SMART BIKE SYSTEM

Team 10: Suhas Pirankar & Zalak Shah

INFO7390 Advances Data Science/Architecture

Table of Contents

INTRODUCTION

Smart Bike systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

From this dataset, we will attempt to answer the following questions:
1) Predict bike rental demand for any given time.
2) Identify which attributes of a smart bike dataset can affect these demands of bikes.
3) Does the usage of bikes by customer has strong dependency on wind speed.
4) Does the available number of bikes ever go down to zero and customers are not able to ride a bike because of shortage of bikes.
5) How does the data of casual customer differ than the registered customers

To answer these questions, we have built model focusing on various factors that contribute to give us the correct count of bikes that are needed on a given time. Attributes that are useful in predicting are date, time, season, holidays, working day, weather, temperature, feels like temperature, humidity, wind speed and so on. This model will identify the key factors that will help us in predicting the correct number of casual and registered users.

Azure Credentials:

User Name: - i7390Summer2016team10@outlook.com
Password: Login10. *Note: There is a "." After 10 in password*

Azure Experiment Name: Smart Bike Registered & Smart Bike Casual
Website URL: smartbike.azurewebsites.net
GitHub URL: https://github.com/zalak13/ADS_Team10/FinalProject

DATA UNDERSTANDING

We have selected the dataset for Smart bike company based in Washington. Below are the description of each and every column in our data set. Below information we have categorized into two groups. Independent variables are date, time, season, holidays, working day, weather, temperature, feels like temperature, humidity, wind speed. Dependent variables are registered, casual and count. We have added the description of each column below

The dataset shows hourly rental data for two years (2011 and 2012). The training data set is for the first 19 days of each month. The test dataset is from 20th day to month's end. We are required to predict the total count of bikes rented during each hour covered by the test set.
In the training data set, they have separately given bike demand by registered, casual users and sum of both is given as count.
Training data set has 12 variables (see below) and Test has 9 (excluding registered, casual and count).

The Smart Bike dataset have following columns in a csv file.

Independent Variables:

Date time:	date and hour in "mm/dd/yyyy hh:mm" format
season:	Four categories-> 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday:	whether the day is a holiday or not (1/0)
Working Day:	whether the day is neither a weekend nor holiday (1/0)

Four Categories of Weather

- Clear, Few clouds, Partly cloudy, Partly cloudy
- Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
- Light Snow and Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Temperature:	Hourly Temperature in Celsius
A-temp:	"Feels Like" temperature in Celsius
Humidity:	Relative humidity
Wind speed:	Speed of wind

Dependent Variables:

Registered:	Number of registered user
Casual:	Number of non-registered user
Count:	Number of total rentals (registered + casual)

DATA CLEANING:

After uploading data in azure and summarizing it. We found a lot of missing data in the wind speed column. As wind speed is very important attribute in our prediction we could not simple replace it with zero's or mean of the column or with NA. We decided to using the existing wind speed values and use those values to train a model and predict the remaining values of the wind speed. Here we used random forest regression algorithm to predict the values. Below is the R code to fill all the zero values in the wind speed attribute.

```
#Use random forest and predict windspeed
library(randomForest)
set.seed(415)
fit <- randomForest(windspeed ~ season+weather +humidity +month+temp+ year+atemp,
data=wind_1,importance=TRUE, ntree=250)
pred=predict(fit,wind_0)
wind_0$windspeed=pred
```

After this step we visualized the wind speed column in azure and found that there we no more zero values. This helped us increase the overall accuracy of the results

DATA EXPLORATION:

Our aim was to predict the count of bike that will be needed at any given time of a day.

Below are the steps in data cleaning that we performed.
1)Convert discrete variables into factor (season, weather, holiday, working day)

```
data$season=as.factor(data$season)
data$weather=as.factor(data$weather)
data$holiday=as.factor(data$holiday)
data$workingday=as.factor(data$workingday)
```

2)Hourly trend: We don't have the variable 'hour' with us. So we can extract it using the date time column.

```
data$hour=substr(data$datetime,12,13)
data$hour=as.factor(data$hour)
train$hour=as.integer(train$hour) # convert hour to integer
test$hour=as.integer(test$hour) # modifying in both train and test dataset
```

1)Daily Trend: Like Hour, we will generate a variable for day from date time variable

```
date=substr(data$datetime,1,10)
days<-weekdays(as.Date(date))
data$day=days
```

2)Time: Let's extract year of each observation from the date time column and see the trend of bike demand over year.

```
data$year=substr(data$datetime,1,4)
data$year=as.factor(data$year)
```

DATA TRANSFORMATION AND REDUCTION:

Feature Engineering

In addition to existing independent variables, we created new variables to improve the prediction power of model. We generated new variables like hour, month, day and year. After visualizing data and looking at the nodes we can create different hour bucket for registered users.

```
data=rbind(train,test)
data$dp_reg=0
data$dp_reg[data$hour<8]=1
data$dp_reg[data$hour>=22]=2
data$dp_reg[data$hour>9 & data$hour<18]=3
data$dp_reg[data$hour==8]=4
data$dp_reg[data$hour==9]=5
data$dp_reg[data$hour==20 | data$hour==21]=6
data$dp_reg[data$hour==19 | data$hour==18]=7
```

Similarly, we created day_part for casual users also (dp_cas).

```
data$dp_cas=0
data$dp_cas[data$hour<=8]=1
data$dp_cas[data$hour==9]=2
data$dp_cas[data$hour>=10 & data$hour<=19]=3
data$dp_cas[data$hour>19]=4
```

Temp Groups**:** We have created bins for temperature for both registered and casuals users. Variables created are (temp_reg and temp_cas).

Year Groups: As bike demand will increase over time. We have created 8 bins (quarterly) for two years. Jan-Mar 2011 as 1 …..Oct-Dec2012 as 8.

```
data$year_part[data$year=='2011']=1
data$year_part[data$year=='2011' & data$month>3]=2
data$year_part[data$year=='2011' & data$month>6]=3
data$year_part[data$year=='2011' & data$month>9]=4
data$year_part[data$year=='2012']=5
```

```
data$year_part[data$year=='2012' & data$month>3]=6
data$year_part[data$year=='2012' & data$month>6]=7
data$year_part[data$year=='2012' & data$month>9]=8
table(data$year_part)
```

Day _Type: Created a variable having categories like “weekday”, “weekend” and “holiday”.

```
data$day_type=""
data$day_type[data$holiday==0 & data$workingday==0]="weekend"
data$day_type[data$holiday==1]="holiday"
data$day_type[data$holiday==0 & data$workingday==1]="working day"
```

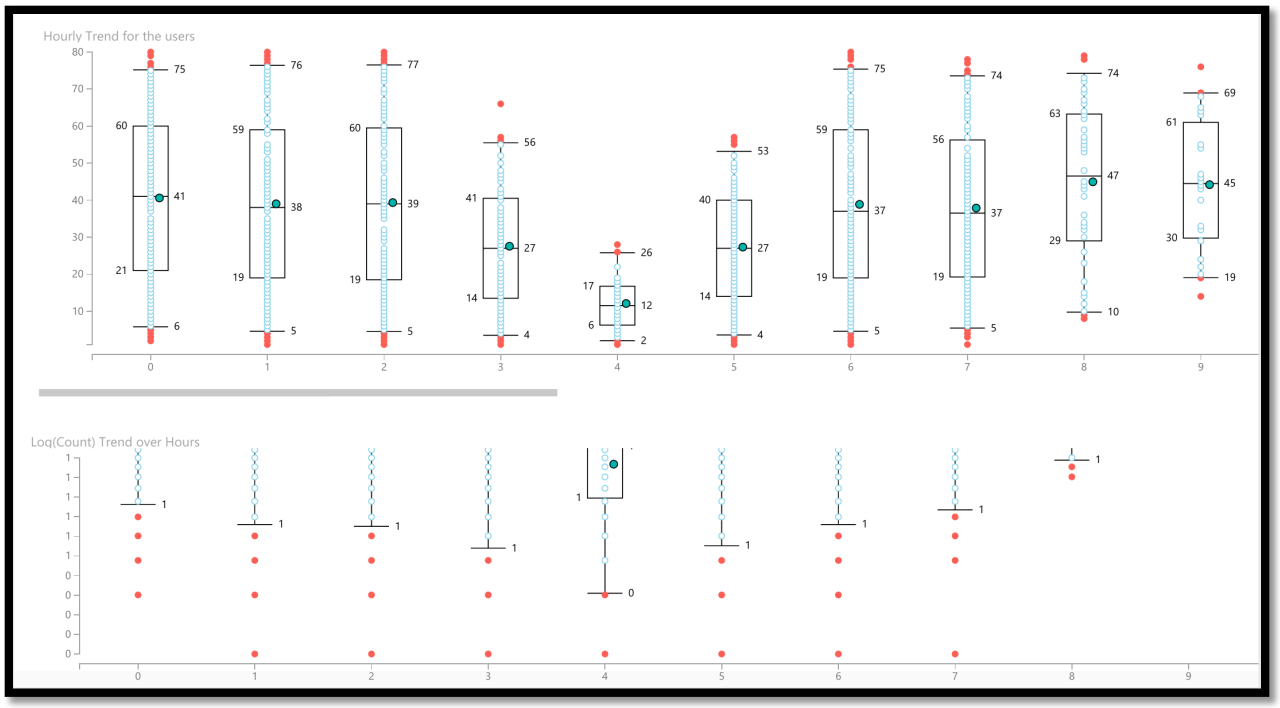
Weekend: Created a separate variable for weekend (0/1)

```
data$weekend=0
data$weekend[data$day=="Sunday" | data$day=="Saturday" ]=1
```

DATA VISUALIZATION:

Inferences from above data visualization:

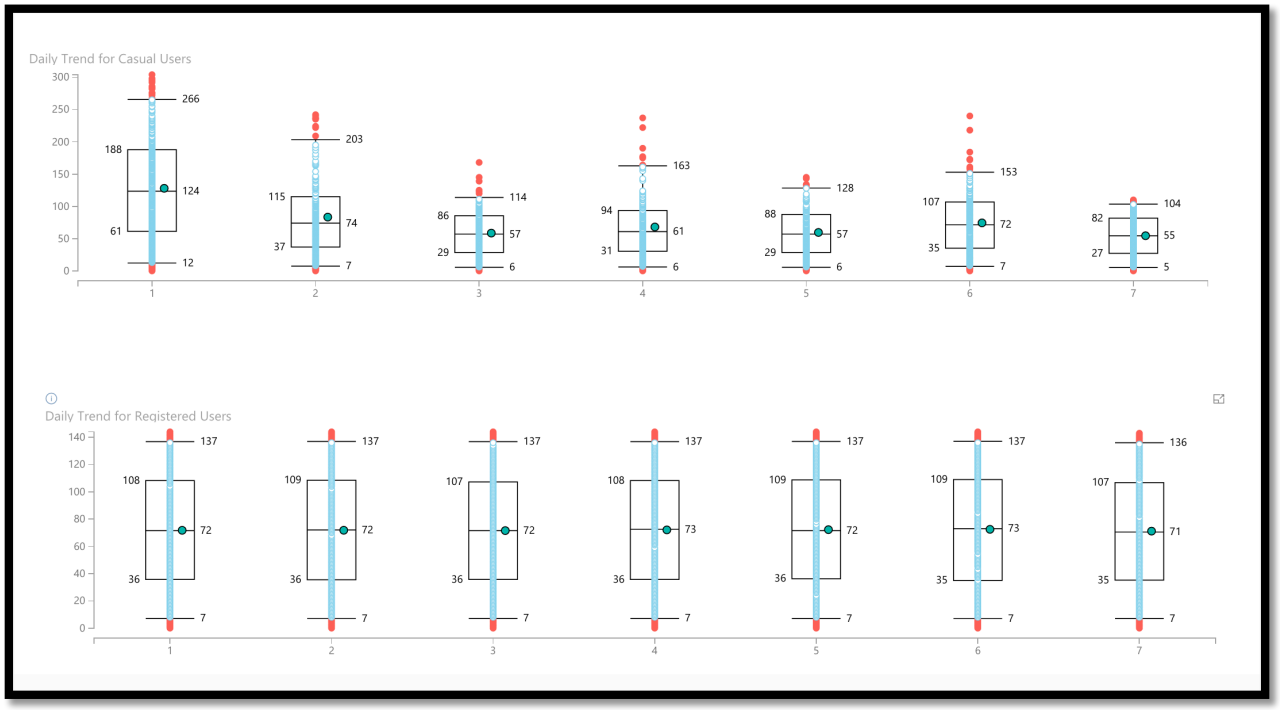
- ✓ Season has four categories of almost equal distribution
- ✓ Weather 1 has higher contribution i.e. mostly clear weather.
- ✓ As expected, mostly working days and variable holiday is also showing a similar inference.
- ✓ Variables temperature, Day of Week and wind speed looks naturally distributed.



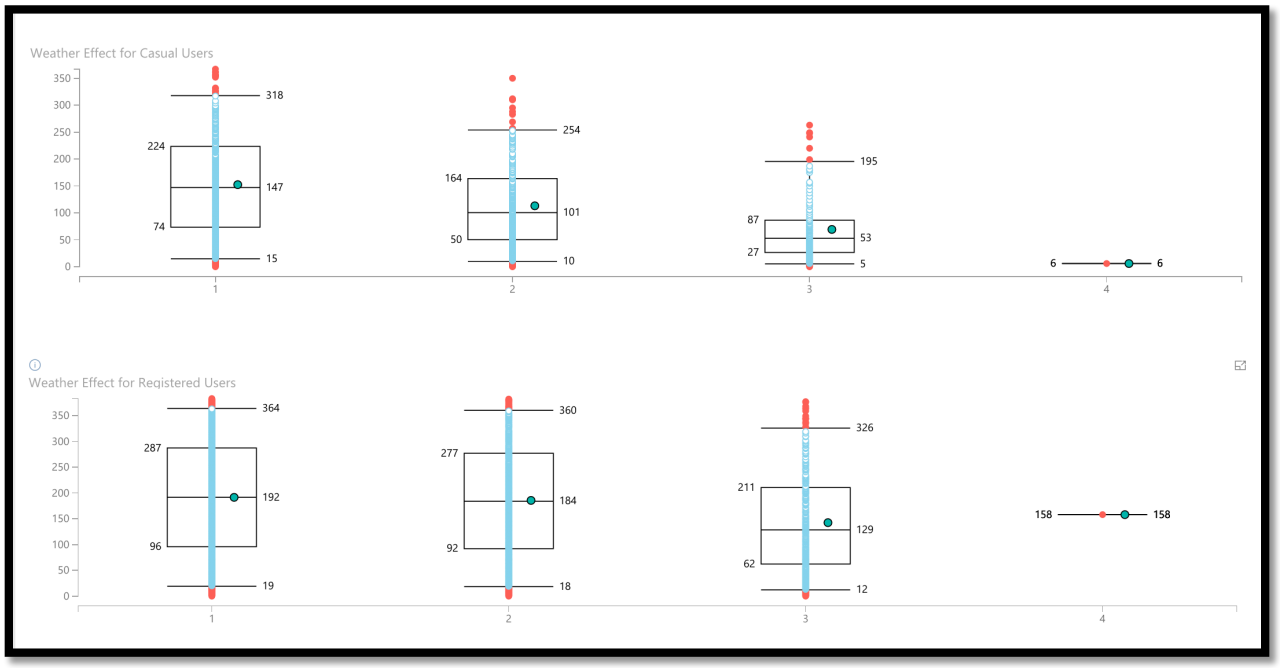
We can see the see the trend of bike demand over hours and analyzed the distribution of total bike demand. We segregated the bike demand in three categories:

- ✓ High: 7-9 and 17-19 hours
- ✓ Average: 10-16 hours
- ✓ Low: 0-6 and 20-24 hours

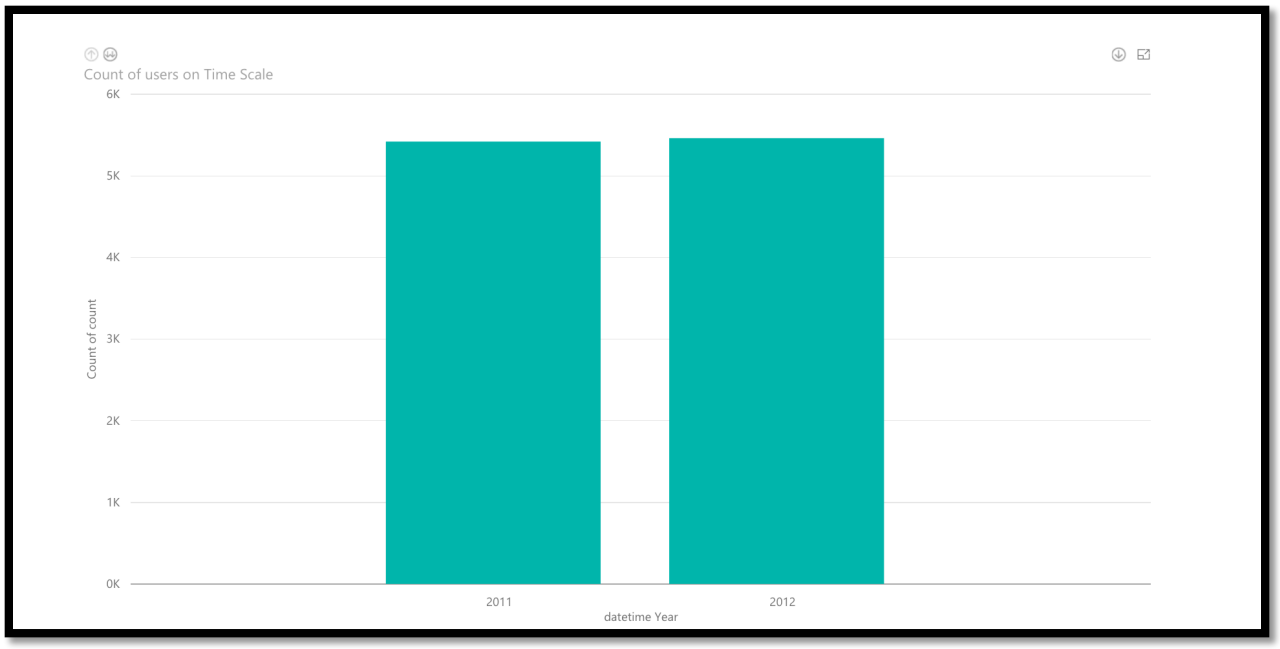
There are a lot of outliers while plotting the count of registered and casual users. These values are not generated due to error. They might be a result of groups of people taking up cycling (who are not registered). To treat such outliers, we will use logarithm transformation.



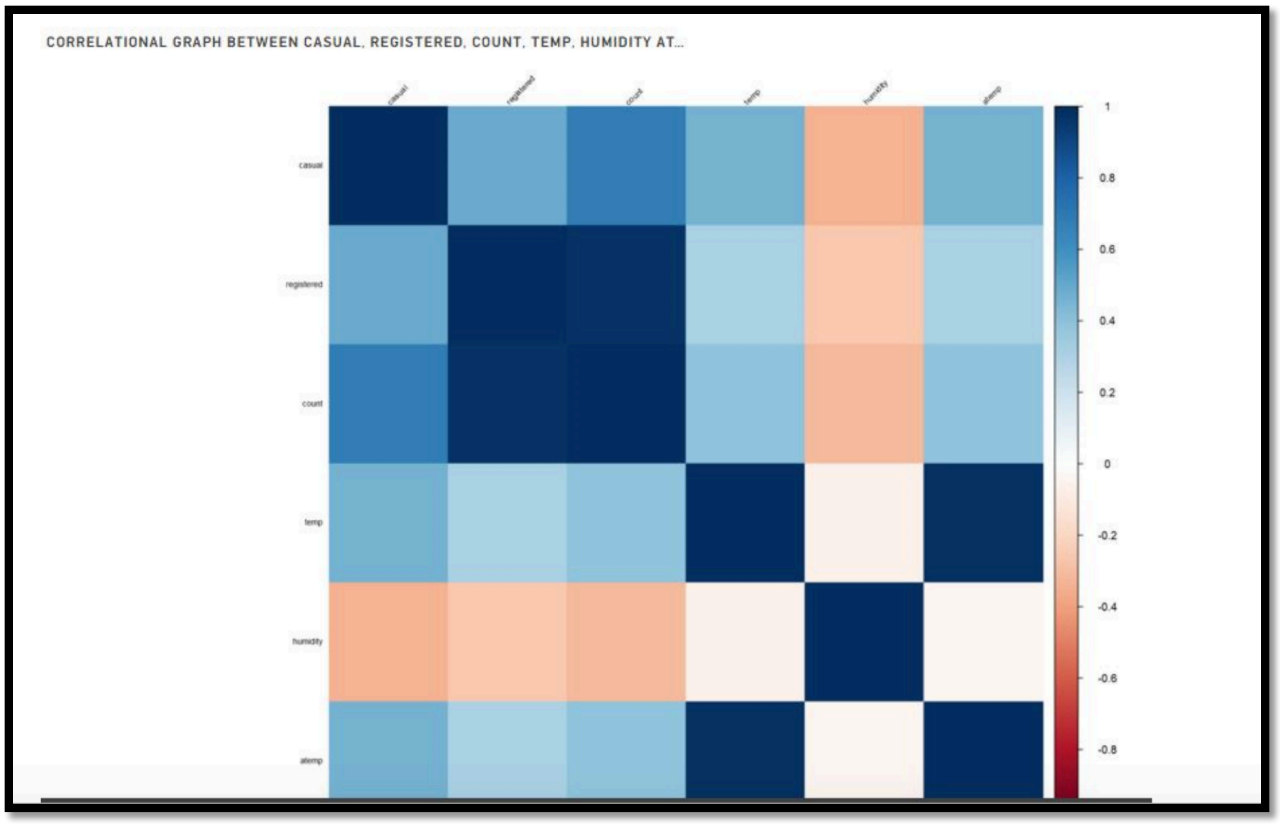
Above screen shot shows us the distribution of registered and casual users separately.
We can see that registered users have similar trend as count. Whereas, casual users have different trend. Thus, we can say that 'hour' is significant variable and our hypothesis is 'true'.



Weather: We don't have the 'rain' variable with us but have 'weather' which is sufficient to test our hypothesis. As per variable description, weather 3 represents light rain and weather 4 represents heavy rain. We can infer that during light rain the number of people using bikes are very less and during heavy rains the number of users are very small in number.



Time: Above image shows us the trend of bike demand over year. 2012 has a slightly more number of users.



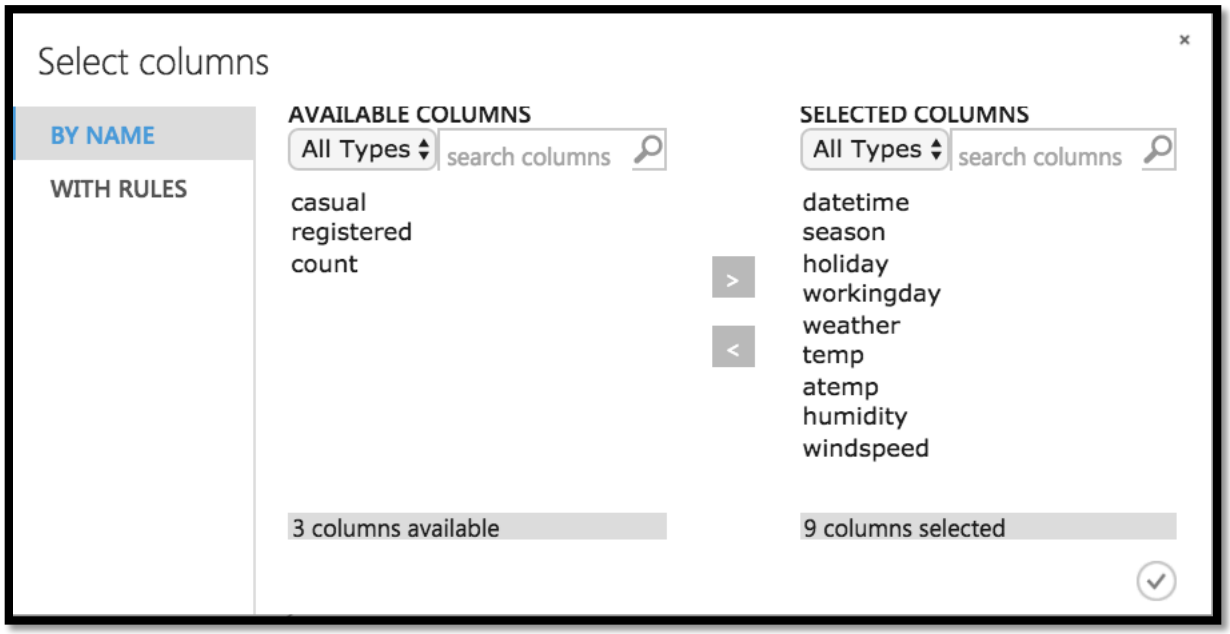
Temperature, Wind speed and Humidity are continuous variables so we can look at the correlation factor to validate hypothesis. Above image shows how each input parameter are correlated to each other. We can infer that count, casual, registered attribute are least related to humidity and temperature is related to feels like temperature so it is shown in dark blue.

DATA MODELING:

After visualization and understanding the data better we build the model in Azure. We have used Boosted Decision Tree Regression, Decision Forest Regression, Neural Network Regression and Bayesian Linear Regression model and tried to evaluate on the basis of RMSE and AME to determine best model for our dataset.

Before building model we pre-processed the data. We create new columns in the dataset using the existing columns. We used feature engineering and built columns that will help is give more accurate predictions. We have filled all the values for wind speed as there were many zeros in wind speed data. We used random forest in execute R script in azure to do predict the values for wind speed. We used the already existing non-zero wind speed values to train our random forest algorithm and predicted wind speed using that model. All this operation was performed before in the Execute R script before the model was ran. Since azure does not allow us to create one predictive experiment with two train model we created separate model for predicting count of casual and registered user. The skeleton of the model remains the same just the input to the models differ in terms of casual for casual users.

In Select Column we selected the variable that were important in building a model keeping in mind about user's perspective as if inputs that we will take in the front end is relevant to user.
The following are the columns that we selected in building our model.



After pre-processing we visualized the data in Azure whether the dataset is clean or not using summarize data which gives the result of each column and its detail. Below is screen shot of how our data looks when we visualize summary data:

Smart Bike Registered › Summarize Data › Results dataset

rows 17 columns 23

view as

Feature	Count	Unique Value Count	Missing Value Count	Min	Max	Mean
season	10886	4	0			
holiday	10886	2	0			
workingday	10886	2	0			
weather	10886	4	0			
atemp	10886	60	0	0.76	45.455	23.655084
humidity	10886	89	0	0	100	61.88646
windspeed	10886	1006	0	6.0032	56.9969	14.209379
hour	10886	24	0			
day	10886	7	0			
year	10886	2	0			
day_part	10886	4	0			
dp_reg	10886	7	0	1	7	3.009462
temp_reg	10886	4	0			
year_part	10886	2	0	1	5	3.007716
day_type	10886	7	0			
weekend	10886	2	0	0	1	0.290557
logreg	10886	731	0	0	6.787845	4.395774

After the data is pre-processed we split the data in to 70% Train Data and 30% Test Data and ran all model and evaluated the models in the end. The following are results of score models:

rows 2177 columns 19

season	holiday	workingday	weather	atemp	humidity	windspeed	hour	day	year	day_part	dp_reg	temp_reg	year_part	day_type	weekend	logreg	Scored Label Mean
4	0	1	1	20.455	76	9.884467	21	Tuesday	2011	4	6	2	1	6	0	5.135798	5.239453
1	0	0	2	21.97	88	6.0032	10	Saturday	2012	3	3	2	5	3	1	5.384495	5.399723
4	0	1	1	15.15	66	27.9993	8	Tuesday	2012	1	4	2	5	4	0	5.921578	5.949354
1	0	0	3	15.15	100	19.9995	18	Sunday	2011	3	7	2	1	7	1	3.178054	3.612122
3	0	1	1	26.515	64	7.0015	9	Wednesday	2012	2	5	2	5	5	0	6.590301	6.56068
1	0	0	3	21.97	100	15.0013	16	Sunday	2011	3	3	2	1	3	1	1.609438	3.421971
4	0	1	1	17.425	93	6.0032	6	Monday	2012	1	1	2	5	1	0	3.663562	3.524304

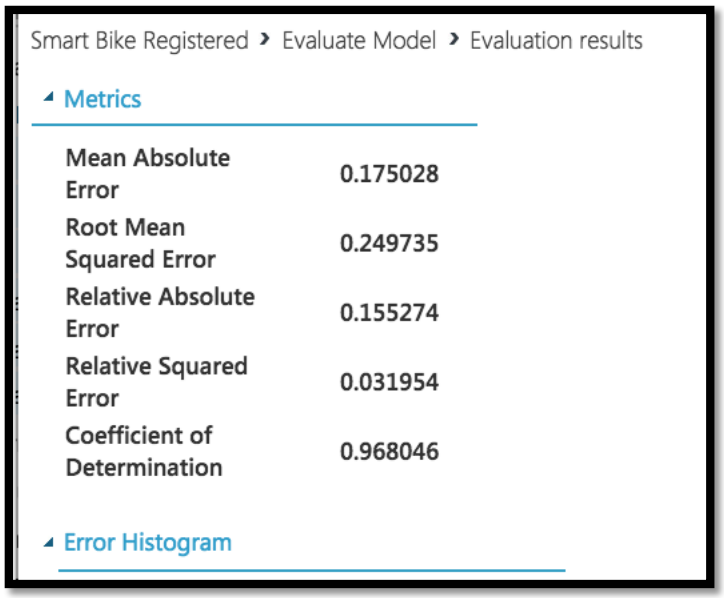
Score Model Results

EVALUATE MODEL:

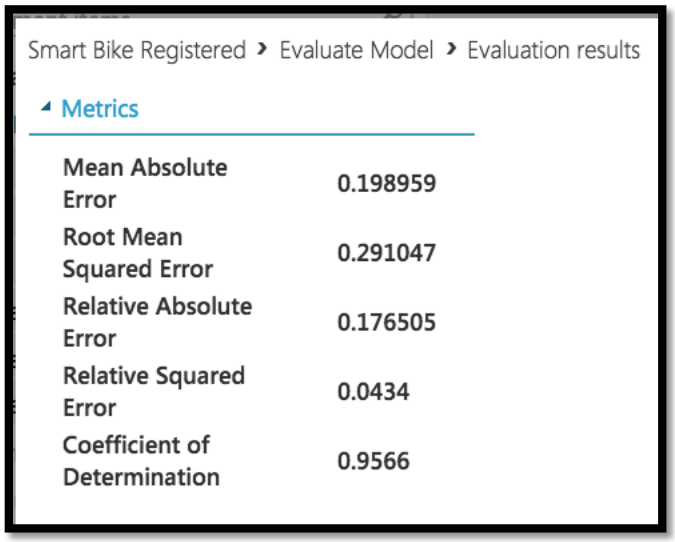
Results of Decision Forest Regression.

Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
0.005127	0.01407	0.004111	0.000089	0.999911

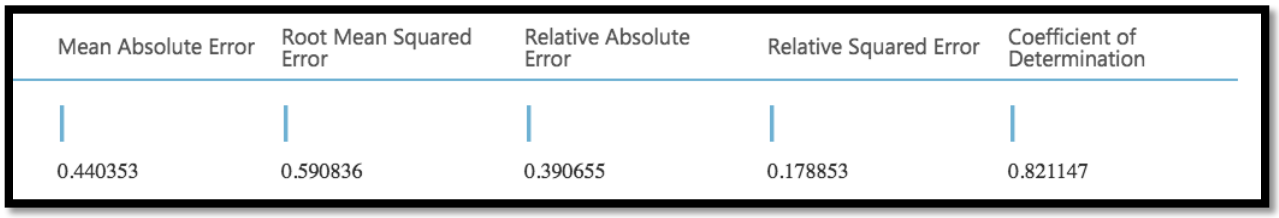
Result of Boosted Decision Tree:



Result of Neural Network Regression:



Result of Bayesian Regression:



Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
0.440353	0.590836	0.390655	0.178853	0.821147

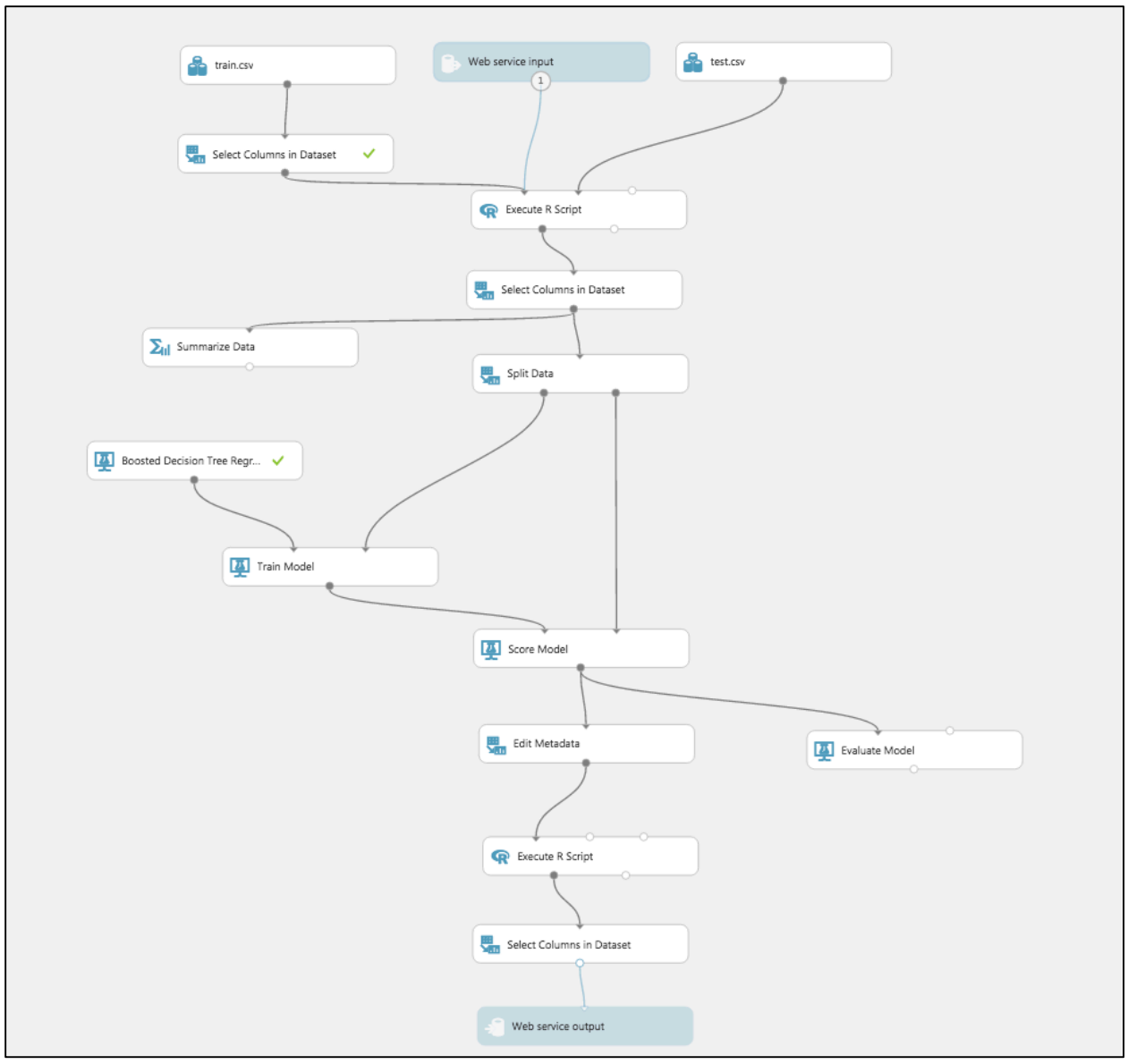
RegressionAlgorithm	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
Decision Forest	0.0051	0.01407	0.00411	0.00089	0.98
Boosted Decision Tree	0.17501	0.2497	0.1552	0.03194	0.968
Neural Network	0.1989	0.2901	0.1765	0.0434	0.9566
Bayesian	0.4403	0.5908	0.3906	0.1788	0.8211

We evaluated our model on the basis of RMSE, MAE and co-efficient of determination. From the above figure we can say that in our dataset the Boosted Decision Tree Regression is the best predictive model as it's RMSE and AME is minimum and Co-efficient of Determination is maximum as compared to other models.

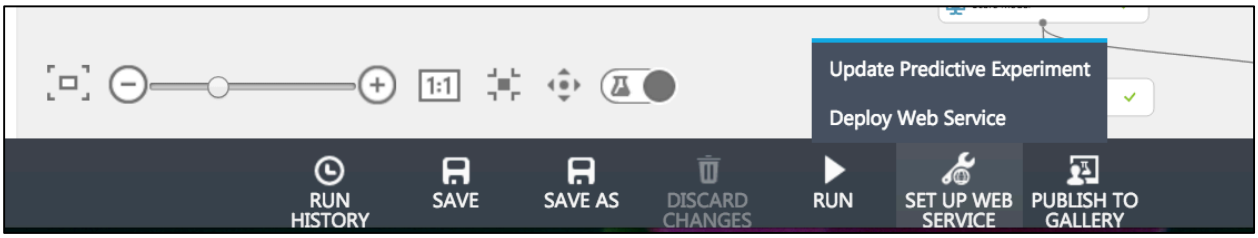
The evaluation metrics available for regression models are: Mean Absolute Error, Root Mean Absolute Error, Relative Absolute Error, Relative Squared Error, and the Coefficient of Determination.
The term "error" here represents the difference between the predicted value and the true value. The absolute value or the square of this difference are usually computed to capture the total magnitude of error across all instances, as the difference between the predicted and true value could be negative in some cases. The error metrics measure the predictive performance of a regression model in terms of the mean deviation of its predictions from the true values. Lower error values mean the model is more accurate in making predictions. An overall error metric of 0 means that the model fits the data perfectly.
The coefficient of determination, which is also known as R squared, is also a standard way of measuring how well the model fits the data. It can be interpreted as the proportion of variation explained by the model. A higher proportion is better in this case, where 1 indicates a perfect fit.

DEPLOYING WEBSERVICE:

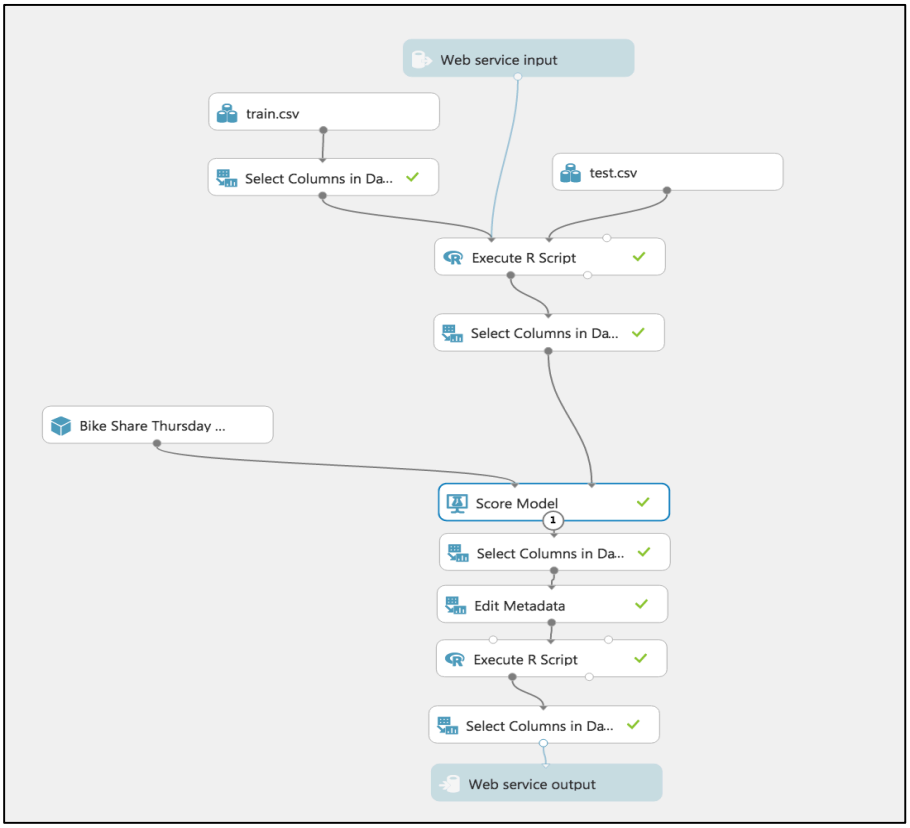
In our dataset we have Boosted Decision Tree Regression as the best predictive model. So for web service we need to select the train model of boosted decision tree regression in order to build the predictive experiment.



By converting to a predictive experiment, we are getting your trained model ready to be deployed as a scoring web service. Users of the web service will send input data to our model and our model will send back the prediction results.



Predictive Model



After running the predictive web service, we need to make certain changes in the predictive experiment created by azure. We need to the select columns after the web input and remove the predicted columns from the input. This will make sure that the model will not ask what we are predicting.
Now we have our predictive experiment, we can deploy it as an Azure web service. Using the web service, users can send data to your model and the model will return its predictions.

After we deploy web service we are redirected to azure web service dashboard which has the API key and the Test link, on clicking the Test link a dialog pops up for the input data for the service. These are the columns expected by the scoring experiment on entering a set of data and then clicking OK the results are generated by the web service are displayed at the bottom of the dashboard.

Now we have API and the request/response service of azure web service which we can consume in our end to display the predicted value. If we Click **REQUEST/RESPONSE** under **API HELP PAGE** on the service Dashboard to view the API help page, aside from the URI, we have input and output definitions and code samples. The API input and response, for our web service, is shown below and is the payload of the API call.

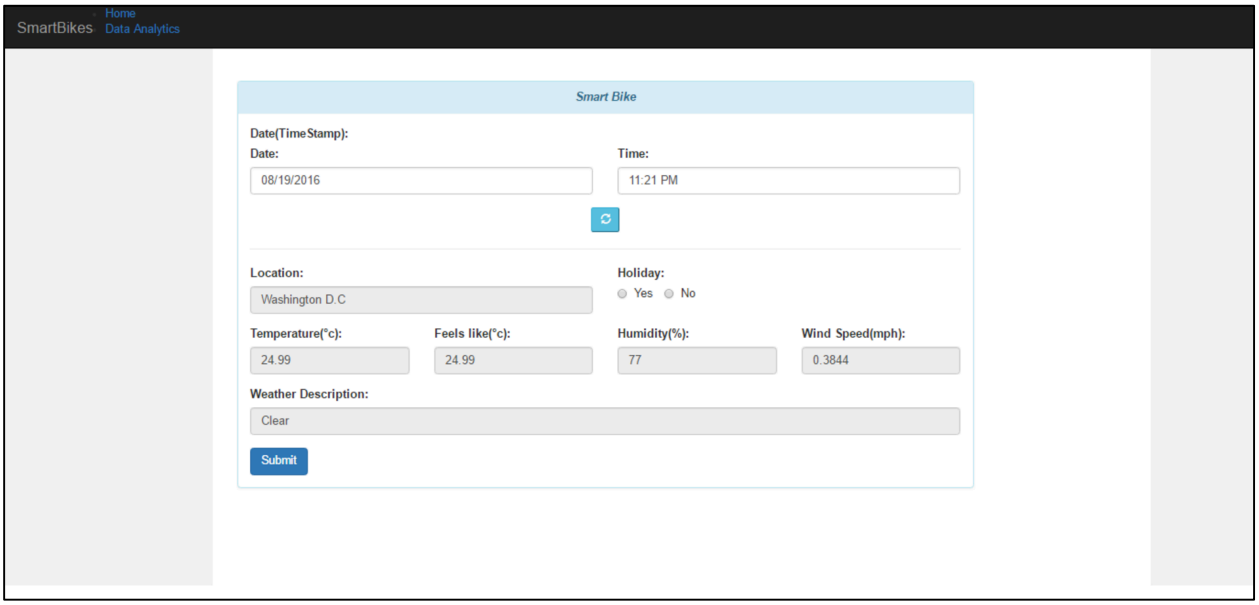
Sample Request


```
{
  "Inputs": {
    "input1": {
      "ColumnNames": [
        "datetime",
        "season",
        "holiday",
        "workingday",
        "weather",
        "temp",
        "atemp",
        "humidity",
        "windspeed"
      ],
```

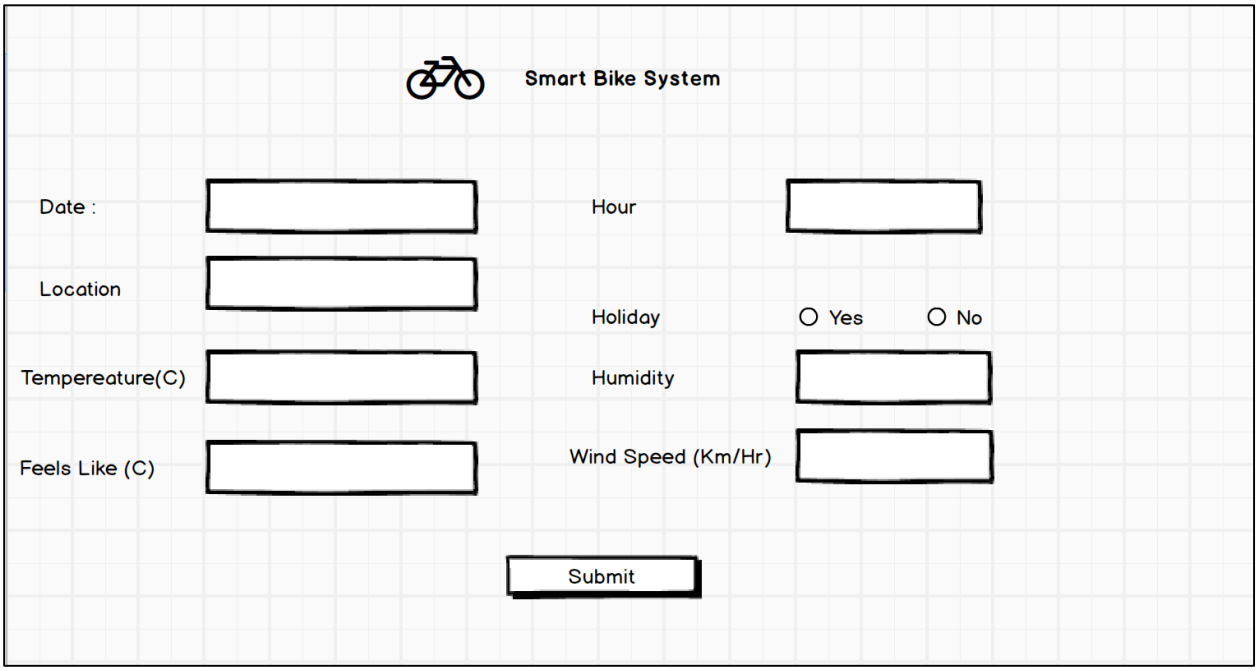

Sample Response


```
{
  "Results": {
    "output1": {
      "type": "DataTable",
      "value": {
        "ColumnNames": [
          "Predicted_Logreg"
        ],
        "ColumnTypes": [
          "Numeric"
        ],
        "Values": [
          [
            "0"
          ],
          [
            "0"
          ]
        ]
      }
    }
  }
}
```

Web Page:



Balsamiq Mockup:



C# Code to access web service

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
namespace CallRequestResponseService
{   public class StringTable
    {
        public string[] ColumnNames { get; set; }
        public string[,] Values { get; set; }
    }
class Program
    {
        static void Main(string[] args)
        {
            InvokeRequestResponseService().Wait();
        }
        static async Task InvokeRequestResponseService()
        {
            using (var client = new HttpClient())
            {   var scoreRequest = new
                {

                    Inputs = new Dictionary<string, StringTable> () {
                        {
                            "input1",
                            new StringTable()
                            {
                                ColumnNames = new string[] {"datetime",
"season", "holiday", "workingday", "weather", "temp", "atemp", "humidity",
"windspeed"},
                                Values = new string[,] {  { "", "0", "0",
"0", "0", "0", "0", "0", "0" },  { "", "0", "0", "0", "0", "0", "0", "0", "0"
},  }
                            }
                        },
                    },
```

```
                    GlobalParameters = new Dictionary<string, string>() {
}
                };
                const string apiKey = "abc123"; // Replace this with the API
key for the web service
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue( "Bearer", apiKey);

                client.BaseAddress = new
Uri("https://ussouthcentral.services.azureml.net/workspaces/65817a10f4a4452db
8059313c8182a68/services/478ba1eac0a0419eb3212699f0cd6e33/execute?api-
version=2.0&details=true");
                HttpResponseMessage response = await
client.PostAsJsonAsync("", scoreRequest);
                if (response.IsSuccessStatusCode)
             {string result = await response.Content.ReadAsStringAsync();
                    Console.WriteLine("Result: {0}", result);
                } else{
                    Console.WriteLine(string.Format("The request failed with
status code: {0}", response.StatusCode));
                    // Print the headers - they include the requert ID and
the timestamp, which are useful for debugging the failure
                    Console.WriteLine(response.Headers.ToString());
                    string responseContent = await
response.Content.ReadAsStringAsync();
                    Console.WriteLine(responseContent);
                }
            }
        }
    }
}
```

Bibliography

https://azure.microsoft.com/en-us/?b=16.24

https://en.wikipedia.org/wiki/SmartBike_DC

https://msdn.microsoft.com/en-us/library/azure/dn905801.aspx

https://balsamiq.com/

http://scholarship.shu.edu/cgi/viewcontent.cgi?article=1482&context=dissertations

https://cran.r-project.org/manuals.html

http://forecast.io/lines/