



MACHINE LEARNING USING SPARK

SUBMITTED BY:
PRAGATI DEBATA-A21022



● Research Objective:

- To discover the clusters of Uber data based on the longitude and latitude, then we will analyze the cluster centers by date/time.
- Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group are more like each other than to those in other groups.



Importance of our Objective:

- In today's world the connectivity is a cakewalk. The cab system has reached heights and generated a huge data left to analyze.
- Deep diving into this data we will explore:
 1. Which hours of the day and which cluster had the highest number of pickups?
 2. How many pickups occurred in each cluster?
 3. How many pickups occurred in each hour?





● Approach:



Creating the spark environment



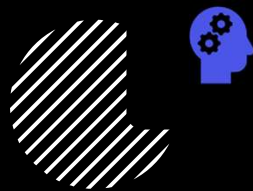
Loading the required libraries



Understanding the dataset



Data Exploration



Applying various models and predictions



Understanding the Data Set

- Our data is Uber trip data from <http://data.beta.nyc/dataset/uber-trip-data-foiled-apr-sep-2014>.

`Date/Time`:The date and time of the Uber pickup

`Lat`:The latitude of the Uber pickup

`Lon`:The longitude of the Uber pickup

`Base`:The TLC base company affiliated with the Uber pickup



● The Coding Part:

- To initiate the spark environment in colab we will first need to install JAVA.

```
!apt update > /dev/null  
!apt install openjdk-8-jdk-headless -qq > /dev/null
```

- Creating the spark session

sc

SparkContext

[Spark UI](#)

Version

v3.1.2

Master

local[*]

AppName

pyspark-shell






Loading the data:

- First, we need to import all the libraries.
- Our data is Uber trip data from <http://data.beta.nyc/dataset/uber-trip-data-foiled-apr-sep-2014>.

```
uber_df.show(5)
```

	dt	lat	lon	base
	2014-08-01 00:00:00	40.729	-73.9422	B02598
	2014-08-01 00:00:00	40.7476	-73.9871	B02598
	2014-08-01 00:00:00	40.7424	-74.0044	B02598
	2014-08-01 00:00:00	40.751	-73.9869	B02598
	2014-08-01 00:00:00	40.7406	-73.9902	B02598

only showing top 5 rows





Statistics Summary:

- Spark DataFrames include some built-in functions for statistical processing.
- The `describe()` function performs summary statistics calculations on all numeric columns and returns them as a DataFrame.

```
uber_df.describe(["dt", "lat", "lon"]).show()
```

summary	lat	lon
count	829275	829275
mean	40.73778073582407	-73.97016031317641
stddev	0.043628060846854146	0.06148272834515576
min	39.6569	-74.7737
max	41.3182	-72.3359



Split into Training and Testing Set:

- The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.
- Before splitting we will vectorize the columns.

```
feature_assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
```

```
uber_assembled_df = feature_assembler.transform(uber_df)  
uber_assembled_df.cache()
```

```
uber_assembled_df.show(5)
```

```
+-----+-----+-----+-----+-----+  
|          dt|    lat|    lon|   base|    features|  
+-----+-----+-----+-----+-----+  
|2014-08-01 00:00:00| 40.729|-73.9422|B02598|[40.729,-73.9422]|  
|2014-08-01 00:00:00|40.7476|-73.9871|B02598|[40.7476,-73.9871]|  
|2014-08-01 00:00:00|40.7424|-74.0044|B02598|[40.7424,-74.0044]|  
|2014-08-01 00:00:00| 40.751|-73.9869|B02598|[40.751,-73.9869]|  
|2014-08-01 00:00:00|40.7406|-73.9902|B02598|[40.7406,-73.9902]|  
+-----+-----+-----+-----+-----+
```

only showing top 5 rows



● Continued:

```
train_df, test_df = uber_assembled_df.randomSplit([0.7, 0.3], seed=rnd_seed)
```

```
train_df.cache()  
test_df.cache()
```

- Removing the unnecessary data frames

```
uber_df.unpersist()  
uber_assembled_df.unpersist()
```

```
uber_df.show(5)
```

```
+-----+-----+-----+-----+  
|          dt|    lat|    lon|   base|  
+-----+-----+-----+-----+  
|2014-08-01 00:00:00| 40.729|-73.9422|B02598|  
|2014-08-01 00:00:00|40.7476|-73.9871|B02598|  
|2014-08-01 00:00:00|40.7424|-74.0044|B02598|  
|2014-08-01 00:00:00| 40.751|-73.9869|B02598|  
|2014-08-01 00:00:00|40.7406|-73.9902|B02598|  
+-----+-----+-----+-----+  
only showing top 5 rows
```





MODELS

● K-Means Clustering

- k-means is one of the most popular clustering algorithms. In this algorithm, a user-specified number of clusters (k) are randomly assigned to different points in the dataset.
- Choosing the right value for k is an extremely important aspect of using this algorithm successfully. There is no real prescription for the number of clusters you need, so we will likely have to experiment with different values and consider what we would like the end result to be.



● Importing libraries:

- K-means is one of the most used clustering algorithms that clusters the data points into a predefined number of clusters.
- K-means is implemented as an estimator and generates a KMeansModel as the base model.

```
from pyspark.ml.clustering import KMeans  
from pyspark.ml.evaluation import ClusteringEvaluator
```



● Training and fitting the Model

```
kmeans = KMeans(k=8, initMode='k-means||', featuresCol='features', predictionCol='cluster', maxIter=10)
```

```
kmModel = kmeans.fit(train_df)
```

```
print("KMeans Cluster Centers: ")  
for center in kmModel.clusterCenters():  
    print(center)
```

```
KMeans Cluster Centers:
```

```
[ 40.7611 -73.8722]  
[ 40.7693 -73.97  ]  
[ 40.6865 -73.9639]  
[ 40.766  -73.4801]  
[ 40.905  -73.8711]  
[ 40.6562 -73.7797]  
[ 40.7324 -73.9971]  
[ 40.6997 -74.2005]
```



Predictions

- As said earlier, we will be predicting clusters based on longitude and latitude columns of our dataset.
- The picture shows the different clusters that have been assigned.

```
test_preds = kmModel.transform(test_df)
test_preds.cache()
```

```
DataFrame[dt: timestamp, lat: double, lon: double, base: string, features: vector, cluster: int]
```

```
test_preds.show(5)
```

dt	lat	lon	base	features	cluster
2014-08-01 00:00:00	40.6754	-74.017	B02682	[40.6754, -74.017]	2
2014-08-01 00:00:00	40.6982	-73.9669	B02617	[40.6982, -73.9669]	2
2014-08-01 00:00:00	40.7063	-73.9223	B02617	[40.7063, -73.9223]	2
2014-08-01 00:00:00	40.7134	-74.0091	B02682	[40.7134, -74.0091]	6
2014-08-01 00:00:00	40.7303	-74.0029	B02682	[40.7303, -74.0029]	6

only showing top 5 rows



● Summary:

- k-means includes a summary class that we can use to evaluate our model. This class provides some common measures for k-means success. The k-means summary includes information about the clusters created.

```
print(kmModel.summary.clusterSizes)
```

```
[26431, 185757, 87002, 1657, 7060, 18311, 248176, 6055]
```



● Gaussian Mixture Models

- Gaussian mixture models (GMM) are another clustering algorithm that try to group data by reducing the sum of squared distances from the center of the cluster.
- Gaussian mixture models assume that each cluster produces data based upon random draws from a Gaussian distribution.
- A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians





Fitting and testing the model:

- The picture shows the different clusters that have been assigned.

```
gmmModel = gmm.fit(train_df)
```

```
gmmModel.transform(test_df).show(5)
```

dt	lat	lon	base	features	probability	cluster
2014-08-01 00:00:00	40.6754	-74.017	802682	[40.6754, -74.017]	[0.01651681562722...	1
2014-08-01 00:00:00	40.6982	-73.9669	802617	[40.6982, -73.9669]	[0.04109997177197...	5
2014-08-01 00:00:00	40.7063	-73.9223	802617	[40.7063, -73.9223]	[0.28313398784207...	5
2014-08-01 00:00:00	40.7134	-74.0091	802682	[40.7134, -74.0091]	[0.01263642997046...	4
2014-08-01 00:00:00	40.7303	-74.0029	802682	[40.7303, -74.0029]	[0.00900845633139...	4

only showing top 5 rows






Summary:

- Like our k-means clustering algorithm, Gaussian mixture models include a summary class to help with model evaluation. This includes information about the clusters created, like the weights, the means, and the covariance of the Gaussian mixture, which can help us learn more about the underlying structure inside of our data

```
gmmModel.weights
```

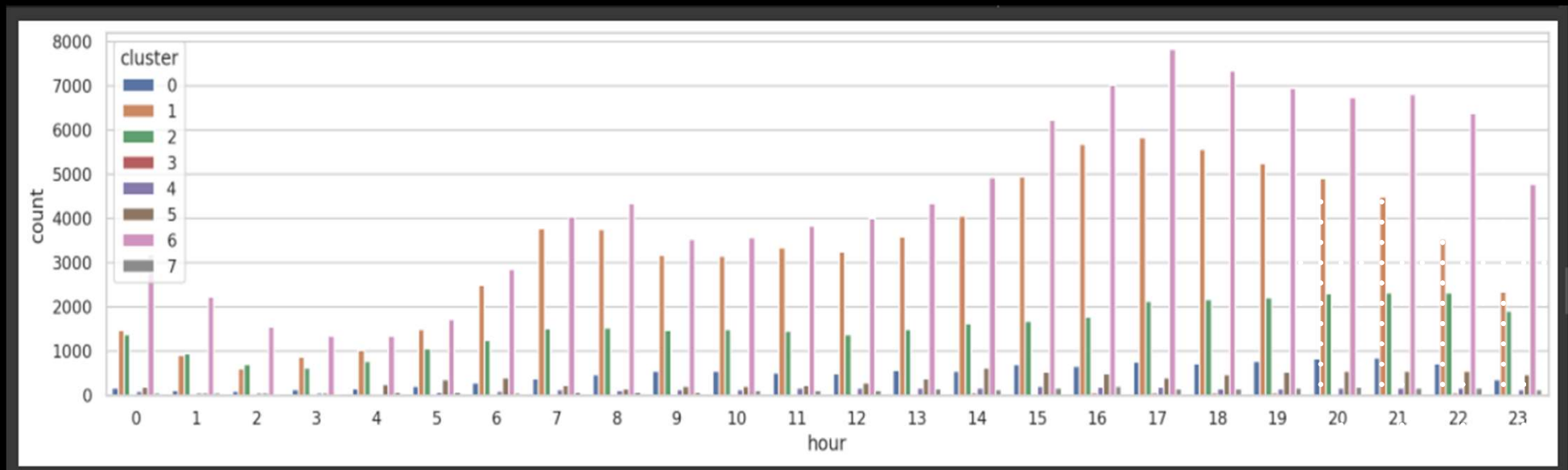
```
[0.06344261362392324,  
0.0359304395523446,  
0.07520962320323114,  
0.10691410992080919,  
0.5255596609254812,  
0.14360127260109098,  
0.019438069333575034,  
0.02990421083954458]
```



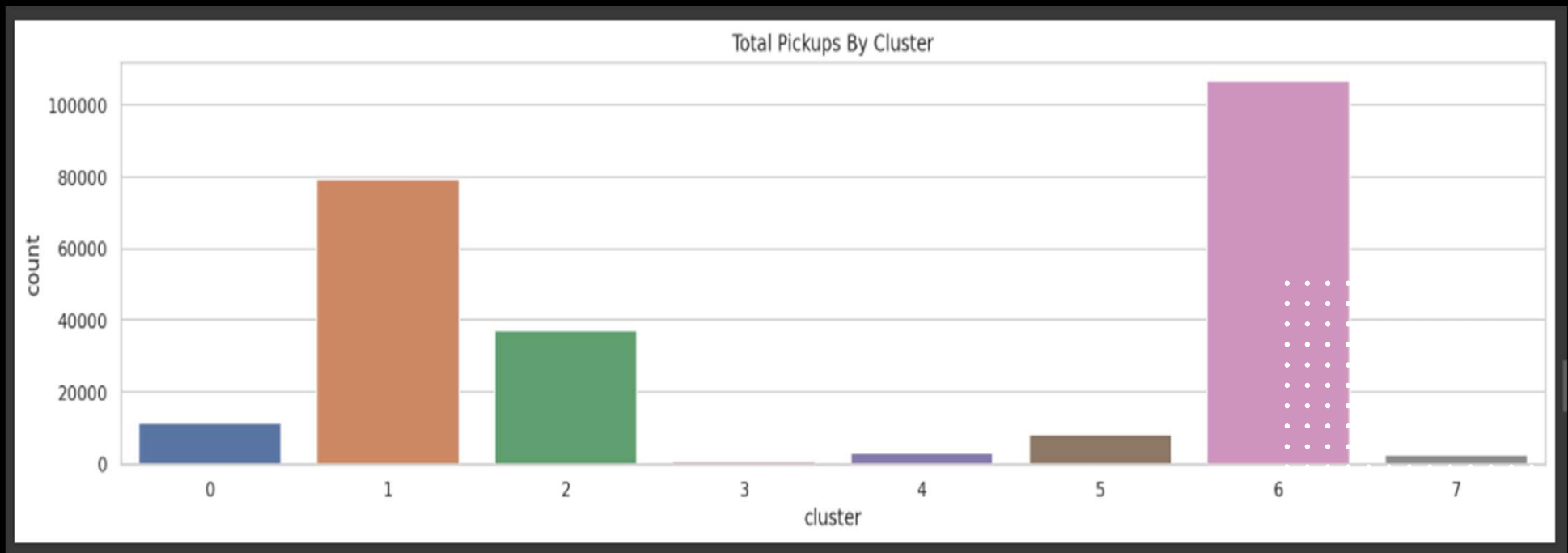


**LET'S
ANSWER FEW
QUESTIONS**

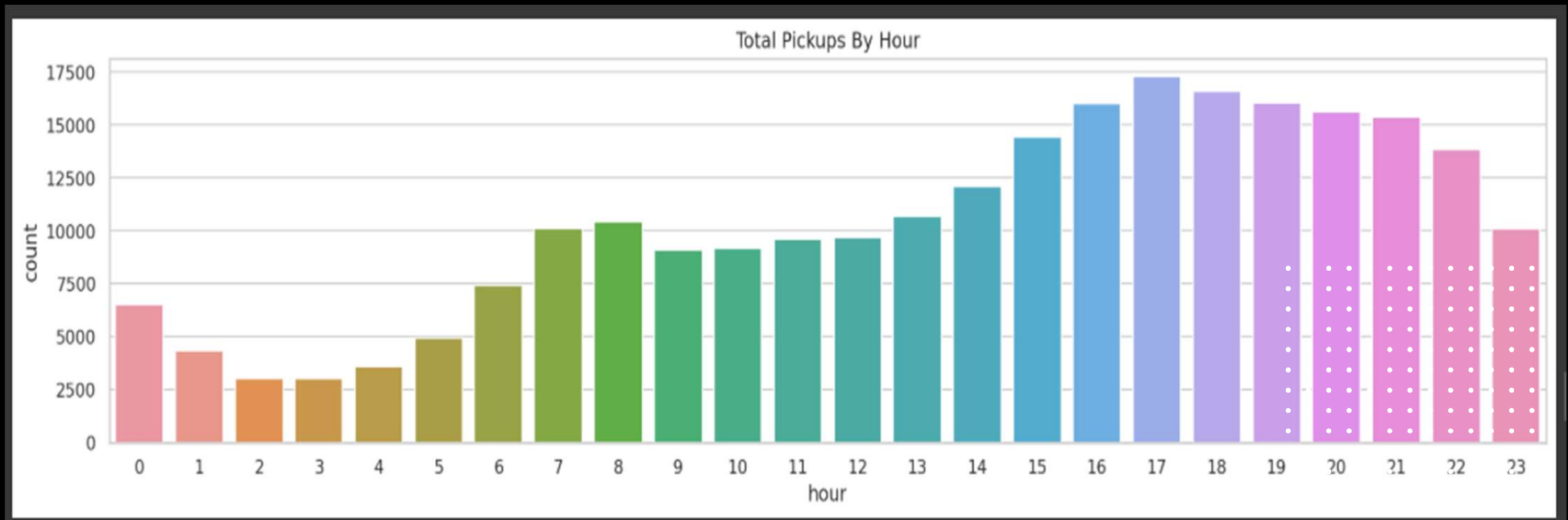
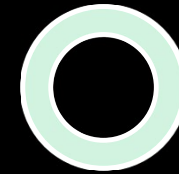
WHICH HOURS OF THE DAY
AND WHICH CLUSTER HAD
THE HIGHEST NUMBER OF
PICKUPS?



HOW MANY PICKUPS
OCCURRED IN EACH
CLUSTER?



HOW MANY PICKUPS
OCCURRED IN EACH HOUR?



THANK
YOU

