Template Documentation

MLOps Documentation

2022

Table of Contents

[1. Introduction: 2](#_Toc106709336)

[1.1 Existing Problems/Challenges: 2](#_Toc106709337)

[2. Industrialization Overview and Scope: 3](#_Toc106709338)

[3. Industrialization as a Service – MLOps: 3](#_Toc106709339)

[3.1 Experiment Tracking: 3](#_Toc106709340)

[3.2 Modularization, Parameterization & Optimization: 5](#_Toc106709341)

[3.3 Orchestration 11](#_Toc106709342)

[3.4 Code Versioning and Deployments: 15](#_Toc106709343)

[3.5 Artefact Logging: 25](#_Toc106709344)

[3.6 KPI storage & schema: 25](#_Toc106709345)

[3.7 Drift Detection: 28](#_Toc106709346)

[3.8 Data Quality Measurement: 30](#_Toc106709347)

[3.9 PowerBI Ops Monitoring Dashboard: 30](#_Toc106709348)

## Introduction:

PROJECT as a solution is a Forecasting solution that works towards producing promotional demand forecasts for the Products that The Company sells in that market The level at which the forecasts are made is at the Customer (Distributors of The Company products) and the Product level (GTIN/SKU level).

As part of Industrialization, the existing process is to be stabilized, automated, optimized (from an execution standpoint) and organized to ensure the solution performs stably and is producing these required forecasts within SLAs defined by market planners and brings in transparency to the Forecast Analyst on the data quality, forecast quality and notify in case of issues proactively before the SLAs.

### Existing Problems/Challenges:

Before the Industrialization starts, a 1-week discovery phase is setup where the current DEV team gives a solution walkthrough highlighting the codes, Azure systems, Downstream systems, Challenges, Data Quality problems faced and mode of weekly executions. Based on the discussion that the MLOps team has had with the DEV team during the Discovery sessions, the below list summarizes the problems existing with the DEV solutions:

1. The solution contains of many scripts (~30 scripts) with some of the scripts running in PROD, DEV and local machines and makes the solution execution hard to maintain and cumbersome
2. The codes are not modularized and often debugging becomes a challenge based on these huge scripts
3. Solution is executed manually line by line by the developers every week and if issues are present the DEV team makes changes on the fly. This causes issues in terms of logging the changes and track changes
4. Code versioning does not exist because of which history of changes are not maintained which further causes issues in terms of rolling back changes that did not work
5. Input files consists of a lot of manual files which the FA gets in SharePoint, Data lakes and emails as well. Week on week some of the files that changes are dependent on FA to send through emails. Solution automations are not in place and there is a vast scope to automate the solution
6. FA inputs are required to check the values at some intermediate and final steps before the data is sent back to the downstream systems. This is also a scope for the automation piece mentioned above
7. No monitoring is done on the Forecast Accuracies/BIAS along with other KPIs which gives a sense of how well/poorly the models are performing. Market planners come back with issues in forecasts and this process can be made proactive rather than reactive
8. Organization of the files, scripts, logs and outputs are not arranged well, and many other market’s solutions are in the same Azure Resource group, and it becomes hard to track the solution scripts that are pertaining to a single market/generation of the model
9. Reproducing a previous run is nearly impossible as the relevant artefacts of the run are not logged, and lessons learnt from previous run are not being logged
10. There is no deployment process in place and “production” runs are taking place between DEV, PROD and local machines. There is no code auditing or code reviews in place to ensure the code maintainability and readability is present.

## Industrialization Overview and Scope:

As part of Industrialization, best practices from MLOps are introduced in the solution end-to-end to help stabilize the solutions and systems, reduce manual intervention wherever possible, bring in clear traceability and monitoring, maintainability and reduce the level of bugs in current solutions, follow proper deployment guidelines, automation and orchestration.

MLOps as best practices guidelines are introduced here to ensure a standard framework of practice is established and bring in reusability across markets as well. The following best practices were introduced in PROJECT Country1 to address the challenges mentioned in the previous section which we will be diving deep in upcoming sections:

1. Experiment Tracking
2. Modularization, Parameterization & Optimization
3. Orchestration
4. Code Versioning and Deployments
5. Artefact Logging
6. KPI storage & schema
7. PowerBI Ops Monitoring Dashboard
8. Drift Detection

These are iteratively improving systems and the quality of stability and optimization will keep improving over time with more exceptions being handled.

## Industrialization as a Service – MLOps:

### Experiment Tracking:

Experiment Tracking is achieved in the solutions by leveraging [mlflow](https://docs.microsoft.com/en-us/azure/databricks/applications/mlflow/tracking), an open-source tool for MLOps developed by Apache & Databricks creators. It integrates well with the Databricks eco-system and GUI despite the cloud service provider on top of which Databricks was instantiated (in this case Azure).

Experiment tracking in PROJECT Country1 is brought in to bring in the following visibility:

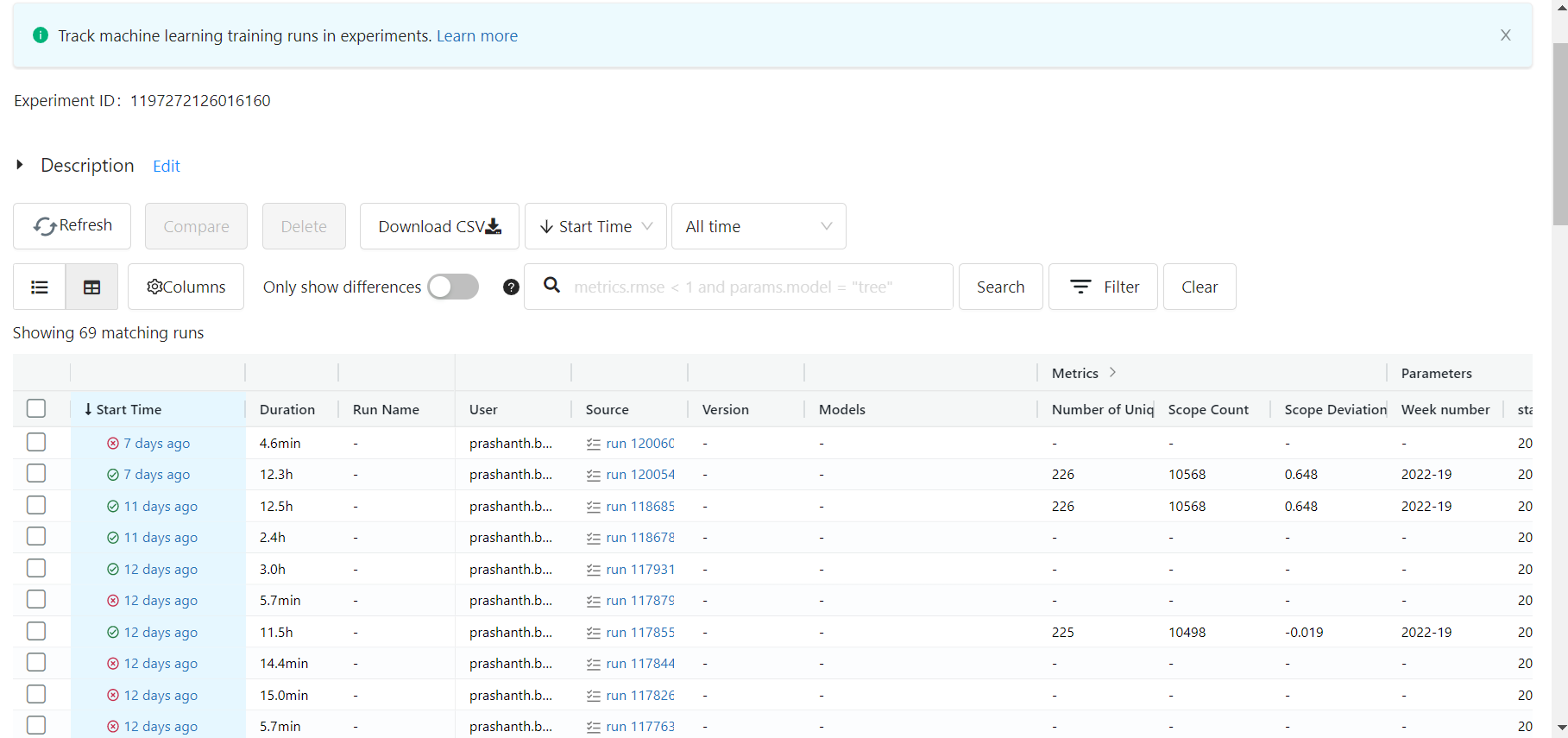
1. Log of Weekly runs
2. Metrics captured as part of weekly run sanity check
3. Duration & run trigger details

These things are achieved by adding a few lines of codes to the notebooks that are part of the weekly runs as illustrated in the below screenshot.



By calling mlflow.set\_experiment, the run is given an experiment name which is synonymous to run name in this case. Mlflow.start\_run() command gives a clear indication to the tool that the lines of codes that proceeds is part of the experiment.

Mlflow.log\_param() and mlflw.log\_metric() classes are used to track relevant parameters and metrics important to that run which will be used to judge the quality of run and other common issues that are faced during the run. These metrics will provide the solution with a first step of checks and balances in the automation of the solution. The logged KPIs can be viewed in the Databricks GUI under the experiments tab in the sidebar, which is depicted in the illustration below.



As we can see above all the relevant sanity KPIs and details pertaining to the run with a status of execution (success/fail) that are shown in the tab.

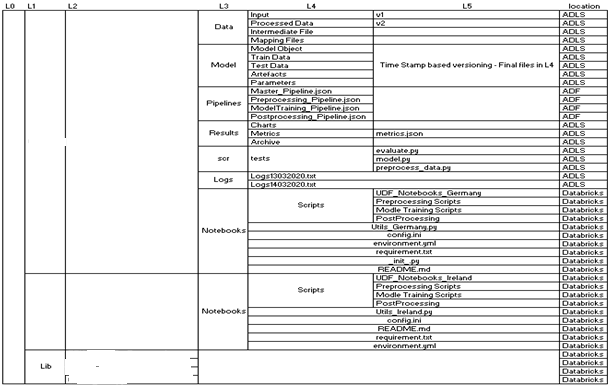
### Modularization, Parameterization & Optimization:

To leverage the best performance, maintainability, cost reduction and stability of ML Models/systems, it is imperative to bring in modularity in codes, files and folders with proper documentation. This is the first level of best practice that MLOps tends to enforce to enhance the quality of the solution.

Modularization covers a modular folder structure, modularization of codes (long never ending notebooks to modular classes and functions using [OOPs concepts](https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP). This promotes reusability across solutions and makes it very easy for new developers to understand the solution better and make continuous changes on top of the existing solution.

#### Modular Folder Structure

Developers attempt to organize all their files into new and unrelated folders and end up creating a complex folder structure which can lead to losing track of folders and files. As part of Industrialization, we have followed the below folder structure to keep it flexible, intuitive, and scalable. Developers can easily add markets, generation and stages of the processes and maintain it easily which is understandable by other developers as well (across markets). The folder structure was maintained well in ADLS to keep the input flat files, output and intermediate files in their respective folders with a weekly archive in it. Same was followed for all the Databricks notebooks to keep clarity on the purpose of the notebook on which it was created.



The entire PROJECT solution can be maintained in 4 levels of hierarchical folder management (From L0 to L3 as depicted in the Illustration above). This ensures PROJECT as a solution is following a well-structured and easy to maintain file system. Levels L4 and onwards shows how within each Generation of model (in a particular market) can store files, notebooks and outputs.

##### ADLS – Storage

**Storage Type:** Data Lake Storage Gen1

**Name:** sample\_name

**Location:** West Europe

**ADL URI:** URL

**Directory:** /The Company/MLOps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sl No. | Folder Name | Level | Description | Examples |
| 1 | Country\_name | Country | This folder contains all the data at Country level | Ex: Country1, Country2, Country3, Lib |
| 2 | Lib | Core Library | This folder contains Files those can be used by All Markets | Ex: Calendar, Holiday, etc |
| 3 | Country\_name\_project | Country and Gen | This folder contains all the data at Country level and Gen level | Ex: Country1\_PROJECT\_Gen2, Country1\_PROJECT\_Gen3 |
| 4 | Data | Country level Data | This folder contains all the data used for PROJECT Country1 | Ex: Input, Intermediate and Output Data Sources for that PROJECT project |
| 5 | -Input Data | Project Level Data | This folder contains all the input data sources those are use in the PROJECT Country1 Project | Ex: Hard\_Transition.xlsx,  Delist.xlsx, Promo Basis, sales data from SALESSOURCE |
| 6 | -Intermediate Data | Project Level Data | This folder contains all the intermediate files created during the PROJECT Country1 weekly process | Ex: shipment\_data1\_dataframe\_202216.parquet, Parallel\_result\_Foods\_Promotion.csv |
| 7 | -Mapping Data | Project Level Data | This folder contains all the mapping files those are used for in the PROJECT Country1 Project | Ex: apg\_mapping.csv, SalesSource\_mapping, prodMapping, etc |
| 8 | -Processed Data | Project Level Data | This folder contains all the output files of the weekly PROJECT Country1 run | Ex: ful\_forecast\_df \_202216.parquet, promo\_forecast\_df \_202216.parquet |
| 9 | Logs | Project Level Data | This Folder contains the data related to Power BI Backend | Ex: Power BI Backend |
| 10 | -Power BI DB | Project Level Data | This folder contains all the Data Sources for Power Bi Dashboard | Ex: Data, DQM and DS |
| 11 | --Data | Project Level Data | This folder contains all the Data file for Power Bi Dashboard for KPI metrics | Ex: Region\_bpc.sell\_in\_proper\_new, dash\_hhc.sell\_in\_data, etc |
| 12 | --DQM | Project Level Data | This folder contains all the Data Sources of DQM Power Bi Dashboard | Ex: completeness\_20221504.csv, timeliness\_20221504.csv, uniqueness\_20221504.csv, etc |
| 13 | --DS | Project Level Data | This folder contains all the Data files for data Science metrics in Power Bi Dashboard | Ex: Drift, Importance, Results |
| 14 | Model | Project Level Data | This folder contains all the data related to Models used for PROJECT Country1 | Ex: Folders like Artefacts, Train Data, etc |
| 15 | -Artefacts | Project Level Data | This folder contains all the Data files for data Science metrics in Power Bi Dashboard | Ex: Weekly Folder like 2022-14 that contain Train and test, test-based data |
| 16 | Results | Project Level Data | This folder contains all the Output Files related to weekly PROJECT Country1 run | Ex: Archives |
| 17 | -Archives | Project Level Data | This folder contains all the Forecasting results in respective Weekly Folders | Ex: Weekly Folder like 2022-14 |

##### 3.2.1.2 Databricks – DEV

Resource Group: Sample\_name

Databricks Workspace Name: Sample\_name

Location: West Europe

URL:

Cluster Used: PROJECT\_Country1\_MLOps\_clus1 (Standard\_DS5\_v2 56 GB Memory, 16 Cores Min 2 Max 8 workers)

Link for PROJECT Country1 Gen3:

Databrick Workspace> PROJECT > Country1 > Country1\_PROJECT\_Gen3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sl No. | File/Folder Name | Description | Example | Link |
| 1 | Config | This folder consists of all file paths for pre-processing and post-processing scripts. |  | /PROJECT/Country1/Country1\_PROJECT\_Gen3/config |
| 2 | Scripts | This folder consists of all pre-processing, model training and post-processing scripts |  |  |
| 3 | -Preprocessing\_Scripts | This folder includes pre-processing scripts |  |  |
| 4 | --CRM\_Preprocessing | This script is used for pre-processing CRM promotion data received from BDL location |  |  |
| 5 | --Datalake\_Extraction | This script us used for performing pre-processing on sales data/SALESSOURCE and promotion data. | EX: Datalake\_Extraction |  |
| 6 | -Model\_Training\_Scripts | This folder contains ML Preprocessing class and parallel ML run class for all categories along with their required config file. | Ex: ML\_Run\_Class, Preprocess\_Class and config |  |
| 7 | -Postprocessing Scripts | This folder consists of scripts which are part of postprocessing |  |  |
| 8 | -- Baseline\_Dissaggregation  \_new\_logic | This file helps in getting Baseline disaggregation | Ex: Baseline\_Dissaggregation  \_new\_logic\_main |  |
| 9 | -- Country1\_Post\_processing | This file provides parallel forecast | Ex: country1\_post\_processing  \_main |  |
| 10 | -- ERP\_Total\_Volume  \_Forecast(Dev) | This file provides total volume forecast | Ex; ERP\_Total\_Volume  \_Forecast\_main |  |
| 11 | -- ERP\_Total\_Volume  \_Forecast (Prod) | This file provides the ERP raw file. | Ex; ERP\_total\_volume  \_forcast\_main |  |
| 12 | -- Local\_execution\_scripts | These are the scripts where provides baseline comparison and finalized baseline apg | Ex: Variablity\_HC\_BPC, finalized\_baseline |  |
| 13 | UDF\_Notebooks\_Country1 | This folder contains the notebook which have core function for PROJECT Country1 | Ex: udf\_pyspark\_final |  |

#### 3.2.2 Parameterization and Optimization

There are no manual changes to be done in the code to change the file paths. In the current code setup variables are assigned dynamic values to fetch current week and previous week numbers.

##### 3.2.2.1 Optimizing Certain code blocks

In preprocessing for CRM all the pandas data frame is converted into PySpark. In postprocessing for the scripts which runs using local python files converted into notebooks and runs are triggered using ADF.

##### 3.2.2.2 Scripts Reduction

ML – ML Preprocessing & ML model training and prediction parts for all categories used to complete in 28 scripts in total for 6 categories, each category’s total scripts ranging from 3 to 6. With our modularized code end to end ML processes from preprocessing to model prediction completes using just 5 scripts, out of these 2 are trigger files, same scripts are utilized for all categories as they are parametrized. Different functions inside the modularized code takes care of different steps like Actuals processing, Misc processing, Promotion/Shipment processing etc. which earlier was being done using a separate notebook in each category.

##### Classes/ OOPs

In ML, four classes are bring used in total; They are as follows:

1. MLpreprocess\_base : Base class in the first script (config notebook), this takes 4 parameters - current week, previous week, category name and parameters. This class loads all the csv/excel files that are required for a category’s process and contains basic functions common to all processes in further classes.
2. MLpreprocess: This class inherits the base class, this takes 4 parameters required by base class - current week, previous week, category name and parameters. The functions related to loading files in base class are classed inside this class. Apart from loading relevant files, this class also contains functions related to multiple category preprocessing that happens before scope creation namely – Actuals processing, Shipment Processing, Promotion Processing, get SALESSOURCE data, update Actuals table etc. When the final function in this class is called, we get a dictionary of dataframes which will be used as an input in the ML class.
3. ML\_Run : ML run class also inherits the base class as the common functions are utilized in ML processes also, it takes in 4 parameters - current week, previous week, category name and Preprocessing output dictionary. It contains functions to create scopes, prepare train and test data, forecasting loop for multivariate and univariate models, future replacements, and other process on predicted data. In trigger file, 2 main functions are called from this class – pre\_ml\_processing(gives out dictionary of dataframes and list required in forecasting like scopes) and ml\_processing(gives out final output dictionary of predicted values after all treatments). The sub-processes inside forecasting loops are also split into multiple functions.
4. ML\_New\_list : Inherits the ML\_Run class, takes the same input Parameters as ML\_Run - - current week, previous week, category name and Preprocessing output dictionary. It has functions related to the newlist process namely getting proxy hierarchies, replacement proxies etc. It utilizes the forecasting loop function inherited from ML\_Run to make predictions for newlist hierarchies. Newlist is triggered through a different trigger file which calls pre\_ml\_processing and new\_list\_process functions from the NL\_New\_list class.

##### 3.2.2.4 Config

For PROJECT Country1 solution, config file stores the input file paths. For ML, the config notebook contains base class, file paths, category wise required feature names and table names. The file paths, feature names and table names are saved as nested dictionaries with category names as keys which are then used as input in successive functions in preprocessing, ML training. Parameters is one such nested dictionary which contains output and input table names and is used as in input in Preprocess class. Training\_cols is another such nested dictionary which is used while creating training data as different categories have different columns for modelling.

### 3.3 Orchestration

In the Industrialized PROJECT Country1 solution, Orchestration aims to bring in automation and scheduled triggers to executing the scripts. This is achieved by using the Azure Data Factory (ADF) pipelines which ensures that the runs execute in an automated fashion as per a defined schedule.

**DEVELOPMENT STAGE**

Name of Azure Data Factory: Sample\_name

Location: West Europe

Feature Branch Used: Feature/PROJECT\_Country1\_MLOps\_ADF\_Push branch

Number of Pipelines: 6

Folder Name: PROJECT\_COUNTRY1\_MLOPS

**QA STAGE**

Name of Azure Data Factory: Sample\_name

Location: West Europe

Feature Branch Used: Feature/PROJECT\_Country1\_MLOps\_ADF\_Push branch

Number of Pipelines: 6

Folder Name: PROJECT\_COUNTRY1\_MLOPS

**PROD STAGE**

Name of Azure Data Factory: Sample\_name

Location: West Europe

Number of Pipelines: 6

Folder Name: PROJECT\_COUNTRY1\_MLOPS

|  |  |  |
| --- | --- | --- |
| Sl No. | Pipeline Name | Production URL |
| 1 | PL\_PROJECT\_COUNTRY1\_CDI\_UPLOAD | [Link](https://adf.azure.com/en/authoring/pipeline/PL_FEU_GERMANY_CDI_UPLOAD?factory=%2Fsubscriptions%2Fe2b0e829-5210-40ae-b73f-20805aa01351%2FresourceGroups%2Fbnlwe-da01-p-901372-rg%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fbnlwe-da01-p-901372-adf-01) |
| 2 | PL\_PROJECT\_COUNTRY1\_GEN3\_MASTER | [Link](https://adf.azure.com/en/authoring/pipeline/PL_FEU_GERMANY_GEN3_MASTER?factory=%2Fsubscriptions%2Fe2b0e829-5210-40ae-b73f-20805aa01351%2FresourceGroups%2Fbnlwe-da01-p-901372-rg%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fbnlwe-da01-p-901372-adf-01) |
| 3 | PL\_PROJECT\_COUNTRY1\_GEN3\_ML\_POST\_PROCESSING | [Link](https://adf.azure.com/en/authoring/pipeline/PL_FEU_GERMANY_GEN3_ML_POST_PROCESSING?factory=%2Fsubscriptions%2Fe2b0e829-5210-40ae-b73f-20805aa01351%2FresourceGroups%2Fbnlwe-da01-p-901372-rg%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fbnlwe-da01-p-901372-adf-01) |
| 4 | PL\_PROJECT\_COUNTRY1\_GEN3\_ML\_TRAINING | [Link](https://adf.azure.com/en/authoring/pipeline/PL_FEU_GERMANY_GEN3_ML_TRAINING?factory=%2Fsubscriptions%2Fe2b0e829-5210-40ae-b73f-20805aa01351%2FresourceGroups%2Fbnlwe-da01-p-901372-rg%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fbnlwe-da01-p-901372-adf-01) |
| 5 | PL\_PROJECT\_COUNTRY1\_GEN3\_PREPROCESSING | [Link](https://adf.azure.com/en/authoring/pipeline/PL_FEU_GERMANY_GEN3_PREPROCESSING?factory=%2Fsubscriptions%2Fe2b0e829-5210-40ae-b73f-20805aa01351%2FresourceGroups%2Fbnlwe-da01-p-901372-rg%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fbnlwe-da01-p-901372-adf-01) |
| 6 | PL\_PROJECT\_COUNTRY1\_POSTPROCESSING | [Link](https://adf.azure.com/en/authoring/pipeline/PL_FEU_GERMANY_POSTPROCESSING?factory=%2Fsubscriptions%2Fe2b0e829-5210-40ae-b73f-20805aa01351%2FresourceGroups%2Fbnlwe-da01-p-901372-rg%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fbnlwe-da01-p-901372-adf-01) |

#### 3.3.1 PL\_PROJECT\_COUNTRY1\_CDI\_UPLOAD:

Description: Pipeline triggered on every Tuesday. This pipeline uses previous week results so there is no ML training and preprocessing needed for Tuesday run.

There are five scripts in total and one config file which is used for CDI upload. Auto upload list is shared by Mahendra every week. CDF file is used for fs disaggregation. Incremental forecast data and CDI data are also used as input for CDI upload. There is no CRM output, only ERP data is uploaded every Tuesday. Uploads should be completed by 6.30pm.

Total run time for CDI upload on Tuesday takes approximately 1 hour.

Trigger: Tuesday 15:00 UTC

Timeline

Description automatically generated

#### 3.3.2 PL\_PROJECT\_COUNTRY1\_GEN3\_MASTER:

Description: This is the master pipeline which triggers preprocessing, ML training and ML post processing.

Trigger: Wednesday 18.15pm UTC

Graphical user interface

Description automatically generated with medium confidence

#### 3.3.3 PL\_PROJECT\_COUNTRY1\_GEN3\_ML\_POST\_PROCESSING:

Description: This pipeline has for each activity which runs parallelly for all the categories. Category names are provided as input variables to the notebook.

There are 6 categories TEA, Foods, HC, BPC, IC and FS. The maximum run time for category post processing is 15minutes. Each category total volume and baseline results are calculated using ML outputs.

Trigger: Run within Master Pipeline

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

#### 3.3.4 PL\_PROJECT\_COUNTRY1\_GEN3\_ML\_TRAINING:

Description: For each activity for ML Run as well as Newlist Processes, both are triggered parallelly. ML Run executes for 6 categories – BPC, Foods, IC, HC, FS and Tea while Newlist is run for 4 categories BPC, Foods, IC, HC. When both ML Run and Newlist processes are complete Drift calculation is triggered which calculates Model wise drift for all categories and saved the output in ADLS.

Trigger: Run within the Master Pipeline

Graphical user interface, text, application, chat or text message

Description automatically generated

#### 3.3.5 PL\_PROJECT\_COUNTRY1\_GEN3\_PREPROCESSING:

Description: Preprocessing has 3 scripts. In the first script source data which is used for CRM preprocessing is saved in hive table and Hard Transition received from mail is filtered as per the requirement and saved in adls location.

In CRM preprocessing there are two outputs, one is shipment data and other is promotion data.

In Datalake extraction actuals and transition are processed and saved in hive table which is used in ML training.

Approximate run time for preprocessing takes around 20 minutes.

Trigger: Run within the Master Pipeline

A picture containing diagram

Description automatically generated

#### 3.3.6 PL\_PROJECT\_COUNTRY1\_POSTPROCESSING:

Description: This pipeline is triggered every Thursday. All the post processing steps are linked, and final output data is updated for CRM and ERP uploads.

Postprocessing requires Manual files from HANA system. Total volume and baseline results calculated from category postprocessing are collated in post processing and forecast results are uploaded for both CRM and ERP on Thursday. Two files are uploaded for CRM and one file for ERP. Intermediate data is stored in hive tables.

Trigger: Thursday 10:00 UTC

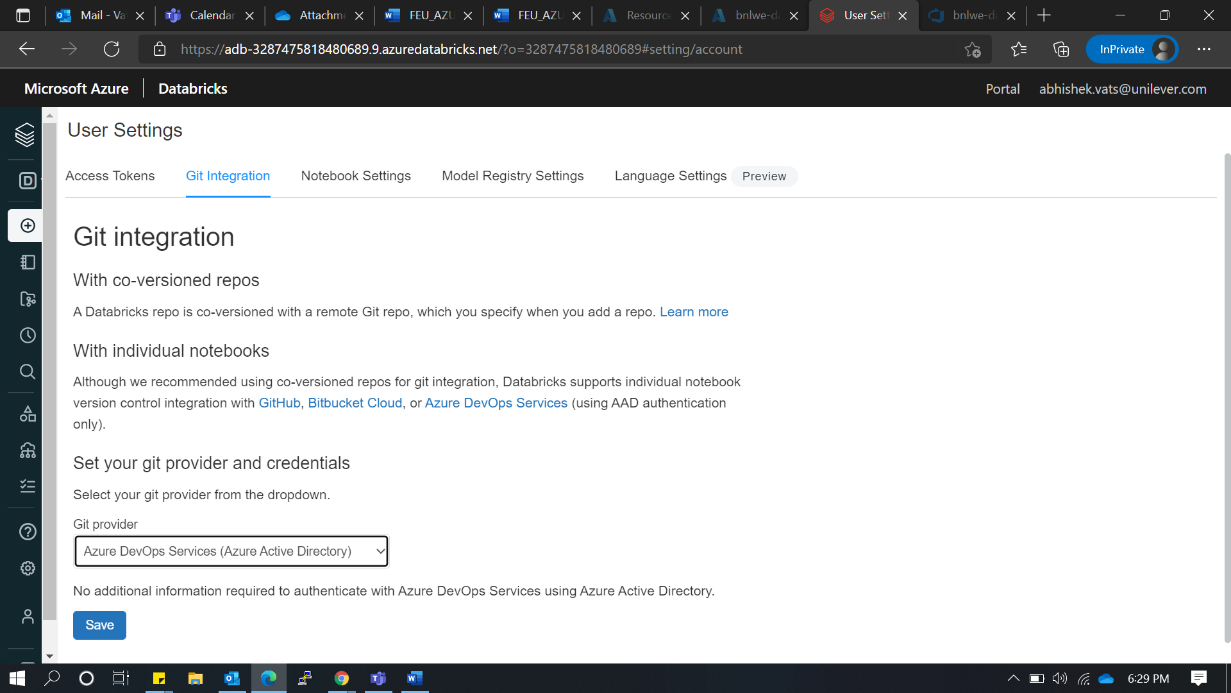
Graphical user interface

Description automatically generated

### 3.4 Code Versioning and Deployments:

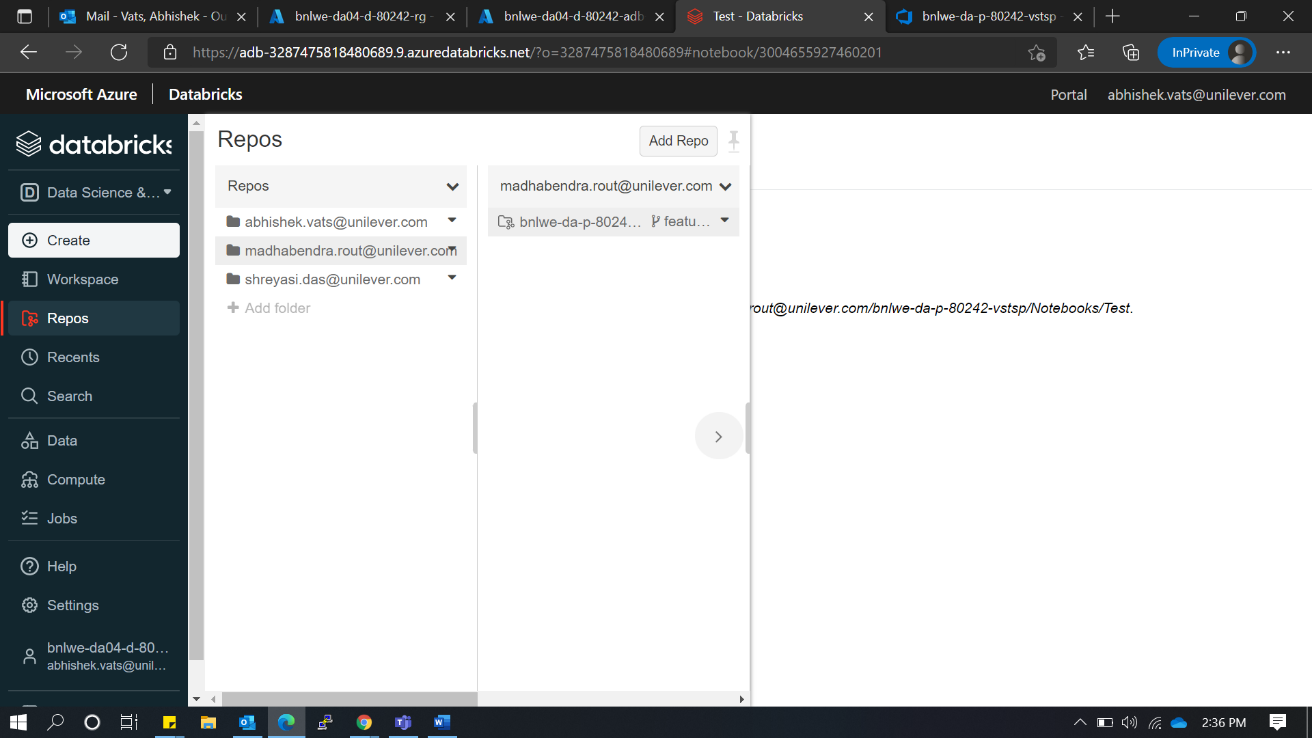
#### 3.4.1 Repository Integration with Databricks:

* **Navigate to your ADB workspace**
* Integrate Git service using **User>User Setting>Git integration menu**
* You do not need additional Token for Azure Repos Integration

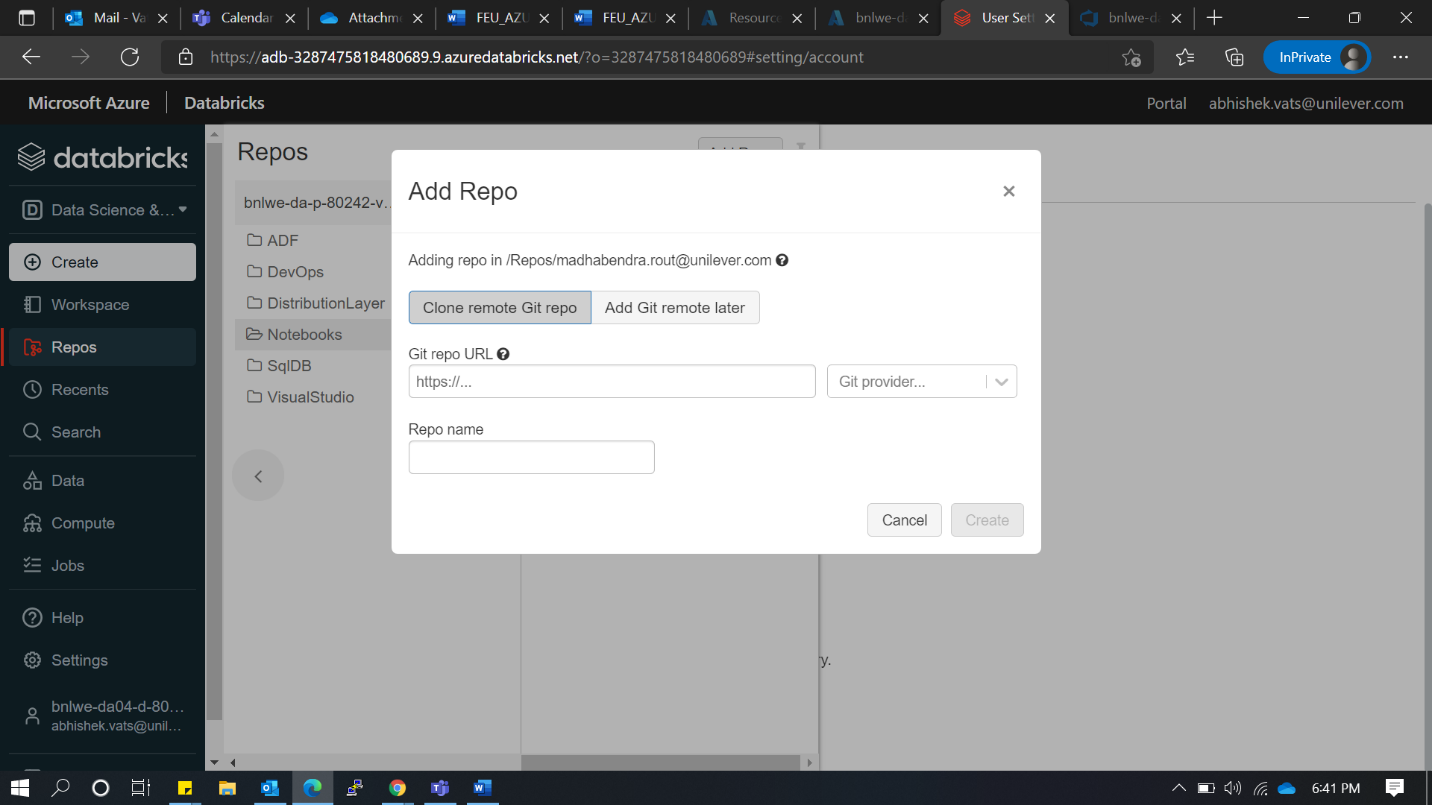


#### 3.4.2 Repository Creation:

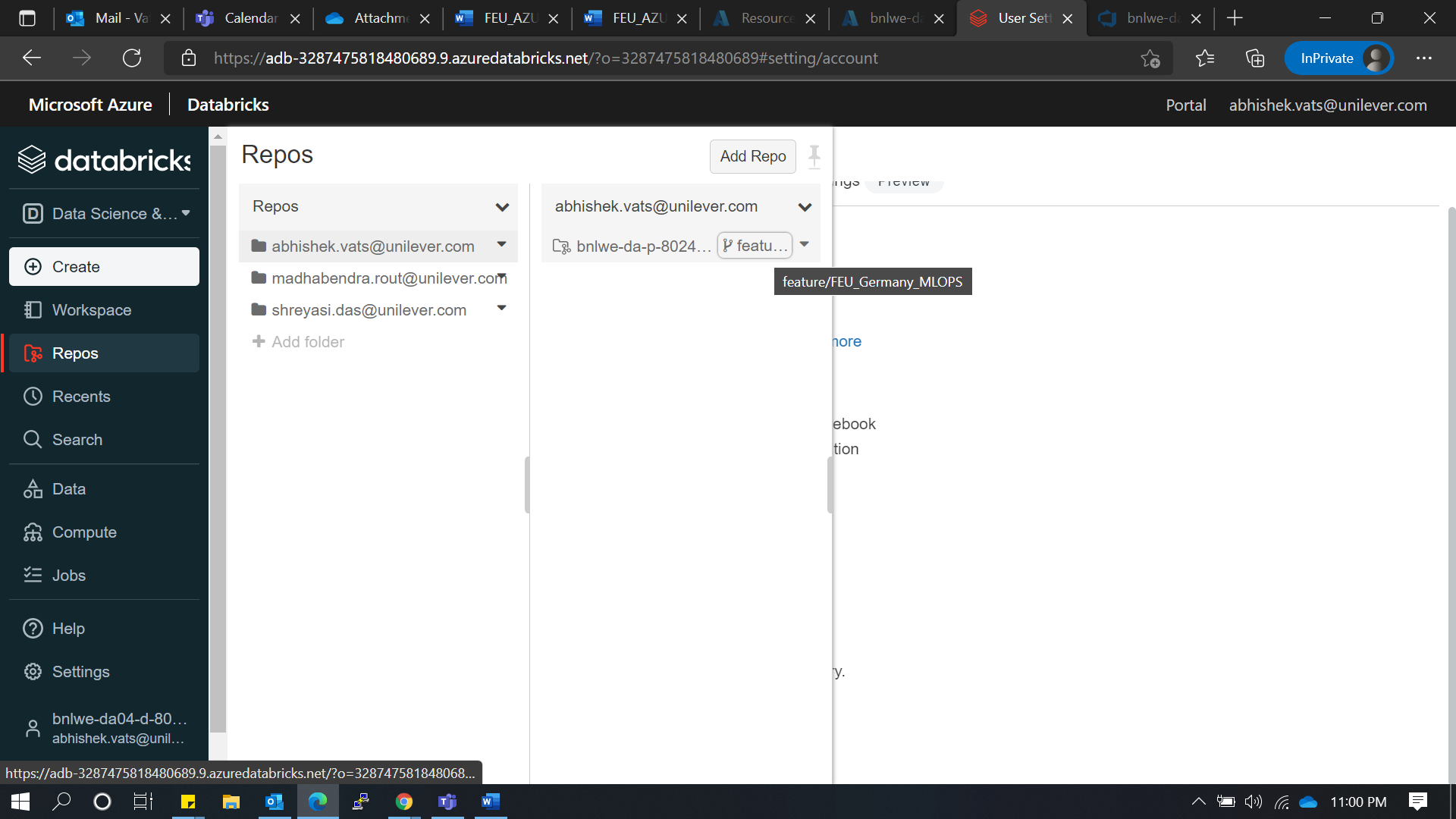
* Click on Repos>Add repo

**

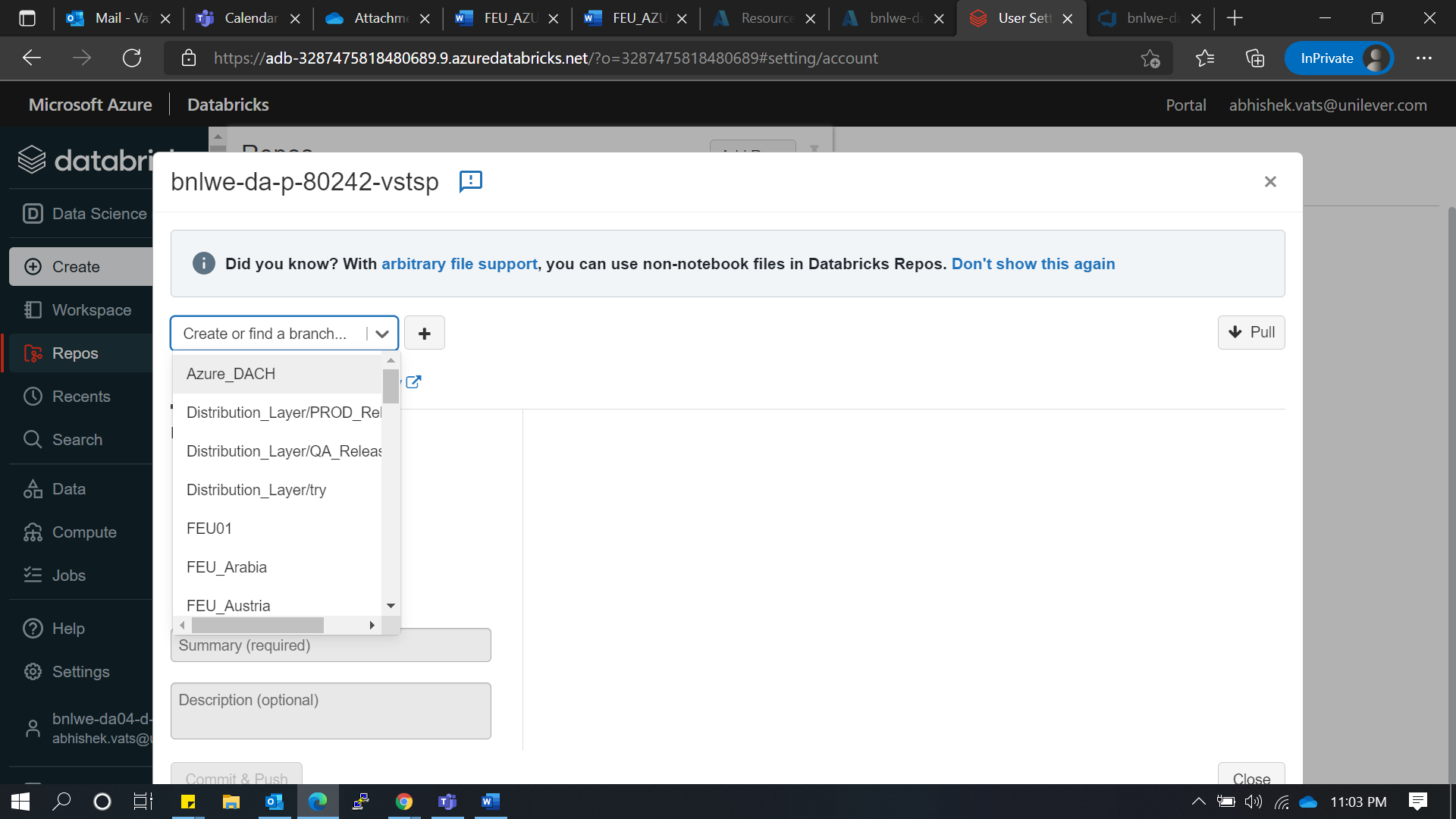
* Fill details
  + git clone URL
  + Git Provider
  + Repo name



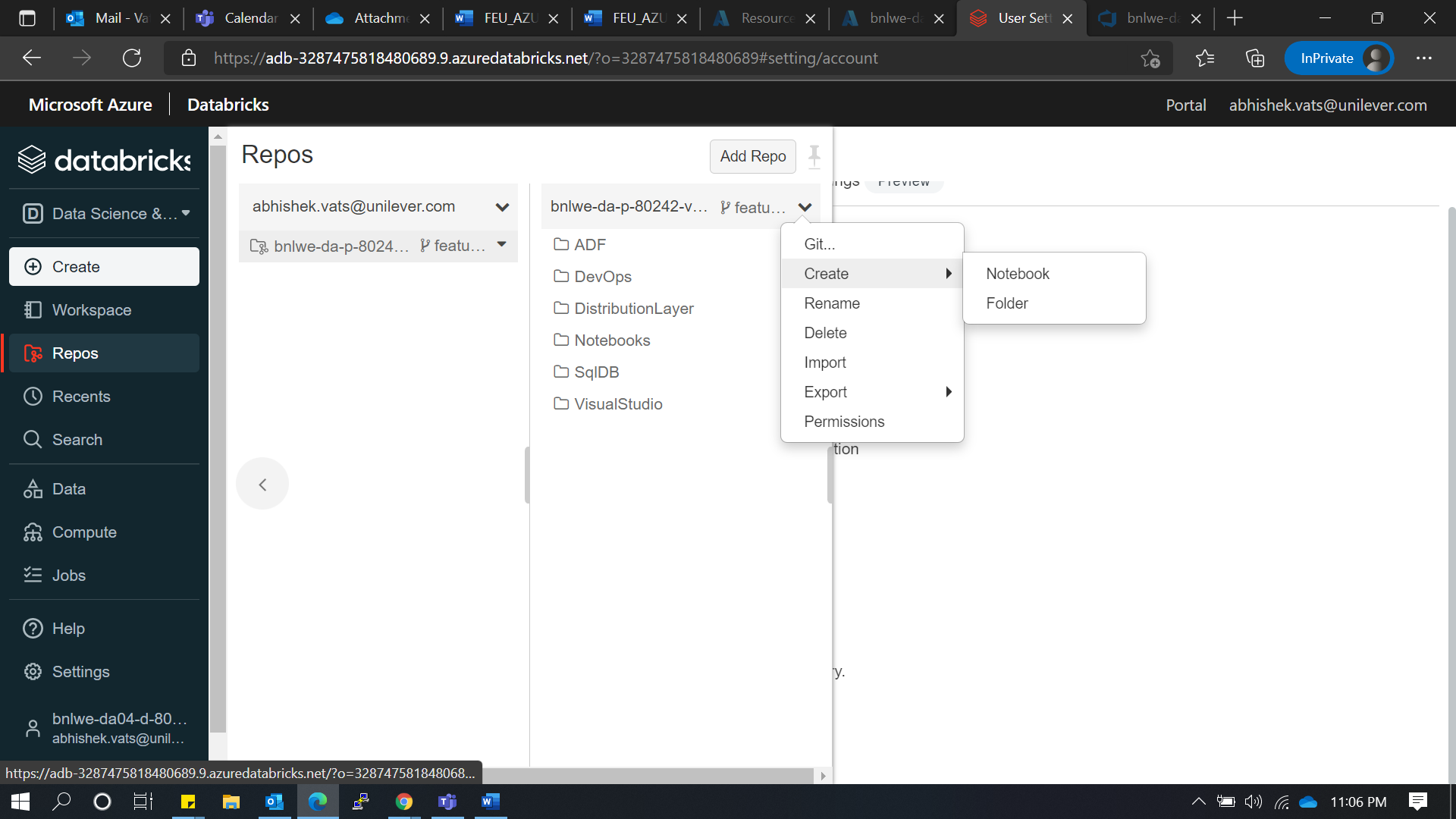
* Change the branch using branch option in your repo folder



* You can change or create a new branch using the highlighted option

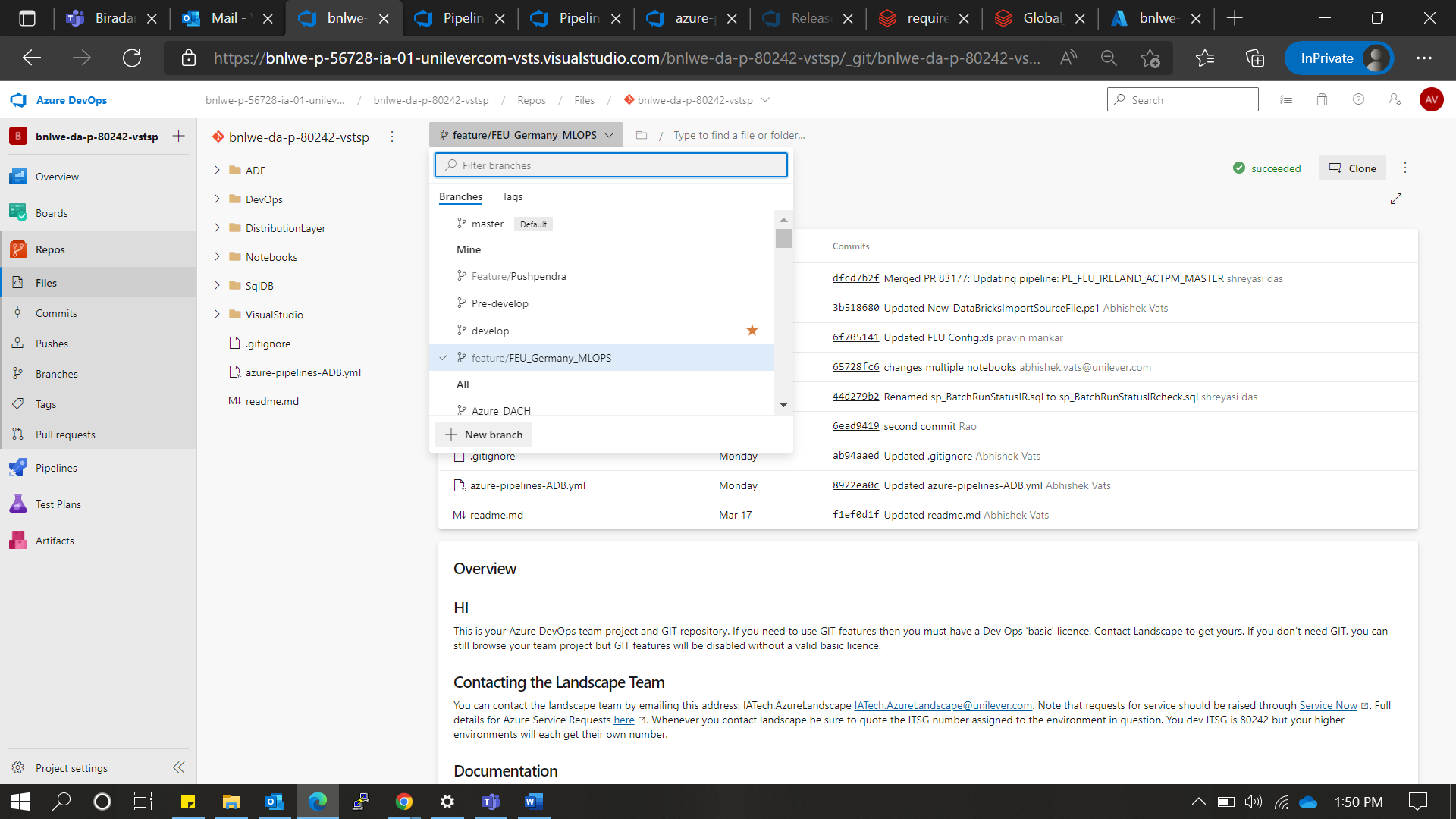


* For adding new folder/Notebook use create option from Repos option or use Import

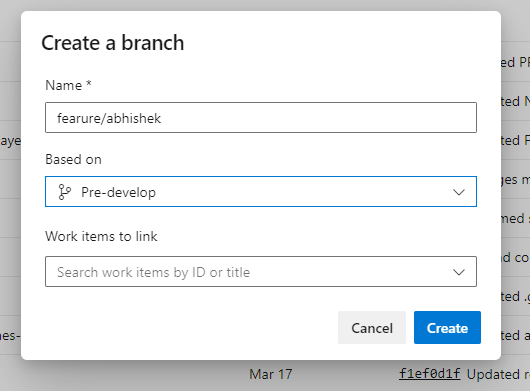


#### 3.4.3 Branch Creation:

* Create new branch

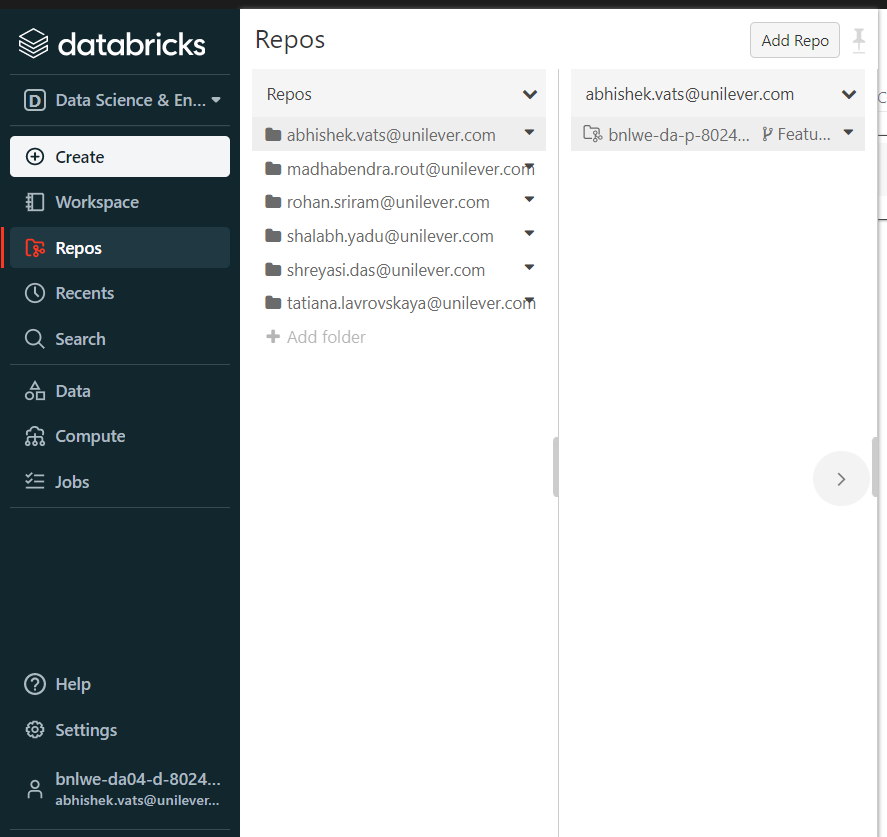


* It should based on Pre-develop

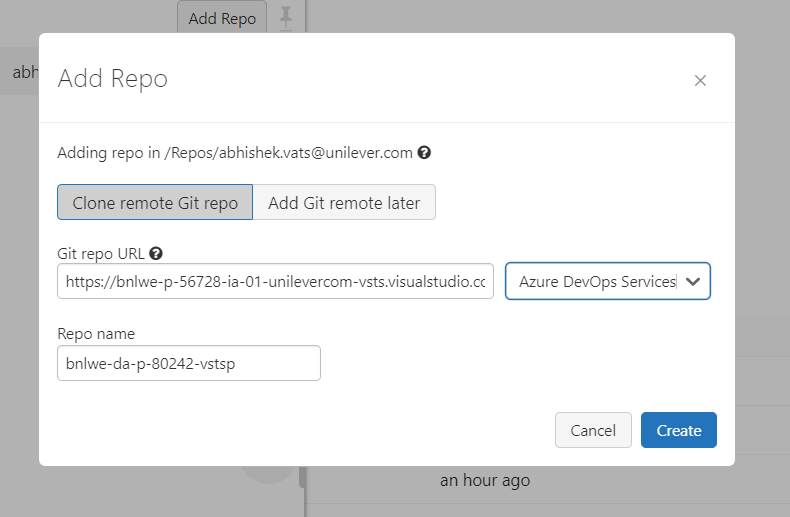


#### 3.4.4 Add repo in Databricks:

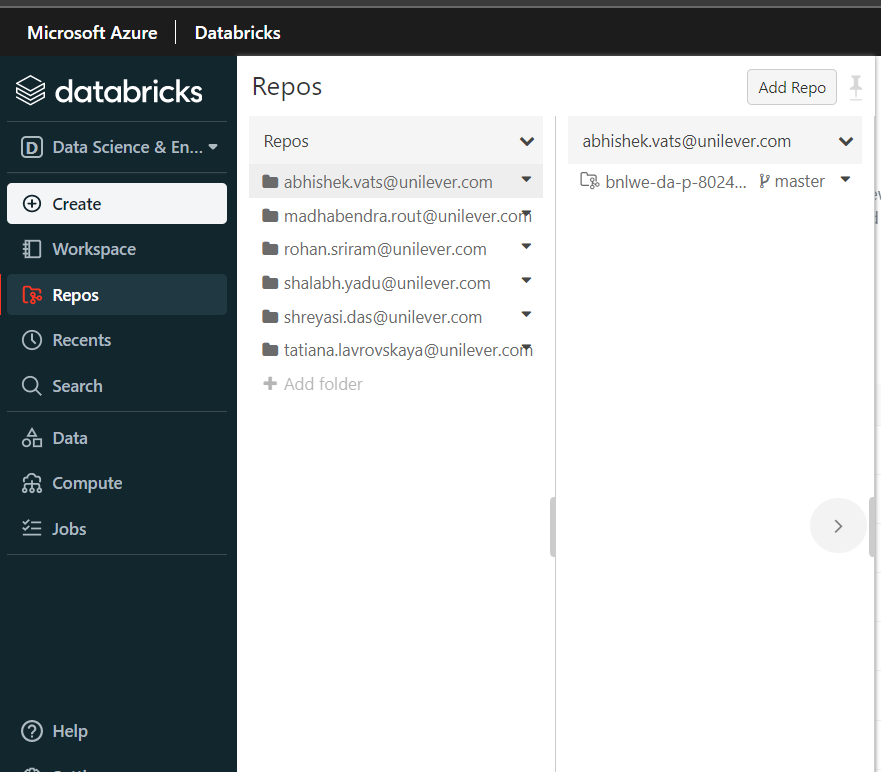
* Now go to Repos section in Azure databricks



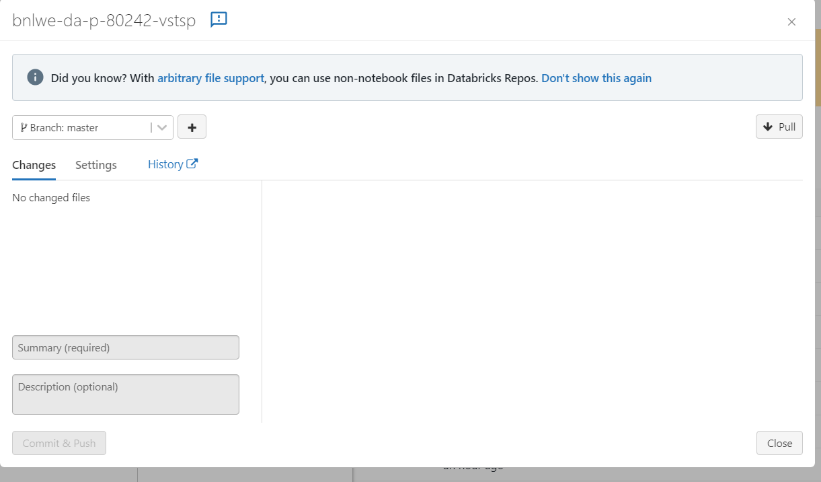
* Click in add repo



* Add your repo clone URL and provider as Azure DevOps services
* Now we get the azure repo in ADB

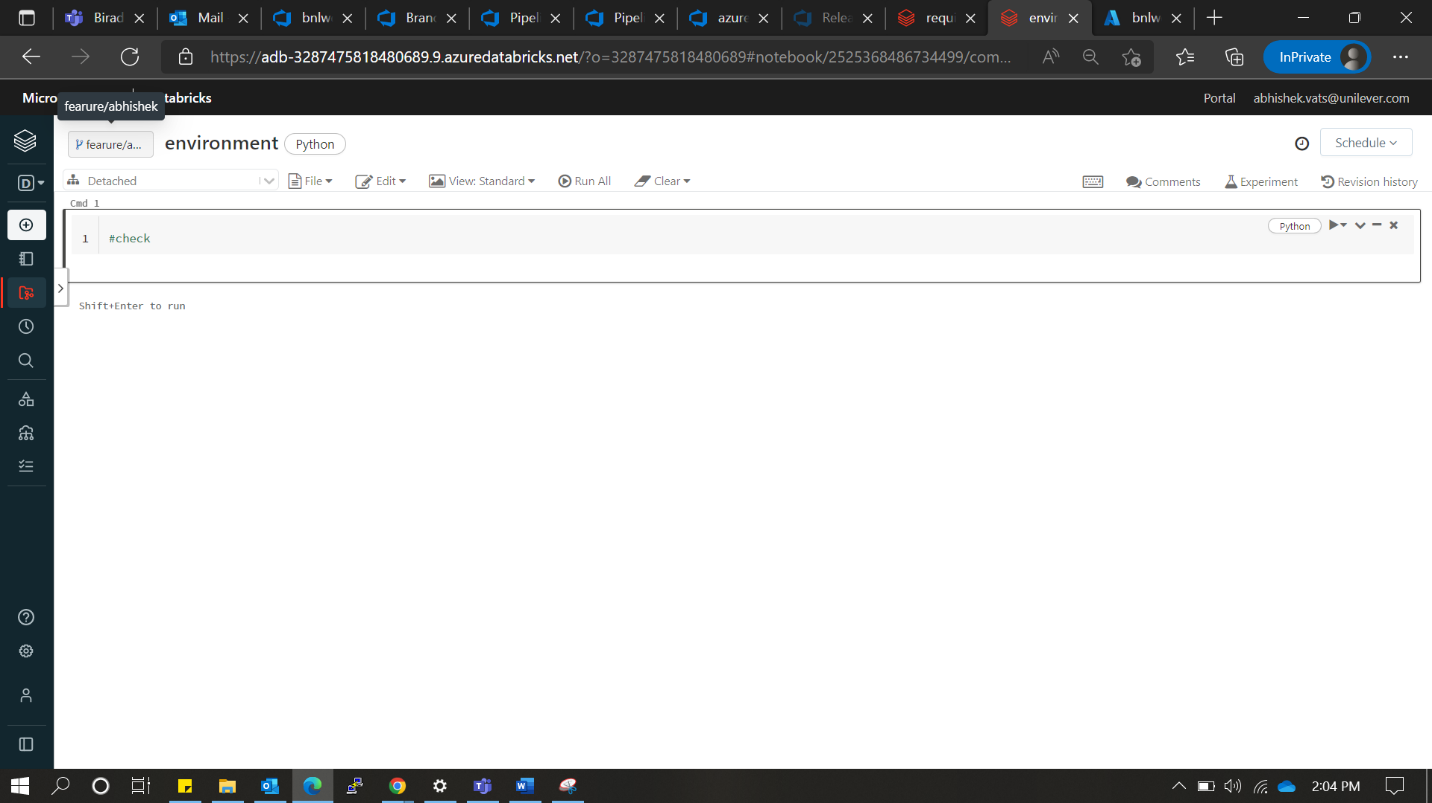


* using branch option, you can change ADB working branch
* **NOTE**: Always choose your feature branch to work

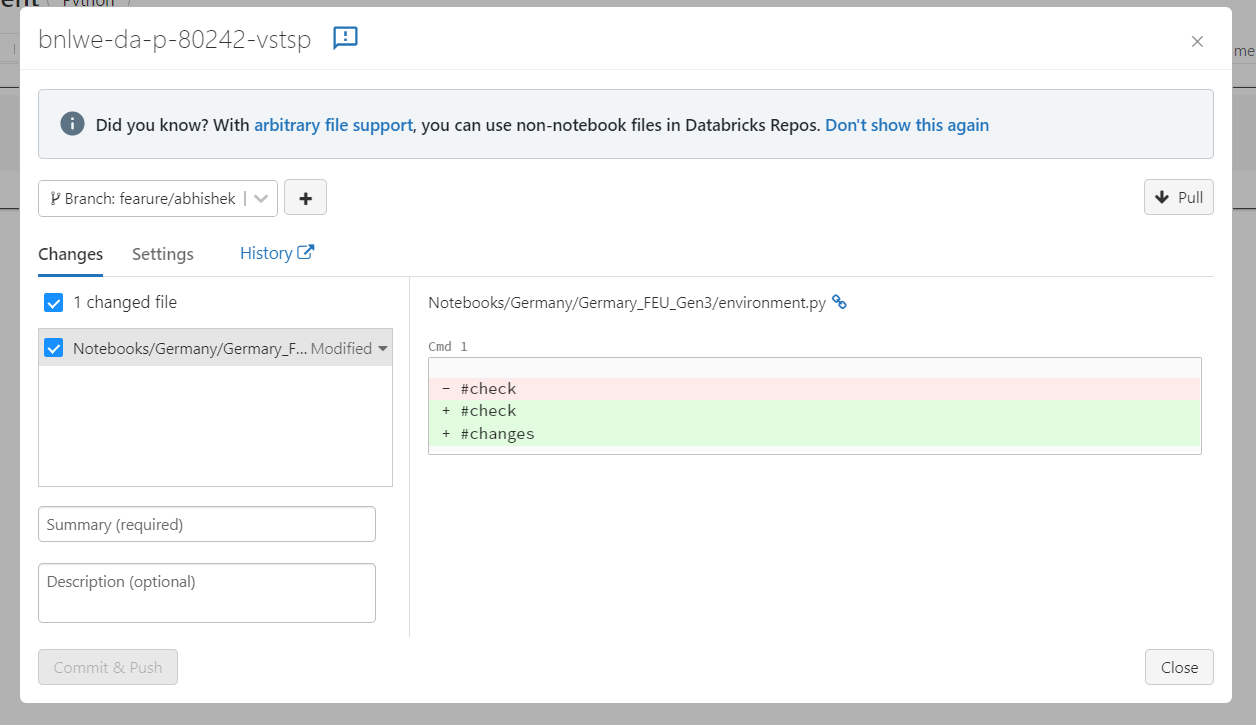


#### 3.4.5 Committing Files in Databricks:

* Now start working on Notebooks and when you are done with your changes, commit the code to Repo
  + **NOTE**: You can add any folder structure in NOTEBOOKS/COUNTRY1 folder it will get replicate in respective workspaces
  + **NOTE: Crosscheck branch name before commit**
* Use the git button to Commit and push the changes

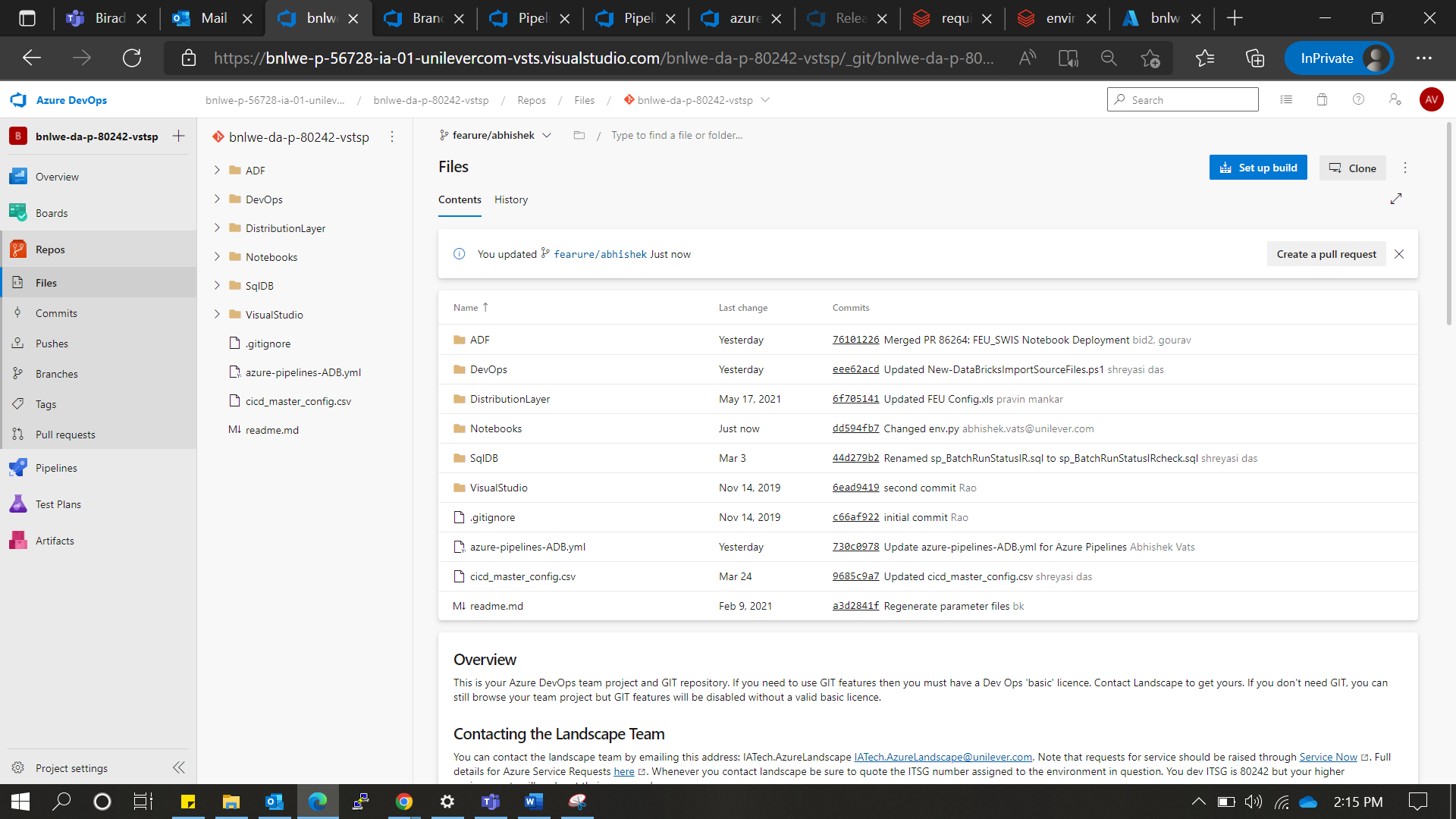


* It shows all your notebooks that you update review and write Summary then commit and push

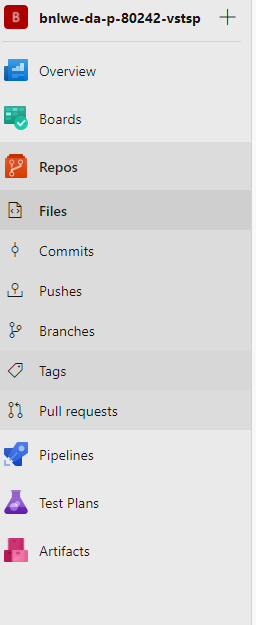


#### 3.4.6 Pull Request Creation:

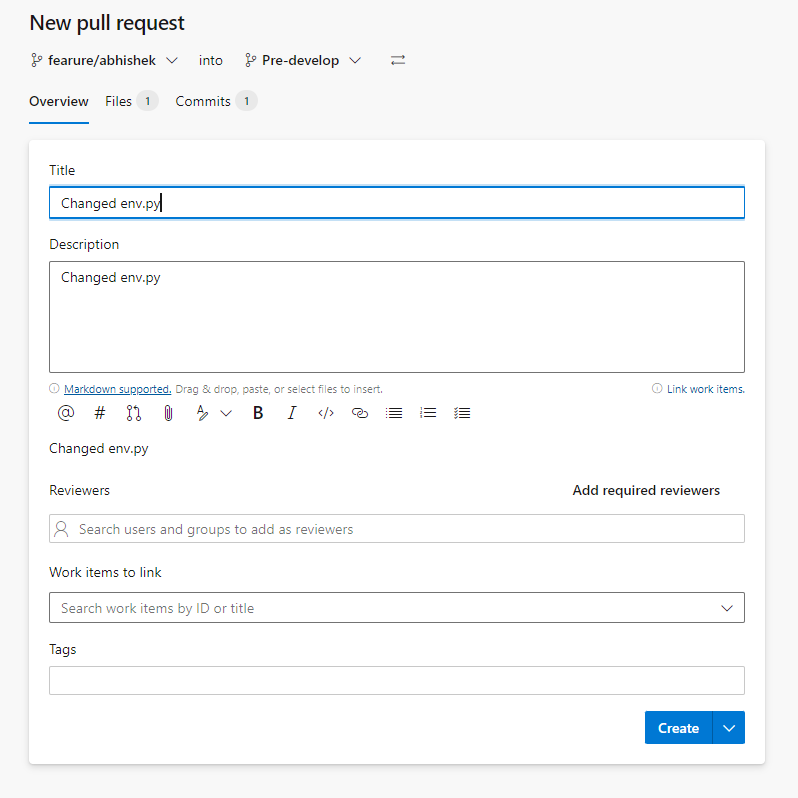
* Now your changes start reflecting in respective branch you pushed

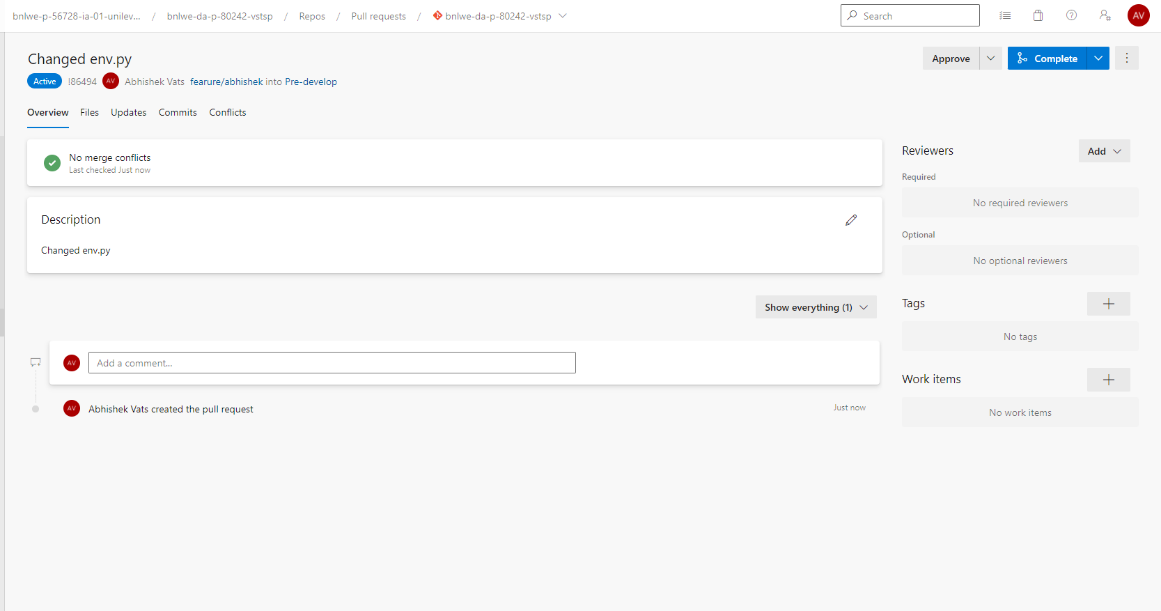


* When we are done with all changes and want to merge respective feature branch into Pre-Develop
* **NOTE**: Pre-Develop branch deploys Notebooks in ADB dev (80242) workspace for Testing
* Use Pull-requests to merge the branch into Pre-develop



* Create new PR
* Always create PR from your feature branch to Pre-Develop branch
* Here we can also check the files and commits



* We can add reviewer and merge the changes
* When approver approves your PR, you can merger the code into Pre-Develop

### 3.5 Artefact Logging:

For ML Runs in each category artifacts namely – train data, test data and model’s feature importance are being saved every week in ADLS. These files are being saved in ADLS in folders by week number. Train and Test data are important in PROJECT Country1 use case because as the model hyperparameters are not changing week by week, to recreate any week’s prediction we’ll need to have train and test data for that model hierarchy. These files saved as csv in ADLS are further used in successive steps for Drift calculation.

Graphical user interface, text, application, email

Description automatically generated

Feature importance data is also saved as a csv in ADLS and is utilized by the power BI dashboard to show feature importance for each model hierarchy.

Table

Description automatically generated

Since forecasting happens in a loop where each iteration is model training and prediction for one model hierarchy, we use the train and test data created in this step add a category column and timestamp column and append this data to a master train, test data frames. At the end of the loop, we get 2 master dataframes that are a combination of all train and test data, with additional identifier columns like – model hierarchy, timestamp and category. These files are then copied to ADLS in a folder specified by the current week, inside the folder the files are saved with the file names having the category and type.

### 3.6 KPI storage & schema:

**About PBI:**

**Build on Desktop version –**

**Graphical user interface, text, application

Description automatically generated**

**Access:**

Need to request access to admin - Read, Reshare

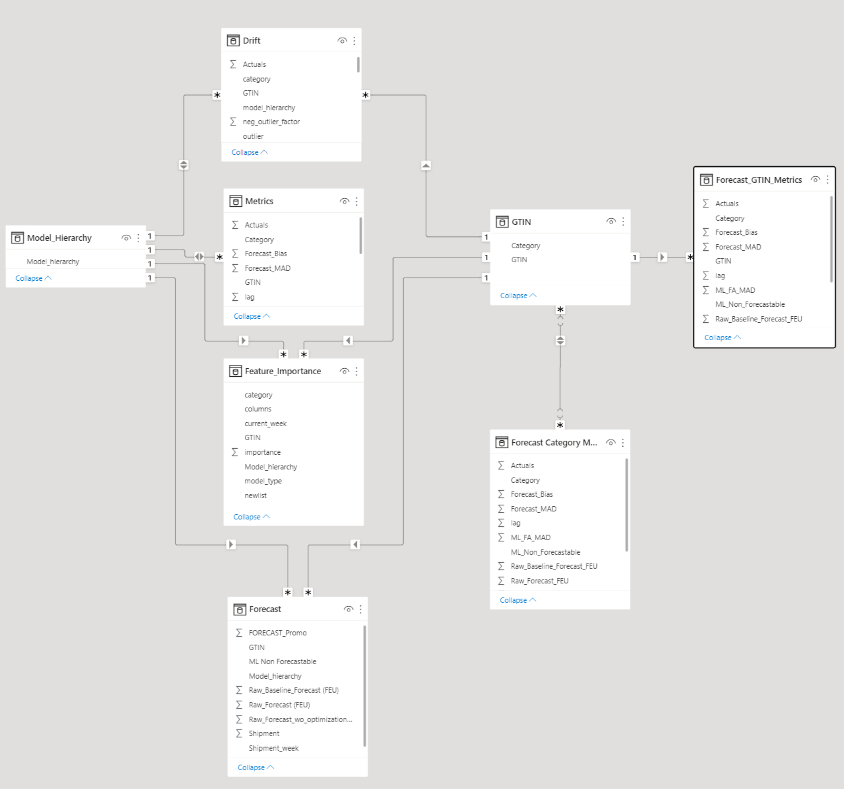
**Description of all the KPI used in power BI dashboard.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PBI KPI's** | | | | |
| **Metrics** | **Description** | **Level of granularity** | **Schedule** | **Formula** |
| **Baseline vs predicted vs Forecasted** | It’s a comparison between baseline, actuals, and the forecasted values (only key lags considered) | Model Hierarchy level/GTIN/Category | weekly | actual values/forecasted values |
| **Forecast Bias -( +ve and -ve Bias)** | A forecast bias occurs when there are consistent differences between actual outcomes and previously generated forecasts | Model Hierarchy level/GTIN/Category | weekly | (Actuals/Forecast)-1 |
| **MAD** | Forecasting measuring metrics | Model Hierarchy level/GTIN/Category | weekly | ABS(Actual-Forecast) |
| **Drift** | Change in data, seasonality over period | Model Hierarchy level/GTIN/Category | weekly |  |
| **Feature Importance** | Important features of models (RF/XGB) | Model Hierarchy level/GTIN/Category | weekly |  |
| **DQM Score** | Overall DQM score over different intervals (uniqueness, completeness, timeliness and datatype) | Table with their respective columns | weekly | sum of scores of all the DQM checks |

**Schema Description:**

The Schema has been designed into two parts 1st one is more about the Data science aspect which mostly talks about Feature Importance, Drift, forecasted values, MAD & BIAS. 2nd part is about the Data quality metrics which mainly focuses on completeness, Timeliness, Uniqueness and Data types check of the data.

Below is the back-end architecture for the DS aspect.



**Model Hierarchy:**It contains all the unique model hierarchy.

Script path:

**GTIN:**

**Feature Importance:**

It contains feature importance score, Model type(rf/xgb), the new list information and their respective categories.

Script path: /The Company/MLOps/Country1/Country1\_PROJECT\_Gen3/Logs/Power BI DB/DS/Importance/Final\_Result

Data source path:

**Drift**

It contains all the outlier values and their percentile which are being used in dashboard for drift KPI representation

Script path: /The Company/MLOps/Country1/Country1\_PROJECT\_Gen3/Logs/Power BI DB/DS/Drift/2022-19

Data source path:

**Metrics**

It contains all the actuals from all the categories from model along with categories, Lag, MAD and BIAS which are being used in dashboard for KPI representation

Script path :

**Forecasted**

It contains all the forecasted values from all the categories from models along with the baseline data, Model hierarchy and shipment week.

Script path : [Parallel\_Collation - Databricks (azuredatabricks.net)](https://adb-8577169965921698.18.azuredatabricks.net/?o=8577169965921698#notebook/2383507989734980/command/4167963855666163)

**Data Quality Metrics Backend Schema:**

Graphical user interface, application

Description automatically generated

**Script Path:**

[Databricks (azuredatabricks.net)](https://adb-8577169965921698.18.azuredatabricks.net/?o=8577169965921698#folder/2799724392399356)

**Completeness:** 100 -% of missing data in column/dataset

**Uniqueness:** 100 -% of duplicates (based on unique row identifier)

**Timeliness:** Status based on scheduled update of data sets

**Data Type:** Flag to indicate if input and expected data type matches (1= match, 0 = data type discrepancy)

### 3.7 Drift Detection:

Data Drift in machine learning is defined as the variation in production data from the data the model was trained and tested on. It can be gradual or sudden shift in data distribution due to factors like customer behavior change, seasonality, data source errors etc.

Drift indicates that the assumptions taken during model building might not be relevant in present scenario and that the predictions made by the model now might not be as accurate as it was during the time model was trained. It acts as a proxy for accuracy until the ground values are available.

By looking at the drift values one can make decisions whether the current model, hyperparameters work with newer data or can it lead to higher errors and the model needs to be retrained.

#### 3.7.1 PROJECT Country1 Drift Setup

In Country1 PROJECT the ML model is being trained every week and prediction is being done for the next 105 weeks. Train data consists of weekly sales for each model hierarchy merged with holiday data, so train data increases by 1 week’s data or 1 row for each week. In this case distribution shift doesn’t make much sense as the difference between 2 successive train data would be of just 1 week.

Here we look at the latest week’s sales data for a model hierarchy and then calculate how different this latest week’s sales data is as compared to rest of the sales data’s distribution for that specific Model Hierarchy. The latest week’s sales data value has a direct impact in the next few weeks’ prediction value as the sales values are used as lags columns also in the train data.

The latest week’s sales data being an outlier will indicate that this week’s predictions are going to have some variation as compared to previous weeks. The other columns in the train set don’t have as much impact to contribute to drift value as most of the other columns are dummy variables and have a lot of zeroes.

#### 3.7.2 Metrics

Now that we’re clear that the data drift must be estimated using latest Sales data (Actuals) the metrics being calculated to show the possible drift in data are:

* **LOF** - The main metric being used to identify if the latest sales data is an outlier, we used sklearn’s LocalOutlierFactor for this calculation. It measures the local deviation of the density of a given sample with respect to its neighbors. We show this value as ‘negative\_outlier\_factor’, Inliers tend to have -1, while outliers tend to have a larger LOF score. Based on the %contamination in the hyperparameter a threshold is calculated by LOF and the value is classified as “Outlier” or “Inliner” by giving the output as “True” or “False” respectively. The hyperparameters fo this model was chosen after experimentation.

Chart, scatter chart

Description automatically generated

An example shown above is a scatter plot for the actuals value for a model hierarchy, the x-axis is the serial number of the week and y-axis shows the corresponding Actuals values. The red dots are the values identified as outliers by the LOF model.

* **Percentile** – Gives the percentage of values in the Actuals distribution of that model hierarchy which are lower than the latest week’s sales data. Gives an idea of where the latest week’s data lies as compared to the distribution.
* **Q1, Q3** – Quartile 1 (25th percentile) and Quartile 3(75th percentile) values of the actual’s distribution for that model hierarchy. This gives an idea about inter quartile range and thus help estimate the spread of data.
* **Seasonal Drift lag 1** – Checks yearly seasonality in the data by comparing last week’s data with the data from the same week of previous year. If the current data lies outside the range of 2 standard deviation of last year’s local mean around the same time, the value is flagged as having seasonal drift and the output is True else False. The local mean and standard deviation for this purpose is calculated using the values of 9 weeks, 4 before and 4 after the last year’s week.
* **Seasonal Drift lag 2** – Same Procedure as seasonal drift lag 1, but its calculated using last to last year.

### Data Quality Measurement:

Data Quality Metrics (DQM) is an essential process in making sense of data which aims to help organizations to point out errors in their data that need to be resolved. It also aims at assessing if the data in their process is accurate to serve the intended purpose. All the basic operations of a business are managed quickly and efficiently when the data is managed properly. Adopting MLOps also include to improve the quality of data in a large-scale process by few key data quality metrics like Completeness, Uniqueness, Timeliness and Data Integrity which is describe below:

**Completeness:**

Data is considered to be complete when it fulfills certain expectations of comprehensiveness in an organization. Data completeness indicates if there is enough of it that can draw meaningful conclusions.

For each table we have taken a list of mandate column which need to be filled up after every data update. Here we can see the percentage of filled values in column and table also.

Table

Description automatically generated

Firstly, the column completeness is calculated looking for blank, null or NA in the field values and then they have their individual score which calculate the percentage of number of value filled/number of records as shown above. Table completeness Score was calculated by taking average of all the individual column completeness score. 100% show the table is filled completely without any column left blank.

**Uniqueness:**

The duplicacy of dataset is not acceptable by any system. Each data has their own identity. In business there is a requirement of data survey and duplicate dataset can lead to chaos to the whole system or organization.

Table

Description automatically generated

For each table, the set of columns are taken which will be consider as a key indicator for the table. The table score is calculated by 1 minus total number of duplicates taking the set of column from total number of records. To understand which particular column has more duplicate , column uniqueness score was kept same as table score.

**Timeliness:**

This refers to the updated data. We might not sometimes update our previous data through which we can get trouble with real time data which might require to rerun the entire process again.

Table

Description automatically generated

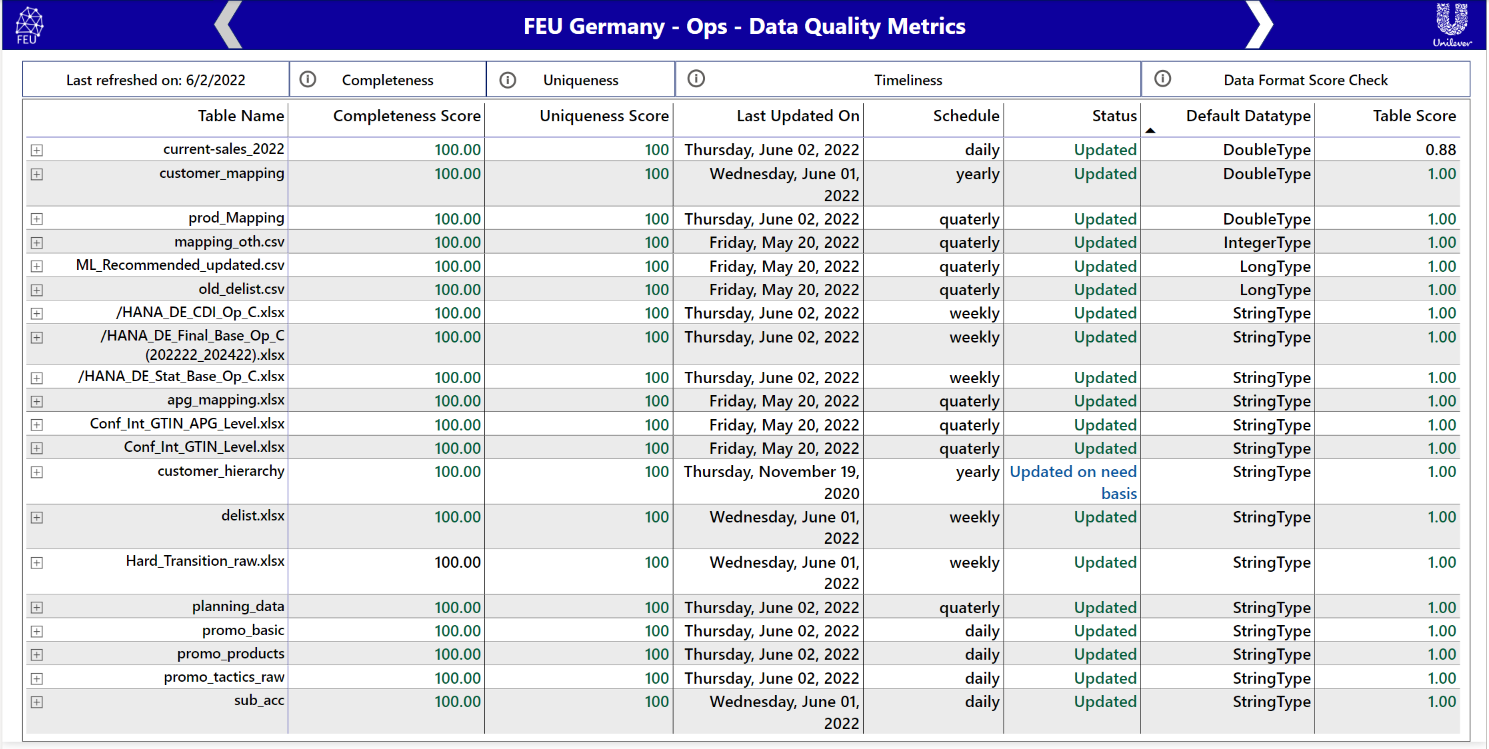
For each table, we have updated the Schedule column as when the table is updated frequently. Is it daily/weekly/monthly/quarterly/yearly? And found out the Last updated/modified date in ADLS and check where it was calculated hour difference between the last updated and current time is below the limit that was shown above. Status column finally conclude that whether that table is Updated or not in the DQM Dashboard

**Data Integrity (Data Format Score Check):**

This refers to structurally check the data contain the appropriate data types or not. For each table we have predefined metadata table where the intended datatype was filled. Then we check the data type of both the table and whichever matches we set the score as 1 else 0. We also check the count of column should also match considering it as extra column score. For overall table score we take the average of all the individual column score (including the count of column score.)



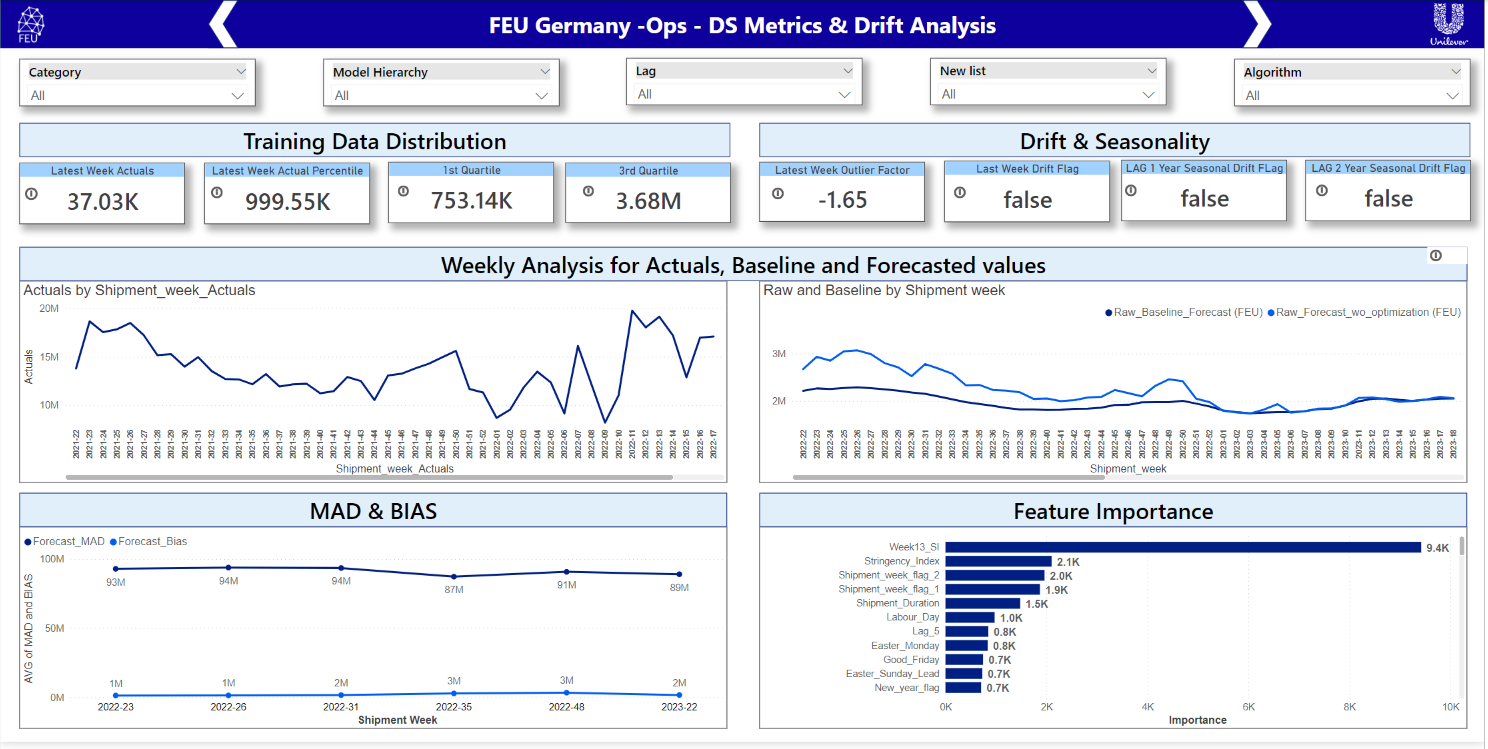
Note: DQM metrics got updated by running all the DQM Scripts and Last refreshed is updated in the Top left side of the DQM Dashboard.



As discussed above you can relate the Data Quality Metrics (DQM) shown in the above image which consists of Table name, Completeness Score which describes whether data received is absolute or not, then column next to it is of Uniqueness which check and let us know if there are distinct values or duplicates. After that the next section is of Timeliness where last update of file, it’s schedule and Status is shown. The last section of DQM is Data Format Score Check, where it compares data type of all columns of table and compare it with predetermined format to provide table score.

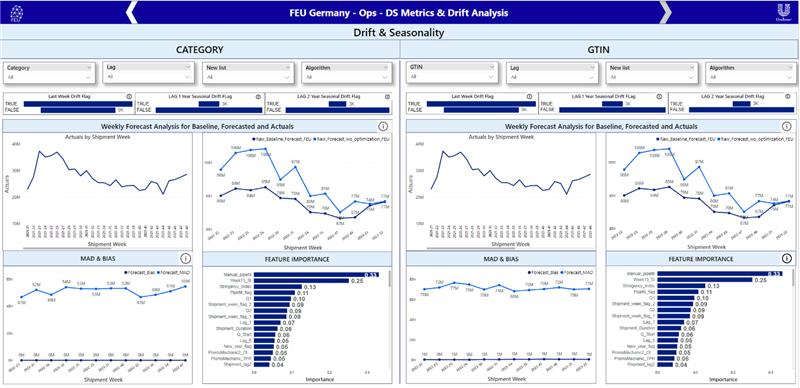
### 

### 3.9 PowerBI Ops Monitoring Dashboard:



This page provides insights about how Training Data Distribution is present, what are values in Drift and Seasonality and visuals showing actuals and forecasted values at Model Hierarchy Level

* Section 1 is the different slicers provided for necessary selection. Included slicers are Category, Model Hierarchy, Lag, New List, Algorithm
* Section 2 contains data related to Training Data Distribution and Drift & Seasonality.
  + KPI’s included in Training Data Distribution is Latest Week Actuals, Latest Week Actual Percentile, 1st Quartile and 3rd quartile.
  + Next to that Drift and Seasonality KPI’s – Latest Week Outlier Factor, Latest Week Drift Flag, Lag 1 Year Seasonal Drift Flag
* Section 3 consists of visuals
  + Actuals by Shipment Week – describes about distribution of actuals over given period of shipment weeks.
  + Raw and Baseline by Shipment Week – provides description about Raw Baseline Forecast (PROJECT) and Raw Forecast wo Optimization (PROJECT) and the difference in values between them over given period of shipment week.
  + MAD & BIAS – Forecasted MAD and Forecasted BIAS over shipment weeks.
  + Feature Importance – Sorting and showing different features based on importance calculated.



This page also provides insights about how Training Data Distribution is present, what are values in Drift and Seasonality and visuals showing actuals and forecasted values but at Category Level (left) and GTIN level (right).