**method: return** (return the predicted value)

Class: Changesythesiser (Injecting anomalies)

**method: detect method: detect\_by\_regression**.

(training and predicting using regression model)

**Preprocessor.py and timeseriesdecomposition.py**

(Removing the trend and seasonality)

Data Flow for Correlation\_change

**Class: evaluate**

(return auc and prauc score.)

**Starting with example\_change.py:**

* Importing the csv file

**Class: Changesythesiser (Injecting anomalies)**

* From sythesiser.py
* Method used- like add\_point\_anomalies and add\_segment\_anomalies
* Output: syn\_series1 ,syn\_series2 and syn\_labels.

**Class: ChangeController (ChangeController implements an automagical pipeline of time series correlation changes detection)**

* Preprocessor.py: Removing the trend and seasonality in the data.

method: detect method: detect\_by\_regression predictor used= **RF, SLR, HR, RT)**

output: dict\_series

* Input used for training regression model:

model1 = RegressionPredictor(predictor)  
model1.train(first\_train\_features1, first\_train\_response1)  
model2 = RegressionPredictor(predictor)  
model2.train(first\_train\_features2, first\_train\_response2)

* Prediction using regression**:**

this\_predicted\_series1 = model2.predict(this\_predicted\_features2)  
this\_practical\_series1 = self.dict\_series[**"detected\_series1"**][st:ed]  
this\_predicted\_series2 = model1.predict(this\_predicted\_features1)  
this\_practical\_series2 = self.dict\_series[**"detected\_series2"**][st:ed]

* method: \_filter\_anomaly\_regression

return: the change ratios of time series 1, the change ratios of time series 2, the change labels to identity the anomalies. the change scores measuring the anomaly degree.

* **Controller.return: return the predicted value**

**Class: evaluate** (return auc and prauc score.)