marlabs
driving digital agility

Madvisor -AutoML

Table of Contents

Key Features:

- Pre-process and clean the data.
- Select and construct appropriate features.
- Select an appropriate model family.
- Optimize model hyperparameters.

Azure mAdvisor AutoML API will take care of data preparation to model building.

API Usage:

User Registration and Plan Subscription

A Licence Key is required to use this container image, register yourself to activate the 30 days free trial. Connect with Marlabs mAdvisor team to purchase a paid licence.

1.0. POST https://madvisor-dbc.marlabsai.com/automl/account/sign-up

End point for creating user account. Verification link will be sent to email on successful post request.

Request Header

email: Valid email ID

required: yes

username: Username for account

required: yes

password: Password for account

required: yes

2.0. POST https://madvisor-dbc.marlabsai.com/automl /subscription/subscribe-plan/

From the provided username and password, user authentication is done, if authentication is successful, token will be generated. Token will be shared to the registered email ID of the customer post successful token generation.

Request Header

username: username provided while creating account.

password: user account password

subscription_type: TRIAL

plan: FREE

3.0. POST https://madvisor-dbc.marlabsai.com/automl/subscription/active-plans/

This end point can be used to get the details of active subscriptions

Request Header

username : username provided while creating account.

passsword : user account password

AutoML

To start using the AutoML training, prediction and feature engineering

4.0. POST /automl

Used to initiate automl job, both model training and scoring can be done using this end point.

4.1. Request Header

target: target variable name

required: yes

label_level: Target variable sub-level to calculate model evaluation metrics, not required in case of regression.

required: Optional, Sub-level with maximum number of occurance is selected by default.

app_type: Whether it is a "classification" or "regression" problem.

required: Optional, AutoML automatically tags the app_type in case of no user input.

prediction_algorithm: algorithm to be used for prediction.

required: Optional, Best algorithm in selected by default.

for "classification" use: 'Random Forest', 'XGBoost', 'LightGBM'

for "regression" use: 'Linear Regression', 'GBT Regression', 'DTREE Regression' or 'RF Regression'

token: Token generated on subscription

required: yes

4.2. Request Body

The request body includes the following form parameter:

train_file : train dataset to be used for training the model

file format: csv

required: yes

test_file : test dataset to do prediction with the trained model

file format: csv

required: yes

4.3. Response Header

content-type: HTTP content type

supported values: “application /json”

4.4. Response Body

model_result:

model_slug: value is used to refer to trained model

type: String

model_evaluation_results: model evaluation matrix

type: JSON

score_result:

prediction rules: Rules generated from the data in natural language

type: List

predicted data: test dataset with prediction and probabilities

type: JSON

4.5. Example

Request

POST /document

headers {"target": , "label_level": ,"app_type": ,”prediction_algorithm": ,”token”:}

body {"train_file": , "test_file": }

Response :


```
{
  "model result": {
    "model_slug": "Model-129svoaupl",
    "model_evaluation_results": {
      "Algorithm": [ "XGBoost","Random Forest","LightGBM"],
      "Accuracy":  ["100","98","100"],
      "Precision": ["100","100", "100"],
      "Recall": [ "100","98", "100"],
      "ROC-AUC": [ "100", "99","100"]
    }
  },
  "score result": {
    "prediction rules": [
      "If the PetalLengthCm does not fall in ['Above Average', 'Average', 'Below Average', 'Low'] and the PetalWidthCm falls among ['Average', 'Below Average', 'Low'] then there is  <b>100%  <b>probability that the Species would be Iris-virginica.",
```

"If the PetalLengthCm does not fall in ['Below Average', 'Low'] and the PetalWidthCm does not fall in ['Average', 'Below Average', 'Low'] then there is <b>100% <b>probability that the Species would be Iris-virginica.",

"If the PetalLengthCm falls among ['Below Average', 'Low'] then there is <b>100% <b>probability that the Species would be Iris-setosa.",

"When the PetalLengthCm falls among ['Above Average', 'Average'], the PetalLengthCm does not fall in ['Below Average', 'Low'] and the PetalWidthCm falls among ['Average', 'Below Average', 'Low'] then there is <b>100% <b>chance that Species would be Iris-versicolor."

],

"predicted data": {

<testdata with prediction>

}

}

}

5.0 POST automl/train

Used to initiate AutoML training job.

5.1. Request Header

target: target variable name

required: yes

label_level: Target variable sub-level to calculate model evaluation metrics, not required in case of regression.

required: Optional, Sub-level with maximum number of occurance is selected by default.

app_type: Whether it is a "classification" or "regression" problem.

required: Optional, AutoML automatically tags the app_type in case of no user input.

token: Token generated on subscription

required: yes

5.2. Request Body

The request body includes the following **form** parameter:

train_file : train dataset to be used for training the model

file format: csv

required: yes

5.3. Response Header

content-type: HTTP content type

supported values: “application /json”

5.4. Response Body

model_result:

model_slug: value is used to refer to trained model

type: String

model_evaluation_results: model evaluation matrix

type: JSON

5.5. Example

Request

POST /document

headers {"target": , "label_level": ,"app_type":,” token”: }

body {"train_file": }

Response:


```
{
  "model result": {
    "model_slug": "Model-129svoaupl",
    "model_evaluation_results": {
      "Algorithm": [ "XGBoost","Random Forest","LightGBM"],
      "Accuracy":  ["100","100","100"],
      "Precision": ["100","100", "100"],
      "Recall": [ "100","100", "100"],
      "ROC-AUC": [ "100", "100","100"]
    }
  }
}
```


6.0 POST automl/score

Used to initiate automl scoring job.

6.1. Request Header

training_slug: Training slug is the unique name given to the model while training.

required: yes

prediction_algorithm: algorithm to be used for prediction.

required: Optional, Best algorithm in selected by default.

for "classification" use: 'Random Forest', 'XGBoost', 'LightGBM'

for "regression" use: 'Linear Regression', 'GBT Regression', 'DTREE Regression' or 'RF Regression'

token: Token generated on subscription

required: yes

6.2. Request Body

The request body includes the following **form** parameter:

test_file : test dataset to do prediction with the trained model

file format: csv

required: yes

6.3. Response Header

content-type: HTTP content type

supported values: “application /json”

6.4. Response Body

score_result:

prediction rules: Rules generated from the data in natural language

type: List

predicted data: test dataset with prediction and probabilities

type: JSON

6.5. Example

Request

POST /document

headers {"target": , "label_level": ,"app_type": , "training_slug": ,”prediction_algorithm": ,” token”:}

body {"train_file": , "test_file": }

Response:

{

"score result": {

"prediction rules": [

"If the PetalLengthCm does not fall in ['Above Average', 'Average', 'Below Average', 'Low'] and the PetalWidthCm falls among ['Average', 'Below Average', 'Low'] then there is <b>100% <b>probability that the Species would be Iris-virginica.",

"If the PetalLengthCm does not fall in ['Below Average', 'Low'] and the PetalWidthCm does not fall in ['Average', 'Below Average', 'Low'] then there is <b>100% <b>probability that the Species would be Iris-virginica.",

"If the PetalLengthCm falls among ['Below Average', 'Low'] then there is <b>100% <b>probability that the Species would be Iris-setosa.",

"When the PetalLengthCm falls among ['Above Average', 'Average'], the PetalLengthCm does not fall in ['Below Average', 'Low'] and the PetalWidthCm falls among ['Average', 'Below Average', 'Low'] then there is <b>100% <b>chance that Species would be Iris-versicolor."

],

"predicted data": {

<testdata with prediction>

}}}

7.0. POST automl/auto-fe

Used for automated data preparation of train data. Data Pre-processing, Feature Engineering and Feature Selection are done in a single go. Output will have 2 datasets, one for linear algorithms and another for tree-based algorithms.

7.1. Request Header

target: target variable name

required: yes

app_type: classification/regression

required: optional

token: Token generated on subscription

required: yes

7.2. Request Body

The request body includes the following **form** parameter:

train_file : dataset

file format: csv

required: yes

7.3. Response Header

content-type: HTTP content type

supported values: “application /json”

7.4. Response Body

Output dataset

type: JSON

8.0 POST automl/auto-fe-test-data

Used for automated data preparation of test dataset. Slug value got from train dataset processing must be used here for proper pre-processing feature engineering and feature selection of the test data. Output will have 2 datasets, one for linear algorithms and another tree-based algorithms.

8.1. Request Header

Fe-slug: Slug value got from train data feature engineering

required: yes

token: Token generated on subscription

required: yes

8.2. Request Body

The request body includes the following **form** parameter:

test_file : dataset

file format: csv

required: yes

8.3. Response Header

content-type: HTTP content type

supported values: “application /json”

8.4. Response Body

Output dataset

type: JSON

9.0 Examples with Python Request module:

1.Install python request module:

Requests is a simple, yet elegant HTTP library.

- run 'pip install requests' in your python environment.

2. Install python requests_toolbelt module:

For streaming multipart form-data object.

- run 'pip install requests_ toolbelt' in your python environment.

Examples:

9.1. For account creation

```
import requests
base_url = 'https://madvisor-dbc.marlabsai.com/automl'
url = base_url + '/account/sign-up'
data= {'email': '<email id>',
 'username':'<user name>',
 'password':'<password>'
}
response = requests.post(url, data=data)
if response.status_code != 200:
    print('Failed response code {}'.format(response.status_code))
print('Output result: {}'.format(response.json()))
```

9.2. For subscription

```
base_url = 'https://madvisor-dbc.marlabsai.com/automl'
url = base_url+'/subscription/subscribe-plan'
data= {'username':'<user name>',
 'password':'<password>',
 'subscription_type': 'TRIAL',
 'plan': 'FREE'
 }

response = requests.post(url, data=data)
if response.status_code != 200:
    print('Failed response code {}'.format(response.status_code))
print('Output result: {}'.format(response.json()))
```

9.3. Check Active Subscriptions

```
base_url = 'https://madvisor-dbc.marlabsai.com/automl'
url = base_url+'/subscription/active-plans'
data= {'username':'<user name>',
 'password':'<password>'
 }

response = requests.post(url, data=data)
if response.status_code != 200:
    print('Failed response code {}'.format(response.status_code))
print('Output result: {}'.format(response.json()))
```

9.4. For automl Job:

```
import requests
from requests_toolbelt.multipart.encoder import MultipartEncoder
import os
# This should be your base url
base_url = 'your base url from deployment'
url = base_url + '/automl'
# train data
train_file = "<path to your train file (csv)>"
file_stats = os.stat(train_file)
file_name = os.path.basename(train_file)
# test data
test_file = "<path to your test file (csv)>"
file_stats_test = os.stat(test_file)
file_name_test = os.path.basename(test_file)
# file upload
multipart_data = MultipartEncoder(fields={
    'train_file': (file_name, open(train_file, "rb")),
    'test_file': (file_name_test, open(test_file, "rb"))
})
# define header
head = {"Content-Type": multipart_data.content_type, \
        "target": "Species", "label_level": None, \
        "app_type": None, \
        "prediction_algorithm": None,
        "token": "License Key"
}
# Making post request for automl job
response = requests.post(url, data=multipart_data, headers=head)
if response.status_code != 200:
    print('Failed response code {}'.format(response.status_code))
else:
    print('Output result: {}'.format(response.json()))
```

9.5. For automl train job:

```
#base_url should be your endpoit entry point.
base_url = 'your base url from deployment'
url = base_url + '/automl/train'
# path for train data
train_file = '<path to your train file (csv)>'
file_stats = os.stat(train_file)
file_name = os.path.basename(train_file)
# file upload field
multipart_data = MultipartEncoder(fields={'train_file': (file_name,
                                  open(train_file, 'rb'))})
# define header
head = {
    'Content-Type': multipart_data.content_type,
    'target': 'Species',
    'label_level': None,
    'app_type': None, #'classification or regression',
    'token': 'License Key'
    }
# request for automl train job.
response = requests.post(url, data=multipart_data, headers=head)
if response.status_code != 200:
    print('Failed response code {}'.format(response.status_code))
else:
    print('Output result: {}'.format(response.json()))
```

9.6. For automl scoring job:

>We need to use the obtained training slug from train job: eg. “`Model-5g3ku4bl5i`”

```
base_url = 'your base url from deployment'
url = base_url + '/automl/score'
# path for test data
test_file= '<path to your test file (csv)>'
file_stats_ = os.stat(test_file)
file_name = os.path.basename(test_file)
# Multiple file upload
multipart_data = MultipartEncoder(fields={'test_file': (file_name,\
                                        open(test_file, 'rb'))})
# define header
head = {"Content-Type": multipart_data.content_type,
        "training-slug": "Model-5g3ku4bl5i",
        "token": "License Key"}
#request for automl score job.
response = requests.post(url, data=multipart_data, headers=head)
if response.status_code != 200:
```

```
    print('Failed response code {}'.format(response.status_code))
else:
    print('Output result: {}'.format(response.json()))
```

9.7 For Auto Feature Engineering for Train data

```
base_url = 'your base url from deployment'
url = base_url + '/automl/auto-fe'
# path for test data
train_file= '<path to your train file (csv)>'
file_stats_ = os.stat(train_file)
file_name = os.path.basename(train_file)
# Multiple file upload
multipart_data = MultipartEncoder(fields={'train_file': (file_name,\
                                        open(train_file, 'rb'))})
# define header
head = {'Content-Type': multipart_data.content_type,
        "app_type" : "classification or regression",
        "token": "License Key",
        "target": "Target variable name"}
#request for automl feature engineering job on train data.
response = requests.post(url, data=multipart_data, headers=head)
if response.status_code != 200:
    print('Failed response code {}'.format(response.status_code))
else:
    print('Output result: {}'.format(response.json()))
```

9.8 For Auto Feature Engineering for test data

```
base_url = 'your base url from deployment'
url = base_url + '/automl/auto-fe-test-data'
# path for test data
test_file= '<path to your test file (csv)>'
file_stats_ = os.stat(test_file)
file_name = os.path.basename(test_file)
# Multiple file upload
multipart_data = MultipartEncoder(fields={'test_file': (file_name,\
                                        open(test_file, 'rb'))})
# define header
head = {"Content-Type": multipart_data.content_type,
        "fe-slug" : "slug value from train feature engineering",
```

```
        "token" : "License Key"}
#request for automl feature engineering job on test data.
response = requests.post(url, data=multipart_data, headers=head)
if response.status_code != 200:
    print('Failed response code {}'.format(response.status_code))
else:
    print('Output result: {}'.format(response.json()))
```