model_api

October 30, 2018

1 Model API

In [3]:
```
"""
API for the bike rentals prediction model.
"""

import sklearn as sl
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.utils import shuffle
from sklearn.model_selection import cross_val_score
import pandas as pd
import warnings


# MODEL META SPECIFICATION ==================================
# Hardcoded for now, should be loaded from model persistency

model_class = GradientBoostingRegressor(
                n_estimators = 150,
                max_depth = 7)

attributes = {'features': ['season', 'yr', 'mnth', 'hr', 'weekday', 'workingday',
                           'weathersit', 'temp', 'atemp', 'hum', 'windspeed'],
              'target': 'cnt'}

assessment = {'n_cv': 2,
              'min_avg_abs_err': 50}

# ===========================================================


model_in_charge = None


def _model_class_assessment(X, Y):
    scores = cross_val_score(model_class, X, Y,
                             cv=assessment['n_cv'],
```

```
                                scoring= 'neg_mean_absolute_error')
        avg_cv_error = abs(scores.mean())
        if avg_cv_error > assessment['min_avg_abs_err']:
            warnings.warn("Model assessment not passed. Prediction error inacceptable.")
        return avg_cv_error



    # EXPOSED API ================================================

    def calibrate_model(data_csv_file):
        """Returns a model calibrated on the provided csv file. The model will be of the t
        data = pd.read_csv(data_csv_file)
        X, Y = shuffle(data[attributes['features']].values,
                               data[attributes['target']].values)

        avg_cv_error = _model_class_assessment(X, Y)
        model = model_class.fit(X,Y)
        print("Model successful calibrated with an avg absolute error of", avg_cv_error)
        return model


    def set_model(model) -> None:
        """Sets the model to be in charge after a sanity check first."""
        #if formal_sanity_check():
        global model_in_charge
        model_in_charge = model


    def apply_model(data_csv_file):
        """Returns predictions of the model in charge onto the provided data."""
        X = pd.read_csv(data_csv_file)[attributes['features']].values
        Y_predicted = model_in_charge.predict(X)
        return Y_predicted
```

1.1 Sample of Usage

```
In [2]: test_csv = 'c:\\dev\\bike\\data\\hour.csv'
        model = calibrate_model(test_csv)
        set_model(model)
        print("\nThe following model in currently in charge:",model_in_charge)
        apply_model(test_csv)[1:10]
```

```
Model successful calibrated with an avg absolute error of 25.84788458786423

The following model in currently in charge: GradientBoostingRegressor(alpha=0.9, criterion='fr
             learning_rate=0.1, loss='ls', max_depth=7, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
```

```
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=150, presort='auto', random_state=None,
            subsample=1.0, verbose=0, warm_start=False)


Out[2]: array([26.56119772, 20.85544789,  7.26543453,  1.29762827, -4.89840759,
                1.32193112, -5.32916457,  8.80788073, 45.1557695 ])
```