Amazon SageMaker Autopilot Candidate Definition Notebook

This notebook was automatically generated by the AutoML job **automl-fraudcase-03-00-14-16**. This notebook allows you to customize the candidate definitions and execute the SageMaker Autopilot workflow.

The dataset has **29** columns and the column named **Class** is used as the target column. This is being treated as a **BinaryClassification** problem. The dataset also has **2** classes. This notebook will build a **BinaryClassification (https://en.wikipedia.org/wiki/Binary_classification)** model that **maximizes** the "**F1**" quality metric of the trained models. The "**F1**" metric applies for binary classification with a positive and negative class. It mixes between precision and recall, and is recommended in cases where there are more negative examples compared to positive examples.

As part of the AutoML job, the input dataset has been randomly split into two pieces, one for **training** and one for **validation**. This notebook helps you inspect and modify the data transformation approaches proposed by Amazon SageMaker Autopilot. You can interactively train the data transformation models and use them to transform the data. Finally, you can execute a multiple algorithm hyperparameter optimization (multi-algo HPO) job that helps you find the best model for your dataset by jointly optimizing the data transformations and machine learning algorithms.

> 💡 **Available Knobs** Look for sections like this for recommended settings that you can change.

Contents

Sagemaker Setup

Before you launch the SageMaker Autopilot jobs, we'll setup the environment for Amazon SageMaker

- Check environment & dependencies.
- Create a few helper objects/function to organize input/output data and SageMaker sessions.

Minimal Environment Requirements

- Jupyter: Tested on `JupyterLab 1.0.6` , `jupyter_core 4.5.0` and `IPython 6.4.0`
- Kernel: `conda_python3`
- Dependencies required
  - `sagemaker-python-sdk>=2.40.0`
    - Use `!pip install sagemaker==2.40.0` to download this dependency.
    - Kernel may need to be restarted after download.
- Expected Execution Role/permission
  - S3 access to the bucket that stores the notebook.

Downloading Generated Modules

Download the generated data transformation modules and an SageMaker Autopilot helper module used by this notebook. Those artifacts will be downloaded to **automl-fraudcase-03-00-14-16-artifacts** folder.

In [ ]:

```
!mkdir -p automl-fraudcase-03-00-14-16-artifacts
!aws s3 sync s3://sagemaker-us-east-1-629589781736/cc_fraud/output/automl-fra
!aws s3 sync s3://sagemaker-us-east-1-629589781736/cc_fraud/output/automl-fra

import sys
sys.path.append("automl-fraudcase-03-00-14-16-artifacts")
```

SageMaker Autopilot Job and Amazon Simple Storage Service (Amazon S3) Configuration

The following configuration has been derived from the SageMaker Autopilot job. These items configure where this notebook will look for generated candidates, and where input and output data is stored on Amazon S3.

In [ ]:
```
from sagemaker_automl import uid, AutoMLLocalRunConfig

# Where the preprocessed data from the existing AutoML job is stored
BASE_AUTOML_JOB_NAME = 'automl-fraudcase-03-00-14-16'
BASE_AUTOML_JOB_CONFIG = {
    'automl_job_name': BASE_AUTOML_JOB_NAME,
    'automl_output_s3_base_path': 's3://sagemaker-us-east-1-629589781736/cc_f
    'data_transformer_image_repo_version': '2.5-1-cpu-py3',
    'algo_image_repo_versions': {'xgboost': '1.3-1-cpu-py3', 'linear-learner'
    'algo_inference_image_repo_versions': {'xgboost': '1.3-1-cpu-py3', 'linea
}

# Path conventions of the output data storage path from the local AutoML job
LOCAL_AUTOML_JOB_NAME = 'automl-fra-notebook-run-{}'.format(uid())
LOCAL_AUTOML_JOB_CONFIG = {
    'local_automl_job_name': LOCAL_AUTOML_JOB_NAME,
    'local_automl_job_output_s3_base_path': 's3://sagemaker-us-east-1-6295897
    'data_processing_model_dir': 'data-processor-models',
    'data_processing_transformed_output_dir': 'transformed-data',
    'multi_algo_tuning_output_dir': 'multi-algo-tuning'
}

AUTOML_LOCAL_RUN_CONFIG = AutoMLLocalRunConfig(
    role='arn:aws:iam::629589781736:role/service-role/AmazonSageMaker-Executi
    base_automl_job_config=BASE_AUTOML_JOB_CONFIG,
    local_automl_job_config=LOCAL_AUTOML_JOB_CONFIG,
    security_config={'EnableInterContainerTrafficEncryption': False, 'VpcConf

AUTOML_LOCAL_RUN_CONFIG.display()
```

Candidate Pipelines

The `AutoMLLocalRunner` keeps track of selected candidates and automates many of the steps needed to execute feature engineering and tuning steps.

In [ ]:
```
from sagemaker_automl import AutoMLInteractiveRunner, AutoMLLocalCandidate

automl_interactive_runner = AutoMLInteractiveRunner(AUTOML_LOCAL_RUN_CONFIG)
```

Generated Candidates

The SageMaker Autopilot Job has analyzed the dataset and has generated **4** machine learning pipeline(s) that use **3** algorithm(s). Each pipeline contains a set of feature transformers and an algorithm.

Available Knobs

1. The resource configuration: instance type & count
2. Select candidate pipeline definitions by cells

3. The linked data transformation script can be reviewed and updated. Please refer to the [README.md (./automl-fraudcase-03-00-14-16-artifacts/generated_module/README.md)](./automl-fraudcase-03-00-14-16-artifacts/generated_module/README.md) for detailed customization instructions.

[dpp0-xgboost (automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp0.py)](automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp0.py): This data transformation strategy first transforms 'numeric' features using [RobustImputer (converts missing values to nan) (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/impute/base.py)](https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/impute/base.py). It merges all the generated features and applies [RobustStandardScaler (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/preprocessing/data.py)](https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/preprocessing/data.py). The transformed data will be used to tune a *xgboost* model. Here is the definition:

In [ ]:
```
automl_interactive_runner.select_candidate({
    "data_transformer": {
        "name": "dpp0",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
            "volume_size_in_gb":  50
        },
        "transform_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
        "transforms_label": True,
        "transformed_data_format": "text/csv",
        "sparse_encoding": False
    },
    "algorithm": {
        "name": "xgboost",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
    }
})
```

[dpp1-xgboost (automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp1.py)](automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp1.py): This data transformation strategy first transforms 'numeric' features using [RobustImputer (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/impute/base.py)](https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/impute/base.py). It merges all the generated features and applies [RobustPCA (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/decomposition/robust_pca.py)](https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/decomposition/robust_pca.py) followed by [RobustStandardScaler (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/preprocessing/data.py)](https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/preprocessing/data.py). The transformed data will be used to tune a *xgboost* model. Here is the definition:

In [ ]:

```
automl_interactive_runner.select_candidate({
    "data_transformer": {
        "name": "dpp1",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
            "volume_size_in_gb":  50
        },
        "transform_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
        "transforms_label": True,
        "transformed_data_format": "text/csv",
        "sparse_encoding": False
    },
    "algorithm": {
        "name": "xgboost",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
    }
})
```

dpp2-linear-learner (automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp2.py): This data transformation strategy first transforms 'numeric' features using combined RobustImputer and RobustMissingIndicator (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/impute/base.py) followed by QuantileExtremeValuesTransformer (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/preprocessing/base.py). It merges all the generated features and applies RobustPCA (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/decomposition/robust_pca.py) followed by RobustStandardScaler (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/preprocessing/data.py). The transformed data will be used to tune a *linear-learner* model. Here is the definition:

In [ ]:
```
automl_interactive_runner.select_candidate({
    "data_transformer": {
        "name": "dpp2",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
            "volume_size_in_gb":  50
        },
        "transform_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
        "transforms_label": True,
        "transformed_data_format": "application/x-recordio-protobuf",
        "sparse_encoding": False
    },
    "algorithm": {
        "name": "linear-learner",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
    }
})
```

dpp3-mlp (automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp3.py): This data transformation strategy transforms 'numeric' features using RobustImputer (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/impute/base.py) followed by RobustStandardScaler (https://github.com/aws/sagemaker-scikit-learn-extension/blob/master/src/sagemaker_sklearn_extension/preprocessing/data.py). The transformed data will be used to tune a *mlp* model. Here is the definition:

In [ ]:
```
automl_interactive_runner.select_candidate({
    "data_transformer": {
        "name": "dpp3",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
            "volume_size_in_gb":  50
        },
        "transform_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
        "transforms_label": True,
        "transformed_data_format": "text/csv",
        "sparse_encoding": False
    },
    "algorithm": {
        "name": "mlp",
        "training_resource_config": {
            "instance_type": "ml.m5.4xlarge",
            "instance_count": 1,
        },
        "candidate_specific_static_hyperparameters": {
            "num_categorical_features": '0',
        }
    }
})
```

Selected Candidates

You have selected the following candidates (please run the cell below and click on the feature transformer links for details):

In [ ]:
```
automl_interactive_runner.display_candidates()
```

The feature engineering pipeline consists of two SageMaker jobs:

1. Generated trainable data transformer Python modules like [dpp0.py (automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp0.py)](automl-fraudcase-03-00-14-16-artifacts/generated_module/candidate_data_processors/dpp0.py), which has been downloaded to the local file system
2. A **training** job to train the data transformers
3. A **batch transform** job to apply the trained transformation to the dataset to generate the algorithm compatible data

The transformers and its training pipeline are built using open sourced **[sagemaker-scikit-learn-container (https://github.com/aws/sagemaker-scikit-learn-container)](https://github.com/aws/sagemaker-scikit-learn-container)** and **[sagemaker-scikit-learn-extension (https://github.com/aws/sagemaker-scikit-learn-extension)](https://github.com/aws/sagemaker-scikit-learn-extension)**.

Executing the Candidate Pipelines

Each candidate pipeline consists of two steps, feature transformation and algorithm training. For efficiency first execute the feature transformation step which will generate a featurized dataset on S3 for each pipeline.

After each featurized dataset is prepared, execute a multi-algorithm tuning job that will run tuning jobs in parallel for each pipeline. This tuning job will execute training jobs to find the best set of hyper-parameters for each pipeline, as well as finding the overall best performing pipeline.

Run Data Transformation Steps

Now you are ready to start execution all data transformation steps. The cell below may take some time to finish, feel free to go grab a cup of coffee. To expedite the process you can set the number of `parallel_jobs` to be up to 10. Please check the account limits to increase the limits before increasing the number of jobs to run in parallel.

In [ ]:
```
automl_interactive_runner.fit_data_transformers(parallel_jobs=2)
```

Multi Algorithm Hyperparameter Tuning

Now that the algorithm compatible transformed datasets are ready, you can start the multi-algorithm model tuning job to find the best predictive model. The following algorithm training job configuration for each algorithm is auto-generated by the AutoML Job as part of the recommendation.

💡 **Available Knobs**

1. Hyperparameter ranges
2. Objective metrics
3. Recommended static algorithm hyperparameters.

Please refers to [Xgboost tuning (https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-tuning.html)](https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-tuning.html) and [Linear learner tuning (https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner-tuning.html)](https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner-tuning.html) for detailed explanations of the parameters.

The AutoML recommendation job has recommended the following hyperparameters, objectives and accuracy metrics for the algorithm and problem type:

In [ ]:

```
ALGORITHM_OBJECTIVE_METRICS = {
    'xgboost': 'validation:f1_binary',
    'linear-learner': 'validation:binary_f_beta',
    'mlp': 'validation:binary_f_beta',
}

STATIC_HYPERPARAMETERS = {
    'xgboost': {
        'objective': 'binary:logistic',
        'eval_metric': 'accuracy,f1_binary,auc',
        'scale_pos_weight': 577.2893401015228,
        'save_model_on_termination': 'true',
    },
    'linear-learner': {
        'predictor_type': 'binary_classifier',
        'ml_application': 'linear_learner',
        'loss_function': 'SoftmaxCrossEntropyLoss',
        'reporting_metrics': 'binary_classification_accuracy,binary_f_beta,rc
        'positive_example_weight_mult': 577.2893401015228,
        'eval_metric': 'binary_f_beta',
    },
    'mlp': {
        'problem_type': 'binary_classification',
        'positive_example_weight_mult': 577.2893401015228,
        'ml_application': 'mlp',
        'use_batchnorm': 'true',
        'activation': 'relu',
        'warmup_epochs': 10,
        'reporting_metrics': 'accuracy,binary_f_1,roc_auc',
        'eval_metric': 'binary_f_1',
    },
}
```

The following tunable hyperparameters search ranges are recommended for the Multi-Algo tuning job:

In [ ]:
```
from sagemaker.parameter import CategoricalParameter, ContinuousParameter, Ir

ALGORITHM_TUNABLE_HYPERPARAMETER_RANGES = {
    'xgboost': {
        'num_round': IntegerParameter(64, 1024, scaling_type='Logarithmic'),
        'max_depth': IntegerParameter(2, 8, scaling_type='Logarithmic'),
        'eta': ContinuousParameter(1e-3, 1.0, scaling_type='Logarithmic'),
        'gamma': ContinuousParameter(1e-6, 64.0, scaling_type='Logarithmic'),
        'min_child_weight': ContinuousParameter(1e-6, 32.0, scaling_type='Log
        'subsample': ContinuousParameter(0.5, 1.0, scaling_type='Linear'),
        'colsample_bytree': ContinuousParameter(0.3, 1.0, scaling_type='Linea
        'lambda': ContinuousParameter(1e-6, 2.0, scaling_type='Logarithmic'),
        'alpha': ContinuousParameter(1e-6, 2.0, scaling_type='Logarithmic'),
    },
    'linear-learner': {
        'mini_batch_size': IntegerParameter(128, 512, scaling_type='Linear'),
        'wd': ContinuousParameter(1e-12, 1e-2, scaling_type='Logarithmic'),
        'learning_rate': ContinuousParameter(1e-6, 1e-2, scaling_type='Logari
    },
    'mlp': {
        'mini_batch_size': IntegerParameter(128, 512, scaling_type='Linear'),
        'learning_rate': ContinuousParameter(1e-6, 1e-2, scaling_type='Logari
        'weight_decay': ContinuousParameter(1e-12, 1e-2, scaling_type='Logari
        'dropout_prob': ContinuousParameter(0.25, 0.5, scaling_type='Linear')
        'embedding_size_factor': ContinuousParameter(0.65, 0.95, scaling_type
        'network_type': CategoricalParameter(['feedforward', 'widedeep']),
        'layers': CategoricalParameter(['256', '50, 25', '100, 50', '200, 100
    },
}
```

Prepare Multi-Algorithm Tuner Input

To use the multi-algorithm HPO tuner, prepare some inputs and parameters. Prepare a dictionary whose key is the name of the trained pipeline candidates and the values are respectively:

1. Estimators for the recommended algorithm
2. Hyperparameters search ranges
3. Objective metrics

In [ ]:
```
multi_algo_tuning_parameters = automl_interactive_runner.prepare_multi_algo_p
    objective_metrics=ALGORITHM_OBJECTIVE_METRICS,
    static_hyperparameters=STATIC_HYPERPARAMETERS,
    hyperparameters_search_ranges=ALGORITHM_TUNABLE_HYPERPARAMETER_RANGES)
```

Below you prepare the inputs data to the multi-algo tuner:

In [ ]:
```
multi_algo_tuning_inputs = automl_interactive_runner.prepare_multi_algo_input
```

Create Multi-Algorithm Tuner

With the recommended Hyperparameter ranges and the transformed dataset, create a multi-algorithm model tuning job that coordinates hyper parameter optimizations across the different possible algorithms and feature processing strategies.

Available Knobs

1. Tuner strategy: [Bayesian (https://en.wikipedia.org/wiki/Hyperparameter_optimization#Bayesian_optimization)], [Random Search (https://en.wikipedia.org/wiki/Hyperparameter_optimization#Random_search)]
2. Objective type: `Minimize`, `Maximize`, see [optimization (https://en.wikipedia.org/wiki/Mathematical_optimization)]
3. Max Job size: the max number of training jobs HPO would be launching to run experiments. Note the default value is **250** which is the default of the managed flow.
4. Parallelism. Number of jobs that will be executed in parallel. Higher value will expedite the tuning process. Please check the account limits to increase the limits before increasing the number of jobs to run in parallel
5. Please use a different tuning job name if you re-run this cell after applied customizations.

In [ ]:
```
from sagemaker.tuner import HyperparameterTuner

base_tuning_job_name = "{}-tuning".format(AUTOML_LOCAL_RUN_CONFIG.local_autom

tuner = HyperparameterTuner.create(
    base_tuning_job_name=base_tuning_job_name,
    strategy='Bayesian',
    objective_type='Maximize',
    max_parallel_jobs=2,
    max_jobs=250,
    **multi_algo_tuning_parameters,
)
```

Run Multi-Algorithm Tuning

Now you are ready to start running the **Multi-Algo Tuning** job. After the job is finished, store the tuning job name which you use to select models in the next section. The tuning process will take some time, please track the progress in the Amazon SageMaker Hyperparameter tuning jobs console.

In [ ]:
```
from IPython.display import display, Markdown

# Run tuning
tuner.fit(inputs=multi_algo_tuning_inputs, include_cls_metadata=None)
tuning_job_name = tuner.latest_tuning_job.name

display(
    Markdown(f"Tuning Job {tuning_job_name} started, please track the progres

# Wait for tuning job to finish
tuner.wait()
```

Model Selection and Deployment

This section guides you through the model selection process. Afterward, you construct an inference pipeline on Amazon SageMaker to host the best candidate.

Because you executed the feature transformation and algorithm training in two separate steps, you now need to manually link each trained model with the feature transformer that it is associated with. When running a regular Amazon SageMaker Autopilot job, this will automatically be done for you.

Tuning Job Result Overview

The performance of each candidate pipeline can be viewed as a Pandas dataframe. For more interactive usage please refers to [model tuning monitor (https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-monitor.html)](https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-monitor.html).

In [ ]:
```
from pprint import pprint
from sagemaker.analytics import HyperparameterTuningJobAnalytics

SAGEMAKER_SESSION = AUTOML_LOCAL_RUN_CONFIG.sagemaker_session
SAGEMAKER_ROLE = AUTOML_LOCAL_RUN_CONFIG.role

tuner_analytics = HyperparameterTuningJobAnalytics(
    tuner.latest_tuning_job.name, sagemaker_session=SAGEMAKER_SESSION)

df_tuning_job_analytics = tuner_analytics.dataframe()

# Sort the tuning job analytics by the final metrics value
df_tuning_job_analytics.sort_values(
    by=['FinalObjectiveValue'],
    inplace=True,
    ascending=False if tuner.objective_type == "Maximize" else True)

# Show detailed analytics for the top 20 models
df_tuning_job_analytics.head(20)
```

The best training job can be selected as below:

💡 **Tips:** You could select alternative job by using the value from `TrainingJobName` column above and assign to `best_training_job` below

In [ ]:
```
attached_tuner = HyperparameterTuner.attach(tuner.latest_tuning_job.name, sag
best_training_job = attached_tuner.best_training_job()

print("Best Multi Algorithm HPO training job name is {}".format(best_training
```

Linking Best Training Job with Feature Pipelines

Finally, deploy the best training job to Amazon SageMaker along with its companion feature engineering models. At the end of the section, you get an endpoint that's ready to serve online inference or start batch transform jobs!

Deploy a [PipelineModel (https://sagemaker.readthedocs.io/en/stable/pipeline.html)](https://sagemaker.readthedocs.io/en/stable/pipeline.html) that has multiple containers of the following:

1. Data Transformation Container: a container built from the model we selected and trained during the data transformer sections
2. Algorithm Container: a container built from the trained model we selected above from the best HPO training job.
3. Inverse Label Transformer Container: a container that converts numerical intermediate prediction value back to non-numerical label value.

Get both best data transformation model and algorithm model from best training job and create an pipeline model:

In [ ]:

```
from sagemaker.estimator import Estimator
from sagemaker import PipelineModel
from sagemaker_automl import select_inference_output

# Get a data transformation model from chosen candidate
best_candidate = automl_interactive_runner.choose_candidate(df_tuning_job_ana
best_data_transformer_model = best_candidate.get_data_transformer_model(role=

# Our first data transformation container will always return recordio-protobu
best_data_transformer_model.env["SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT"] = 'ap
# Add environment variable for sparse encoding
if best_candidate.data_transformer_step.sparse_encoding:
    best_data_transformer_model.env["AUTOML_SPARSE_ENCODE_RECORDIO_PROTOBUF"]

# Get a algo model from chosen training job of the candidate
algo_estimator = Estimator.attach(best_training_job)
best_algo_model = algo_estimator.create_model(**best_candidate.algo_step.get_

# Final pipeline model is composed of data transformation models and algo mod
# inverse label transform model if we need to transform the intermediates bac
model_containers = [best_data_transformer_model, best_algo_model]
if best_candidate.transforms_label:
    model_containers.append(best_candidate.get_data_transformer_model(
        transform_mode="inverse-label-transform",
        role=SAGEMAKER_ROLE,
        sagemaker_session=SAGEMAKER_SESSION))

# This model can emit response ['predicted_label', 'probability', 'labels', '
# of the response content, pass the keys to `output_key` keyword argument in

model_containers = select_inference_output("BinaryClassification", model_cont


pipeline_model = PipelineModel(
    name="AutoML-{}".format(AUTOML_LOCAL_RUN_CONFIG.local_automl_job_name),
    role=SAGEMAKER_ROLE,
    models=model_containers,
    vpc_config=AUTOML_LOCAL_RUN_CONFIG.vpc_config)
```

Deploying Best Pipeline

💡 **Available Knobs**

1. You can customize the initial instance count and instance type used to deploy this model.
2. Endpoint name can be changed to avoid conflict with existing endpoints.

Finally, deploy the model to SageMaker to make it functional.

In [ ]:
```
pipeline_model.deploy(initial_instance_count=1,
                      instance_type='ml.m5.2xlarge',
                      endpoint_name=pipeline_model.name,
                      wait=True)
```

Congratulations! Now you could visit the sagemaker [endpoint console page (https://us-east-1.console.aws.amazon.com/sagemaker/home?region=us-east-1#/endpoints)](https://us-east-1.console.aws.amazon.com/sagemaker/home?region=us-east-1#/endpoints) to find the deployed endpoint (it'll take a few minutes to be in service).

To rerun this notebook, delete or change the name of your endpoint!
If you rerun this notebook, you'll run into an error on the last step because the endpoint already exists. You can either delete the endpoint from the endpoint console page or you can change the `endpoint_name` in the previous code block.