UDACITY

‹ Return to Classroom

DISCUSS ON STUDENT HUB

Optimizing an ML Pipeline in Azure

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Bright Udacian,

Congratulations on this submission! You have successfully met all the rubric specifications 👏🏼👏🏼
You've demonstrated how well you have learned in Optimizing an ML Pipeline in Azure. Completing this project allows you to learn and experience Machine Learning's wide range of applications and its incredible ability to adapt and provide solutions to complex problems efficiently, effectively and quickly.

Here are some documentation (in case you haven't seen) that explains the general concept of this project:

- AutoML Config
- Azure Machine Learning SDK
- Hyperparameter Tuning
- In case you'd be needing, this How to write a good README is a helpful resource about writing a high-quality README

Thank you for your hard work and I hope this feedback helps.
Best of luck!

Documentation

The README contains an explanation of:

- **The pipeline architecture, including data, hyperparameter tuning, and classification algorithm.**
- **The benefits of the chosen parameter sampler.**
- **The benefits of the chosen early stopping policy.**

- Good job in discussing the pipeline architecture. You were able to identify correctly what this project seeks to predict. 💯
- You've also provided the benefits of chosen parameter sampler `RandomParameterSampler` and early stopping policy `BanditPolicy` and discussed why you have chosen these amongst the parameter sampler and early stopping policies in Azure. ✅

Suggestion

You can consider adding a block diagram which represents the pipeline architecture to provide an overall view of this project.

The README contains:

- **One or more sentences describing the model and parameters generated by AutoML.**
- **Two or more sentences comparing the two models and their performance.**

- You have successfully identified the best performing model `Voting Ensemble` and the parameters generated by AutoML. Indeed, AutoMl uses different set of ML algorithm to perform the training. Some of these are `Xgboost`, `ExtremeRandomTrees`, `StandardScalerWrapper`, `RandomForest` etc.
- You've successfully compared the two models' performances based on accuracy ✅

Suggestion

Model interpretability allows you to understand why your models made predictions, and the underlying feature importance values. The SDK includes various packages for enabling model interpretability features, both at training and inference time, for local and deployed models.
In this documentation you can learn to enable interpretability features specifically within AutoML experiments.

The README contains two or more sentences explaining potential improvements for a future experiment and why these improvements might improve the model.

Very well! I liked that you mentioned about the imbalanced data problem. Hence, trying another metric such as AUC_weighted, is a potential improvement for this project.

Training Pipeline and AutoML

All specifiable parameters of the training script are specified in the hyperdrive config.

```
hyperdrive_config = HyperDriveConfig(
    hyperparameter_sampling =ps,
    primary_metric_name='Accuracy',
    primary_metric_goal = PrimaryMetricGoal.MAXIMIZE,
    max_total_runs = 20,
    policy = policy,
    estimator = est

)
```

You used `RandomParameterSampling` as parameter sampler and `BanditPolicy` as early stopping policy.
Azure Machine Learning supports the following methods:

- Random sampling
- Grid sampling
- Bayesian sampling

Good job in choosing `RandomParameterSampling` as it supports the use of early termination policy unlike `BayesianSampling` which do not, and supports both continuous and discrete hyperparamenters unlike `GridSampling` that supports discrete only.

A hyperdrive config is used and includes:

- **A parameter sampler**
- **A policy for early stopping**

```
ps = RandomParameterSampling({
    '--C': choice(0.1, 0.2, 0.3, 0.5, 1, 2, 3, 5, 10),
    '--max_iter' : choice(20, 30, 40, 50, 75, 90, 100)
})

# Specify a Policy
### YOUR CODE HERE ###
policy = BanditPolicy(slack_factor=0.15, evaluation_interval = 5)
```

The HyperDrive configuration includes information about hyperparameter space sampling, termination policy, primary metric, estimator, and the compute target to execute the experiment runs on. ✅

The hyperdrive run is passed to the *RunDetails* widget.

```
hyperdrive_submission = exp.submit(config = hyperdrive_config, show_output = True)
RunDetails(hyperdrive_submission).show()
hyperdrive_submission.wait_for_completion(show_output = True)
```

The hyperdrive run is passed to the *RunDetails* widget to show parent run properties, logs, child runs, primary metric chart, and parallel coordinate chart of hyperparameters. ✅

`.get_best_run_by_primary_metric()` is used on the hyperdrive run to retrieve the best run.

```
best_model = hyperdrive_submission.get_best_run_by_primary_metric()
print('Metrics: {0}',format(best_model.get_metrics()))
print(best_model.get_details()['runDefinition']['arguments'])
```

`.get_best_run_by_primary_metric()` is used on the hyperdrive run to retrieve the best run. ✅

The solution notebook includes an AutoML config, which contains the following parameters:

- `task`
- `primary_metric`
- `experiment_timeout_minutes`
- `training_data`
- `label_column_name`
- `n_cross_validations`

```
automl_config = AutoMLConfig(
    experiment_timeout_minutes=30,
    task='classification',
    primary_metric='accuracy',
    training_data=train_data,
    label_column_name='y',
    n_cross_validations=5)
```

AutoML configuration object contains and persists the parameters for configuring the experiment run includes `

- task ✅
- primary_metric ✅
- experiment_timeout_minutes ✅
- training_data ✅
- label_column_name ✅
- n_cross_validations ✅

Infrastructure

A compute cluster is created using the Azure SDK and the `ComputeTarget` and `AmlCompute` objects.

```
cluster_config = AmlCompute.provisioning_configuration(vm_size='Standard_D2_V
2', max_nodes=4, min_nodes=1)
compute_target = ComputeTarget.create(ws, 'ml-cluster', cluster_config)
```

The solution notebook includes code to create a compute cluster

Suggestion

- You can include a try/except block that allows reuse of existing clusters using using the Azure SDK and the `ComputeTarget` and `AmlCompute objects`

A `TabularDatasetFactory` is used to create a dataset from the provided link.

The `delete` method of the `AmlCompute` object is used to remove the cluster following training.

OR

An image of the compute cluster being selected for deletion is included in the README.

`compute_target.delete()`

The `delete` method of the `AmlCompute` object is used to remove the cluster following training ✅

Learning Note

Why the cluster should be deleted after the training?

The purpose of this is for you to apply the practice of cleaning your workspace after experimentation to conserve resources.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START