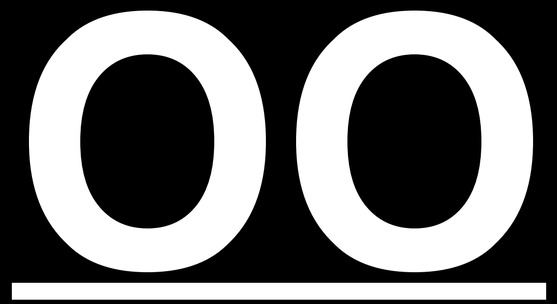
DS Team

Requirements

1. Change your working directory to our repository
2. git pull
3. pip install redis

Introduction

- **Modules**
  - redis module
    - NoSQL Database, M-Queue
  - click module
    - similar with getup module
  - sklearn module
  - keras module
  - pandas module

- **Programs**
  - algorithm.py - modeling class
  - builder.py - A wrapper class includes dataset and algorithm classes
  - app.py - main script to execute the above tasks

```
class IrisModelBuilder(object):
    def __init__(self, model_path, channels):
        self.model_path = model_path
        self.channels = channels

        self.redis = redis.Redis()
        self.pubsub = self.redis.pubsub()
        self.pubsub.subscribe(self.channels)

    def build(self):
        print("build iris model... ")

        # train model
        iris_dnn = IrisDNN(self.model_path)
        iris_dnn.train_model()
        iris_dnn.save_model()

    def run(self):
        print("Listen build {} model command...".format(self.channels))

        for item in self.pubsub.listen():
            if item['data'] == EVENT_TRAIN:
                self.build()
```

初始化方法

建模方法論

傾聽事件發生，若有
「EVENT_TRAIN」發生，則重建模型

Hands On

Use any Tree-Based model instead of Keras

1. add a new class in algorithm.py
2. modify the builder.py
3. take the reference Miles' sharing last Friday

Is it friendly for Modeler / Analyzer ?

```
class IrisModelBuilder(object):
    def __init__(self, model_path, channels):
        self.model_path = model_path
        self.channels = channels

        self.redis = redis.Redis()
        self.pubsub = self.redis.pubsub()
        self.pubsub.subscribe(self.channels)

    def build(self):
        print("build iris model... ")

        # train model
        iris_dnn = IrisDNN(self.model_path)
        iris_dnn.train_model()
        iris_dnn.save_model()

    def run(self):
        print("Listen build {} model command...".format(self.channels))

        for item in self.pubsub.listen():
            if item['data'] == EVENT_TRAIN:
                self.build()
```

不重要！

其實，只有這地方是 Modeler / Analyzer 重視！

更不重要！

Object-Oriented Concept

```
class MLBuilder(object):
    def __init__(self, dataset_path, model_path, channels):
        self.model_path = model_path
        self.dataset_path = dataset_path
        self.channels = channels

        self.listener = MLListener(channels=channels)

    def build(self):
        raise NotImplementedError

    def run(self):
        print("Listen build {} model command...".format(self.listener.get_channels()))

        for item in self.listener.listen():
            if item['data'] == EVENT_TRAIN:
                self.build()
```

```
from interface import MLBuilder

from algorithm import IrisDNN

class IrisModelBuilder(MLBuilder):
    def build_model(self):
        print("build iris model... ")

        # train model
        iris_dnn = IrisDNN(self.model_path)
        iris_dnn.train_model()
        iris_dnn.save_model()
```

Hands On

```
from interface import MLBuilder

from algorithm import IrisDNN

class IrisModelBuilder(MLBuilder):
    def build(self, model):
        print("build iris model... ")

        # train model
        model.__init()
        model.train()
        model.save()
```

```
class MLAlgorithm(object):
    def __init(self, model_path):
        iris = load_iris()
        self.X = iris.data
        self.Y = iris.target

        self.model_path = model_path

    def split_data(self):
        train_X, test_X, train_y, test_y = train_test_split(
            self.X, self.Y, train_size=0.5, random_state=0)
        train_y_ohe, uniques = encode_one_hot(train_y)
        test_y_ohe, _ = encode_one_hot(test_y)
        self.classes = uniques

        return train_X, train_y_ohe, test_X, test_y_ohe

    def build_network(self):
        model = Sequential()
        # layer 1: 4 -> 16
        model.add(Dense(input_dim=4, output_dim=16))
        model.add(Activation(('relu')))
        model.add(Dropout(0.2))
        # layer 2: 16 -> 16
        model.add(Dense(output_dim=16))
        model.add(Activation(('relu')))
        model.add(Dropout(0.2))
```