Centre For Development Of Advanced Computing, Mumbai

Course : Post-Graduation Diploma in Big Data Analytics (DBDA)

Title : Data insights unveiling world bank story

Group Members:

1. Vibhishan Chougule (Project Lead)
2. Chirag Satpute
3. Saikiran kundrapu
4. Piyush Shejwal
5. Priti Singh

PROBLEM STATEMENT

Analyze historical data of International Bank for Reconstruction and Development (IBRD). This data mainly consists of Region, Country, Borrower, Guarantor, Loan Type, Loan Status, Interest Rate, Project Name, Original principal amount etc

The objective of our project is to analyze regional and country-specific trends in IBRD lending activities and to understand the characteristics of IBRD borrowers, including government entities, public agencies, private enterprises, and other organizations. The project focuses on assessing different types of loans offered by the IBRD (e.g., investment loans, development policy loans) and analyzing their status (e.g., active, closed, canceled). This could involve examining the purposes of loans, disbursement schedules, repayment terms, and project outcomes.

Another objective is to explore the dynamics of interest rates charged by the IBRD over time and across different regions and countries.

What is IBRD?

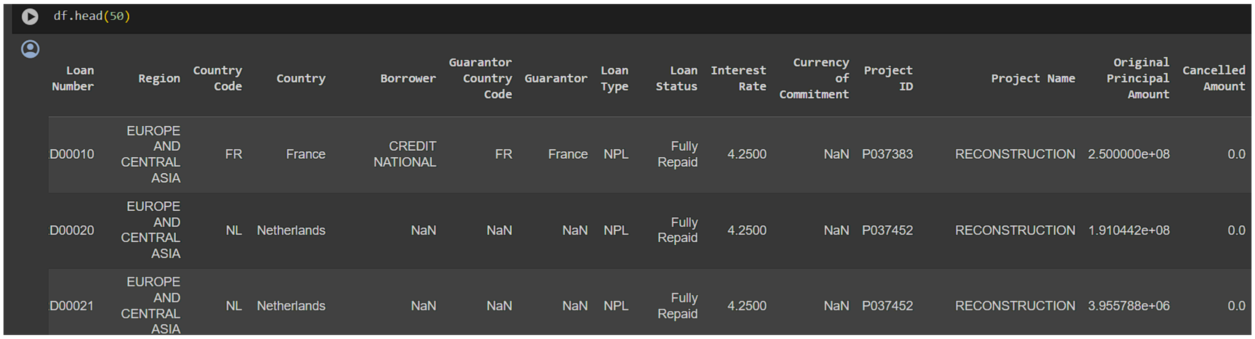
The International Bank for Reconstruction and Development (IBRD) is a specialized institution within the World Bank Group. Established in 1944, its primary aim is to provide financial and technical assistance to middle-income and creditworthy low-income countries for development projects.

About Dataset: The International Bank for Reconstruction and Development (IBRD) loans are public and publicly guaranteed debt extended by the World Bank Group. IBRD loans are made to, or guaranteed by, countries that are members of IBRD. IBRD may also make loans to IFC. IBRD lends at market rates. Data are in U.S. dollars calculated using historical rates. This dataset contains historical snapshots of the Statement of Loans including the latest available snapshots. The World Bank complies with all sanctions applicable to World Bank transactions.

EDA Insights

1. Dimensions of the dataset
(1278093, 34)

2. Few Records of the dataset



```
df.head(50)
```

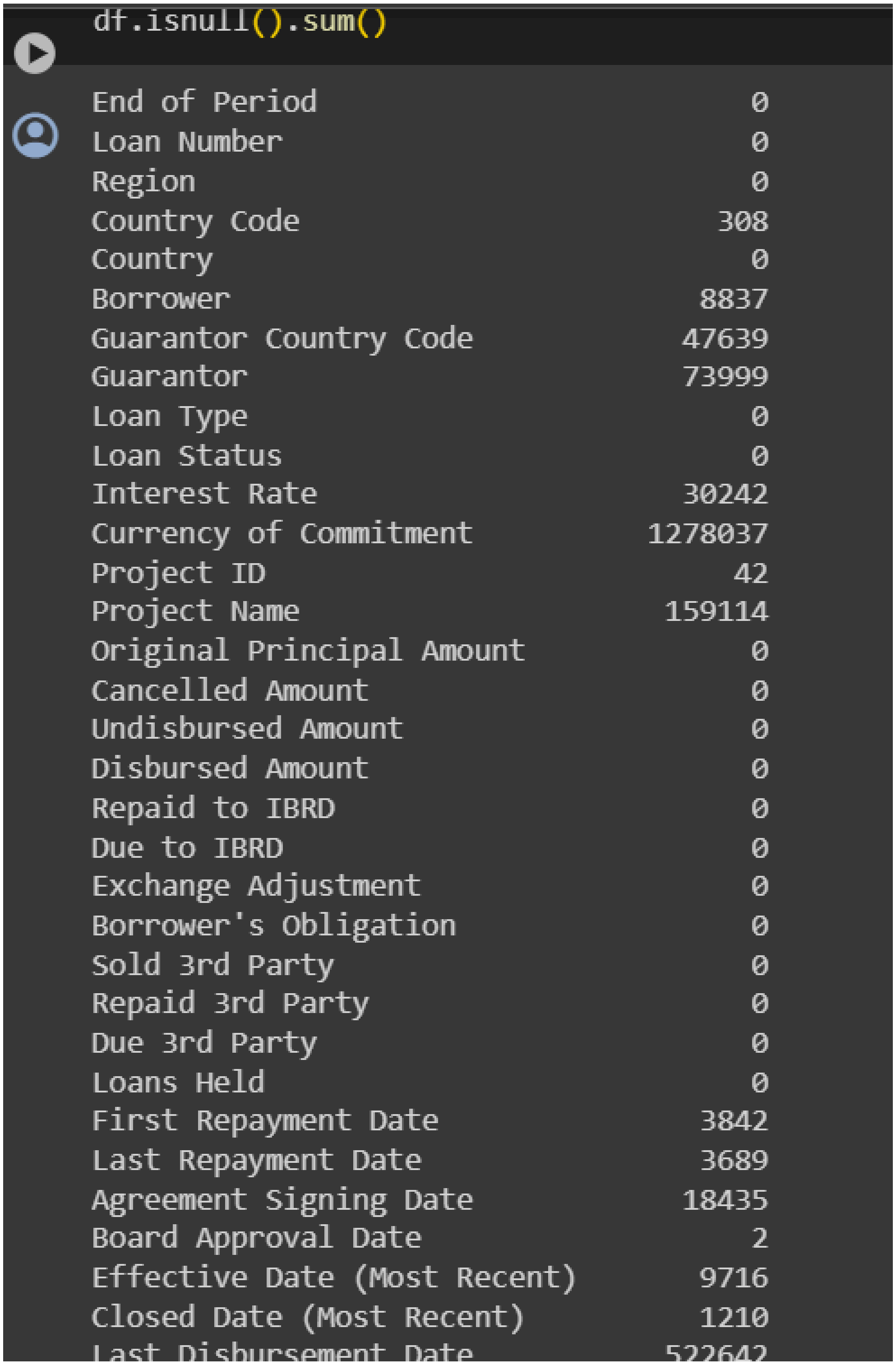
Loan Number	Region	Country Code	Country	Borrower	Guarantor Country Code	Guarantor	Loan Type	Loan Status	Interest Rate	Currency of Commitment	Project ID	Project Name	Original Principal Amount	Cancelled Amount
D00010	EUROPE AND CENTRAL ASIA	FR	France	CREDIT NATIONAL	FR	France	NPL	Fully Repaid	4.2500	NaN	P037383	RECONSTRUCTION	2.500000e+08	0.0
D00020	EUROPE AND CENTRAL ASIA	NL	Netherlands	NaN	NaN	NaN	NPL	Fully Repaid	4.2500	NaN	P037452	RECONSTRUCTION	1.910442e+08	0.0
D00021	EUROPE AND CENTRAL ASIA	NL	Netherlands	NaN	NaN	NaN	NPL	Fully Repaid	4.2500	NaN	P037452	RECONSTRUCTION	3.955788e+06	0.0

3.Feature names

End of Period, Loan Number, Region, Country Code, Country, Borrower, Guarantor Country Code, Guarantor, Loan Type, Loan Status, Interest Rate, Currency of Commitment, Project ID, Project Name, Original Principal Amount, Cancelled Amount, Undisbursed Amount, Disbursed Amount, Repaid to IBRD, Due to IBRD, Exchange Adjustment, Borrower's Obligation, Sold 3rd Party, Repaid 3rd Party, Due 3rd Party, Loans Held, First Repayment Date, Last Repayment Date, Agreement Signing Date, Board Approval Date, Effective Date (Most Recent), Closed Date (Most Recent), Last Disbursement Date, Days Difference

4. Null Value Count:

Our dataset has some null values in some attributes and duplicates data were also present. There were 93 duplicate rows present in our dataset.



```
df.isnull().sum()
```

```
End of Period                          0
Loan Number                            0
Region                                 0
Country Code                         308
Country                                0
Borrower                            8837
Guarantor Country Code             47639
Guarantor                          73999
Loan Type                              0
Loan Status                            0
Interest Rate                      30242
Currency of Commitment           1278037
Project ID                            42
Project Name                      159114
Original Principal Amount              0
Cancelled Amount                       0
Undisbursed Amount                     0
Disbursed Amount                       0
Repaid to IBRD                         0
Due to IBRD                            0
Exchange Adjustment                    0
Borrower's Obligation                  0
Sold 3rd Party                         0
Repaid 3rd Party                       0
Due 3rd Party                          0
Loans Held                             0
First Repayment Date                3842
Last Repayment Date                 3689
Agreement Signing Date             18435
Board Approval Date                    2
Effective Date (Most Recent)        9716
Closed Date (Most Recent)           1210
Last Disbursement Date            522642
```

Preprocessing

Preprocessing is done in Pyspark.

PySpark, released by Apache Spark community, is basically a Python API for supporting Python with Spark. By utilizing PySpark, we can work and integrate with RDD easily in Python. The library Py4j helps to achieve this feature.

There are several features of PySpark framework:

1. Faster processing than other frameworks.
2. Real-time computations and low latency due to in-memory processing.
3. Polyglot, which means compatible with several languages like Java, Python, Scala and R.
4. Powerful caching and efficient disk persistence.
5. Deployment can be performed by Hadoop through Yarn.

```
data=spark.read.format("csv").option("header","true").load("/mnt/Bird03/raw-data/IBRD_Statement_Of_Loans_-_Historical_Data_20240207.csv")
```

This line of code reads a CSV file from the specified path (`/mnt/Bird03/raw-data/IBRD_Statement_Of_Loans_-_Historical_Data_20240207.csv`) into a Spark DataFrame using PySpark. The `format("csv")` method specifies that the file is in CSV format, and the `option("header", "true")` method indicates that the first row of the CSV file contains the column headers. Finally, the `load` method is used to load the CSV file into a DataFrame called `data`.

```
data.show()
```

The `data.show()` method is used to display the contents of the DataFrame `data` in a tabular format. PySpark will output the first 20 rows of the DataFrame `data` to the console so we can inspect the data and its structure.

```
data.printSchema()
```

The `data.printSchema()` method is used to display the schema of the DataFrame `data`. The schema defines the structure of the DataFrame, including the names of the columns and their data types. PySpark will output the schema of the DataFrame `data` to the console, showing the column names and their corresponding data types.

```
num_rows = data.count()
num_columns = len(data.columns)
print("Shape of the DataFrame: ({}, {})".format(num_rows, num_columns))
```

1. `num_rows = data.count()`: This line calculates the number of rows in the DataFrame `data` using the `count()` method, which returns the total number of rows in the DataFrame.
2. `num_columns = len(data.columns)`: This line calculates the number of columns in the DataFrame `data` by taking the length of the `columns` attribute, which is a list of column names in the DataFrame.
3. `print("Shape of the DataFrame: ({}, {})".format(num_rows, num_columns))`: This line prints the shape of the DataFrame `data`, which includes the number of rows and columns, in the format `(num_rows, num_columns)`.

```
data = data.dropDuplicates()
num_rows_no_duplicates = data.count()
print("Number of duplicate rows removed:", num_rows - num_rows_no_duplicates)
```

1. `data = data.dropDuplicates()`: This line removes duplicate rows from the DataFrame `data` using the `dropDuplicates()` method.
2. `num_rows_no_duplicates = data.count()`: This line calculates the number of rows in the DataFrame `data` after removing duplicates using the `count()` method.
3. `print("Number of duplicate rows removed:", num_rows - num_rows_no_duplicates)`: This line prints the number of duplicate rows that were removed from the original DataFrame `data`.

```
column_types = data.dtypes
for column_name, data_type in column_types:
    print(f"Column '{column_name}' : '{data_type}'")
```

1. `column_types = data.dtypes`: This line retrieves the data types of all columns in the DataFrame `data` and stores them as a list of tuples, where each tuple contains the column name and its data type.
2. `for column_name, data_type in column_types:`: This line iterates over each tuple in the `column_types` list, unpacking the tuple into `column_name` (the column name) and `data_type` (the data type).
3. `print(f"Column '{column_name}' : '{data_type}'")`: This line prints the column name and its corresponding data type using an f-string for formatting.

```
from pyspark.sql.functions import when
data = data.withColumn("Loan Status",
                       when(data["Loan Status"] == "Fully Repaid", "Repaid")
                       .when(data["Loan Status"] == "Fully Cancelled",
"Cancelled")
                       .when(data["Loan Status"] == "Fully Transferred",
"Disbursed")
                       .when(data["Loan Status"] == "Fully Disbursed",
"Disbursed")
                       .when(data["Loan Status"] == "Disbursing&Repaying",
"Disbursing")
                       .when(data["Loan Status"] == "Repaying", "Effective")
                       .when(data["Loan Status"] == "Negotiated", "Signed")
                       .when(data["Loan Status"] == "Draft", "Signed")
                       .otherwise(data["Loan Status"]))
```

This code uses the `when` function from PySpark to replace values in the "Loan Status" column of a DataFrame `data`. Each `when` statement checks a condition (e.g., `data["Loan Status"] == "Fully Repaid"`) and replaces the value with a new one (e.g., `"Repaid"`).
- `data.withColumn("Loan Status", ...)` creates a new DataFrame where the "Loan Status" column is replaced with the result of the `when` conditions.
- Each `when` statement checks a condition and returns a new value if the condition is true. If none of the conditions are true, the original value (`data["Loan Status"]`) is returned using `otherwise`.

```
from pyspark.sql.functions import col
data = data.filter(col("Loan Status") != "8000000")
```

This code filters out rows from the DataFrame `data` where the value in the "Loan Status" column is "8000000". The `col` function is used to refer to the "Loan Status" column, and the `filter` method is used to keep only those rows where the value is not equal to "8000000".

```
from pyspark.sql.functions import to_date
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")

data = data.withColumn("First Repayment Date", to_date("First Repayment Date", "MM/dd/yyyy"))
data = data.withColumn("Last Repayment Date", to_date("Last Repayment Date", "MM/dd/yyyy"))
data = data.withColumn("Agreement Signing Date", to_date("Agreement Signing Date", "MM/dd/yyyy"))
data = data.withColumn("Board Approval Date", to_date("Board Approval Date", "MM/dd/yyyy"))
data = data.withColumn("Effective Date (Most Recent)", to_date("Effective Date (Most Recent)", "MM/dd/yyyy"))
data = data.withColumn("Closed Date (Most Recent)", to_date("Closed Date (Most Recent)", "MM/dd/yyyy"))
data = data.withColumn("Last Disbursement Date", to_date("Last Disbursement Date", "MM/dd/yyyy"))

data = data.withColumn("End of Period", to_date("End of Period", "MM/dd/yyyy HH:mm:ss"))
```

This code converts columns in the DataFrame `data` to date format using the `to_date` function from PySpark. The format `"MM/dd/yyyy"` specifies the input format of the date string in the column, and the function converts it to a date object.

- For columns like "First Repayment Date", "Last Repayment Date", "Agreement Signing Date", "Board Approval Date", "Effective Date (Most Recent)", "Closed Date (Most Recent)", and "Last Disbursement Date", the code converts them to date format directly.
- For the "End of Period" column, the code uses a format of `"MM/dd/yyyy HH:mm:ss"` to handle both date and time components.

The `spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")` line is used to set the time parser policy to `LEGACY`, which is required for the `to_date` function to work with certain date formats in older versions of PySpark.

```
from pyspark.sql.functions import col

data = data.withColumn("Interest Rate", col("Interest Rate").cast("float"))
data = data.withColumn("Currency of Commitment", col("Currency of
Commitment").cast("float"))
data = data.withColumn("Original Principal Amount", col("Original Principal
Amount").cast("float"))
data = data.withColumn("Cancelled Amount", col("Cancelled
Amount").cast("float"))
data = data.withColumn("Undisbursed Amount", col("Undisbursed
Amount").cast("float"))
data = data.withColumn("Disbursed Amount", col("Disbursed
Amount").cast("float"))
data = data.withColumn("Repaid to IBRD", col("Repaid to IBRD").cast("float"))
data = data.withColumn("Due to IBRD", col("Due to IBRD").cast("float"))
data = data.withColumn("Exchange Adjustment", col("Exchange
Adjustment").cast("float"))

data = data.withColumn("Borrower's Obligation", col("Borrower's
Obligation").cast("float"))
data = data.withColumn("Sold 3rd Party", col("Sold 3rd Party").cast("float"))
data = data.withColumn("Repaid 3rd Party", col("Repaid 3rd
Party").cast("float"))

data = data.withColumn("Due 3rd Party", col("Due 3rd Party").cast("float"))
data = data.withColumn("Loans Held", col("Loans Held").cast("float"))
```

This code snippet converts columns in the DataFrame `data` to float format using the `cast` method from PySpark. The `col` function is used to refer to the columns, and the `cast("float")` method is applied to convert them to float format.

```
from pyspark.sql.functions import col, sum

null_percentages = data.agg(*[
    (sum(col(c).isNull().cast("int")) / total_rows * 100).alias(c)
    for c in data.columns
])
```

- `sum(col(c).isNull().cast("int"))` calculates the total number of null values in column `c`.
- `(sum(col(c).isNull().cast("int")) / total_rows * 100)` calculates the percentage of null values in column `c` by dividing the total number of null values by the total number of rows (`total_rows`) and multiplying by 100.
- `alias(c)` assigns the column name `c` to the result of the calculation, so the resulting DataFrame will have columns named after the original columns in `data`.

The code uses a list comprehension to iterate over all columns in `data` and apply the calculation to each column. The `agg` function is then used to aggregate the results into a single row, which contains the percentage of null values for each column.

```
feature_data = data.select("End of Period", "Region", "Country", "Borrower",
"Guarantor", "Loan Type", "Loan Status", "Interest Rate",
                           "Original Principal Amount", "Cancelled Amount",
"Undisbursed Amount", "Project Name ",
                          "Disbursed Amount", "Repaid to IBRD", "Due to IBRD",
"Exchange Adjustment", "Borrower's Obligation",
                           "Loans Held", "First Repayment Date", "Last
Repayment Date", "Board Approval Date", "Effective Date (Most Recent)" ,
"Closed Date (Most Recent)")
```

New Data Frame is created i.e feature_data, in which useful columns are selected for further processing.

```
from pyspark.sql.functions import mean, col

numerical_columns = ['Interest Rate','Original Principal Amount', 'Cancelled
Amount', 'Undisbursed Amount', 'Disbursed Amount', 'Repaid to IBRD', 'Due to
IBRD', 'Exchange Adjustment', "Borrower's Obligation", 'Loans Held']

mean_values = feature_data.select(*(mean(col(c)).alias(c) for c in
numerical_columns)).collect()[0].asDict()
for column in numerical_columns:
    feature_data = feature_data.withColumn(column, when(col(column).isNull(),
mean_values[column]).otherwise(col(column)))

string_columns = ['Borrower', 'Loan Type', 'Loan Status']
mode_values = {}
for column in string_columns:
    mode_value =
feature_data.groupBy(column).count().orderBy(col("count").desc()).select(column
).first()[0]
    mode_values[column] = mode_value

for column in string_columns:
    feature_data = feature_data.withColumn(column, when(col(column).isNull(),
mode_values[column]).otherwise(col(column)))

feature_data = feature_data.withColumn("Project Name ", when(col("Project Name
").isNull(), "Unknown").otherwise(col("Project Name ")))
feature_data = feature_data.withColumn("Guarantor",
when(col("Guarantor").isNull(), "Unknown").otherwise(col("Guarantor")))

date_columns = ['End of Period', 'First Repayment Date', 'Last Repayment Date']
mode_values = {}
for column in date_columns:
    mode_value =
feature_data.groupBy(column).count().orderBy(col("count").desc()).select(column
).first()[0]
    mode_values[column] = mode_value
```

1. **Fill Null Values with Mean (for Numerical Columns):**
   - Calculates the mean value for each numerical column specified in `numerical_columns`.
   - Fills null values in each numerical column with its corresponding mean value.
2. **Fill Null Values with Mode (for String Columns):**
   - Calculates the mode value (most frequent value) for each string column specified in `string_columns`.
   - Fills null values in each string column with its corresponding mode value.
3. **Fill Null Values with "Unknown" (for 'Project Name' and 'Guarantor' Columns):**
   - Replaces null values in the 'Project Name' and 'Guarantor' columns with the string "Unknown".
4. **Fill Null Values with Mode (for Date Columns):**
   - Calculates the mode value for each date column specified in `date_columns`.
   - Fills null values in each date column with its corresponding mode value.

```
from pyspark.sql.functions import datediff

# Calculate the difference in days between the two dates
feature_data = feature_data.withColumn("Days Difference", datediff("Last
Repayment Date", "First Repayment Date"))
```

This code calculates the difference in days between the "Last Repayment Date" and "First Repayment Date" columns in the `feature_data` DataFrame using the `datediff` function from PySpark. It creates a new column called "Days Difference" containing the calculated differences in days.

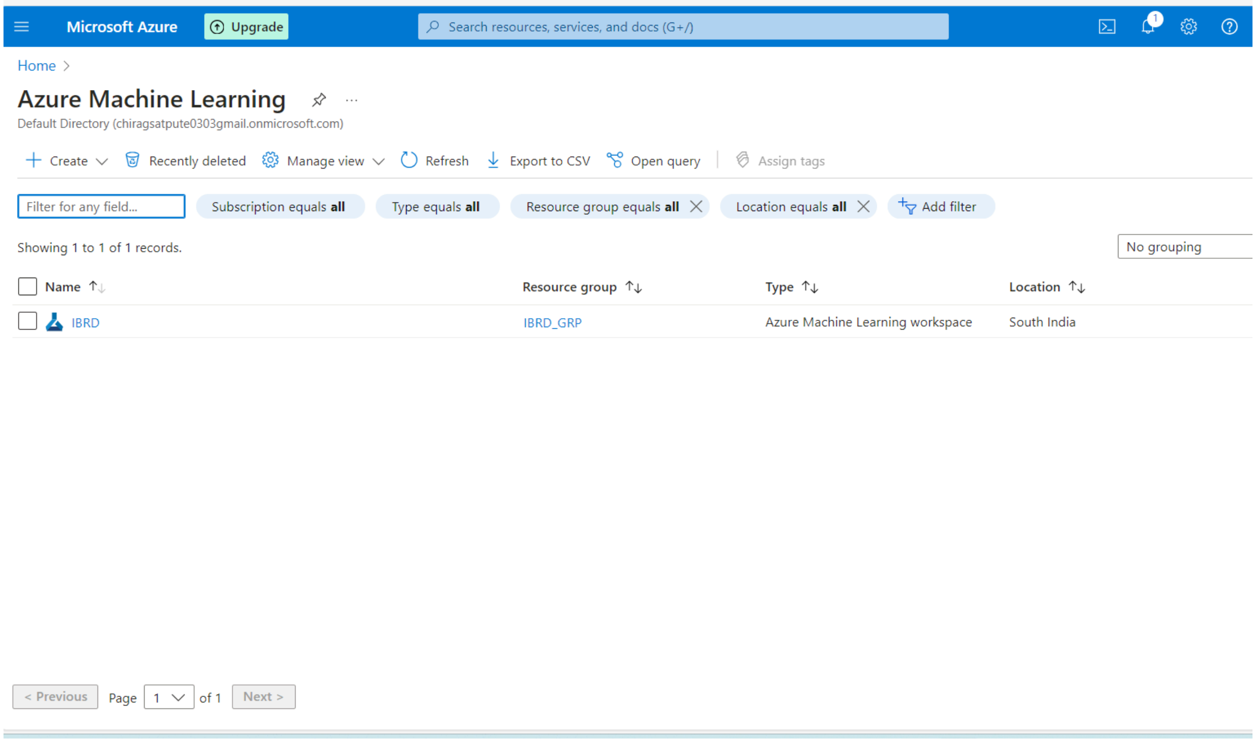
```
from pyspark.ml.feature import StringIndexer

# Apply StringIndexer to column 'country'
indexer_country = StringIndexer(inputCol='Country', outputCol='country_index')
feature_data = indexer_country.fit(feature_data).transform(feature_data)
```

This code uses the `StringIndexer` class from PySpark's MLlib to convert the categorical column "Country" into numerical indices. Here's how it works:
- `StringIndexer(inputCol='Country', outputCol='country_index')`: Creates a `StringIndexer` instance with input column "Country" and output column "country_index". This will assign a unique numerical index to each distinct country in the "Country" column.
- `indexer_country.fit(feature_data)`: Fits the `StringIndexer` on the `feature_data` DataFrame to learn the mapping from country names to numerical indices.
- `transform(feature_data)`: Transforms the `feature_data` DataFrame using the fitted `StringIndexer`, adding a new column "country_index" containing the numerical indices corresponding to the countries in the "Country" column.

Model Building

1. Access Azure Machine Learning Studio:

- Sign in to the Azure portal (https://portal.azure.com/).
- Navigate to the Azure Machine Learning workspace you've created.



Azure Machine Learning

Default Directory (chiragsatpute0303gmail.onmicrosoft.com)

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

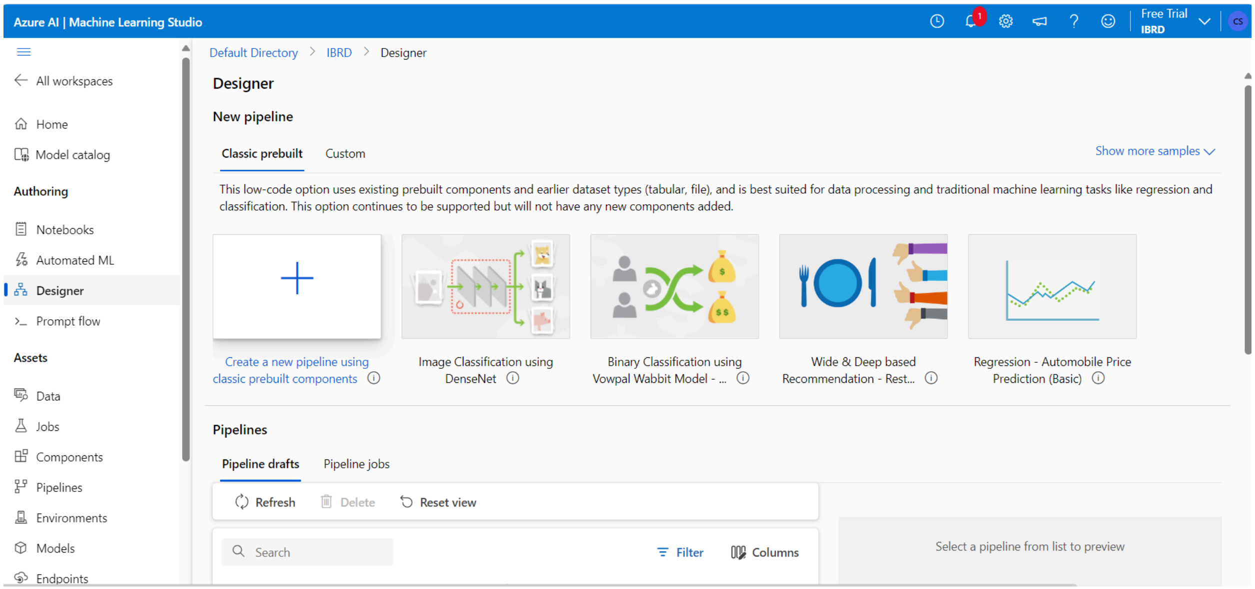
Showing 1 to 1 of 1 records.

No grouping

□ Name	Resource group	Type	Location
□ IBRD	IBRD_GRP	Azure Machine Learning workspace	South India

2. Create or Open a Pipeline:

- In the left menu, select "Designer" under the "Author" section.
- Here, you can create a new pipeline or open an existing one.



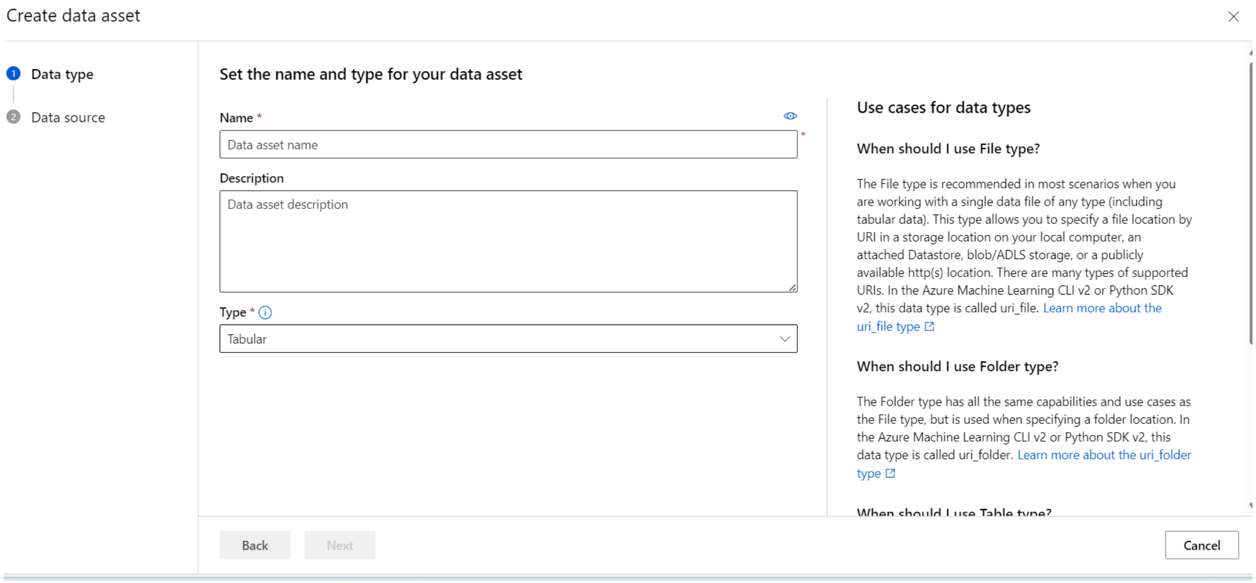
Azure AI | Machine Learning Studio
Free Trial
IBRD
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Default Directory > IBRD > Designer
Designer
New pipeline
Classic prebuilt
Custom
Show more samples

This low-code option uses existing prebuilt components and earlier dataset types (tabular, file), and is best suited for data processing and traditional machine learning tasks like regression and classification. This option continues to be supported but will not have any new components added.

Create a new pipeline using classic prebuilt components
Image Classification using DenseNet
Binary Classification using Vowpal Wabbit Model - ...
Wide & Deep based Recommendation - Rest...
Regression - Automobile Price Prediction (Basic)
Pipelines
Pipeline drafts
Pipeline jobs
Refresh
Delete
Reset view
Search
Filter
Columns
Select a pipeline from list to preview

3. Add Data:

- Drag the "Dataset" module from the "Data Input and Output" category onto the canvas.
- Connect it to your data source, such as Azure Blob Storage.
- Configure the dataset to load your training data.



Create data asset

Set the name and type for your data asset

Name *

Data asset name

Description

Data asset description

Type *

Tabular

Use cases for data types

When should I use File type?

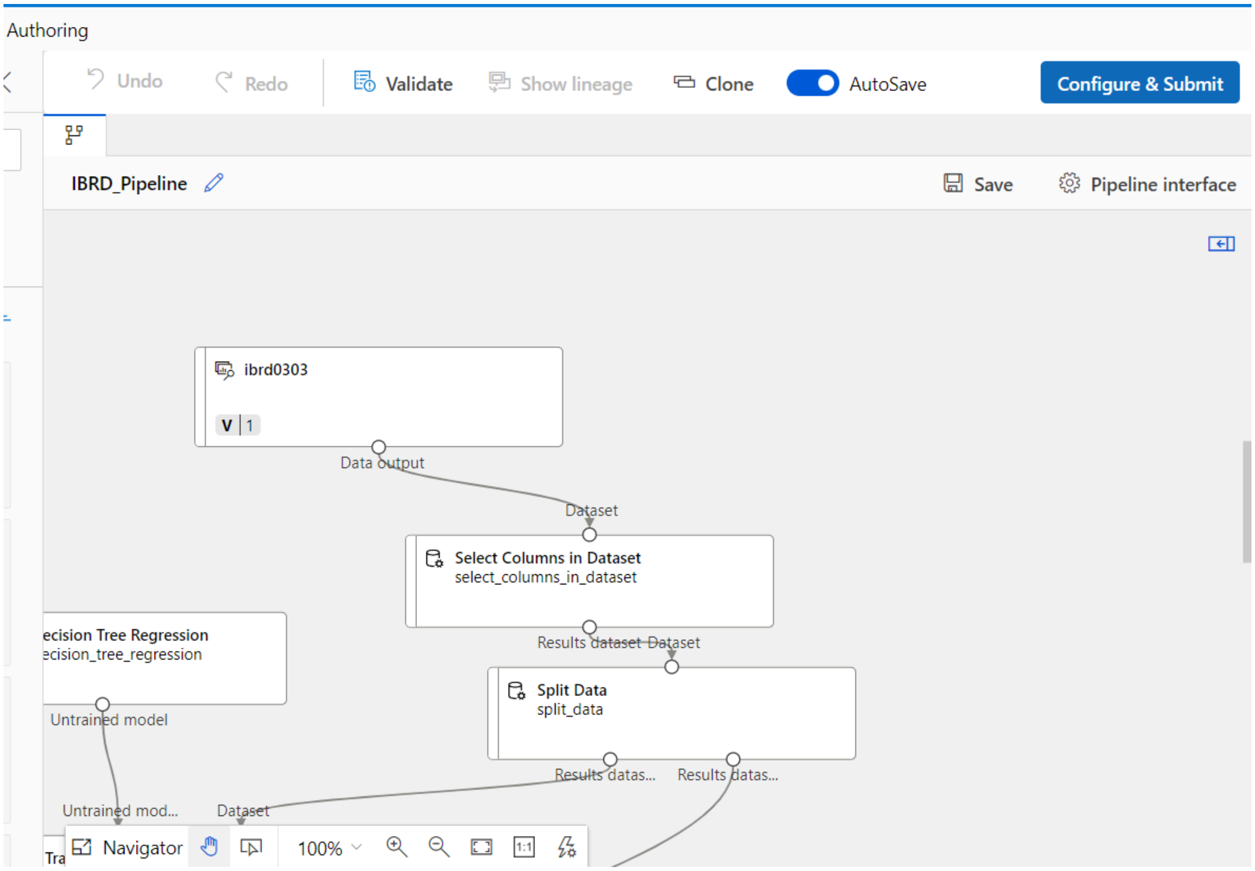
The File type is recommended in most scenarios when you are working with a single data file of any type (including tabular data). This type allows you to specify a file location by URI in a storage location on your local computer, an attached Datastore, blob/ADLS storage, or a publicly available http(s) location. There are many types of supported URIs. In the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called uri_file. Learn more about the uri_file type

When should I use Folder type?

The Folder type has all the same capabilities and use cases as the File type, but is used when specifying a folder location. In the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called uri_folder. Learn more about the uri_folder type

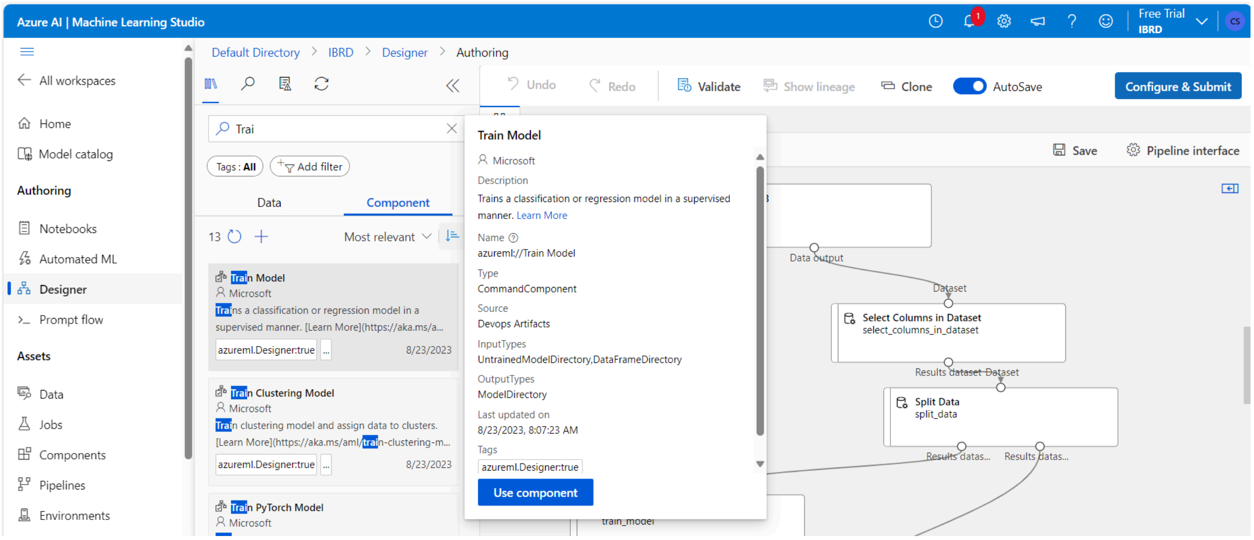
When should I use Table type?

4. Preprocess Data :

- Use modules like "Clean Missing Data," "Select Columns in Dataset," and to preprocess your data as needed.
- Drag these modules onto the canvas and connect them to your dataset.
- Splitting data into 70,30 train and test and selected columns are **country,loan type,interest rate,Disbursed Amount**
- 

5. Train Model:

- Drag the appropriate training algorithm module (e.g., "Train Model," "Train Regression Model") onto the canvas.
- Connect it to your preprocessed data.
- Configure the module by selecting the target column and any relevant settings.
- Our target column is **Interest rate.**



Train Model
Microsoft
Trains a classification or regression model in a supervised manner. [Learn More](https://aka.ms/a...
azureml.Designer:true ... 8/23/2023

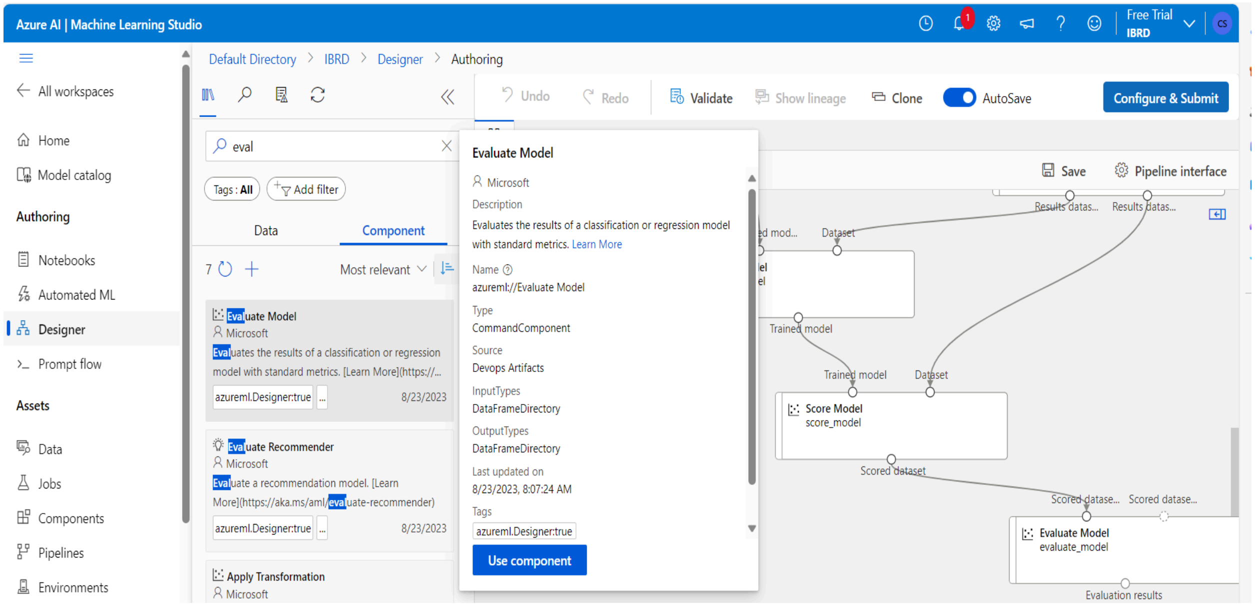
Train Clustering Model
Microsoft
Train clustering model and assign data to clusters. [Learn More](https://aka.ms/aml/train-clustering-m...
azureml.Designer:true ... 8/23/2023

Train PyTorch Model
Microsoft

Train Model
Microsoft
Description
Trains a classification or regression model in a supervised manner. Learn More
Name
azureml://Train Model
Type
CommandComponent
Source
Devops Artifacts
InputTypes
UntrainedModelDirectory,DataFrameDirectory
OutputTypes
ModelDirectory
Last updated on
8/23/2023, 8:07:23 AM
Tags
azureml.Designer:true

6. Evaluate Model:

- Add an evaluation module ("Score Model," "Evaluate Model") to assess the performance of your trained model.
- Connect it to your trained model and evaluation dataset.
- Configure the module to use appropriate evaluation metrics.



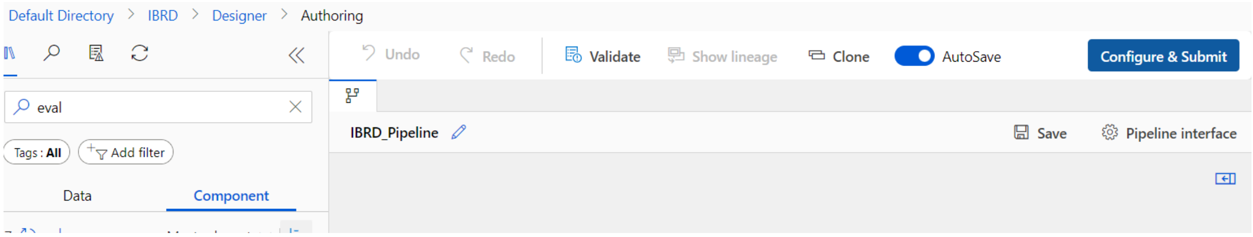
Evaluate Model
Microsoft
Evaluates the results of a classification or regression model with standard metrics. [Learn More](https://...
azureml.Designer:true ... 8/23/2023

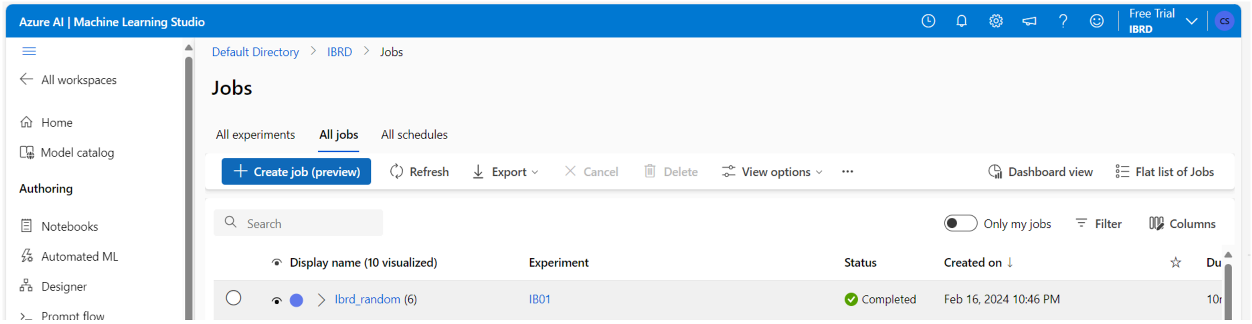
Evaluate Recommender
Microsoft
Evaluate a recommendation model. [Learn More](https://aka.ms/aml/evaluate-recommender)
azureml.Designer:true ... 8/23/2023

Evaluate Model
Microsoft
Description
Evaluates the results of a classification or regression model with standard metrics. Learn More
Name
azureml://Evaluate Model
Type
CommandComponent
Source
Devops Artifacts
InputTypes
DataFrameDirectory
OutputTypes
DataFrameDirectory
Last updated on
8/23/2023, 8:07:24 AM
Tags
azureml.Designer:true

7.Run Pipeline:

- Click "Configure Submit" or "Run" to execute your pipeline.
- This will process your data, train the model, evaluate its performance, and deploy it as a web service.

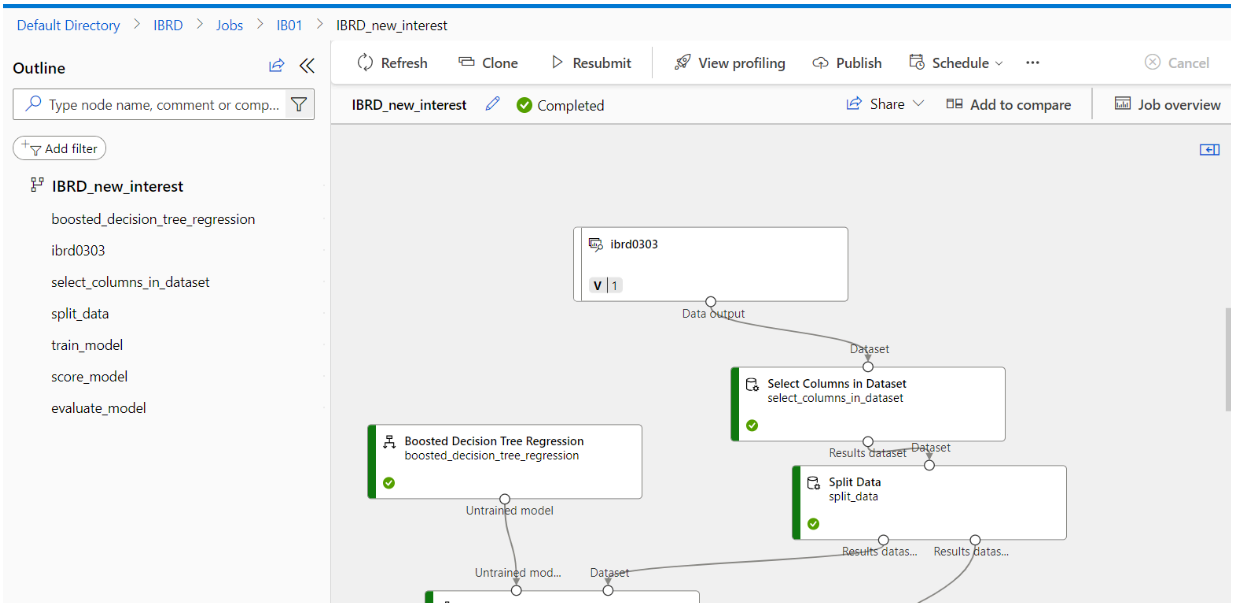


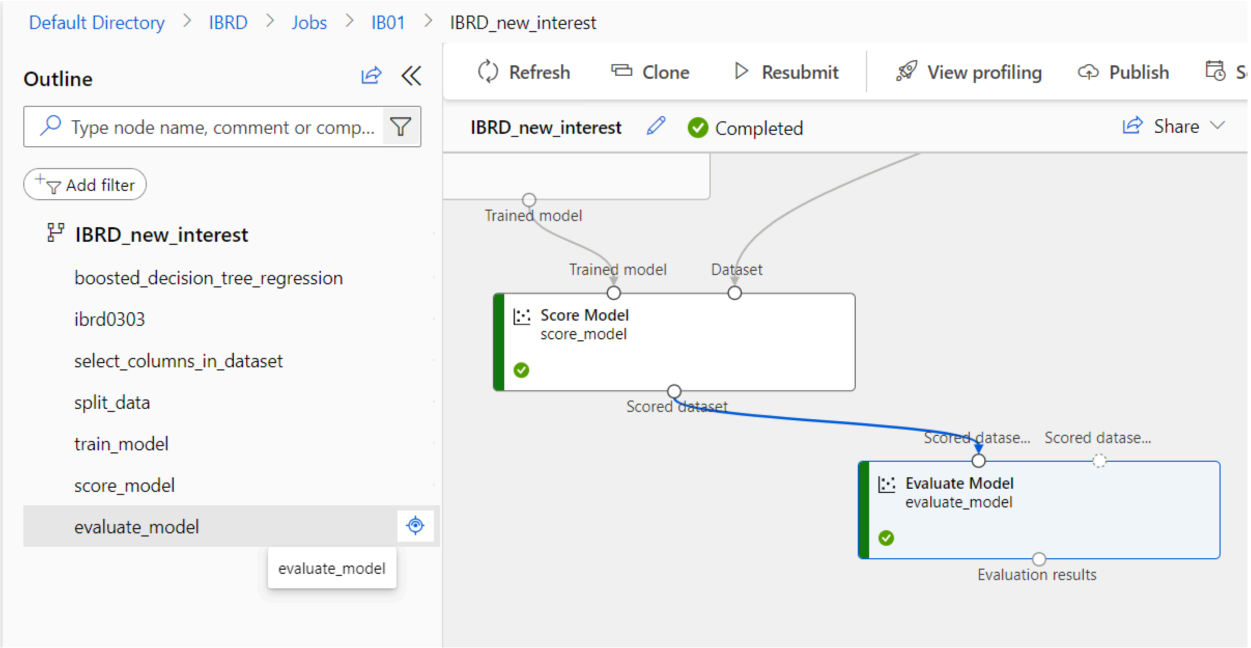


Display name (10 visualized)	Experiment	Status	Created on ↓	Du
Ibrd_random (6)	IB01	Completed	Feb 16, 2024 10:46 PM	10r

Monitor Pipeline Progress:

- Keep an eye on the pipeline's progress as it executes. You can view the status of the pipeline run in real-time in the AML Studio.
- Navigate to the "Pipeline runs" tab to see details such as the start time, duration, and current status of the pipeline run.



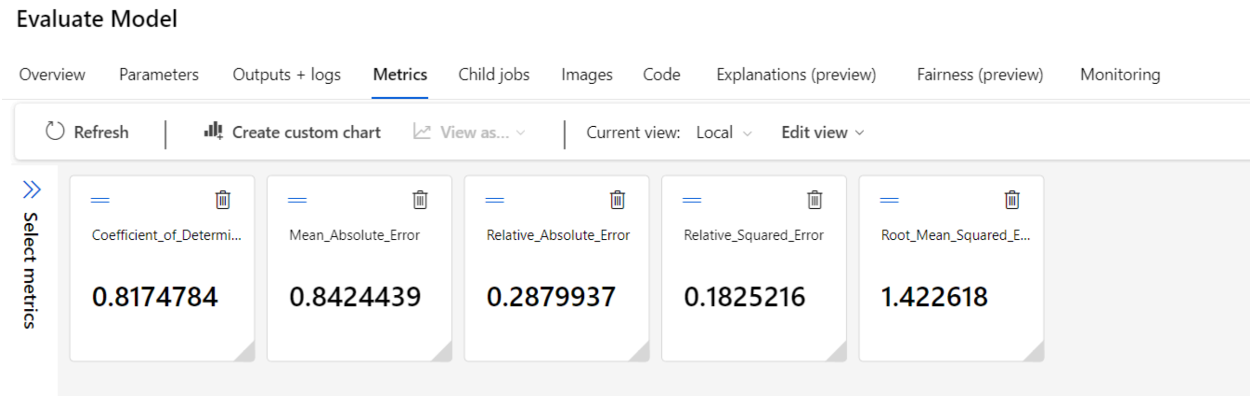


Retrieve Evaluation Metrics:

- If the pipeline includes modules for model evaluation, locate the output containing evaluation metrics.
- Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score, ROC curve, and AUC.

For regression tasks

- Evaluation metrics may include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared. We got **81.74%** accuracy for **Decision Forest Regression Model** and for **The Neural Network Regression** we got **68.4%**



Evaluate Model

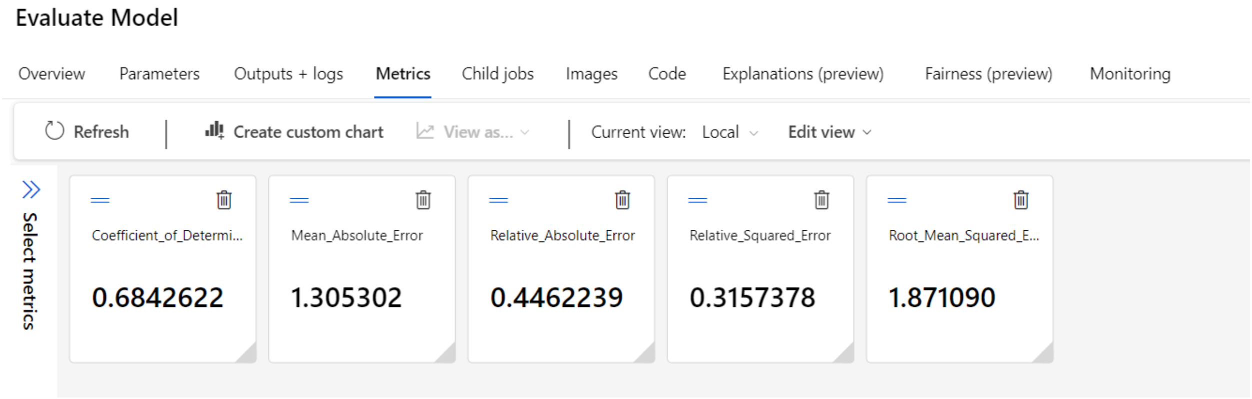
Coefficient_of_Determi... 0.8174784

Mean_Absolute_Error 0.8424439

Relative_Absolute_Error 0.2879937

Relative_Squared_Error 0.1825216

Root_Mean_Squared_E... 1.422618



Evaluate Model

Coefficient_of_Determi... 0.6842622

Mean_Absolute_Error 1.305302

Relative_Absolute_Error 0.4462239

Relative_Squared_Error 0.3157378

Root_Mean_Squared_E... 1.871090

Models used:

1.**Decision Forest Regression Model:** It is an ensemble learning technique that consists of multiple decision trees. Each decision tree in the ensemble outputs a Gaussian distribution as a prediction, and the final prediction is determined by aggregating the distributions from all trees.
The Decision Forest Regression component in Azure Machine Learning designer is used to create a regression model based on an ensemble of decision trees. Here's a brief explanation:

How it Works:

- Decision trees are non-parametric models that make a sequence of simple tests for each instance, traversing a binary tree data structure until a leaf node (decision) is reached.
- The regression model created by this component consists of an ensemble of decision trees. Each tree predicts a Gaussian distribution, and an aggregation is performed over the ensemble to find the Gaussian distribution closest to the combined distribution for all trees in the model.

Advantages of Decision Trees:

- Decision trees are efficient in both computation and memory usage during training and prediction.
- They can represent non-linear decision boundaries.
- They perform integrated feature selection and classification and are resilient in the presence of noisy features.

2.Boosted Decision Tree Regression: It is a powerful component in Azure Machine Learning designer that allows you to create an ensemble of regression trees using boosting. Boosting entails building a series of trees where each subsequent tree learns from the errors of the previous ones, thereby improving accuracy over time. This method is particularly effective for regression tasks and is based on the LightGBM algorithm.
Boosted regression trees, a type of ensemble model, leverage gradient boosting to iteratively build a series of decision trees. The model optimizes a predefined loss function at each step to minimize prediction errors, resulting in a robust ensemble model that combines the predictions of multiple weaker models.

3.The Neural Network Regression: It is in Azure Machine Learning designer is used to create a regression model using a customizable neural network algorithm. Here's an overview of this component:

Component Overview:

- Neural network regression is a supervised learning method used for predicting numerical values.
- Neural networks can approximate nonlinear functions of their inputs, making them suitable for regression problems where traditional models may not fit.

Advantages:

- Flexibility: Neural networks can model complex relationships between inputs and outputs.
- Adaptability: They can be customized extensively to suit the problem at hand.
- Automatic Feature Extraction: Neural networks can automatically learn relevant features from the data, reducing the need for manual feature engineering.

4. Linear Regression:

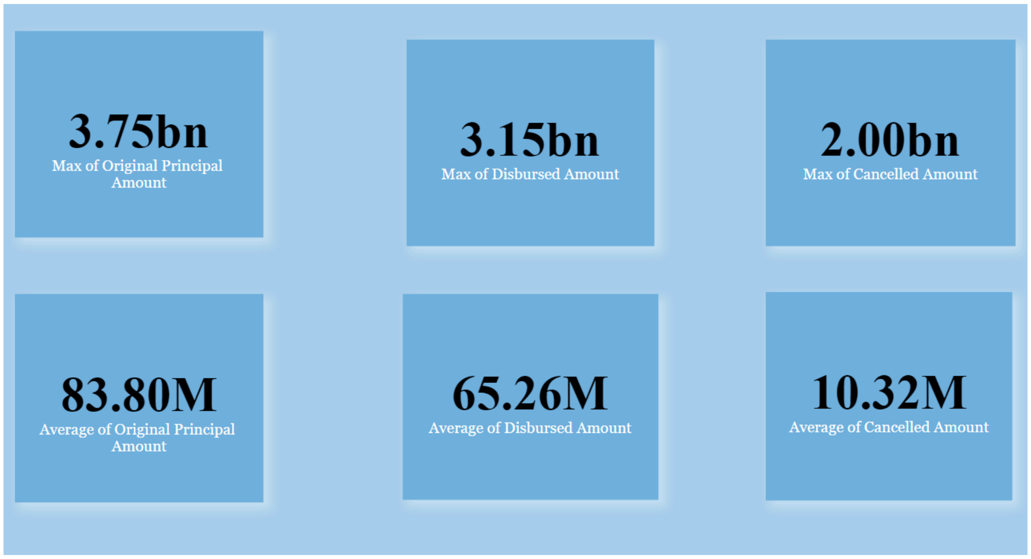
- Linear regression establishes a linear relationship between one or more independent variables and a numeric outcome, also known as the dependent variable.
- It's a common statistical method used for prediction tasks.
- Linear regression is suitable for simple predictive tasks and works well with high-dimensional, sparse datasets.

Data Visualization

We have used Power Bi for Data Visualization for our project. We chose Power BI for our project because it offers robust data visualization and analysis capabilities, enabling us to easily connect to various data sources, transform data, and create interactive reports and dashboards.

Insights:

1.



3.75bn
Max of Original Principal Amount

3.15bn
Max of Disbursed Amount

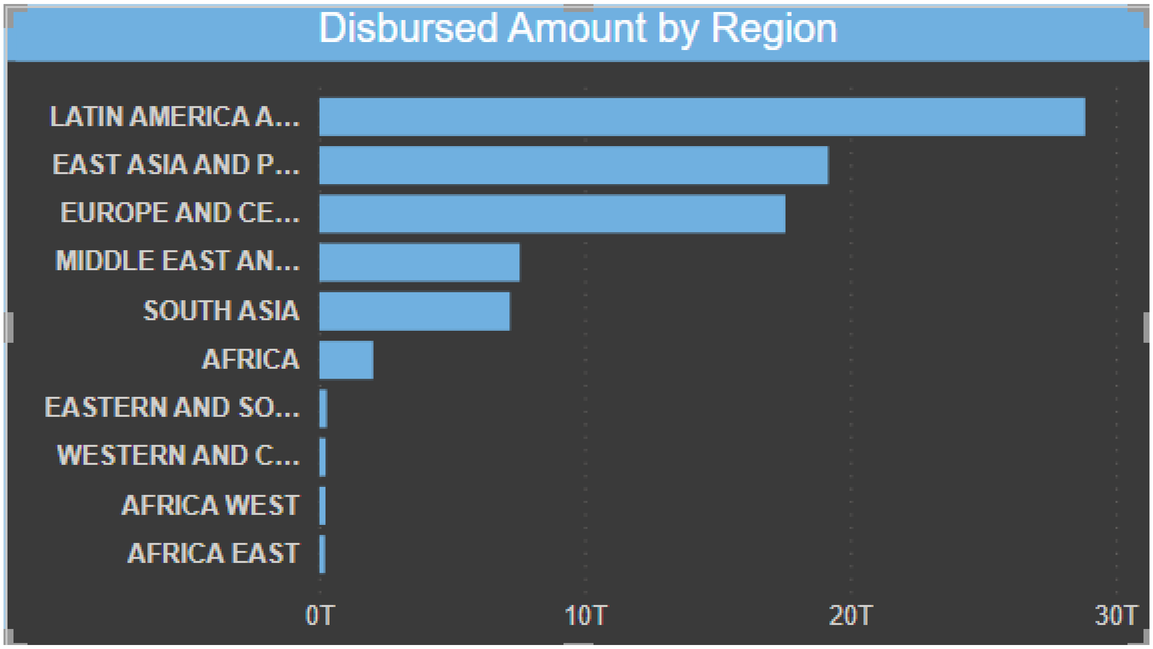
2.00bn
Max of Cancelled Amount

83.80M
Average of Original Principal Amount

65.26M
Average of Disbursed Amount

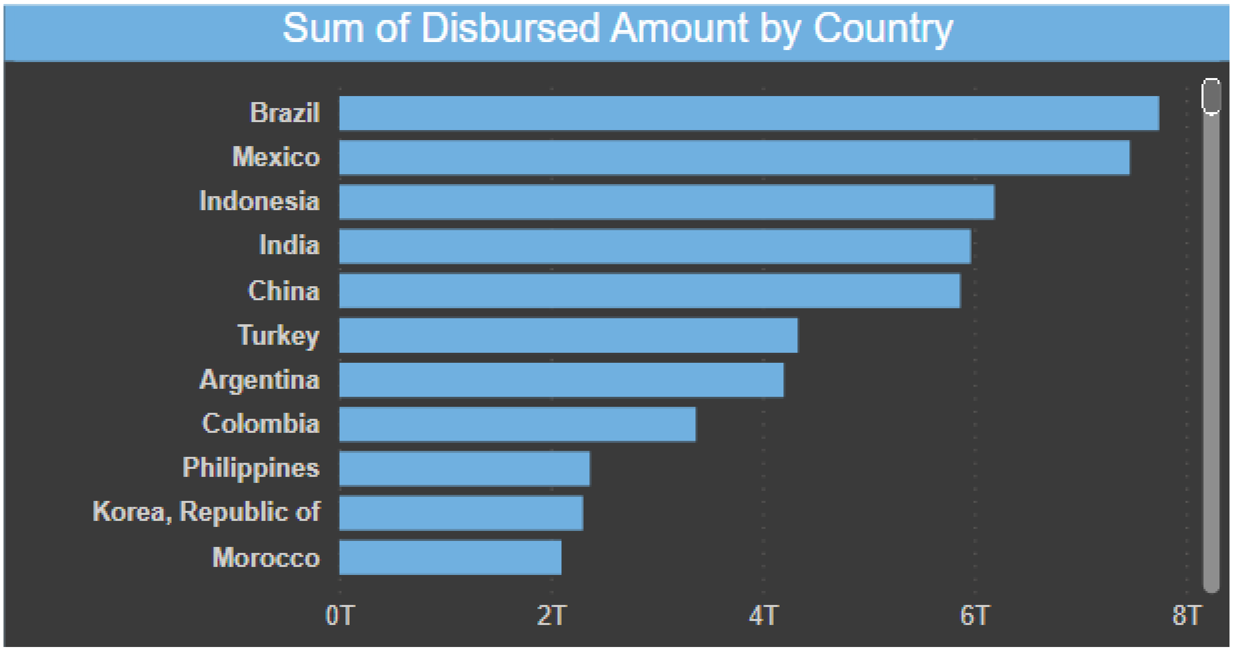
10.32M
Average of Cancelled Amount

2.



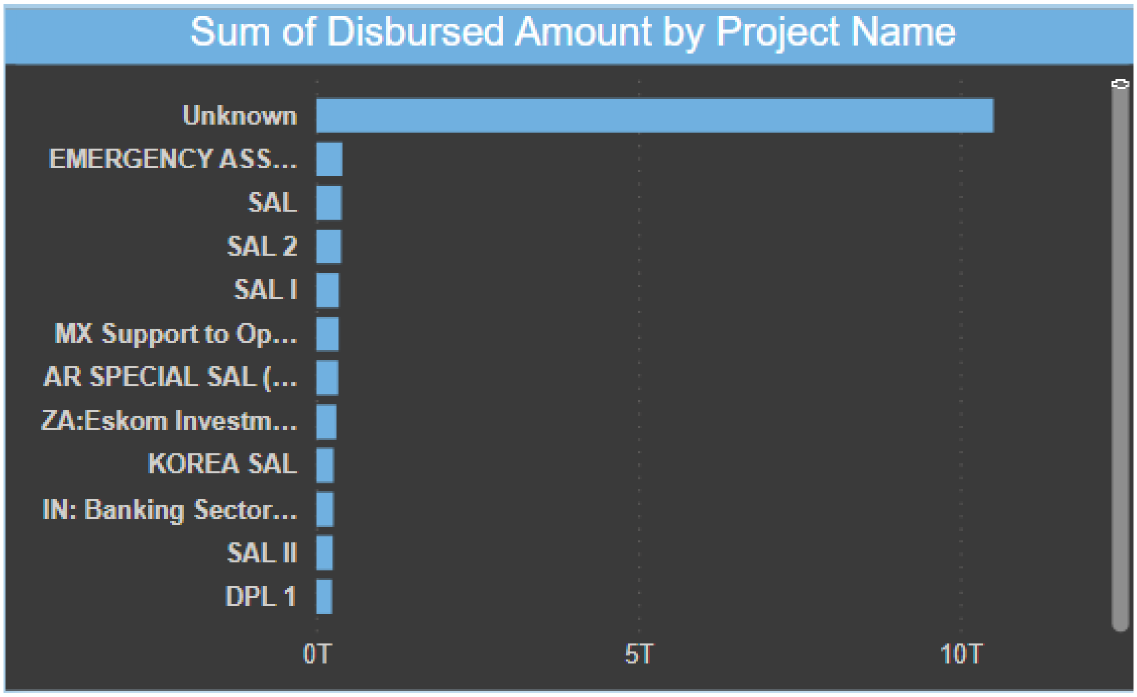
This is a horizontal bar chart. We have shown the amount disbursed by region over the period of 1947-2023.
Maximum disbursed amount is disbursed in the region of Latin America followed by East Asia and Europe.
Many countries in Latin America (Brazil, Mexico), especially in the mid to late 20th century, required extensive infrastructure development. This included building roads, bridges, ports, and utilities such as electricity and water systems. Some Latin American countries (Argentina, columbia) faced economic challenges such as poverty, inequality, and underdevelopment. The IBRD provided loans to support economic development initiatives, including education, healthcare, agriculture, and industry, with the goal of improving living standards and reducing poverty in the region.

3.



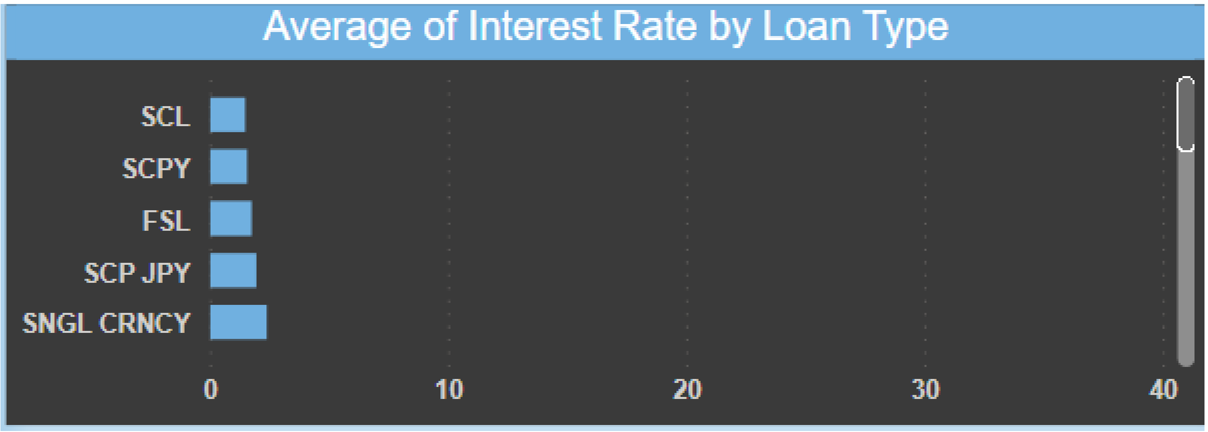
The Horizontal bar chart showing Disbursed Amount by country Over a period of 1947-2023
Maximum disbursed amount is disbursed in the country of Brazil followed by Mexico and Indonesia for projects that include infrastructure development (such as roads, bridges, and energy projects), social programs (education and healthcare), or economic reforms aimed at fostering sustainable growth.

4.



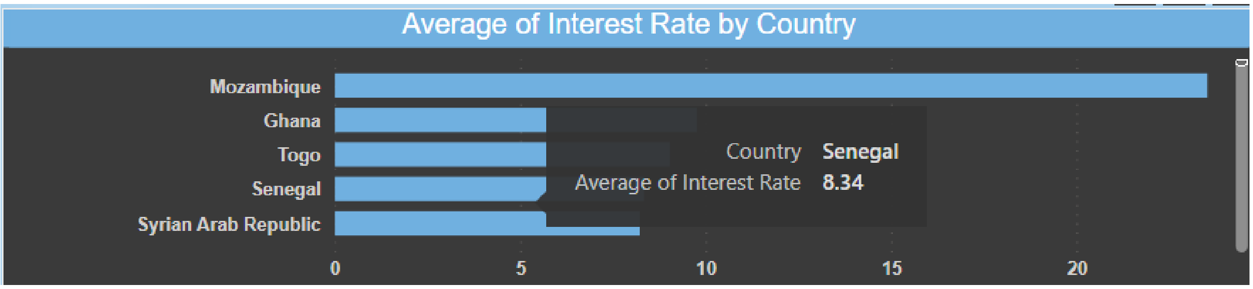
Here we have disbursed the amount by project name. This graph indicates what kind of project the IBRD has given the loan. Here the “Unknown” project may include Infrastructure Development which includes projects related to the construction and improvement of roads, highways, railways, ports, airports, water supply systems etc. Also it may include Healthcare facilities, Private Sector Development etc.

5.



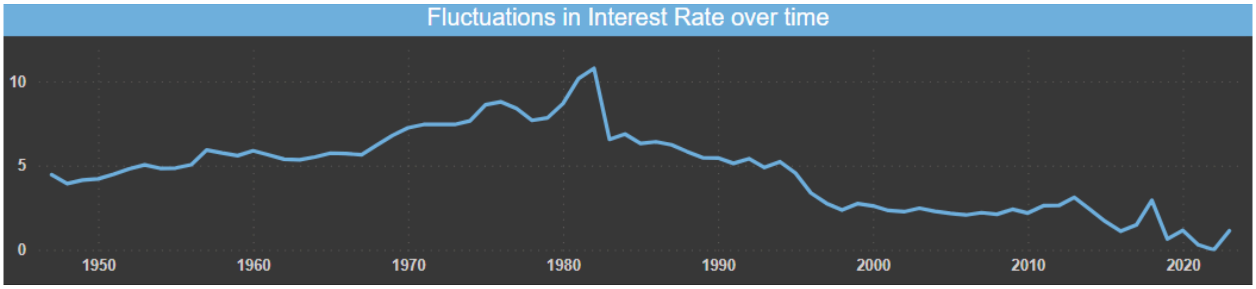
This graph simply shows the average interest rate by loan type.

6.



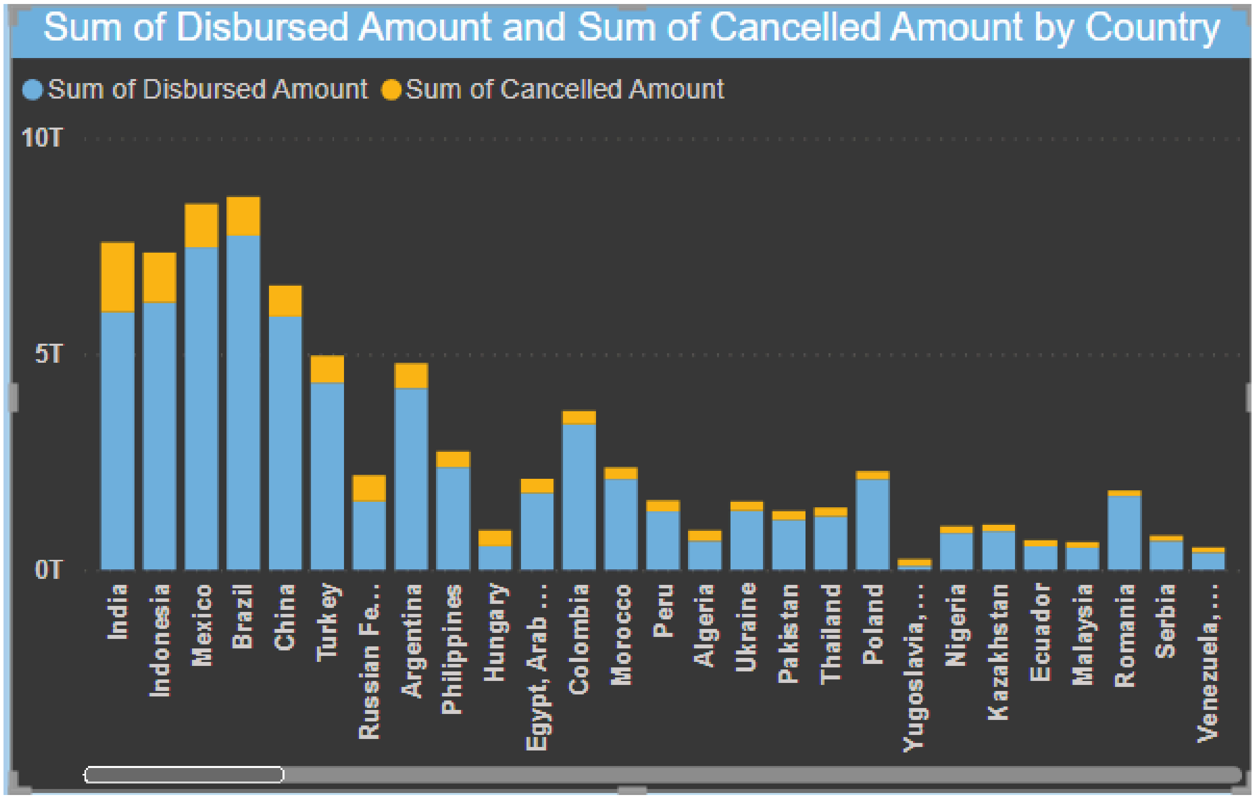
This graph shows the average interest rates charged by IBRD over a period of 1947-2023 by countries. Here we can see that the countries like Mozambique, ghana, togo etc have given the maximum interest rate to IBRD. IBRD charges interest rates on its loans that are based on several factors, including the creditworthiness of the borrowing country, the term of the loan, and prevailing market conditions. Here IBRD has considered risk associated with lending to each country when determining interest rates. Countries with weaker economic fundamentals, higher levels of debt, lower credit ratings, or a history of default may be considered higher credit risks by lenders like the IBRD. As a result, these countries may be charged higher interest rates to compensate for the increased risk of loan default. The IBRD may offer lower interest rates to countries that demonstrate strong commitment to policy reforms, good governance practices, and transparency. Conversely, countries with governance challenges, corruption issues, or weak institutional capacity may face higher interest rates as a reflection of increased risk.

7.



This shows the fluctuation in interest rate over a period of 1947-2023. The fluctuation in interest rate can be associated with Global economic condition, Monetary policy change, currency exchange rate, Funding strategy etc. Here we can see that the duration of the late 1970s and early 1980s has a maximum interest rate. The reason behind this is in the late 1970s there was a global recession which was triggered by the oil prices shock and this has put upward pressure on interest rates globally. The early 1980s saw the onset of a severe debt crisis in many developing countries, particularly in Latin America, Africa, and parts of Asia and had an adverse impact on interest rates.

8.



This graph shows the sum of the Disbursed amount and sum of cancelled amount with respect to countries. The IBRD has cancelled some amount from the original principle demanded by country. The reasons could be Country's Economic Conditions, Policy Changes or Reforms, Development Priorities, Political instability etc

Azure

Microsoft Azure offers discounts and free credits for students and educational institutions. Azure also provides extensive educational resources, including tutorials, documentation, and learning paths, making it easier for students to get started with cloud computing.

Aws has free services but some services are paid which is required for the project while azur gives Credits with that we can use any required services From the perspective of students Azur is best as compare to AWS

Cloud storage

Blob Storage:

Blob Storage is a type of object-based cloud storage designed for unstructured or Blob Storage is designed for unstructured data storage semi-structured data.
Blob storage cost is lower.
Blob Storage is used for storing and retrieving large files, such as images, videos.

Data Lake Gen2:

It is basically designed to store and process large volumes of data in various formats.
Data Lake is designed for big data analytics uses a distributed file system to provide parallel access to data.
It also integrates with a variety of big data processing frameworks and tools, such as Hadoop, Spark, and Azure Data Factory, enabling you to perform advanced analytics and machine learning on your data.

Databricks:

IT is a software company founded by the creators of Apache Spark. The company has also created famous software such as Delta Lake, MLflow Azure Databricks is a fast, easy, and collaborative Apache Spark-based analytics platform that is built on top of the Microsoft Azure cloud

Machine learning Studio :

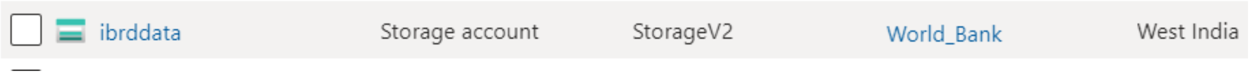
Azure ML Studio is primarily focused on providing a codeless experience through drag-and-drop elements.

Microsoft Azure Machine Learning Studio is a collaborative, drag-and-drop tool you can use to build, test, and deploy predictive analytics solutions on your data. Machine Learning Studio publishes models as web services that can easily be consumed by custom apps or BI tools such as Excel.

Cloud Pipeline

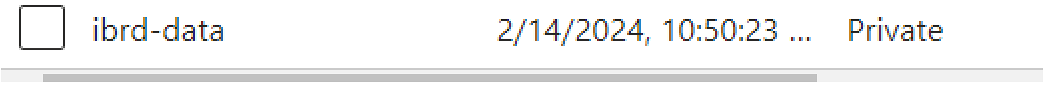
We started by loading the data on data Lake Gen 2

1) For that first create Storage Account



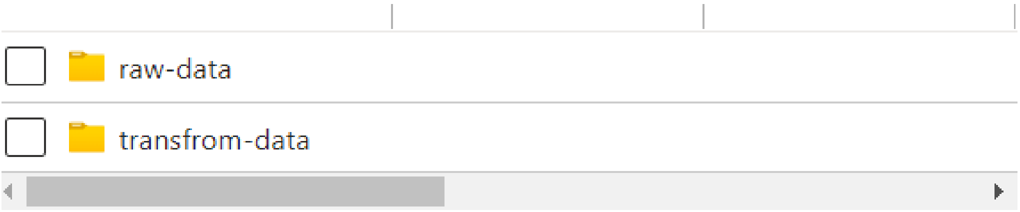
□	ibrddata	Storage account	StorageV2	World_Bank	West India

2) after creating account we create container to store the File



□	ibrd-data	2/14/2024, 10:50:23 ...	Private

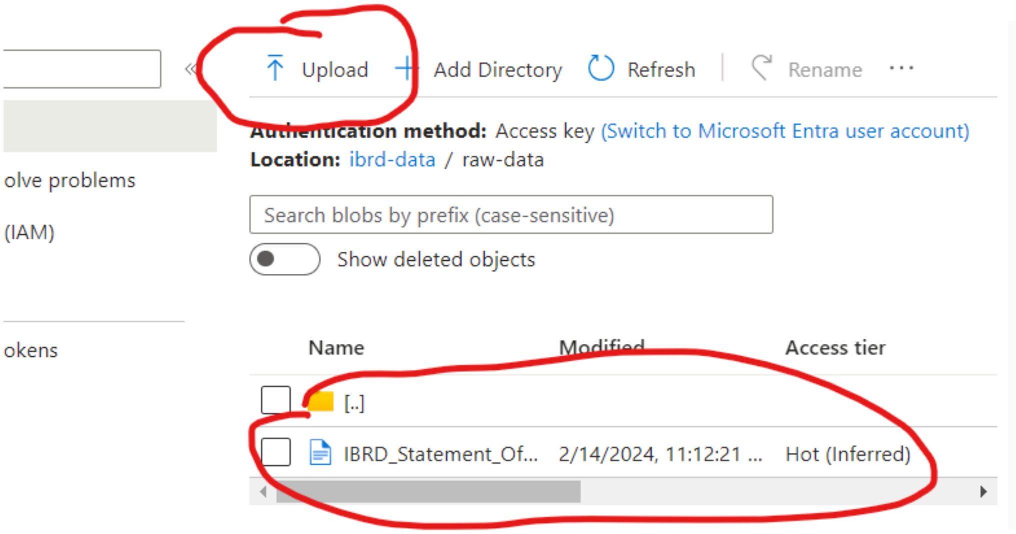
3) now we create 2 folders in that 1st to store Raw data 2nd to keep transform clean data after processing



□ raw-data

□ transfrom-data

4) now we export the dataset from local system



Upload | Add Directory | Refresh | Rename | ···

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: ibrd-data / raw-data

Search blobs by prefix (case-sensitive)

Show deleted objects

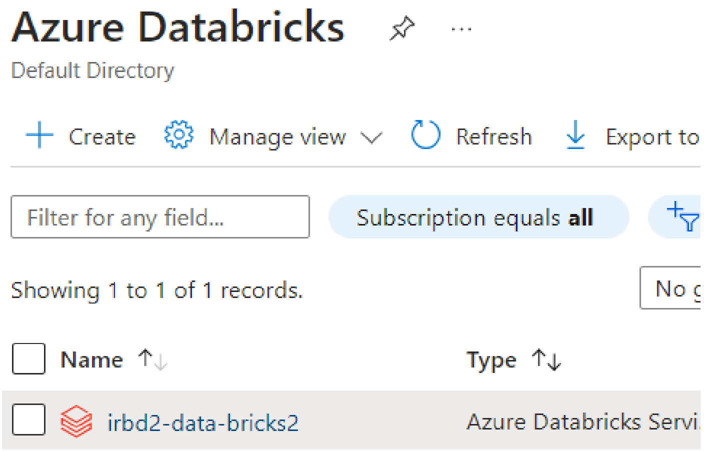
olve problems

(IAM)

okens

	Name	Modified	Access tier
□	[..]		
□	IBRD_Statement_Of...	2/14/2024, 11:12:21 ...	Hot (Inferred)

5)Now we want to start pre-processing som we start databricks



Azure Databricks

Default Directory

Create | Manage view | Refresh | Export to

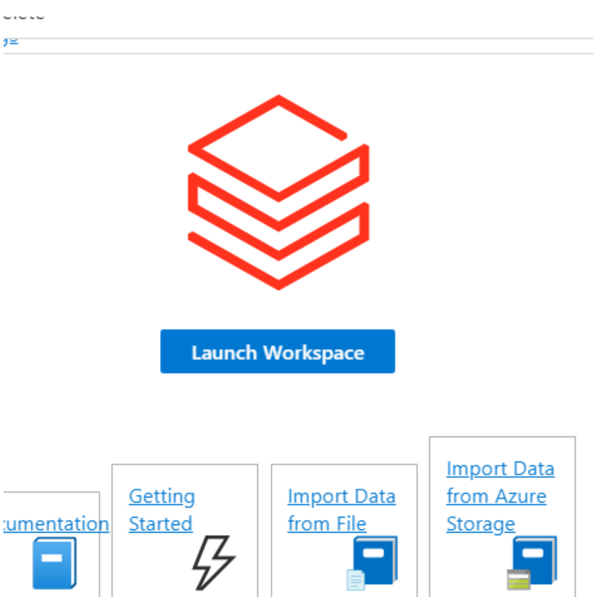
Filter for any field...

Subscription equals **all**

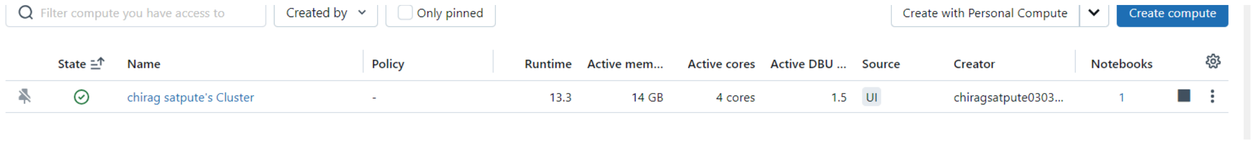
Showing 1 to 1 of 1 records.

No g

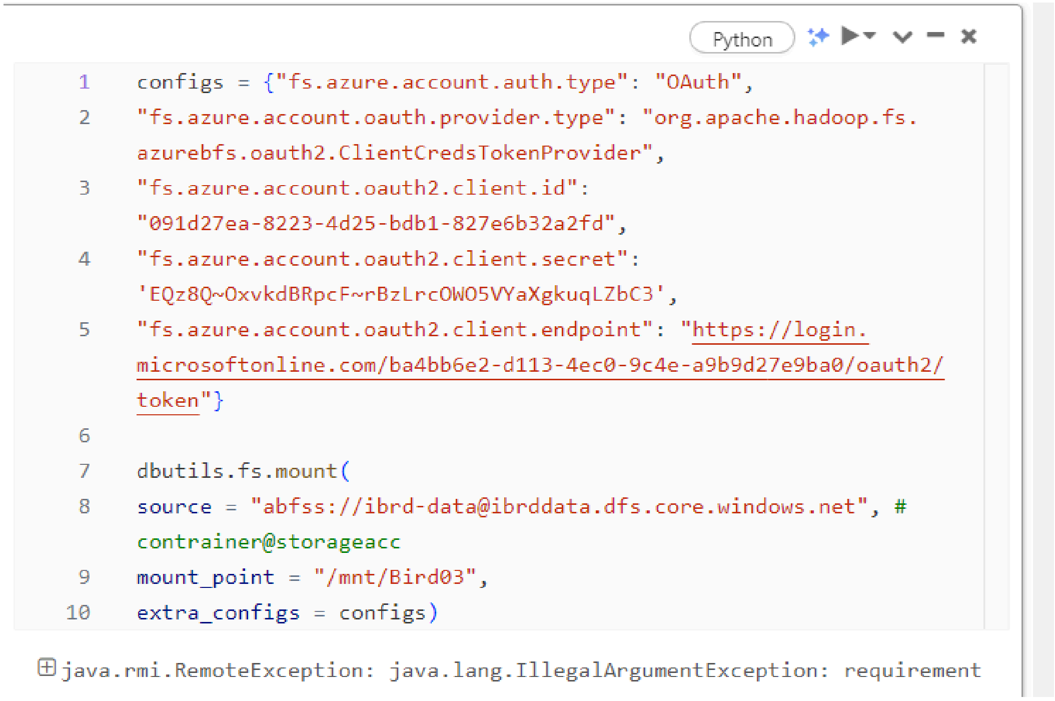
□	Name ↑↓	Type ↑↓
□	irbd2-data-bricks2	Azure Databricks Servi



Now Create Cluster


Filter compute you have access to
Created by
□ Only pinned
Create with Personal Compute
Create compute

State	Name	Policy	Runtime	Active mem...	Active cores	Active DBU ...	Source	Creator	Notebooks
✓	chirag satpute's Cluster	-	13.3	14 GB	4 cores	1.5	UI	chiragsatpute0303...	1

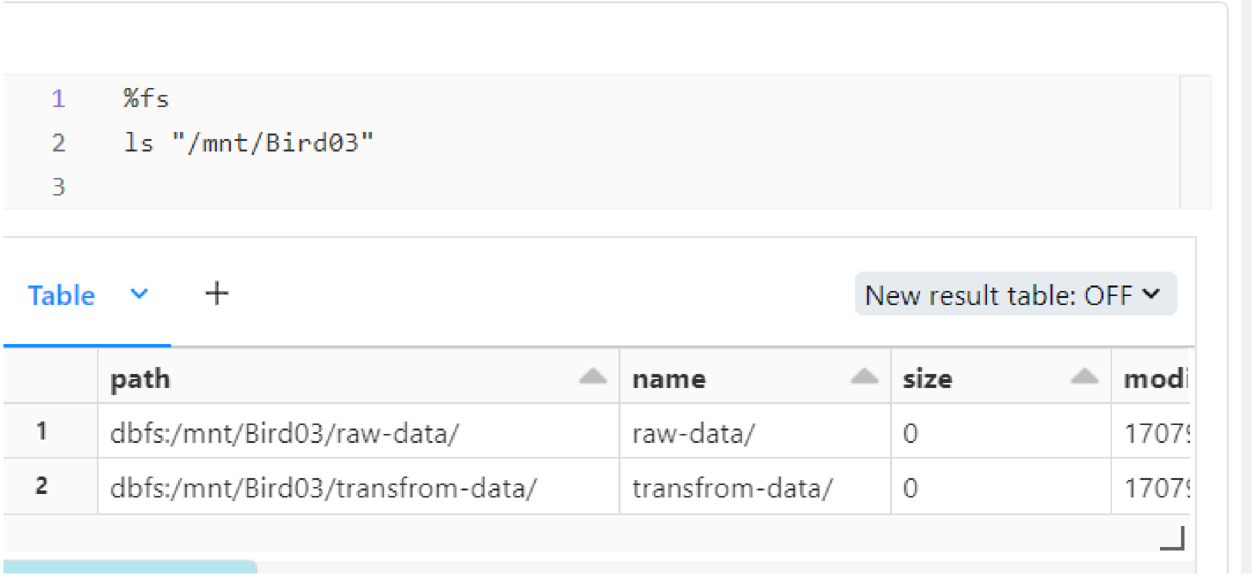


```
configs = {"fs.azure.account.auth.type": "OAuth",
"fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.
azurebfs.oauth2.ClientCredsTokenProvider",
"fs.azure.account.oauth2.client.id":
"091d27ea-8223-4d25-bdb1-827e6b32a2fd",
"fs.azure.account.oauth2.client.secret":
'EQz8Q~OxvkdBRpcF~rBzLrcOWO5VYaXgkuqLZbC3',
"fs.azure.account.oauth2.client.endpoint": "https://login.
microsoftonline.com/ba4bb6e2-d113-4ec0-9c4e-a9b9d27e9ba0/oauth2/
token"}

dbutils.fs.mount(
source = "abfss://ibrd-data@ibrddata.dfs.core.windows.net", #
contrainer@storageacc
mount_point = "/mnt/Bird03",
extra_configs = configs)
```

```
⊞java.rmi.RemoteException: java.lang.IllegalArgumentException: requirement
```

```
configs = {"fs.azure.account.auth.type": "OAuth",
"fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
"fs.azure.account.oauth2.client.id": "091d27ea-8223-4d25-bdb1-827e6b32a2fd",
"fs.azure.account.oauth2.client.secret":
'EQz8Q~OxvkdBRpcF~rBzLrcOWO5VYaXgkuqLZbC3',
"fs.azure.account.oauth2.client.endpoint":
"https://login.microsoftonline.com/ba4bb6e2-d113-4ec0-9c4e-a9b9d27e9ba0/oauth2/token"}

dbutils.fs.mount(
source = "abfss://ibrd-data@ibrddata.dfs.core.windows.net", # contrainer@storageacc
mount_point = "/mnt/Bird03",
extra_configs = configs)
```

Connecting Code data-bricks and data lake storage Gen2
This is the main connection code
There are 3 imp key in your pipeline code :
1)Client ID; 091d27ea-8223-4d25-bdb1-827e6b32a2fd
2) tenant Id:EQz8Q~OxvkdBRpcF~rBzLrcOWO5VYaXgkuqLZbC3
3)Secrect key:ba4bb6e2-d113-4ec0-9c4e-a9b9d27e9ba0a0
This are always required to make the connect with storage location
We mount the files with data-bricks



```
%fs
ls "/mnt/Bird03"

```

	path	name	size	modi
1	dbfs:/mnt/Bird03/raw-data/	raw-data/	0	1707!
2	dbfs:/mnt/Bird03/transfrom-data/	transfrom-data/	0	1707!

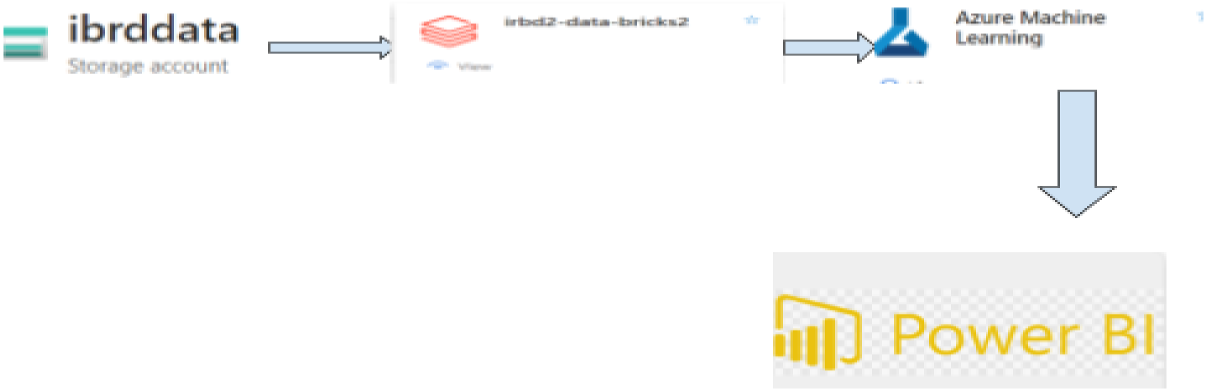
Command to see the storage files which are in the container

```
data=spark.read.format("csv").option("header","true").load("/mnt/Bird03/raw-data/IBRD_Statement_Of_Loans_-_Historical_Data_20240207.csv")
```

Command to load the the CSV data

```
feature_data.write.mode("overwrite").option("header","true").csv("/dbfs:/mnt/BirdMl/azureml/feature_data")
```

This is the code which make you pipeline to store the transform data to your destination transformation file



Flow of your Project

References

Data Source : https://finances.worldbank.org/Loans-and-Credits/IBRD-Statement-Of-Loans-Historical-Data/zucq-nrc3/about_data

Azure : https://www.youtube.com/watch?v=IaA9YNlg5hM&t=3757s

Pyspark: https://data-flair.training/blogs/pyspark-tutorial/

Machine Learning : https://youtu.be/kBz6zCzcPBQ?si=79-RI1btVGKQwTol

Models:https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/boosted-decision-tree-regression?view=azureml-api-2