans_pca.fit(pca_result)
# Prediksi cluster pca_predictions = pca_kmeans_model.transform(pca_result)
pca_predictions.select("pca_features", "prediction").show()
```



Visualisasi Cluster Hasil PCA

```
# Gabungkan hasil clustering dengan data PCA

cluster_data = pca_predictions.select("pca_features", "prediction").collect()
x = [float(d[0][0]) for d in cluster_data]
y = [float(d[0][1]) for d in cluster_data]
clusters = [int(d[1]) for d in cluster_data]
plt.scatter(x, y, c=clusters)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Clustering after PCA')
```

plt.colorbar()
plt.show()