Fraud Detection using feature store with xgboost

Setup SageMaker FeatureStore

Setting up sessions

In [1]:
```
#importing all the necessary libraries
import boto3
import sagemaker
from sagemaker.session import Session

#instantiating region variable
region = boto3.Session().region_name

# instantiating the boto session
boto_session = boto3.Session(region_name=region)

# instantiating the sagemaker session
sagemaker_client = boto_session.client(service_name='sagemaker', region_name=region)

# instantiating the feature_store_runtime session
featurestore_runtime = boto_session.client(service_name='sagemaker-featurestore-runtime', region_name=region)

feature_store_session = Session(
    boto_session=boto_session,
    sagemaker_client=sagemaker_client,
    sagemaker_featurestore_runtime_client=featurestore_runtime
)
```

In [2]:
```
# bucket for feature store
default_s3_bucket_name = feature_store_session.default_bucket()
prefix = 'sagemaker-featurestore'

print(default_s3_bucket_name)
```

```
sagemaker-ap-south-1-080451317723
```

In [3]:
```
from sagemaker import get_execution_role

# role with sagemaker full access
role = get_execution_role()
print (role)
```

```
arn:aws:iam::080451317723:role/service-role/AmazonSageMaker-ExecutionRole-20211126T134068
```

Importing the data

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import io

s3_client = boto3.client('s3', region_name=region)

fraud_detection_bucket_name = default_s3_bucket_name
data_file_key = 'data/fraud-detection/credit-dataset.csv'

data_object = s3_client.get_object(Bucket=fraud_detection_bucket_name, Key=data_file_key)

transaction_data = pd.read_csv(io.BytesIO(data_object['Body'].read()))

transaction_data.head()
```

Out[4]:

	time	v1	v2	v3	v4	v5	v6	v7	v8	v9	...	v23	v24	v25	v26	
0	77471.0	1.208979	0.176456	-0.038631	0.230011	0.167064	-0.325209	0.264019	-0.124005	-0.444907	...	0.103200	-0.508589	0.170985	0.003063	-0
1	72209.0	-2.355663	-1.062768	2.881997	1.612670	-0.239989	1.462987	-1.756041	1.441738	0.836500	...	-0.222708	-0.210299	0.335265	0.064117	0
2	81336.0	1.253596	0.232190	-0.091524	0.946319	0.077062	-0.487822	0.207207	-0.163356	0.098120	...	-0.220452	-0.452818	0.776564	-0.277749	0
3	137708.0	1.129706	-1.207823	-2.350405	1.656583	0.024243	-1.347598	1.436114	-0.634697	-0.126660	...	-0.434681	0.056208	0.275236	-0.536987	-0
4	44569.0	-0.434785	0.872972	0.455299	0.869779	-0.373575	-0.220424	0.479959	0.453247	-0.829709	...	0.101452	-0.027277	-0.115668	-0.324329	-0

5 rows × 33 columns

Ingest Data into FeatureStore , creating feature groups

Creating the FeatureGroups representing the transaction and identity tables.

In [5]:
```
from time import gmtime, strftime, sleep

fd_feature_group_name = 'transactionfeaturegroup'
```

In [6]:
```
from sagemaker.feature_store.feature_group import FeatureGroup
#creating a feature group
fd_feature_group = FeatureGroup(name=fd_feature_group_name, sagemaker_session=feature_store_session)
```

In [7]:
```
import time
# creating feature group definition
current_time_sec = int(round(time.time()))

def cast_object_to_string(data_frame):
    for label in data_frame.columns:
        if data_frame.dtypes[label] == 'object':
            data_frame[label] = data_frame[label].astype("str").astype("string")

# casting object dtype to string.
#FeatureStore map the string dtype to String feature type.
cast_object_to_string(transaction_data)

# record identifier and event time feature names
record_identifier_feature_name = "record_id"
event_time_feature_name = "event_time"

# appending EventTime feature
transaction_data[event_time_feature_name] = pd.Series([current_time_sec]*len(transaction_data), dtype="float64")

# load feature definitions to the feature group
fd_feature_group.load_feature_definitions(data_frame=transaction_data);
```

Create FeatureGroups in SageMaker FeatureStore

In [8]:
```
def wait_for_feature_group_creation_complete(feature_group):
    status = feature_group.describe().get("FeatureGroupStatus")
    while status == "Creating":
        print("Waiting for Feature Group Creation")
        time.sleep(5)
        status = feature_group.describe().get("FeatureGroupStatus")
    if status != "Created":
        raise RuntimeError(f"Failed to create feature group {feature_group.name}")
    print(f"FeatureGroup {feature_group.name} successfully created.")

fd_feature_group.create(
    s3_uri=f"s3://{default_s3_bucket_name}/{prefix}",
    record_identifier_name=record_identifier_feature_name,
    event_time_feature_name=event_time_feature_name,
    role_arn=role,
    enable_online_store=True,

)

wait_for_feature_group_creation_complete(feature_group=fd_feature_group)
```

```
Waiting for Feature Group Creation
Waiting for Feature Group Creation
Waiting for Feature Group Creation
FeatureGroup transactionfeaturegroup successfully created.
```

In [9]:
```
#describing the feature group created automatically based on the data type of the columns.
fd_feature_group.describe()
```

Out[9]:
```
{'FeatureGroupArn': 'arn:aws:sagemaker:ap-south-1:080451317723:feature-group/transactionfeaturegroup',
 'FeatureGroupName': 'transactionfeaturegroup',
 'RecordIdentifierFeatureName': 'record_id',
 'EventTimeFeatureName': 'event_time',
 'FeatureDefinitions': [{'FeatureName': 'time', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v1', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v2', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v3', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v4', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v5', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v6', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v7', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v8', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v9', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v10', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v11', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v12', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v13', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v14', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v15', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v16', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v17', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v18', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v19', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v20', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v21', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v22', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v23', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v24', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v25', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v26', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v27', 'FeatureType': 'Fractional'},
  {'FeatureName': 'v28', 'FeatureType': 'Fractional'},
  {'FeatureName': 'amount', 'FeatureType': 'Fractional'},
  {'FeatureName': 'class', 'FeatureType': 'Integral'},
  {'FeatureName': 'event_time', 'FeatureType': 'Fractional'},
  {'FeatureName': 'record_id', 'FeatureType': 'Integral'}],
 'CreationTime': datetime.datetime(2021, 12, 14, 13, 17, 4, 924000, tzinfo=tzlocal()),
 'OnlineStoreConfig': {'EnableOnlineStore': True},
 'OfflineStoreConfig': {'S3StorageConfig': {'S3Uri': 's3://sagemaker-ap-south-1-080451317723/sagemaker-featurestore',
   'ResolvedOutputS3Uri': 's3://sagemaker-ap-south-1-080451317723/sagemaker-featurestore/080451317723/sagemaker/ap-south-1/offline-store/transactionfeaturegroup-1639487824/data'},
  'DisableGlueTableCreation': False,
  'DataCatalogConfig': {'TableName': 'transactionfeaturegroup-1639487824',
   'Catalog': 'AwsDataCatalog',
   'Database': 'sagemaker_featurestore'}},
 'RoleArn': 'arn:aws:iam::080451317723:role/service-role/AmazonSageMaker-ExecutionRole-20211126T134068',
 'FeatureGroupStatus': 'Created',
 'ResponseMetadata': {'RequestId': '0e7ed006-11f4-4271-90a2-ec30ba529e82',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '0e7ed006-11f4-4271-90a2-ec30ba529e82',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '2957',
   'date': 'Tue, 14 Dec 2021 13:17:20 GMT'},
  'RetryAttempts': 0}}
```

Ingesting records into FeatureGroup

In [10]:
```
#Ingesting the data into feature group created above
fd_feature_group.ingest(
    data_frame=transaction_data, max_workers=3, wait=False
)
```

Out[10]:
```
IngestionManagerPandas(feature_group_name='transactionfeaturegroup', sagemaker_fs_runtime_client_config=<botocore.config.Config object at 0x7f71a1a4f490>, max_workers=3, max_processes=1, _async_result=<multiprocess.pool.MapResult object at 0x7f719fdbc950>, _processing_pool=<pool ProcessPool(ncpus=1)>, _failed_indices=[])
```

Fetching data to check

In [11]:
```
record_identifier_value = str(100)

featurestore_runtime.get_record(FeatureGroupName=fd_feature_group_name, RecordIdentifierValueAsString=record_identifier_value)
```

Out[11]:
```
{'ResponseMetadata': {'RequestId': '7b5f872f-40c2-4d44-b90d-9a3e3dca5701',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '7b5f872f-40c2-4d44-b90d-9a3e3dca5701',
   'content-type': 'application/json',
   'content-length': '15',
   'date': 'Tue, 14 Dec 2021 13:17:20 GMT'},
  'RetryAttempts': 0}}
```

In [12]:
```
%store fd_feature_group_name
```

```
Stored 'fd_feature_group_name' (str)
```

In [ ]:
```

```