AutoAI | Part of IBM Watson® Studio Experiment Notebook

Experiment Notebook - AutoAI Notebook v1.15.4

This notebook contains the steps and code to demonstrate support of AutoAI experiments in Watson Machine Learning service. It introduces Python API commands for data retrieval, training experiments, persisting pipelines, testing pipelines, refining pipelines, and scoring the resulting model.

Note: Notebook code generated using AutoAI will execute successfully. If code is modified or reordered, there is no guarantee it will successfully execute. For details, see: [Saving an Auto AI experiment as a notebook (https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-notebook.html)](https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-notebook.html)

Some familiarity with Python is helpful. This notebook uses Python 3.8 and `ibm_watson_machine_learning` package.

Notebook goals

The learning goals of this notebook are:

- Defining an AutoAI experiment
- Training AutoAI models
- Comparing trained models
- Deploying the model as a web service
- Scoring the model to generate predictions.

Contents

This notebook contains the following parts:

Setup
  Package installation
  Watson Machine Learning connection
Experiment configuration
  Experiment metadata
Working with completed AutoAI experiment
  Get fitted AutoAI optimizer
  Pipelines comparison
  Get pipeline as scikit-learn pipeline model
  Inspect pipeline
    Visualize pipeline model
    Preview pipeline model as python code
Deploy and Score
  Working with spaces
Running AutoAI experiment with Python API
Clean up
Next steps
Copyrights

Setup

Package installation

Before you use the sample code in this notebook, install the following packages:

- ibm-watson-machine-learning,
- autoai-libs,
- lale,
- scikit-learn,
- xgboost,
- lightgbm,
- snapml.

In [ ]:
```
!pip install ibm-watson-machine-learning | tail -n 1
!pip install -U autoai-libs==1.12.13 | tail -n 1
!pip install -U 'lale>=0.5.3,<0.6' | tail -n 1
!pip install -U scikit-learn==0.23.2 | tail -n 1
!pip install -U xgboost==1.3.3 | tail -n 1
!pip install -U lightgbm==3.1.1 | tail -n 1
!pip install -U snapml==1.7.4 | tail -n 1
```

Experiment configuration

Experiment metadata

This cell defines the metadata for the experiment, including: training_data_reference, training_result_reference, experiment_metadata.

In [ ]:
```
from ibm_watson_machine_learning.helpers import DataConnection
from ibm_watson_machine_learning.helpers import S3Connection, S3Lo
cation

training_data_reference = [
    DataConnection(
    connection=S3Connection(
        api_key='CFTaknyIPqotjaHSVO2OiaNGeNapmkI_D07uyYYCC4nW',
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3.ap.cloud-object-storage.appdomai
n.cloud'
    ),
        location=S3Location(
            bucket='aiassistedfarming-donotdelete-pr-vdhfw2plbkonq
u',
            path='crop_production.csv'
        )
    ),
]
training_result_reference = DataConnection(
    connection=S3Connection(
        api_key='CFTaknyIPqotjaHSVO2OiaNGeNapmkI_D07uyYYCC4nW',
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3.ap.cloud-object-storage.appdomai
n.cloud'
    ),
    location=S3Location(
        bucket='aiassistedfarming-donotdelete-pr-vdhfw2plbkonqu',
        path='auto_ml/712192f1-9c91-48ad-af46-ce1da48e1f9b/wml_dat
a/e39c26fa-741c-4161-bede-f8de5be66350/data/automl',
        model_location='auto_ml/712192f1-9c91-48ad-af46-ce1da48e1f
9b/wml_data/e39c26fa-741c-4161-bede-f8de5be66350/data/automl/pre_h
po_d_output/Pipeline1/model.pickle',
        training_status='auto_ml/712192f1-9c91-48ad-af46-ce1da48e1
f9b/wml_data/e39c26fa-741c-4161-bede-f8de5be66350/training-status.
json'
    )
)
```

In [ ]:
```
experiment_metadata = dict(
    prediction_type='regression',
    prediction_column='Production',
    holdout_size=0.1,
    scoring='neg_root_mean_squared_error',
    csv_separator=',',
    random_state=33,
    max_number_of_estimators=2,
    training_data_reference=training_data_reference,
    training_result_reference=training_result_reference,
    deployment_url='https://jp-tok.ml.cloud.ibm.com',
    project_id='2aec647c-4697-4923-87ba-df9aac7f0110',
    drop_duplicates=True
)
```

Watson Machine Learning connection

This cell defines the credentials required to work with the Watson Machine Learning service.

Action: Please provide IBM Cloud apikey following [docs (https://cloud.ibm.com/docs/account?topic=account-userapikey)](https://cloud.ibm.com/docs/account?topic=account-userapikey).

In [ ]:
```
api_key = 'PUT_YOUR_APIKEY_HERE'
```

In [ ]:
```
wml_credentials = {
    "apikey": api_key,
    "url": experiment_metadata['deployment_url']
}
```

Working with the completed AutoAI experiment

This cell imports the pipelines generated for the experiment so they can be compared to find the optimal pipeline to save as a model.

Get fitted AutoAI optimizer

In [ ]:
```
from ibm_watson_machine_learning.experiment import AutoAI

pipeline_optimizer = AutoAI(wml_credentials, project_id=experiment_metadata['project_id']).runs.get_optimizer(metadata=experiment_metadata)
```

Use `get_params()` - to retrieve configuration parameters.

In [ ]:
```
pipeline_optimizer.get_params()
```

Pipelines comparison

Use the `summary()` method to list trained pipelines and evaluation metrics information in the form of a Pandas DataFrame. You can use the DataFrame to compare all discovered pipelines and select the one you like for further testing.

In [ ]:
```
summary = pipeline_optimizer.summary()
best_pipeline_name = list(summary.index)[0]
summary
```

Get pipeline as scikit-learn pipeline model

After you compare the pipelines, download and save a scikit-learn pipeline model object from the AutoAI training job.

Tip: To get a specific pipeline pass the pipeline name in:

```
pipeline_optimizer.get_pipeline(pipeline_name=pipeline_name)
```

In [ ]:
```
pipeline_model = pipeline_optimizer.get_pipeline()
```

Next, check features importance for selected pipeline.

In [ ]:
```
pipeline_optimizer.get_pipeline_details()['features_importance']
```

Tip: If you want to check all model evaluation metrics-details, use:

```
pipeline_optimizer.get_pipeline_details()
```

Inspect pipeline

Visualize pipeline model

Preview pipeline model stages as a graph. Each node's name links to a detailed description of the stage.

In [ ]:
```
pipeline_model.visualize()
```

Preview pipeline model as Python code

In the next cell, you can preview the saved pipeline model as Python code.
You can review the exact steps used to create the model.

Note: If you want to get sklearn representation, add the following parameter to `pretty_print` call: `astype='sklearn'`.

In [ ]:
```
pipeline_model.pretty_print(combinators=False, ipython_display=True)
```

Calling the `predict` method

If you want to get a prediction using pipeline model object, call `pipeline_model.predict()` .

Note: If you want to work with pure sklearn model:

- add the following parameter to `get_pipeline` call: `astype='sklearn'` ,
- or `scikit_learn_pipeline = pipeline_model.export_to_sklearn_pipeline()`

Deploy and Score

In this section you will learn how to deploy and score the model as a web service.

Working with spaces

In this section you will specify a deployment space for organizing the assets for deploying and scoring the model. If you do not have an existing space, you can use Deployment Spaces Dashboard (https://dataplatform.cloud.ibm.com/ml-runtime/spaces?context=cpdaas) to create a new space, following these steps:

- Click **New Deployment Space**.
- Create an empty space.
- Select Cloud Object Storage.
- Select Watson Machine Learning instance and press **Create**.
- Copy `space_id` and paste it below.

Tip: You can also use the API to prepare the space for your work. Learn more here (https://github.com/IBM/watson-machine-learning-samples/blob/master/notebooks/python_sdk/instance-management/Space%20management.ipynb).

Action: assign or update space ID below

Deployment creation

In [ ]:
```
target_space_id = "PUT_YOUR_TARGET_SPACE_ID_HERE"

from ibm_watson_machine_learning.deployment import WebService

service = WebService(
    source_wml_credentials=wml_credentials,
    target_wml_credentials=wml_credentials,
    source_project_id=experiment_metadata['project_id'],
    target_space_id=target_space_id
)
service.create(
    model=best_pipeline_name,
    metadata=experiment_metadata,
    deployment_name='Best_pipeline_webservice'
)
```

Use the `print` method for the deployment object to show basic information about the service:

In [ ]:
```
print(service)
```

To show all available information about the deployment use the `.get_params()` method.

In [ ]:
```
service.get_params()
```

Scoring of webservice

You can make scoring request by calling `score()` on the deployed pipeline.

If you want to work with the web service in an external Python application,follow these steps to retrieve the service object:

- Initialize the service by `service = WebService(wml_credentials)`
- Get deployment_id by `service.list()` method
- Get webservice object by `service.get('deployment_id')` method

After that you can call `service.score(score_records_df)` method. The `score()` method accepts `pandas.DataFrame` object.

Deleting deployment

You can delete the existing deployment by calling the `service.delete()` command. To list the existing web services, use `service.list()` .

Running AutoAI experiment with Python API

If you want to run the AutoAI experiment using the Python API, follow these. The experiment settings were generated basing on parameters set in the AutoAI UI.

- Go to your COS dashboard.
- In Service credentials tab, click New Credential.
- Add the inline configuration parameter: `{“HMAC”:true}` , click Add. This configuration parameter adds the following section to the instance credentials, (for use later in this notebook):

```
”cos_hmac_keys”: {
   “access_key_id”: “***“,
   “secret_access_key”: “***”
}
```

Action: Please provide cos credentials in following cells.

- Use provided markdown cells to run code.

```
from ibm_watson_machine_learning.experiment import AutoAI

experiment = AutoAI(wml_credentials, project_id=experiment_metadata['pr
oject_id'])
```

```
#@hidden_cell
cos_hmac_keys = {
    "access_key_id": "PLACE_YOUR_ACCESS_KEY_ID_HERE",
    "secret_access_key": "PLACE_YOUR_SECRET_ACCESS_KEY_HERE"
  }

cos_api_key = "PLACE_YOUR_API_KEY_HERE"
OPTIMIZER_NAME = 'custom_name'
```

```
from ibm_watson_machine_learning.helpers import DataConnection
from ibm_watson_machine_learning.helpers import S3Connection, S3Locatio
n

training_data_reference = [
    DataConnection(
    connection=S3Connection(
        api_key='CFTaknyIPqotjaHSVO2OiaNGeNapmkI_D07uyYYCC4nW',
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3.ap.cloud-object-storage.appdomain.clou
d',
        access_key_id = cos_hmac_keys['access_key_id'],
        secret_access_key = cos_hmac_keys['secret_access_key']
    ),
        location=S3Location(
            bucket='aiassistedfarming-donotdelete-pr-vdhfw2plbkonqu',
            path='crop_production.csv'
        )
    ),
]
training_result_reference = DataConnection(
    connection=S3Connection(
        api_key=cos_api_key,
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3.ap.cloud-object-storage.appdomain.clou
d',
        access_key_id = cos_hmac_keys['access_key_id'],
        secret_access_key = cos_hmac_keys['secret_access_key']
    ),
    location=S3Location(
        bucket='aiassistedfarming-donotdelete-pr-vdhfw2plbkonqu',
        path='auto_ml/712192f1-9c91-48ad-af46-ce1da48e1f9b/wml_data/e39
c26fa-741c-4161-bede-f8de5be66350/data/automl',
        model_location='auto_ml/712192f1-9c91-48ad-af46-ce1da48e1f9b/wm
l_data/e39c26fa-741c-4161-bede-f8de5be66350/data/automl/pre_hpo_d_outpu
t/Pipeline1/model.pickle',
        training_status='auto_ml/712192f1-9c91-48ad-af46-ce1da48e1f9b/w
ml_data/e39c26fa-741c-4161-bede-f8de5be66350/training-status.json'
    )
)
```

The new pipeline optimizer will be created and training will be triggered.

```
pipeline_optimizer = experiment.optimizer(
    name=OPTIMIZER_NAME,
    prediction_type=experiment_metadata['prediction_type'],
    prediction_column=experiment_metadata['prediction_column'],
    scoring=experiment_metadata['scoring'],
    holdout_size=experiment_metadata['holdout_size'],
    csv_separator=experiment_metadata['csv_separator'],
    drop_duplicates=experiment_metadata['drop_duplicates'],
)
```

```
pipeline_optimizer.fit(
    training_data_reference=training_data_reference,
    training_results_reference=training_result_reference,
    background_mode=False,
)
```

Next steps

Online Documentation (https://www.ibm.com/cloud/watson-studio/autoai)

Copyrights