

# New York City

## Taxi Trip Duration

Azure (Storage + Databricks)

**Aditya Pradip Kulkarni**

DTSC 701 - Introduction to Big Data

ID: 1330344

### Table of contents:

1.	Data Lake
2.	GOAL
3.	Architecture
4.	Dataset
5.	Azure services
6.	Procedure
7.	Conclusion
8.	References

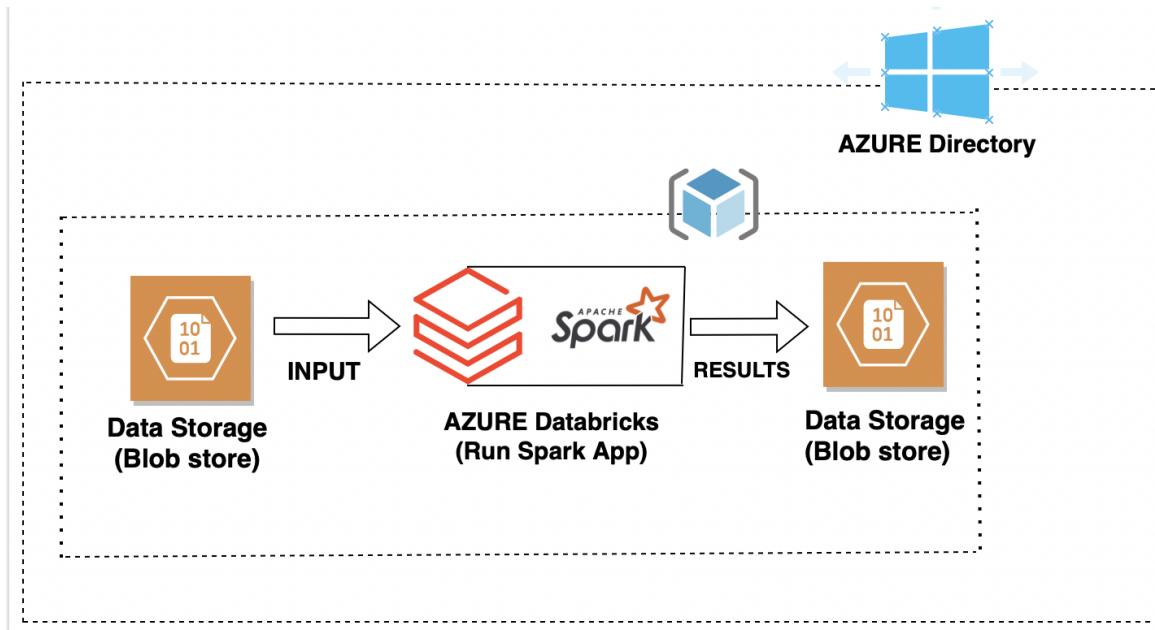
# Data Lake:

- I am implementing a data lake solution for performing the data analysis, transformation and later training machine learning model.
- The flow will be azure blob store > store dataset > mount azure blob on azure databricks > perform preprocessing and transformation > train models > store the visualizations in azure blob.
- As my data is not completely transformed and ready to use for training ML models, I need to store the raw data in blob storage and later perform operations on it in azure databricks which will then give me a processed spark dataframe.
- This new spark dataframe is then used for further analysis and training models.
- Unlike data warehouse, which requires storing structured data and performing SQL queries on the data for further analysis.
- Also this solution will provide good scalability in future, when it comes to storing and analyzing large amounts of data.
- Hence, I choose to go with this solution.

# G.O.A.L

- Analyze the NYC taxi trip data.
- Store data in Azure blob store.
- Perform the data transformation, preprocessing in Azure Databricks notebook using Apache Spark as processing engine.
- Train machine learning models, and evaluate the models based on certain metrics.
- Store the EDA plots in the blob container for further analysis and use.

# Project Architecture



## Components:

1. **Azure blob stores (hot store)**: For storing the dataset, Visualization plots, results.
2. **Azure databricks**: Performing Preprocessing, Exploratory data analysis, data transformation, and training machine learning models.
3. **Compute engine**: A compute engine named(ETL Compute), this is the cluster/machine on which the databricks spark notebook will be running and handling the spark jobs. Specifications : 14GB RAM, 4 Core CPU.
4. **Resource group**: A resource group is a logical container for resources deployed in an Azure subscription. It helps you manage and organize resources efficiently.

# Dataset

- **Data Source:** The provided information pertains to New York City Taxi Cab trips, specifically from the year 2014, sourced from both driver input and GPS coordinates via New York City's Open Data website, the dataset is taken from

Kaggle.

- **Data size:** The data size is 2.36 GB, with shape (14999999, 18)
- **Data Subset:** The shared data represents a subset of the complete dataset, containing the first 15 million records from 2014, compared to the original dataset size of over 165 million trips.
- **Data Format and Fields:** The data is provided in CSV format and includes various fields such as vendor ID, pickup/dropoff datetime, passenger count, trip distance, location coordinates, payment details, fare amount, and additional information collected during the taxi trips.
- **Kaggle:** <https://www.kaggle.com/datasets/kentonlp/2014-new-york-city-taxi-trips/data>

# Data

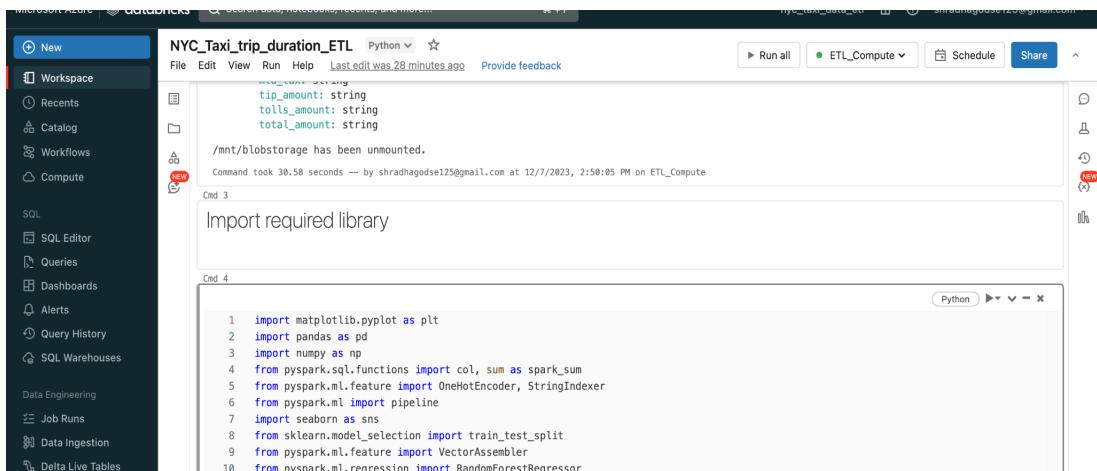
	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.982155	40.767937
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.980415	40.738564
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-73.979027	40.763939
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040	40.719971	-74.010040	40.719971
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053	40.793209	-73.973053	40.793209

# Azure services

## Azure blob store

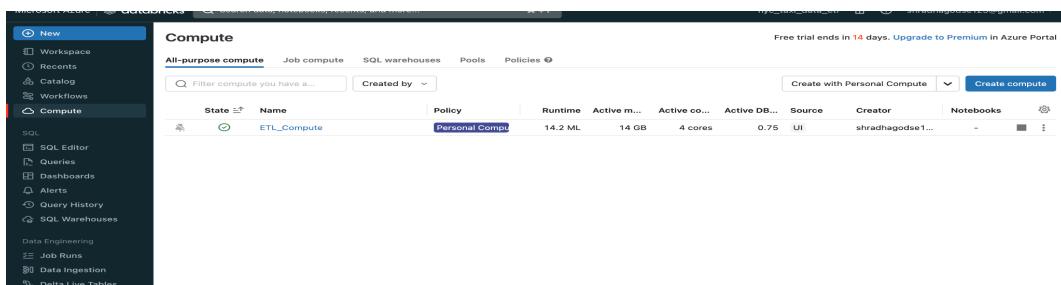
The screenshot shows the Azure Storage Accounts interface for the 'volstore' container. The 'datastore' container is selected. The top navigation bar includes Home, Storage accounts, volstore, Containers, and LTI. The main area has sections for Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access tokens, Access policy, Properties, and Metadata. The 'Overview' section shows blob statistics: 1 blob, 1 snapshot, and 1 lease. It includes a search bar for blobs by prefix and a 'Show deleted blobs' toggle. The 'Authentication method' is set to 'Access key'. The 'Location' is listed as 'datastore'. Below these are buttons for Upload, Change access level, Refresh, Delete, Change tier, Acquire lease, Break lease, View snapshots, Create snapshot, and Give feedback. A table at the bottom lists blobs with columns: Name, Modified, Access tier, Archive status, Blob type, Size, and Lease state. Two blobs are listed: 'Dataset' and 'Visualization'.

## Azure databricks notebook



The screenshot shows the Azure Databricks notebook interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, and Delta Live Tables. The main area displays a notebook titled "NYC\_Taxi\_trip\_duration\_ETL" with a Python script. The script imports various libraries such as matplotlib, pandas, numpy, and sklearn, and performs ETL operations on a dataset named "taxi.parquet". The notebook also shows a command history with Cmd 3 and Cmd 4.

## Azure compute engine



The screenshot shows the Azure Compute Engine interface. The sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, and Delta Live Tables. The main area displays a table of compute resources. One row is selected, showing details for "ETL\_Compute" which has a runtime of 14.2 ML, active memory of 14 GB, 4 cores, and a source of Personal Compute. The creator is listed as "shradhagodse1...".

# PROCEDURE:

1. Create an Azure account.  
(Link:<https://azure.microsoft.com/en-us/free/>).
2. After the account is created, you will get free access to some of the interesting azure tools(Link:<https://azure.microsoft.com/en-us/free/#all-free-services>).

### 3. Create a Azure blob storage account:

- i. Type storage account on the search bar situated on the top center part of the azure portal.(*See this written Search resources, service, and docs*)

The screenshot shows the Microsoft Azure portal's search interface. The search bar at the top contains the text "storage account". Below the search bar, a navigation bar includes "Microsoft Azure", "Upgrade", and a search icon. The main content area has tabs for "All", "Services (20)", "Marketplace (5)", "Documentation (99+)", "Resources (0)", and "Resource Groups (0)". A sub-header "Services" is displayed above a list of service icons. The "Storage accounts" icon is highlighted with a gray background. Other listed services include "Storage accounts (classic)", "Storage browser", "Storage movers", "Automation Accounts", "Batch accounts", "Genomics accounts", and "Integration accounts". To the left, a sidebar shows "Azure services" with a "Create a resource" button and a "Resources" section listing recent resources like "volstore", "nyc\_taxi\_data", "dbstorage4t", "databricks-rg", and "BigData". The "Marketplace" and "Documentation" sections are also visible.

- ii. Click on storage account, and later you will be redirected to the storage account page, click on create option to create a new storage account.

The screenshot shows the "Storage accounts" list page in the Microsoft Azure portal. The top navigation bar includes "Home > Storage accounts". Below the navigation, there are buttons for "+ Create", "Restore", "Manage view", "Refresh", "Export to CSV", "Open query", "Assign tags", and "Delete". A filter bar allows filtering by "Subscription equals all", "Resource group equals all", and "Location equals all". The main table displays two storage accounts: "dbstorage4t3jfe27f5fc" and "volstore". The columns in the table are "Name", "Type", "Kind", "Resource group", "Location", and "SKU". Both accounts are of type "Storage account" and kind "StorageV2". The first account is associated with "databricks-rg-nyc\_taxi\_data\_etl..." and is located in "East US". The second account is associated with "BigData" and is also located in "East US".

- iii. While creating a storage account, you need to mention the

resource group name you have or wish to create which will hold these resources(blob store, Azure databricks) into a single container.

The screenshot shows the 'Create a storage account' wizard. In the 'Project details' section, the subscription is set to 'Azure subscription 1' and the resource group is 'BigData'. In the 'Instance details' section, the storage account name is left empty, the region is '(US) East US', and the 'Deploy to an edge zone' option is unchecked.

The screenshot shows the 'Create an Azure Databricks workspace' wizard. In the 'Project Details' section, the subscription is 'Azure subscription 1' and the resource group is 'BigData'. In the 'Instance Details' section, the workspace name is 'Enter name for Databricks workspace', the region is 'East US', and the pricing tier is 'Premium (+ Role-based access controls)'. A note indicates that the selected tier is recommended. There is also a field for 'Managed Resource Group name' with 'Enter name for managed resource group' placeholder text.

- iv. Once you create a storage account, you can see that listed in the storage account landing page, now go to the containers option, which is on the left hand side of the storage account panel.

- v. Go to the containers page, and create a new container, keep the settings default unless you want to modify some settings as per your requirements.

- vi. Once the container is created, you will now be able to upload the required data files, any kind of files

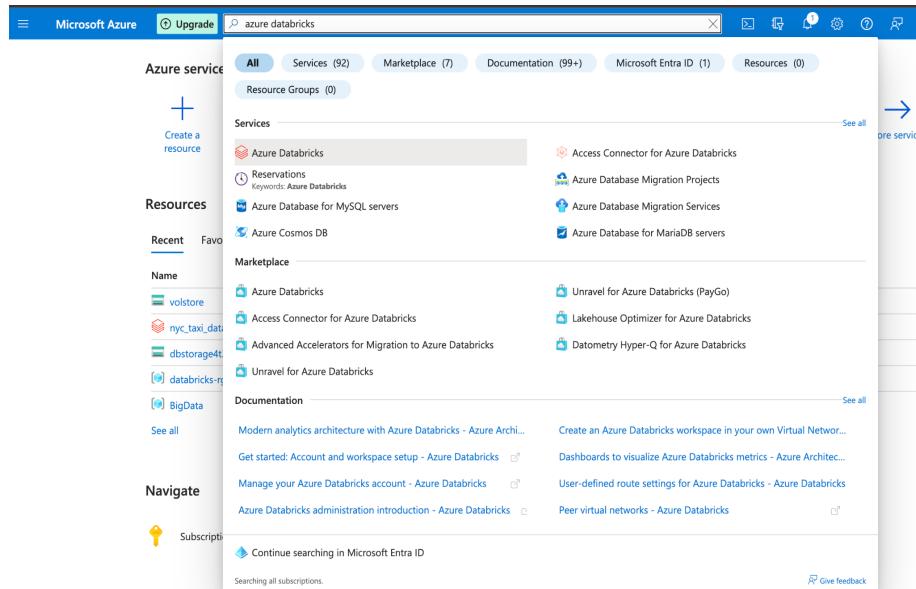
(csv,excel,pdf,parquet, etc.) as this is a storage that supports all kinds and all sizes of formats.

The screenshot shows the Azure Storage Explorer interface. On the left, there's a sidebar with options like Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access tokens, Access policy, Properties, and Metadata. The main area shows a table with columns: Name, Modified, Access tier, Archive status, Blob type, Size, and Lease state. There is one item listed: 'nyc\_taxi\_data\_2014.csv' with a modified date of 12/7/2023, 11:38:22 ..., Hot (Inferred) access tier, Block blob type, 2.36 GB size, and Available lease state. At the top, it says 'Authentication method: Access key [Switch to Microsoft Entra user account]' and 'Location: datastore / Dataset / 2014 / nyc\_taxi\_data\_2014'.

- vii. Please kindly copy the container name, storage account name and the storage account key which we will need when we mount the container onto the azure databricks notebook. You can find the storage access key in the access keys option located on the left side of the panel.

The screenshot shows the 'Access keys' section of the Azure Storage Explorer. It lists two keys: 'key1' and 'key2'. Both keys were last rotated on 12/7/2023 (3 days ago). The 'key1' section includes a 'Show' button for the key and connection string. The 'key2' section also includes a 'Show' button for its details. The sidebar on the left shows other storage account options like 'dbstorage43jfe27f5fc' and 'volstore'.

4. Now, on the search bar type databricks, and click on azure databricks option.



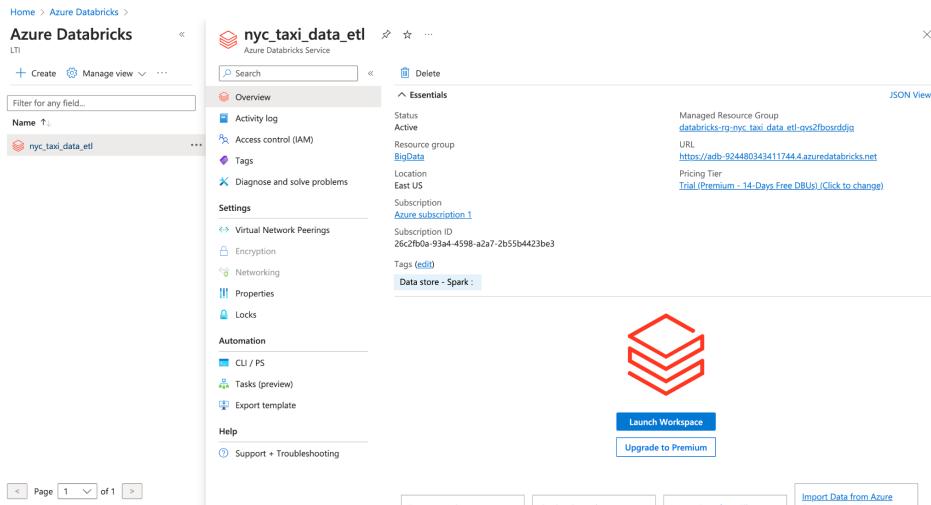
5. Once you are on the azure databricks landing page which looks like this:

- Click on the create workspace button to create a new azure databricks workspace in which you can create notebooks for performing ETL, schedule jobs etc.

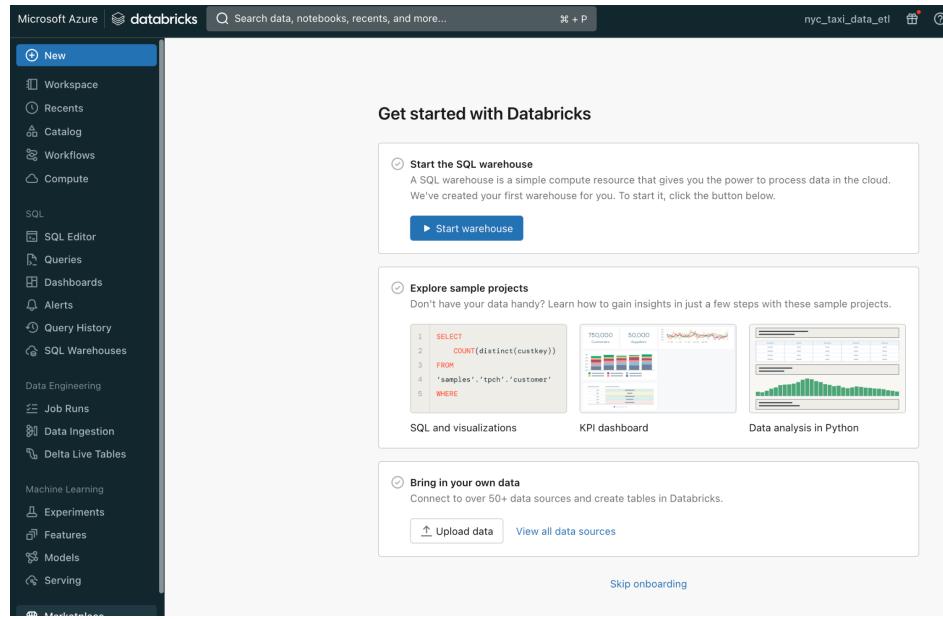
Here, enter the workspace name, select the resource

group where the azure blob storage account is present.  
Select the pricing as per your requirement(*free trial recommended*).

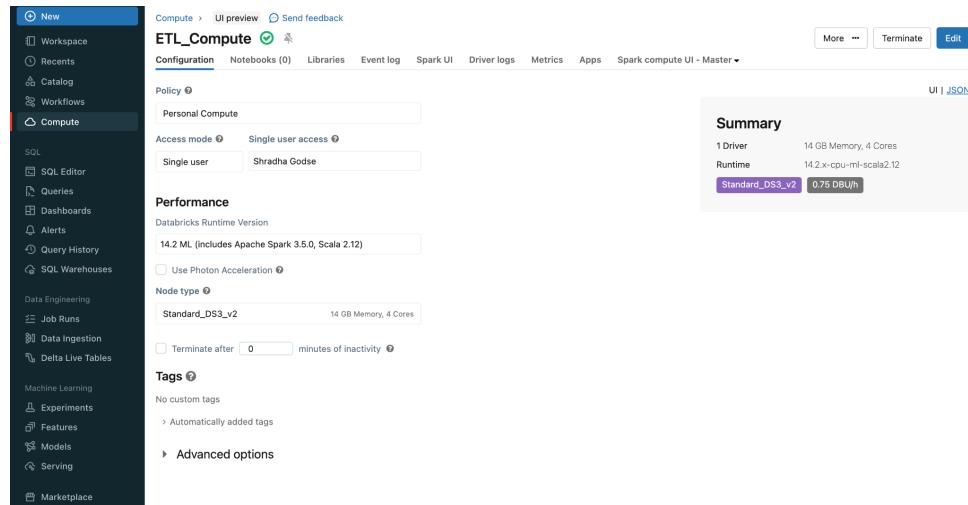
- ii. Once the workspace is created you will see the following screen.



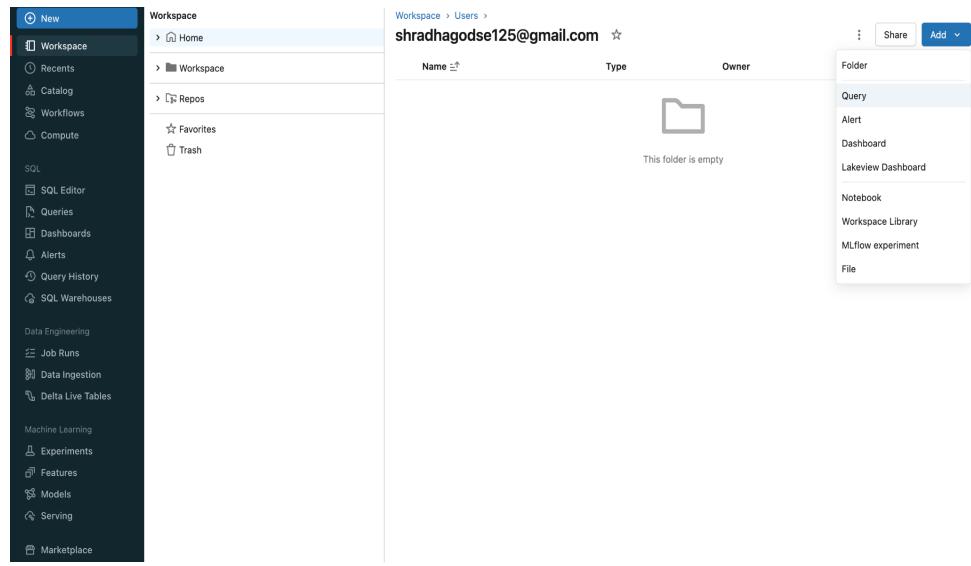
- iii. Click on the launch workspace button, which will redirect you to the azure databricks page.



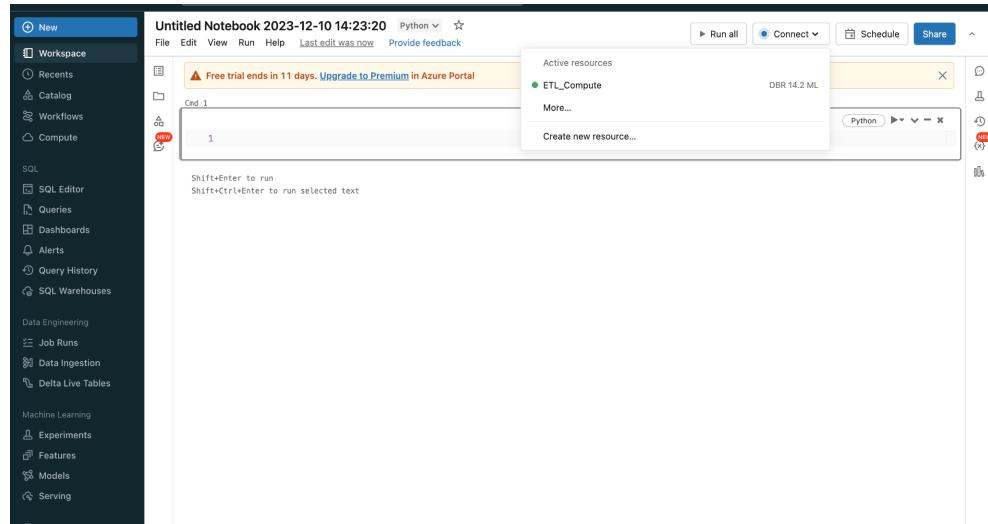
- iv. Go to compute option, once you are on the landing page of compute, click on create compute and keep the below settings. *Note: I kept the compute 14GB RAM, 4 Core cpu for handling the workload more efficiently and to improve performance of the overall data processing and training ML model, also it was free for 14 days).*



- v. Once the compute is created, go to the workspace option located on the left side of the page. Then click on the Add option on the top right corner of the page. And click on the notebook option.



- vi. Once you have created a notebook, now attach the compute we created in Step 2, to the notebook as a compute engine.



6. Once the notebook is created, you can mount the Azure blob storage on the cluster(ETL Compute) which will act as a local storage for this

notebook. We can do that by using the following code:

```
dbutils.fs.unmount("/mnt/blobstorage")

spark.conf.set(
    "fs.azure.account.key." + storage_account_name + ".blob.core.windows.net",
    "Storage_account_access_key"
)

# Define the storage account details
storage_account_name = "storage_account_name"
container_name = "container_name"
storage_account_key = ""

# Mount the Blob Storage container
dbutils.fs.mount(
    source = f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net/",
    mount_point = "/mnt/blobstorage",
    extra_configs =
    {f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": storage_account_key}
)

df = spark.read.option("header",
    "true").csv("/mnt/blobstorage/Dataset/2014/nyc_taxi_data_2014/nyc_taxi_data_2014.csv")
```

*Note:* `dbutils.fs.unmount("/mnt/blobstorage")` this line is to unmount the existing mounted storage.

7. Now, we can continue performing the data preprocessing and transformation process inside the databricks notebook. As this is a python + spark notebook all the required libraries are available and we need to use them just by importing into the code.

The screenshot shows a Databricks notebook interface. The left sidebar contains navigation links for New, Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, and SQL Warehouses. The main area displays a notebook titled "NYC\_Taxi\_trip\_duration\_ETL". The notebook has a single cell titled "Import required library" containing Python code to import various libraries like matplotlib, pandas, numpy, and machine learning models from PySpark and scikit-learn. Below this cell is another cell titled "Data Cleaning". A status bar at the bottom indicates the command took 0.09 seconds to run.

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import numpy as np
4 from pyspark.sql.functions import col, sum as spark_sum
5 from pyspark.ml.feature import OneHotEncoder, StringIndexer
6 from pyspark.ml import Pipeline
7 import seaborn as sns
8 from sklearn.model_selection import train_test_split
9 from pyspark.ml.feature import VectorAssembler
10 from pyspark.ml.regression import RandomForestRegressor
11 from pyspark.ml.classification import LogisticRegression
12 from pyspark.ml.evaluation import RegressionEvaluator, BinaryClassificationEvaluator
13 from pyspark.ml import Pipeline
14 from pyspark.ml.regression import GBTRegressor
15 from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
16 from sklearn.ensemble import RandomForestRegressor
17 from sklearn.ensemble import GradientBoostingRegressor
18 from sklearn.metrics import mean_squared_error
19 from lightgbm import LGBMRegressor as lgb
```

Command took 0.09 seconds -- by shradhagodse125@gmail.com at 12/7/2023, 5:38:15 PM on ETL\_Compute

8. The following preprocessing and transformations are performed on the dataset:

## 1. Data cleaning:

- a. Check the shape of data.
  - b. Check for duplicate records.
  - c. Check for null values.
  - d. Applying filters.

Cmd 5

## Data Cleaning

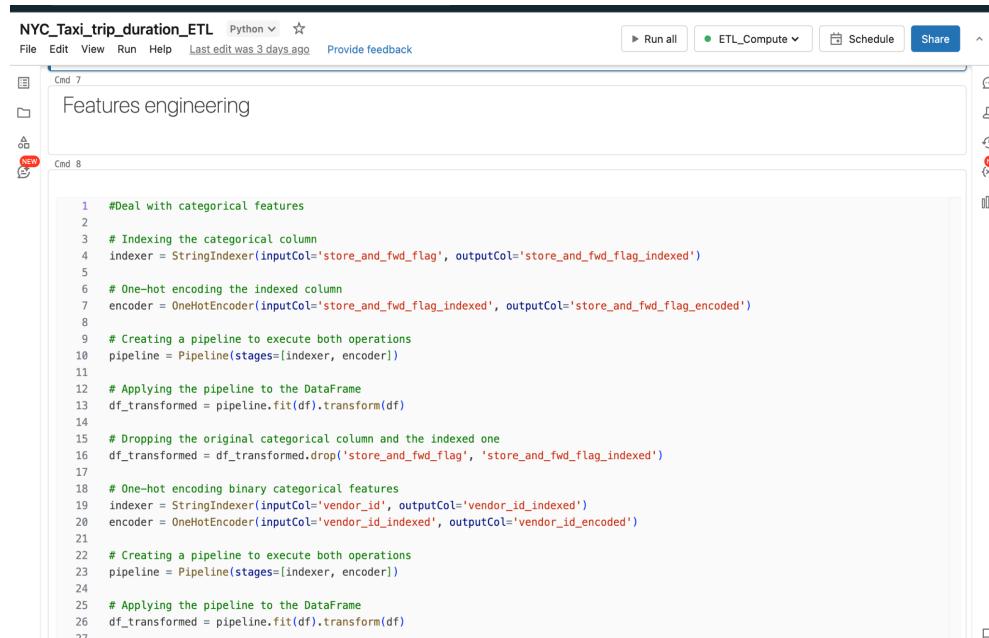
Cmd 6

```
Python ▶ ▷ ▷ - x
```

```
1 # Display data in the dataset
2 df.show(5)
3
4 # Check the number of rows
5 num_rows = df.count()
6
7 # Check the number of columns
8 num_columns = len(df.columns)
9
10 # Display the shape
11 print(f"Shape of the DataFrame: ({num_rows}, {num_columns})")
12
13 # Identify and drop duplicate rows
14 df_no_duplicates = df.dropDuplicates()
15
16 # Count the number of null values for each column
17 null_counts = df.select([spark.sum(col.c).isNotNull().cast("int")).alias(c) for c in df.columns])
18
19 #Only keep trips that lasted less than 5900 seconds
20 df = df[(df.trip_distance < 5900)]
21
22 #Only keep trips with passengers
23 df = df[(df.passenger_count > 0)]
```

## 2. Feature engineering:

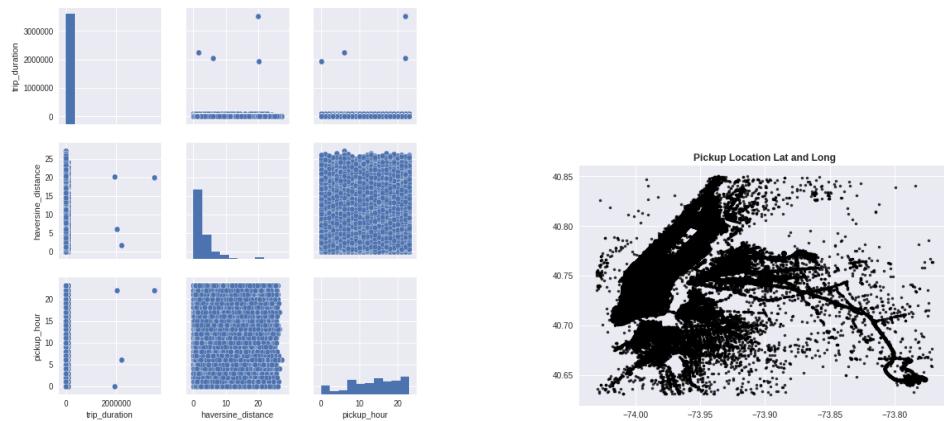
- a. Indexing.
- b. Label encoding categorical columns.
- c. Dropping the columns which are not required.

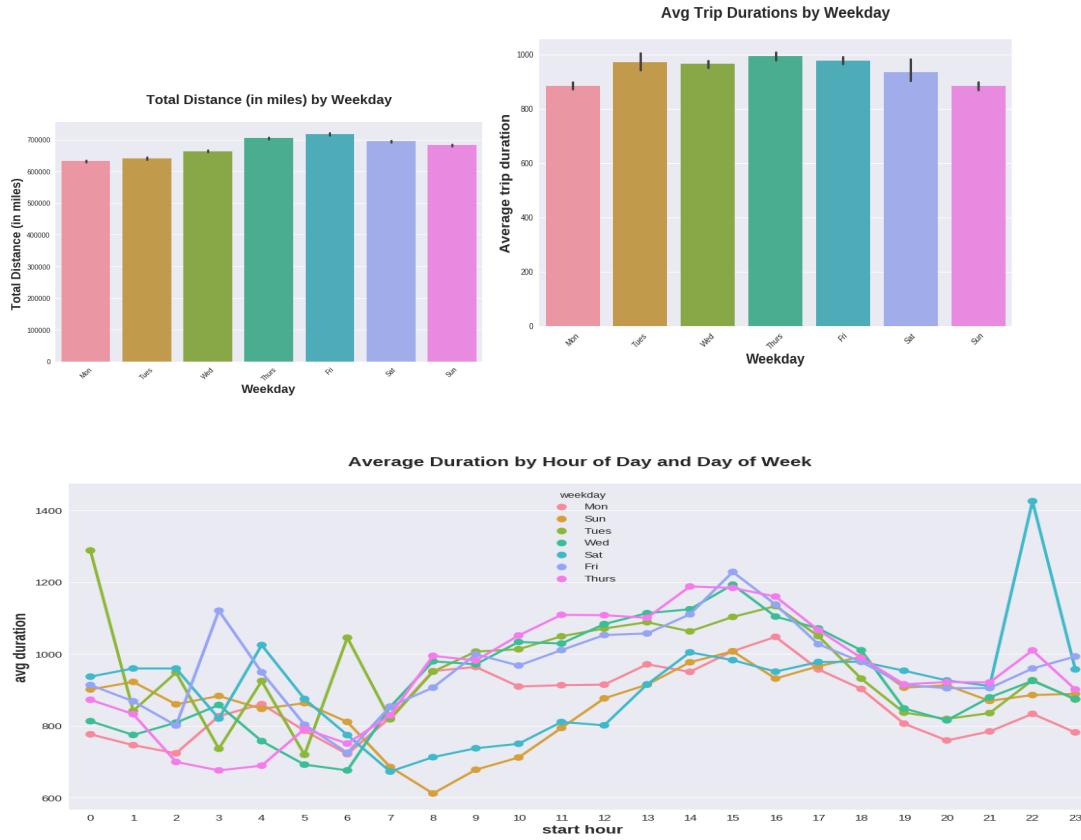


The screenshot shows a Jupyter Notebook interface with the title "NYC\_Taxi\_trip\_duration\_ETL" and a Python file named "Features engineering". The code in the notebook is as follows:

```
1 #Deal with categorical features
2
3 # Indexing the categorical column
4 indexer = StringIndexer(inputCol='store_and_fwd_flag', outputCol='store_and_fwd_flag_indexed')
5
6 # One-hot encoding the indexed column
7 encoder = OneHotEncoder(inputCol='store_and_fwd_flag_indexed', outputCol='store_and_fwd_flag_encoded')
8
9 # Creating a pipeline to execute both operations
10 pipeline = Pipeline(stages=[indexer, encoder])
11
12 # Applying the pipeline to the DataFrame
13 df_transformed = pipeline.fit(df).transform(df)
14
15 # Dropping the original categorical column and the indexed one
16 df_transformed = df_transformed.drop('store_and_fwd_flag', 'store_and_fwd_flag_indexed')
17
18 # One-hot encoding binary categorical features
19 indexer = StringIndexer(inputCol='vendor_id', outputCol='vendor_id_indexed')
20 encoder = OneHotEncoder(inputCol='vendor_id_indexed', outputCol='vendor_id_encoded')
21
22 # Creating a pipeline to execute both operations
23 pipeline = Pipeline(stages=[indexer, encoder])
24
25 # Applying the pipeline to the DataFrame
26 df_transformed = pipeline.fit(df).transform(df)
```

## 3. Exploratory Data Analysis:





## 4. Splitting the data:

```
Cmd 11
Train & Test Split
```

---

```
Cmd 12
1 df['hour']=subset_train.pickup_hour.astype(str)
2 df = df[['log_duration','log_haversine_distance', 'weekday','month','hour']]
3
4 X = df.drop("log_duration",axis=1)
5 y = df["log_duration"]
6 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## 5. Training the model:

I have used 2 machine learning models, gradient boosting and random forest model. Where the split size is 80%-20%.

```

NYC_Taxi_trip_duration_ETL Python ▾ ☆
File Edit View Run Help Last edit was now Provide feedback
Run all ETL_Compute Schedule Share
Training the model
1. Gradient Boosting.
2. Random forest.
Cmd 14 Python
1 # Train the models
2
3 # GradientBoosting
4 gb = GradientBoostingRegressor()
5 gb.fit(X_train, y_train)
6 #print(gb.score(X_train, y_train), gb.score(X_test, y_test))
7 print("classic gradient boosting")
8 print(np.sqrt(mean_squared_error(y_test, np.clip(gb.predict(X_test),0,None))))
9
10 # RandomForest
11 rfm = RandomForestRegressor(bootstrap=True,max_depth=90,max_features='auto',min_samples_split=15,min_samples_leaf=10,
n_estimators=30)
12 rfm.fit(X_train, y_train)
13 print("random forest")
14 print(rfm.score(X_train, y_train), rfm.score(X_test, y_test),rfm.score(y_test, log_duration))

classic gradient boosting
0.74849497458840355 0.7434332056656134
random forest
0.876998567217304 0.813376227423164

Command took 0.04 seconds -- by shradhagodse125@gmail.com at 12/7/2023, 5:32:57 PM on ETL_Compute
Shift+Enter to run
Shift+Ctrl+Enter to run selected text

```

# Conclusion:

- Azure distributed cloud service offers a good solution when it comes to data engineering and big data analysis.
- Azure blob helps in storing unstructured data which can help in building a data lake solution.
- Azure databricks provide intensive features which are really beneficial for performing operations on the dataset. Not only ETL, it also provides features to create dashboards, create workflows where we can schedule jobs(notebooks) for execution.
- The dataset is of NYC Taxi trip duration, and the aim was to predict the time duration required for the taxi.

# References:

- [Azure databricks training](#).
- [Kaggle dataset](#).
- [Azure blob store](#).