Cài đặt PySpark

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
```

Chuyển đổi file excel sang csv

```
!pip install gdown pyspark
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (5.2.0)
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.3)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.16.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.6)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.6)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.8.30)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7.1)
```

```
import gdown
from pyspark.sql import SparkSession
from pyspark.sql import Row
import numpy as np
import pandas as pd
from pyspark.sql import DataFrame
```

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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Phân tích dữ liệu cơ bản

```
from pyspark.sql import SparkSession
from datetime import datetime
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, TimestampType

spark = SparkSession.builder.appName("Optimized RDD Data Preprocessing").getOrCreate()

# Load the CSV file
df = spark.read.csv('/content/drive/MyDrive/CỦA TÔI/NHOM2_CK/OnlineRetail.csv', header=True, inferSchema=True)

# Convert DataFrame to RDD
rdd = df.rdd

# Filter out rows where InvoiceNo starts with "C"
filtered_rdd = rdd.filter(lambda row: not row.InvoiceNo.startswith("C"))

# Parse the date correctly
def parse_date(row):
    try:
        date = datetime.strptime(row['InvoiceDate'], '%d-%m-%Y %H:%M')
    except:
        date = None
    return (row['InvoiceNo'], row['StockCode'], row['Description'], row['Quantity'],
            row['InvoiceDate'], row['UnitPrice'], row['CustomerID'], row['Country'], date)
```

```
# Map the RDD to include the parsed date
rdd_with_date = filtered_rdd.map(parse_date)

# Define the schema
schema = StructType([
    StructField("InvoiceNo", StringType(), True),
    StructField("StockCode", StringType(), True),
    StructField("Description", StringType(), True),
    StructField("Quantity", IntegerType(), True),
    StructField("InvoiceDate", StringType(), True),
    StructField("UnitPrice", DoubleType(), True),
    StructField("CustomerID", StringType(), True),
    StructField("Country", StringType(), True),
    StructField("date", TimestampType(), True)
])

# Create DataFrame from the RDD with the correct schema
df_processed = spark.createDataFrame(rdd_with_date, schema)
df_processed.show(5, truncate=False)
```

```
+---------+---------+----------------------------------+--------+----------------+---------+----------+--------------+-----------------
|InvoiceNo|StockCode|Description                       |Quantity|InvoiceDate     |UnitPrice|CustomerID|Country       |date
+---------+---------+----------------------------------+--------+----------------+---------+----------+--------------+-----------------
|536365   |85123A   |WHITE HANGING HEART T-LIGHT HOLDER |6       |01-12-2010 08:26|2.55     |17850     |United Kingdom|2010-12-01 08:26:
|536365   |71053    |WHITE METAL LANTERN               |6       |01-12-2010 08:26|3.39     |17850     |United Kingdom|2010-12-01 08:26:
|536365   |84406B   |CREAM CUPID HEARTS COAT HANGER     |8       |01-12-2010 08:26|2.75     |17850     |United Kingdom|2010-12-01 08:26:
|536365   |84029G   |KNITTED UNION FLAG HOT WATER BOTTLE|6       |01-12-2010 08:26|3.39     |17850     |United Kingdom|2010-12-01 08:26:
|536365   |84029E   |RED WOOLLY HOTTIE WHITE HEART.     |6       |01-12-2010 08:26|3.39     |17850     |United Kingdom|2010-12-01 08:26:
+---------+---------+----------------------------------+--------+----------------+---------+----------+--------------+-----------------
only showing top 5 rows
```

```
from pyspark.sql import functions as F

invoice_totals = df_processed.withColumn("TotalCost", F.col("Quantity") * F.col("UnitPrice")) \
                              .groupBy("InvoiceNo") \
                              .agg(F.sum("TotalCost").alias("TotalCost"))
revenue_per_customer = df_processed.withColumn("Revenue", F.col("Quantity") * F.col("UnitPrice")) \
                                    .groupBy("CustomerID") \
                                    .agg(F.sum("Revenue").alias("TotalRevenue"))

revenue_per_country = df_processed.withColumn("Revenue", F.col("Quantity") * F.col("UnitPrice")) \
                                   .groupBy("Country") \
                                   .agg(F.sum("Revenue").alias("TotalRevenue"))

orders_per_day = df_processed.filter(F.col("date").isNotNull()) \
                              .withColumn("DayOfWeek", F.date_format(F.col("date"), "EEEE")) \
                              .groupBy("DayOfWeek") \
                              .count() \
                              .orderBy("count", ascending=False)

average_order_value = invoice_totals.agg(F.avg("TotalCost").alias("AverageOrderValue"))

top_selling_products = df_processed.groupBy("StockCode", "Description") \
                                    .agg(F.sum("Quantity").alias("TotalQuantity")) \
                                    .orderBy("TotalQuantity", ascending=False) \
                                    .limit(10)

print("Tổng chi phí của từng hóa đơn:")
invoice_totals.show()

print("Doanh thu theo khách hàng:")
revenue_per_customer.show()

print("Doanh thu theo quốc gia:")
revenue_per_country.show()

print("Số lượng đơn hàng theo ngày trong tuần:")
orders_per_day.show()

print("Giá trị trung bình của đơn hàng:")
average_order_value.show()

print("Các sản phẩm bán chạy nhất:")
top_selling_products.show()
```

```
+------------------+
| AverageOrderValue|
+------------------+
|482.44019325598475|
+------------------+

Các sản phẩm bán chạy nhất:
+---------+--------------------+-------------+
|StockCode|         Description|TotalQuantity|
+---------+--------------------+-------------+
|    23843|PAPER CRAFT , LIT...|        80995|
|    23166|MEDIUM CERAMIC TO...|        78033|
|    84077|WORLD WAR 2 GLIDE...|        55047|
|   85099B|JUMBO BAG RED RET...|        48478|
|   85123A|WHITE HANGING HEA...|        37603|
|    22197|      POPCORN HOLDER|        36761|
|    84879|ASSORTED COLOUR B...|        36461|
|    21212|PACK OF 72 RETROS...|        36419|
|    23084|  RABBIT NIGHT LIGHT|        30788|
|    22492|MINI PAINT SET VI...|        26633|
+---------+--------------------+-------------+
```

```
df_processed.count()
```

```
532621
```

```
df_processed.select('CustomerID').distinct().count()
```

```
4340
```

```
from pyspark.sql.functions import countDistinct, desc
df_processed.groupBy('Country').agg(countDistinct('CustomerID').alias('country_count')).orderBy(desc('country_count')).show()
```

```
+---------------+-------------+
|        Country|country_count|
+---------------+-------------+
| United Kingdom|         3921|
|        Germany|           94|
|         France|           87|
|          Spain|           30|
|        Belgium|           25|
|    Switzerland|           21|
|       Portugal|           19|
|          Italy|           14|
|        Finland|           12|
|        Austria|           11|
|         Norway|           10|
|        Denmark|            9|
|Channel Islands|            9|
|      Australia|            9|
|    Netherlands|            9|
|         Sweden|            8|
|         Cyprus|            8|
|          Japan|            8|
|         Poland|            6|
|         Greece|            4|
+---------------+-------------+
only showing top 20 rows
```

```
from pyspark.sql.functions import min, max
date_range = df_processed.select(min("date").alias("EarliestOrderDate"), max("date").alias("LatestOrderDate"))
date_range.show()
```

```
+-------------------+-------------------+
|  EarliestOrderDate|    LatestOrderDate|
+-------------------+-------------------+
|2010-12-01 08:26:00|2011-12-09 12:50:00|
+-------------------+-------------------+
```

Data preprocessing

[ ] ↳ *12 ô bị ẩn*

Chuẩn hóa dữ liệu

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler

assemble=VectorAssembler(inputCols=[
    'recency','frequency','monetary_value'
], outputCol='features')

assembled_data=assemble.transform(final_df)

scale=StandardScaler(inputCol='features',outputCol='standardized')
data_scale=scale.fit(assembled_data)
data_scale_output=data_scale.transform(assembled_data)
```

```
data_scale_output.select('standardized').show(2,truncate=False)
```

```
+-----------------------------------------------------------+
|standardized                                               |
+-----------------------------------------------------------+
|[2.003833479365148,0.12386277091331907,0.07758679113935381] |
|[0.528481275942952,0.061931385456659535,0.01143113749621664]|
+-----------------------------------------------------------+
only showing top 2 rows
```

Xây dựng mô hình học máy K-means

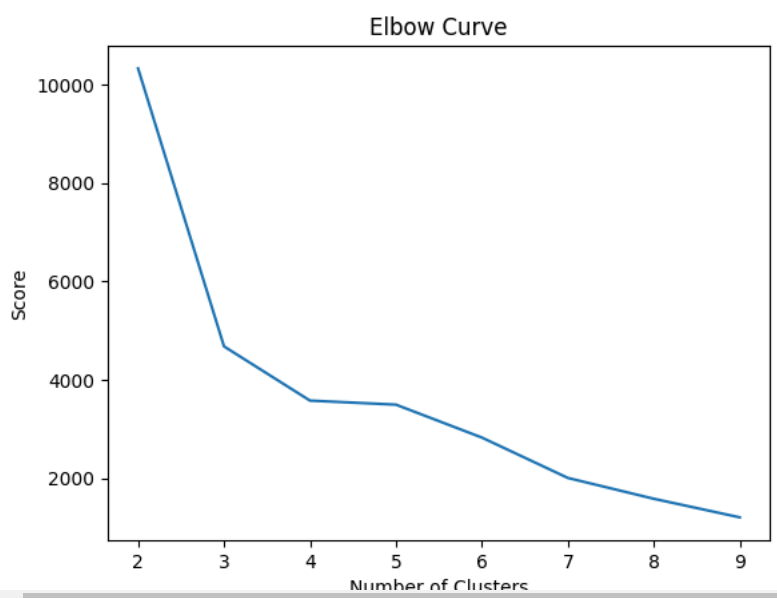
Tìm số cụm K

```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
import numpy as np
from tqdm import tqdm
```

```
cost = np.zeros(10)

evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='standardized',metricName='silhouette', distanceMeasure='squaredEucl

for i in tqdm(range(2,10)):
    KMeans_algo=KMeans(featuresCol='standardized', k=i)
    KMeans_fit=KMeans_algo.fit(data_scale_output)
    output=KMeans_fit.transform(data_scale_output)
    cost[i] = KMeans_fit.summary.trainingCost
```

```
100%|██████████| 8/8 [48:11<00:00, 361.42s/it]
```

```
import pandas as pd
import pylab as pl
df_cost = pd.DataFrame(cost[2:])
df_cost.columns = ["cost"]
new_col = range(2,10)
df_cost.insert(0, 'cluster', new_col)
pl.plot(df_cost.cluster, df_cost.cost)
pl.xlabel('Number of Clusters')
pl.ylabel('Score')
pl.title('Elbow Curve')
pl.show()
```



Training K-means với k=4

```
kmeans_algo=KMeans(featuresCol='standardized', k=4)
kmeans_fit=kmeans_algo.fit(data_scale_output)
```

Tiến hành dự đoán cụm

```
preds=kmeans_fit.transform(data_scale_output)
preds.show(5)
```

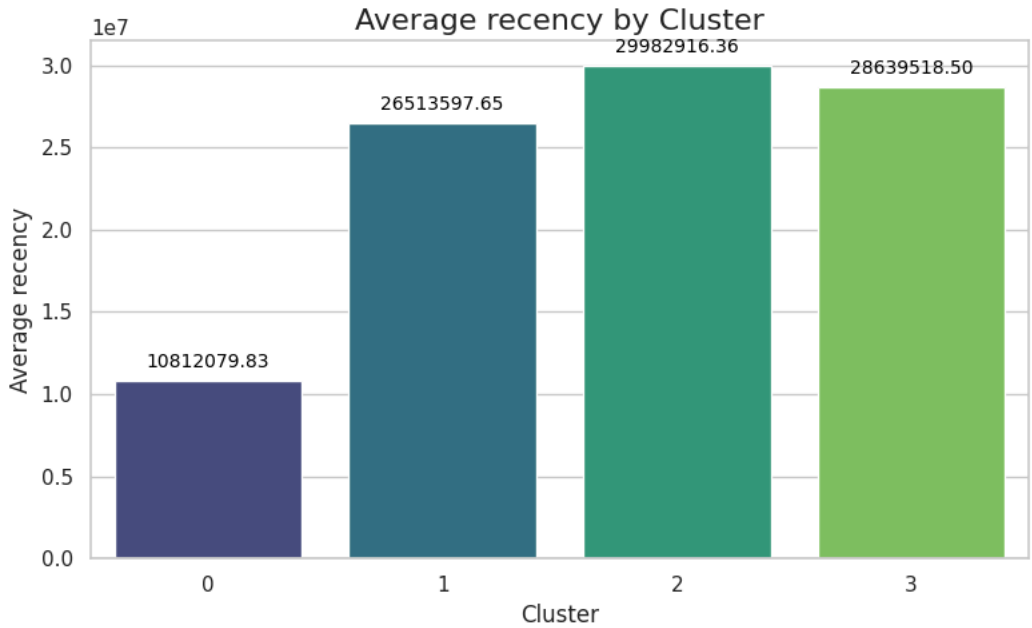
```
+--------+---------+------------------+----------+--------------------+--------------------+----------+
| recency|frequency|    monetary_value|CustomerID|            features|        standardized|prediction|
+--------+---------+------------------+----------+--------------------+--------------------+----------+
| 9700860|        2|389.44000000000005|     16250|[9700860.0,2.0,38...|[1.11772206754745...|         0|
|16947300|        4| 702.2500000000001|     15574|[1.69473E7,4.0,70...|[1.95264865128936...|         0|
|31218780|       16| 4805.169999999994|     15555|[3.121878E7,16.0,...|[3.59699236231725...|         3|
|31643100|       15|2507.0699999999997|     15271|[3.16431E7,15.0,2...|[3.64588203062519...|         3|
| 4586760|        1|             153.0|     17714|[4586760.0,1.0,15...|[0.52848127594295...|         0|
+--------+---------+------------------+----------+--------------------+--------------------+----------+
only showing top 5 rows
```

Trực quan hóa kết quả phân cụm

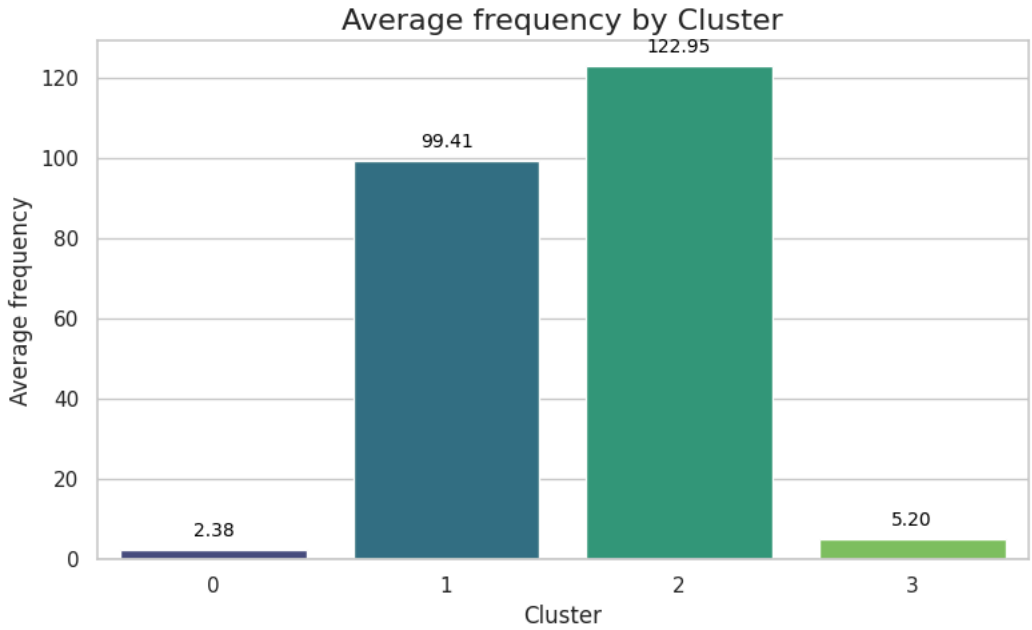
```
import matplotlib.pyplot as plt
import seaborn as sns

df_viz = preds.select('recency', 'frequency', 'monetary_value', 'prediction')
df_viz = df_viz.toPandas()

avg_df = df_viz.groupby(['prediction'], as_index=False).mean()

list_value = ['recency', 'frequency', 'monetary_value']

sns.set(style="whitegrid")

for i in list_value:
    plt.figure(figsize=(8, 5))
    bar_plot = sns.barplot(x='prediction', y=str(i), data=avg_df, palette='viridis')

    plt.title(f'Average {i} by Cluster', fontsize=16)
    plt.xlabel('Cluster', fontsize=12)
    plt.ylabel(f'Average {i}', fontsize=12)
```

```
for p in bar_plot.patches:
    bar_plot.annotate(format(p.get_height(), '.2f'),
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha='center', va='bottom',
                      fontsize=10, color='black', rotation=0,
                      xytext=(0, 5),
                      textcoords='offset points')

plt.tight_layout()
plt.show()
```

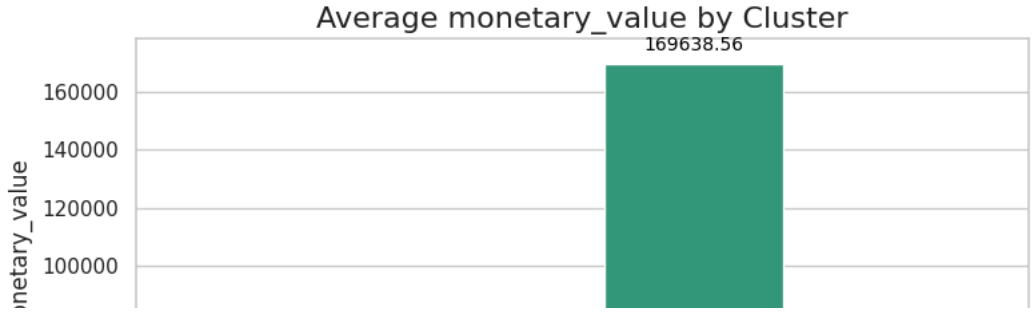
```
<ipython-input-25-59ccef2e5080>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  bar_plot = sns.barplot(x='prediction', y=str(i), data=avg_df, palette='viridis')
```

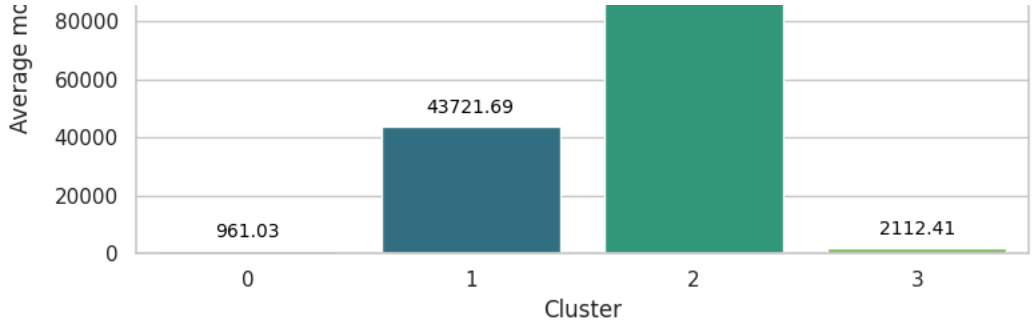



```
<ipython-input-25-59ccef2e5080>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  bar_plot = sns.barplot(x='prediction', y=str(i), data=avg_df, palette='viridis')
```




```
<ipython-input-25-59ccef2e5080>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  bar_plot = sns.barplot(x='prediction', y=str(i), data=avg_df, palette='viridis')
```





RCM

```
df = spark.read.csv('/content/drive/MyDrive/CỦA TÔI/NHOM2_CK/OnlineRetail.csv', header=True, inferSchema=True)
```

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

```
+---------+---------+--------------------+--------+----------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|     InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+----------------+---------+----------+--------------+
|   536365|   85123A|WHITE HANGING HEA...|       6|01-12-2010 08:26|     2.55|     17850|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6|01-12-2010 08:26|     3.39|     17850|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8|01-12-2010 08:26|     2.75|     17850|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6|01-12-2010 08:26|     3.39|     17850|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|01-12-2010 08:26|     3.39|     17850|United Kingdom|
+---------+---------+--------------------+--------+----------------+---------+----------+--------------+
only showing top 5 rows
```

```
from pyspark.ml.recommendation import ALS
```

```
best_model = cv_model.bestModel

predictions = best_model.transform(test_data)
rmse = evaluator.evaluate(predictions)
print(f"Best Model's RMSE on test data: {rmse}")

print(f"Best rank: {best_model.rank}")
print(f"Best maxIter: {best_model._java_obj.parent().getMaxIter()}")
print(f"Best regParam: {best_model._java_obj.parent().getRegParam()}")

best_model.userFactors.show()
best_model.itemFactors.show()

best_model.recommendForAllUsers(3).show()
```

```
Best Model's RMSE on test data: 33.66984784435793
Best rank: 30
Best maxIter: 10
Best regParam: 0.01
+-----+--------------------+
|   id|            features|
+-----+--------------------+
|12350|[0.0, 0.084478065...|
|12360|[0.19722603, 0.09...|
|12370|[0.0, 0.0, 0.0914...|
|12380|[0.0, 0.0, 0.4529...|
|12390|[0.08485217, 0.24...|
|12410|[0.0, 0.0, 0.0, 0...|
|12420|[0.0, 0.0, 0.0, 0...|
|12430|[0.0138090905, 0....|
```