Automation of end to end Machine learning pipeline using Amazon Web Services(AWS)

Vehicle Insurance Claim Fraud Detection using Machine Learning

Naveen Raju Sreerama Raju Govinda Raju
naveenraju100@gmail.com | +1 224 7067718 | +91 9886157101 |
https://www.linkedin.com/in/naveen-raju-s-g-bb1486124
https://naveenrajusg.github.io/Portfolio/

Table of Contents

Please zoom in to view the screenshots clearly

1. Project Description

The goal of this project is to build a model that can detect auto insurance fraud. And also automate different stages of ML life cycle like Data processing, Model training, Model evaluation, model creation, model bias check, model explain-ability check, model registration and model deployment in Amazon web services(AWS).

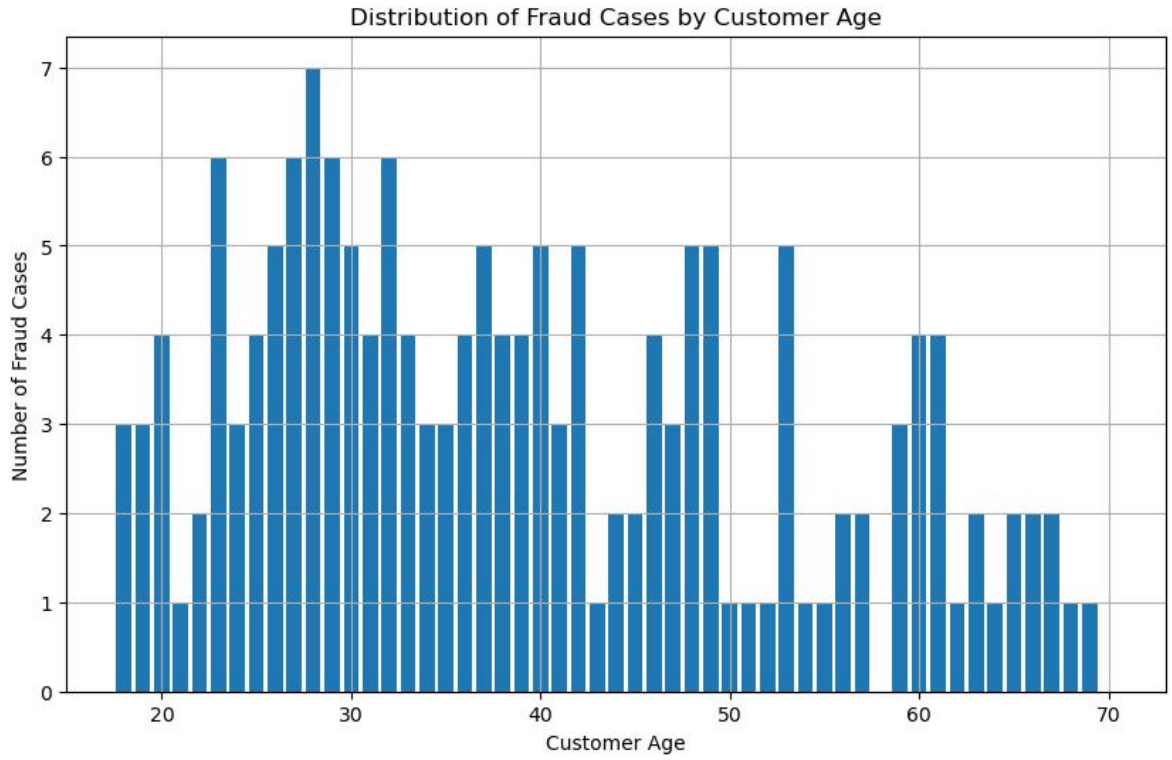
2. Dataset

Attribute	Type
fraud	int64
driver_relationship	category
incident_type	category
collision_type	category
incident_severity	category
authorities_contacted	category
num_vehicles_involved	int64
num_injuries	int64
num_witnesses	int64
police_report_available	float64
injury_claim	int64
vehicle_claim	float64
total_claim_amount	float64
incident_month	int64
incident_day	int64
incident_dow	int64
incident_hour	int64
customer_age	int64
months_as_customer	int64
num_claims_past_year	int64
num_insurers_past_5_years	int64
policy_state	category
policy_deductable	int64
policy_annual_premium	int64
policy_liability	float64
customer_gender	category
customer_education	category
auto_year	int64

3. Exploratory data Analysis

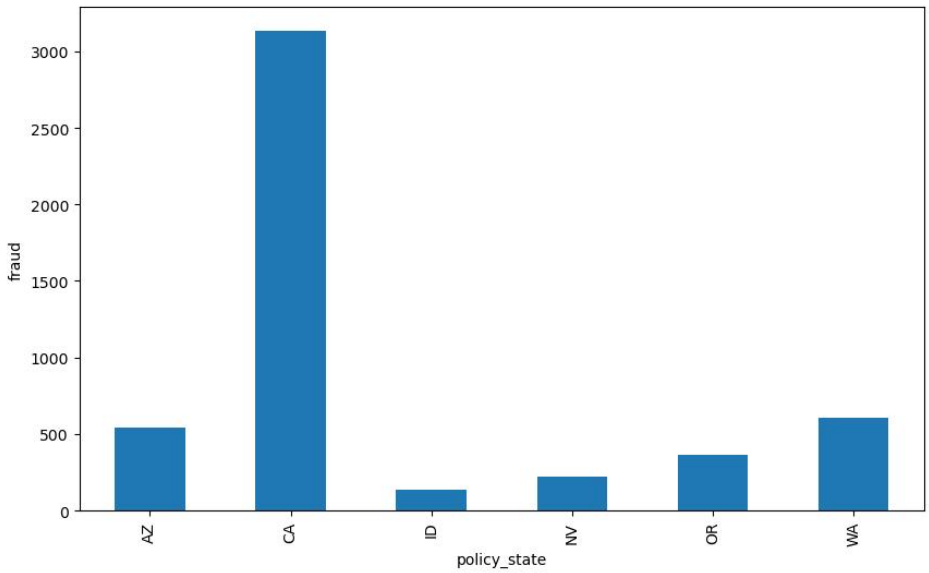
A) Distribution of fraud cases by customer age

We can see customers with age between 20 - 30 have more fraud cases.



B) Distribution of policy_state which is fraud claim.

California has highest fraud claims whereas Idaho state has least fraud claims.



Percentages of fraud claims state wise :-

Idaho state :

Total size of observations of Idaho state= 135
Number of frauds by Idaho state= 4
Percentage of fraud in claims made= 2.96 %

Arizona state :

Total size of observations of arizona state= 539
Number of frauds by Arizona state= 17
Percentage of fraud in claims made= 3.15 %

Nevada state:

Total size of observations of Nevada state= 222
Number of frauds by Nevada state= 6
Percentage of fraud in claims made= 2.7 %

Oregon state:

Total size of observations of Oregon state= 363
Number of frauds by Oregon state= 12
Percentage of fraud in claims made= 3.31 %

Washington state:

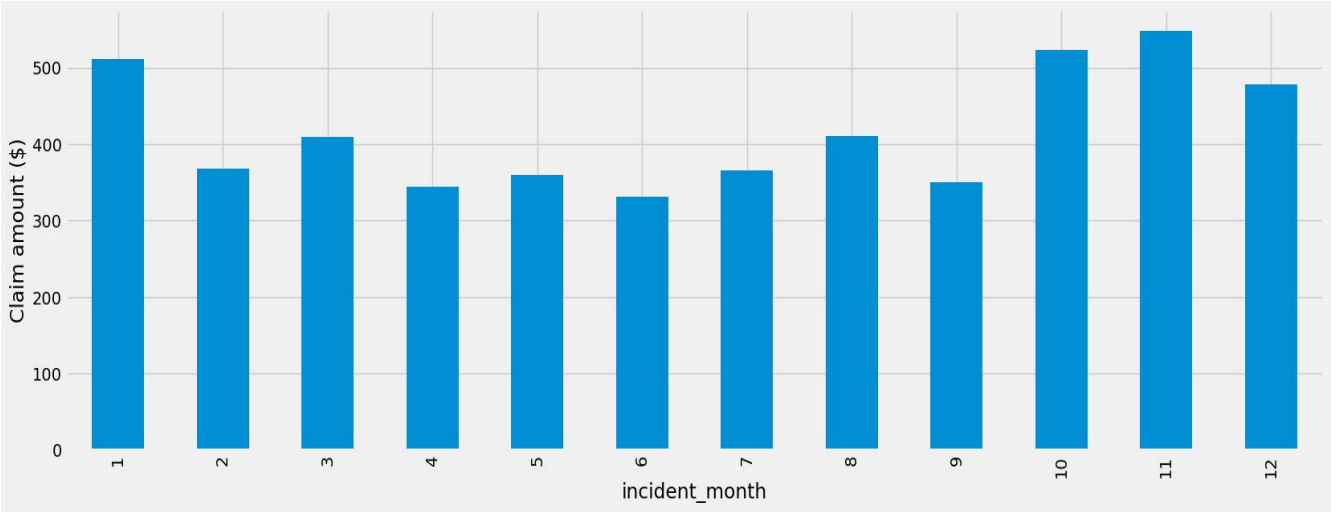
Total size of observations of Washington state= 607
Number of frauds by Washington state= 19
Percentage of fraud in claims made= 3.13 %

California state:

Total size of observations of California state= 3134
Number of frauds by California state= 106
Percentage of fraud in claims made= 3.38 %

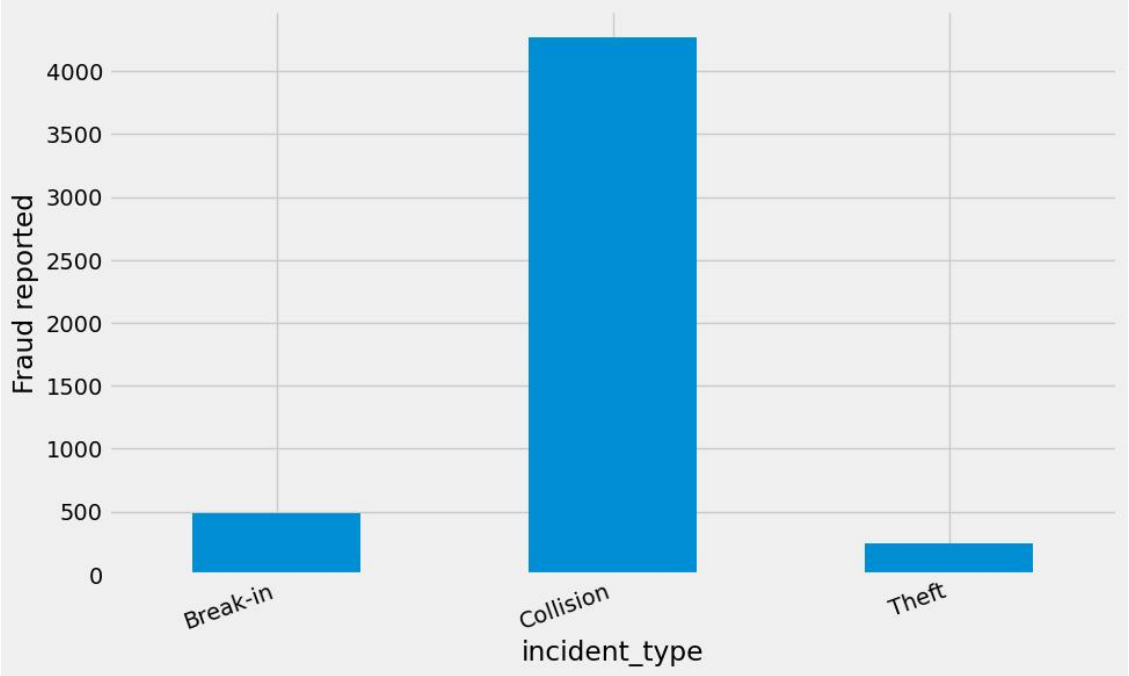
C) Total claim amounts in each month of the year

January,October and November has highest claims compared to other months



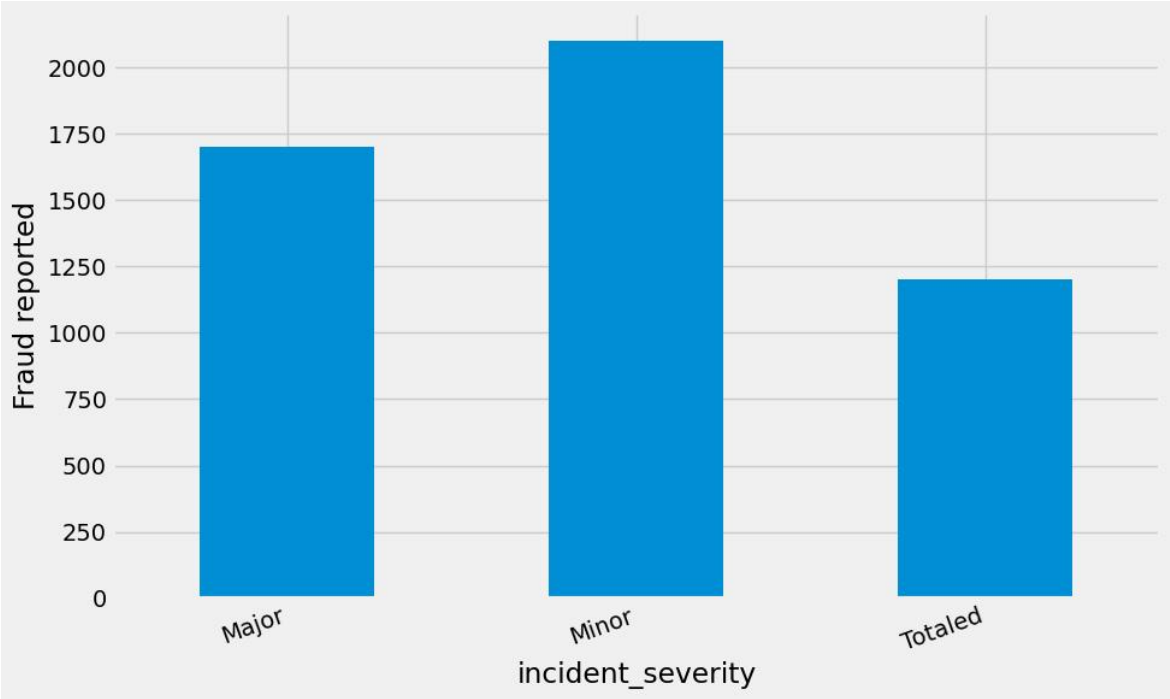
D) Most common type of Incident type of fraud cases

Vehicles claimed to be met with collision is the most fraud cases and least being the theft cases.



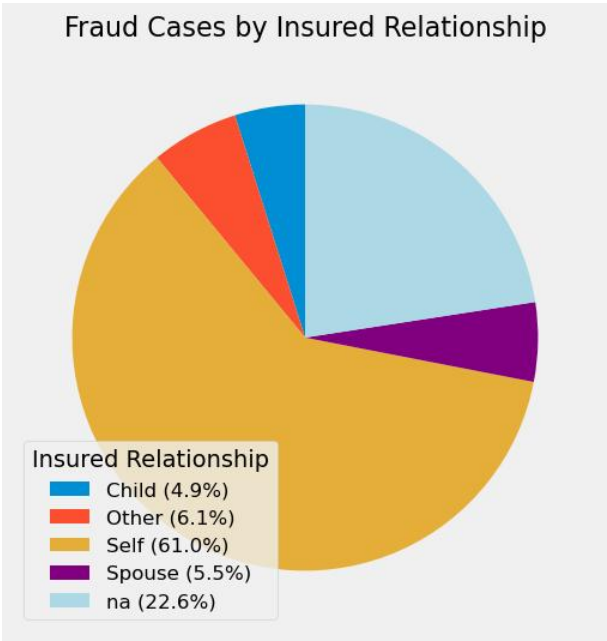
E) <u>Distribution of incident category which is fraud claim.</u>

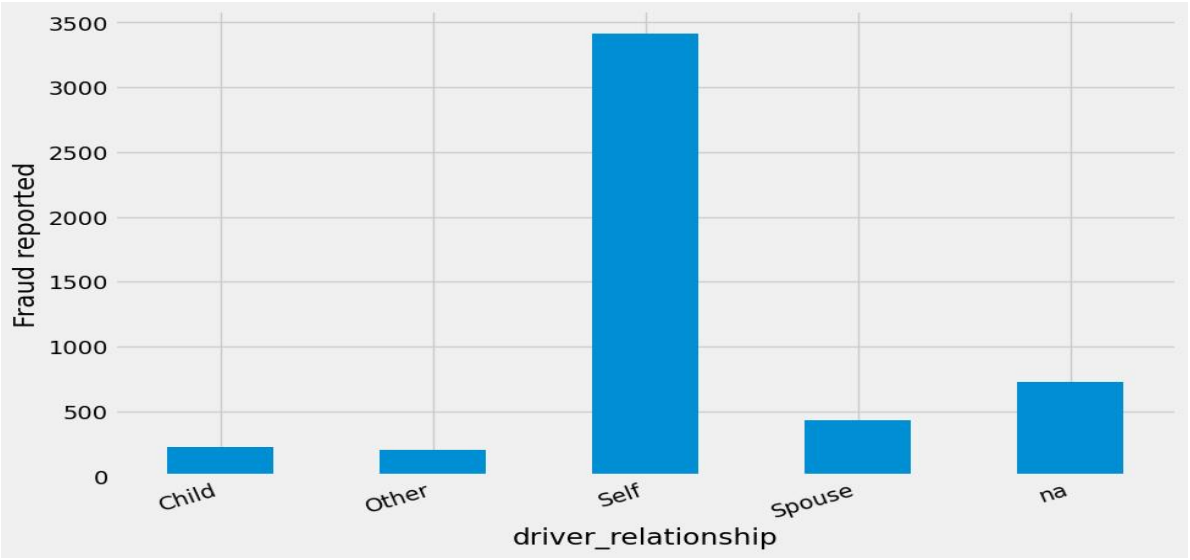
Incident severity category belonging to Minority had most fraud cases, second being Major and least being Totaled.



F) <u>Distribution of driver relationship category which is fraud claim.</u>

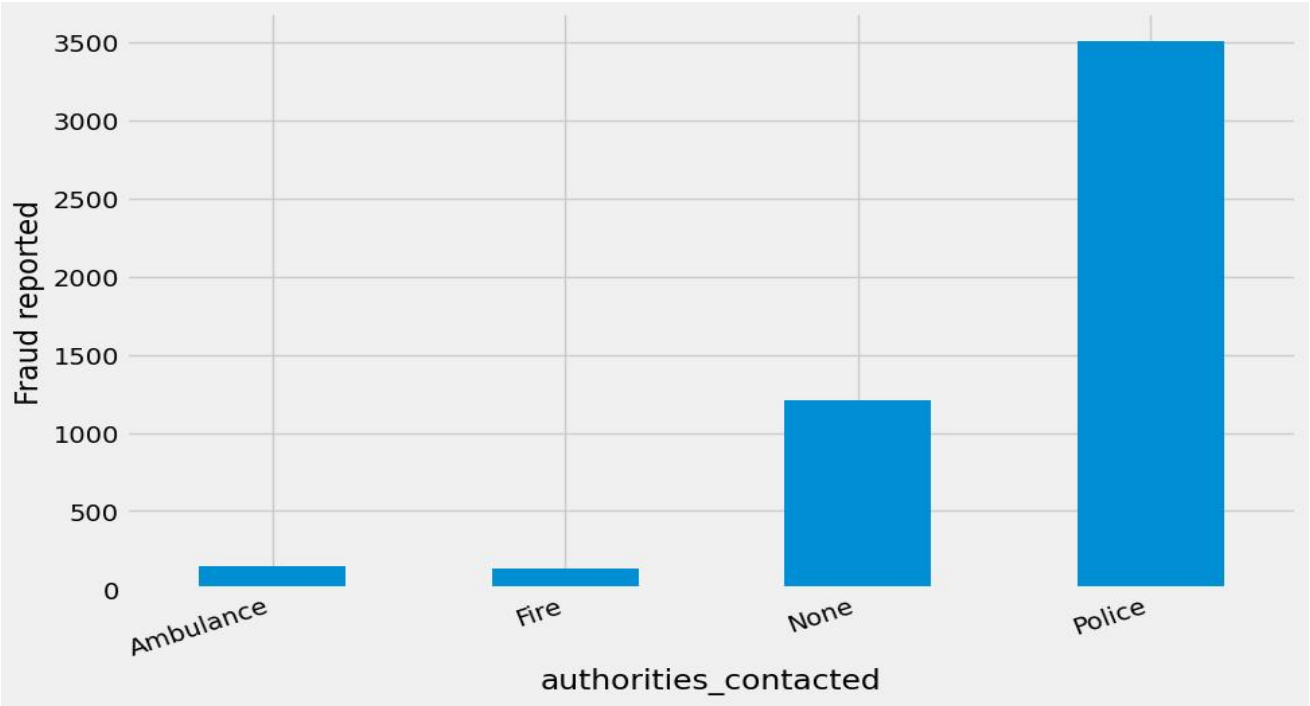
Insurance claims with driver relationship as self has the highest fraudulent cases.





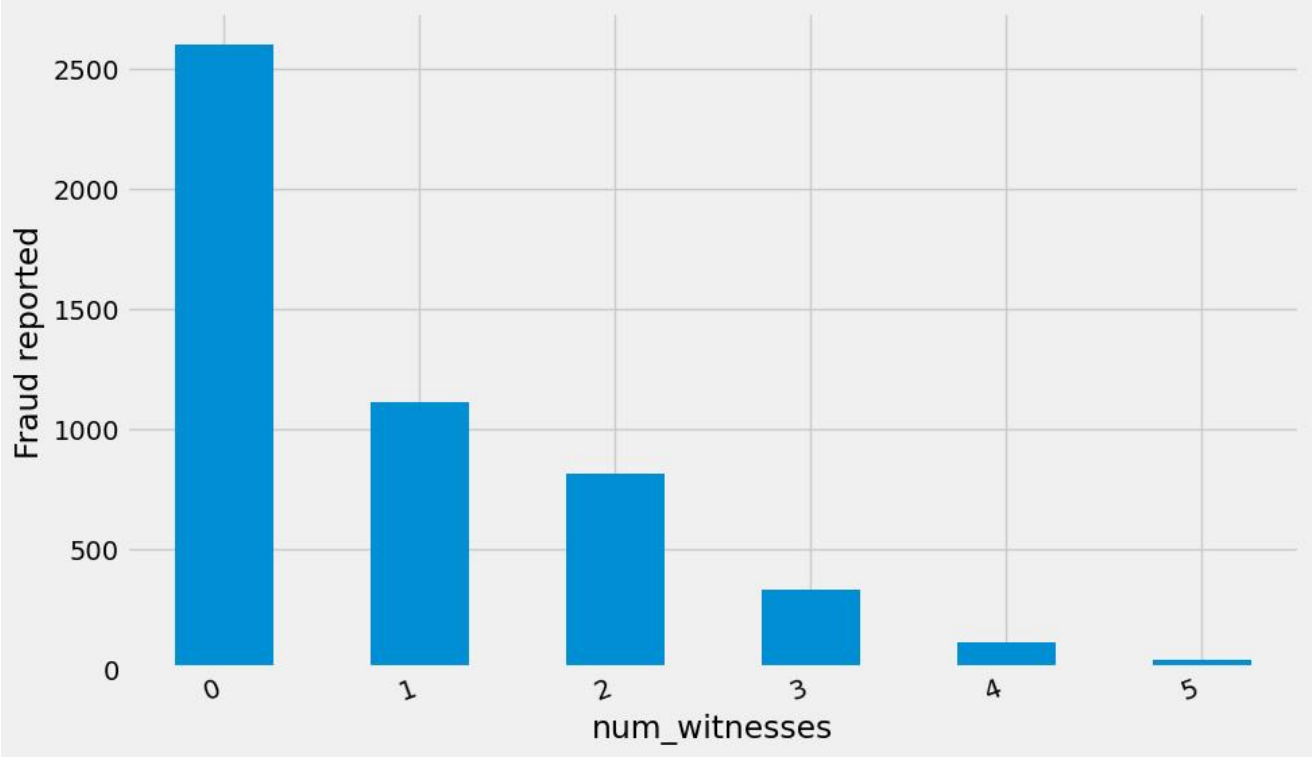
G) <u>Distribution of driver authority category which is fraud claim.</u>

Most fraud claims made were when authorities contacted were Police, which is quite surprising. As this should not be the fraud case.



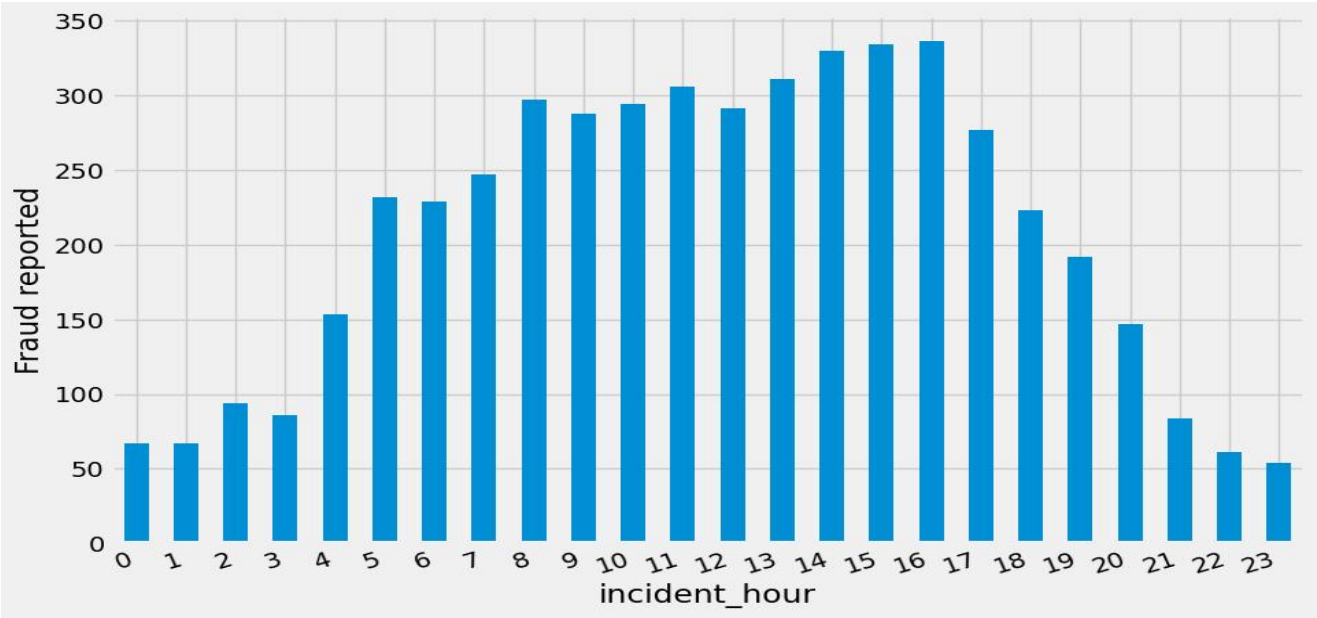
H) <u>Distribution of number of witness category which is fraud claim.</u>

With 0 being the number of witness when fraud cases were filled is maximum which is as expected.



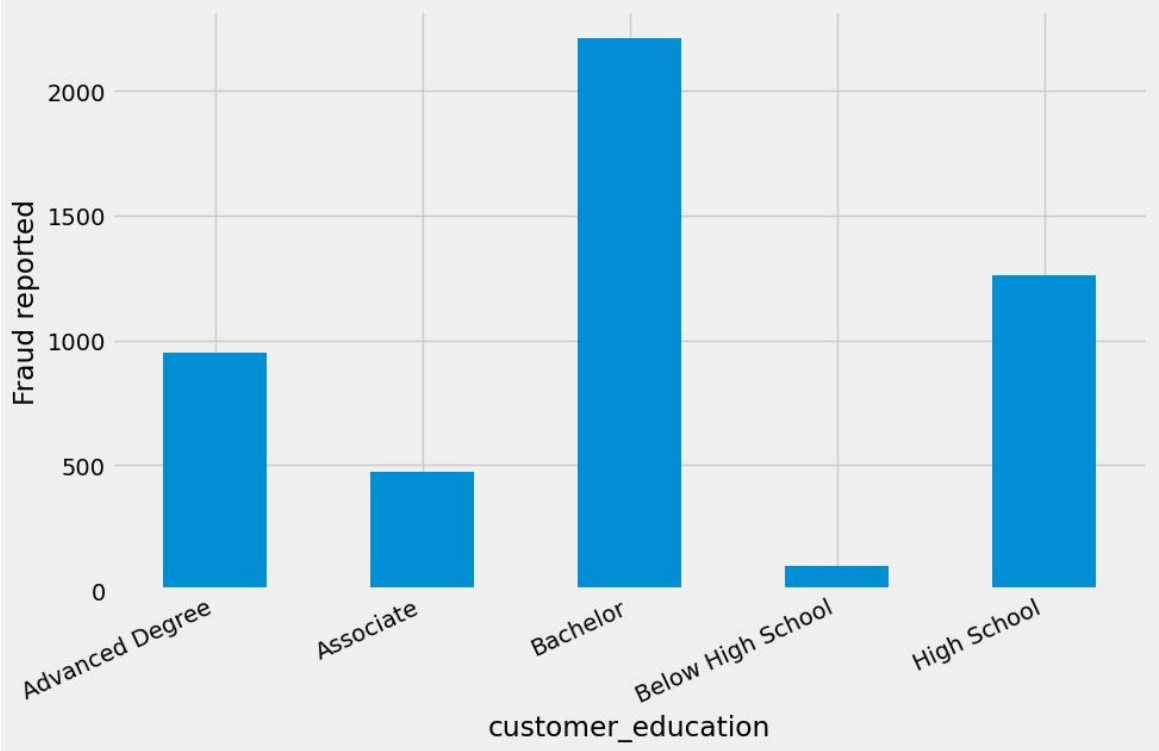
I) <u>Distribution of incident hour category which is fraud claim.</u>

Number of fraudulent cases are considerably high between 8am to 4pm. Least at 9pm-3am.



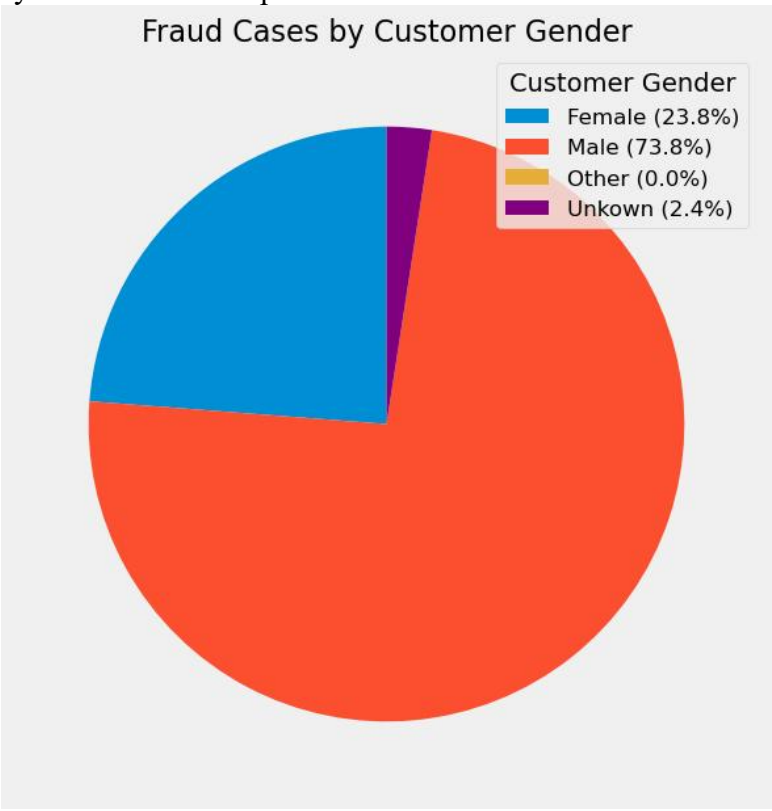
J) <u>Distribution of education category of persons who did fraud claim.</u>

People with Bachelors degree do a maximum fraudulent claims and people with education below high school do less fraud claims.



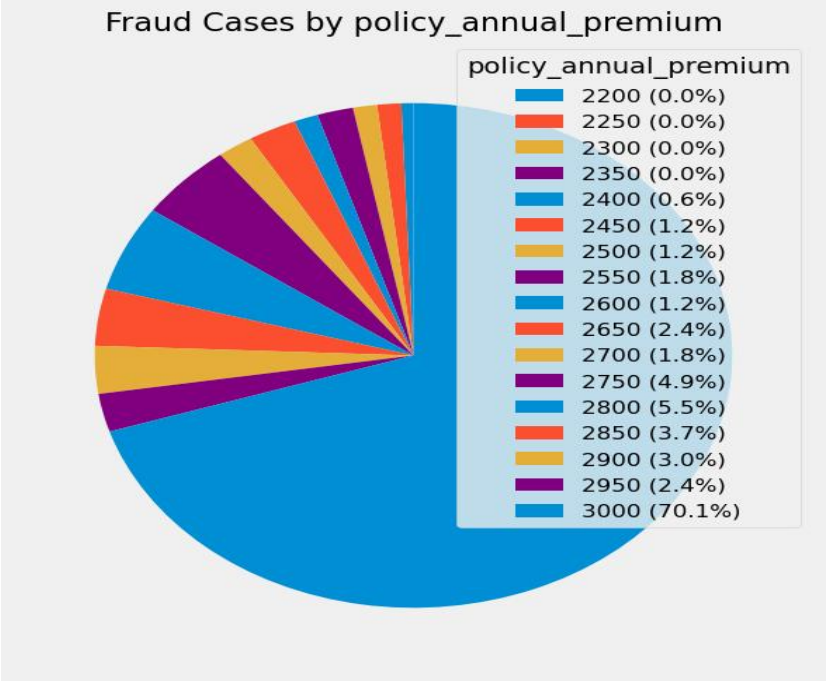
K) <u>Distribution of gender category who did fraud claim.</u>

Males do majority fraud claims compared to females.



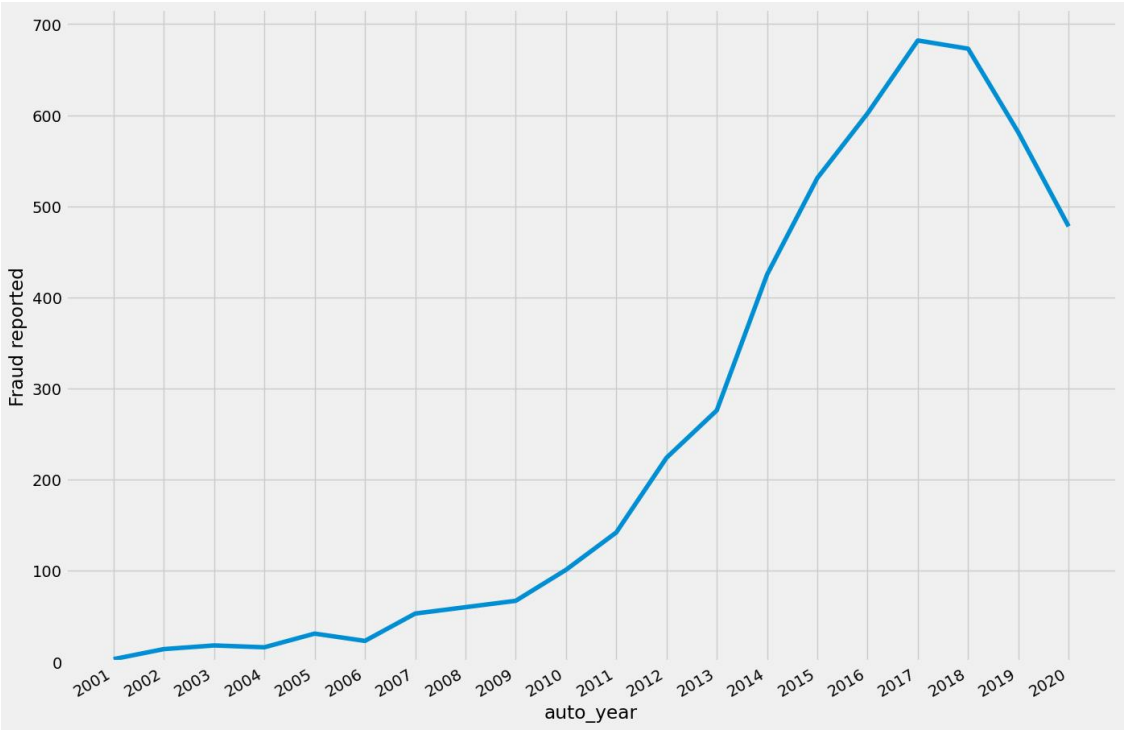
L) <u>Distribution of Annual premium category which has fraud claims.</u>

This describes that Annual Policy with highest Premium gets more fraud claims.



M) <u>Distribution of vehicles make year vs fraud claims.</u>

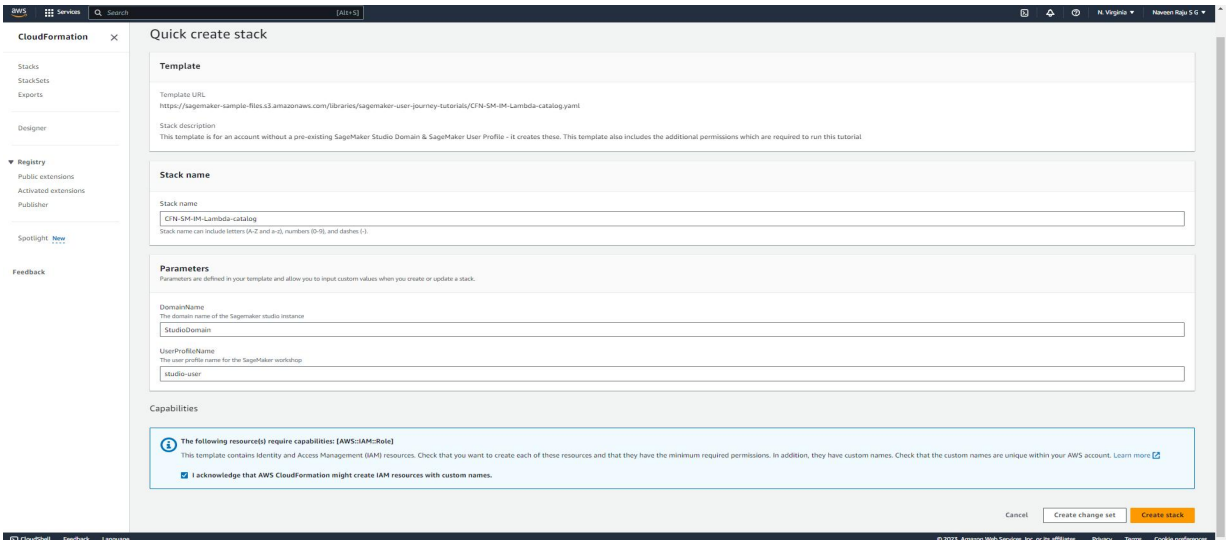
Vehicles with make year starting from 2014 has increasing trend of false claims, with vehicles made in 2017-2018 has highest fraud claims.

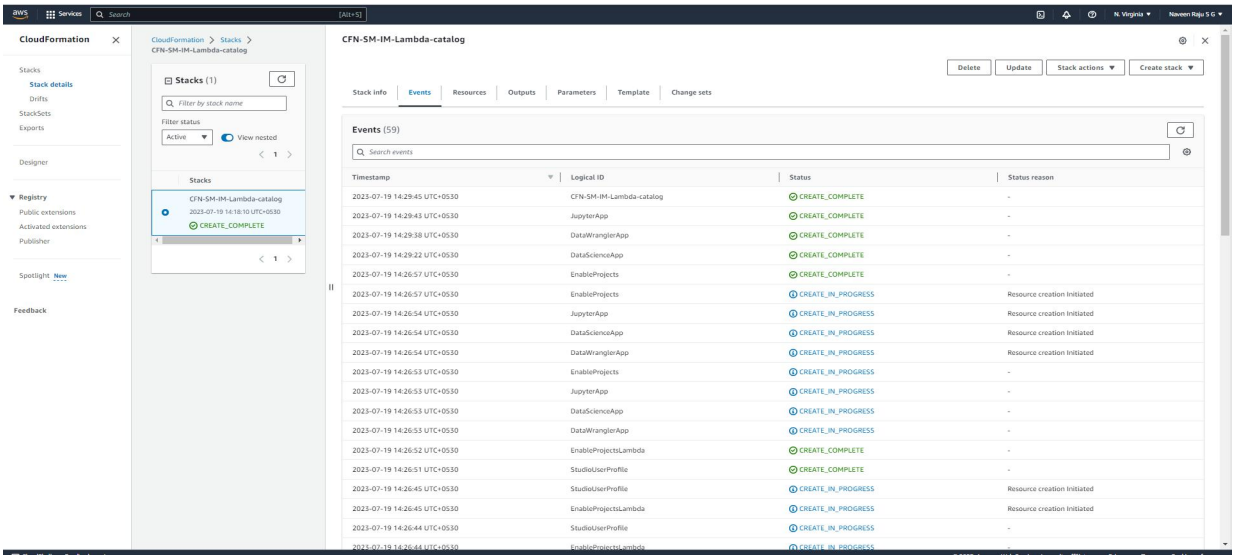


5) Automating Machine Learning Workflows

Step 1 : Setting up SageMaker studio domain

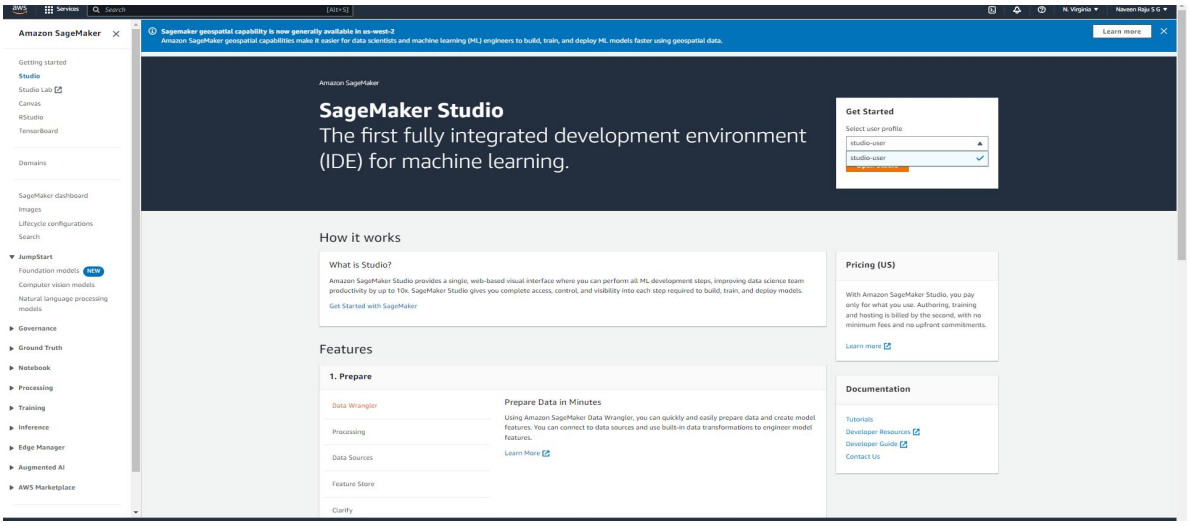
Open AWS Cloud Formation console, enter stack name ‘CFN-SM-IM-Lambda-catalog’ and click create stack, this creates a domain named Studio Domain with user named studio-user. Bascically the stack will have several permissions like ‘AmazonSageMakerFullAccess’ , ‘AWSCloudFormationFullAccess ’, ‘AmazonSageMakerPipelinesIntegrations ’, AWS Lambda related permissions, S3 buckets full access, EC2 instances access.

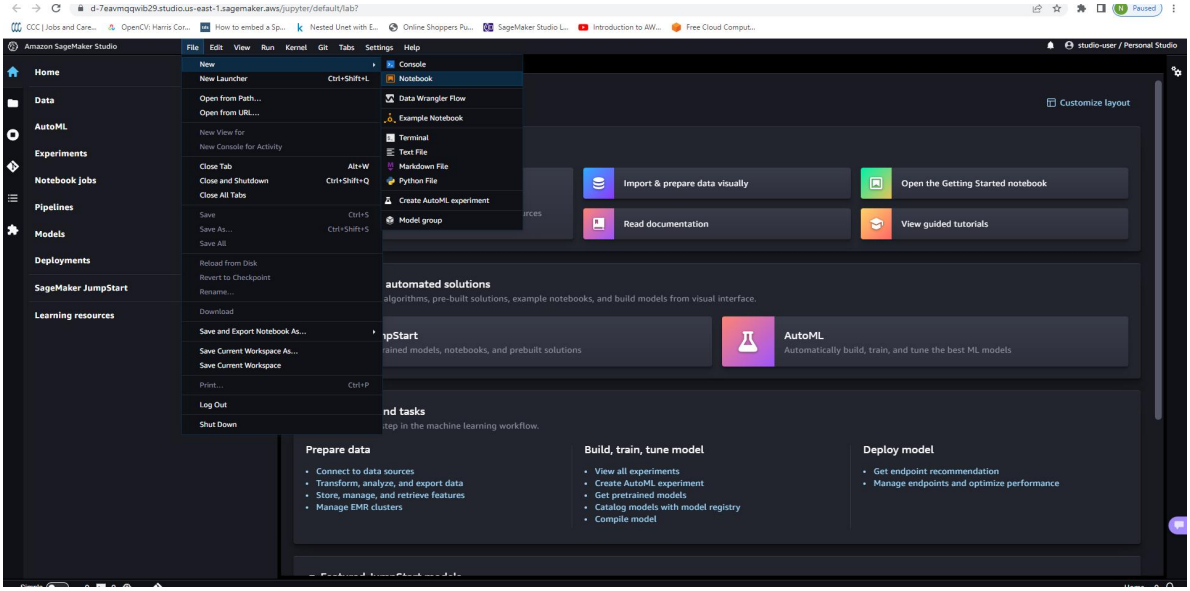


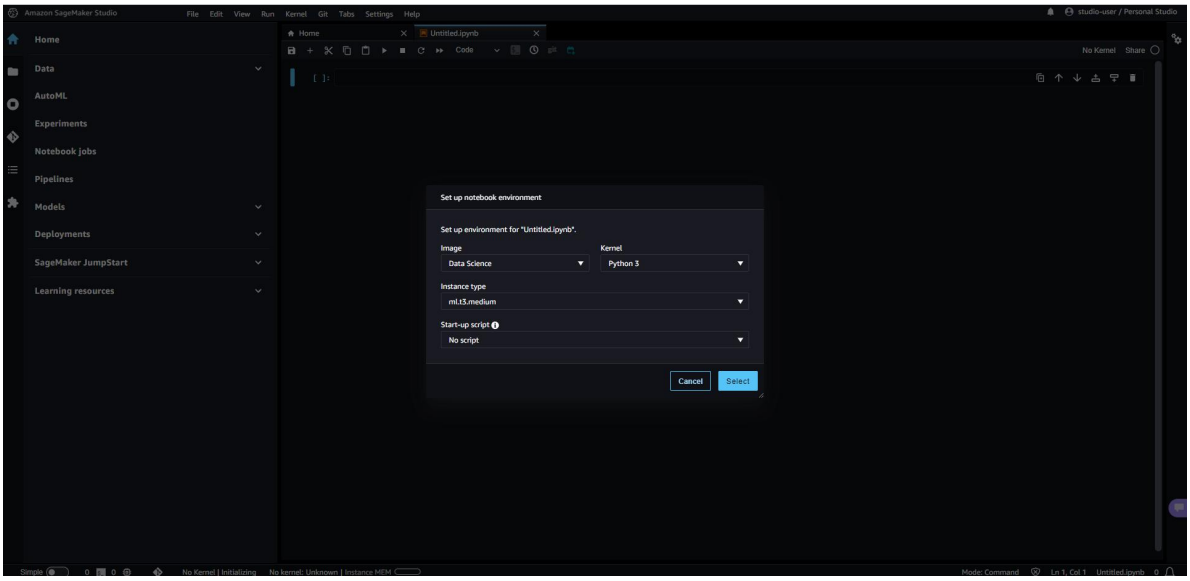


Step 2 : Setting up a SageMaker Studio notebook and parameterize the pipeline

Open SageMaker Studio through AWS console. Select Studio Domain and user studio-user profile and open jupyter notebook with Image - Data Science and Kernel - Python 3.

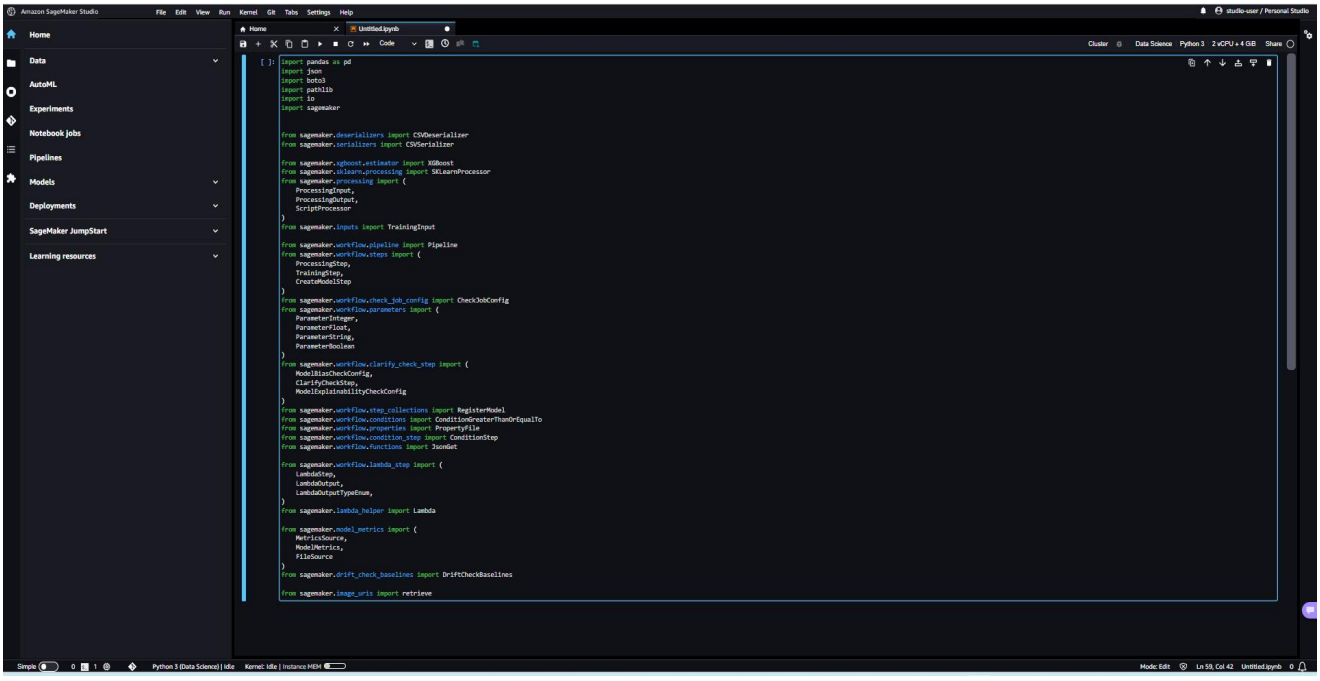




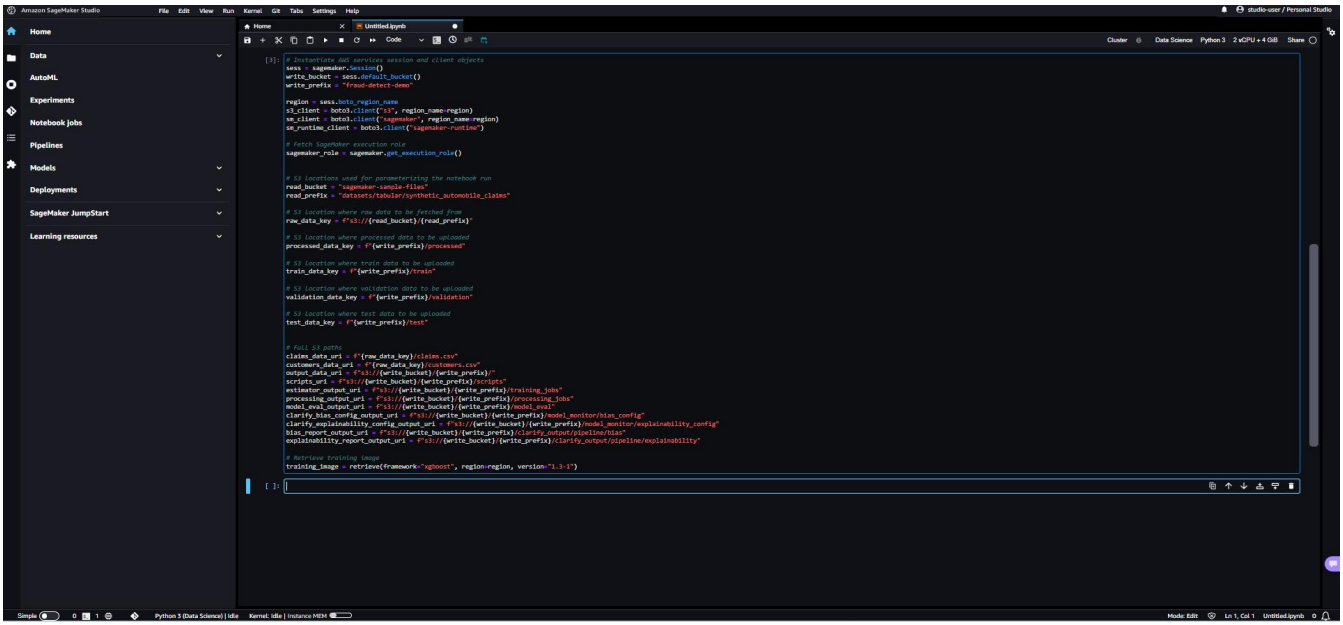


Main.py

Code all the import statements required:

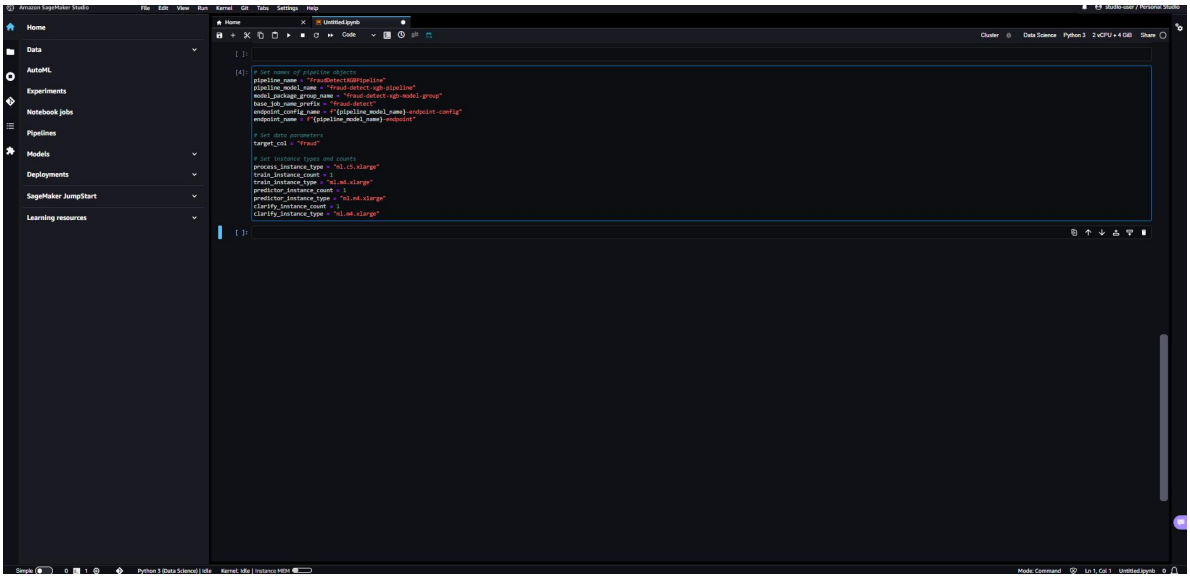


The below code sets up SageMaker and S3 client objects using SageMaker and AWS SDk's. Basically these objects are required to make SageMaker perform actions like deploying, invoking endpoints, and also interact with AWS S3 and AWS Lambda. Also code sets up S3 bucket locations as where original dataset, pre-processed data set and model artifacts need to be present and stored.



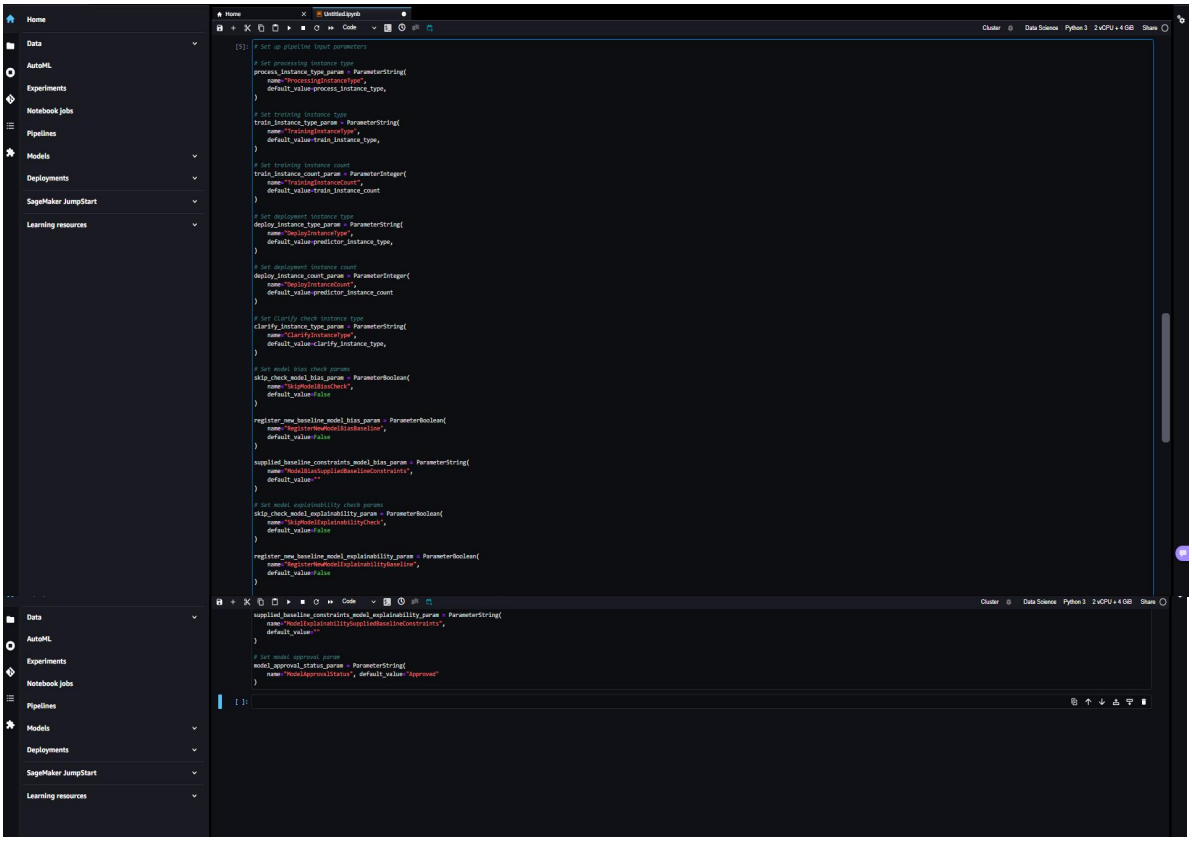
```
# Instantiate AWS services session and client objects
sess = sagemaker.Session()
write_bucket = sess.default_bucket()
write_prefix = "fraud-detect-demo"

region = sess.boto_region_name
s3_client = boto3.client("s3", region_name=region)
sm_client = boto3.client("sagemaker", region_name=region)
sm_runtime_client = boto3.client("sagemaker-runtime")

# Fetch SageMaker execution role
sagemaker_role = sagemaker.get_execution_role()


# S3 locations used for parameterizing the notebook run
read_bucket = "sagemaker-sample-files"
read_prefix = "datasets/tabular/synthetic_automobile_claims"

# S3 location where raw data to be fetched from
raw_data_key = f"s3://{read_bucket}/{read_prefix}"

# S3 location where processed data to be uploaded
processed_data_key = f"{write_prefix}/processed"

# S3 location where train data to be uploaded
train_data_key = f"{write_prefix}/train"

# S3 location where validation data to be uploaded
validation_data_key = f"{write_prefix}/validation"

# S3 location where test data to be uploaded
test_data_key = f"{write_prefix}/test"


# Full S3 paths
claims_data_uri = f"{raw_data_key}/claims.csv"
customers_data_uri = f"{raw_data_key}/customers.csv"
output_data_uri = f"s3://{write_bucket}/{write_prefix}/"
scripts_uri = f"s3://{write_bucket}/{write_prefix}/scripts"
estimator_output_uri = f"s3://{write_bucket}/{write_prefix}/training_jobs"
processing_output_uri = f"s3://{write_bucket}/{write_prefix}/processing_jobs"
model_eval_output_uri = f"s3://{write_bucket}/{write_prefix}/model_eval"
clarify_bias_config_output_uri = f"s3://{write_bucket}/{write_prefix}/model_monitor/bias_config"
clarify_explainability_config_output_uri = f"s3://{write_bucket}/{write_prefix}/model_monitor/explainability_config"
bias_report_output_uri = f"s3://{write_bucket}/{write_prefix}/clarify_output/pipeline/bias"
explainability_report_output_uri = f"s3://{write_bucket}/{write_prefix}/clarify_output/pipeline/explainability"

# Retrieve training image
training_image = retrieve(framework="xgboost", region=region, version="1.3-1")
```

The below code contains names of several SageMaker pipeline components, training and inference instance types and counts.

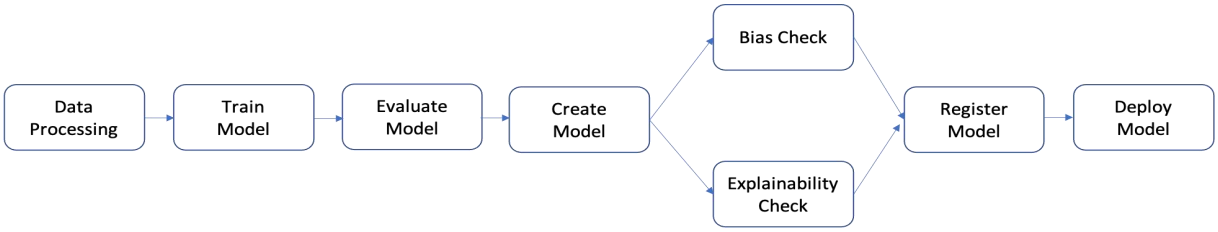


```
# Set names of pipeline objects
pipeline_name = "FraudDetectXGBPipeline"
pipeline_model_name = "fraud-detect-xgb-pipeline"
model_package_group_name = "fraud-detect-xgb-model-group"
base_job_name_prefix = "fraud-detect"
endpoint_config_name = f"{pipeline_model_name}-endpoint-config"
endpoint_name = f"{pipeline_model_name}-endpoint"

# Set data parameters
target_col = "fraud"

# Set instance types and counts
process_instance_type = "ml.c5.xlarge"
train_instance_count = 1
train_instance_type = "ml.m4.xlarge"
predictor_instance_count = 1
predictor_instance_type = "ml.m4.xlarge"
clarify_instance_count = 1
clarify_instance_type = "ml.m4.xlarge"
```

The below code contains parameterization I.e setting up pipeline input parameters like processing instance types, training instance type and count, deployment instance type and count, Clarify check instance and count, model bias check parameter, registering new baseline model bias parameter, model explain ability, model registration parameter, model approval parameter check parameter this allows to specify input parameters at run time without changing pipeline code.
This is achieved by using modules available in sagemaker.workflow.parameters module, those modules are ParameterInteger, ParameterFloat, ParameterString, and ParameterBoolean.



Step 3 : Building the pipeline components



SageMaker encodes the pipeline in a Directed Acyclic Graph (DAG), where each node represents a step and the connections between the nodes represent dependencies.

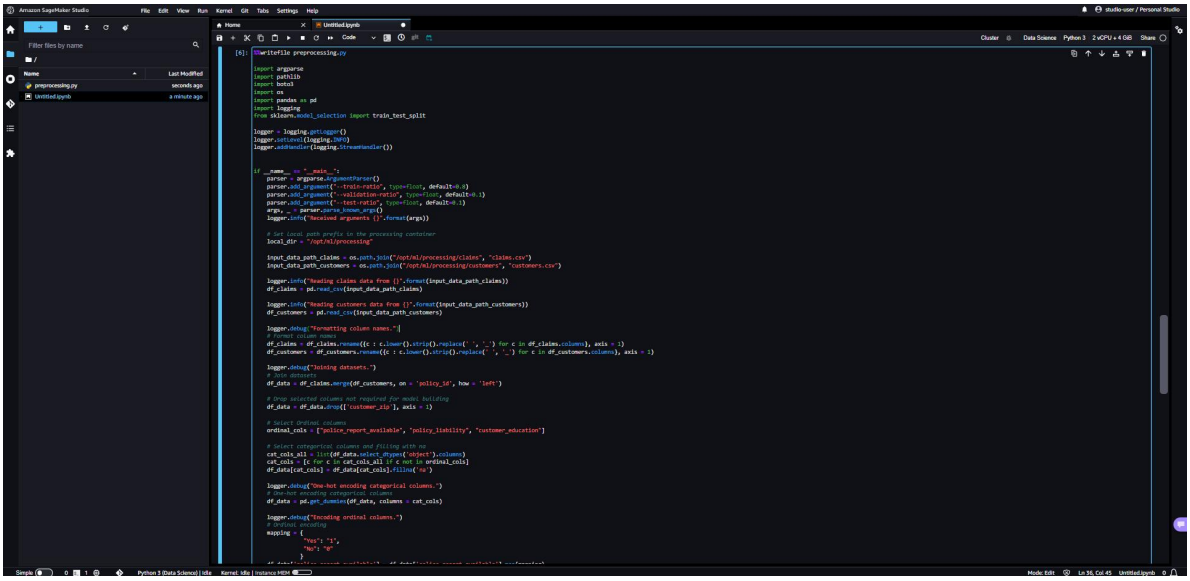
Read original data sets, Preprocess data and do train,test,val data split.

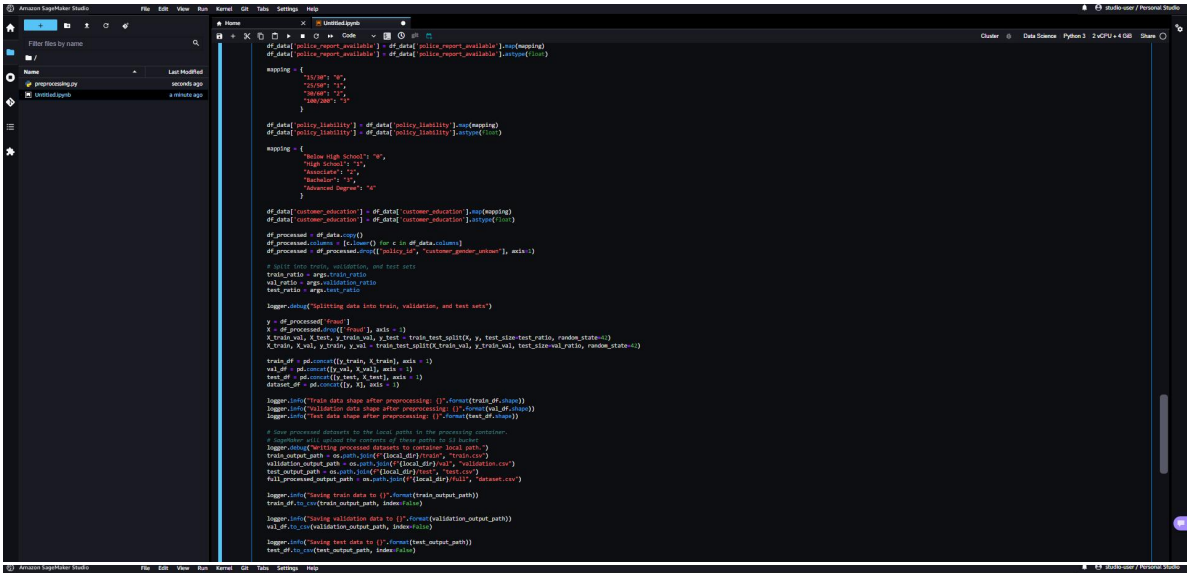
Read dataset : Read claims.csv and customers.csv dataset

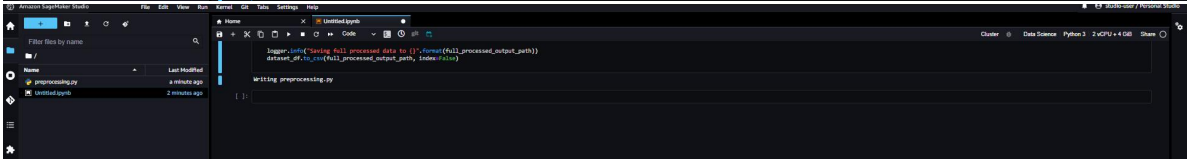
Pre-process dataset:

- make column names to lowercase and replace “ “ in column names with”_”

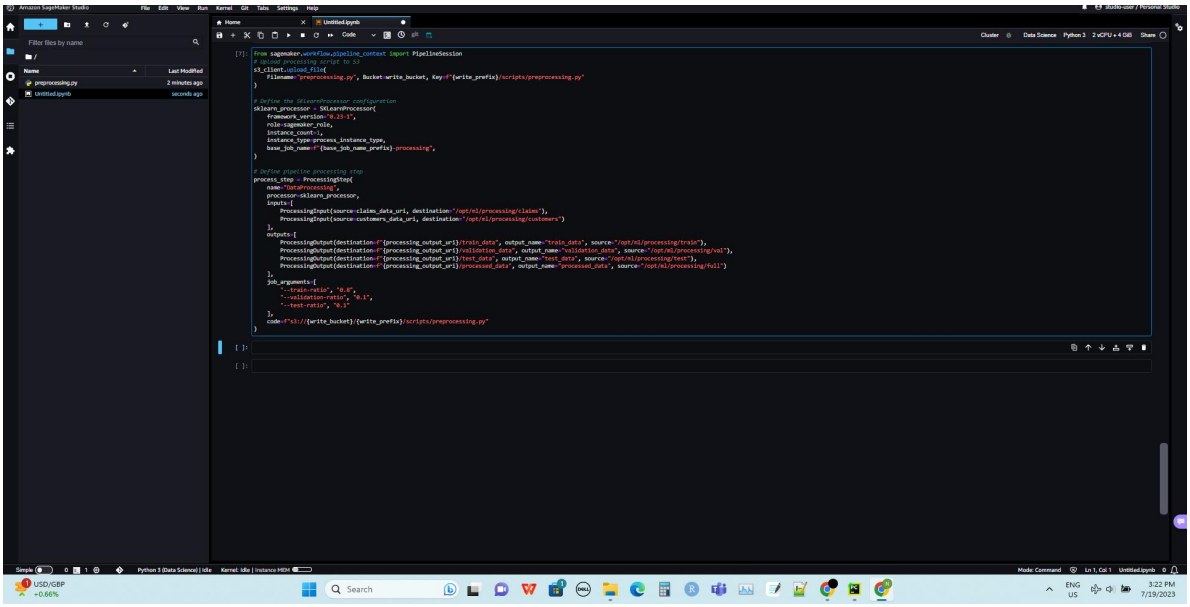
- Do left inner join of 2 datasets using column “policy_id”
- Drop customer_zip column
- Full missing values with‘na’
- One-hot vector encoding of only categorical columns
- Map ordinal columns with user defined ordinal numerical values
- drop policy_id and customer_gender_unkown columns
- Do train test and validation split based on 80:10:10 ratio and store respective files in respective s3 buckets.





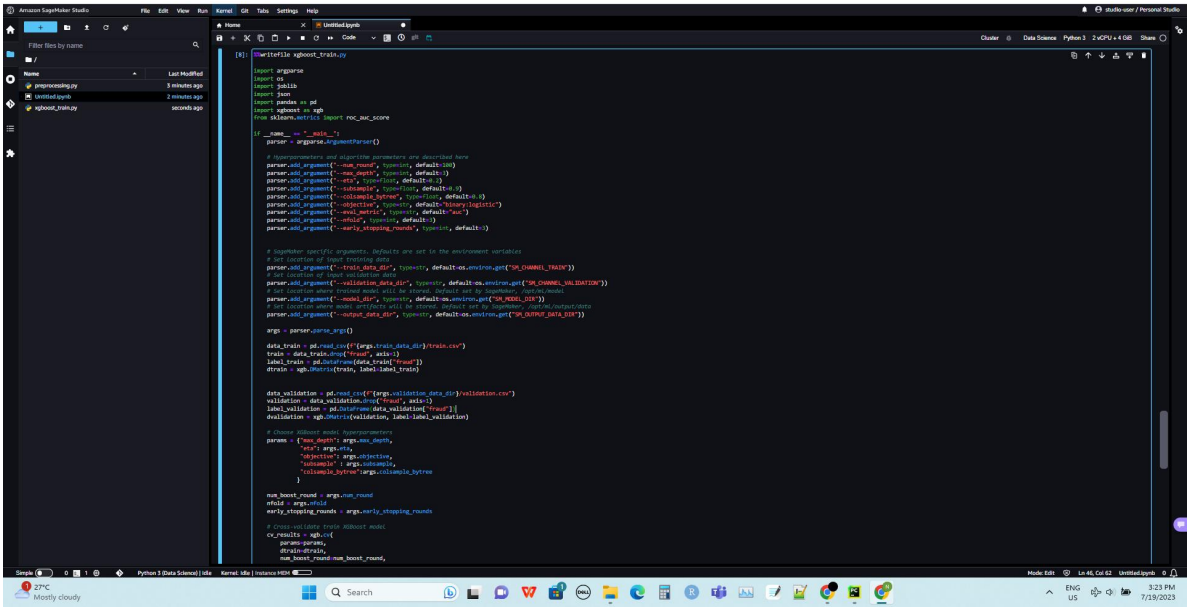


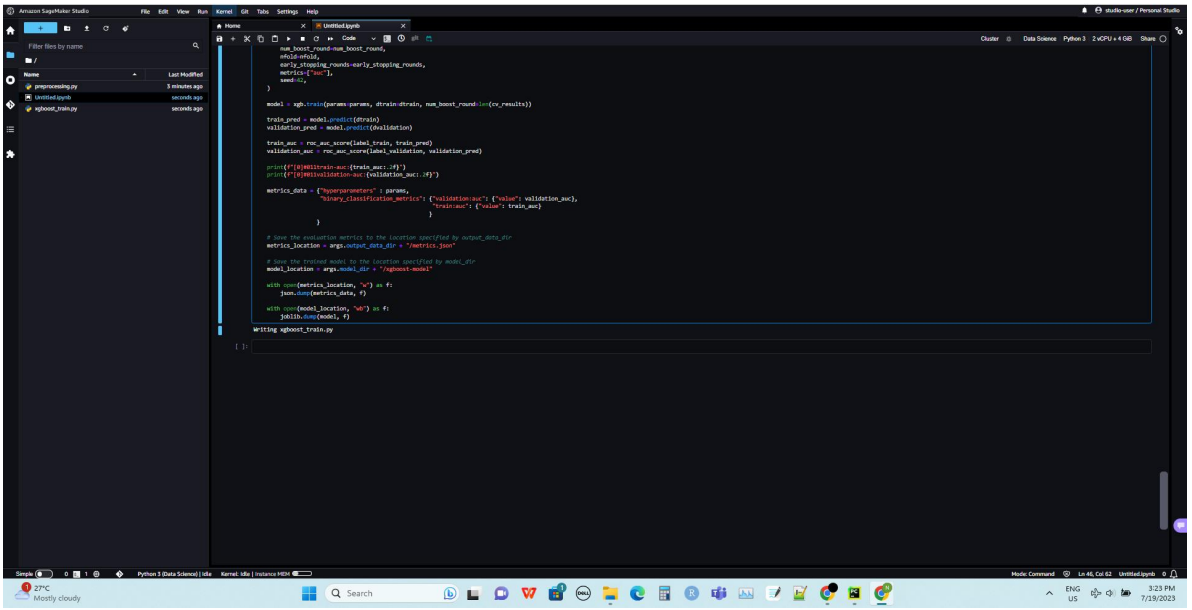
The following code will instantiate processor and SageMaker pipeline step to execute pre-processing script. Because pre-processing script is written in Pandas we use SKLearnProcessor.From this code SageMaker PipelinesStep will take arguments like processor, s3 locations of dataset, output s3 locations to save pre-processed dataset, train,test and validation split ratio.



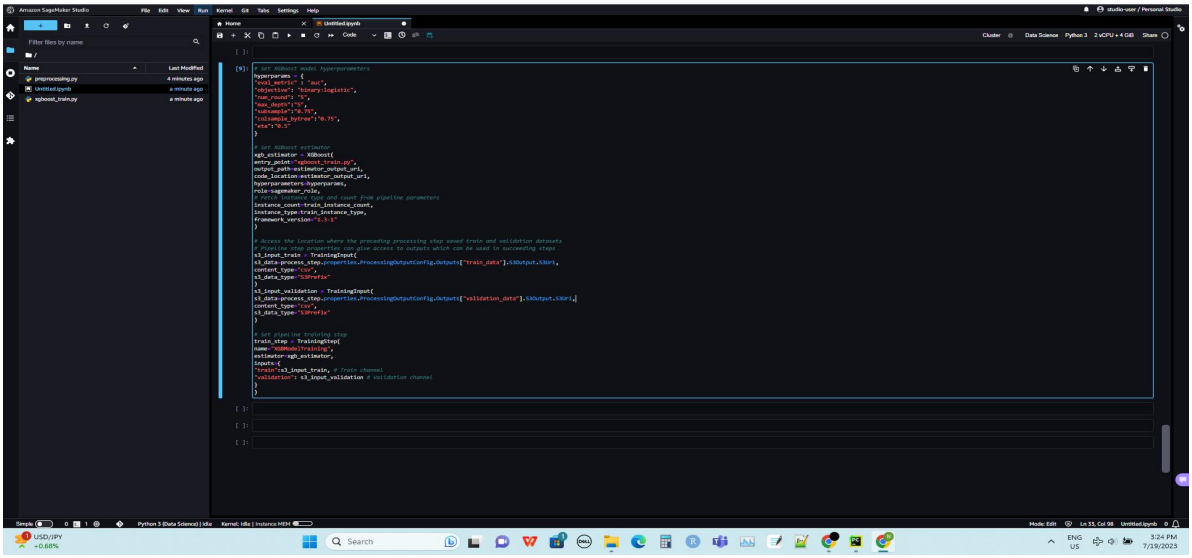
Training step

The following code contains training code of XGBoost binary classifier and hyperparameters, roc_auc score calculation for train and validation data, saving metrics calculated to specified s3 bucket, saving trained model to specified s3 bucket.

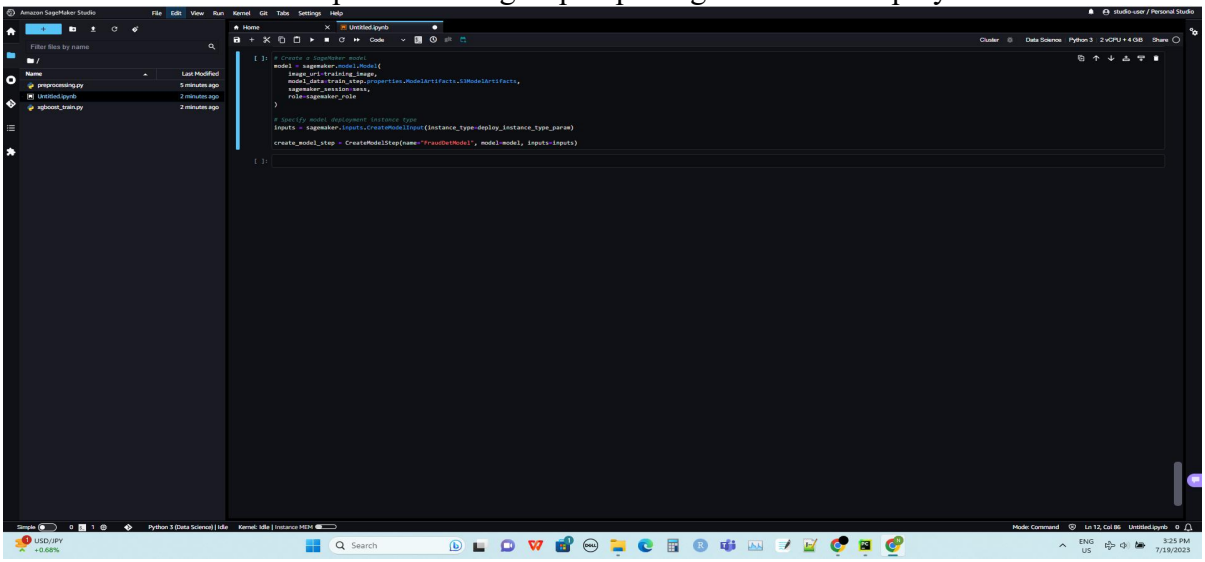




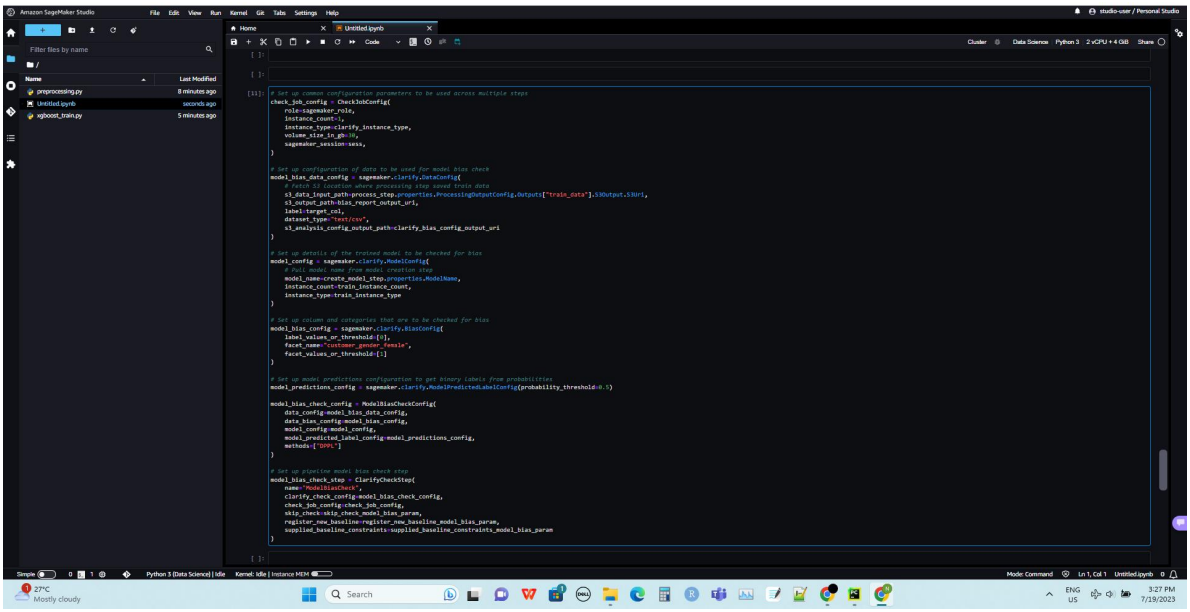
Below code contains setting up the model training using SageMaker XGBoost estimator and SageMaker pipelines training step function.



The below code will create SageMaker model using SageMaker Pipelines CreateModelStep function.This will use output of training step to package model for deployment.

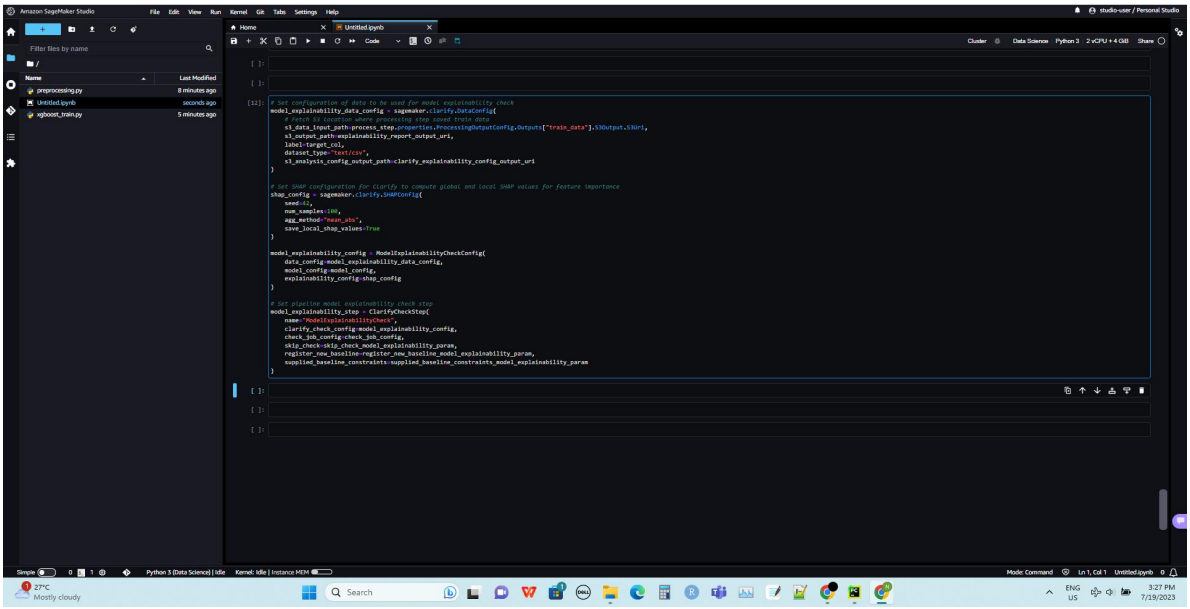


The code below contains data bias check. SageMaker uses SageMaker Clarify for model bias check. It uses training data and model created in previous steps. Model bias check calculates bias metric named (Difference in Positive Proportions in Predicted Labels)DPPL.



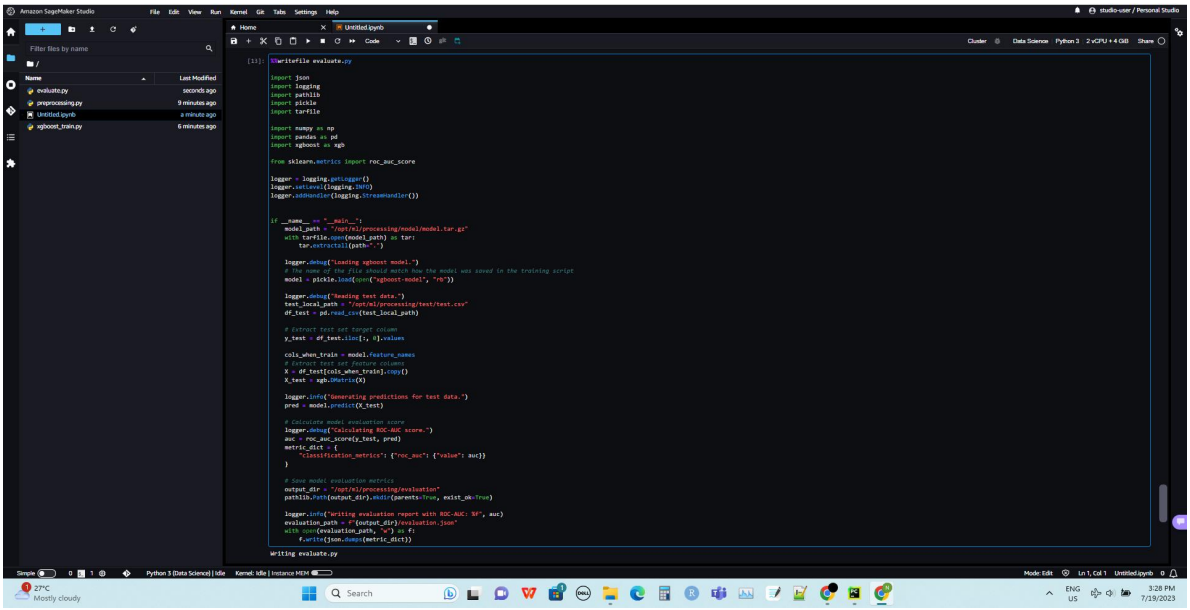
```
# Set up common configuration parameters to be used across multiple steps
check_job_config = CheckJobConfig(
    role=sagemaker_role,
    instance_count=1,
    instance_type=clarify_instance_type,
    volume_size_in_gb=30,
    sagemaker_session=sess,
)

# Set up configuration of data to be used for model bias check
model_bias_data_config = sagemaker.clarify.DataConfig(
    # Fetch S3 location where processing step saved train data
    s3_data_input_path=process_step.properties.ProcessingOutputConfig.Outputs["train_data"].S3Output.S3Uri,
    s3_output_path=bias_report_output_uri,
    label=target_col,
    dataset_type="text/csv",
    s3_analysis_config_output_path=clarify_bias_config_output_uri
)

# Set up details of the trained model to be checked for bias
model_config = sagemaker.clarify.ModelConfig(
    # Full model name from model creation step
    model_name=create_model_step.properties.ModelName,
    instance_count=train_instance_count,
    instance_type=train_instance_type
)

# Set up column and categories that are to be checked for bias
model_bias_config = sagemaker.clarify.BiasConfig(
    label_values_or_threshold=[0],
    facet_name="customer_gender_female",
    facet_values_or_threshold=[1]
)

# Set up model predictions configuration to get binary labels from probabilities
model_predictions_config = sagemaker.clarify.ModelPredictedLabelConfig(probability_threshold=0.5)

model_bias_check_config = ModelBiasCheckConfig(
    data_config=model_bias_data_config,
    data_bias_config=model_bias_config,
    model_config=model_config,
    model_predicted_label_config=model_predictions_config,
    methods=["DPPL"]
)

# Set up pipeline model bias check step
model_bias_check_step = ClarifyCheckStep(
    name="ModelBiasCheck",
    clarify_check_config=model_bias_check_config,
    check_job_config=check_job_config,
    skip_check=skip_check_model_bias_param,
    register_new_baseline=register_new_baseline_model_bias_param,
    supplied_baseline_constraints=supplied_baseline_constraints_model_bias_param
)
```

The below code is for model explain ability check which provides insights such as feature importance.

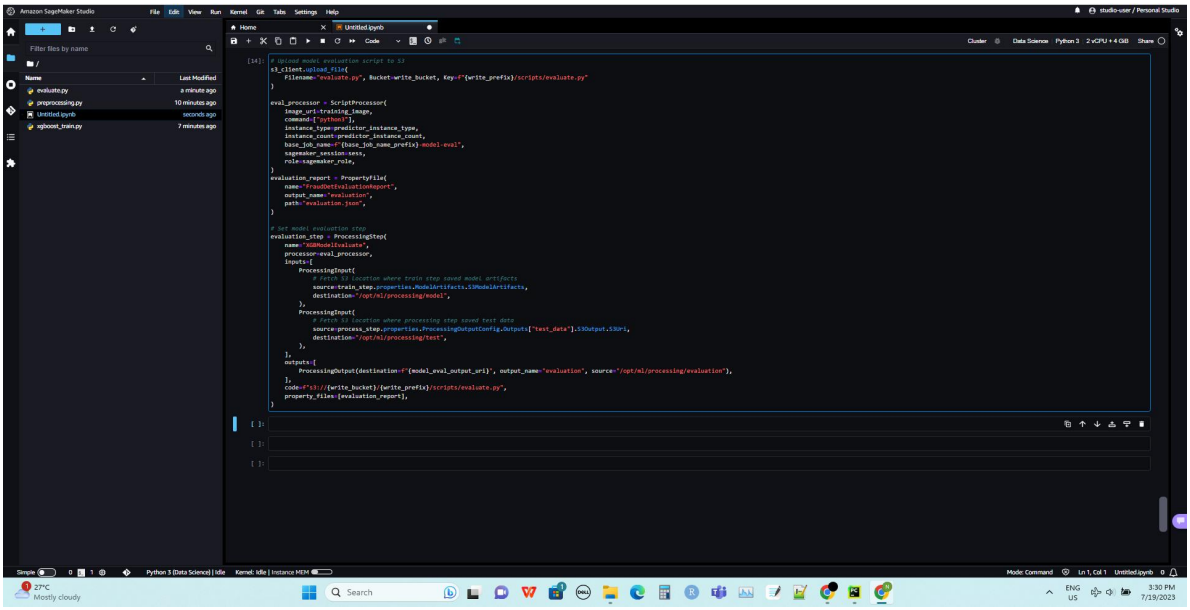


```
# Set configuration of data to be used for model explainability check
model_explainability_data_config = sagemaker.clarify.DataConfig(
    # Fetch S3 location where processing step saved train data
    s3_data_input_path=process_step.properties.ProcessingOutputConfig.Outputs["train_data"].S3Output.S3Uri,
    s3_output_path=explainability_report_output_uri,
    label=target_col,
    dataset_type="text/csv",
    s3_analysis_config_output_path=clarify_explainability_config_output_uri
)

# Set SHAP configuration for Clarify to compute global and local SHAP values for feature importance
shap_config = sagemaker.clarify.SHAPConfig(
    seed=42,
    num_samples=100,
    agg_method="mean_abs",
    save_local_shap_values=True
)

model_explainability_config = ModelExplainabilityCheckConfig(
    data_config=model_explainability_data_config,
    model_config=model_config,
    explainability_config=shap_config
)

# Set pipeline model explainability check step
model_explainability_step = ClarifyCheckStep(
    name="ModelExplainabilityCheck",
    clarify_check_config=model_explainability_config,
    check_job_config=check_job_config,
    skip_check=skip_check_model_explainability_param,
    register_new_baseline=register_new_baseline_model_explainability_param,
    supplied_baseline_constraints=supplied_baseline_constraints_model_explainability_param
)
```

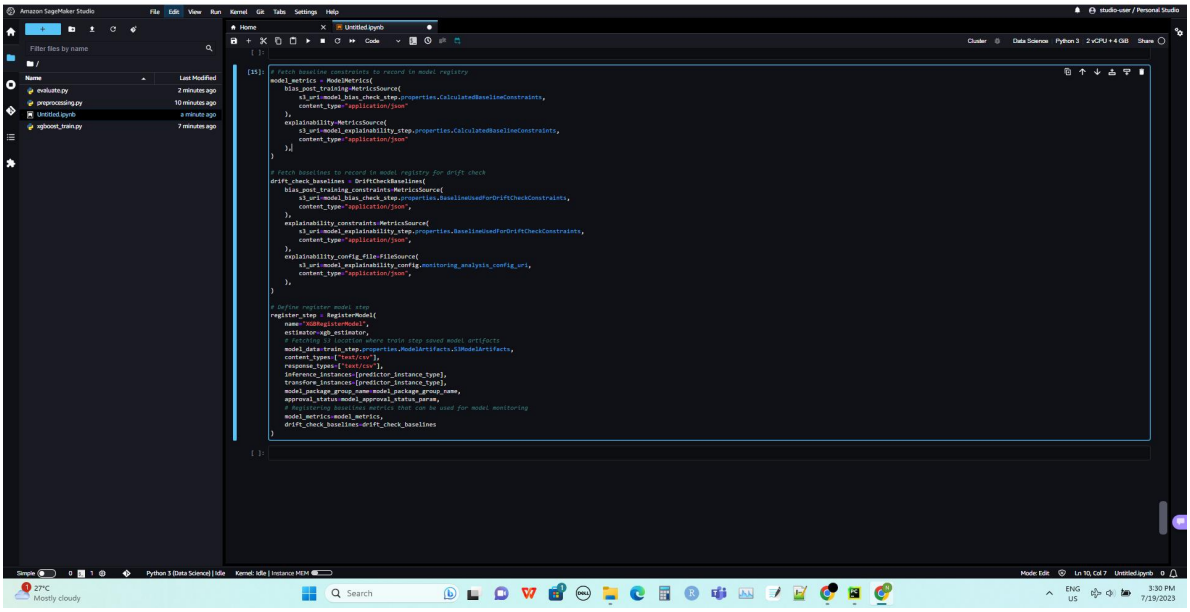
The below code is used for the following: before model is to be registered and deployed it should be verified if the model is performing better hence here we calculate AUC-ROC value on test data set only after it meets a set threshold value, model is accepted.



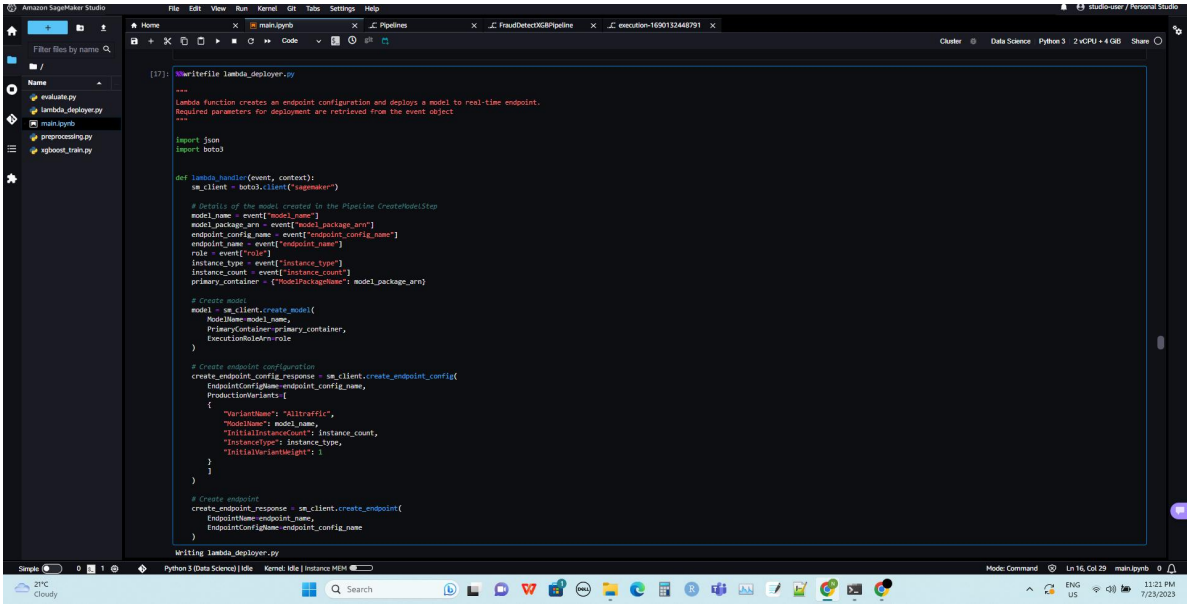
The below code instantiate processor and SageMaker pipeline step to run evaluation script. Since we have write custom evaluation script we use ScriptProcessor. The Processing step takes following arguments: processor, s3 bucket location of test data set, model artifacts, output location to store evaluation results. Additionally we use property_files argument to store information from output of processing step as json file which later will be used in conditional step.



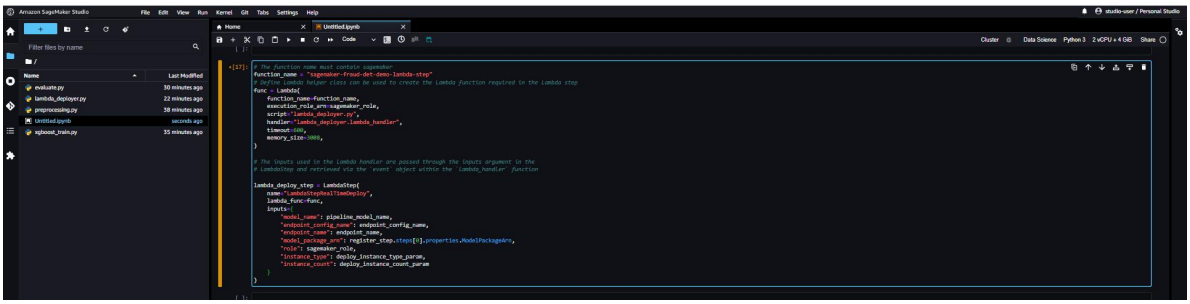
The following codes sets up model registry step. Model registry is used to create model catalogs, manage model versions which can be used to selectively deploy specific model. It uses model_metrics and drift_check_baselines calculated in previous steps using which it configure the baselines associated with the model so they can be used in drift checks and model monitoring jobs.



Below code is used to deploy model for inference using lambda step function. Lambda step function provide integration with AWS Lambda. Lambda step function is for lightweight model deployment and with maximum run time of 10 min. Here we use high-level SageMaker Python SDK which has a Lambda helper convenience class that can be used to create a new Lambda function along with other code defining your pipeline.This codes defines Lambda handler function and a script to take model attributes, such as model name, and deploys to a real-time endpoint.



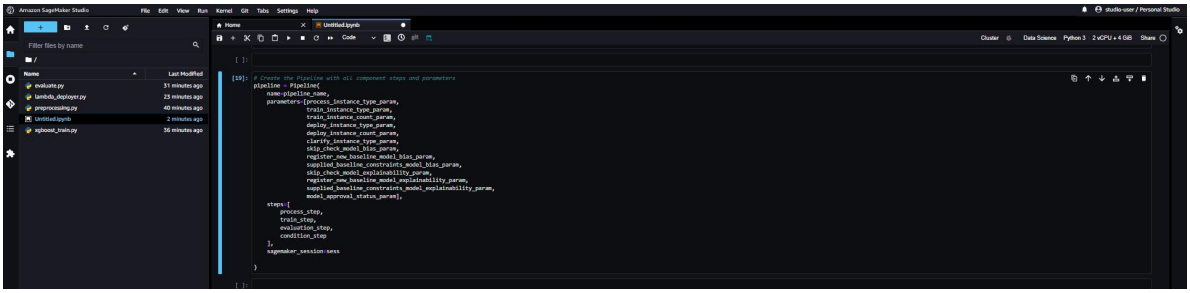
The below code create LambdaStep which takes all parameters such as model, endpoint name, and deployment instance type and count are provided using the inputs argument.



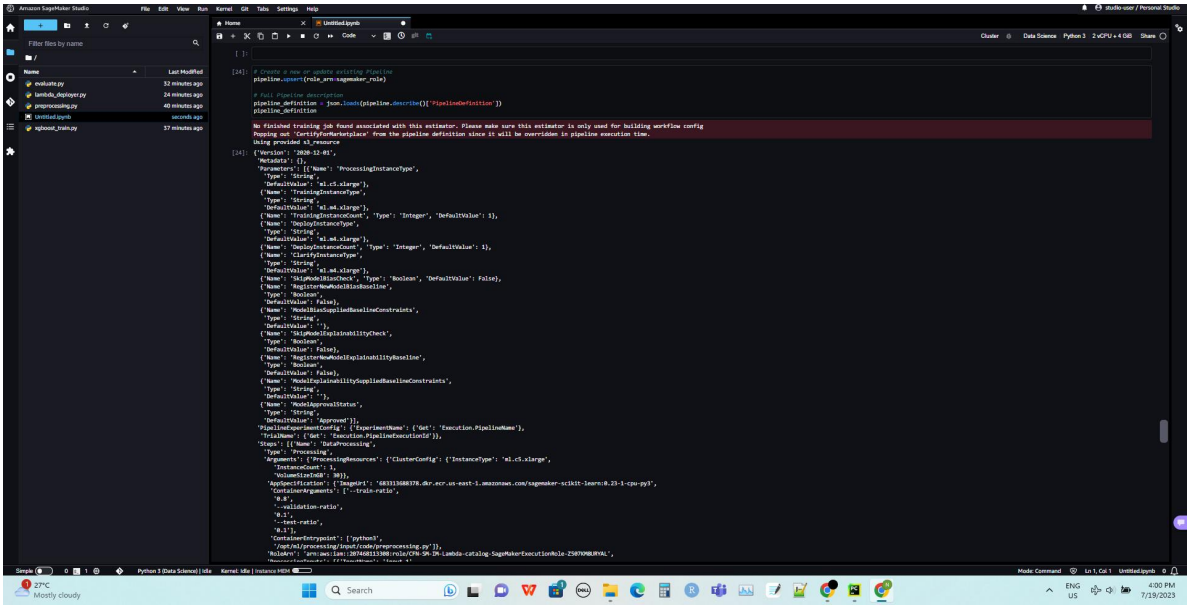
The below code define ConditionStep which compares current model's performance based on AUC-ROC metric. Only if the metric value is greater than AUC - 0.7 the pipeline will perform bias and explain ability checks, register model and deploy it.

Step 4: Build and run the pipeline

The below code sets up the pipeline. The pipeline definition consists of all parameters of step2 and list of all component steps. But steps such as create model, bias and explainability checks, model registration, and lambda deployment are not listed in the pipeline definition because they do not run unless the conditional step evaluates to true.If conditional step is true, the subsequent steps are run in order based on their specified inputs and outputs.

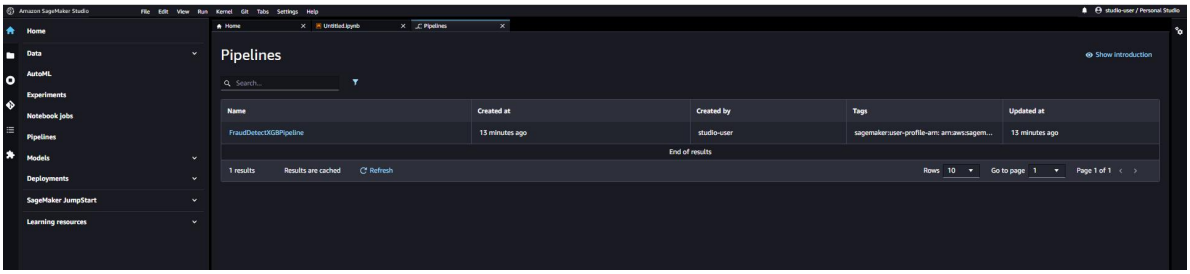


The below code is used to update pipeline if some pipeline already exists or else create new.

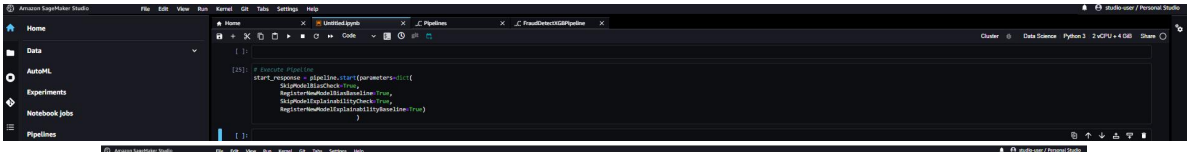


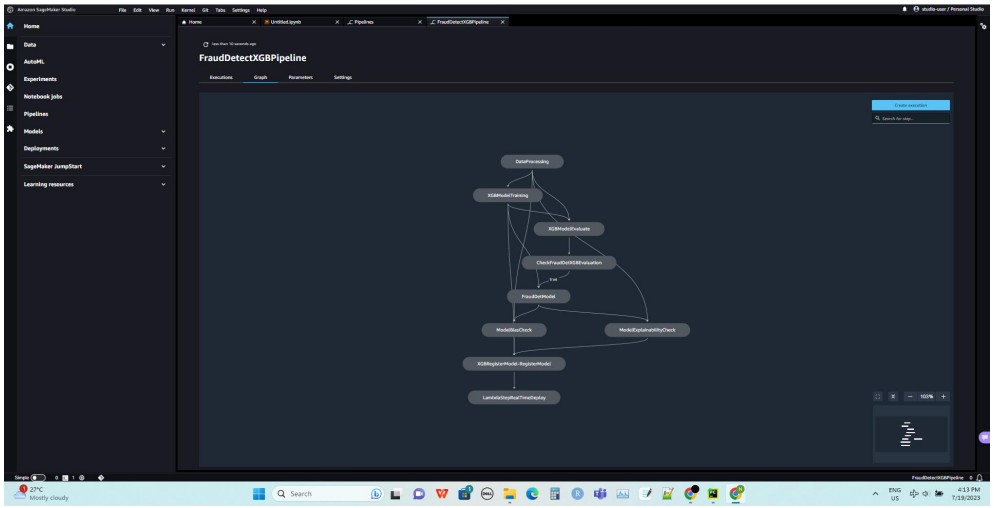
Step 5: Test the pipeline by invoking the endpoint

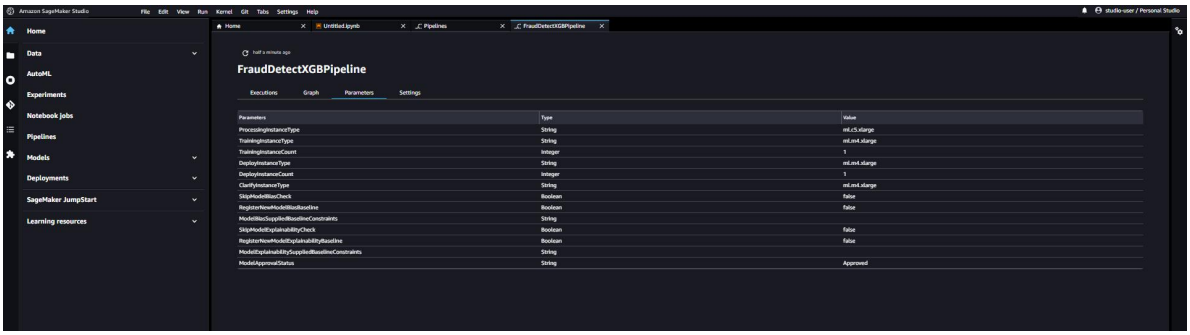
On the left hand side of the screen of SageMaker studio click-on pipelines and then select the pipeline name.



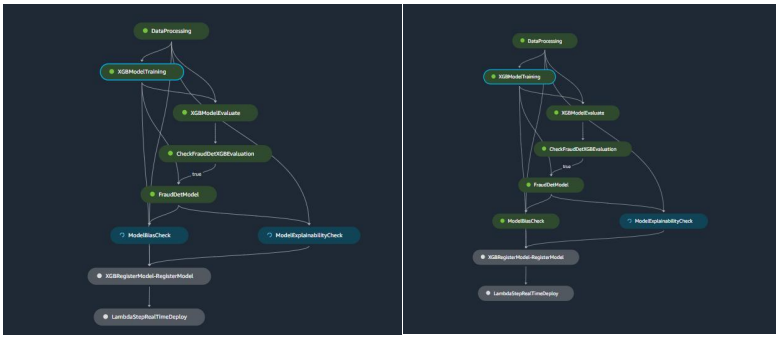
The below code executes the pipeline.

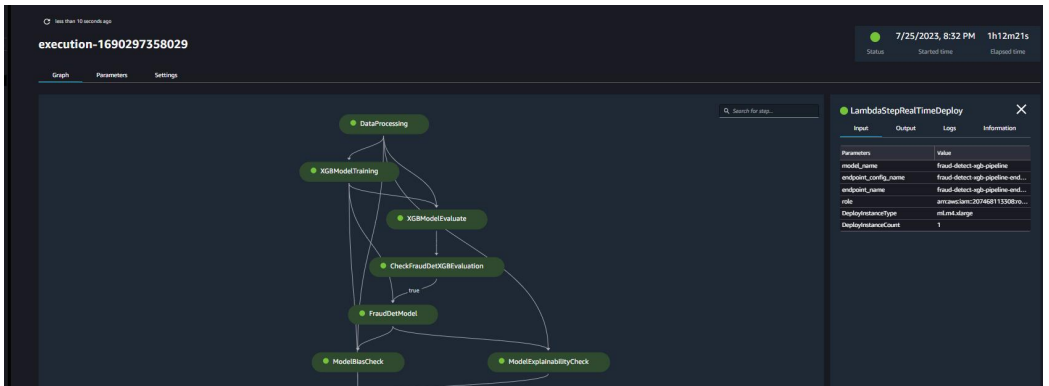




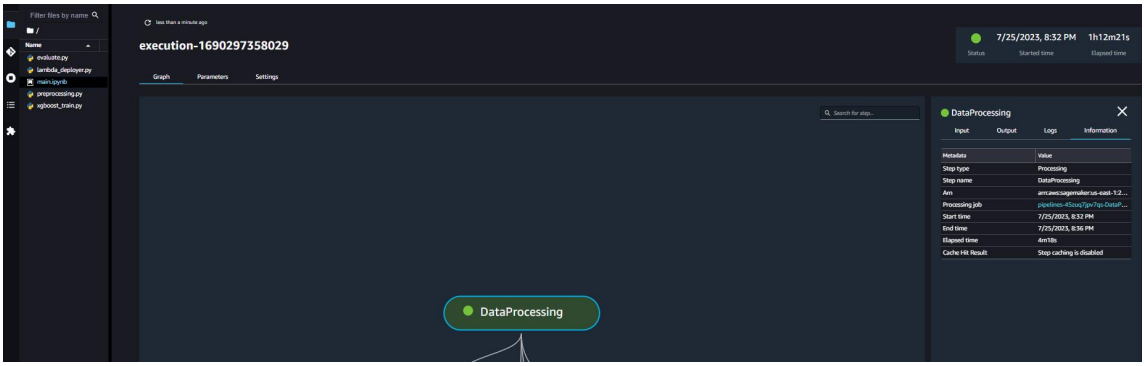


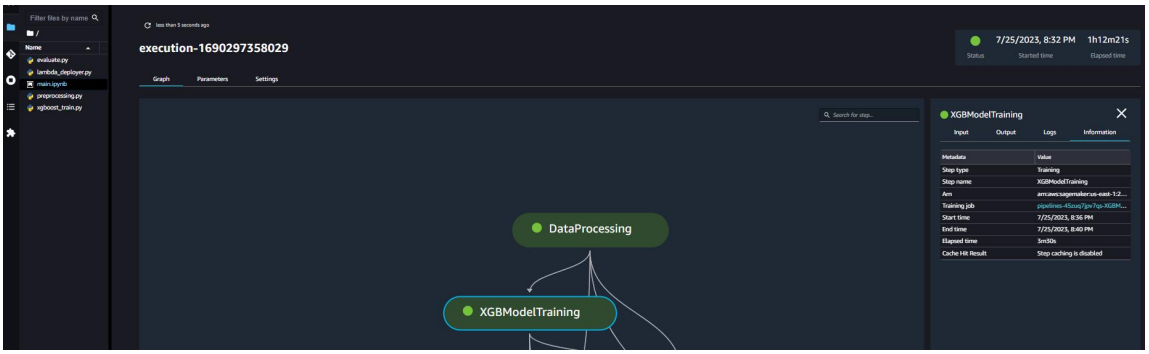
Processing flow:

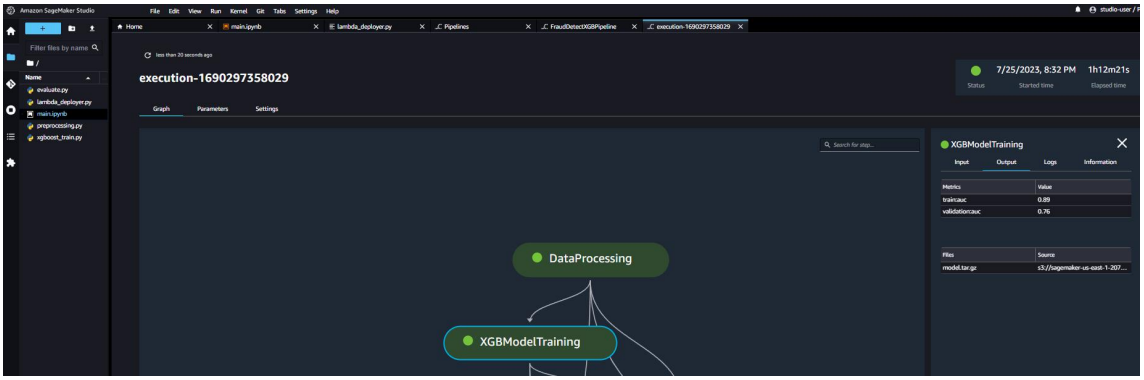


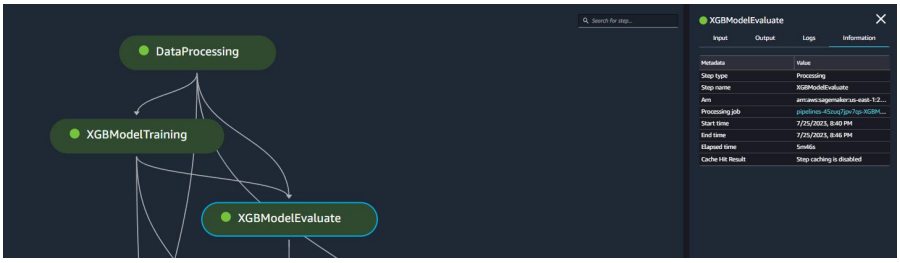


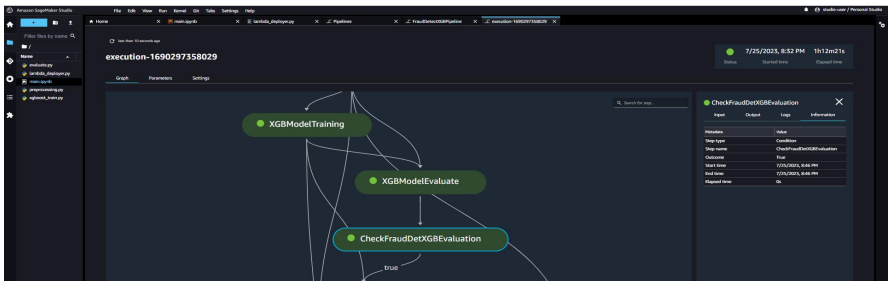
Details of each steps:

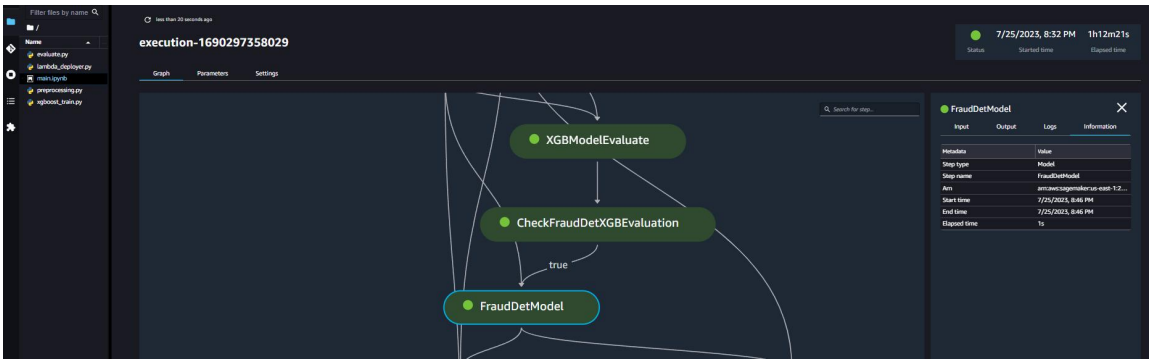


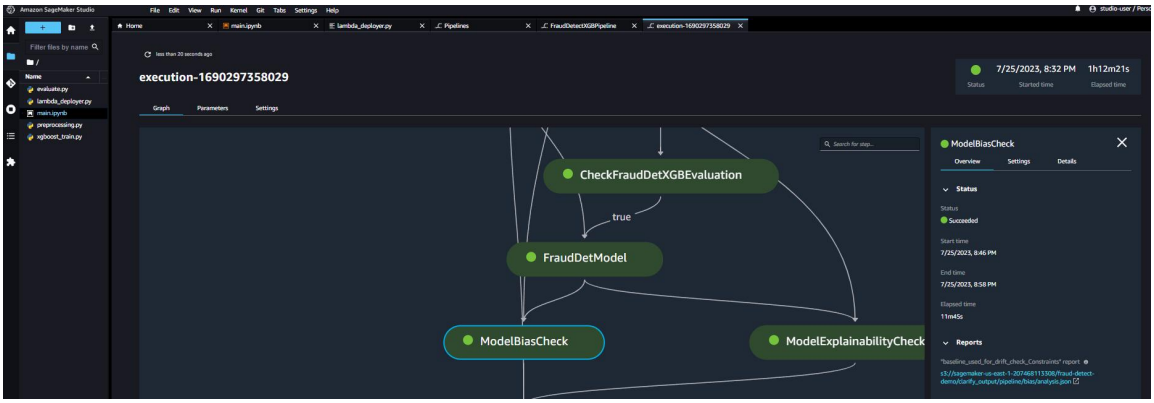


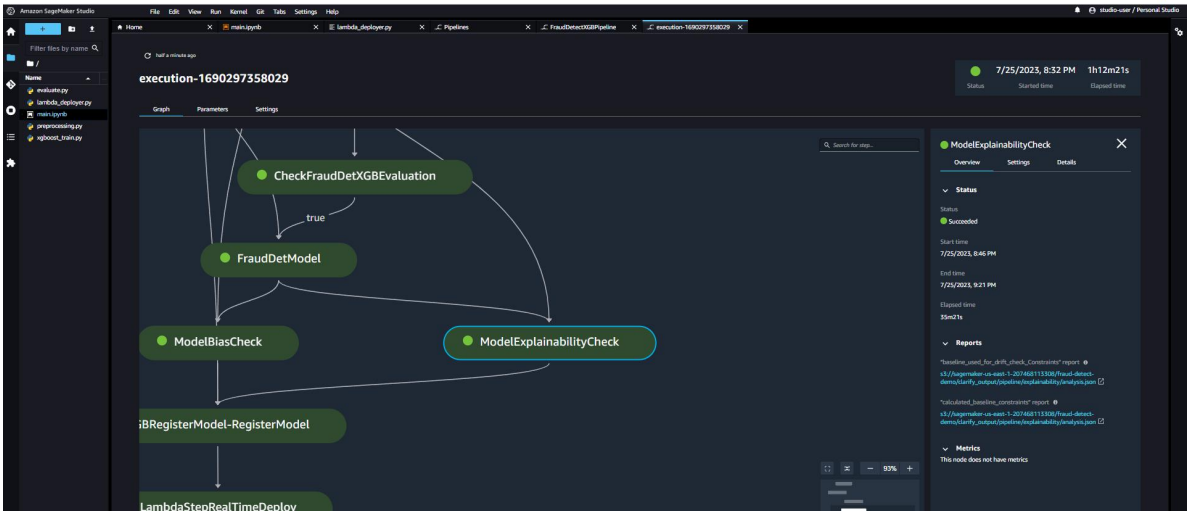


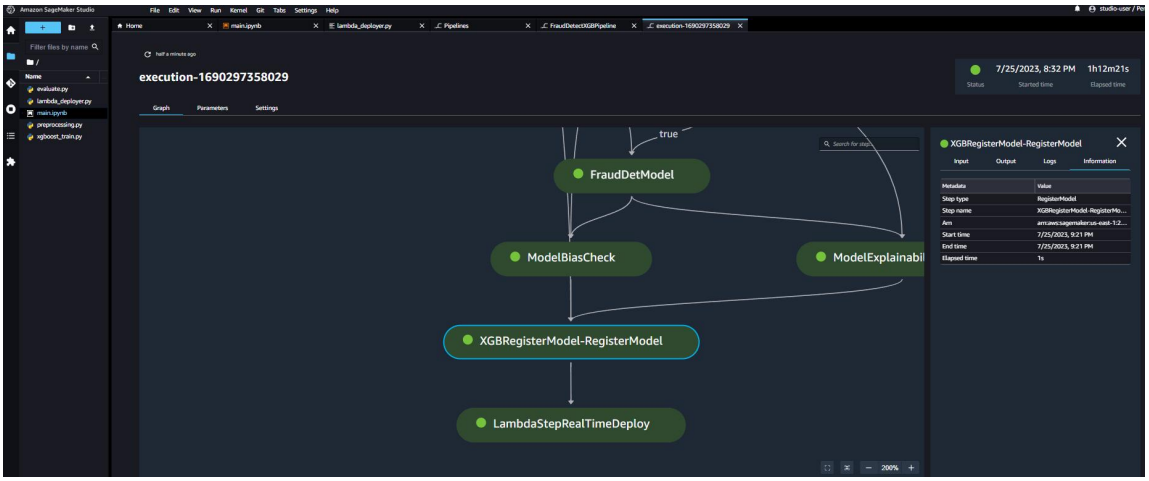


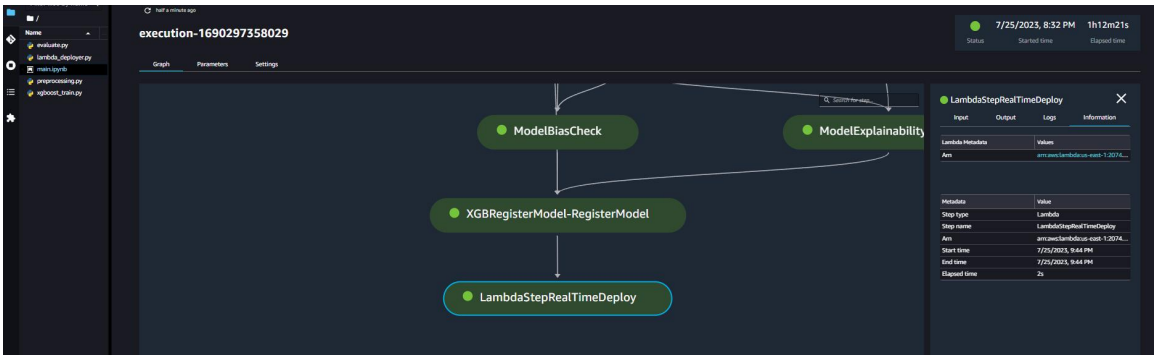



execution-1690297358029
7/25/2023, 8:32 PM
1h12m21s
XGBModelEvaluate
CheckFraudDetXGBEvaluation
true
FraudDetModel

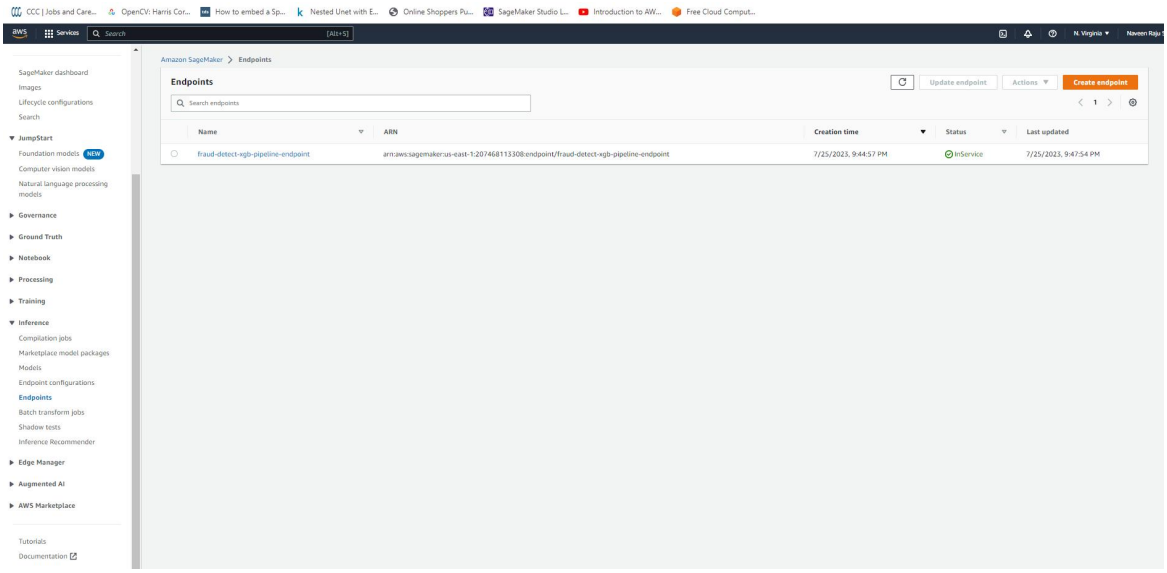

execution-1690297358029
7/25/2023, 8:32 PM
1h12m21s
CheckFraudDetXGBEvaluation
true
FraudDetModel
ModelBiasCheck
ModelExplainabilityCheck

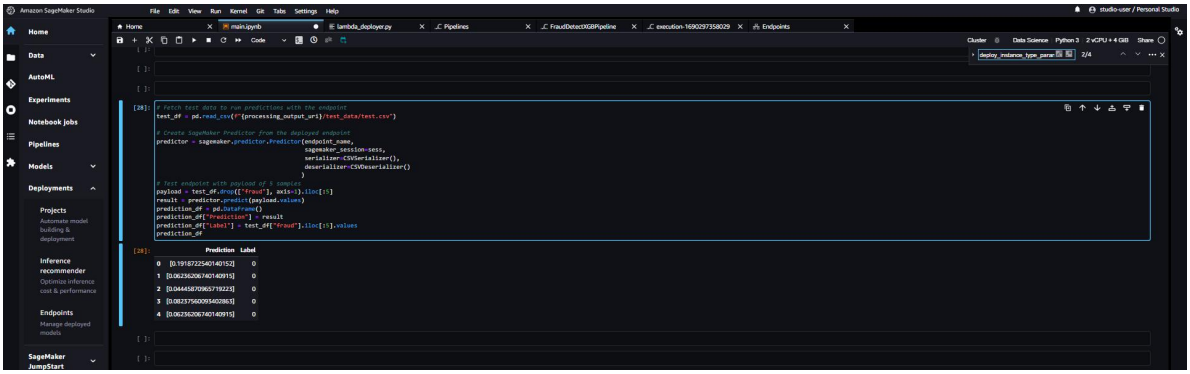

execution-1690297358029
7/25/2023, 8:32 PM
1h12m21s
CheckFraudDetXGBEvaluation
true
FraudDetModel
ModelBiasCheck
ModelExplainabilityCheck
LambdaStepRealTimeDeploy


execution-1690297358029
7/25/2023, 8:32 PM
1h12m21s
true
FraudDetModel
ModelBiasCheck
XGBRegisterModel-RegisterModel
LambdaStepRealTimeDeploy

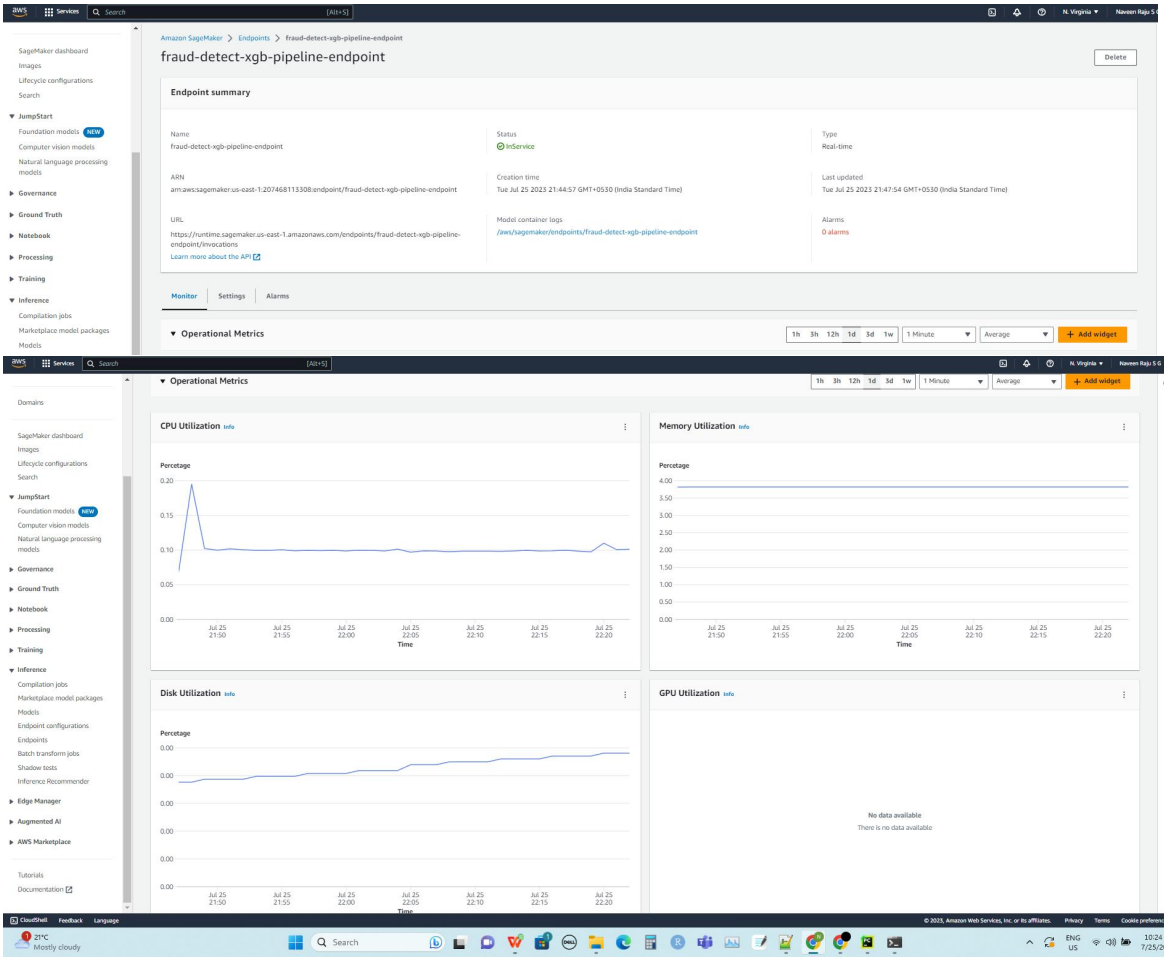


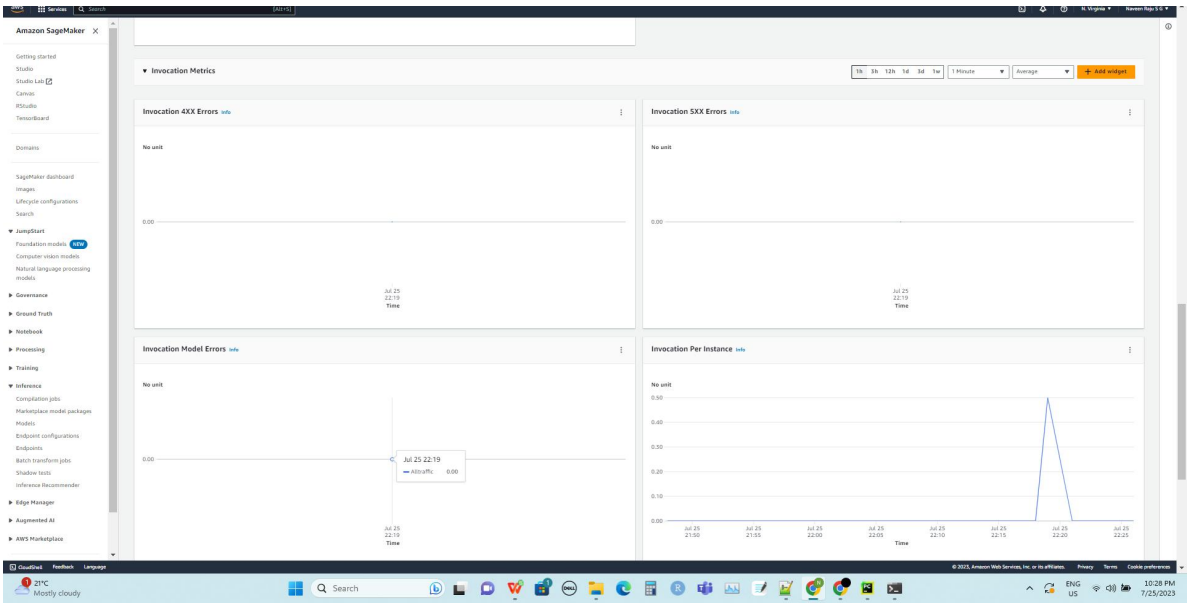
Invoke the endpoint created, the below code return model predictions of first five samples of test set.

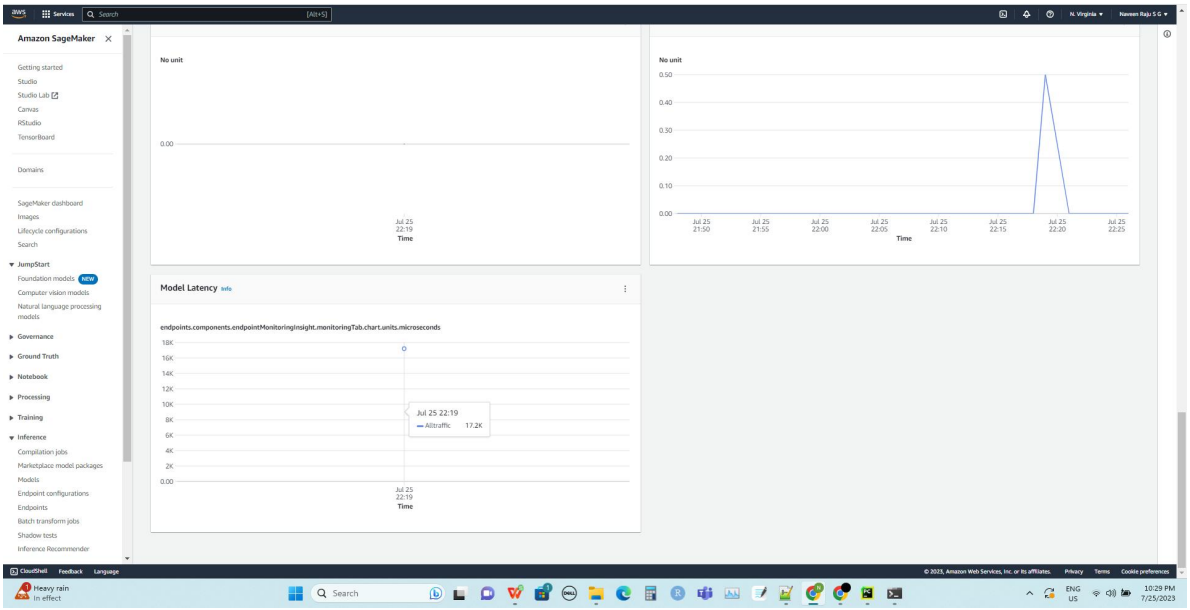


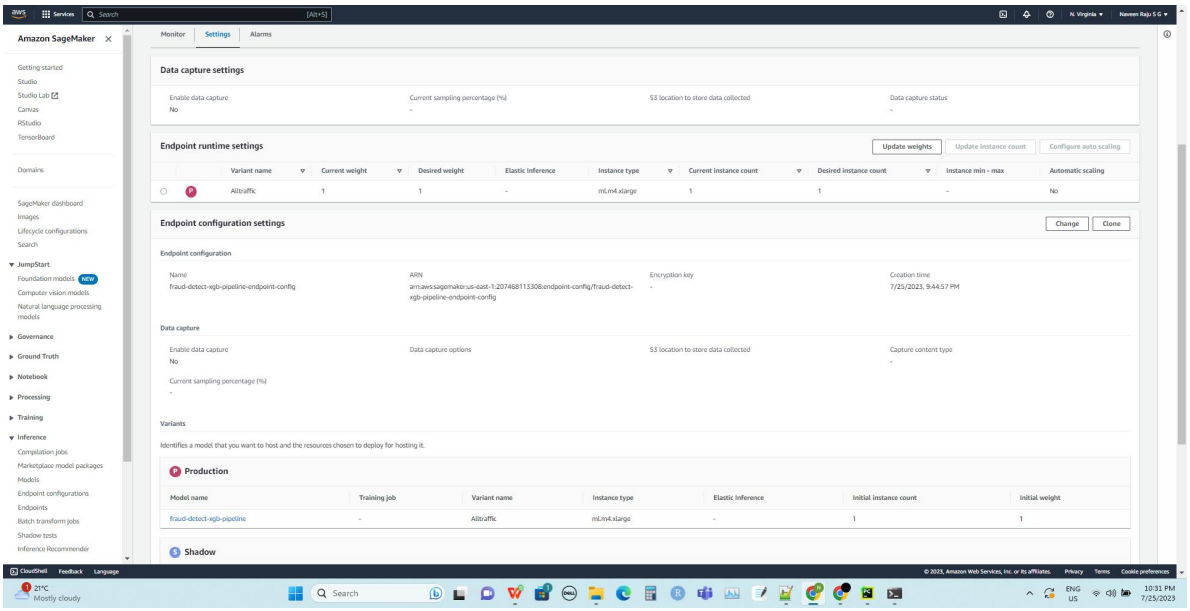


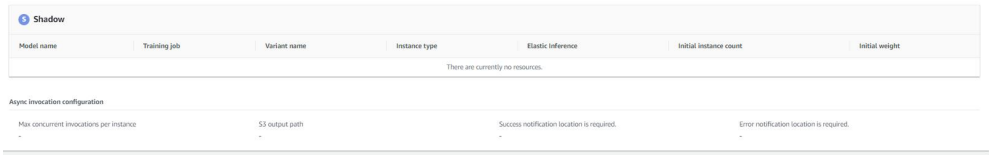
Endpoint monitoring



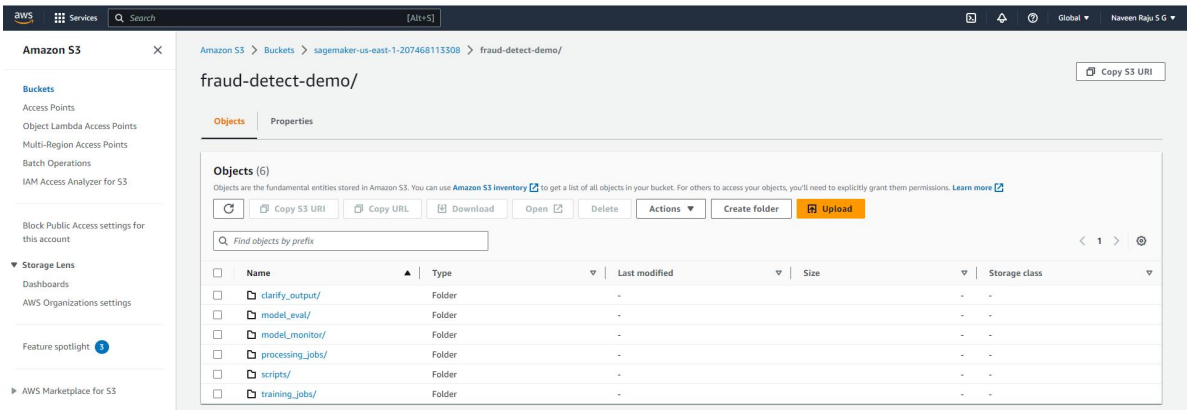



Amazon SageMaker
No unit
0.00
Jul 25 22:19
Time
No unit
0.50
0.40
0.30
0.20
0.10
0.00
Jul 25 21:50
Jul 25 21:55
Jul 25 22:00
Jul 25 22:05
Jul 25 22:10
Jul 25 22:15
Jul 25 22:20
Jul 25 22:25
Time
Model Latency
endpoints.components.endpointMonitoringInsight.monitoringTab.chart.units.microseconds
18K
16K
14K
12K
10K
8K
6K
4K
2K
0.00
Jul 25 22:19
Alltraffic 17.2K
Jul 25 22:19
Time


Amazon SageMaker
Monitor
Settings
Alarms
Data capture settings
Endpoint runtime settings
Update weights
Update instance count
Configure auto scaling
Variant name
Current weight
Desired weight
Elastic inference
Instance type
Current instance count
Desired instance count
Instance min - max
Automatic scaling
Alltraffic
1
1
-
ml.m4.xlarge
1
1
-
No
Endpoint configuration settings
Change
Clone
Endpoint configuration
Name
fraud-detect-xgb-pipeline-endpoint-config
ARN
arn:aws:sagemaker:us-east-1:207468113308:endpoint-config/fraud-detect-xgb-pipeline-endpoint-config
Encryption key
Creation time
7/25/2023, 9:44:57 PM
Data capture
Enable data capture
No
Data capture options
S3 location to store data collected
Capture content type
Current sampling percentage (%)
Variants
Identifies a model that you want to host and the resources chosen to deploy for hosting it.
Production
Model name
Training job
Variant name
Instance type
Elastic inference
Initial instance count
Initial weight
fraud-detect-xgb-pipeline
Alltraffic
ml.m4.xlarge
1
1
Shadow


Shadow
Model name
Training job
Variant name
Instance type
Elastic inference
Initial instance count
Initial weight
There are currently no resources.
Async invocation configuration
Max concurrent invocations per instance
S3 output path
Success notification location is required.
Error notification location is required.

S3 buckets structure



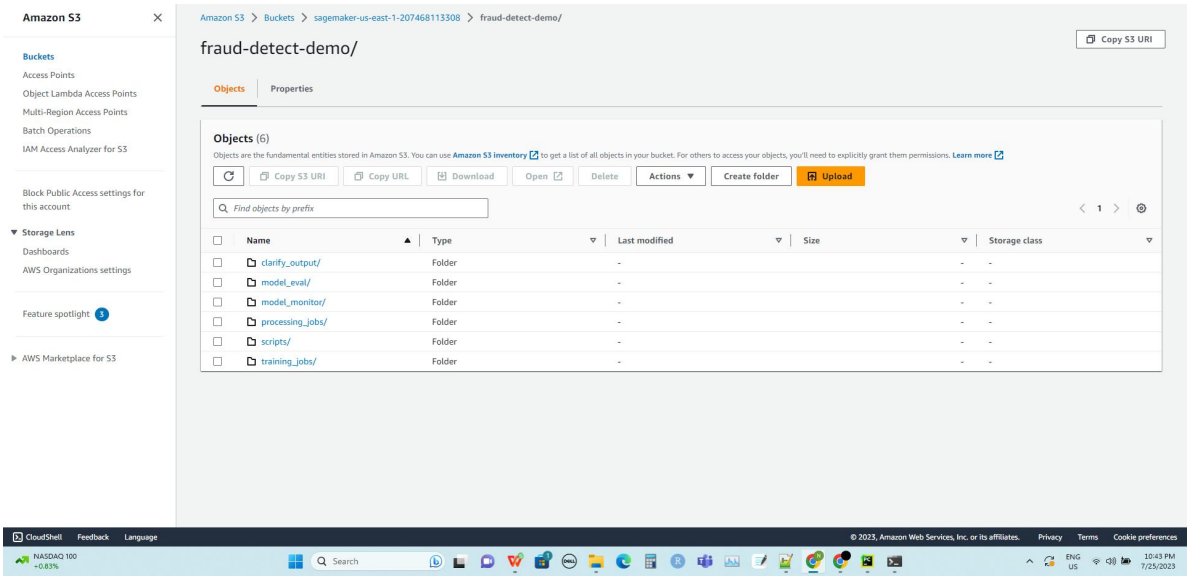
Step 6: Clean up the resources

This below code deletes the following lambda function,endpoint, endpoint config, model and the pipeline.

```
# Delete the lambda function
func.delete()

# Delete the endpoint
sm_client.delete_endpoint(EndpointName=endpoint_name)

# Delete the EndpointConfig
sm_client.delete_endpoint_config(EndpointConfigName=endpoint_config_name)

# Delete the model
sm_client.delete_model(ModelName=pipeline_model_name)

# Delete the pipeline
sm_client.delete_pipeline(PipelineName=pipeline_name)
```

```
{'PipelineArn': 'arn:aws:sagemaker:us-east-1:207468113308:pipeline/FraudDetectXGBPipeline',
 'ResponseMetadata': {'RequestId': 'ac197908-b585-4fa7-9f32-6888fb78ecab',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': 'ac197908-b585-4fa7-9f32-6888fb78ecab',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '90',
   'date': 'Tue, 25 Jul 2023 17:53:06 GMT'},
  'RetryAttempts': 0}}
```

After which we can delete all s3 buckets created



Following which we can terminate all apps running in SageMaker Studio under created domain after which we can delete the stack created through CloudFormation.