ML DEPLOYMENT AWS SAGEMAKER

SAKSHI WADHWA & ANISHA JOSHI

SO MANY OPTIONS

WHY AWS SAGE MAKER

Build, train and deploy

Scalable & less complex

One-Click deployment

End-to-end fully managed service

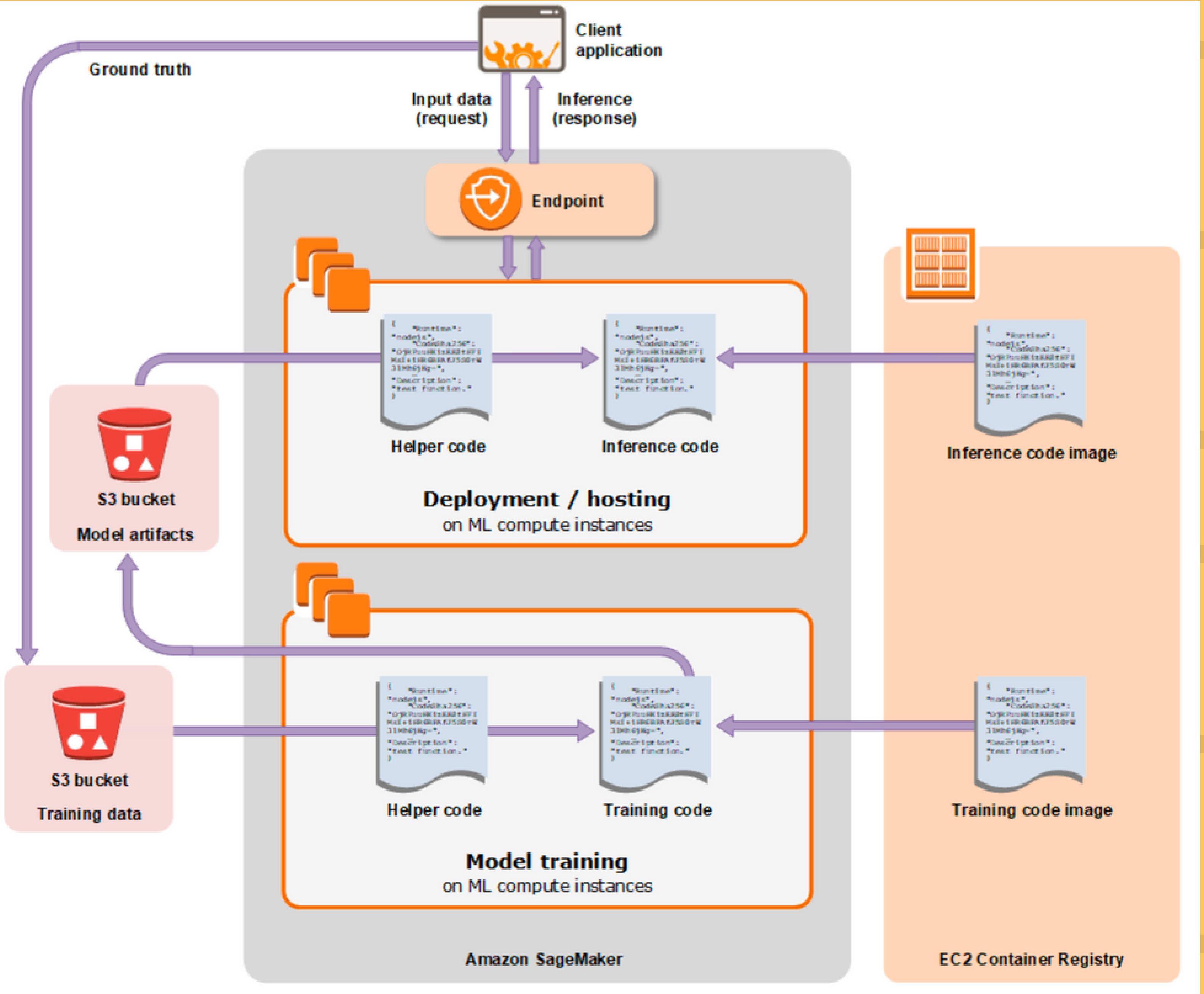
TO DEPLOY ML MODEL IN HOSTED ENVIRONMENT

PROJECT GOAL

ML MODEL DEPLOYMENT USING SAGEMAKER

What AWS SageMaker Does?

- Select & Prepare Training Data
- Choose & Optimize Your ML Model
- Setup & Manage Environment For Training
- Train & Tune Model (Trail & Error)
- Deploy Your ML Model To Production
- Scale & Manage Production Environment


Client application
Ground truth
Input data (request)
Inference (response)
Endpoint
Helper code
Inference code
Inference code image
S3 bucket
Model artifacts
Deployment / hosting
on ML compute instances
S3 bucket
Training data
Helper code
Training code
Training code image
Model training
on ML compute instances
Amazon SageMaker
EC2 Container Registry

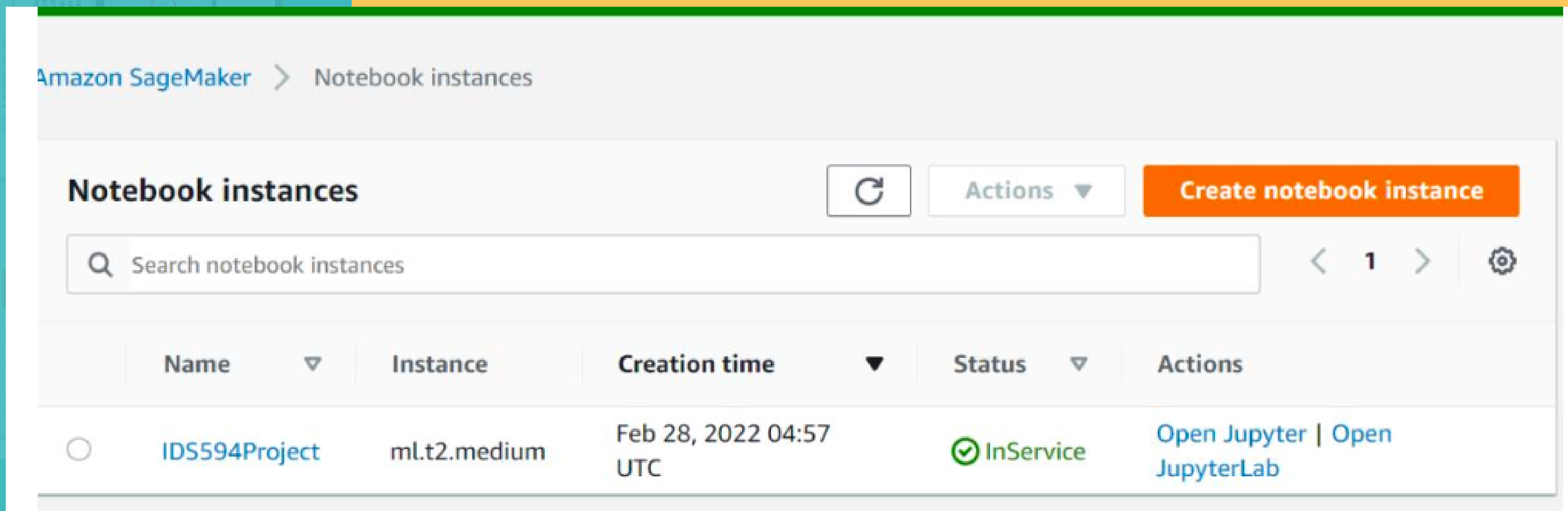
5 STEPS OF DEPLOYMENT

1. Create EC2 Instance
2. Create s3 buckets to store data and also transfer the train & test data to s3 bucket created
3. Train validation split
4. Model Training Process
5. Model Deployment and Endpoint creation

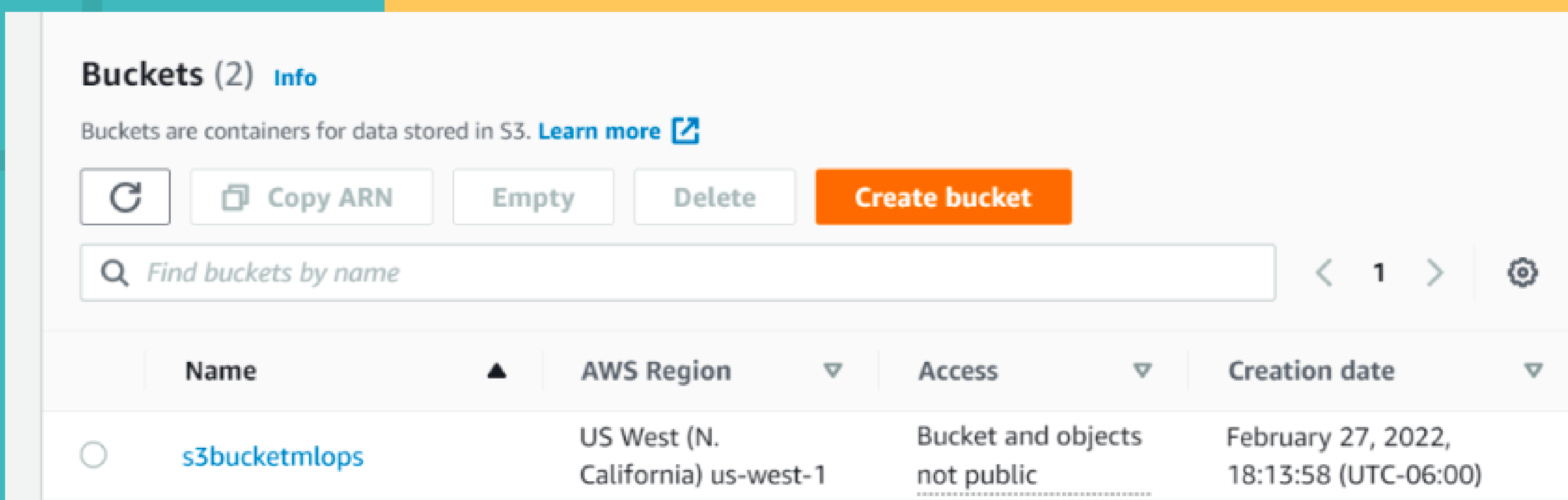
We used the InBuilt XGBoost classifier in the training phase. In SageMaker each algorithm is stored as containers in Elastic Container Registry(ECR). The ECRs are separately maintained for each region.

PROJECT WALK THROUGH

1.Creating a SageMaker Notebook Instance



2. Creating s3 bucket ~ scalable cloud storage



PROJECT WALK THROUGH

3. Upload the dataset into s3 buckets

```
### Train Test split

import numpy as np
train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
print(train_data.shape, test_data.shape)
```

```
(28831, 61) (12357, 61)
```

```
### Saving Train And Test Into Buckets
## We start with Train Data
import os
pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'],
                                                axis=1)],
                                                axis=1).to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train/train.csv')).upload_file('train.csv')
s3_input_train = sagemaker.inputs.TrainingInput(s3_data='s3://{}/{}/train'.format(bucket_name, prefix), content_type='csv')
```

```
# Test Data Into Buckets
pd.concat([test_data['y_yes'], test_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('test.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'test/test.csv')).upload_file('test.csv')
s3_input_test = sagemaker.inputs.TrainingInput(s3_data='s3://{}/{}/test'.format(bucket_name, prefix), content_type='csv')
```

This helps in retrieving the training & test data

PROJECT WALK THROUGH

4. Train the Model

```
# Here it is automatically building an URI Container
container = get_image_uri(boto3.Session().region_name,
                          'xgboost',
                          repo_version='1.0-1')
```

```
# Now we are initialize hyperparameters
hyperparameters = {
        "max_depth":"5",
        "eta":"0.2",
        "gamma":"4",
        "min_child_weight":"6",
        "subsample":"0.7",
        "objective":"binary:logistic",
        "num_round":50
        }
```

5. Construct the SageMaker Estimator

```
# Now we will construct a SageMaker estimator that calls the xgboost-container
estimator = sagemaker.estimator.Estimator(image_uri=container,
                                          hyperparameters=hyperparameters,
                                          role=sagemaker.get_execution_role(),
                                          train_instance_count=1,
                                          train_instance_type='ml.m5.2xlarge',
                                          train_volume_size=5, # 5 GB
                                          output_path=output_path,
                                          train_use_spot_instances=True,
                                          train_max_run=300,
                                          train_max_wait=600)
```

PROJECT WALK THROUGH

6. Deploy the Model & Create Endpoints

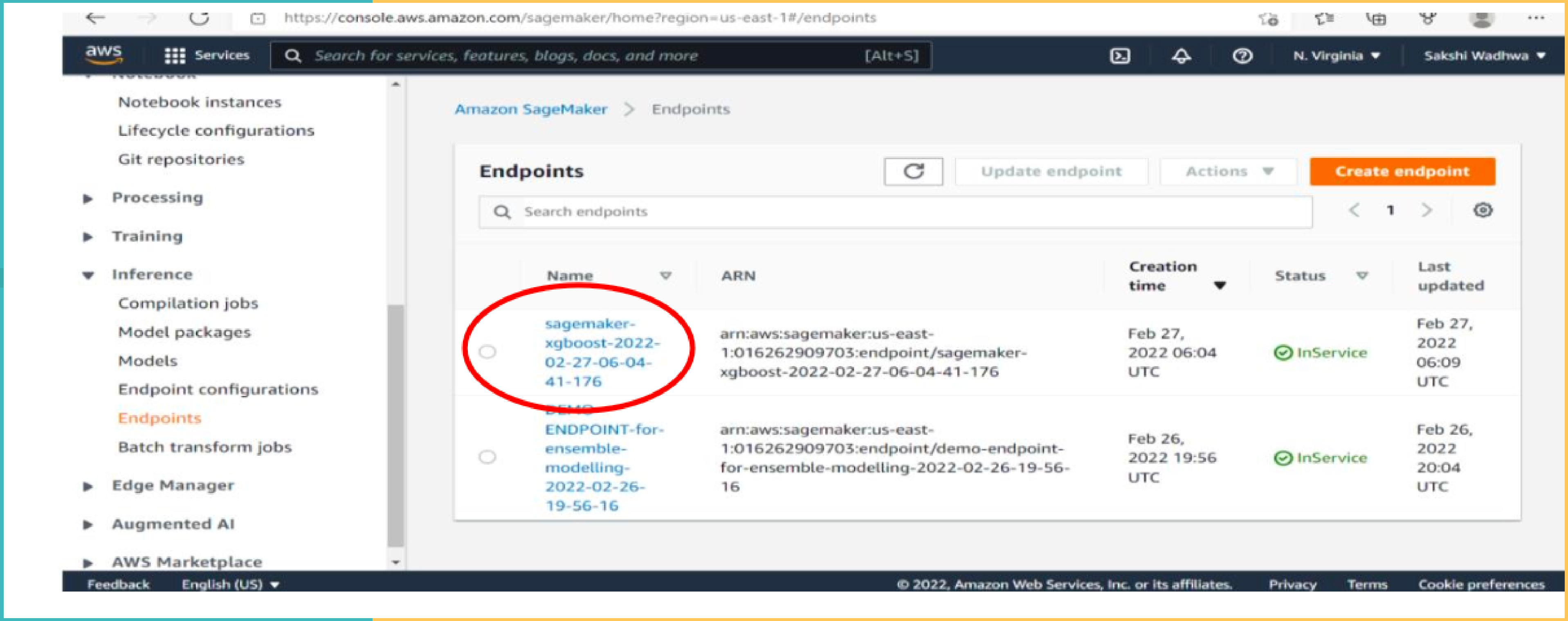
Deploy Machine Learning Model As Endpoints

```
In [38]:  xgb_predictor = estimator.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')
          -----------!

In [40]:  xgb_predictor.__dict__.keys()

Out[40]:  dict_keys(['endpoint_name', 'sagemaker_session', 'serializer', 'deserializer', '_endpoint_config_name', '_model_names', '_co
          ntext'])
```

Endpoint Creation in SageMaker



LESSONS LEARNT

- The target column has to be the first column in the data frame and there should be no headers included in the CSV when uploading to S3.
- S3 bucket should be in the same region as our training job
- The building model is inbuilt and to use the correct model we use get_image_uri()
- Dedicate one EC2 instance to model training while defining the estimator
- Delete the endpoints and bucket objects at the end

WHAT AFTER DEPLOYMENT

Overall Classification Rate: 89.7%

Predicted	No Purchase	Purchase
Observed		
No Purchase	91% (10785)	34% (151)
Purchase	9% (1124)	66% (297)

Our model showed an overall classification rate of 89.7%

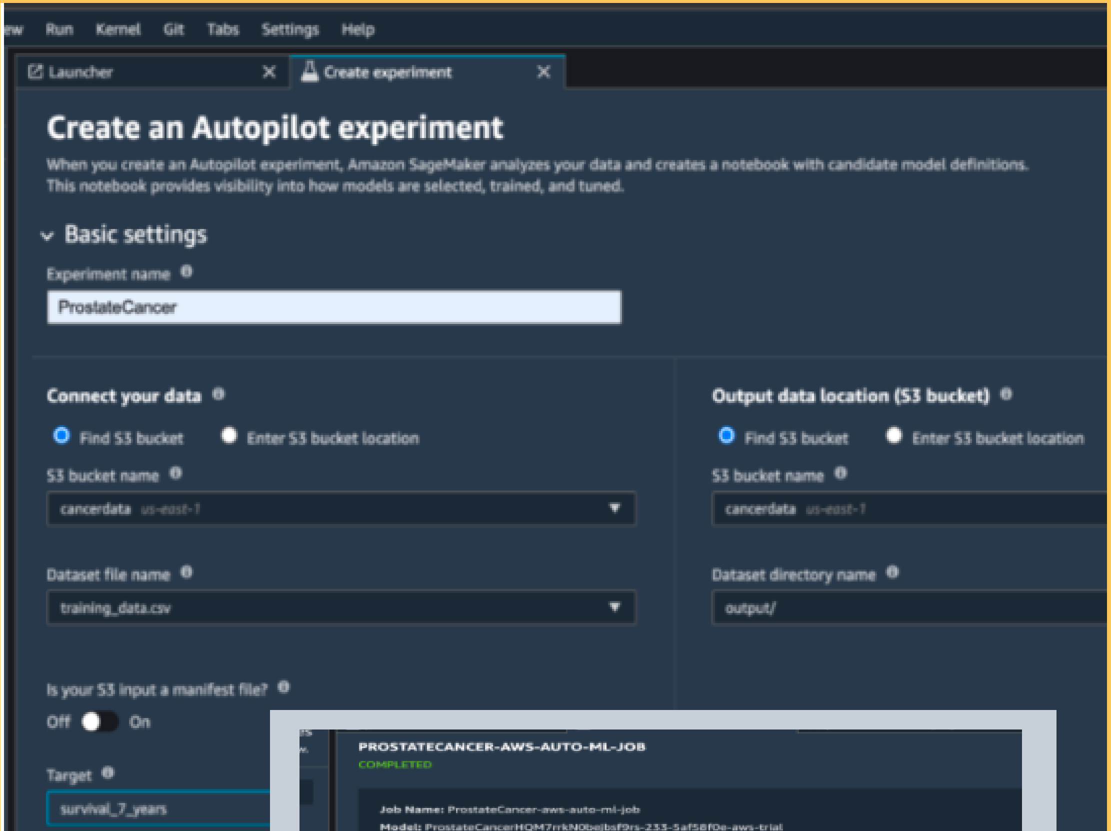
An endpoint is a restful API and Sagemaker automatically creates a restful API for every model that we deploy on Sagemaker. And when we want prediction responses, the service will go and hit the endpoints. These endpoints can be checked in the AWS management console.

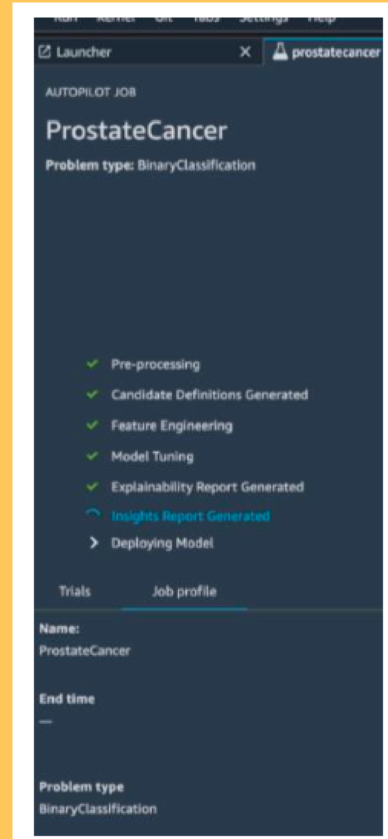
Same team members can use these endpoints, directly when need arises, within the enterprise.

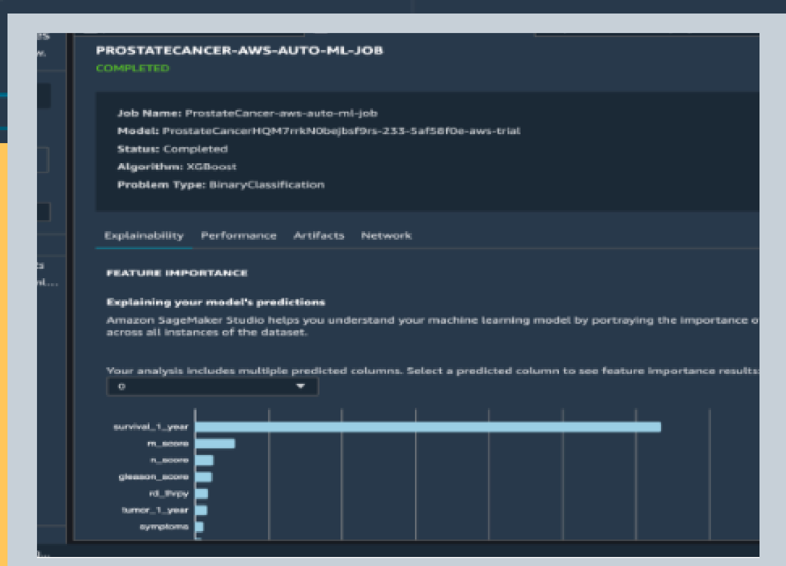
AUTOPILOT

Amazon SageMaker Autopilot eliminates the heavy lifting of building ML models

- Provided a prostate cancer dataset for classification problem.
- Got the Best Model Endpoint & Several descriptive reports







There were several stages of the experiment, and we could see them all one by one. The entire exercise took 3 hours and eventually the best model endpoint was provided to us.

The Various reports were also made available – Feature Importance, Accuracy Measures, Explainability Report, Autopilot Candidate definition notebook etc.

No matter how awesome a model we have built, it would not give any value sitting in a jupyter notebook. Hence, we have to use cloud platforms!

THANK YOU!

Although this project has ended but it is a start for both of us to explore the scalability of ML models.