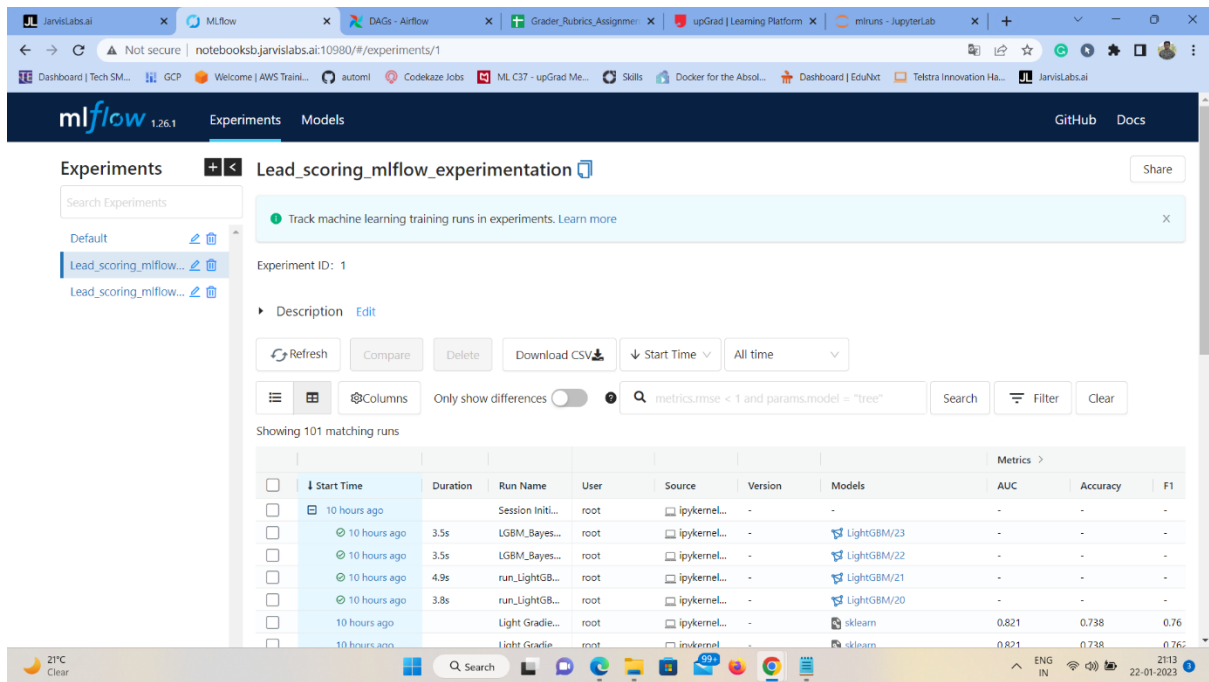


# Assignment Submission

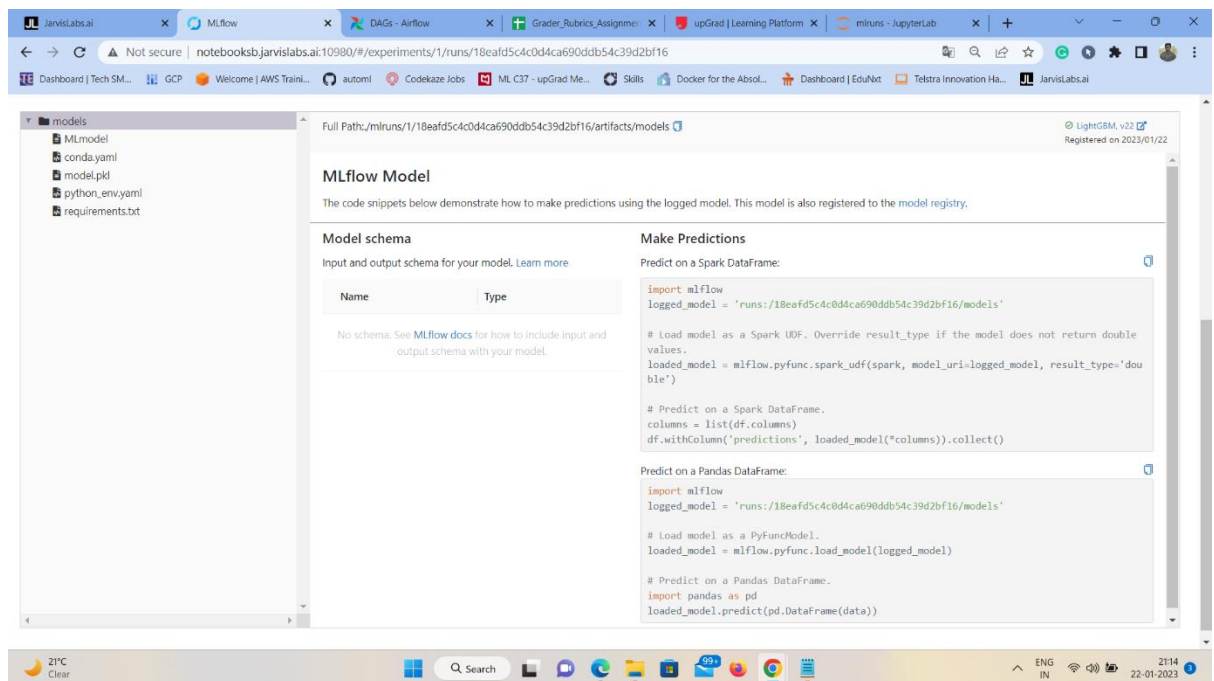
## 1) Screen shot of all the experiments



The screenshot shows the MLflow Experiments page for the experiment 'Lead\_scoring\_mlflow\_experimentation'. The page displays a list of runs with columns for Start Time, Duration, Run Name, User, Source, Version, Models, and Metrics (AUC, Accuracy, F1). The runs are sorted by Start Time, showing 101 matching runs. The table includes the following data:

Start Time	Duration	Run Name	User	Source	Version	Models	AUC	Accuracy	F1
10 hours ago	3.5s	Session Init...	root	ipykernel...	-	-	-	-	-
10 hours ago	3.5s	LGBM_Bayes...	root	ipykernel...	-	LightGBM/23	-	-	-
10 hours ago	3.5s	LGBM_Bayes...	root	ipykernel...	-	LightGBM/22	-	-	-
10 hours ago	4.9s	run_LightGB...	root	ipykernel...	-	LightGBM/21	-	-	-
10 hours ago	3.8s	run_LightGB...	root	ipykernel...	-	LightGBM/20	-	-	-
10 hours ago	-	Light Gradie...	root	ipykernel...	-	sklearn	0.821	0.738	0.76
10 hours ago	-	Light Gradie...	root	ipykernel...	-	sklearn	0.821	0.738	0.76

## 2) screenshot of one experiments with all the artifacts visible



The screenshot shows the MLflow Model page for the model 'LightGBM, v22'. The page displays the model schema and prediction code snippets. The model schema is defined as follows:

Name	Type
MLmodel	
conda.yaml	
model.pkl	
python_env.yaml	
requirements.txt	

The prediction code snippets are as follows:

```
import mlflow
logged_model = 'runs:/18eaf5c4c0d4ca690ddb54c39d2bf16/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:/18eaf5c4c0d4ca690ddb54c39d2bf16/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))
```

## 3) screenshot of all the experiments after dropping features

Start Time	Duration	Run Name	User	Source	Version	Models	AUC	Accuracy	F1	C
10 hours ago	-	Session Init...	root	ipykernel...	-	-	-	-	-	-
10 hours ago	3.5s	LGSM_Bayes...	root	ipykernel...	-	LightGBM/23	-	-	-	-
10 hours ago	3.5s	LGSM_Bayes...	root	ipykernel...	-	LightGBM/22	-	-	-	-
10 hours ago	4.9s	run_LightGB...	root	ipykernel...	-	LightGBM/21	-	-	-	-
10 hours ago	3.8s	run_LightGB...	root	ipykernel...	-	LightGBM/20	-	-	-	-
10 hours ago	-	Light Gradie...	root	ipykernel...	-	sklearn	0.821	0.738	0.76	-
10 hours ago	-	Light Gradie...	root	ipykernel...	-	sklearn	0.821	0.738	0.762	-
10 hours ago	-	Naive Bayes	root	ipykernel...	-	sklearn	0.734	0.672	0.725	-
10 hours ago	-	Linear Discr...	root	ipykernel...	-	sklearn	0.773	0.7	0.727	-
10 hours ago	-	Ridge Classif...	root	ipykernel...	-	sklearn	0	0.7	0.727	-
10 hours ago	-	Logistic Reg...	root	ipykernel...	-	sklearn	0.784	0.71	0.74	1.0
10 hours ago	-	Decision Tre...	root	ipykernel...	-	sklearn	0.817	0.736	0.758	-
10 hours ago	-	Extra Trees C...	root	ipykernel...	-	sklearn	0.817	0.737	0.758	-
10 hours ago	-	Random For...	root	ipykernel...	-	sklearn	0.818	0.737	0.759	-
10 hours ago	-	Light Gradie...	root	ipykernel...	-	sklearn	0.821	0.738	0.762	-
10 hours ago	-	Extreme Gra...	root	ipykernel...	-	-	-	-	-	-
10 hours ago	-	Session Init...	root	ipykernel...	-	-	-	-	-	-
10 hours ago	-	Light Gradie...	root	ipykernel...	-	sklearn	0.821	0.739	0.762	-
10 hours ago	-	Naive Bayes	root	ipykernel...	-	sklearn	0.734	0.662	0.727	-
10 hours ago	-	Linear Discr...	root	ipykernel...	-	sklearn	0.774	0.7	0.728	-
10 hours ago	-	Ridge Classif...	root	ipykernel...	-	sklearn	0	0.7	0.728	-
10 hours ago	-	Logistic Reg...	root	ipykernel...	-	sklearn	0.784	0.71	0.74	1.0
10 hours ago	-	Decision Tre...	root	ipykernel...	-	sklearn	0.817	0.736	0.758	-
10 hours ago	-	Extra Trees C...	root	ipykernel...	-	sklearn	0.818	0.737	0.758	-

#### 4) screenshot of one experiments with all the artifacts visible after dropping features

**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:/b612c1a8162043cdbf3b7b82ab44757e/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()
```

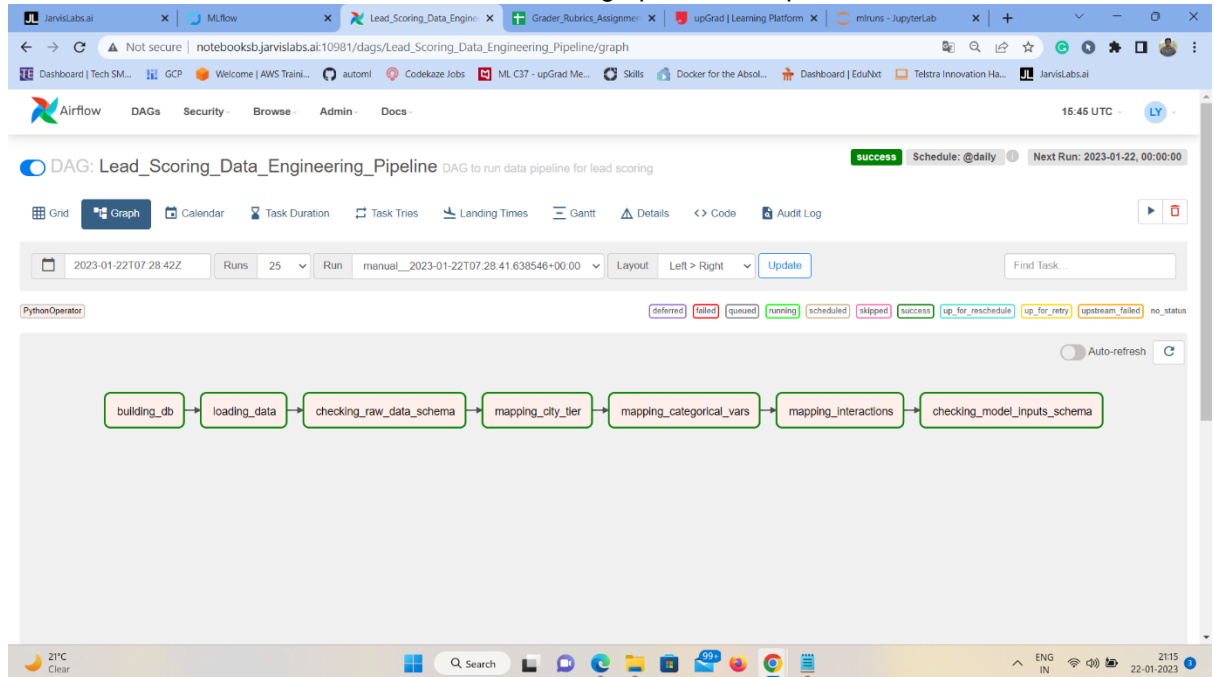
Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/b612c1a8162043cdbf3b7b82ab44757e/models'

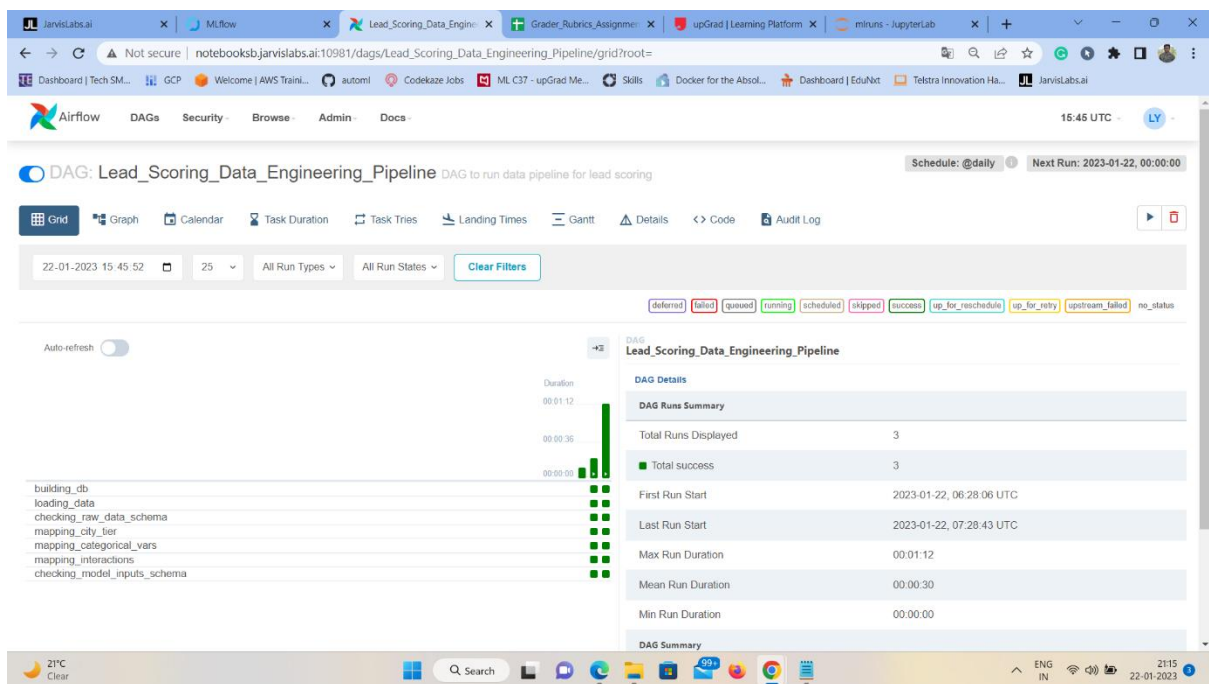
# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

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

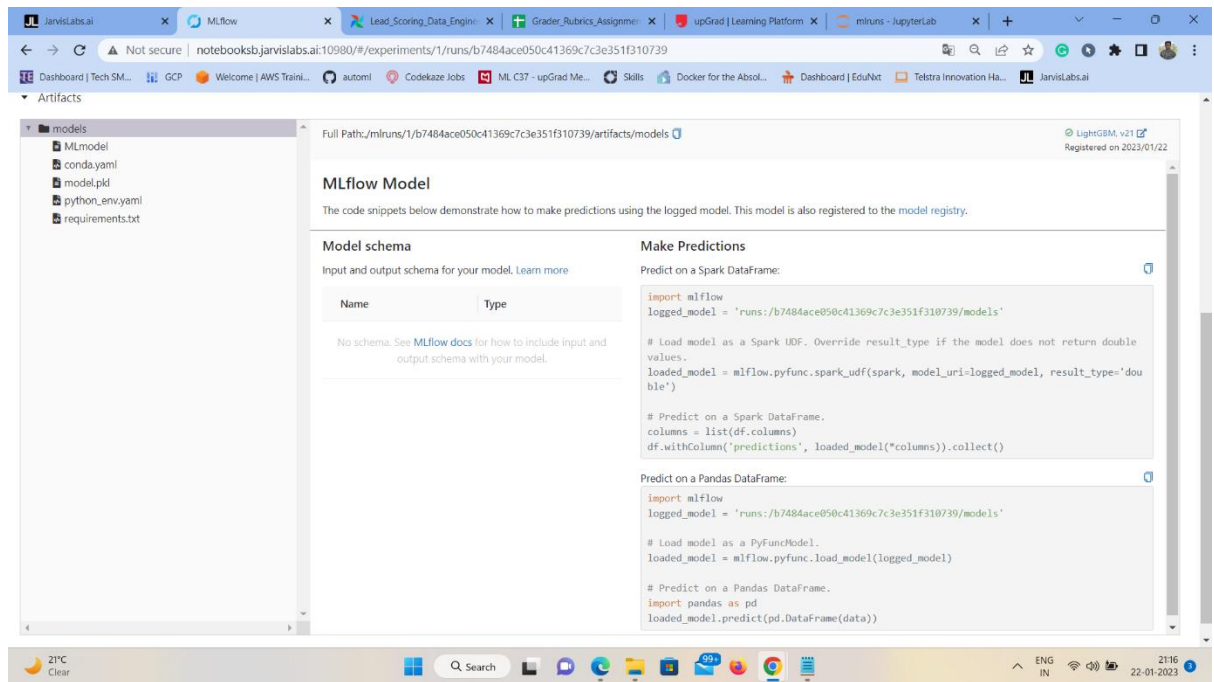
## 5) Screenshot of successful execution Airflow DAG in graph – Data Pipeline



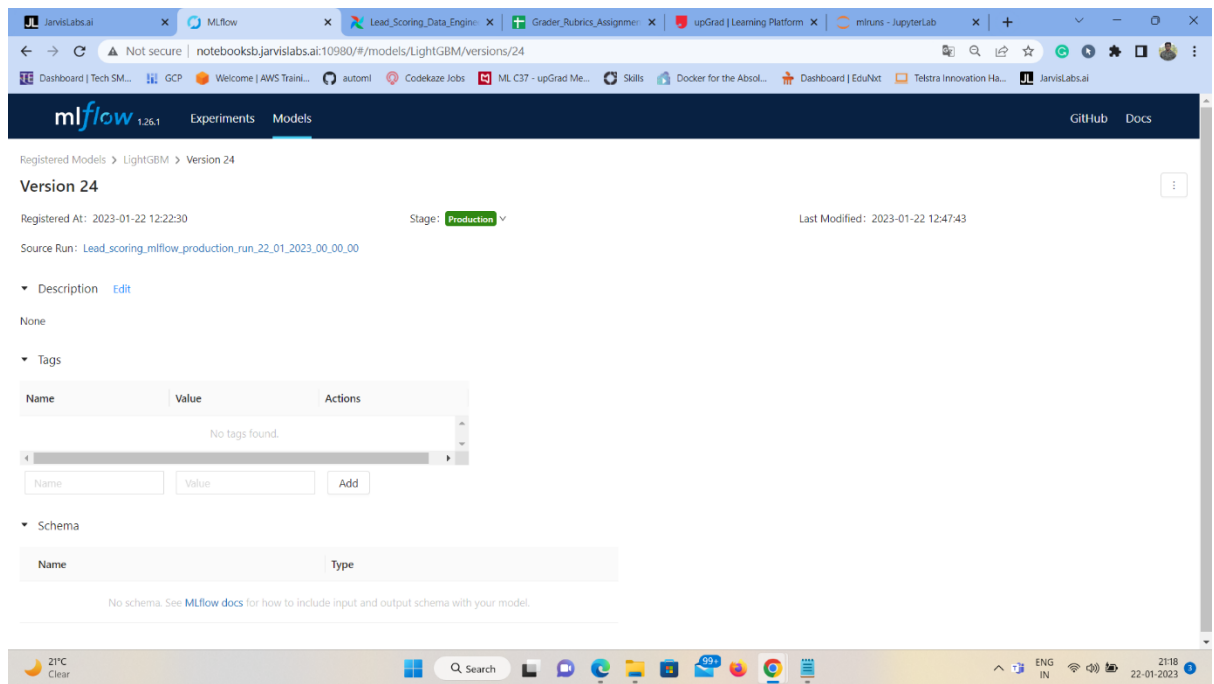
## 6) Screenshot of Airflow UI grid – Data Pipeline



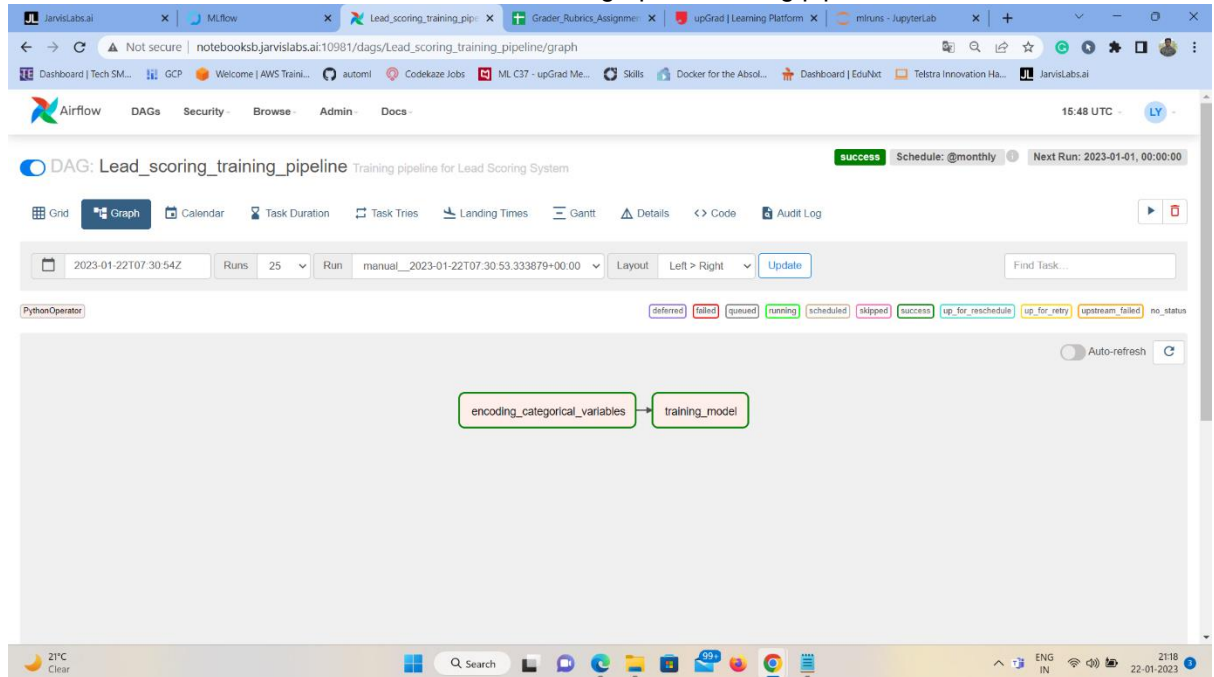
## 7) screenshot of experiments with all the artifacts visible – Training pipeline



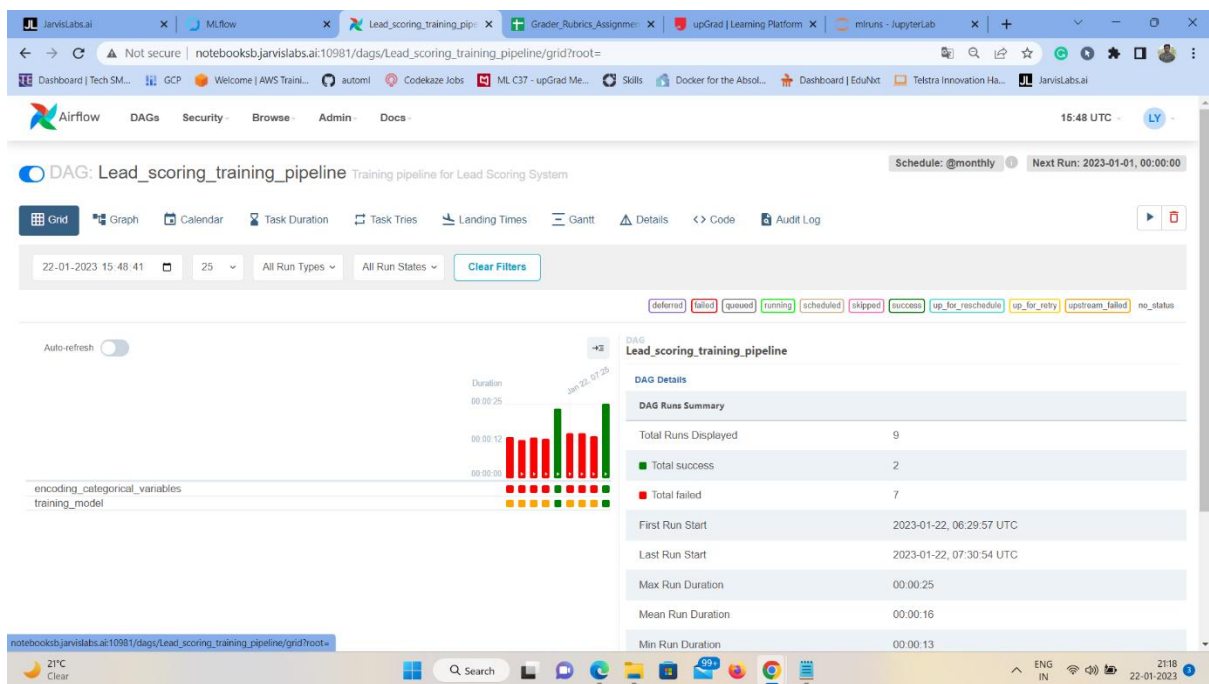
## 8) screenshot of model registry with model name and stage as 'production'



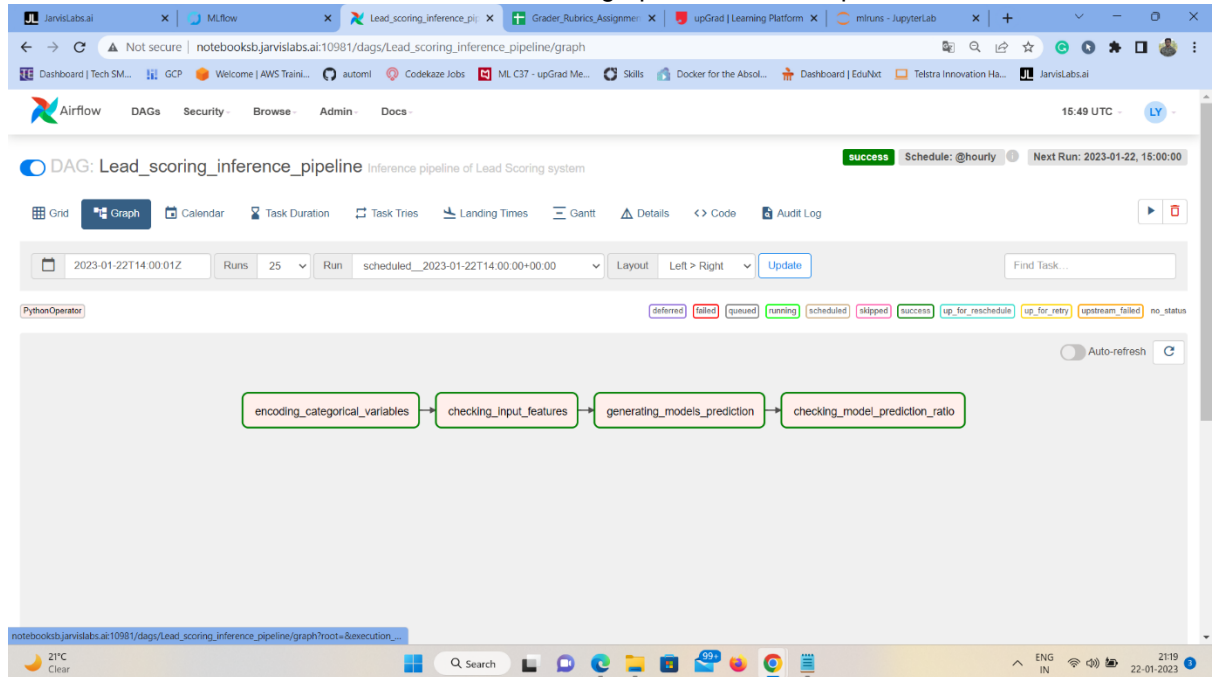
## 9) Screenshot of successful execution Airflow DAG in graph – Training pipeline



## 10) Screenshot of Airflow UI grid – Training pipeline



## 11) Screenshot of successful execution Airflow DAG in graph – Inference Pipeline



## 12) Screenshot of Airflow UI grid – Inference Pipeline

