Azure Microsoft ML Pipeline for Flight Delay Prediction

Prepared by:
Nor El Islam Messedad

I. Introduction:

The case study focuses on the analysis of flight delay data, with the primary objective of predicting flight delays using historical flight and weather data, and deploying the predictive model as part of a pipeline for real-world implementation. The study takes advantage of Microsoft Azure's services, including Azure Blob Storage, Data Factory, and Databricks, to ingest, process, and analyze data, as well as train the model, to ultimately visualize the prediction output through Power BI.

For the purpose of model training, the problem is framed as a binary classification problem, where flight delays are categorized into two class labels: delays (1, indicating a delay of 15 minutes or more for a departing flight) and no delays (0). The model employs Spark MLlib's Two-Class Decision Tree algorithm to build a predictive model that is capable of accurately determining the likelihood of a flight delay based on historical data.

In this study, data from on-premise sources is retrieved using Azure Data Factory, which orchestrates and automates data movement and transformation. The retrieved unseen data is then scored using the deployed predictive model within the Databricks environment. After the scoring process, data transformation and aggregation are performed to create a summary table that presents an overview of flight delays by origin airport and hour. Power BI is ultimately utilized to visualize the summarized final results Through interactive dashboards to better understand the factors contributing to flight delays, enabling more informed decision-making.

As shown in Figure 01 in the appendices, the tools utilized for building Azure implementation were constructed using DataFactory, Databricks and Storage Account. To make a structured order of how the case study was performed using Azure Microsoft tools, we present the solution architecture in a consistent and sequential manner outlined as follows:

- **Azure Blob Storage and Container:** An Azure Storage Account is created to store and retrieve the unseen data from an on-premise source, which is the local machine to allow for scalable and secure data storage in the cloud.
- **Azure Data Factory:** The Data Factory service is employed to orchestrate and automate data movement and transformation, ingesting unseen data from an on-premise source, which is the local machine, into the Azure Blob Storage for prediction purposes. This plays an important role in model deployment and implementation.
- **Azure Databricks:** Which is an Apache Spark-based analytics platform that is utilized for data processing and model training. The DBFS stores the input datasets (AirportCodeLocationLookupClean, FlightDelaysWithAirportCodes, and FlightWeatherWithAirportCode) are used to train a Two-Class Decision Tree model from Spark MLlib, which predicts flight delays based on historical data. The trained model is then deployed and used to score the unseen data, with results stored back in DBFS.
- **Data Factory and Databricks Integration:** A Data Factory pipeline is created, which connects to the Databricks workspace, and uses the Batch Score script to access the Machine Learning pipeline. The pipeline processes the unseen data pumped from the local machine, producing flight delay predictions.

- **Data Transformation and Aggregation:** The final script in the Databricks workspace processes the scored data, creating a summary table that aggregates flight delays by origin airport and hour. The AirportCodeLocationLookupClean dataset is joined to extract airport coordinates. This new summary table displays the anticipated number of delays for each origin airport at a given hour and day of month.
- **Power BI:** The summary table generated in Databricks is then used as a data source in Power BI for visualization and analysis. Power BI enables users to create interactive dashboards and reports to gain insights into the patterns and trends of flight delays.

This solution architecture leverages Azure services and tools to build an end-to-end data processing, predictive modeling, and streamlined workflow, from data ingestion to visualization by harnessing the use of Azure Data Factory, Databricks, and Power BI.

II. Solution Implementation:

1. Databricks Cluster and Data Loading:

To implement the solution in a seamless manner, we employ Azure Databricks and create a cluster as displayed in Figure 02. Our cluster comprises a set of virtual machines that work together to execute Spark applications for processing large-scale data tasks. Providing that the two historical datasets for training have a large size of 406516 x 20 for flight weather and 2719418 x 29 for flight delays, we need to employ Apache Spark to leverage its scalability and in-memory processing, and hence, reducing the time spent on I/O operations and accelerating the training process. When creating the cluster, it is important to connect it to our Blob Storage in the configuration to access and interact with the storage account and container where our on-premise data files will be stored. However, the sample data for training are stored in the Databricks Files System (DBFS) along with the prediction results as shown in Figure 03.

2. Operation of ML:

In our Workspace, we train the model using three Python scripts. The Data Preparation script cleans, feature-selects, and aggregates the sample datasets into a combined set as ***'flight_delays_with_weather'.*** In the training script, we construct an ML pipeline using the Pipeline API after sampling the class and transforming nominal variables. Although we use GridSearch to optimize the hyperparameters of the Decision Tree, the accuracy remains relatively low 62.16% as provided in Figure 04. This could stem from noise / outliers in the data, multicollinearity or high-dimensionality. To enhance performance, we can conduct further pre-processing of the dataset and test K-fold cross-validation with GridSearchCV using several algorithms to select from. Lastly, we save the model in the Databricks File System as a Spark Pipeline for external access and batch scoring.

3. Model Saving and Deployment:

In the third script, we retrieve the experiment from the training script for model deployment as an API. After serving the model (deactivating cluster) for production as shown in Figure 05, we can access the model externally using its URL from the Production version and access token. We demonstrate testing the model for unseen data using a local Python environment in Figure 06. test_input1 is predicted as a delay while test_input2 outputs no delay.

In this context, serving the model is not a requirement for executing the Data Factory pipeline because the model is accessed via the batch scoring setup script, rather than through the model's API. Within the batch scoring script, the new data is processed by being sent through the pipeline and loading the previously saved model. This approach enables us to generate flight delay predictions and subsequently store the results in DBFS for further analysis or other purposes.

The final script carries out a summarization procedure to aggregate flight delay predictions for each airport code, which is extracted from the AirportCodeLocationLookupClean dataset. The ultimate output is then saved in DBFS, making it accessible for Power BI to visualize and analyze the data.

4. Data Factory Pipeline setup and ML scoring:

In the Data Factory configuration, we first set up the integration runtime with a self-hosted node to create a connection between the on-premise server and the cloud service. This allows data files from the on-premise machine to be transferred into the Data Factory. However, we encounter a connectivity issue, as shown in Figure 07. To troubleshoot the problem, we consult Microsoft's documentation on the error and attempt to resolve it using PowerShell. We also add firewall rules for inbound and outbound connections through port 443, but these efforts are unsuccessful.

Ultimately, we decide to delete the resource group and create a new version, ensuring that all resources are connected to the same resource group. This solution effectively addresses the connectivity issue, as demonstrated in the Figures 08 and 09, and allows the Data Factory to function as intended. Bear in mind that Figure 01 initially referenced encompasses the up-to-date resources.

Next, we create a Data Factory pipeline for data movement by selecting the properties for task scheduling, data source, and Blob storage, where the on-premise data will be stored. We also specify the file name as 'FlightsAndWeather.csv'. This pipeline continuously copies time-sliced CSV files from the on-premises directory (C:\Data) to Azure Blob Storage. In the next stage, we encounter an error when connecting the runtime integration with the Host as shown in Figure. This was stemming from disabled access to local storage, which functions successfully after executing. ‘***dmgcmd.exe -elma -DisableLocalFolderPathValidation’*** on PowerShell as administrator. See Figure 10, 11 and 12. Once the pipeline deployment is complete, we access the Data Factory as an Author and insert a Databricks notebook into the design surface. We then link the notebook to the batch scoring script using the Databricks token, where the machine learning model is saved.

In the final stage, we connect the Copy Data activity to the batch scoring activity, publish all changes, and trigger the pipeline to run. This process generates prediction data and stores it in the designated location in DBFS. As demonstrated in Figure 13, the Monitor tab indicates a successful pipeline run.

5. **Connecting PowerBI to Databricks:**

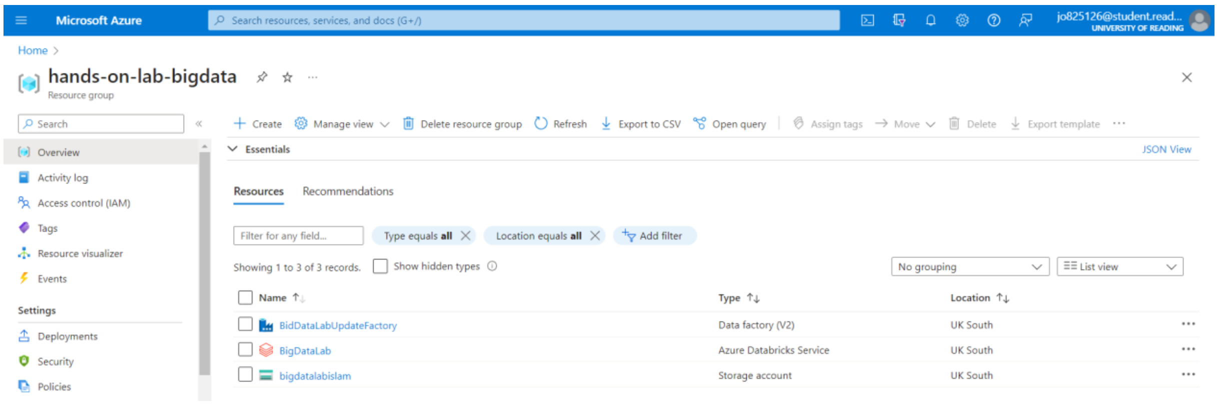
To visualize and analyze the results of our predictive model, we connect Power BI to Databricks through the Spark option to access the summary table generated in the previous step. This process involves establishing a connection to Azure Databricks by providing the necessary credentials and connection details. This involves inputting the Databricks server URL, HTTP path, access token, and the target database where our summary table is stored.

Using the imported data, we create interactive visualizations and dashboards in Power BI to explore patterns and trends in flight delays to help users make informed decisions and identify potential problem areas in flight operations as demonstrated in Figure 14 and 15 and as follows:

- **Map visualization for delays by airport location:** Using (OriginalLatLong), we create a map visualization showing airports as data points, with the size of the points representing the total number of delays.
- **Bar chart for delays over days of the month:** Show the total number of delays per day across the entire month by plotting DayofMonth on the x-axis and NumDelays on the y-axis to identify days with particularly high or low numbers of delays.
- **Treemap of Airports:** Visualize the proportion of delays per airport, with each rectangle representing an airport (OriginAirportCode) and its size proportional to the NumDelays to help compare the number of delays between airports easily.
- **Line chart for delays over days of the month:** Similar to the bar chart, we plot DayofMonth on the x-axis and NumDelays on the y-axis but with a line chart to display patterns in flight delays over time.
- **Scatter chart:** Plot a scatter chart with NumDelays on the y-axis and CRSDepHour on the x-axis. This visualization can help identify any correlation between the number of delays and scheduled departure hours.
- **Bar chart for the number of delays per airport:** Display a bar chart with OriginAirportCode on the x-axis and NumDelays on the y-axis. This visualization can help rank airports based on the total number of flight delays and reveal airports with particularly high or low numbers of delays.

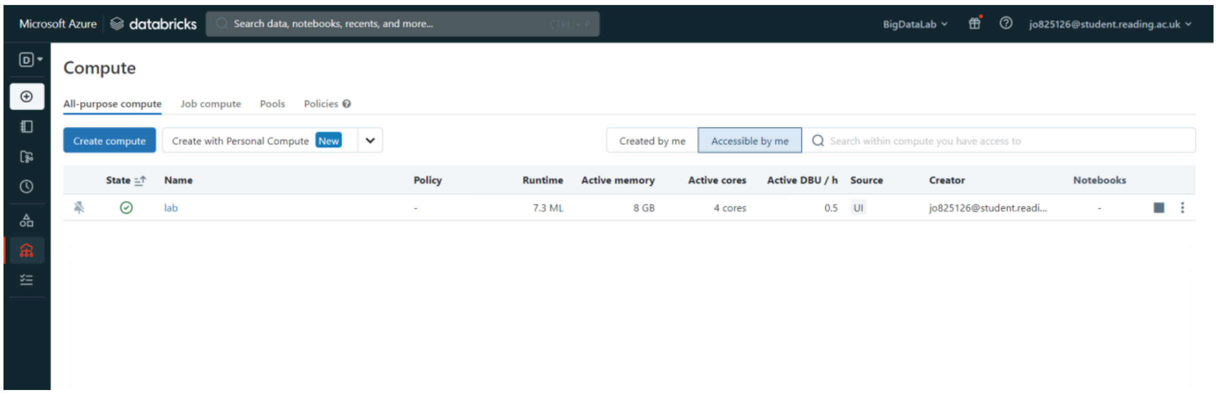
III. Appendices

Figure 01: Resource group creation with the necessary tools



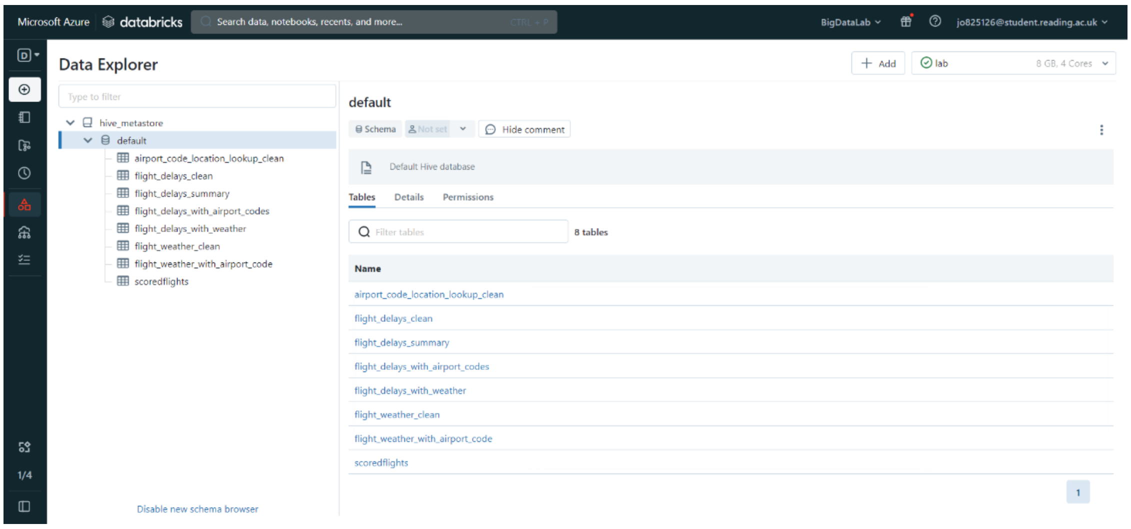
Name	Type	Location
BidDataLabUpdateFactory	Data factory (V2)	UK South
BigDataLab	Azure Databricks Service	UK South
bigdatalabislam	Storage account	UK South

Figure 02: Creation of Apache Spark Cluster


Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
BigDataLab
jo825126@student.reading.ac.uk
Compute
All-purpose compute
Job compute
Pools
Policies
Create compute
Create with Personal Compute
New
Created by me
Accessible by me
Search within compute you have access to

State	Name	Policy	Runtime	Active memory	Active cores	Active DBU / h	Source	Creator	Notebooks
✓	lab	-	7.3 ML	8 GB	4 cores	0.5	UI	jo825126@student.readi...	-

Figure 03: Databricks Files System uploaded and generated files



default

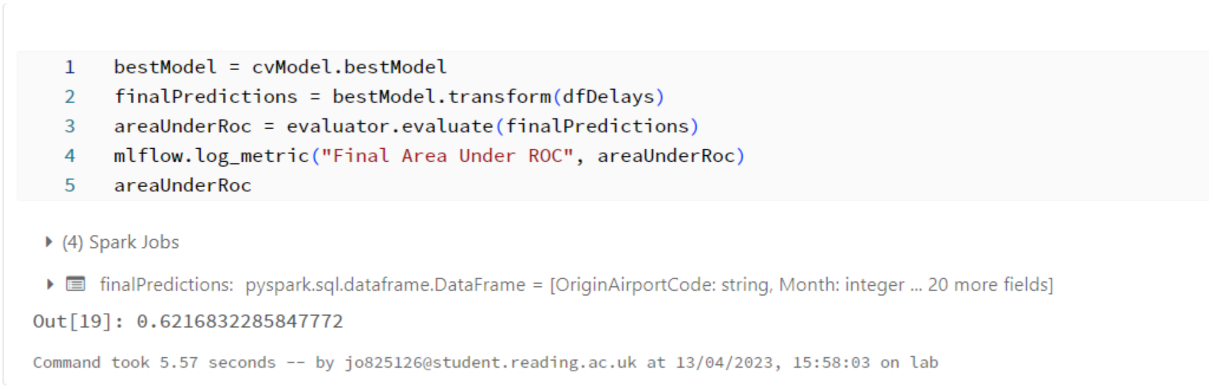
Default Hive database

8 tables

hive_metastore
- default
  - airport_code_location_lookup_clean
  - flight_delays_clean
  - flight_delays_summary
  - flight_delays_with_airport_codes
  - flight_delays_with_weather
  - flight_weather_clean
  - flight_weather_with_airport_code
  - scoredflights

Name
airport_code_location_lookup_clean
flight_delays_clean
flight_delays_summary
flight_delays_with_airport_codes
flight_delays_with_weather
flight_weather_clean
flight_weather_with_airport_code
scoredflights

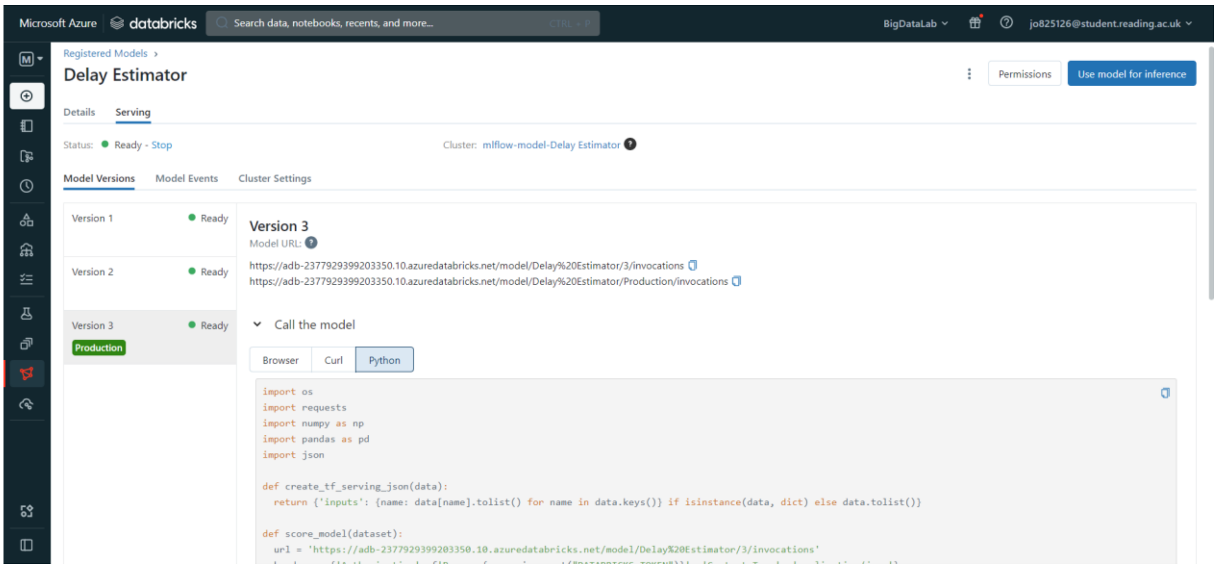
Figure 04: Demonstrating the performance of the optimal model



```
bestModel = cvModel.bestModel
finalPredictions = bestModel.transform(dfDelays)
areaUnderRoc = evaluator.evaluate(finalPredictions)
mlflow.log_metric("Final Area Under ROC", areaUnderRoc)
areaUnderRoc
```

```
(4) Spark Jobs
finalPredictions:  pyspark.sql.dataframe.DataFrame = [OriginAirportCode: string, Month: integer ... 20 more fields]
Out[19]: 0.6216832285847772

Command took 5.57 seconds -- by jo825126@student.reading.ac.uk at 13/04/2023, 15:58:03 on lab
```

Figure 05: Model serving successful



Delay Estimator

Status: Ready - Stop

Cluster: mlflow-model-Delay Estimator

Version 1	Ready
Version 2	Ready
Version 3 Production	Ready

Version 3

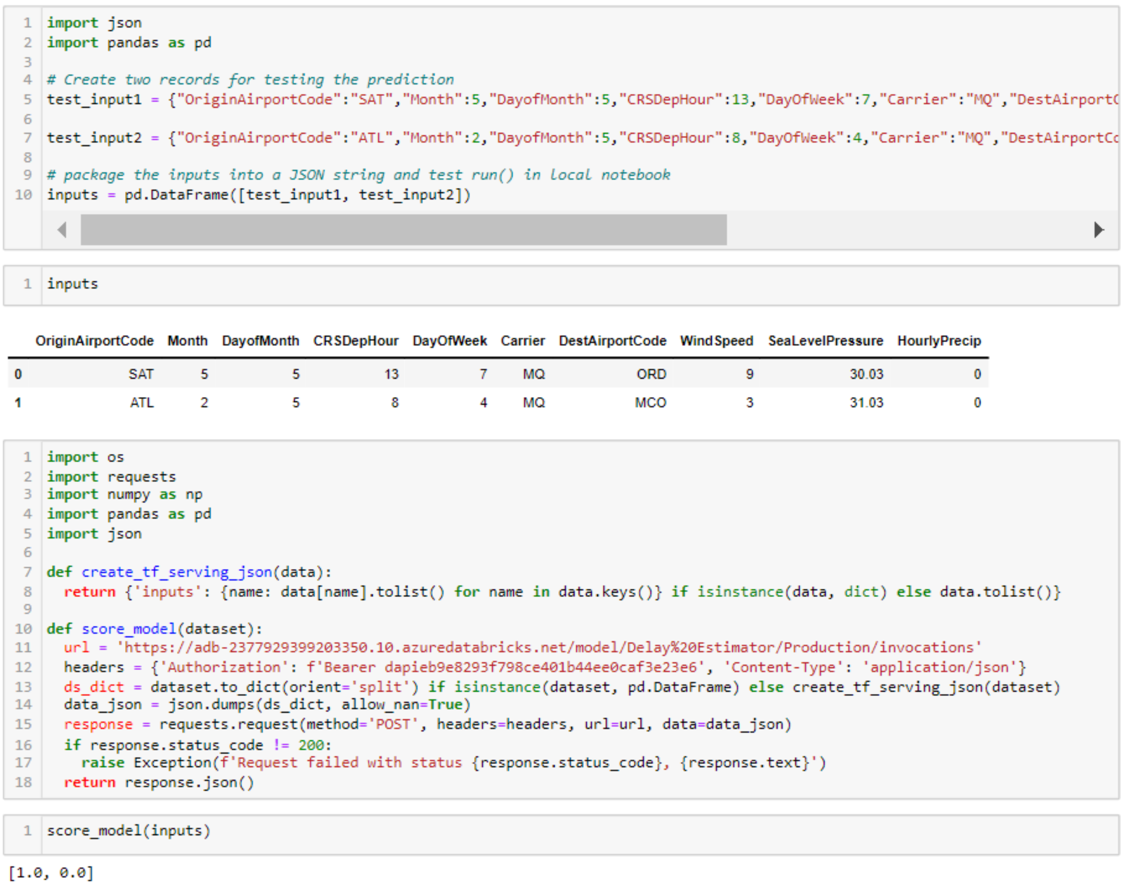
Model URL:

https://adb-2377929399203350.10.azuredatabricks.net/model/Delay%20Estimator/3/invocations
https://adb-2377929399203350.10.azuredatabricks.net/model/Delay%20Estimator/Production/invocations

Call the model

```
import os
import requests
import numpy as np
import pandas as pd
import json

def create_tf_serving_json(data):
  return {'inputs': {name: data[name].tolist() for name in data.keys()} if isinstance(data, dict) else data.tolist()}

def score_model(dataset):
  url = 'https://adb-2377929399203350.10.azuredatabricks.net/model/Delay%20Estimator/3/invocations'
```

Figure 06: Testing the Model API's using a local Python environment



```
import json
import pandas as pd

# Create two records for testing the prediction
test_input1 = {"OriginAirportCode":"SAT","Month":5,"DayofMonth":5,"CRSDepHour":13,"DayOfWeek":7,"Carrier":"MQ","DestAirport(

test_input2 = {"OriginAirportCode":"ATL","Month":2,"DayofMonth":5,"CRSDepHour":8,"DayOfWeek":4,"Carrier":"MQ","DestAirportCo

# package the inputs into a JSON string and test run() in local notebook
inputs = pd.DataFrame([test_input1, test_input2])
```

```
inputs
```

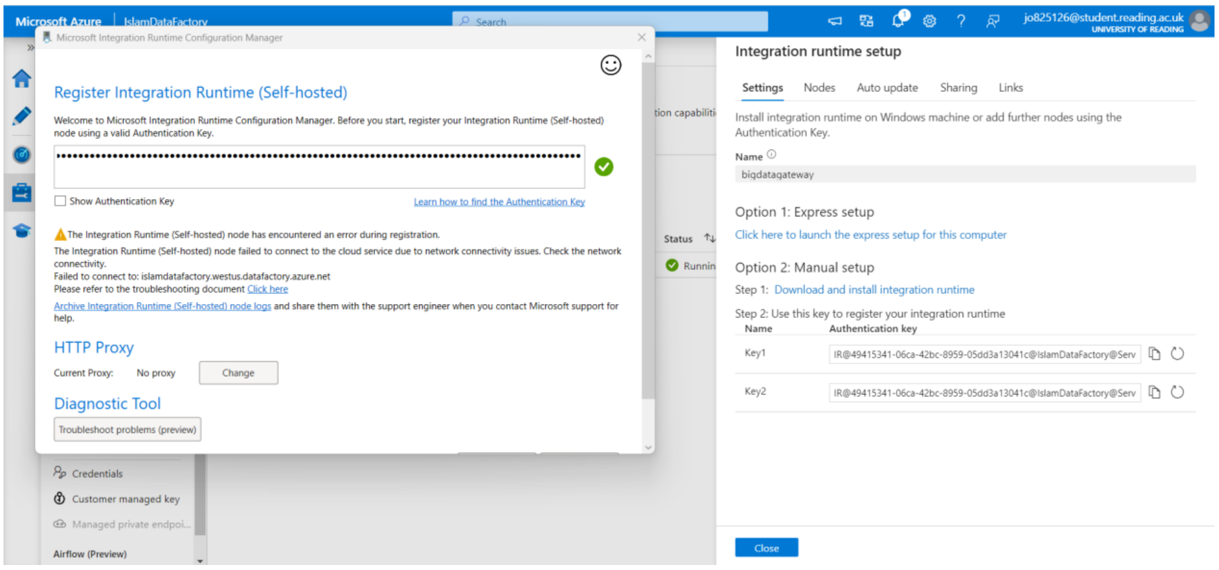
	OriginAirportCode	Month	DayofMonth	CRSDepHour	DayOfWeek	Carrier	DestAirportCode	WindSpeed	SeaLevelPressure	HourlyPrecip
0	SAT	5	5	13	7	MQ	ORD	9	30.03	0
1	ATL	2	5	8	4	MQ	MCO	3	31.03	0

```
import os
import requests
import numpy as np
import pandas as pd
import json

def create_tf_serving_json(data):
  return {'inputs': {name: data[name].tolist() for name in data.keys()} if isinstance(data, dict) else data.tolist()}

def score_model(dataset):
  url = 'https://adb-2377929399203350.10.azuredatabricks.net/model/Delay%20Estimator/Production/invocations'
  headers = {'Authorization': f'Bearer dapieb9e8293f798ce401b44ee0caf3e23e6', 'Content-Type': 'application/json'}
  ds_dict = dataset.to_dict(orient='split') if isinstance(dataset, pd.DataFrame) else create_tf_serving_json(dataset)
  data_json = json.dumps(ds_dict, allow_nan=True)
  response = requests.request(method='POST', headers=headers, url=url, data=data_json)
  if response.status_code != 200:
    raise Exception(f'Request failed with status {response.status_code}, {response.text}')
  return response.json()
```

```
score_model(inputs)
```

```
[1.0, 0.0]
```

Figure 07: Runtime integration connection error



Register Integration Runtime (Self-hosted)

Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key.

••

Show Authentication Key

Learn how to find the Authentication Key

The Integration Runtime (Self-hosted) node has encountered an error during registration.

The Integration Runtime (Self-hosted) node failed to connect to the cloud service due to network connectivity issues. Check the network connectivity.
Failed to connect to: islamdatafactory.westus.datafactory.azure.net
Please refer to the troubleshooting document Click here

Archive Integration Runtime (Self-hosted) node logs and share them with the support engineer when you contact Microsoft support for help.

HTTP Proxy

Current Proxy: No proxy Change

Diagnostic Tool

Troubleshoot problems (preview)

Integration runtime setup

Settings Nodes Auto update Sharing Links

Install integration runtime on Windows machine or add further nodes using the Authentication Key.

Name
bigdatagateway

Option 1: Express setup

Click here to launch the express setup for this computer

Option 2: Manual setup

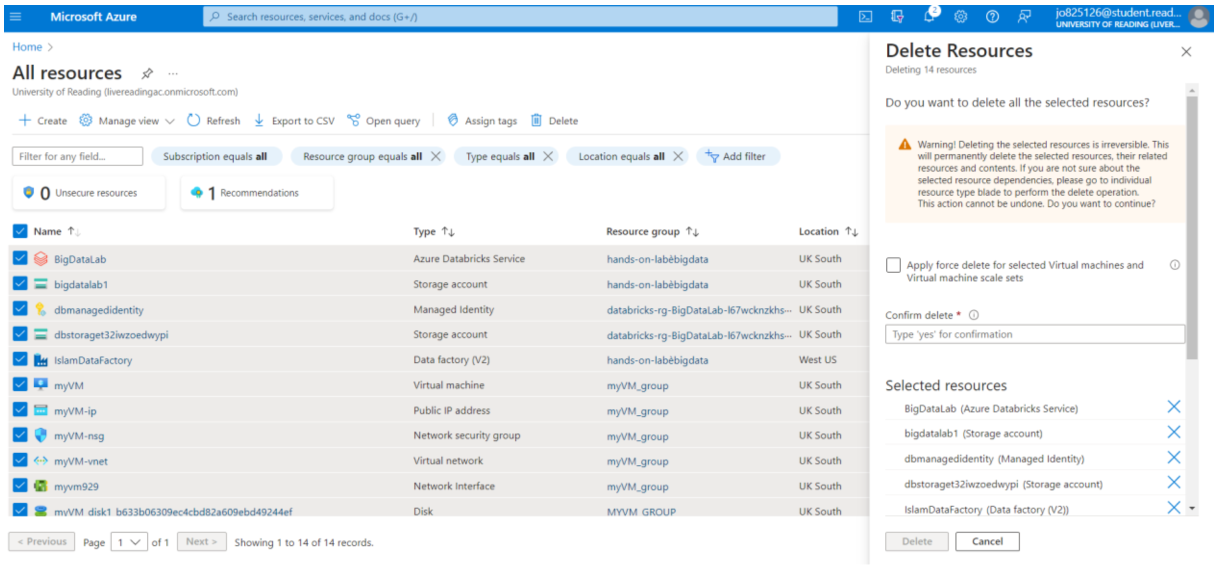
Step 1: Download and install integration runtime

Step 2: Use this key to register your integration runtime

Name	Authentication key
Key1	IR@49415341-06ca-42bc-8959-05dd3a13041c@IslamDataFactory@Serv
Key2	IR@49415341-06ca-42bc-8959-05dd3a13041c@IslamDataFactory@Serv

Close

Figure 08: Deletion of all resources


Home >

All resources

University of Reading (livereadingac.onmicrosoft.com)

Subscription equals **all** | Resource group equals **all** | Type equals **all** | Location equals **all**

0 Unsecure resources | 1 Recommendations

Name	Type	Resource group	Location
BigDataLab	Azure Databricks Service	hands-on-labèbigdata	UK South
bigdatalab1	Storage account	hands-on-labèbigdata	UK South
dbmanagedidentity	Managed Identity	databricks-rg-BigDataLab-l67wcknzkhs···	UK South
dbstoraget32iwzoedwypi	Storage account	databricks-rg-BigDataLab-l67wcknzkhs···	UK South
IslamDataFactory	Data factory (V2)	hands-on-labèbigdata	West US
myVM	Virtual machine	myVM_group	UK South
myVM-ip	Public IP address	myVM_group	UK South
myVM-nsg	Network security group	myVM_group	UK South
myVM-vnet	Virtual network	myVM_group	UK South
myvm929	Network Interface	myVM_group	UK South
myVM_disk1_b633b06309ec4cbd82a609ebd49244ef	Disk	MYVM_GROUP	UK South

Page 1 of 1. Showing 1 to 14 of 14 records.

Delete Resources

Deleting 14 resources

Do you want to delete all the selected resources?

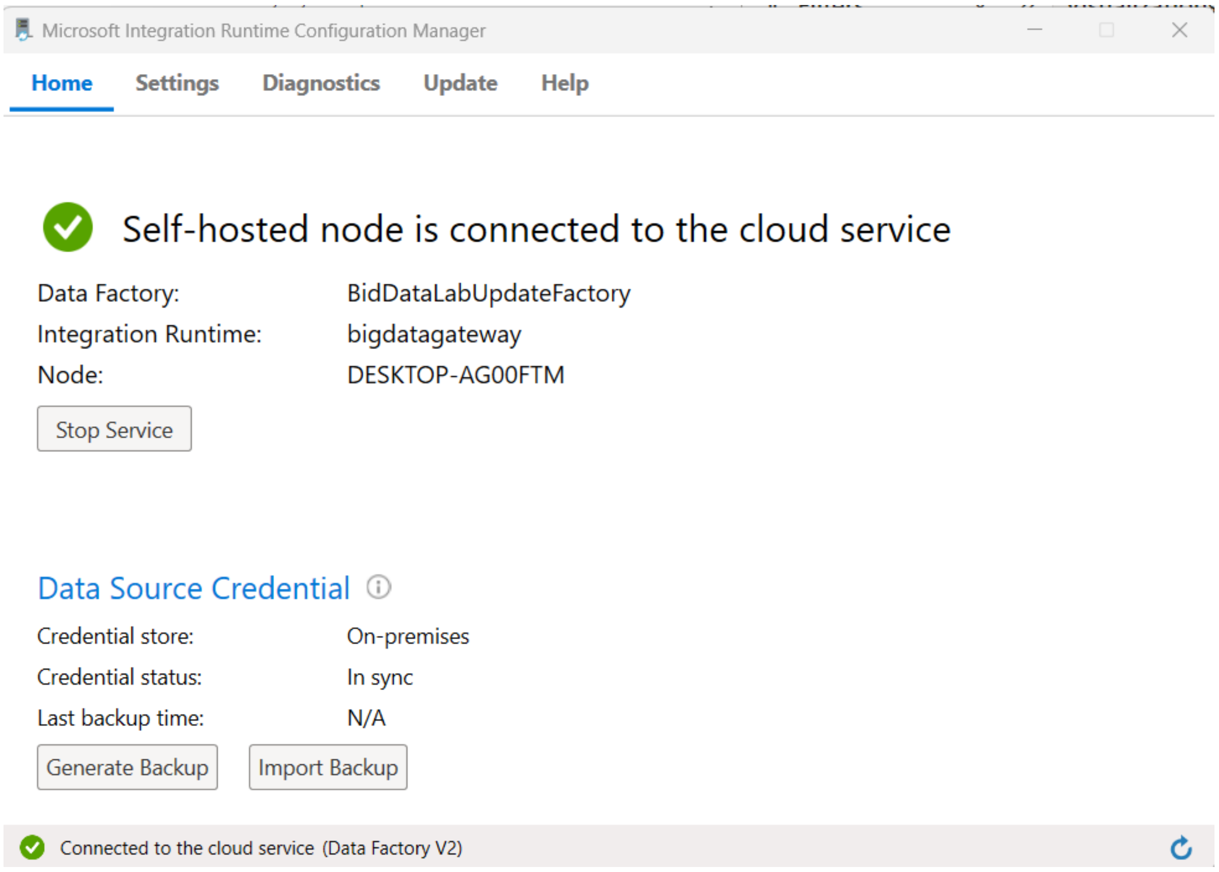
Warning! Deleting the selected resources is irreversible. This will permanently delete the selected resources, their related resources and contents. If you are not sure about the selected resource dependencies, please go to individual resource type blade to perform the delete operation. This action cannot be undone. Do you want to continue?

☐ Apply force delete for selected Virtual machines and Virtual machine scale sets

Confirm delete *: Type 'yes' for confirmation

Selected resources
- BigDataLab (Azure Databricks Service)
- bigdatalab1 (Storage account)
- dbmanagedidentity (Managed Identity)
- dbstoraget32iwzoedwypi (Storage account)
- IslamDataFactory (Data factory (V2))

Figure 09: Successful connection to the runtime integration


Microsoft Integration Runtime Configuration Manager

Self-hosted node is connected to the cloud service

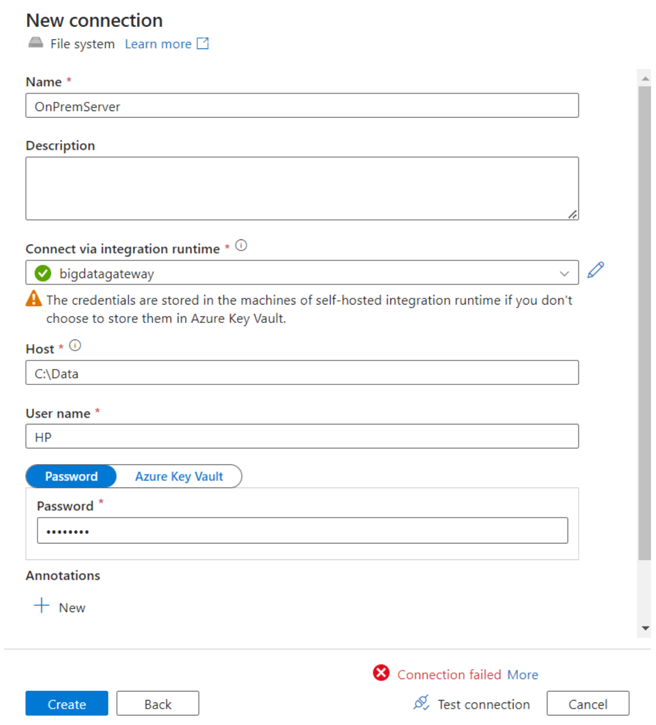
Data Factory:	BidDataLabUpdateFactory
Integration Runtime:	bigdatagateway
Node:	DESKTOP-AG00FTM

Data Source Credential

Credential store:	On-premises
Credential status:	In sync
Last backup time:	N/A

Connected to the cloud service (Data Factory V2)

Figure 10: Host connection error


New connection

File system Learn more

Name *

OnPremServer

Description

Connect via integration runtime *

bigdatagateway

The credentials are stored in the machines of self-hosted integration runtime if you don't choose to store them in Azure Key Vault.

Host *

C:\Data

User name *

HP

Password | Azure Key Vault

Password *

••••••••

Annotations

Figure 11: PowerShell Administrator solution

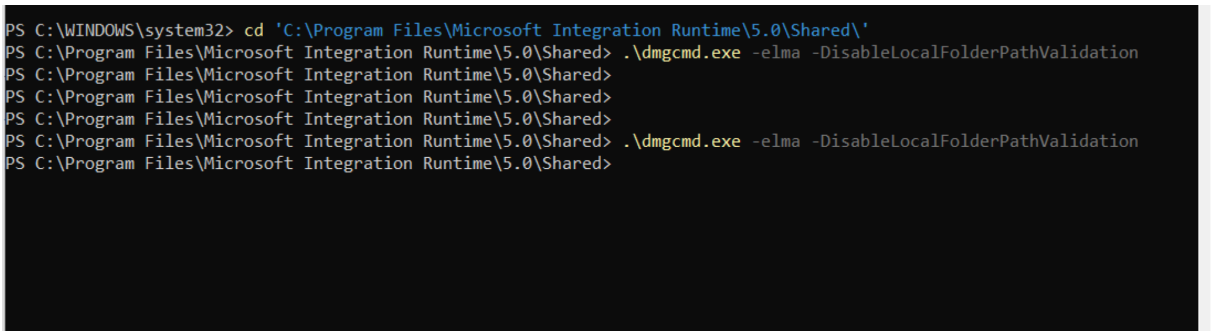
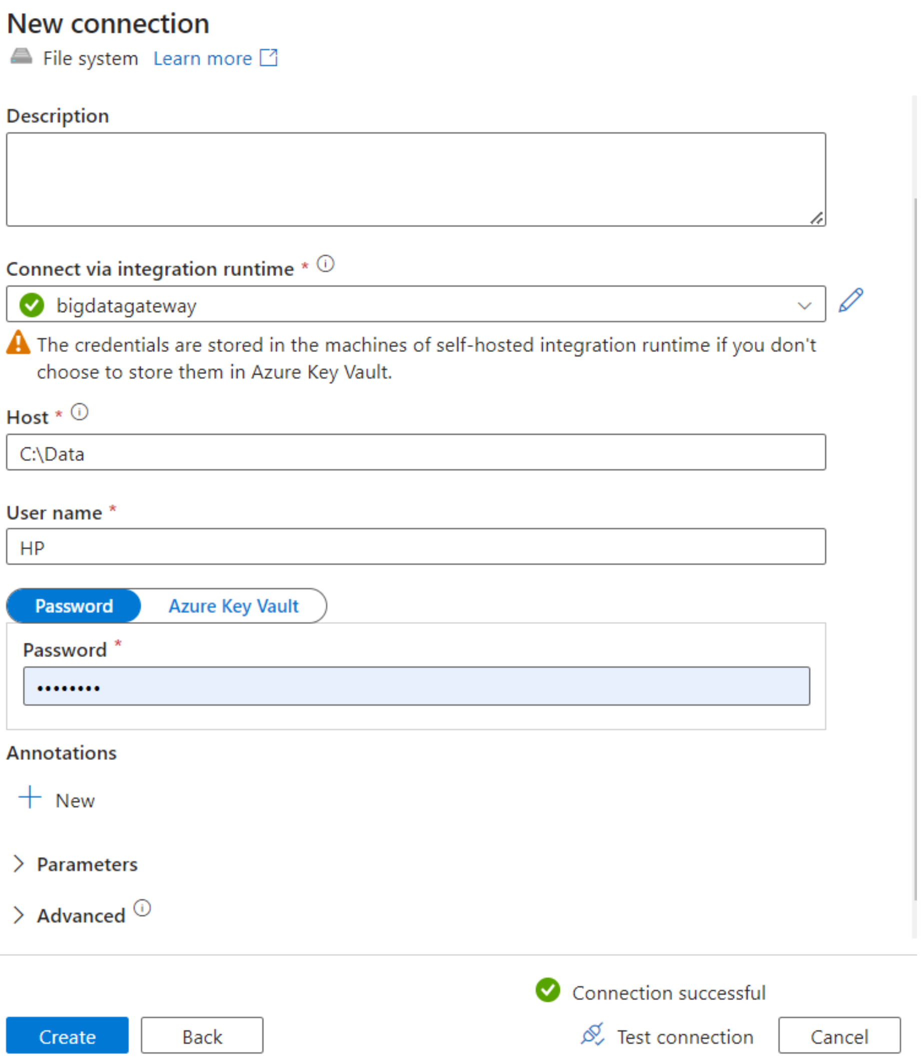

```
PS C:\WINDOWS\system32> cd 'C:\Program Files\Microsoft Integration Runtime\5.0\Shared\'
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared> .\dmgcmd.exe -elma -DisableLocalFolderPathValidation
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared>
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared>
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared>
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared> .\dmgcmd.exe -elma -DisableLocalFolderPathValidation
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared>
```

Figure 12: Host connection successful



New connection

File system Learn more

Description

Connect via integration runtime *

bigdatageway

The credentials are stored in the machines of self-hosted integration runtime if you don't choose to store them in Azure Key Vault.

Host *

C:\Data

User name *

HP

Password | Azure Key Vault

Password *

••••••••

Annotations

New

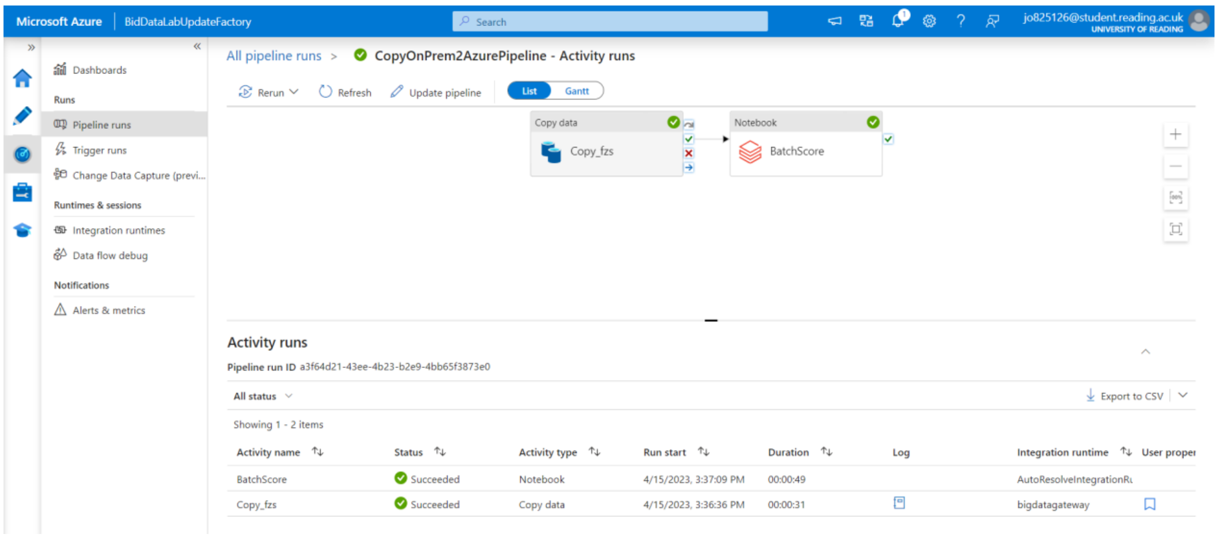
Parameters

Advanced

Connection successful

Create | Back | Test connection | Cancel

Figure 13: Workflow triggering successfully



Microsoft Azure | BidDataLabUpdateFactory

All pipeline runs > CopyOnPrem2AzurePipeline - Activity runs

Activity runs

Pipeline run ID a3f64d21-43ee-4b23-b2e9-4bb65f3873e0

All status

Export to CSV

Showing 1 - 2 items

Activity name	Status	Activity type	Run start	Duration	Log	Integration runtime	User proper
BatchScore	✓ Succeeded	Notebook	4/15/2023, 3:37:09 PM	00:00:49		AutoResolveIntegrationRu	
Copy_fzs	✓ Succeeded	Copy data	4/15/2023, 3:36:36 PM	00:00:31	▤	bigdatagateway	▯

Figure 14: PowerBI first visualization page

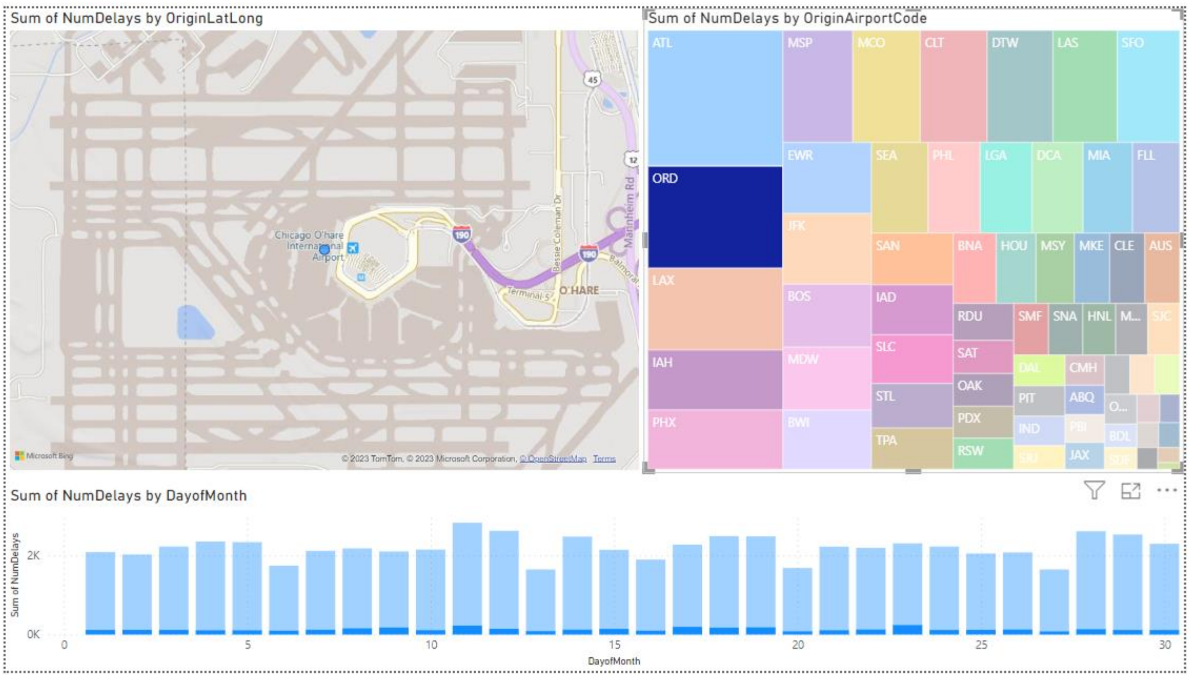
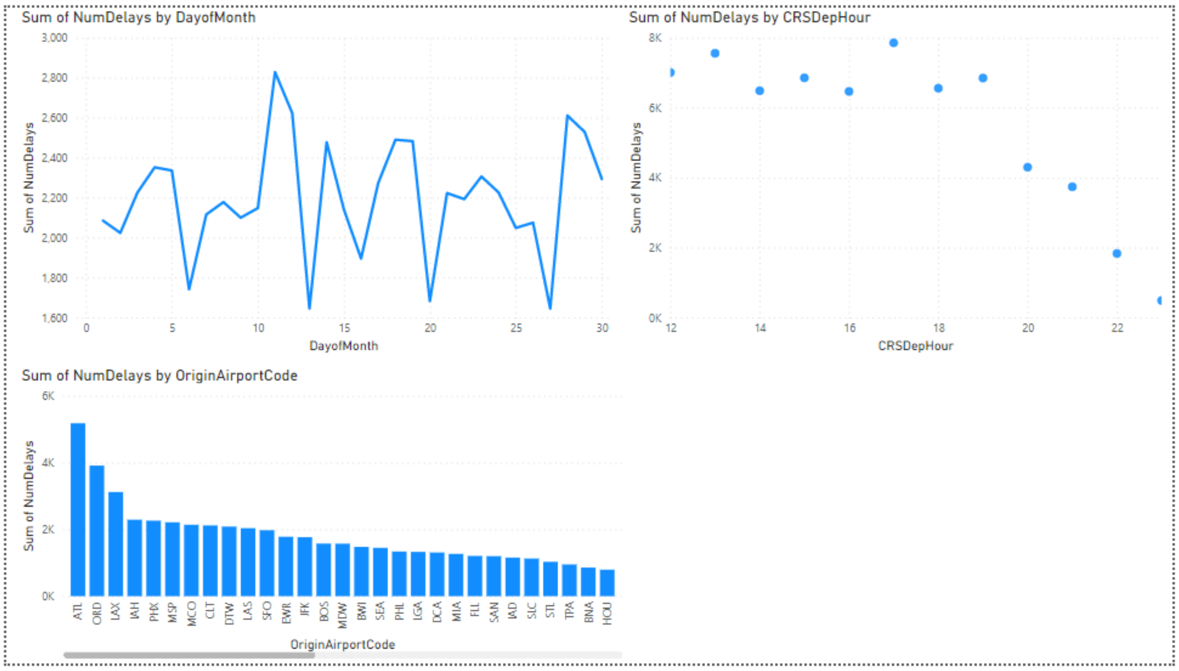


Figure 15: PowerBI second visualization page



Sum of NumDelays by DayofMonth
Sum of NumDelays
3,000
2,800
2,600
2,400
2,200
2,000
1,800
1,600
0
5
10
15
20
25
30
DayofMonth
Sum of NumDelays by CRSDepHour
8K
6K
4K
2K
0K
12
14
16
18
20
22
CRSDepHour
Sum of NumDelays by OriginAirportCode
6K
4K
2K
0K
ATL
ORD
LAX
IAH
PHX
MSP
MCO
CLT
DTW
LAS
SFO
EWR
JFK
BOS
MDW
BWI
SEA
PHL
LGA
DCA
MIA
FLL
SAN
IAD
SLC
STL
TPA
BNA
HOU
OriginAirportCode