References

[1] Swissgrid - The Swiss transmission system, Swissgrid.ch, 2016. [Online]. Available: https://www.swissgrid.ch/swissgrid/en/home/grid/transmission_system.html. [Accessed: 13- Nov- 2016].

[2] “Swissmod”. [Online]. Available http://simlab.ethz.ch/swissmod.php.[Accessed: 24-Apr-2017]

[3] Z. Lukszo, *Securing electricity supply in the cyber age*, 1st ed. Dordrecht: Springer, 2010, pp. 60-62.62

[4] C. CHARLTON, "Germany PAYS people to use electricity: Excess energy created by wind and solar power meant consumers made a profit as prices were driven so low they went NEGATIVE," 10:20 BST, 11 May 2016

[5] Forecasting Electricity Demand."Forecasting Electricity Demand", Siemens.com, 2016. [Online]. Available: http://www.siemens.com/innovation/en/home/pictures- of-the-future/energy-and-efficiency/power-transmission-forecasting-electricity-demand.html. [Accessed: 12- Nov- 2016].

[6] A. Boiron, S. Lo and A. Marot, "Predicting Future Energy Consumption", 2012.

[7] P. Dagnely, T. Ruette, T. Tourwé, E. Tsiporkova and C. Verhelst, "Predicting Hourly Energy Consumption. Can Regression Modeling Improve on an Autoregressive Baseline?", in *Proceedings of 24th Annual Machine Learning Conference*, Benelearn, Delft, The Netherlands, 2015, pp. Pages 105-122.1.

[8] M. Larsson, L. F. Santos, A. Suranyi, W. Sattinger and R. Notter, "Monitoring of oscillations in the continental European transmission grid," *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Vienna, 2013, pp. 4774-4778.

[9] “Wide Area Monitoring”. [Online]. Avaialble https://www.swissgrid.ch/swissgrid/fr/home/reliability/wam.html [Accessed: 24-Apr-2017] [Accessed: 24-Apr-2017]

[10] Predicting the price of electricity is as uncertain as the weather |

@guardianletters."Predicting the price of electricity is as uncertain as the weather | @guardianletters", theGuardian,2016. [Online].Available:https://www.theguardian.com/environment/2014/jul/14/predicting-price-of-electricity-uncertain. [Accessed: 13- Nov- 2016].

[11] Smart Learning Software Predicts Renewable Energy Output with 90% Accuracy."Smart Learning Software Predicts Renewable Energy Output with 90% Accuracy", T reeHugger,2016. [Online].Available:http://www.treehugger.com/clean-technology/learning-software-predicts-renewable-energy-output.html. [Accessed: 12- Nov- 2016].

[12]"Introducing DAS - Data Analytics Server 3.1.0 - WSO2 Documentation", Docs.wso2.com,2016.[Online].Available: https://docs.wso2.com/display/DAS310/Introducing+DAS. [Accessed: 12-Nov- 2016].

[13]”Batch Analytics Using Spark SQL”. [Online]. Available https://docs.wso2.com/display/DAS310/Batch+Analytics+Using+Spark+SQL. [Accessed: 24-Apr-2017]

[14] “Architecture”. [Online]. Available https://docs.wso2.com/display/DAS310/Architecture [Accessed: 24-Apr-2017] [Accessed: 24-Apr-2017]

[15]”Rolling-Window Analysis of Time-Series Modelss”. [Online]. Available https://www.mathworks.com/help/econ/rolling-window-estimation-of-state-space-models.html?requestedDomain=true. [Accessed: 24-Apr-2017]

[16] S. Perera, "Rolling Window Regression: a Simple Approach for Time Series Next value Predictions", *Medium*, 2017. [Online]. Available: https://medium.com/making-sense-of-data/time-series-next-value-prediction-using-regression-over-a-rolling-window-228f0acae363. [Accessed: 03- June- 2016].

[17] scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation."scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation", Scikit-learn.org, 2016. [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 12- Nov- 2016].

[18] "Python Data Analysis Library — pandas: Python Data Analysis Library." [Online]. Available: http://pandas.pydata.org/. [Accessed: 24-Apr-2017].

[19] "Mean absolute error," *Wikipedia*. 24-Apr-2017.

[20] Quantum, "FAST DATA SCIENCE HELPS SWITZERLAND PLAN FOR ELECTRICITY SHORTAGES," 01-Feb-2016. [Online]. Available: http://quantumanalytics.ch/wordpress/wp-content/uploads/2016/03/Quantum-Analytics-Swissgrid-electricity-shortage-in-CH.pdf

[21]"InnovationNewsPressSiemensChina."[Online].Available:http://w1.siemens.com.cn/news_en/frontier_technology_en/2294.aspx. [Accessed: 13- Nov- 2016].

[22] Power transmission."Power transmission", Siemens.com, 2016. [Online]. Available:http://www.siemens.com/innovation/en/home/pictures-of-the- future/energy-and-efficiency/power-transmission-electrictiy-superhighways.html. [Accessed: 13- Nov- 2016].

[23] Smart Power Transmission Saves Millions."Smart Power Transmission Saves Millions",Siemens.com,2016.[Online].Available: http://www.siemens.com/innovation/en/home/pictures-of-the-future/energy-and-efficiency /power-transmission-facts-and-forecasts.html. [Accessed: 12- Nov- 2016].https://www.google.lk/search?q=Data-Stream-Mining-A-Review-on-Windo&oq=Data-Stream-Mining-A-Review-on-Windo&aqs=chrome..69i57j69i60.262j0j4&sourceid=chrome&ie=UTF-8#q=PredictingFutureEnergyConsumption.pdf]

[24] "Swissgrid - Production and consumption." [Online]. Available: https://www.swissgrid.ch/swissgrid/en/home/reliability/griddata/generation.html https://www.google.lk/search?q=Data-Stream-Mining-A-Review-on-Windo&oq=Data-Stream-Mining-A-Review-on-Windo&aqs=chrome..69i57j69i60.262j0j4&sourceid=chrome&ie=UTF-8#q=PredictingFutureEnergyConsumption.pdf] [Accessed: 24-Apr-2017]

[25] Swissgrid - Smart Grid."Swissgrid - Smart Grid", Swissgrid.ch, 2016. [Online]. Available:https://www.swissgrid.ch/swissgrid/en/home/future/smartgrid.html. [Accessed: 13- Nov- 2016].

[26] Swissgrid - Supergrid."Swissgrid - Supergrid", Swissgrid.ch, 2016. [Online]. Available:https://www.swissgrid.ch/swissgrid/en/home/future/supergrid.html. [Accessed: 13- Nov- 2016].

[27] Volatile but predictable: Forecasting renewable power generation."Volatile but predictable: Forecasting renewable power generation", Clean Energy Wire, 2016. [Online]. Available: https://www.cleanenergywire.org/factsheets/volatile- predictable-forecasting-renewable-p ower-generation. [Accessed: 13- Nov- 2016].

[28] "Swissgrid -Production and consumption." [Online].Available: https://www.swissgrid.ch/swissgrid/en/home/reliability/griddata/generation.html. [Accessed: 24-Apr-2017]

[29] E. Zivot and J. Wang, *Modeling financial time series with S-Plus*, 1st ed. New York: Springer, 2006, pp. 299-346.

Event Publisher

```
public class ResearchEventPublisher{

   private static Log log = LogFactory.getLog(ResearchEventPublisher.class);
   private static DataPublisher privateDataPublisher;
   private static DataPublisher currentDataPublisher;
   private static int count = 0;
   public static void main(String[] args) {
                                        System.setProperty("org.xml.sax.driver",
"com.sun.org.apache.xerces.internal.parsers.SAXParser");
       System.setProperty("javax.xml.parsers.DocumentBuilderFactory","com.sun.org.
apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl");
      System.setProperty("javax.xml.parsers.SAXParserFactory","com.sun.org.apache
.xerces.internal.jaxp.SAXParserFactoryImpl");
            System.out.println("Starting WSO2 Event ResearchEventPublisher Stream
Client");
          AgentHolder.setConfigPath(publisher.schedular.util.DataPublisherUtil.filePat
h + "/src/main/java/files/configs/data-agent-config.xml");
         publisher.schedular.util.DataPublisherUtil.setTrustStoreParams();
         String protocol = "thrift";
         String singleNodeHost = "tcp://localhost:7611";
         String username = "admin";
         String password = "admin";
      try{
           privateDataPublisher = new DataPublisher(protocol, singleNodeHost, null,
username, password);
         currentDataPublisher = privateDataPublisher;
                                                List<Object[]> eventsList =
```

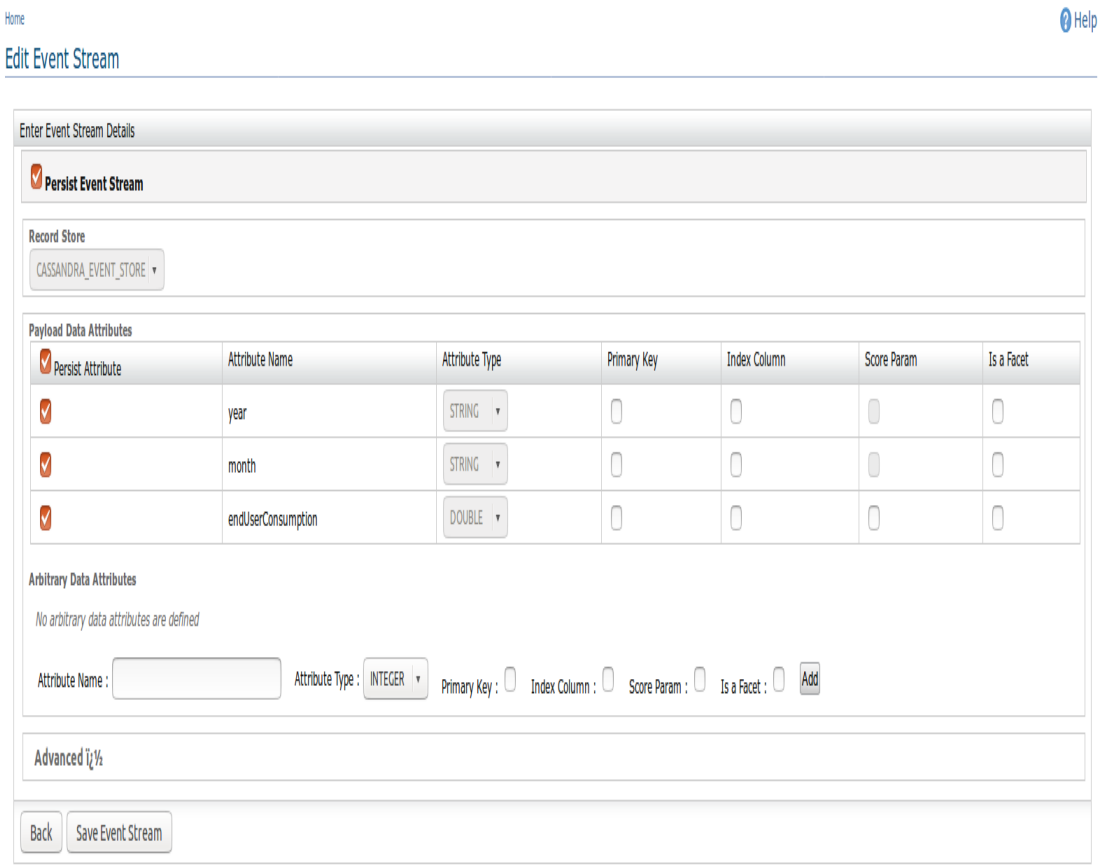
```
StatisticsInputReaderTask.readCurrentValuesFromFile("/home/shani/MSC/research/swiss/resources/swissDataGridData2016.csv");
            for (Object[] eventpayload : eventsList){
                publishEvent(eventpayload, "ControlBlock:1.0.0");
            }
        }catch(Exception e) {
            log.error("Exception occurred while Publishing data",e);
        }
    }
    public static void publishEvent(Object[] eventPayload, String streamId) throws InterruptedException {
            Event event = new Event(streamId, System.currentTimeMillis(), null, null, eventPayload);
        currentDataPublisher.publish(event);
        Thread.sleep(900000);
    }
}
```

Creating the data publisher with the Stream definition

```
public class DataPublisherUtil {
   private static Log log = LogFactory.getLog(DataPublisherUtil.class);
                                    public    static    String    filePath          =
"/home/shani/MSC/research/swiss/eventPublisherMsc/";
   static File securityFile = new File(filePath + "src/main/java/files/configs");
   public static void setTrustStoreParams() {
      String trustStore = securityFile.getAbsolutePath();
       System.setProperty("javax.net.ssl.trustStore", trustStore + "" + File.separator +
"client-truststore.jks");
      System.setProperty("javax.net.ssl.trustStorePassword", "wso2carbon");
   }
   public static Map<String, StreamDefinition> loadStreamDefinitions() {
      String directoryPath = filePath + "/src/main/java/files/streamDefinitions";
      File directory = new File(directoryPath);
         Map<String, StreamDefinition> streamDefinitions = new HashMap<String,
StreamDefinition>();
      if (!directory.exists()) {
                                log.error("Cannot load stream definitions from " +
directory.getAbsolutePath() + " directory not exist");
         return streamDefinitions;
      }
      if (!directory.isDirectory()) {
                                log.error("Cannot load stream definitions from " +
directory.getAbsolutePath() + " not a directory");
         return streamDefinitions;
      }
      File[] defFiles = directory.listFiles();
      if (defFiles != null) {
         for (final File fileEntry : defFiles) {
            if (!fileEntry.isDirectory()) {
               BufferedReader bufferedReader = null;
```

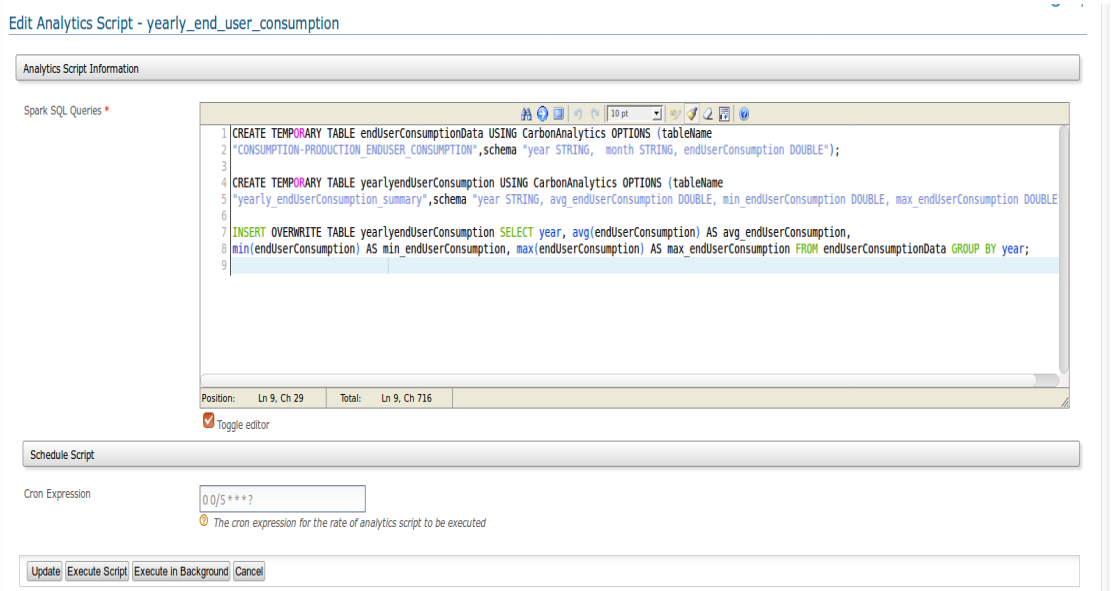
```
                StringBuilder stringBuilder = new StringBuilder();
                try {
                    bufferedReader = new BufferedReader(new FileReader(fileEntry));
                    String line;
                    while ((line = bufferedReader.readLine()) != null) {
                        stringBuilder.append(line).append("\n");
                    }
                                                    StreamDefinition streamDefinition =
EventDefinitionConverterUtils.convertFromJson(stringBuilder.toString().trim());
                                        streamDefinitions.put(streamDefinition.getStreamId(),
streamDefinition);
                } catch (FileNotFoundException e) {
                    log.error("Error in reading file " + fileEntry.getName(), e);
                } catch (IOException e) {
                    log.error("Error in reading file " + fileEntry.getName(), e);
} catch (MalformedStreamDefinitionException e) {
      log.error("Error in converting Stream definition " + e.getMessage(),
} finally {
bufferedReader.close();
   try {
      if (bufferedReader != null) {
      }
   } catch (IOException e) {
log.error("Error occurred when reading the file : " +e.getMessage(), e);
                    }
                }
            }
         }
      }
      return streamDefinitions;
   }
}
```

Persisting the Event



Appendix B Figure 1 persisting events

Creating the Spark SQL script



Appendix B Figure 2 Spark SQL script

Appendix C - Data preprocesser

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
public class DataCruncher {
   private static final String datapath = "home/shani/MSC/research/swiss/datasets";
   private static final List<Datapoint> data = new ArrayList<Datapoint>();
   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new FileReader(datapath +
"/data2015.csv"));
      boolean isHeaderSkipped = false;
      try {
         StringBuilder sb = new StringBuilder();
         String line = br.readLine();
         while (line != null) {
            if (isHeaderSkipped == false){
               isHeaderSkipped = true;
            } else {
               dailyAverage(line);
            }
            line = br.readLine();
         }
      } finally {
         br.close();
      }
      outputResults();
   }
```

```
    private static void outputResults(){
        data.sort(Comparator.comparing(datapoint -> datapoint.hashCode()));
        data.forEach(datePoint -> {
            System.out.println(datePoint.toString());
        });
    }

    private static void dailyAverage(String line){
        final String[] columns = line.split(",");
        int year = Integer.parseInt(columns[0]);
        int month = Integer.parseInt(columns[1]);
        int date = Integer.parseInt(columns[2]);;
        double consumption = Double.parseDouble(columns[3]);
        int season = Integer.parseInt(columns[4]);
        int holidayFlag = Integer.parseInt(columns[5]);



        Datapoint dataPoint = getDataPoint(year, month, date, season, holidayFlag);
        dataPoint.adjustAverage(consumption);
    }

    private static Datapoint getDataPoint(int year, int month, int date, int season, int
holidayFlag){
        Datapoint newDatapoint = new Datapoint(year, month, date, season,
holidayFlag);


        for (Datapoint d : data){
            if (d.equals(newDatapoint)){
                return d;


            }
        }
        data.add(newDatapoint);
```

```
        return newDatapoint;
    }

    static class Datapoint{
        public int month;
        int date;
        int year;
        double totalConsumption = 0.0;
        int dataPoints = 0;
        int season;
        int holidayFlag;
        static NumberFormat formatter = new DecimalFormat("#0.00");

        public Datapoint(int year, int month, int date, int season, int holidayFlag){
            this.year = year;
            this.month = month;
            this.date =  date;
            this.season = season;
            this.holidayFlag = holidayFlag;
        }

        public void adjustAverage(double newConsumptions){
            totalConsumption += newConsumptions;
            dataPoints++;
        }

        @Override
        public String toString(){
            return Integer.toString(year) + "," +           ((month < 10) ? "0" +
Integer.toString(month) : Integer.toString(month)) + "," +
                    ((date < 10) ? "0" + Integer.toString(date) : Integer.toString(date)) + ","
+
                    formatter.format(totalConsumption/dataPoints) + "," +
```

```
            Integer.toString(season) + "," +
            Integer.toString(holidayFlag);
    }
    @Override
    public boolean equals(Object obj){
      if (obj == null) {
        return false;
      }
      if (!Datapoint.class.isAssignableFrom(obj.getClass())) {
        return false;
      }
      final Datapoint other = (Datapoint) obj;
      if (this.year == other.year && this.month == other.month && this.date ==
other.date){
        return true;
      } else {
        return false;
      }
    }
    @Override
    public int hashCode(){
      String hashString = Integer.toString(year) +
            ((month < 10) ? "0" + Integer.toString(month) : Integer.toString(month))
+
            ((date < 10) ? "0" + Integer.toString(date) : Integer.toString(date)) ;
      return  Integer.parseInt(hashString);
    }
  }
}
```

Appendix D – Source for building the model to predict energy consumption – with one factor

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pylab as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

import seaborn as sns
sns.set_style("darkgrid")
sns.set_context("poster")

def rolling_univariate_window(time_series, window_size):
    shape = (time_series.shape[0] - window_size + 1, window_size)
    strides = time_series.strides + (time_series.strides[-1],)
    return np.lib.stride_tricks.as_strided(time_series, shape=shape, strides=strides)

def build_rolling_window_dataset(time_series, window_size):
    last_element = time_series[-1]
    time_series = time_series[:-1]
    X_train = rolling_univariate_window(time_series, window_size)
    y_train = np.array([X_train[i, window_size-1] for i in range(1, X_train.shape[0])])

    return X_train, np.hstack((y_train, last_element))

def train_test_split(no_of_training_instances, X_all, y_all):
    X_train = X_all[0:no_of_training_instances, :]
    X_test = X_all[no_of_training_instances:, :]
    y_train = y_all[0:no_of_training_instances]
    y_test = y_all[no_of_training_instances:]
```

```
    return X_train, X_test, y_train, y_test


def print_graph(X_all, X_test, y_all, y_test, y_pred):
    training_size = X_all.shape[0] - X_test.shape[0]
    x_full_limit = np.linspace(1, X_all.shape[0], X_all.shape[0])
    y_pred_limit = np.linspace(training_size+1, training_size + 1 + X_test.shape[0],
X_test.shape[0])
    plt.plot(x_full_limit, y_all, label='actual', color='b', linewidth=1)
    plt.plot(y_pred_limit, y_pred, '--', color='r', linewidth=2, label='prediction')
    plt.legend(loc=0)
    plt.show()


data_set = pd.read_csv('/home/sajith/shani-project/datasets/consumptions.csv')
data_set = data_set.values.flatten()


window_size = 16
training_set_size = 30


X_all, y_all = build_rolling_window_dataset(data_set, window_size)
X_train, X_test, y_train, y_test = train_test_split(training_set_size, X_all, y_all)


lr = LinearRegression(normalize=True)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print "mean_absolute_error: %f" %(mean_absolute_error(y_test, y_pred))
print_graph(X_all, X_test, y_all, y_test, y_pred)
```

Appendix E – Source code for building the prediction model with multiple regression

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pylab as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

import seaborn as sns
sns.set_style("darkgrid")
sns.set_context("poster")



def train_test_split(no_of_training_instances, X_all, y_all):
    X_train = X_all[0:no_of_training_instances, :]
    X_test = X_all[no_of_training_instances:, :]
    y_train = y_all[0:no_of_training_instances]
    y_test = y_all[no_of_training_instances:]

    return X_train, X_test, y_train, y_test

def print_graph(X_all, X_test, y_all, y_test, y_pred):
    training_size = X_all.shape[0] - X_test.shape[0]
    x_full_limit = np.linspace(1, X_all.shape[0], X_all.shape[0])
    y_pred_limit = np.linspace(training_size+1, training_size + 1 + X_test.shape[0],
X_test.shape[0])
    plt.plot(x_full_limit, y_all, label='actual', color='b', linewidth=1)
    plt.plot(y_pred_limit, y_pred, '--', color='r', linewidth=2, label='prediction')
    plt.legend(loc=0)
    plt.show()
```

```
data_set = pd.read_csv('/home/sajith/shani-project/datasets/summarized.csv')
training_set_size = 30

y_all =
data_set['Total_Energy_Consumed_by_end_users_Swiss_controlblock'].values
X_all = data_set[['year', 'date', 'month', 'season', 'publicHoliday']].values
X_train, X_test, y_train, y_test = train_test_split(training_set_size, X_all, y_all)

lr = LinearRegression(normalize=True)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print "mean_absolute_error: %f" %(mean_absolute_error(y_test, y_pred))
print_graph(X_all, X_test, y_all, y_test, y_pred)
```

Illustration Index