Assignment 3: Machine Learning as a Service in Microsoft Azure ML

Under guidance of:

Prof. Sri Krishnamurthy
Ashwin Dinoriya

Team 1

Amulya Aankul
Nand Govind Modani
Saksham Agrawal

Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal

Step 1: Data Preparation in R

The first step is preparation of data. We have followed the same procedures as previous assignments for pre-processing and wrangling of data in R. To summarize:

- API calls for weather data and holiday data
- Pre-processing of data
- Imputation of missing values
- Removal of outliers
- The clean data from R is uploaded to Microsoft Azure ML Studio datasets

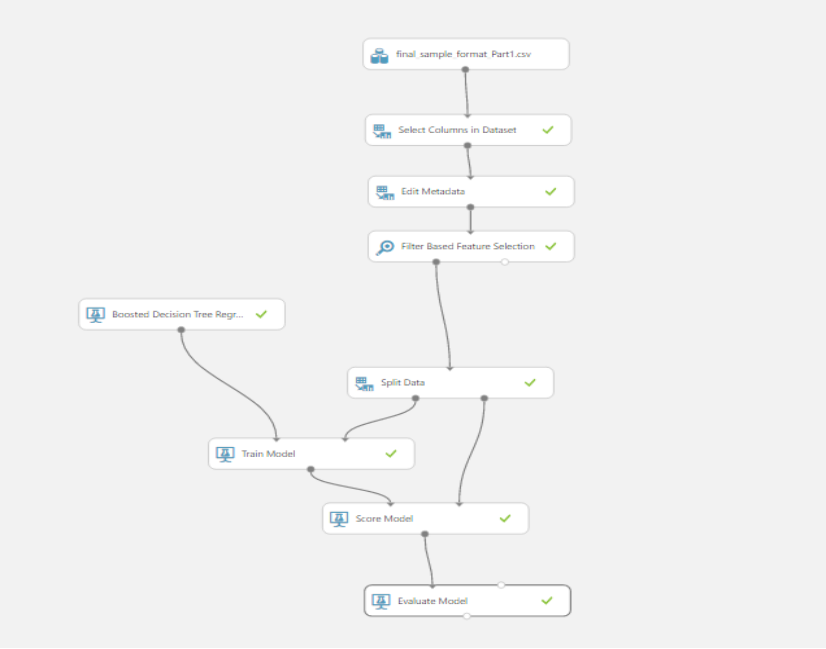
Step 2: Microsoft Azure ML Studio – Building model

The following are the steps to create an experiment in Microsoft Azure ML Studio:

- Select the required column using **Column Selector**
- Specify the data types and categorical variables using **Edit Metadata**
- Selection of features using **Filter based Feature Selection**
- Splitting data into Training and Testing using **Split**
- Training of model using **Train Model**
- Scoring and Evaluation of model using **Score Model** and **Evaluate Model**

Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal



Step 3: Building customized model in R

The following are the steps to create a model in R in Microsoft Azure ML Studio. We noticed that KNN and Hierarchal Clustering are not pre-present in Microsoft Azure. To implement that, we had to use **Create R model** feature in Microsoft Azure ML. It has a Trainer script and Scorer script:

Create R Model

Trainer R script

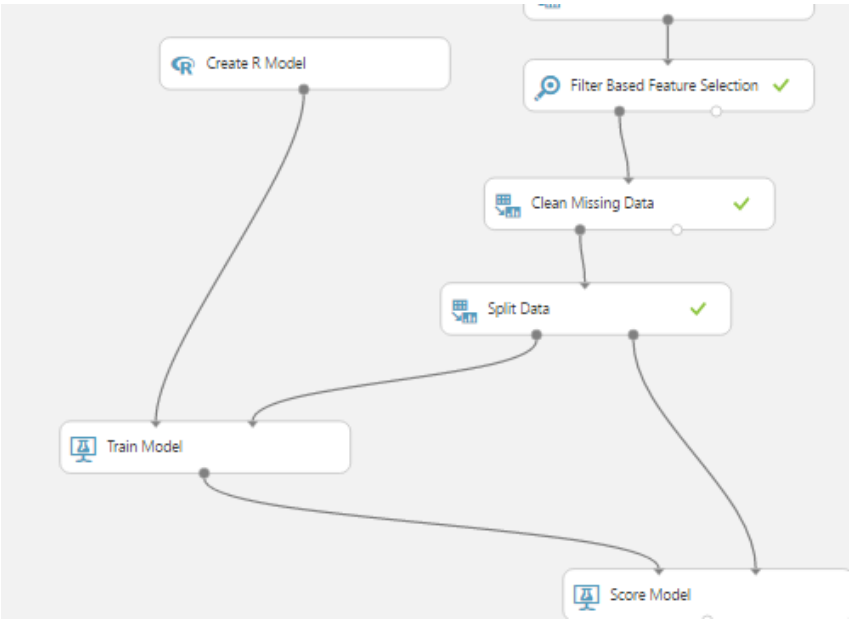
```
# Input: dataset
# Output: model

# The code below is an example which can be replaced with your own code.
# See the help page of "Create R Model" module for the list of predefined functions and constants.

library(kknn)
model <- train.kknn(Norm_Consumption ~ .,dataset, kmax = 15, kernel = c("optimal"), distance = 2)
```

Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal

Scorer R script

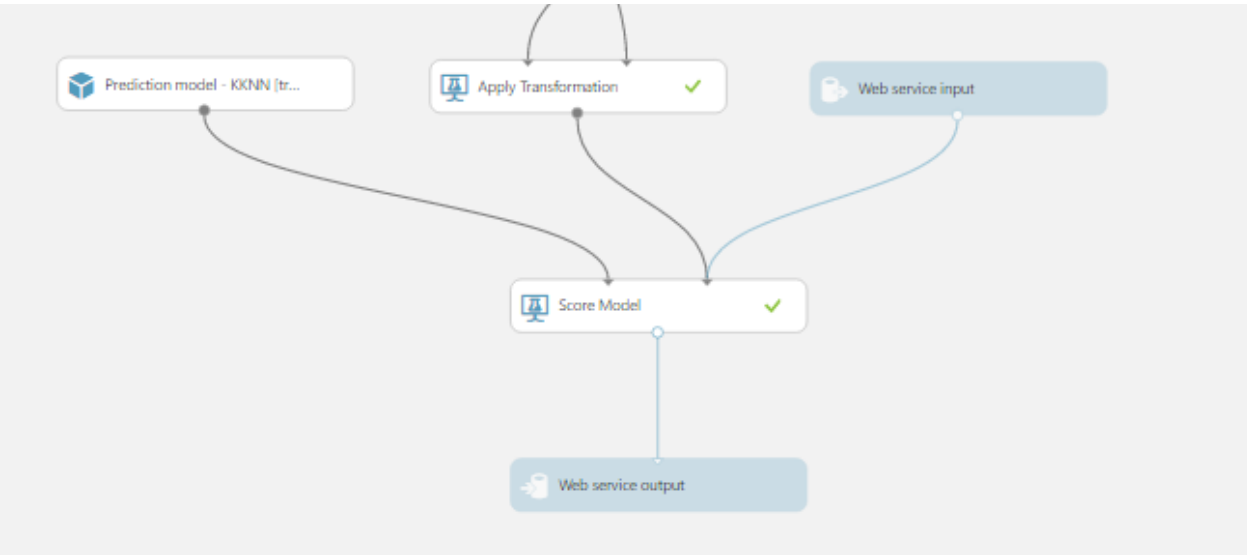
```r
# Input: model, dataset
# Output: scores

# The code below is an example which can be replaced with your own code.
# See the help page of "Create R Model" module for the list of predefined functions and constants.

library(kknn)
scores <- data.frame(predict(model, subset(dataset, select = -c(Norm_Consumption))))

```

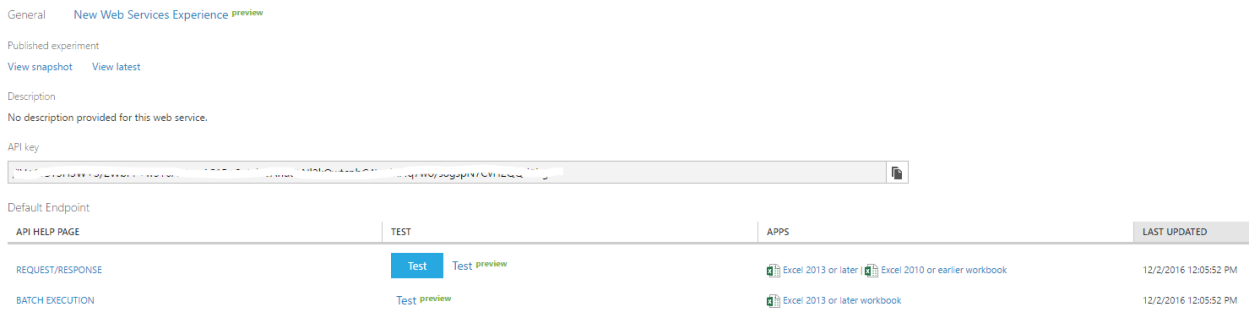


Step 4: Set up and Deploy Web Service

Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal





Step 5: Building a website

We used HTML5, bootstrap and in the backend we have used Flask framework in Python 3+.

Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal

Request-Response Batch

C# Python Python 3+ R

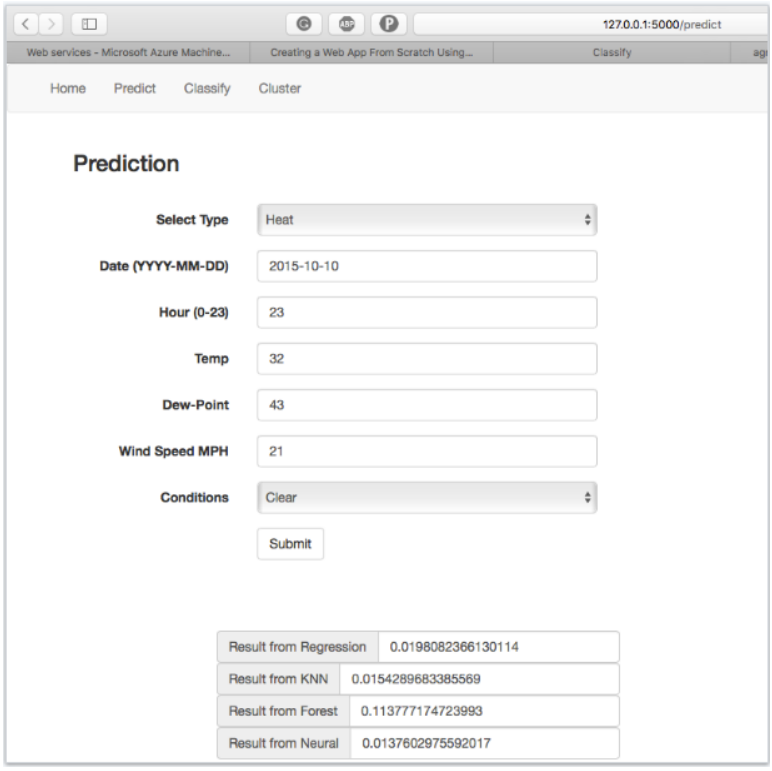
```
import urllib.request
import json

data = {
        "Inputs": {
                "input1":
                [
                    {
                            'Norm_Consumption': "1",
                            'Dew_PointF': "1",
                            'TemperatureF': "1",
                            'month': "1",
                            'Conditions': "Clear",
                            'Weekday': "1",
                            'Base_hour_Flag': "false",
                            'Wind_SpeedMPH': "1",
                            'type': "Dist_Heating",
                    }
                ],
        },
    "GlobalParameters":  {
    }
}
```

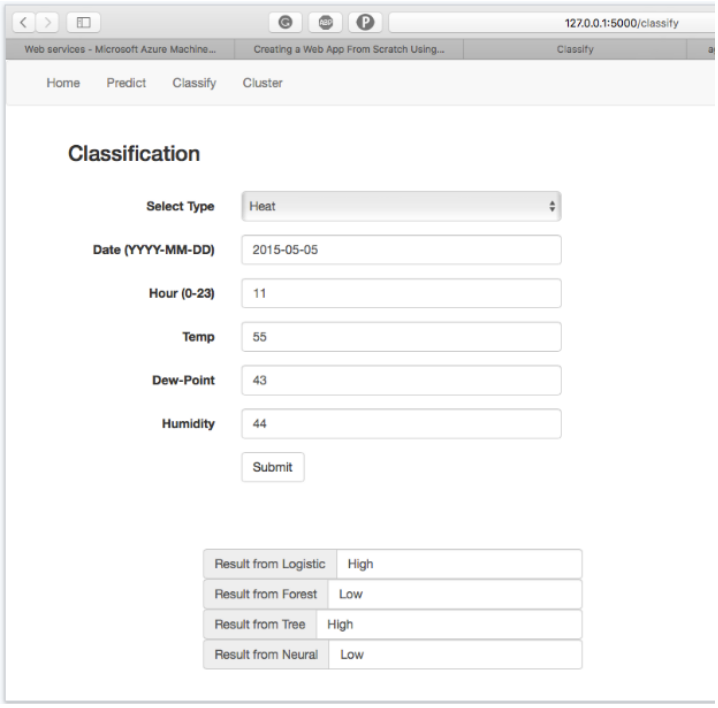
controller.py

```
##################### Clustering Starts ######################
@app.route('/cluster')
def runClust():
    return render_template('/cluster.html')


@app.route('/cluster', methods=['POST'])
def get_data_clust():
    #print("inside POST")
    data = request.form
    area = data['area']
    latitude = data['latitude']
    longitude = data['longitude']
    electric = data['electric']
    heat = data['heat']


    resultKMean = processClust(algo="kMean",area=area,latitude=latitude,longitude=longitude,electric=electric,heat=heat)

    resultKMean = json.loads(resultKMean)

    return render_template('/cluster.html',area=area,latitude=latitude,longitude=longitude,electric=electric,heat=heat,labelKMean=resul


### function for calling K-Means Clustering Azure ML API
def cluster_kMean(body):

    url = 'https://ussouthcentral.services.azureml.net/workspaces/4cf184d6c98d43debeddaef6ffe92725/services/6988ed21dd3348458d85615ae6f
    api_key = '0T02SXPBoZ38LivdPpMWTxnNeHRZx9AMKibe9j36dQWIRgnwDRIwJSdM3cRo1NM/hlK1ZFUEJ9ZEF+vwWMxtIg==' # Replace this with the API ke
    headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key)}


    req = urllib.request.Request(url, body, headers)
    #print("request ready")

    try:
        response = urllib.request.urlopen(req)
        result = response.read().decode("utf-8")
        #print("Response ready")
        return result
    except urllib.error.HTTPError as error:
        print("The request failed with status code: " + str(error.code))

        # Print the headers - they include the requert ID and the timestamp, which are useful for debugging the failure
        print(error.info())
        print(json.loads(error.read().decode("utf8", 'ignore')))
        return None
```

Team Members:

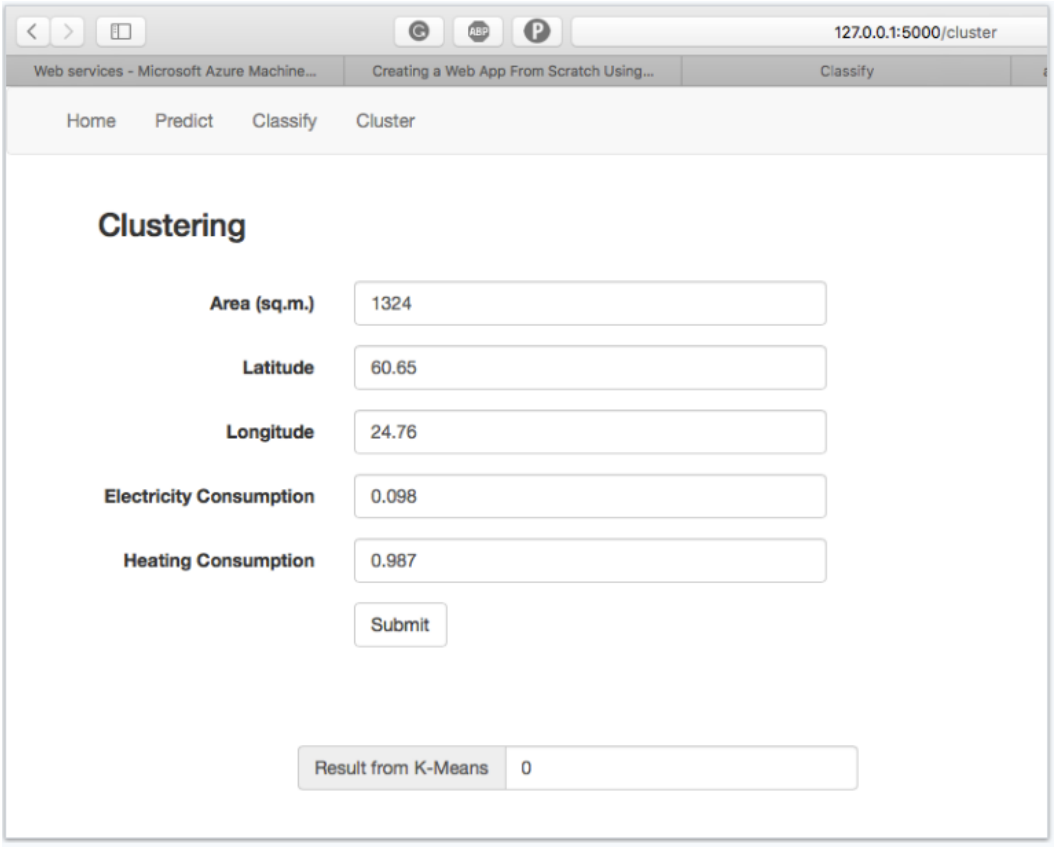
Amulya Aankul
Nand Govind Modani
Saksham Agrawal

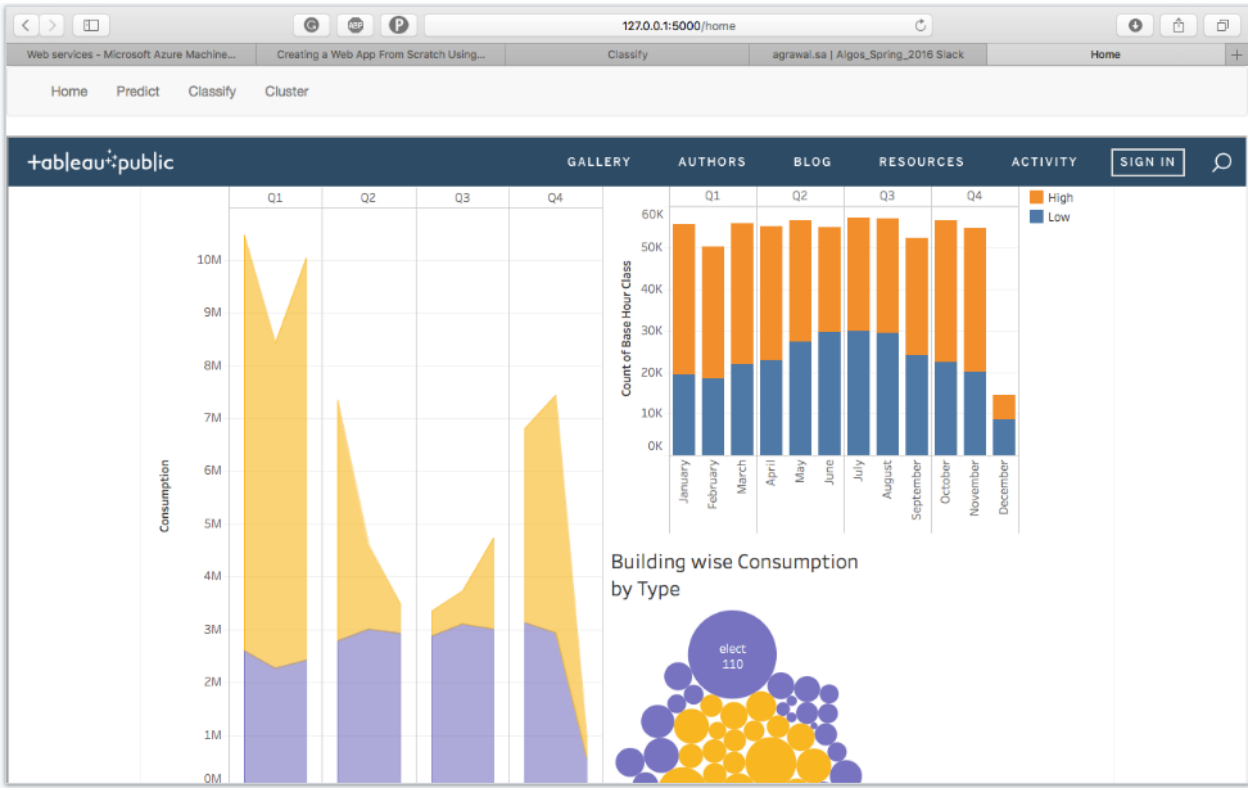




Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal

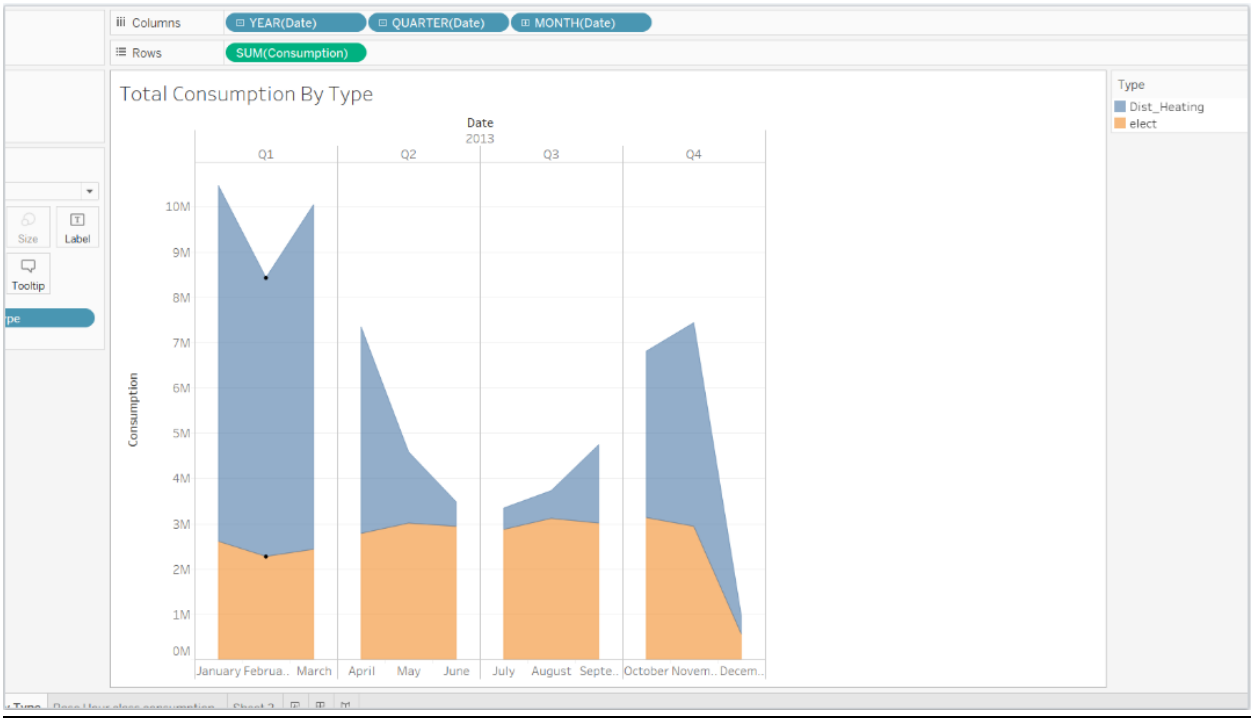




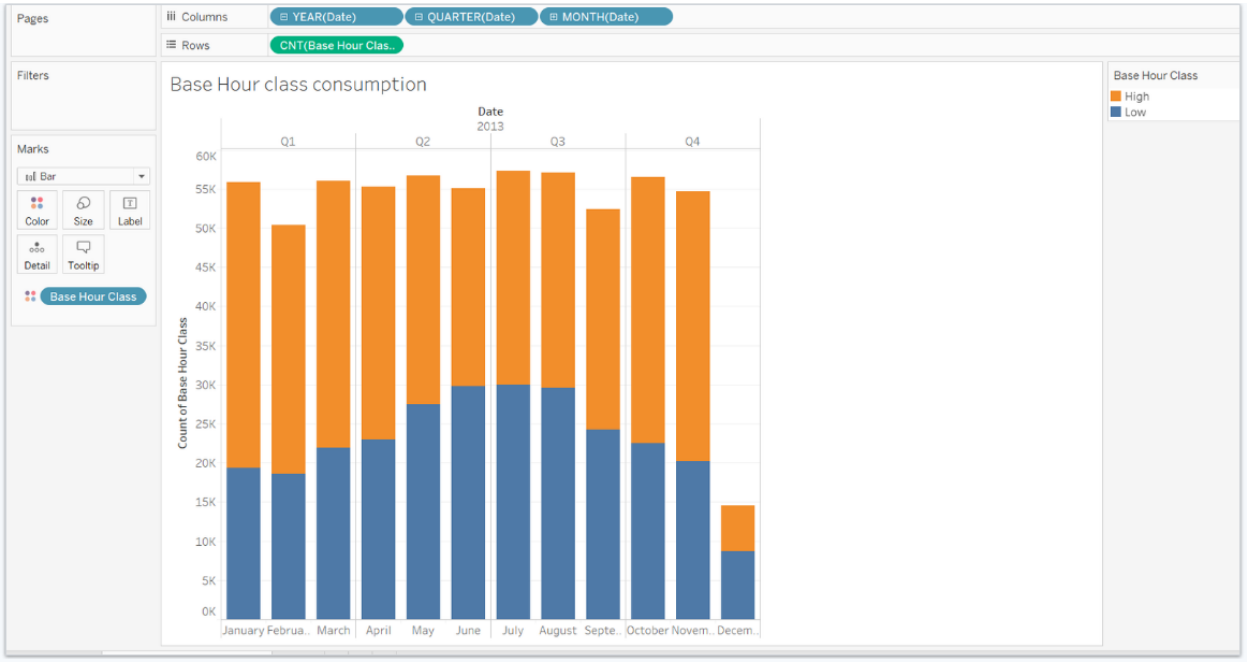
Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal

Step 6: Data Visualization



- The Plot shows the total Energy consumption by type for entire year. This gives the consumption distribution for year.



Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal

- The base hour class is used as a classified label for classification. The above plot shows the exploratory data analysis for count of base hour class values per month.

Team Members:

Amulya Aankul
Nand Govind Modani
Saksham Agrawal