

SCREENSHOT  
Document  
for  
UpGrad's Assignment  
Submission

Prepared By,  
Sanjeev Surendran

## Contents

Airflow Main UI Page .....	4
Initial run of Lead_Scoring_Data_Engineering_Pipeline .....	4
Initial run of Lead_scoring_training_pipeline .....	6
Initial run of Lead_scoring_inference_pipeline .....	7
Next run of Lead_Scoring_Data_Engineering_Pipeline .....	9
Next run of Lead_scoring_inference_pipeline .....	10
MLFlow of Baseline_model_exp01 .....	12
MLFlow of Baseline_model_exp02 .....	13
MLFlow of Lead_scoring_mlflow_production .....	14
Pytest of Data Pipeline .....	19

# Table of Figures

Figure 1: Airflow Main UI Page.....	4
Figure 2: Grid view of Lead_Scoring_Data_Engineering_Pipeline .....	5
Figure 3: Graph view of Lead_Scoring_Data_Engineering_Pipeline .....	5
Figure 4: Grid view of Lead_scoring_training_pipeline .....	6
Figure 5: Graph view of Lead_scoring_training_pipeline .....	7
Figure 6: Grid view of Lead_scoring_inference_pipeline.....	8
Figure 7: Graph view of Lead_scoring_inference_pipeline.....	8
Figure 8: Grid view of Lead_Scoring_Data_Engineering_Pipeline .....	9
Figure 9: Graph view of Lead_Scoring_Data_Engineering_Pipeline .....	10
Figure 10: Grid view of Lead_scoring_inference_pipeline.....	11
Figure 11: Graph view of Lead_scoring_inference_pipeline.....	12
Figure 12: Main MLFlow UI of Baseline_model_exp01 .....	12
Figure 13: MLFlow Artifacts of Baseline_model_exp01.....	13
Figure 14: Main MLFlow UI of Baseline_model_exp02 .....	13
Figure 15: MLFlow Artifacts of Baseline_model_exp02.....	14
Figure 16: Main MLFlow UI of Lead_scoring_mlflow_production.....	15
Figure 17: MLFlow Artifacts of Version 1 model .....	16
Figure 18: MLFlow Artifacts of Version 2 model .....	17
Figure 19: MLFlow Artifacts of Version 3 model .....	18
Figure 20: Version 3 model registered as Production Stage .....	19
Figure 21: Pytest Test Results.....	19

## Airflow Main UI Page

This is the main Airflow UI page that contains all the registered DAGs. We can see our 3 DAGs:

- Lead\_Scoring\_Data\_Engineering\_Pipeline
- Lead\_scoring\_training\_pipeline
- Lead\_scoring\_inference\_pipeline

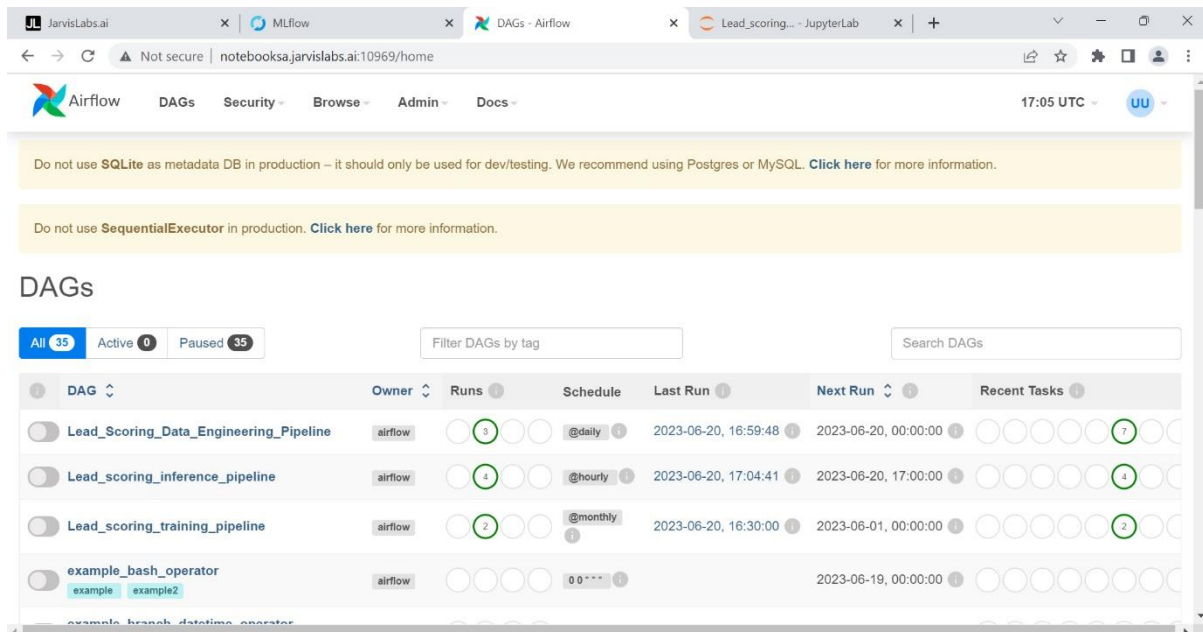


Figure 1: Airflow Main UI Page

## Initial run of Lead\_Scoring\_Data\_Engineering\_Pipeline

We have triggered manual run of Lead\_Scoring\_Data\_Engineering\_Pipeline for data engineering and cleaning data.

Both manual and scheduled runs can be seen in picture. Manual runs have white coloured arrow on top of green bars in Grid view.

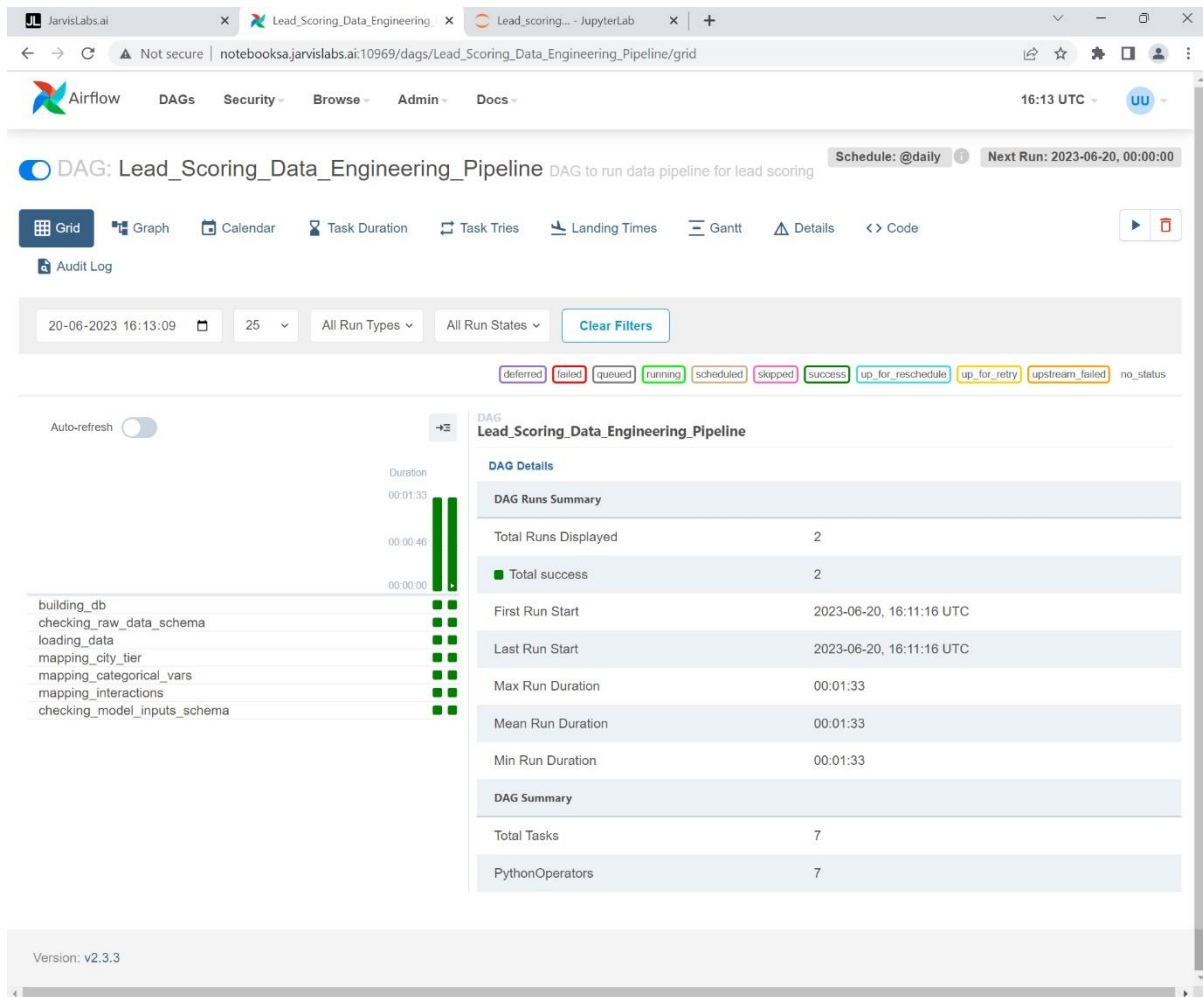


Figure 2: Grid view of Lead\_Scoring\_Data\_Engineering\_Pipeline

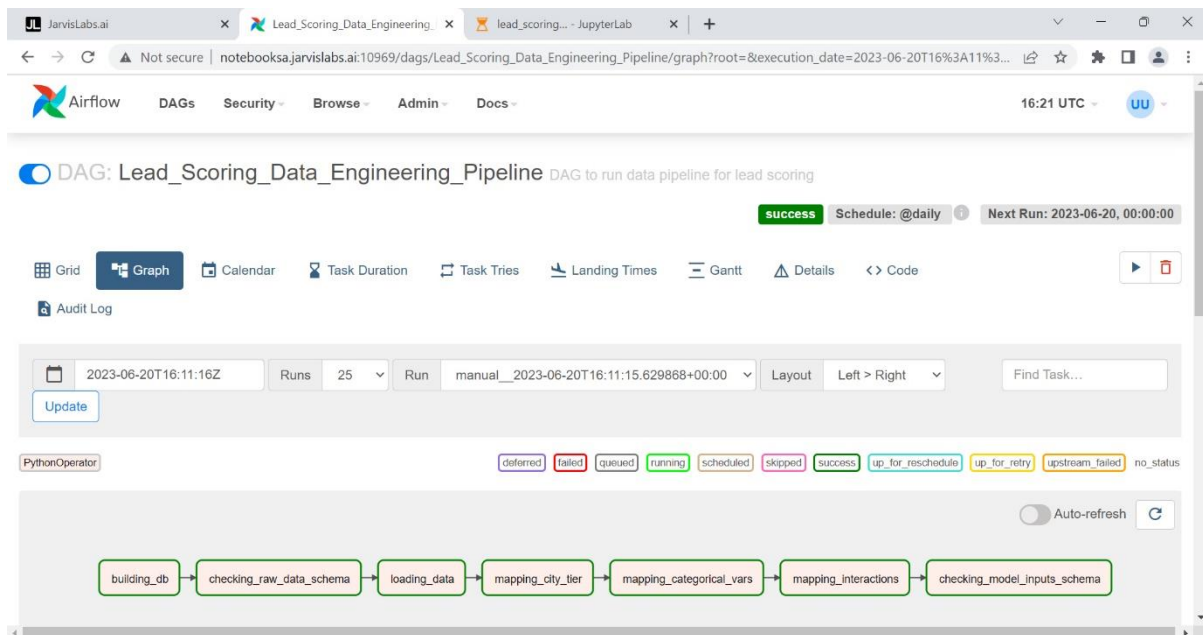


Figure 3: Graph view of Lead\_Scoring\_Data\_Engineering\_Pipeline

## Initial run of Lead\_scoring\_training\_pipeline

We have triggered manual run of Lead\_scoring\_training\_pipeline for training cleaned data.

Both manual and scheduled runs can be seen in picture. Manual runs have white coloured arrow on top of green bars in Grid view.

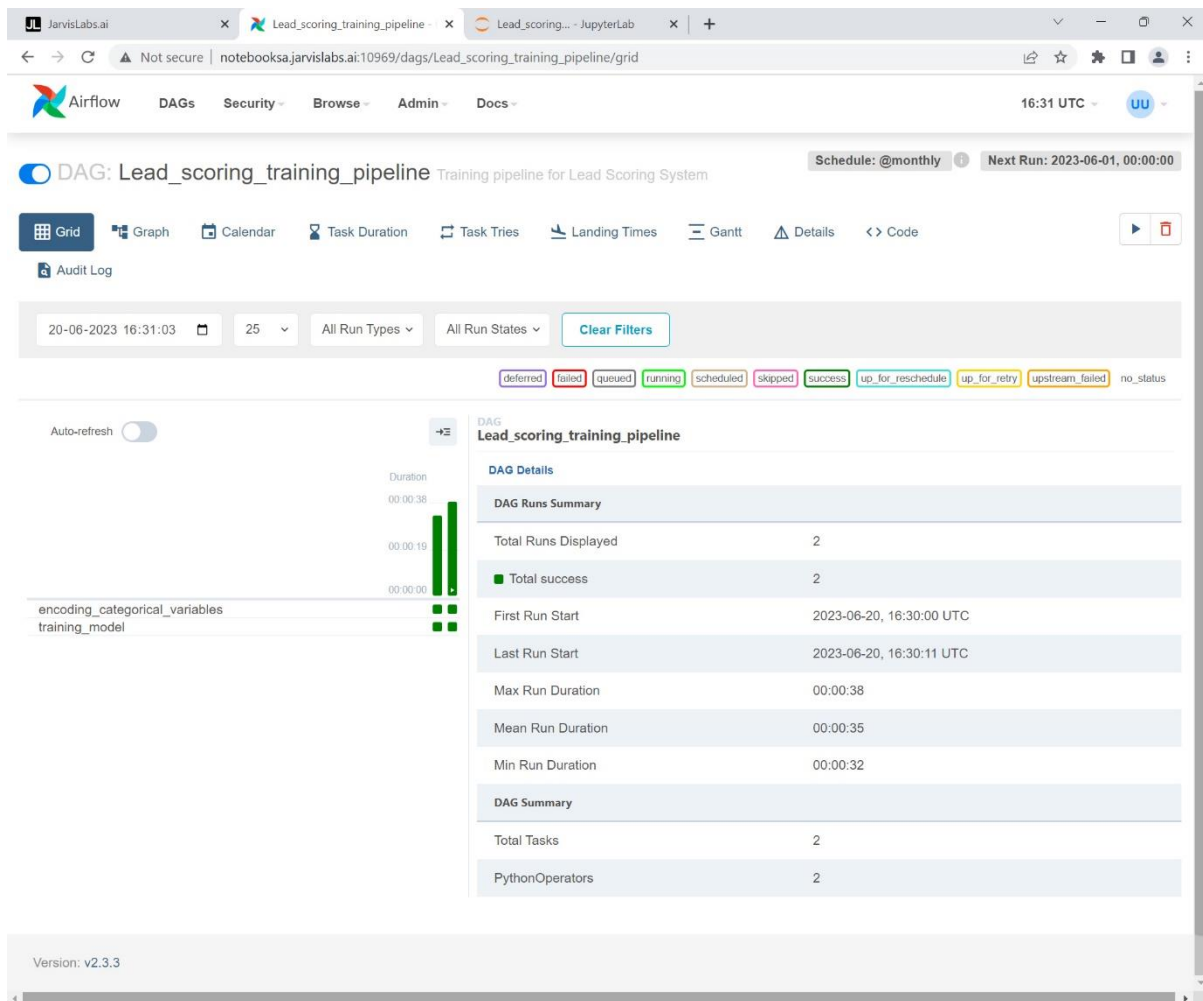


Figure 4: Grid view of Lead\_scoring\_training\_pipeline

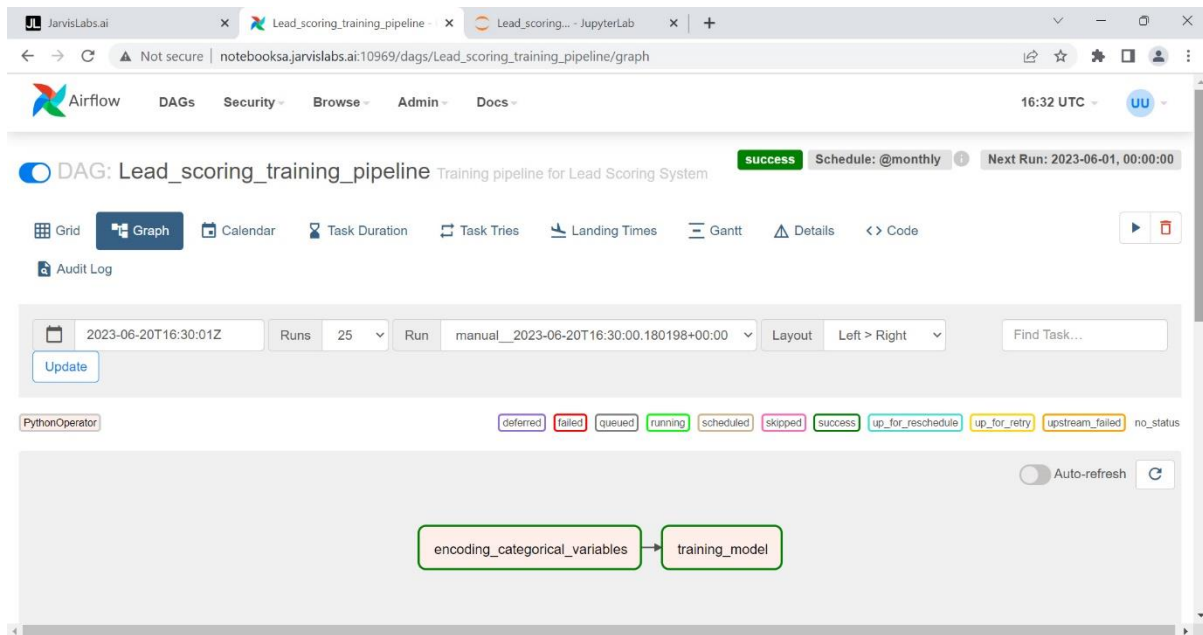


Figure 5: Graph view of Lead\_scoring\_training\_pipeline

## Initial run of Lead\_scoring\_inference\_pipeline

We have triggered manual run of Lead\_scoring\_inference\_pipeline for inferencing incoming data and predict outcome.

Both manual and scheduled runs can be seen in picture. Manual runs have white coloured arrow on top of green bars in Grid view.

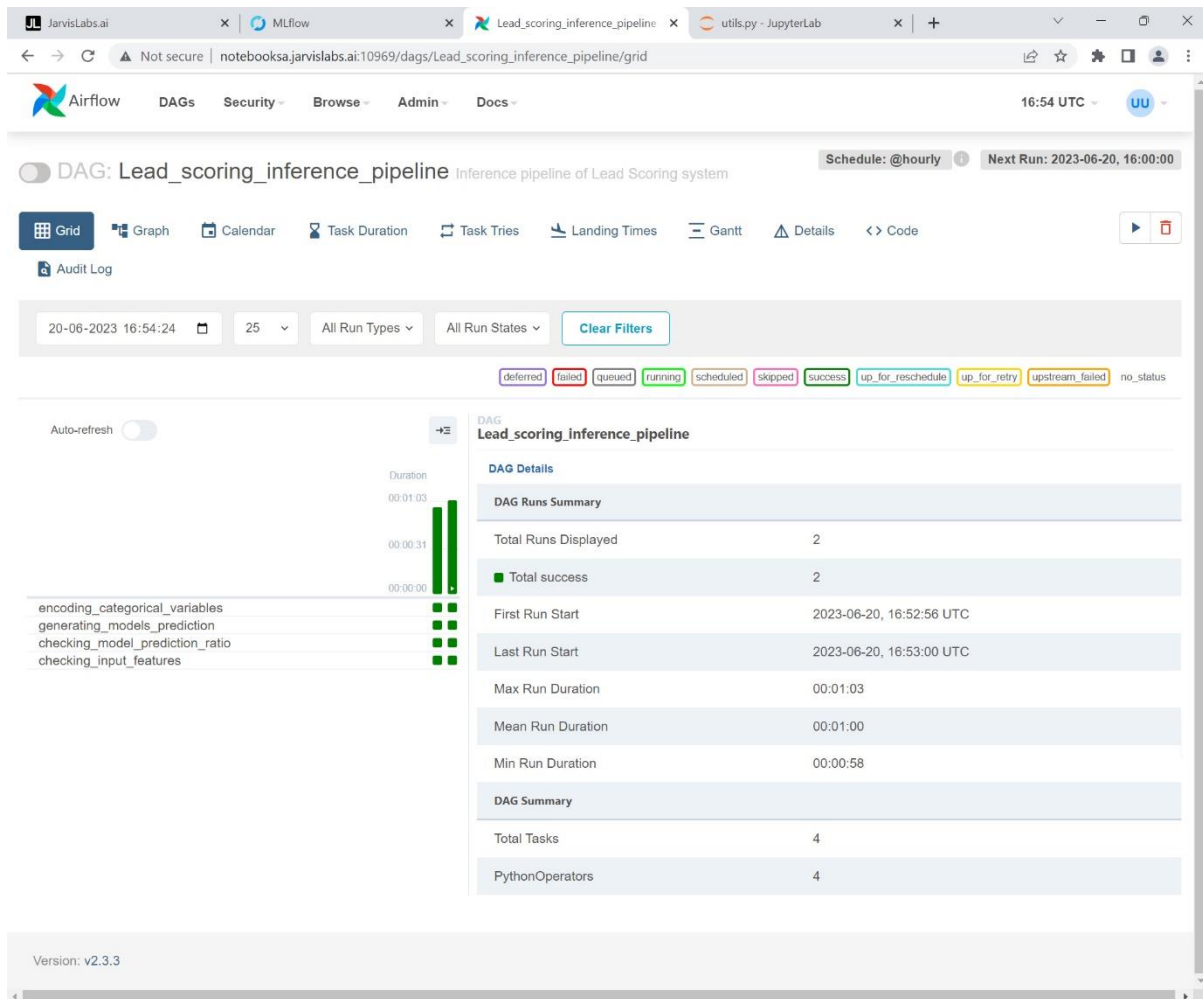


Figure 6: Grid view of Lead\_scoring\_inference\_pipeline

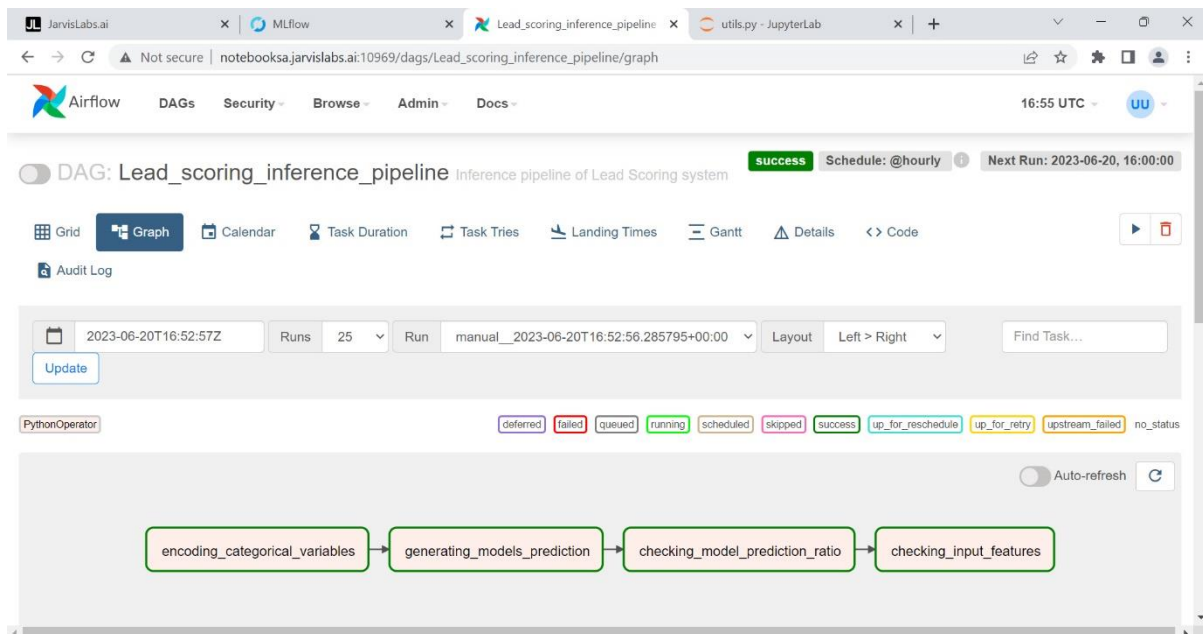


Figure 7: Graph view of Lead\_scoring\_inference\_pipeline



## Next run of Lead\_Scoring\_Data\_Engineering\_Pipeline

We have triggered manual run of Lead\_Scoring\_Data\_Engineering\_Pipeline for cleaning **leadscoring\_inference.csv**.

Both manual and scheduled runs can be seen in picture. Manual runs have white coloured arrow on top of green bars in Grid view.

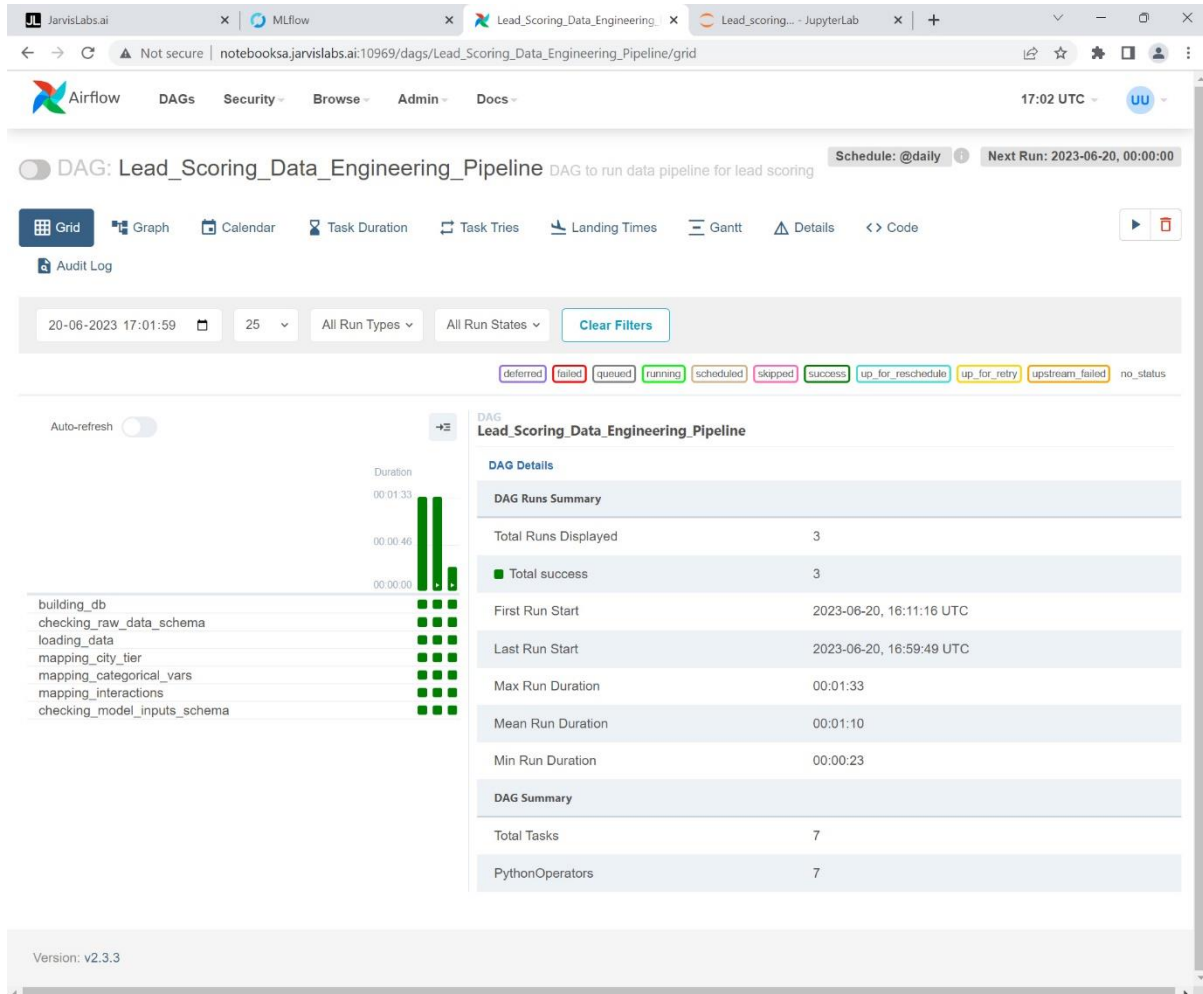
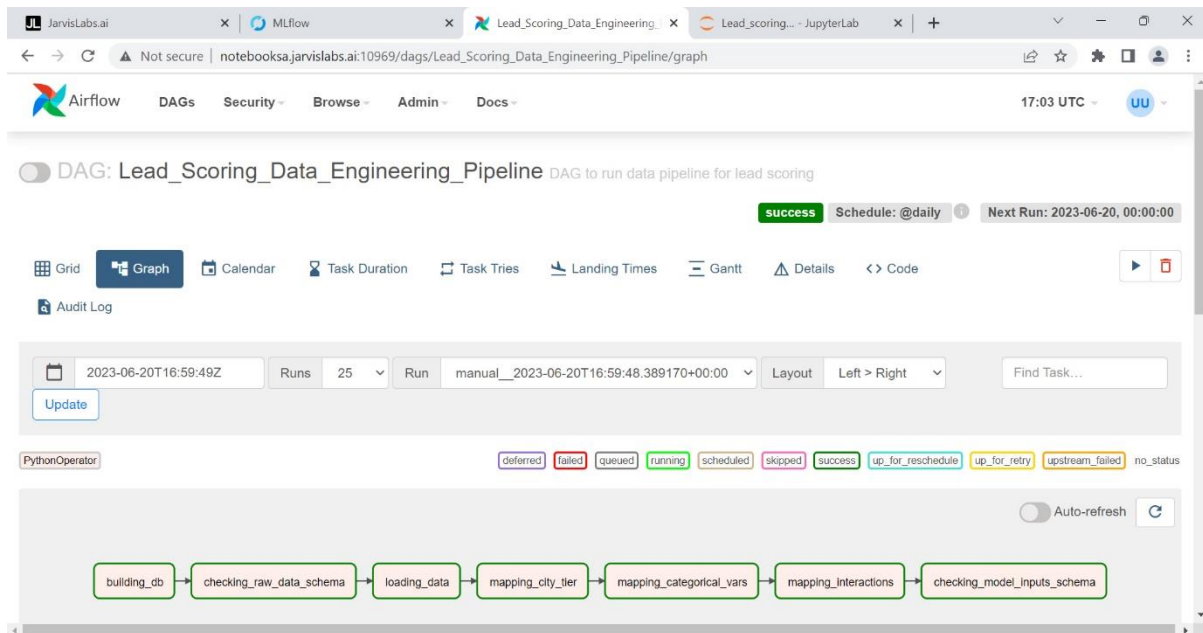


Figure 8: Grid view of Lead\_Scoring\_Data\_Engineering\_Pipeline



Next run of Lead\_scoring\_inference\_pipeline

We have triggered manual run of Lead\_scoring\_inference\_pipeline for inferencing incoming data and predict outcome.

Both manual and scheduled runs can be seen in picture. Manual runs have white coloured arrow on top of green bars in Grid view.

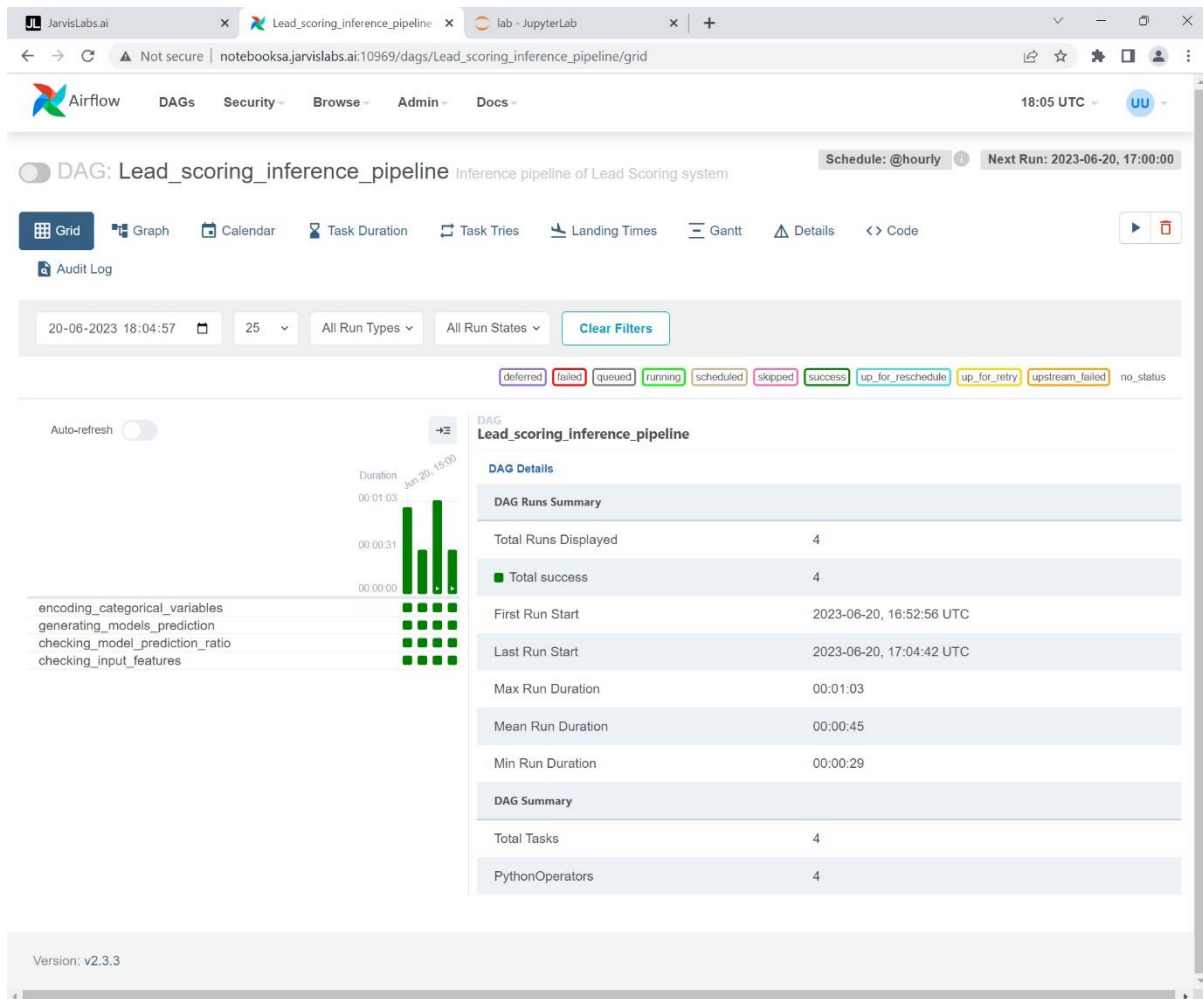


Figure 10: Grid view of Lead\_scoring\_inference\_pipeline

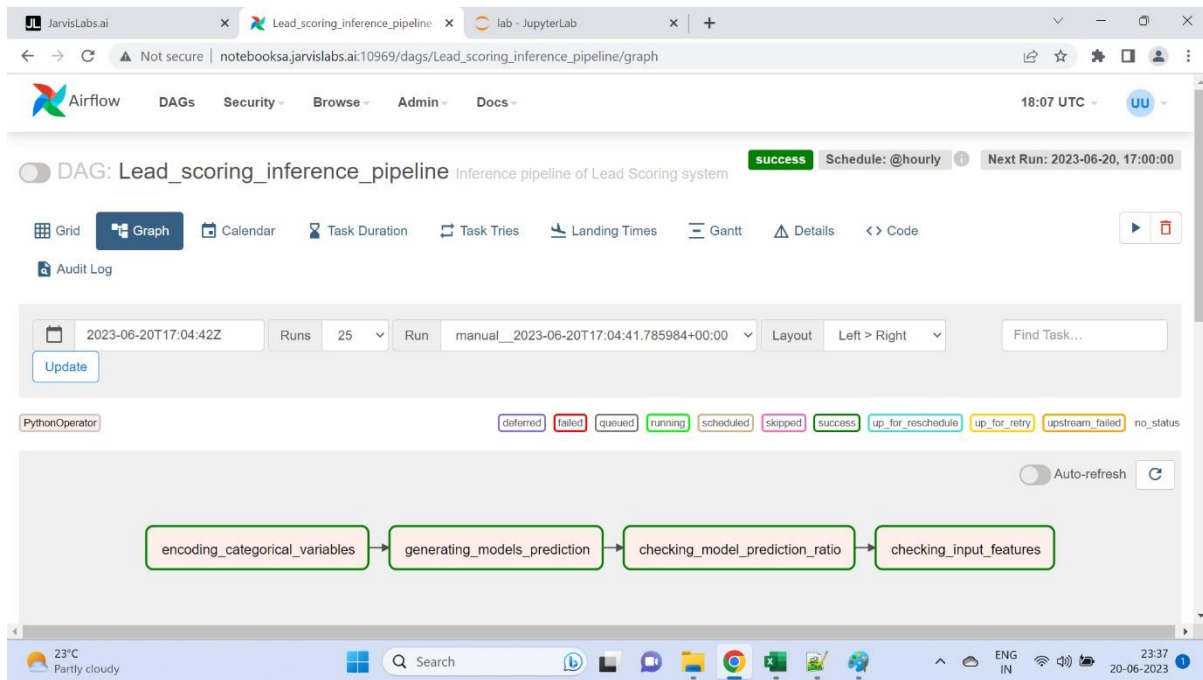


Figure 11: Graph view of Lead\_scoring\_inference\_pipeline

## MLFlow of Baseline\_model\_exp01

Baseline\_model\_exp01 is the initial pycaret setup experiment done for training model.

The screenshot shows the MLFlow UI for the 'Baseline\_model\_exp01' experiment. The interface includes a top navigation bar with 'mlflow', 'Experiments', and 'Models'. A sidebar on the left shows a list of experiments: 'Default', 'Baseline\_model\_exp01', 'Baseline\_model\_exp02', and 'Lead\_scoring\_mlflow...'. The main area displays the experiment details for 'Baseline\_model\_exp01', including the experiment ID '1', a description, and a table of runs. The table shows 11 matching runs, with the first run having a start time of '14 minutes ago' and a duration of 'Session Initi...'. The table columns are 'Start Time', 'Duration', 'Run Name', 'User', 'Source', 'Version', 'Models', and 'Metrics'. The 'Metrics' column shows 'AUC'.

Start Time	Duration	Run Name	User	Source	Version	Models	Metrics
14 minutes ago	Session Initi...		root	ipykernel...	-	-	-

Figure 12: Main MLFlow UI of Baseline\_model\_exp01

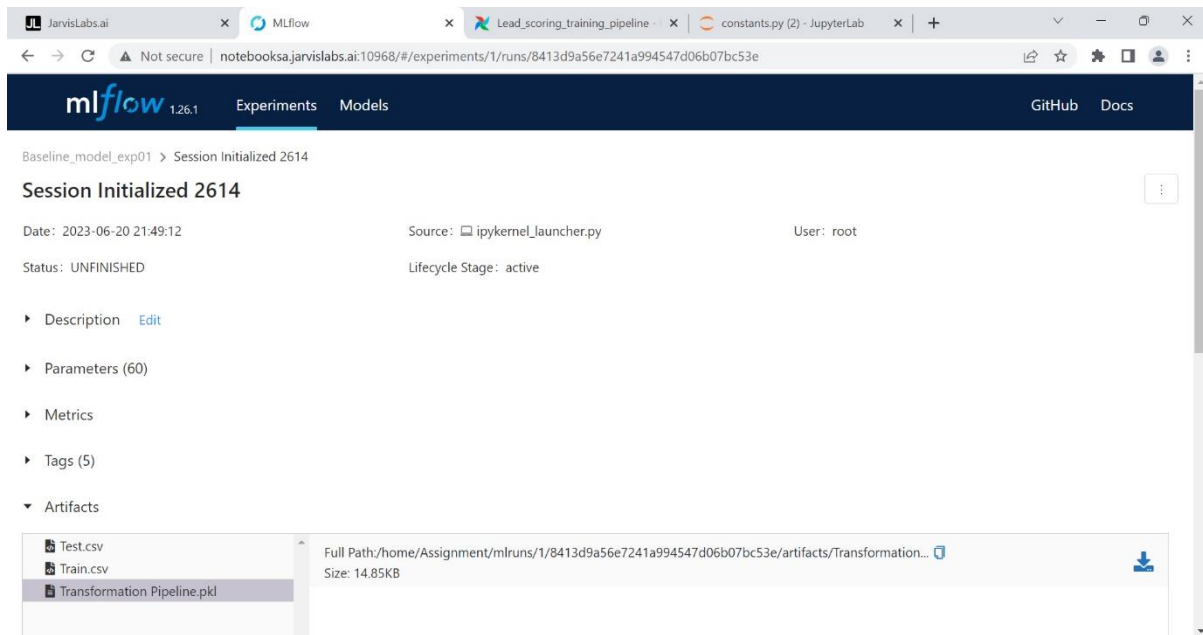


Figure 13: MLFlow Artifacts of Baseline\_model\_exp01

## MLFlow of Baseline\_model\_exp02

Baseline\_model\_exp02 is the pycaret setup experiment done for training model **after removing certain features**.

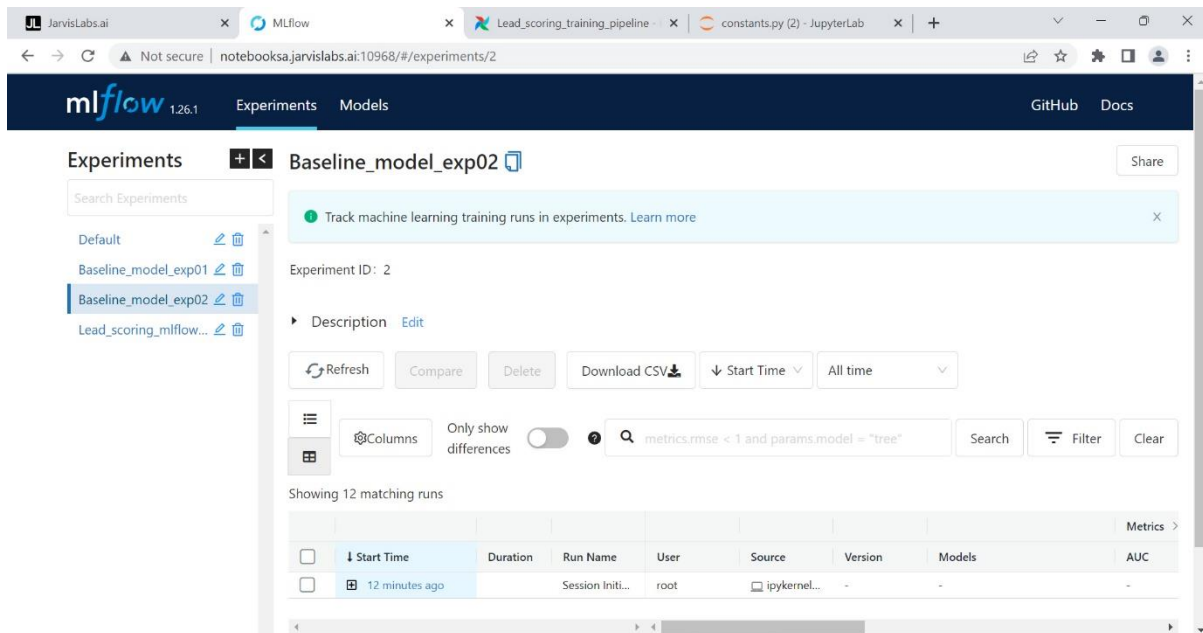


Figure 14: Main MLFlow UI of Baseline\_model\_exp02

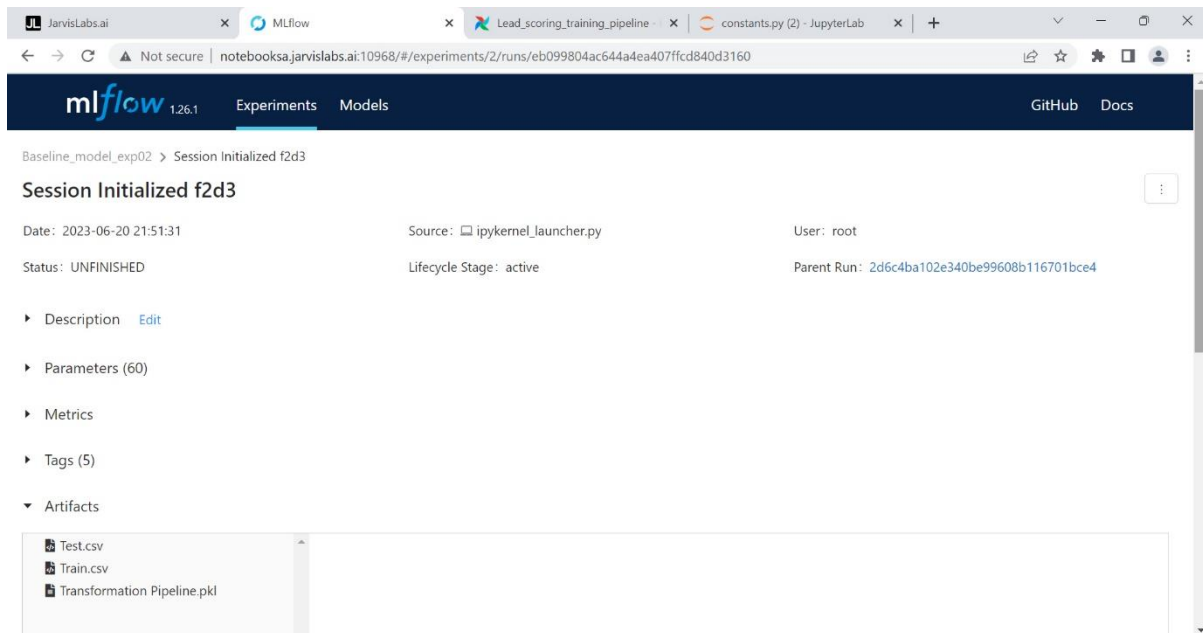


Figure 15: MLFlow Artifacts of Baseline\_model\_exp02

## MLFlow of Lead\_scoring\_mlflow\_production

Lead\_scoring\_mlflow\_production is the final model we have selected. We can see 3 models are generated:

- **Version 1** - One was created when dummy notebook was run to check the working of utils.py functions.
- **Version 2** - One was created during schedule run of airflow (automatic).
- **Version 3** - One was created manually from airflow UI. This is our final model and we have moved this version to **PRODUCTION** stage.

Please check the figure captions to understand the following screenshots.

JarvisLabs.ai MLflow Lead\_scoring\_training\_pipeline constants.py (2) - JupyterLab

Not secure notebooks.jarvislabs.ai:10968/#/experiments/3

mlflow 1.26.1 Experiments Models GitHub Docs

### Experiments

Search Experiments

- Default
- Baseline\_model\_exp01
- Baseline\_model\_exp02
- Lead\_scoring\_mlflow...

## Lead\_scoring\_mlflow\_production

Share

Track machine learning training runs in experiments. [Learn more](#)

Experiment ID: 3

Description Edit

Refresh Compare Delete Download CSV Start Time All time

Columns Only show differences metrics.rmse < 1 and params.model = "tree" Search Filter Clear

Showing 3 matching runs

	Start Time	Duration	Run Name	User	Source	Version	Models	Metrics
<input type="checkbox"/>	4 minutes ago	6.9s	Lead_scoring...	root	airflow	-	LightGBM/3	0.75
<input type="checkbox"/>	4 minutes ago	6.8s	Lead_scoring...	root	airflow	-	LightGBM/2	0.75
<input type="checkbox"/>	5 minutes ago	6.4s	Lead_scoring...	root	ipykernel...	-	LightGBM/1	0.75

Load more

Figure 16: Main MLFlow UI of Lead\_scoring\_mlflow\_production

Lead\_scoring\_mlflow\_production > Lead\_scoring\_mlflow\_production\_20\_06\_2023

### Lead\_scoring\_mlflow\_production\_20\_06\_2023

Date: 2023-06-20 21:59:22 Source: `ipykernel_launcher.py` User: root  
Duration: 6.4s Status: FINISHED Lifecycle Stage: active

- Description [Edit](#)
- Parameters (20)
- Metrics (9)
- Tags
- Artifacts

**models**

- MLmodel
- conda.yaml
- model.pkl
- python\_env.yaml
- requirements.txt

Full Path: `/home/Assignment/mlruns/3/8b4a64d02c9240019cbc8c116d1a5e6c/artifacts/models` [LightGBM, v1](#)  
Registered on 2023/06/20

#### MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the [model registry](#).

##### Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
No schema. See <a href="#">MLflow docs</a> for how to include input and output schema with your model.	

##### Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/8b4a64d02c9240019cbc8c116d1a5e6c/models'

# Load model as a Spark UDF. Override result_type if the model does not return double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double')

# Predict on a Spark DataFrame.
columns = list(df.columns)
df.withColumn('predictions', loaded_model(*columns)).collect()
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/8b4a64d02c9240019cbc8c116d1a5e6c/models'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

Figure 17: MLflow Artifacts of Version 1 model



The screenshot displays the MLflow web interface for a specific model artifact. The browser tabs at the top include 'JarvisLabs.ai', 'MLflow', 'Lead\_scoring\_training\_pi', and 'constants.py (2) - Jupyter'. The URL bar shows a 'Not secure' connection to 'notebooksa.jarvislabs.ai:10968/#/experiments/3/runs/5adfbe1e52fc41d4a5260c8fe740d88b'. The MLflow header shows version 1.26.1 and navigation links for 'Experiments' and 'Models'. The main content area is titled 'Lead\_scoring\_mlflow\_production\_20\_06\_2023' and provides metadata: Date (2023-06-20 22:00:25), Source (airflow), User (root), Duration (6.8s), Status (FINISHED), and Lifecycle Stage (active). A sidebar on the left lists artifacts under the 'models' folder: MLmodel, conda.yaml, model.pkl, python\_env.yaml, and requirements.txt. The main content area features a section titled 'MLflow Model' with a description: 'The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the [model registry](#).' Below this, there are two sections: 'Model schema' and 'Make Predictions'. The 'Model schema' section indicates that there is no schema and provides a link to MLflow docs for more information. The 'Make Predictions' section contains two code snippets. The first snippet shows how to predict on a Spark DataFrame using the MLflow PyFunc API. The second snippet shows how to predict on a Pandas DataFrame using the MLflow PyFunc API.

Full Path: /home/Assignment/mlruns/3/5adfbe1e52fc41d4a5260c8fe740d88b/artifacts/models

LightGBM, v2  
Registered on 2023/06/20

### MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the [model registry](#).

#### Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
No schema. See <a href="#">MLflow docs</a> for how to include input and output schema with your model.	

#### Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/5adfbe1e52fc41d4a5260c8fe740d88b/models'

# Load model as a Spark UDF. Override result_type if the model does not return double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double')

# Predict on a Spark DataFrame.
columns = list(df.columns)
df.withColumn('predictions', loaded_model(*columns)).collect()
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/5adfbe1e52fc41d4a5260c8fe740d88b/models'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

Figure 18: MLFlow Artifacts of Version 2 model

The screenshot displays the MLflow web interface for a production model artifact. The top navigation bar includes 'Experiments' and 'Models' tabs. The main header shows the experiment name 'Lead\_scoring\_mlflow\_production' and the specific model 'Lead\_scoring\_mlflow\_production\_20\_06\_2023'. Below this, metadata is provided: Date (2023-06-20 22:00:42), Source (airflow), User (root), Duration (6.9s), Status (FINISHED), and Lifecycle Stage (active).

The 'Parameters (20)' section lists various model parameters and their values:

Name	Value
boosting_type	gbdt
class_weight	None
colsample_bytree	1.0
importance_type	split
learning_rate	0.1
max_depth	-1
min_child_samples	20
min_child_weight	0.001
min_split_gain	0.0
n_estimators	100
n_jobs	-1
num_leaves	31
objective	None
random_state	42
reg_alpha	0.0
reg_lambda	0.0
silent	warn
subsample	1.0
subsample_for_bin	200000
subsample_freq	0

The 'Metrics (9)' section displays performance metrics:

Name	Value
AUC	0.75
False Negative	1939
False Positive	8430
Precision	0.74
Recall	0.75
True Negative	15213
True Positive	20211
F1	0.738
test_accuracy	0.741

The 'Tags' and 'Artifacts' sections are also visible. The 'Artifacts' section shows a file tree for the 'models' artifact, including files like 'MLmodel', 'conda.yaml', 'model.pkl', 'python\_env.yaml', and 'requirements.txt'.

Below the artifact list, the 'MLflow Model' section provides code snippets for making predictions. It includes a 'Model schema' section with a table for input and output schema (currently empty) and a 'Make Predictions' section with code for both Spark and Pandas DataFrames.

```
# Spark DataFrame
import mlflow
logged_model = 'runs:/52365e3cf06b48de9ee2d036677c86e1/models'

# Load model as a Spark UDF. Override result_type if the model does not return double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double')

# Predict on a Spark DataFrame.
columns = list(df.columns)
df.withColumn('predictions', loaded_model(*columns)).collect()

# Pandas DataFrame
import mlflow
logged_model = 'runs:/52365e3cf06b48de9ee2d036677c86e1/models'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

Figure 19: MLFlow Artifacts of Version 3 model

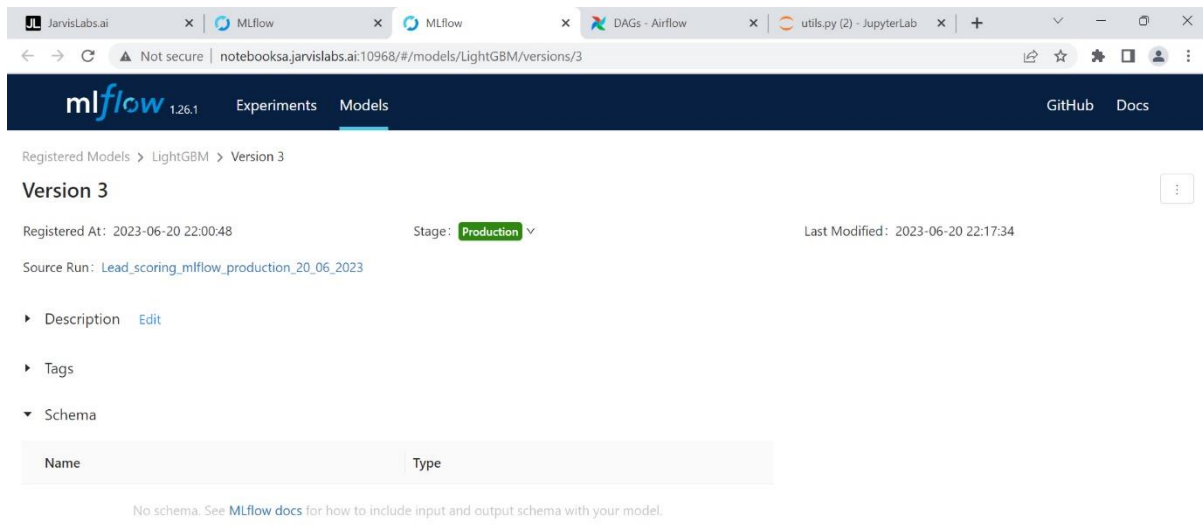


Figure 20: Version 3 model registered as Production Stage

## Pytest of Data Pipeline

Screenshot of pytest done for data pipeline.

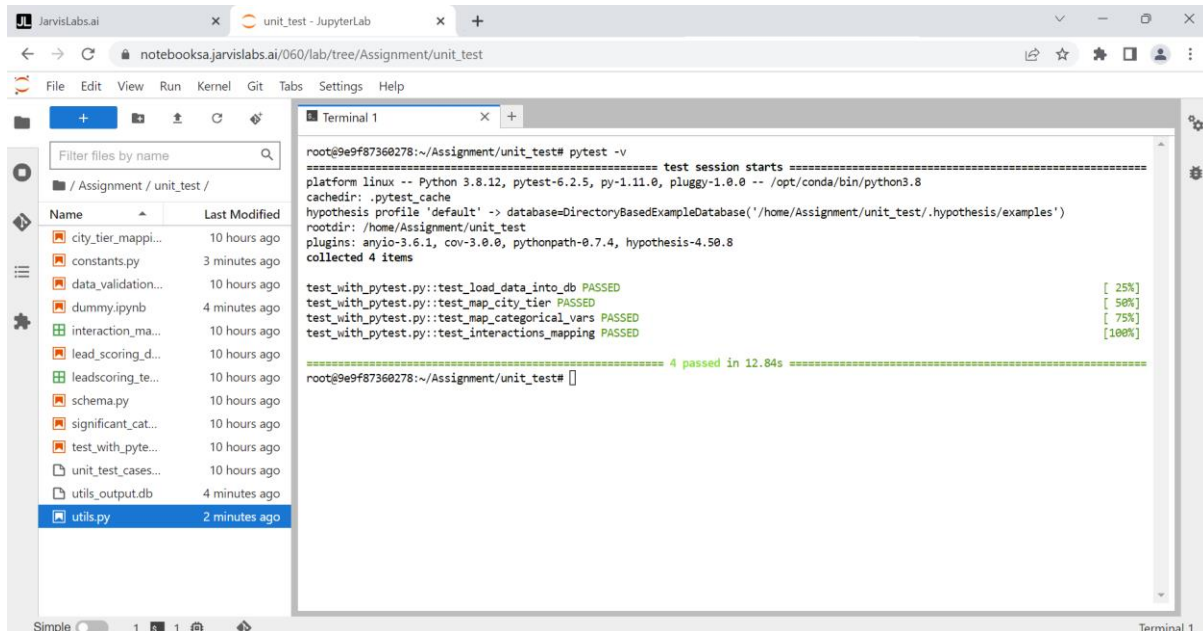


Figure 21: Pytest Test Results