

# FLIGHT DELAY PREDICTION

## GROUP 3 - CAPSTONE PROJECT

- |    |                       |        |
|----|-----------------------|--------|
| 1. | Amol Nikam            | H22002 |
| 2. | Anindita Roy          | H22003 |
| 3. | Anoushka Raj Rao      | H22004 |
| 4. | Kartik Mudgal         | H22007 |
| 5. | Manasi Ghodake        | H22010 |
| 6. | Sidharth Khurana      | H22016 |
| 7. | Srikrishna Srinivasan | H22017 |

PGPDE 2022-23



Flight	Destination	Time	Remarks	Gate
AN 110	HONG KONG	18:10	DELAYED	--
BY 377	KUALA LUMPUR	18:22	DELAYED	--
SX 429	NEW YORK	18:27	DELAYED	--
MH 080	LONDON	18:34	DELAYED	--

# INDEX

1. Background & Business Justification
2. Problem Statement
3. About the dataset
4. Methodology
5. EDA
6. Machine learning model
7. Results
8. Conclusion and Impact
9. Next Steps

# Background & Business Justification

Nowadays, the aviation industry plays a crucial role in the world's transportation sector, and a lot of businesses rely on various airlines to connect them with other parts of the world. One of the key business issues that airlines face is that the vital prices that are related to flights being delayed because of natural occurrences and operational shortcomings that is an upscale affair for the airlines, making issues in scheduling and operations for the end users therefore inflicting unhealthy name and client discontent

To solve this issue, accurately predicting these flight delays allows passengers to be well prepared for the deterrent caused to their journey and enables airlines to respond to the potential causes of the flight delays in advance to diminish the negative impact.

The purpose of this project is to look at the approaches used to build models for predicting flight delays that occur due to bad weather conditions.

In the first part of the project, we primarily focus on gathering a dataset from SQL, Cosmos db, and web API data. We will be using Azure Data factory for transformation - joins, and cleaning .In the second part of the project, we primarily focus on modeling of the data. The Machine Learning Model used for this problem statement is Decision Tree Regressor. Azure ML Studio provides us with a No Code way of preparing our dataset and training our model.

# PROBLEM STATEMENT

- The problem statement is to develop a data engineering pipeline for **Flight Delay Prediction**, which involves integrating and cleaning data from multiple sources, performing EDA, transforming the data, and building and deploying a machine learning model.
- The aim is to create a reliable and efficient system for predicting flight departure delays, using a consistent and accurate dataset that can inform the predictive model.
- A successful application of this model could lead to improved operational efficiency, customer satisfaction, and profitability for the airline industry.



# Salient Points :



## DATA INGESTION

Ingest data from different sources namely - SQL database, NoSQL database and Web API

## DATA CLEANING

Perform a simple preprocessing of data, removing redundant columns, Treating missing values, etc.

## DATA TRANSFORMATION

Join the different datasets which are in different formats using azure data flow

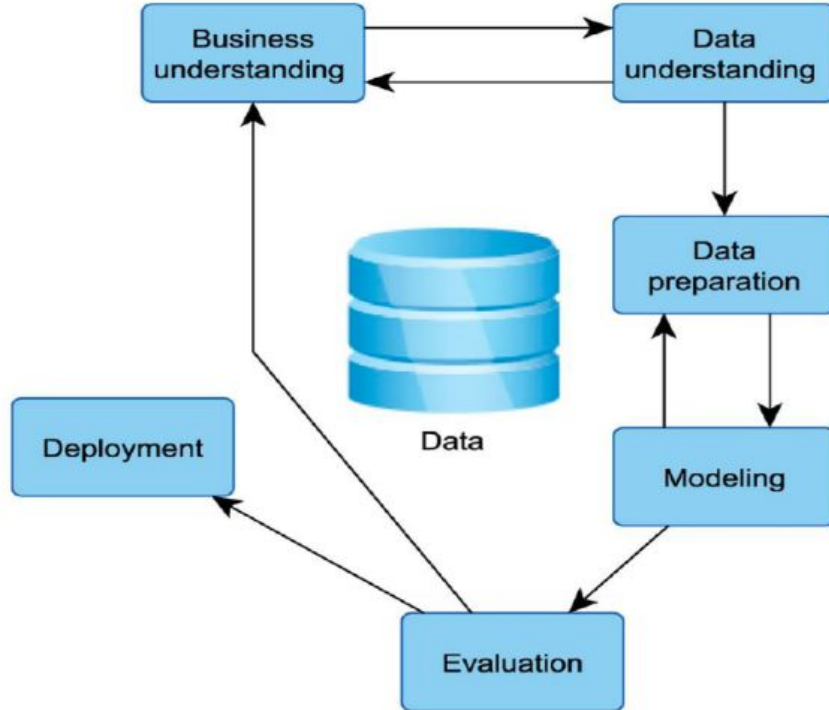
## EDA

Perform extensive analysis of the combined master dataset which was made in azure data flow

## DATA MODELLING

Prepare a ML pipeline for predicting departure delays and automate from ingestion to modelling in ADF.

# ARCHITECTURE:




# ABOUT THE DATASET

The data used for this project is a comprehensive collection of flight information from US airports. It is divided into 4 datasets, as follows:

Dataset : <https://drive.google.com/drive/folders/1KYK2lp6fNSc247zQ6QwkDdcQ6L7CSCLe>

The Flights data, Airports Data and Airlines data were initially in CSV file, but to simulate how a organization receives data from multiple sources , we have changed then saved the files in SQL and Cosmos db

1. Flights Data - this transactional dataset contains over 6 million records, covering flight schedules and actual departure and arrival times- SQL database
  2. Airports Data - this dataset contains IATA airport codes, names and other geographic details.- Cosmos db
  3. Airlines Data - this dataset contains IATA airline codes and names.- Cosmos db
  4. Weather Data - this dataset contains details of weather conditions during each scheduled flight- Raw data of Web API data stored data lake
- 

# METHODOLOGY - DATA INGESTION

## Data Ingestion Pipeline

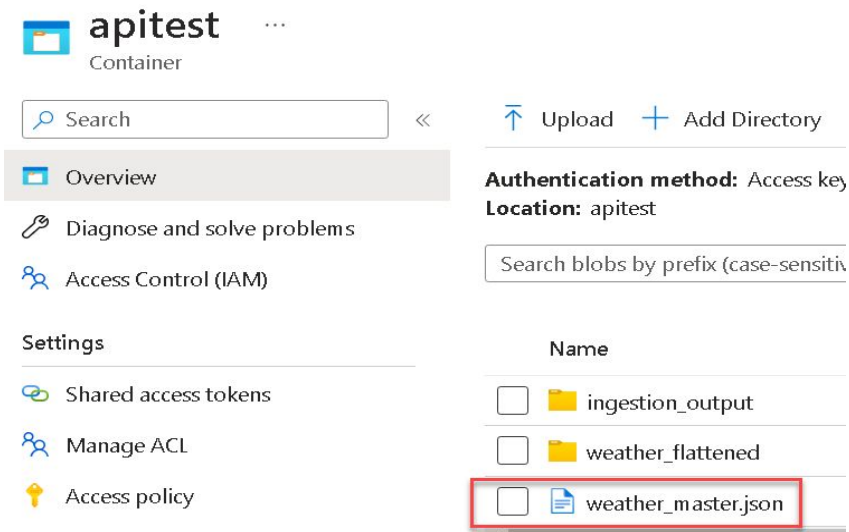
- Sources - Azure SQL DB, CosmosDB, web API
- Tool used - Azure Data Factory
- Steps - data collection, transformation - joins, cleaning





# Data ingestion(Web api)

The weather data is fetched for the year of 2015 from a web api which has limits on the number of calls per minute, hence the data was gathered locally and raw output from the api was combined and imported into the data lake.



The screenshot shows the Azure Storage Explorer interface for a container named 'apitest'. The left sidebar contains a search bar and a navigation menu with 'Overview' selected, followed by 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. The 'Settings' section lists 'Shared access tokens', 'Manage ACL', and 'Access policy'. The main pane displays the container's contents, including a search bar, 'Upload' and 'Add Directory' buttons, and a table of blobs. The table has a header 'Name' and lists three items: 'ingestion\_output' (folder), 'weather\_flattened' (folder), and 'weather\_master.json' (file). The 'weather\_master.json' file is highlighted with a red box.

**apitest** Container

Search

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Manage ACL

Access policy

Authentication method: Access key  
Location: apitest










Search blobs by prefix (case-sensitive)

Name
<input type="checkbox"/> ingestion_output
<input type="checkbox"/> weather_flattened
<input type="checkbox"/> weather_master.json



# Data ingestion(Web api)

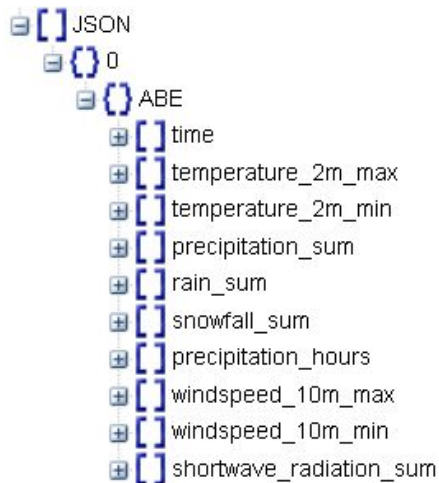
The initial schema of the raw json file is not suitable for joining with the other relatively structured datasets. Hence a databricks notebook in the pipeline processes the json file using spark and stores it back into the data lake as a single partition csv.

Name		Name
<input type="checkbox"/>		ingestion_output
<input type="checkbox"/>		weather_flattened
<input type="checkbox"/>		weather_master.json
<input type="checkbox"/>		[.]
<input type="checkbox"/>		_committed_7126474433227887174
<input type="checkbox"/>		_committed_790820328927012786
<input type="checkbox"/>		_committed_vacuum984594236269849187
<input type="checkbox"/>		_started_790820328927012786
<input type="checkbox"/>		part-00000-tid-790820328927012786-36f34a7e-1..

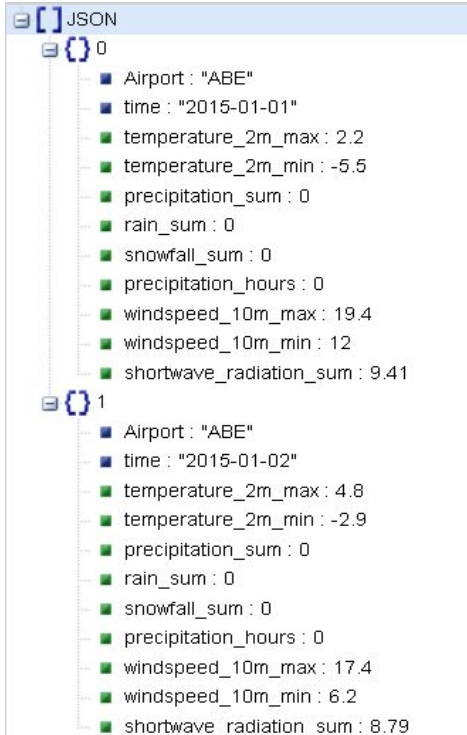


# Data ingestion(Web api)

## Schema before



## Schema after




# Data ingestion(Web api)

A dataset is created in azure data factory that points to the processed csv file present in data lake.



DelimitedText  
**api**

Connection	Schema	Parameters
Linked service *	AzureDataLakeStorage1  Test	
File path *	apitest / weather_flattened	
Compression type	None	
Column delimiter ⓘ	Comma (,)	
Row delimiter ⓘ	Default (\r,\n, or \r\n)	
Encoding ⓘ	Default(UTF-8)	
Escape character ⓘ	Backslash (\)	



# Data ingestion(SQL database)

The sql database contains transactional data on flight departures and arrivals.

flights (capserver/flights) | Query editor (preview) ☆ ...

SQL database

Search

Overview  
Activity log  
Tags  
Diagnose and solve problems  
Getting started  
Query editor (preview)  
Settings  
Compute + storage  
Connection strings  
Properties  
Locks  
Data management  
Replicas  
Sync to other databases  
Integrations  
Azure Synapse Link  
Stream analytics (preview)  
Add Azure Search  
Power Platform  
Power BI  
Power Apps  
Power Automate  
Security  
Auditing  
Ledger  
Data Discovery & Classification

dbo.mastertransaction

- YEAR (PK, int, not null)
- MONTH (PK, int, not null)
- DAY (PK, int, not null)
- DAY\_OF\_WEEK (int, null)
- AIRLINE (PK, nvarchar, not null)
- FLIGHT\_NUMBER (PK, int, not null)
- TAIL\_NUMBER (nvarchar, null)
- ORIGIN\_AIRPORT (PK, nvarchar, not null)
- DESTINATION\_AIRPORT (PK, nvarchar, not null)
- SCHEDULED\_DEPARTURE (int, null)
- DEPARTURE\_TIME (int, null)
- DEPARTURE\_DELAY (int, null)
- TAXI\_OUT (int, null)
- WHEELS\_OFF (int, null)
- SCHEDULED\_TIME (int, null)
- ELAPSED\_TIME (int, null)
- AIR\_TIME (int, null)
- DISTANCE (int, null)
- WHEELS\_ON (int, null)
- TAXI\_IN (int, null)
- SCHEDULED\_ARRIVAL (int, null)
- ARRIVAL\_TIME (int, null)
- ARRIVAL\_DELAY (int, null)
- DIVERTED (int, null)
- CANCELLED (int, null)
- CANCELLATION\_REASON (nvarchar, null)
- AIR\_SYSTEM\_DELAY (int, null)
- SECURITY\_DELAY (int, null)
- AIRLINE\_DELAY (int, null)
- LATE\_AIRCRAFT\_DELAY (int, null)
- WEATHER\_DELAY (int, null)

Query 1

Run Cancel query Save query Export data as Show only Editor

```
1 select count(*) from [dbo].[mastertransaction]
```

Results Messages

Search to filter items...

5819079



# Data ingestion(SQL database)

A dataset pointing to the SQL database is created in azure data factory



Azure SQL Database  
**mastertransaction**

Connection

Schema

Parameters

Linked service \*

 mastersql



 Test connection



Edit



New



Table

dbo

mastertransaction



Edit



# Data ingestion(nosql database)

In azure's Cosmos db the container contains non transactional details on airlines with name of airline and corresponding IATA codes.

Home > Recent > krishcosmos



krishcosmos | Data Explorer ☆ ...

Azure Cosmos DB account

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Cost Management
- Quick start
- Notifications
- Data Explorer

## Settings

- Features
- Replicate data globally
- Default consistency

NOSQL API

### DATA

#### id

Scale

#### airlines

Items

Settings

Stored Procedures

User Defined Functions

Triggers

#### airports

### NOTEBOOKS

Notebooks is currently not available. We are working on it.

Home

airlines - Items

Settings

Query 1

1. SELECT \* FROM c

Results

Query Stats

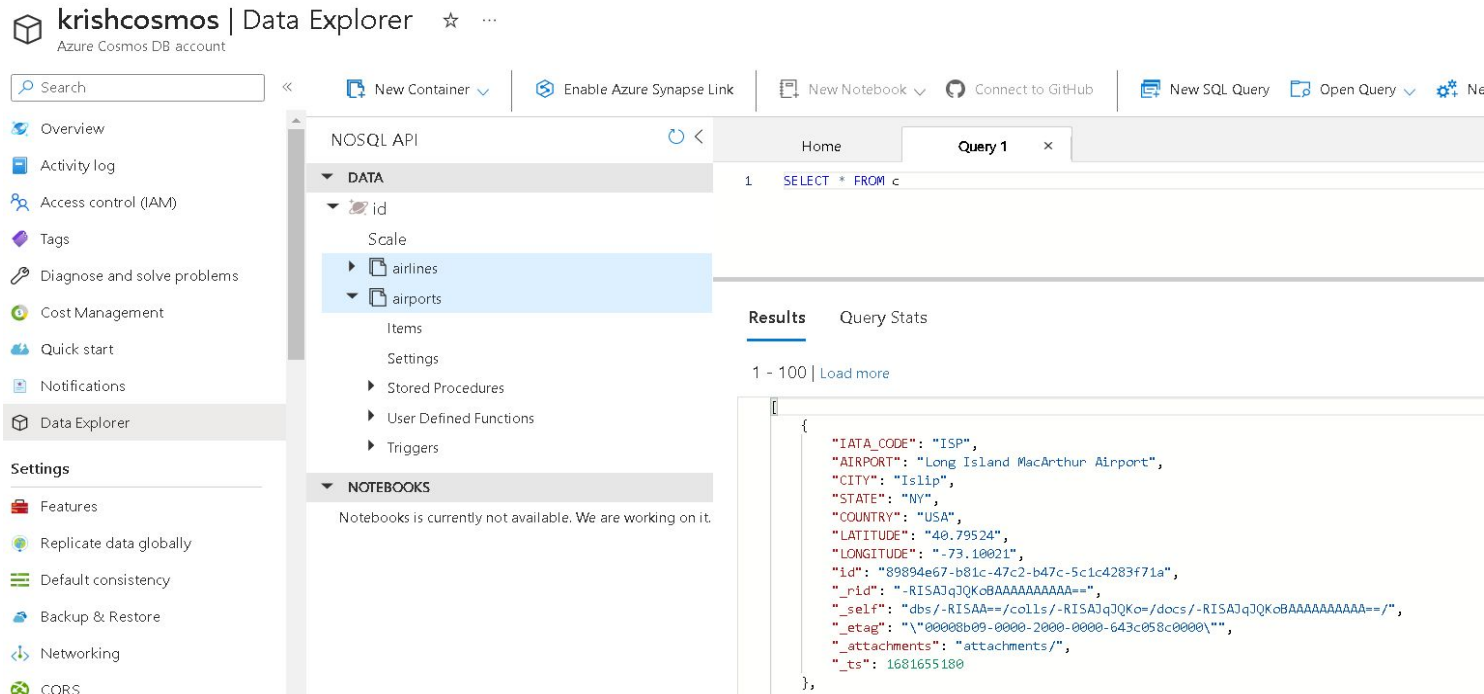
1 - 14

```
{
  "IATA_CODE": "VX",
  "AIRLINE": "Virgin America",
  "id": "79f080e5-ce6a-41d5-a75a-b20d7e2621c1",
  "_rid": "-RISAJ0sDskBAAAAAAAAA==",
  "_self": "dbs/-RISAA==/colls/-RISAJ0sDsk=/docs/-RISAJ0sDskBAAAAAAAAA==/",
  "_etag": "\"00007d09-0000-2000-0000-643c05540000\"",
  "_attachments": "attachments/",
  "_ts": 1681655124
}
```



# Data ingestion(nosql database)

In azure's Cosmos db the container contains non transactional details on airports geographical location and IATA\_codes.



The screenshot displays the Azure Cosmos DB Data Explorer interface for the 'krishcosmos' account. The left sidebar shows navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Cost Management, Quick start, Notifications, Data Explorer, and Settings. The main pane is divided into a left sidebar for the 'NOSQL API' and a right pane for the query results. The left sidebar shows the 'airports' collection under the 'DATA' section. The right pane shows a query result for the 'airports' collection, displaying a single document with details for 'Long Island MacArthur Airport'.

**krishcosmos | Data Explorer** ☆ ...  
Azure Cosmos DB account

Search << New Container Enable Azure Synapse Link New Notebook Connect to GitHub New SQL Query Open Query Ne

**NOSQL API**

**DATA**

- id
- Scale
- airlines
- airports
  - Items
  - Settings
  - Stored Procedures
  - User Defined Functions
  - Triggers

**NOTEBOOKS**

Notebooks is currently not available. We are working on it.

**Query 1**

```
1 SELECT * FROM c
```

**Results** Query Stats

1 - 100 | Load more

```
{
  "IATA_CODE": "ISP",
  "AIRPORT": "Long Island MacArthur Airport",
  "CITY": "Islip",
  "STATE": "NY",
  "COUNTRY": "USA",
  "LATITUDE": "40.79524",
  "LONGITUDE": "-73.10021",
  "id": "89894e67-b81c-47c2-b47c-5c1c4283f71a",
  "_rid": "-RISAJqJQKcBAAAAAAAAA==",
  "_self": "dbs/-RISAA=/colls/-RISAJqJQKcBAAAAAAAAA=/docs/-RISAJqJQKcBAAAAAAAAA=/",
  "_etag": "\"00000b09-0000-2000-0000-643c058c0000\"",
  "_attachments": "attachments/",
  "_ts": 1681655180
}
```





# Data ingestion(nosql database)

2 datasets pointing to the cosmos database container are created in azure data factory



Azure Cosmos DB for NoSQL  
**airlines\_cosmos**

Connection Schema Parameters

Linked service \*

 CosmosDbNoSql1



Test connection



Edit



New

[Learn more](#)

Container

airlines

☐ Edit



Azure Cosmos DB for NoSQL

**airport\_cosmos**

Connection Schema Parameters

Import schema

Clear

Column name

Type

IATA\_CODE

abc string

AIRPORT

abc string

CITY

abc string

STATE

abc string

COUNTRY

abc string

LATITUDE

abc string

LONGITUDE

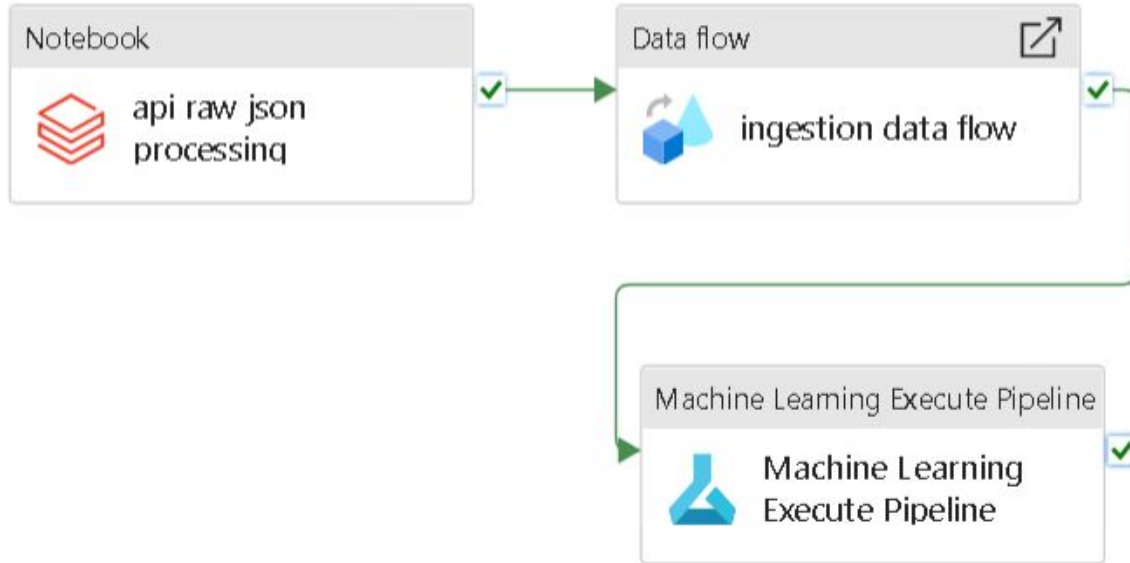
abc string

# Data ingestion flow pipeline in ADF



# Master pipeline overview

The master pipeline consists of 3 different components as shown below.

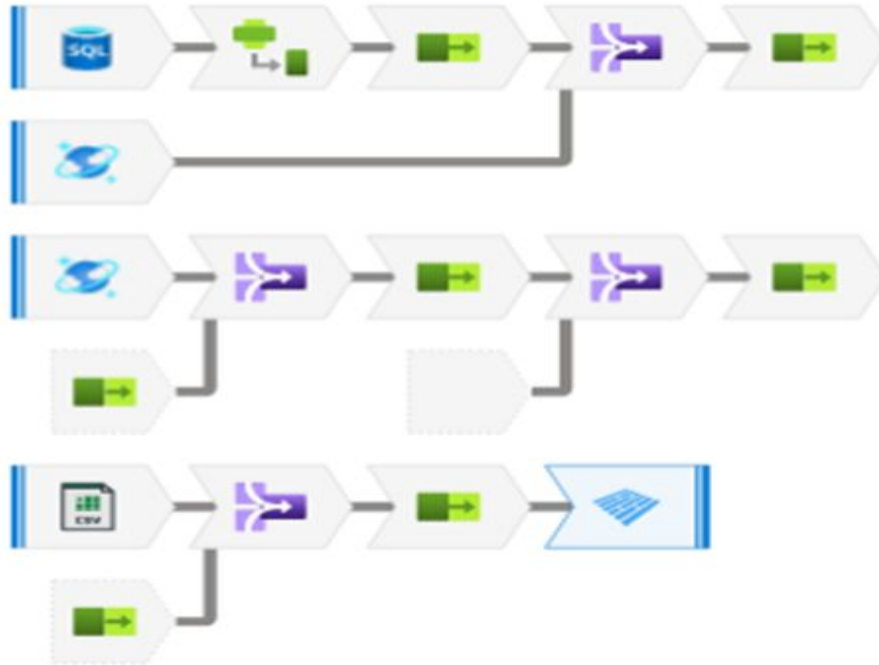


# Api data processing

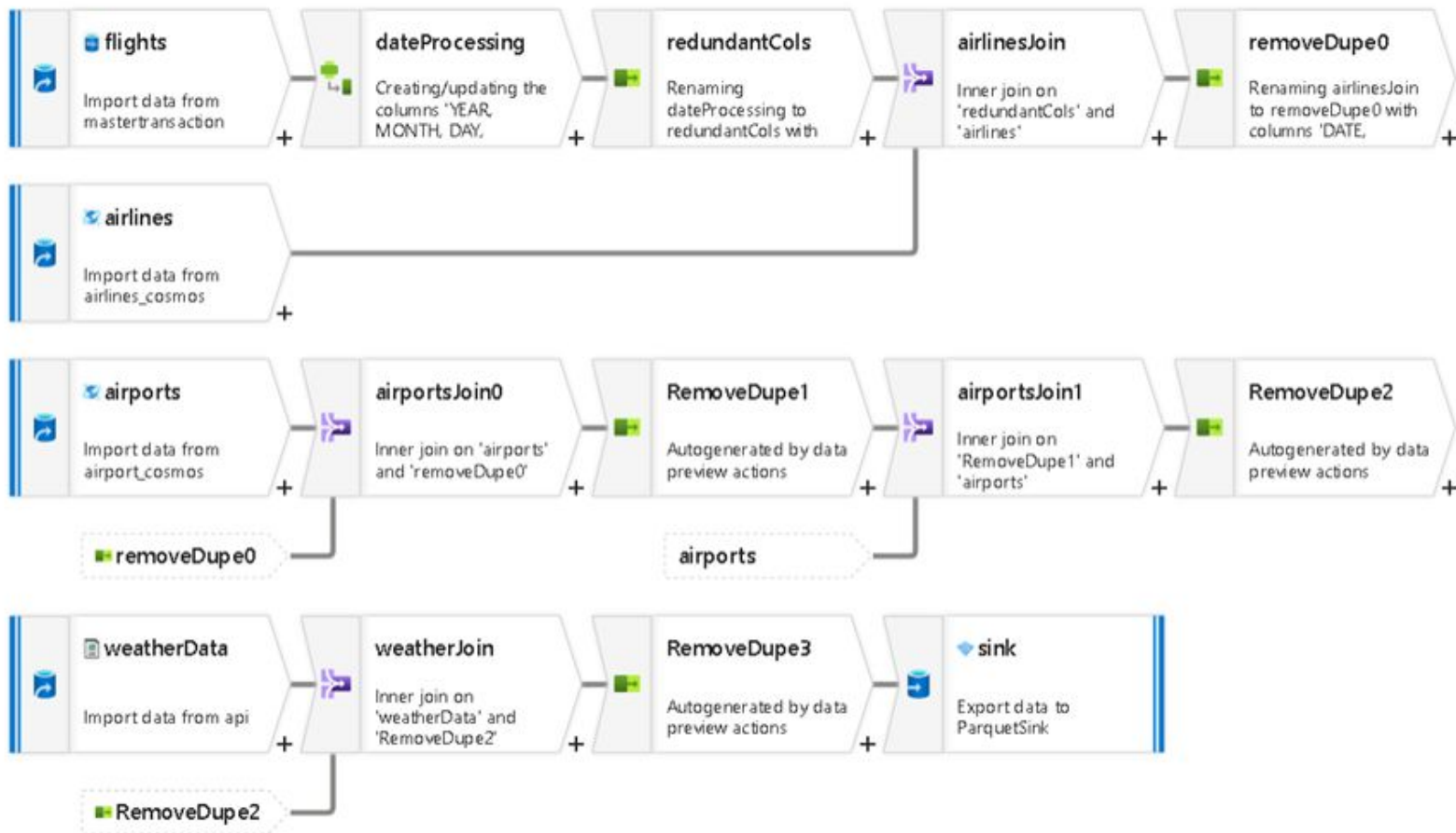
```
1  import pandas as pd
2  weather_df = pd.DataFrame()
3  for i in range(len(weather_data)):
4      for key in weather_data[i].keys():
5          weather_df = pd.concat([weather_df, pd.DataFrame(weather_data[i][key])])
6
7  airport_df = pd.DataFrame()
8  for i in range(len(weather_data)):
9      for key in weather_data[i].keys():
10         d = {'Airport': [key]*365}
11         airport_df = pd.concat([airport_df, pd.DataFrame(d)])
12
13  weather_df['Airport'] = airport_df['Airport']
```

# Data flow pipeline overview

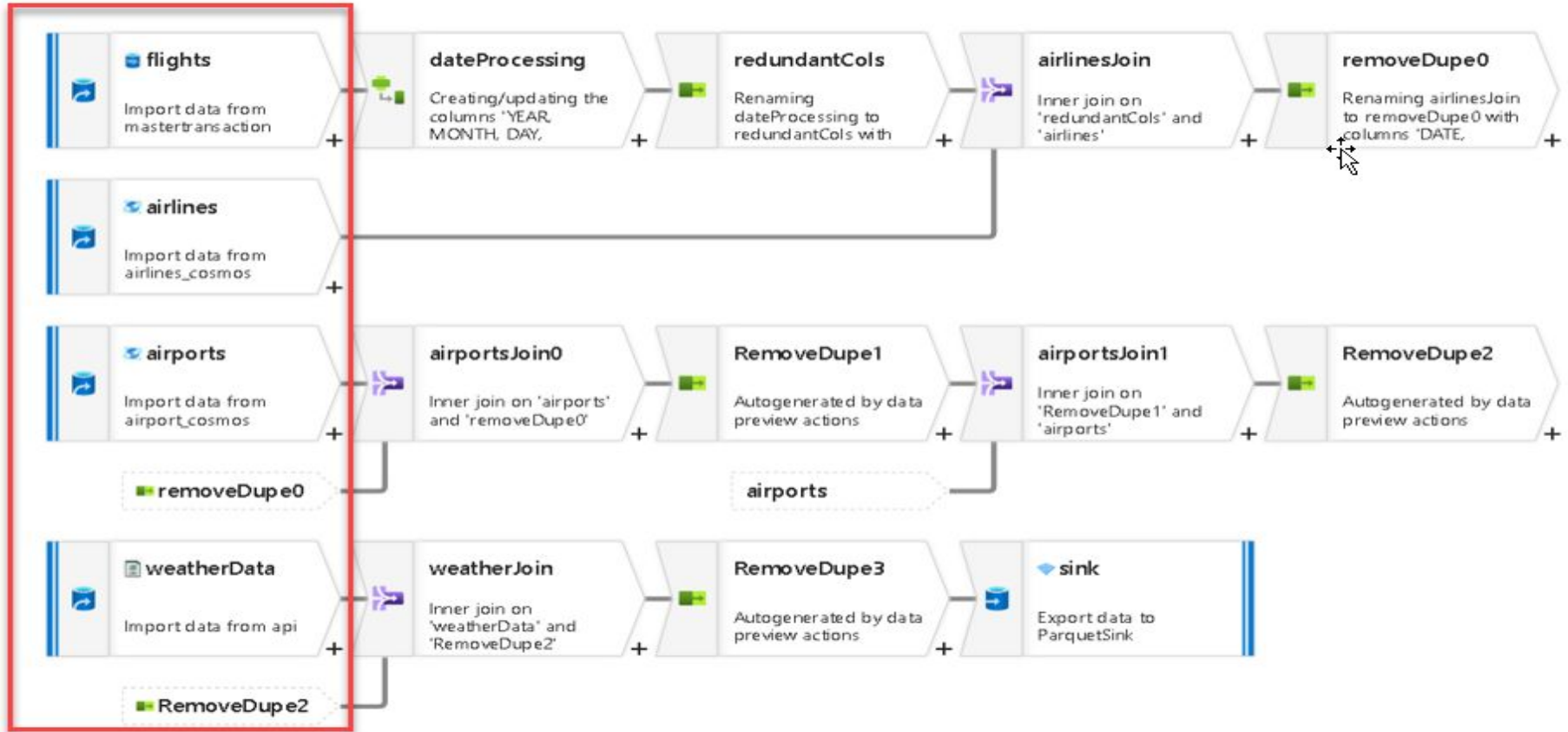
The data flow pipeline purpose is to join all the 4 input datasets from different sources, clean them and store the combined dataset into a sink in the data lake.



# Data flow pipeline overview



# Data sources configuration



# Data sources configuration

Source settings

Source options

Projection

Optimize

Inspect

Data preview

Output stream name \*

flights

Description

Import data from mastertransaction

Source type \*

Dataset

Inline

Dataset \*

mastertransaction

Options

☒ Allow schema drift ⓘ

☐ Infer drifted column types ⓘ

☐ Validate schema ⓘ

Sampling \* ⓘ

☐ Enable

☒ Disable

Source settings

Source options

Projection

Optimize

Inspect

Data preview

Partition option \*

☒ Use current partitioning

☐ Single partition

☐ Set partitioning



# Data cleansing configuration



# Data cleansing configuration

## Dataflow expression builder

dateProcessing

### Derived Columns

+ Create new ▼

abc DATE

### Column name \*

DATE

### Expression

```
toString(toDate(concat(toString(YEAR), '-', toString(MONTH), '-', toString(DAY))))
```

### Select settings

Optimize

Inspect

Data preview

### Output stream name \*

redundantCols

[Learn more](#) 

### Description

AIRLINE\_DELAY,  
LATE\_AIRCRAFT\_DELAY,  
WEATHER\_DELAY"

 Reset

### Incoming stream \*

dateProcessing ▼

### Options

☒ Skip duplicate input columns ⓘ

☒ Skip duplicate output columns ⓘ

### Input columns \*

☐ Auto mapping ⓘ

 Reset

+ Add mapping

 Delete

29 mappings: 3 column(s) from



dateProcessing's column



Name as



abc DATE ▼



DATE



123 DAY\_OF\_WEEK ▼



DAY\_OF\_WEEK



# Data Joining configuration



# Data Joining configuration

Join settings

Optimize

Inspect

Data preview

Output stream name \*

weatherJoin

[Learn more](#) 

Description

Inner join on 'weatherData' and  
'RemoveDupe2'

 Reset

Left stream \*

weatherData

Right stream \*

RemoveDupe2

Join type \*



Full outer



Inner



Left outer



Right outer



Custom (cross)

Use fuzzy matching 

☐

Join conditions \*

Left: weatherData's column

Right: RemoveDupe2's column

abc Airport

==

abc ORIGIN\_AIRPORT\_CODE



abc time

==

abc DEPARTURE\_DATE



# Data Sink configuration



# Data Sink configuration

Sink Settings Errors Mapping Optimize Inspect Data preview

Output stream name \*  [Learn more](#)

Description  [Reset](#)

Incoming stream \*

Sink type \* 

Dataset

Inline

Cache

Dataset \*  [Test connection](#)

Options ☒ Allow schema drift [?](#)  
☐ Validate schema [?](#)

Sink Settings Errors Mapping Optimize Inspect Data preview

Clear the folder ☒

File name option \*

Umask ⓘ

Owner	<input type="checkbox"/> R	<input type="checkbox"/> W	<input type="checkbox"/> X
Group	<input type="checkbox"/> R	<input checked="" type="checkbox"/> W	<input type="checkbox"/> X
Others	<input type="checkbox"/> R	<input checked="" type="checkbox"/> W	<input type="checkbox"/> X

Sink Settings Errors Mapping **Optimize** Inspect Data preview

Partition option \* ☒ Use current partitioning ☐ Single partition ☐ Set partitioning

# Data Sink configuration



Parquet

**ParquetSink**

Connection

Schema

Parameters

Linked service \*

 AzureDataLakeStorage1

 Test

File path \*

apitest

/

ingestion\_output

Compression type

snappy



# Exploratory Data Analysis (EDA)





# METHODOLOGY - EDA

## Exploratory Data Analysis

- This includes exploring the dataset to understand its characteristics and relationships between variables.
- It helps in identifying patterns, trends, and potential outliers that could affect the accuracy of the prediction model.
- Presentation of the analysis in a graphical or pictorial manner helps to visualize the features better, and make inferences based on correlations found in the data.



# Missing Values in the Dataset

```
1 missing_df = df.isnull().sum(axis=0).reset_index()
2 missing_df.columns = ['variable', 'missing values']
3 missing_df['filling factor (%)']=(df.shape[0]-missing_df['missing values']/df.shape[0]*100
4 missing_df.sort_values('filling factor (%)').reset_index(drop = True)
```

	variable	missing values	filling factor (%)
0	LATE_AIRCRAFT_DELAY	4325596	18.818597
1	WEATHER_DELAY	4325596	18.818597
2	AIRLINE_DELAY	4325596	18.818597
3	AIR_SYSTEM_DELAY	4325596	18.818597
4	SECURITY_DELAY	4325596	18.818597
5	ARRIVAL_DELAY	101740	98.090576
6	DEPARTURE_DELAY	83781	98.427625
7	SCHEDULED_TIME	6	99.999887
8	AIRLINE	0	100.000000
9	ORIGIN_windspeed_max	0	100.000000
10	ORIGIN_precipitation_hours	0	100.000000
11	ORIGIN_snowfall_sum	0	100.000000
12	ORIGIN_rain_sum	0	100.000000
13	ORIGIN_precipitation_sum	0	100.000000
14	ORIGIN_temperature_min	0	100.000000

Top 5 columns have 20% filling rate, which is quite low

They come into picture only when there is some delay, so these do not have missing data but rather context specific data.

Need to remove rows with null values of Departure and Arrival Delays as these are variables to be predicted

AIRLINE	FLIGHT_COUNT	ON_TIME_COUNT	ON_TIME_PERCENT
Delta Air Lines Inc.	792941	558374	70.42
Alaska Airlines Inc.	157025	104574	66.60
American Airlines Inc.	636554	406985	63.94
United Air Lines Inc.	462086	287589	62.24
American Eagle Airlines Inc.	257130	158363	61.59
Southwest Airlines Co.	1136750	697063	61.32
Atlantic Southeast Airlines	508958	311215	61.15
Skywest Airlines Inc.	528328	322557	61.05
JetBlue Airways	240304	146455	60.95
US Airways Inc.	194223	117938	60.72
Virgin America	55813	33569	60.15
Hawaiian Airlines Inc.	69815	41881	59.99
Frontier Airlines Inc.	81861	43238	52.82
Spirit Air Lines	104781	52277	49.89

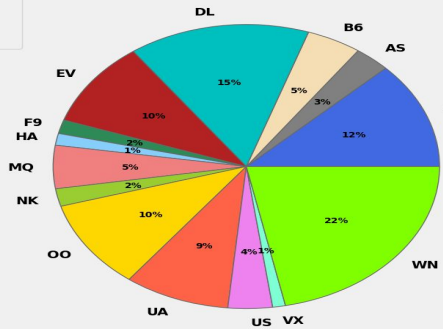
## Ratio of flights ON TIME by each airline

### Observations -

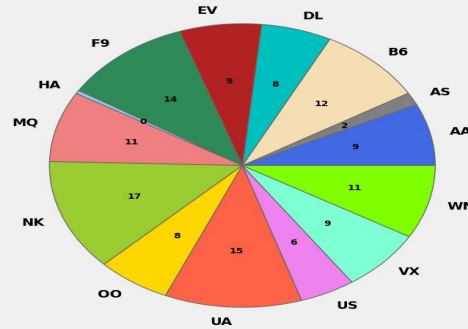
- Delta airlines seem to have the best record in terms of being on time
- Alaska Airlines has flown quite less compared to others, so their record is of a lower base.
- Southwest has flown the most, and has a decent record
- Spirit Airlines has a less than 50% record of completing the flight on time

AA: American Airlines Inc.  
 AS: Alaska Airlines Inc.  
 B6: JetBlue Airways  
 DL: Delta Air Lines Inc.  
 EV: Atlantic Southeast Airlines  
 F9: Frontier Airlines Inc.  
 HA: Hawaiian Airlines Inc.  
 MQ: American Eagle Airlines Inc.  
 NK: Spirit Air Lines  
 OO: Skywest Airlines Inc.  
 UA: United Air Lines Inc.  
 US: US Airways Inc.  
 VX: Virgin America  
 WN: Southwest Airlines Co.

% of flights per company



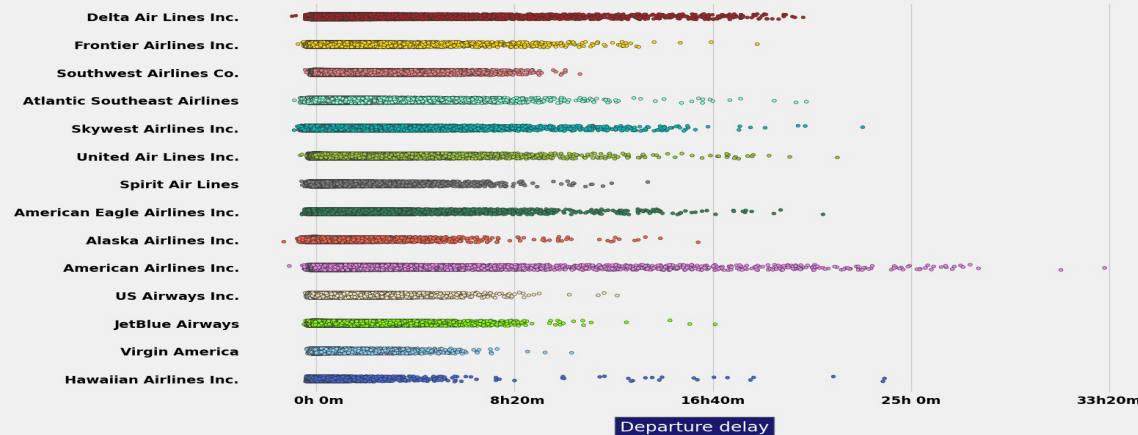
Mean delay at origin



Southwest Airlines accounts for ~ 20% of the flights which is equal to flights chartered by the 7 tiniest airlines.

~ 11 ± 7 minutes would correctly represent all mean delays.

Occasionally, we can face really large delays that can reach a few tens of hours !



## Duration of delays per each airline

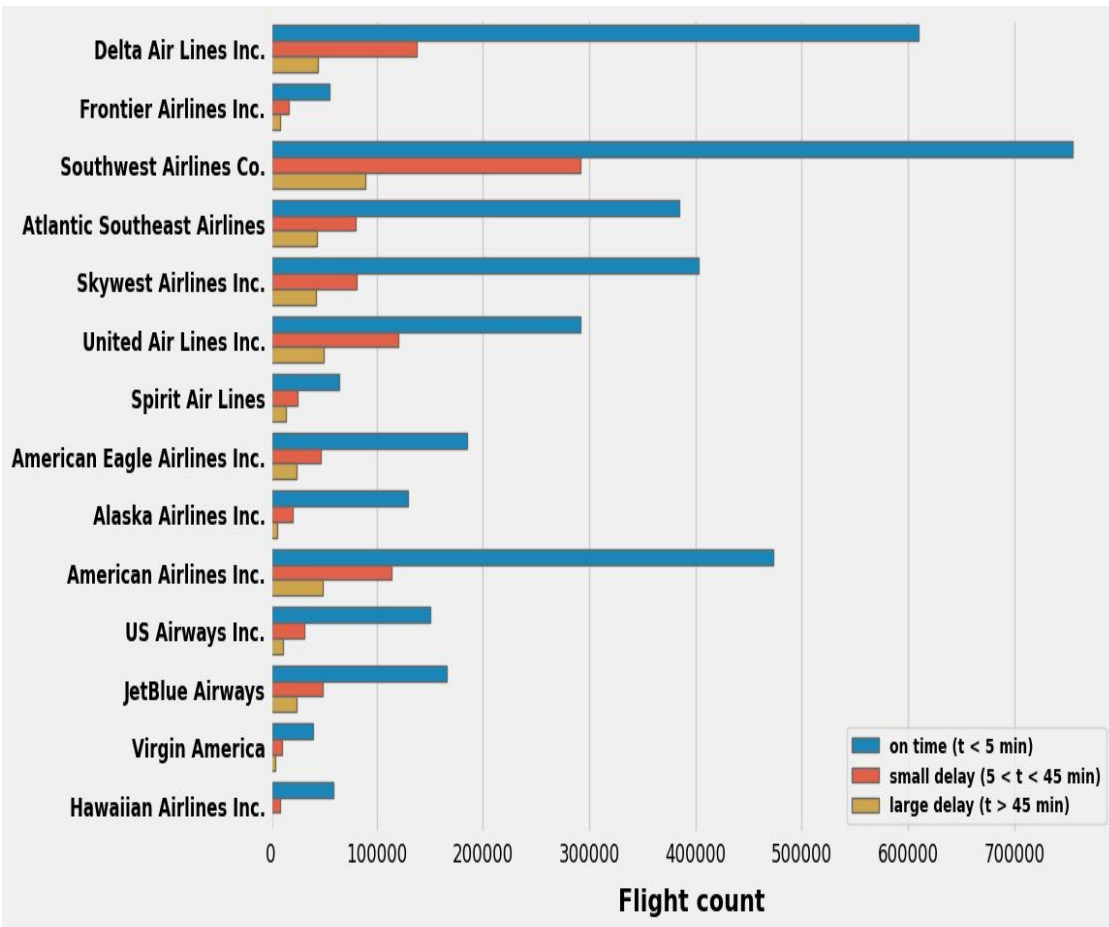
This figure gives a count of the delays as -

1. Less than 5 minutes,
2.  $5 < t < 45$  min
3. Greater than 45 minutes.

Delays greater than 45 minutes only account for a few percents.

In the case of SkyWest Airlines, the delays  $> 45$  minutes are only lower by ~30% with respect to delays in the range  $5 < t < 45$  min.

Things are better for SouthWest Airlines since delays  $> 45$  minutes are 3 times less frequent than delays in the range  $5 < t < 45$  min



## Analyzing total delays by each airport

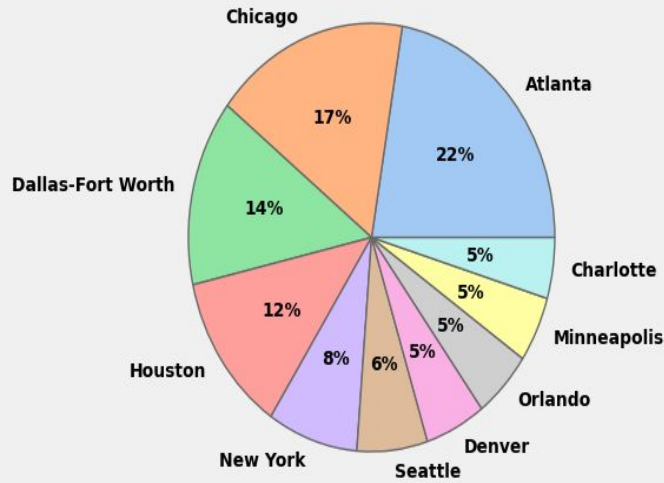
	SECURITY_DELAY	AIR_SYSTEM_DELAY	AIRLINE_DELAY	WEATHER_DELAY	LATE_AIRCRAFT_DELAY	TOTAL_DELAYS
ORIGIN_AIRPORT_NAME						
Chicago O'Hare International Airport	2697.0	947333.0	1260840.0	422593.0	1543993.0	4177456.0
Hartsfield-Jackson Atlanta International Airport	1334.0	634336.0	1207193.0	313491.0	1052413.0	3208767.0
Dallas/Fort Worth International Airport	5704.0	532221.0	1068866.0	256145.0	1042585.0	2905521.0
Denver International Airport	894.0	541548.0	757379.0	92635.0	1026488.0	2418944.0
Los Angeles International Airport	3215.0	436875.0	706498.0	32181.0	999620.0	2178389.0
George Bush Intercontinental Airport	2404.0	439734.0	614859.0	85568.0	643952.0	1786517.0
San Francisco International Airport	1783.0	287977.0	557758.0	26520.0	860377.0	1734415.0
McCarran International Airport	1054.0	303069.0	493739.0	31389.0	686130.0	1515381.0
LaGuardia Airport (Marine Air Terminal)	554.0	365359.0	354413.0	57349.0	703435.0	1481110.0
Orlando International Airport	1900.0	343101.0	410131.0	102521.0	586584.0	1444237.0
Phoenix Sky Harbor International Airport	5991.0	284333.0	515965.0	32599.0	530829.0	1369717.0
Newark Liberty International Airport	1699.0	278599.0	471903.0	57812.0	521971.0	1331984.0
John F. Kennedy International Airport (New York International Airport)	5882.0	335556.0	449859.0	92963.0	399769.0	1284029.0
Gen. Edward Lawrence Logan International Airport	2087.0	418765.0	348986.0	71672.0	419172.0	1260682.0
Detroit Metropolitan Airport	1170.0	299951.0	473649.0	43425.0	349062.0	1167257.0
Baltimore-Washington International Airport	2961.0	202666.0	410531.0	62754.0	412634.0	1091546.0
Charlotte Douglas International Airport	5435.0	335570.0	385763.0	42952.0	299302.0	1069022.0
Minneapolis-Saint Paul International Airport	641.0	288882.0	359620.0	35034.0	322811.0	1006988.0
Chicago Midway International Airport	428.0	150837.0	316097.0	57128.0	402415.0	926905.0
Miami International Airport	2574.0	215381.0	335073.0	53035.0	304223.0	910286.0

Chicago's airport clearly has the most amount of delays followed by airport of Atlanta and Dallas.

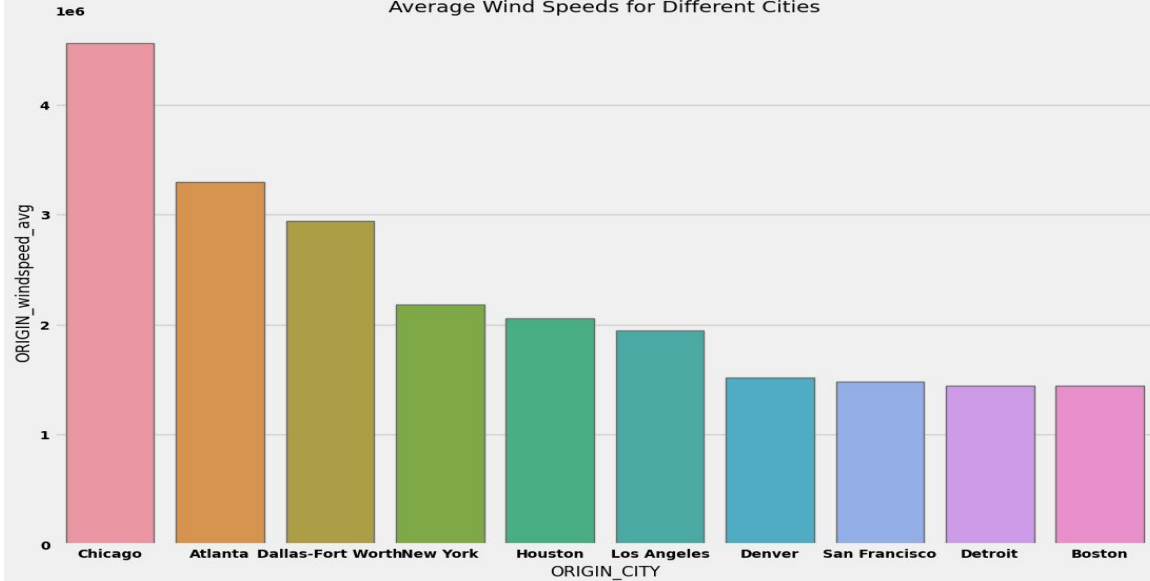
We will analyze further probable reasons as to why these airports face these delays

## Weather Conditions for different cities

Precipitation for different cities



Average Wind Speeds for Different Cities



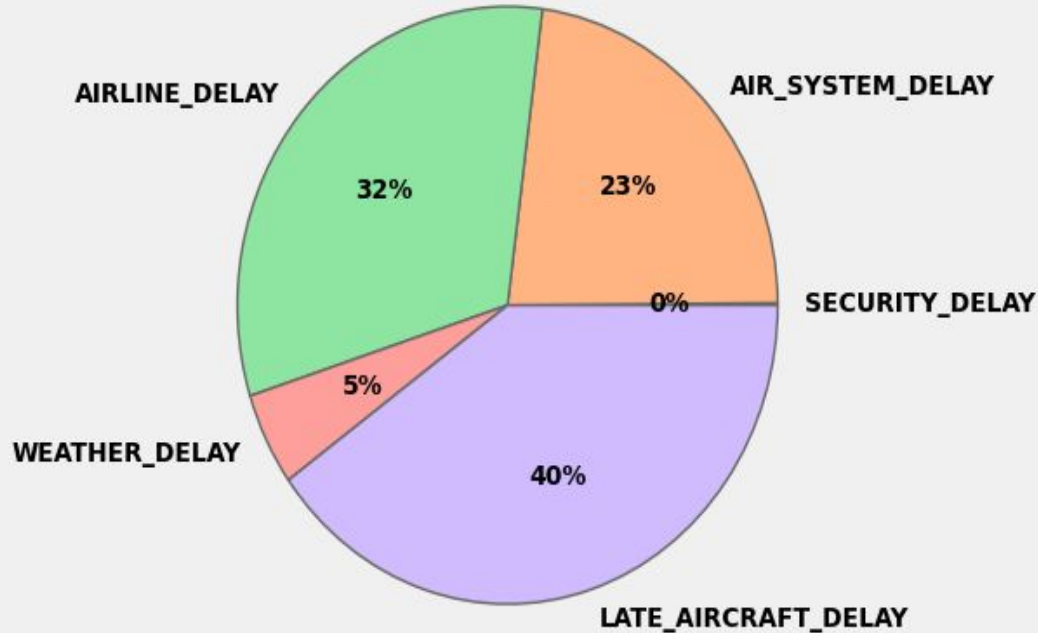
Observation -

Chicago, Atlanta and Dallas being on top of precipitation charts and wind charts and their airports on top of delay charts does not seem to be a coincidence.

The delays are probably due to the weather conditions of these cities.

## Comparing different delays

Ratio of the delays



1. LATE\_AIRCRAFT\_DELAY -> 23748821.0
2. AIRLINE\_DELAY -> 18955762.0
3. AIR\_SYSTEM\_DELAY -> 13523157.0
4. WEATHER\_DELAY -> 2987643.0
5. SECURITY\_DELAY -> 77945.0



# METHODOLOGY - DATA TRANSFORMATION

## Data Cleaning

- After completing the EDA, it is necessary to perform data cleaning to remove or impute missing values, remove duplicates, and correct data inconsistencies.
- This step ensures that the data is clean and ready for modelling.

## Data Manipulation

- The next step is to manipulate the data to create new variables or features that could be useful in predicting flight delays.
- This step could include feature engineering, such as extracting the week or hour of the day from the flight schedule data.



# METHODOLOGY

## Modelling

- The Machine Learning Model used for this problem statement is:
  - **Decision Tree Regressor**
- The decision tree regressor is a popular and effective algorithm for predicting airline delays because it can capture nonlinear relationships between features and the target variable, and it can handle categorical and numerical features.
- **Tool used** - Azure Machine Learning Studio
- Azure ML Studio provides us with a No Code way of preparing our dataset and training our model.



# ML Studio Pipeline



# Create & Configure Compute Cluster


Microsoft Azure Machine Learning Studio

Default Directory > krishML > Compute

Compute

The "Kubernetes clusters" tab is now where you can access previous versions of "inference clusters" (also known as "AKS clusters") and "attached Kubernetes" compute types along with Kubernetes clusters.

Compute instances   **Compute clusters**   Kubernetes clusters   Attached computes



Scale your compute cluster from a single node to a multi node workload

Create a single or multi node compute cluster for your training, batch inferencing or reinforcement learning workloads. [Learn more](#)

**+ New**

[View Azure Machine Learning tutorials](#) [View available quota](#)

Left sidebar menu: New, Home, Authoring, Notebooks, Automated ML, Designer, Assets, Data, Jobs, Components, Pipelines, Environments, Models, Endpoints, Manage, **Compute**, Linked Services, Data Labeling.

Create compute cluster

**Virtual Machine**

Select the virtual machine size you would like to use for your compute cluster.

Location \*

Central India

Virtual machine tier

☒ Dedicated ☐ Low priority

Virtual machine type

☒ CPU ☐ GPU

Virtual machine size

☒ Select from recommended options ☐ Select from all options

Name ↑	Category	Workload types	Available quota	Cost
<input type="radio"/> Standard_DS11_v2 2 cores, 14GB RAM, 28GB storage	Memory optimized	Development on Notebooks (or other IDE) and light weight testing	6 cores	\$0.19/hr
<input type="radio"/> Standard_DS3_v2 4 cores, 14GB RAM, 28GB storage	General purpose	Classical ML model training on small datasets	6 cores	\$0.34/hr
<input checked="" type="radio"/> Standard_DS12_v2 4 cores, 28GB RAM, 28GB storage	Memory optimized	Data manipulation and training on medium-sized datasets (1-10GB)	6 cores	\$0.38/hr
<input type="radio"/> Standard_F4s_v2	Compute-optimized	Data manipulation and training on large datasets (> 10 GB)	16 cores	\$0.17/hr

Back Next

To execute the pipeline a compute is needed to be added and configured

# Create Dataset in Azure ML Studio

The screenshot displays the Microsoft Azure Machine Learning Studio interface. On the left, the 'Data' tab is selected in the sidebar. The main pane shows the 'Datastores' section, which lists existing datastores. A red box highlights the '+ Create' button. To the right, the 'Create datastore' dialog is open, with several fields highlighted by red boxes: 'Datastore name' (set to 'capstoneflightdelay'), 'Datastore type' (set to 'Azure Blob Storage'), 'Storage account' (set to 'capstonekrish'), and 'Blob container' (set to 'apitest'). The 'Authentication type' is set to 'Account key', and the 'Account key' field is empty. The 'Save credentials with the datastore for data access' checkbox is checked. The 'Create' button is at the bottom right of the dialog.

Microsoft Azure Machine Learning Studio

Default Directory > krishML > Data

Data assets Datastores Dataset monitors view

Datastores securely connect to a storage service on Azure by storing connection information. With datastores, you no longer need to provide credential information in your code.

+ Create Refresh Unregister Set as default datastore Edit columns Reset view

Search

Showing 1-6 of 6 datastores

Name	Type	Storage name
azureml	Azure Blob Storage	krishml7643601414
flightdelaydata	Azure Blob Storage	capstonekrish
workspacefilestore	Azure file share	krishml7643601414
workspaceworkingdirectory	Azure file share	krishml7643601414
workspaceartifactstore	Azure Blob Storage	krishml7643601414
workspaceblobstore (Default)	Azure Blob Storage	krishml7643601414

Create datastore

Datastore name \* capstoneflightdelay

Datastore type \* Azure Blob Storage

Account selection method

☒ From Azure subscription

☐ Enter manually

Subscription ID \* Azure for Students (950880b0-6637-4400-a2be-5997a027f42f)

Storage account \* capstonekrish (capstone)

Blob container \* apitest

☒ Save credentials with the datastore for data access

Authentication type \* Account key

Account key \*

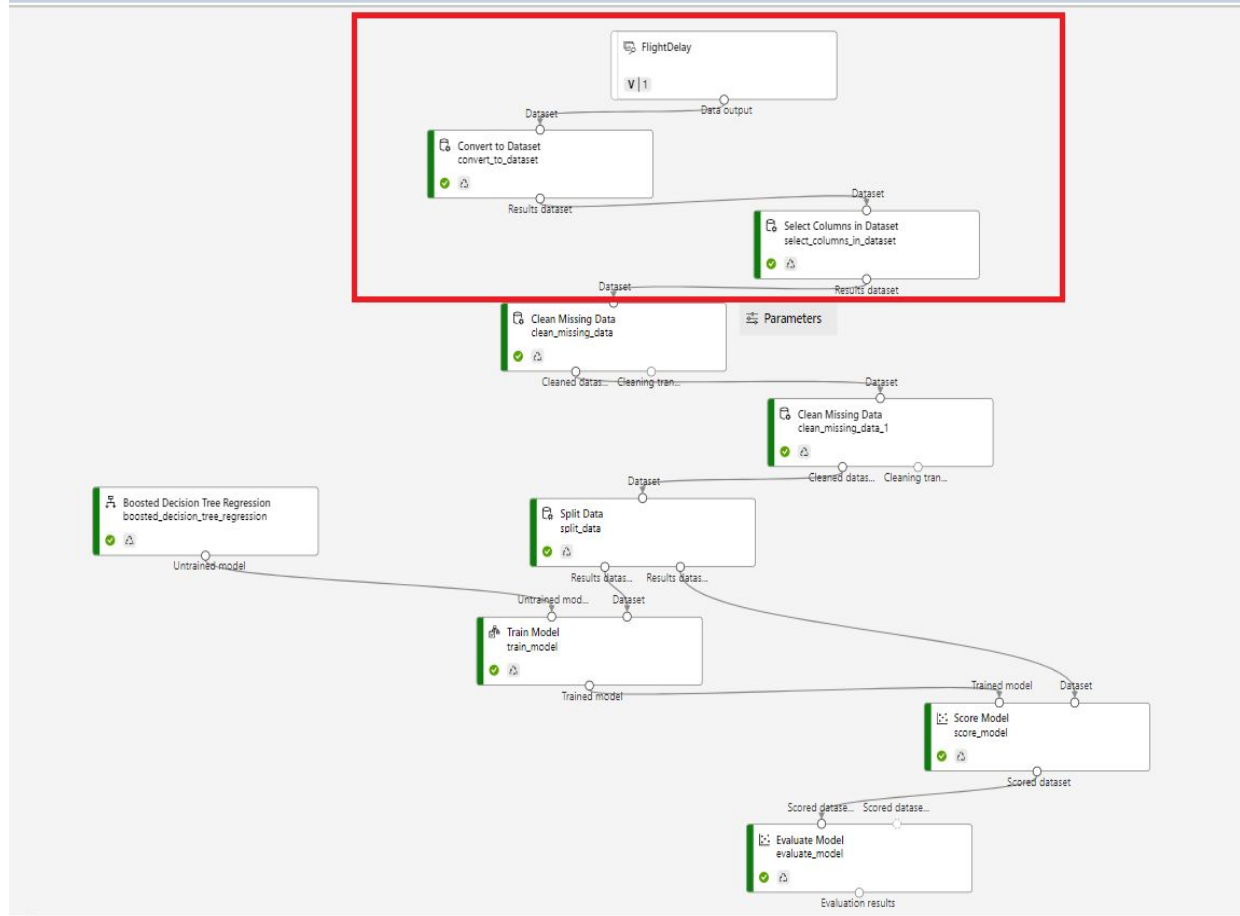
☐ Use workspace managed identity for data preview and profiling in Azure Machine Learning studio

Note: Azure Machine Learning service does not validate whether the underlying data source is accessible.

Create Cancel

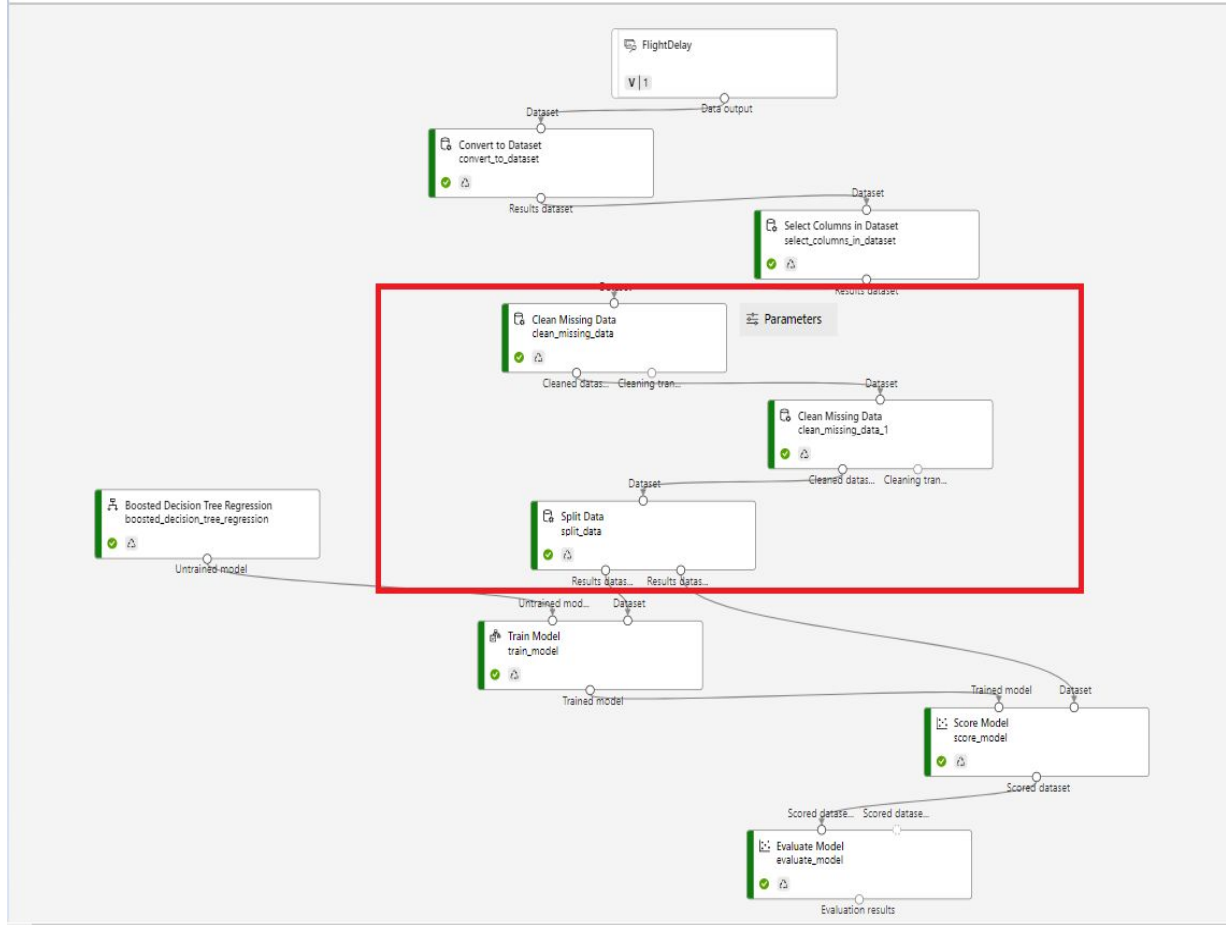
Go to Data > Datastore > + Create > Provide Datastore Type > Provide Storage Account > Provide the Account Key > Create

# Design the ML Flow Pipeline



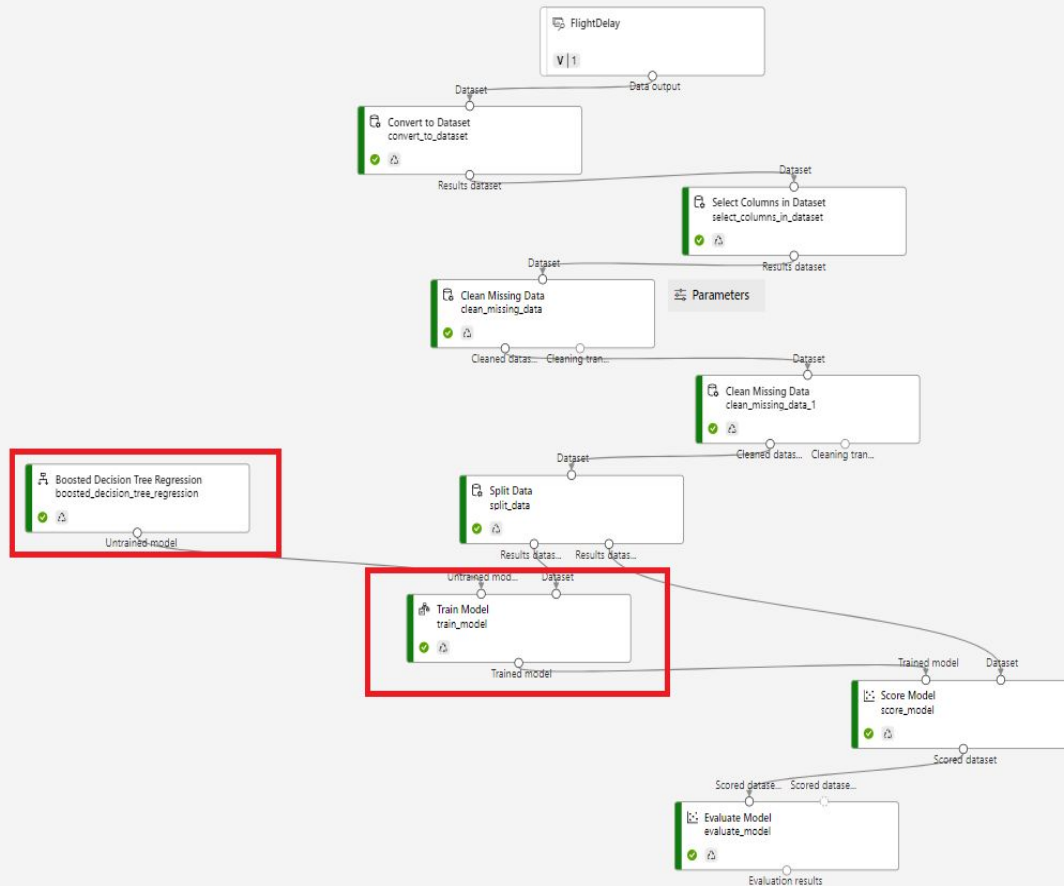
- Starting with flight delay dataset created in the previous slide
- Component added for Converting it to Dataset
- Select the columns of importance for the model

# Design the ML Flow Pipeline contd..



- First of two clean missing data components to remove null values for the dependent variable i.e Departure Delay
- Second of two clean data component to replace the null values of 5 types of delay with 0
- Split data component added to split it into train and test data

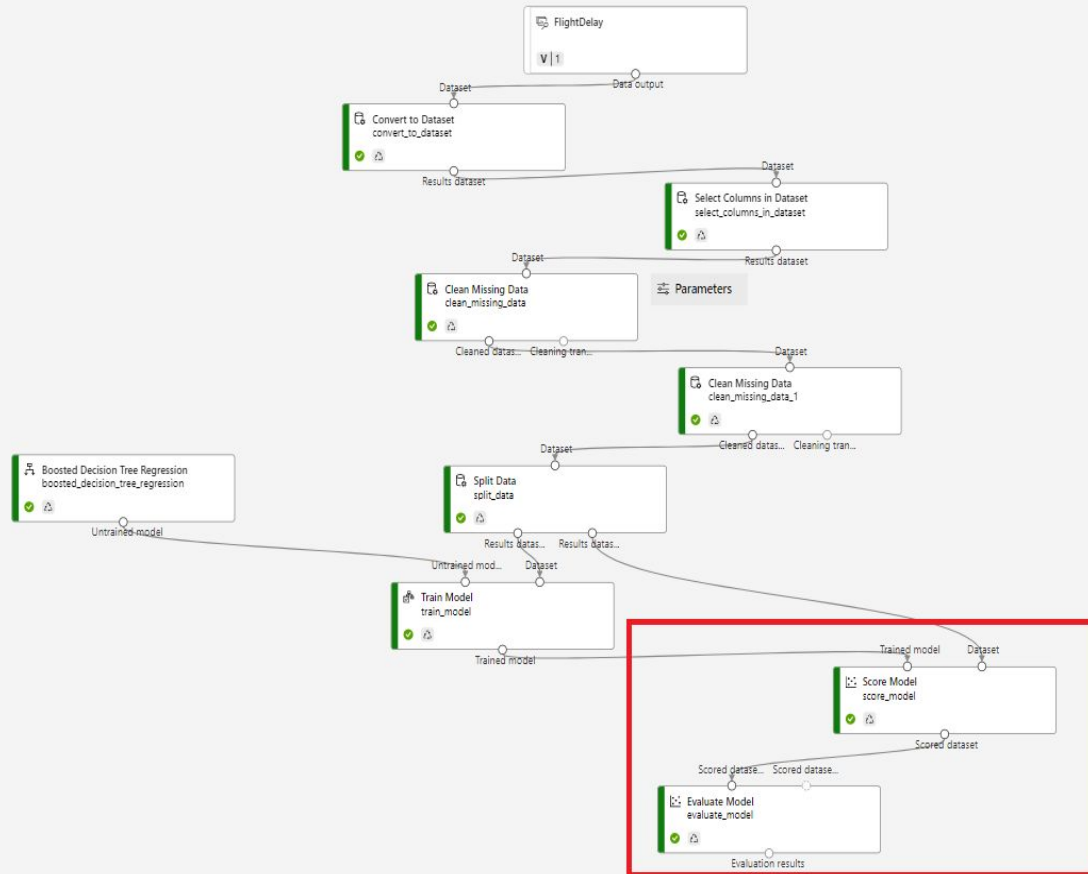
# Design the ML Flow Pipeline contd..



- Model component added. Boosted Decision Tree Regression Model
- Train Model component taking inputs from untrained model above and the train part of the split data component



# Design the ML Flow Pipeline contd..



- Score model component taking inputs from Train model and Test dataset from Split data
- Score Model predicts the output on the test dataset
- Next Evaluate Model to check the model metrics and performance on the test dataset

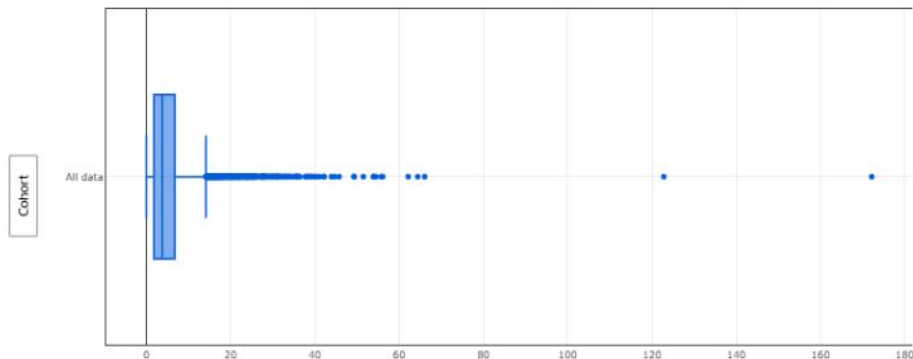
## Feature Importance

# Train Model Output

ORIGIN\_temperature\_max  
DESTINATION\_AIRPORT\_CODE  
ORIGIN\_AIRPORT\_CODE  
AIRLINE\_IATA\_CODE  
ORIGIN\_STATE  
ORIGIN\_CITY  
WEATHER\_DELAY  
LATE\_AIRCRAFT\_DELAY  
AIRLINE\_DELAY  
SECURITY\_DELAY  
DESTINATION\_CITY  
AIR\_SYSTEM\_DELAY  
SCHEDULED\_TIME  
ORIGIN\_shortwave\_radiation\_sum  
ORIGIN\_windspeed\_min  
ORIGIN\_windspeed\_max  
ORIGIN\_precipitation\_hours  
ORIGIN\_snowfall\_sum  
ORIGIN\_rain\_sum  
ORIGIN\_precipitation\_sum  
ORIGIN\_temperature\_min  
DISTANCE  
DESTINATION\_STATE

Model performance   Dataset explorer   Aggregate feature importance   Individual feature importance

Evaluate the performance of your model by exploring the distribution of your prediction values and the values of your model performance metrics. You can further investigate your model by looking at a comparative analysis of its performance across different cohorts or subgroups of your dataset. Select filters along y-value and x-value to cut across different dimensions.



Sample size: 5,000  
Mean absolute error: 5.444  
Mean squared error: 74.643  
R<sup>2</sup>: 0.948  
Mean prediction: 9.466

- Feature importance in descending order from the trained model (on the left)
- Model performance on trained dataset (on the right)

# PERFORMANCE

1. **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values.
2. **Root Mean Squared Error (RMSE):** Measures the square root of MSE, giving a measure of the average absolute error.
3. **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual values.
4. **R-squared (R<sup>2</sup>):** Measures the proportion of variance in the target variable that can be explained by the model.

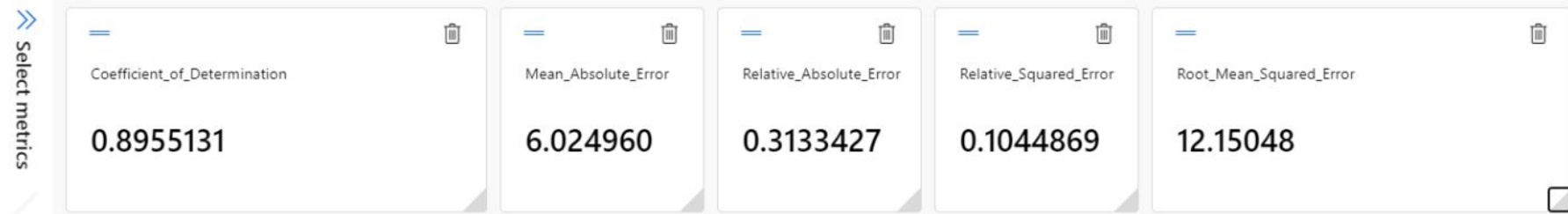


# Test Data Metrics

## Evaluate Model

Overview Parameters Outputs + logs **Metrics** Child jobs Images Code Explanations (preview) Fairness (preview) Monitoring

Refresh | Create custom chart | View as... | Current view: Local | Edit view



Here we have the Test Data metrics from Evaluate Model Component

Coefficient of Determination ( $R^2$ )  $\sim 0.89$  reflects a good model output

# Create Inference Cluster & Pipeline

**Microsoft Azure Machine Learning Studio**

Default Directory > krishML > Compute

**Compute**

The "Kubernetes clusters" tab is now where you can access previous versions of "inference clusters" (also known as "AKS clusters") and "attached Kubernetes" compute types along with any previously created clusters.

Compute instances    Compute clusters    **Kubernetes clusters**    Attached computes

**Train or deploy models with your self-managed Kubernetes clusters anywhere**

Bring your own Kubernetes clusters in any infrastructure across multi-cloud, on-premises, and the edge to use as a compute target with Azure Machine Learning workspace. [Learn more](#)

**+ New**

- Kubernetes
- AksCompute**

**Microsoft Azure Machine Learning Studio**

Default Directory > krishML > Compute

**Create AksCompute**

This wizard creates or attaches Azure Kubernetes Services cluster for AzureML API v1. [Learn more](#) to attach Azure Kubernetes Service cluster using the recommended approach for v2.

**Virtual Machine**

Select the virtual machine size you would like to use for your inference cluster.

**Kubernetes Service**

☒ Create new ☐ Use existing

**Location \***

Central India

Showing 379 VM sizes | Current selection: Standard\_A2\_v2

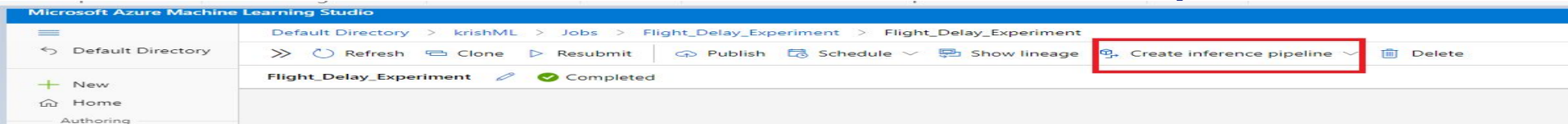
Name	Category	Available quota
<input checked="" type="radio"/> Standard_A2_v2 2 cores, 4GB RAM, 20GB storage	General purpose	4 cores
<input type="radio"/> Standard_A2m_v2 2 cores, 16GB RAM, 32GB storage	General purpose	4 cores
<input type="radio"/> Standard_A4_v2 4 cores, 8GB RAM, 40GB storage	General purpose	4 cores
<input type="radio"/> Standard_A4m_v2 4 cores, 32GB RAM, 40GB storage	General purpose	4 cores
<input type="radio"/> Standard_A8_v2 8 cores, 16GB RAM, 80GB storage	General purpose	4 cores
<input type="radio"/> Standard_A8m_v2 8 cores, 64GB RAM, 80GB storage	General purpose	4 cores

**Back** **Next**

Create an Aks Compute Cluster and Configure it

This is being done to create an endpoint for our pipeline

# Create Inference Cluster & Pipeline

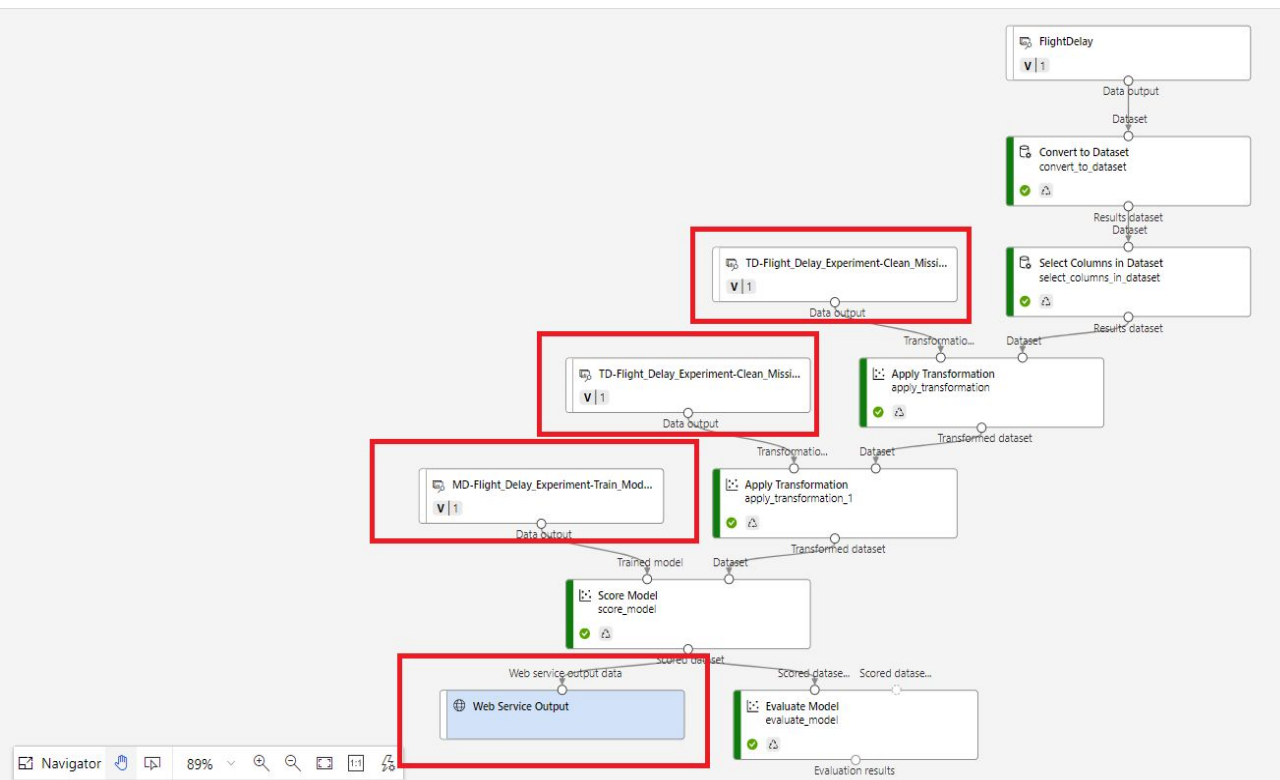


After creating an inference pipeline -

The Clean Data components along with Split Data and Model converted to transformations.

Web Service Output component auto added for REST API output which can be used externally.

Deploy it to the Aks Server for Endpoint



# Master Pipeline Run

The screenshot displays the Azure Data Factory (ADF) console interface. On the left, the 'Factory Resources' pane shows a list of resources under the 'Master Pipeline' folder, including Pipelines (1), Datasets (5), Data flows (1), and Power Query (0). The 'Activities' pane on the right shows a search bar and a list of activities: Move & transform, Synapse, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, HDInsight, Iteration & conditionals, Machine Learning, and Power Query.

The main canvas shows the 'Master Pipeline' flow, which consists of three activities: a Notebook (api raw json processing), a Data flow (ingestion data flow), and a Machine Learning Execute Pipeline (Machine Learning Execute Pipeline1). The flow is currently in a 'Debug' state, as indicated by the 'Data flow debug' toggle being turned on.

Below the flow, the 'Output' tab is selected, showing the 'Pipeline run ID: a6f23255-61f9-4a64-b8e5-e2b9ab613bb9'. A message states: 'Data flow activity for this debug run will start as soon as the data flow debug session is ready.' Below this, a table displays the run details:

	Run start	Duration	Status	Integ
ecute P...	4/21/2023, 10:12:27 PM	00:00:21	✓ Succeeded	Autof
	4/21/2023, 10:06:40 PM	00:05:46	✓ Succeeded	debu
	4/21/2023, 9:59:02 PM	00:07:37	✓ Succeeded	Autof

After successfully publishing our ML Studio Pipeline,

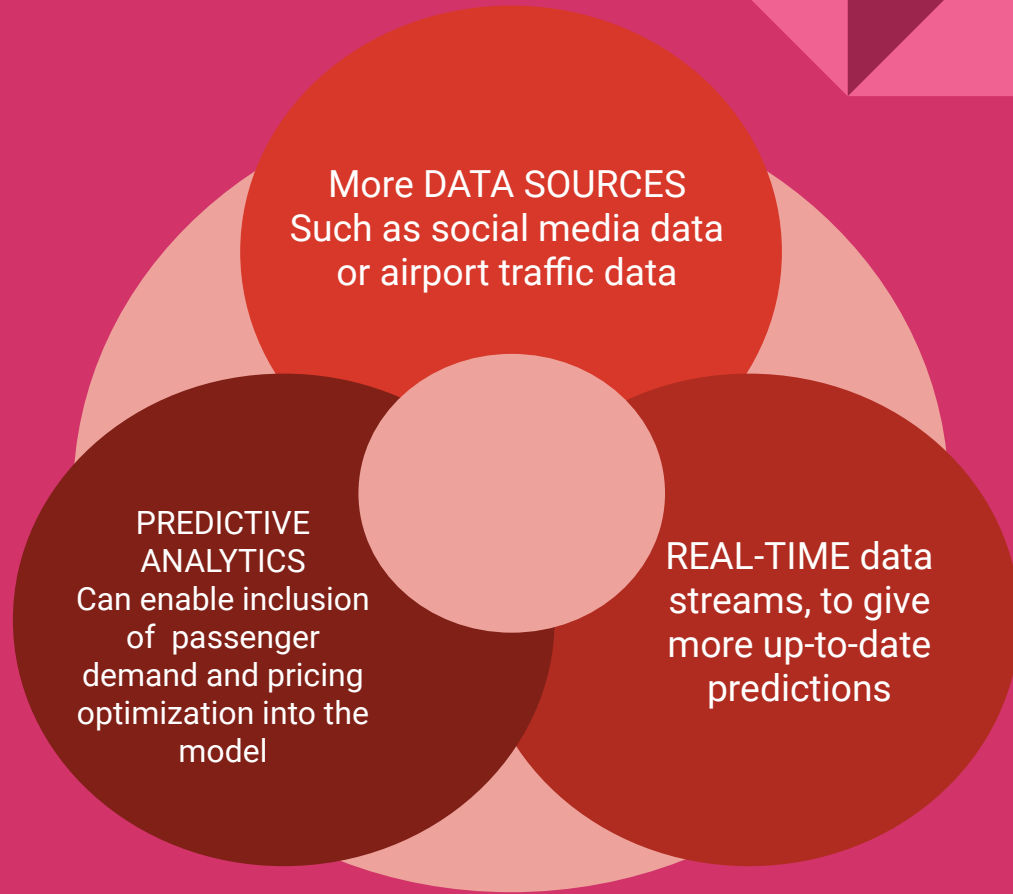
The Master ADF Pipeline was triggered to run the whole flow

# IMPACT

- By analysing historical data using machine learning techniques, airlines can predict the likelihood of flight delays and take necessary measures to minimize or avoid them.
- This can lead to better operational efficiency, better management of customer expectations, and improved profitability by reducing customer dissatisfaction and operational costs for the airline industry.



# Next Steps





Thank you!