User Guide for Aitomatic library

Introduction

The WebModel class is a Python wrapper that allows you to make inference calls to models served via the Aitomatic Web API. The class takes as input your Aitomatic API token and the name of the model that you want to use, and provides a simple interface for making predictions.

Installation

```
pip install -i https://test.pypi.org/simple/ aitomatic
```

Authentication

You will need an API token from your Aitomatic account to access the model
The best way to use the API_KEY is setting it as the env variable `API_TOKEN`
Once you have it set up you can start using the library for inferencing, building or tuning model

The call using in this library is making to
`https://model-api-prod.platform.aitomatic.com`
`https://production.platform.aitomatic.com/api/client`

Environment variables

Before using the API, you can set the following environment variables

```
AITOMATIC_API_TOKEN=
PROJECT_NAME=
```

Alternatively, you can always send them through the API call as `api_token` and `project_name`

Inferencing with WebModel

```
from aitomatic.api.web_model import WebModel

model = WebModel(api_token=api_token, project_name=project_name, model_name=model_name)

input_data = read_table(data_path).to_pandas()
result = model.predict({'X': input_data})
```

where project_name and model_name should be provided appropriately and the input X should be a Pandas dataframe

Building models

Build a model

To build the model from the library, you can instatiate a ModelBuilder and call `build_model` with the appropriate parameters

```
from aitomatic.api.build import ModelBuilder

builder = ModelBuilder()
resp = builder.build_model(model_type=model_type, model_name=model_name, knowledge_name=
```

Where

- `model_type` is among `{'ORACLE', 'K-COLLABORATOR'}`
- `model_name` is the model name for the build
- `knowledge_name` is the knowledge name that the model is building from
- `data_name` is the data name that the model is building from
- `params` is a dict that contain the the parameters and hyper-parameters for the k-architecture model

Example of params:

Access model parameter

- To retrive the params from a built model you can call `get_existing_model_params`

  ```
  builder.get_base_model_params(model_name=model_name)
  ```

- `model_name` the name of the model to retrieve the params from

- To retrieve the base parameters for a model

```
base_params = builder.get_base_model_params(model_type=model_type, knowledge_set_nar
```

  - `model_type` is among `{'ORACLE', 'K-COLLABORATOR'}`
  - `knowledge_set_name` is the knowledge set that the model built from
  - `students` is an array of ml model names among `{'CNNClassifier', 'K-XGBCLassifier', 'LinearRegression', 'RandomForest'}`