Advances in Data sciences Assignment3 Report

Team 3: Amitha Murali, Divyansh Srivastava, Jyoti Sharma

Index

Telecommunication Network Activity Analytics

1.1Problem statement

The study of socio-technical systems has been revolutionized by the unprecedented amount of digital records that are constantly being produced by human activities such as accessing Internet services, using mobile devices, and consuming energy and knowledge. In this paper, we describe the richest open multisource dataset ever released on two geographical areas. The dataset is composed of telecommunications, weather, news, social networks and electricity data from the city of Milan and the Province of Trentino.

In this case study, we focus only on the Internet and mobile activities in the Province of Trentino.

1.2 Background & Summary

The almost universal adoption of mobile phones and the exponential increase in the use of Internet services is generating an enormous amount of data that can be used to provide new fundamental and quantitative insights on socio-technical systems. The Call Detail Records (CDRs) of the 6.8 billion mobile phone subscribers worldwide potentially represent the most invaluable proxy for people's communication and mobility habits at a global scale. The availability of these data is indeed defining a novel area of research that exploits CDRs to extract human mobility. Moreover, the emergence of new geo located Information and Communications Technology (ICT) services like Twitter and Foursquare introduces further opportunities for researchers to inspect quantitatively different aspects of human behavior such as the social well-being of individuals and communities, socio-economic status of geographical regions, and people's mobility.

In this context, research challenges that provide access to a large number of research teams to the same dataset are becoming a truly valuable framework to advance the state of the art in the field. A prototypical example is offered by Orange's ‘Data for Development’ (D4D) initiative in 2013 and 2014–2015. Analogously, Telecom Italia in association with EIT ICT Labs, SpazioDati, MIT Media Lab, Northeastern University, Polytechnic University of Milan, Fondazione Bruno Kessler, University of Trento and Trento RISE recently organized the

‘Telecom Italia Big Data Challenge’, providing various geo-referenced and anonymized datasets. In the 2014edition they provided data of two Italian areas: the city of Milan and the Province of Trentino.

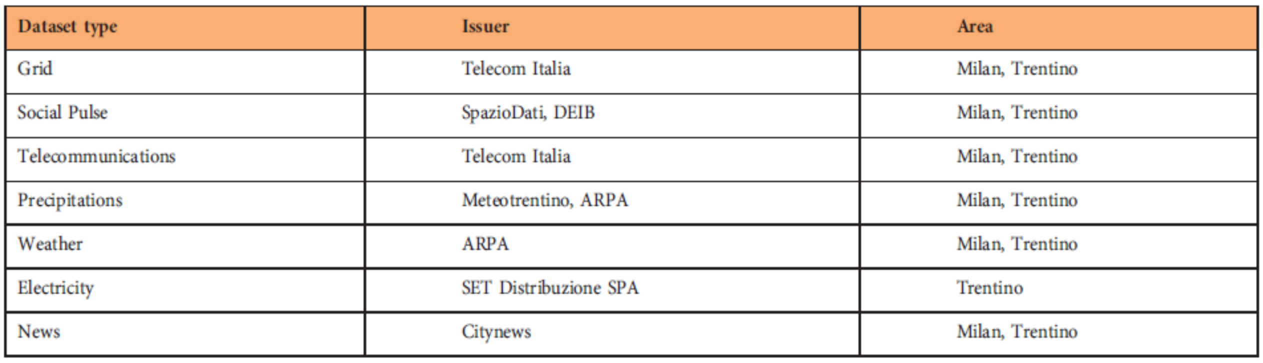
The Telecom Italia Big Data Challenge dataset is unique in that, since it is a rich, open multi-source aggregation of telecommunications, weather, news, social networks and electricity data from the city of Milan and the Province of Trentino. The multi-source nature of the current dataset permits the modeling of multiple dimensions of a given geographical area and to address a variety of problems and scientific issues that range from the classic human mobility and traffic analysis studies to energy consumption and linguistic studies. The dataset has been released to the whole research community and here we provide a detailed description of the data records' structure, and present the methodology used in the data collection/aggregation process. Finally, we validate the data and describe the usage notes and license.

Methods

Since the datasets come from various companies which have adopted different standards, their spatial distribution irregularity is aggregated in a grid with square cells. This allows comparisons between different areas and eases the geographical management of the data. Thus, the area of Milan is composed of a grid overlay of 1,000 (squares with size of about 235 × 235 meters and Trentino is composed of a grid overlay of 6,575 squares. This grid is projected with the WGS84 (EPSG:4326) standard.

1.2.1 Call detail records

The Call Detail Records (CDRs) are provided by the Semantics and Knowledge Innovation Lab (SKIL) of Telecom Italia. Every time a user engages a telecommunication



Dataset type	Issuer	Area
Grid	Telecom Italia	Milan, Trentino
Social Pulse	SpazioDati, DEIB	Milan, Trentino
Telecommunications	Telecom Italia	Milan, Trentino
Precipitations	Meteotrentino, ARPA	Milan, Trentino
Weather	ARPA	Milan, Trentino
Electricity	SET Distribuzione SPA	Trentino
News	Citynews	Milan, Trentino

interaction, a Radio Base Station (RBS) is assigned by the operator and delivers the communication through the network. Then, a new CDR is created recording the time of the interaction and the RBS which handled it. From the RBS it is possible to obtain an indication of the user's geographical location, thanks to the coverage maps Cmap which associates each RBS to the portion of territory which it serves.

In order to spatially aggregate the CDRs inside the grid, each interaction is associated with the

coverage area v of the RBS which handled it. Hence, the number of records si(t) in a grid square i at time t is computed as follows:

$$S_i(t) = \sum_{v \in C_{map}} R_v(t) \frac{A_{v \cap i}}{A_v}$$

where Rv, j(t) is the number of records in the coverage area v at time t, Av is the surface of the coverage area v and Av ∩ i is the surface of the spatial intersection between v and the square i.

There are many types of CDRs and Telecom Italia has recorded the following activities:

Received SMS a CDR is generated each time a user receives an SMS

Sent SMS a CDR is generated each time a user sends an SMS

Incoming Call a CDR is generated each time a user receives a call

Outgoing Call a CDR is generated each time a user issues a call

Internet a CDR is generated each time a user starts an Internet connection or ends an Internet

connection. During the same connection a CDR is generated if the connection lasts for more than 15 min or the user transferred more than 5 MB.

The shared datasets were created combining all this anonymous information, with a temporal aggregation of time slots of ten minutes. The number of records in the datasets follows the rule:

$$S_i'(t) = S_i(t)k$$

where k is a constant defined by Telecom Italia, which hides the true number of calls, SMS and connections.

Telecommunications interactions

Two types of CDR datasets were also produced to measure the interaction intensity between different locations: one from a particular area (Trentino) to any of the Italian provinces and one quantifying the interactions within the city/province (e.g., Trento to Trento).

Since Telecom Italia only possesses the data of its own customers, the computed interactions are only between them. This means that (at most) 34% of population's data is collected, due to Telecom Italia's market share. Moreover, there is no information about missed calls.

Data Records

The data has been collected over two months, from November 1st, 2013 to January 1st, 2014 and the information is geo-referenced to the city of Milan and to the Province of Trentino. Milan is the main industrial, commercial, and financial centre of Italy. The city has a population of about 1.3 million. Trentino is an autonomous province of Italy, located in the northern part of the country. It covers an area of more than 6,000 km2, with a total population of about 0.5 million.

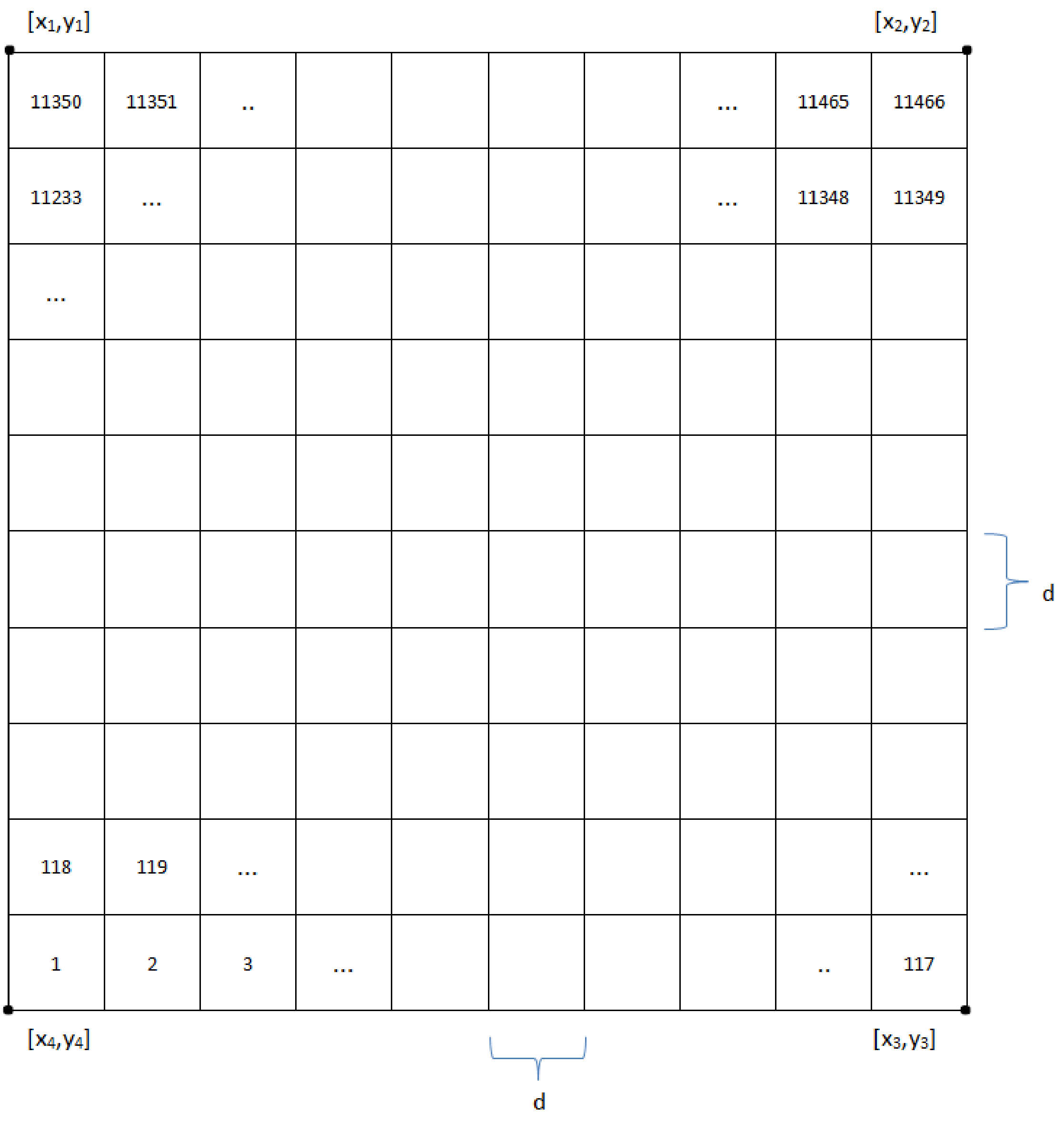
1.2.2 Trentino Grid

Some of the datasets are spatially aggregated using a regular grid over layed on the territory. The Grid dataset provides the geographical reference of each square which composes the grid in the reference system:

- ***square id***: identification string of a given square of the Milan or Trentino GRID; *TYPE: integer*
- ***Time Interval***: The cell geometry expressed as geoJSON; *TYPE: Geometry*

Description

The Trentino Grid has the following spatial description:



WGS84 (EPSG:4326)

$[x_1,y_1]$ = [10.46259520181421, 46.561715063458095 ***]***

$[x_2,y_2]$ = [11.987546900385569, 46.532036602887004 ***]***

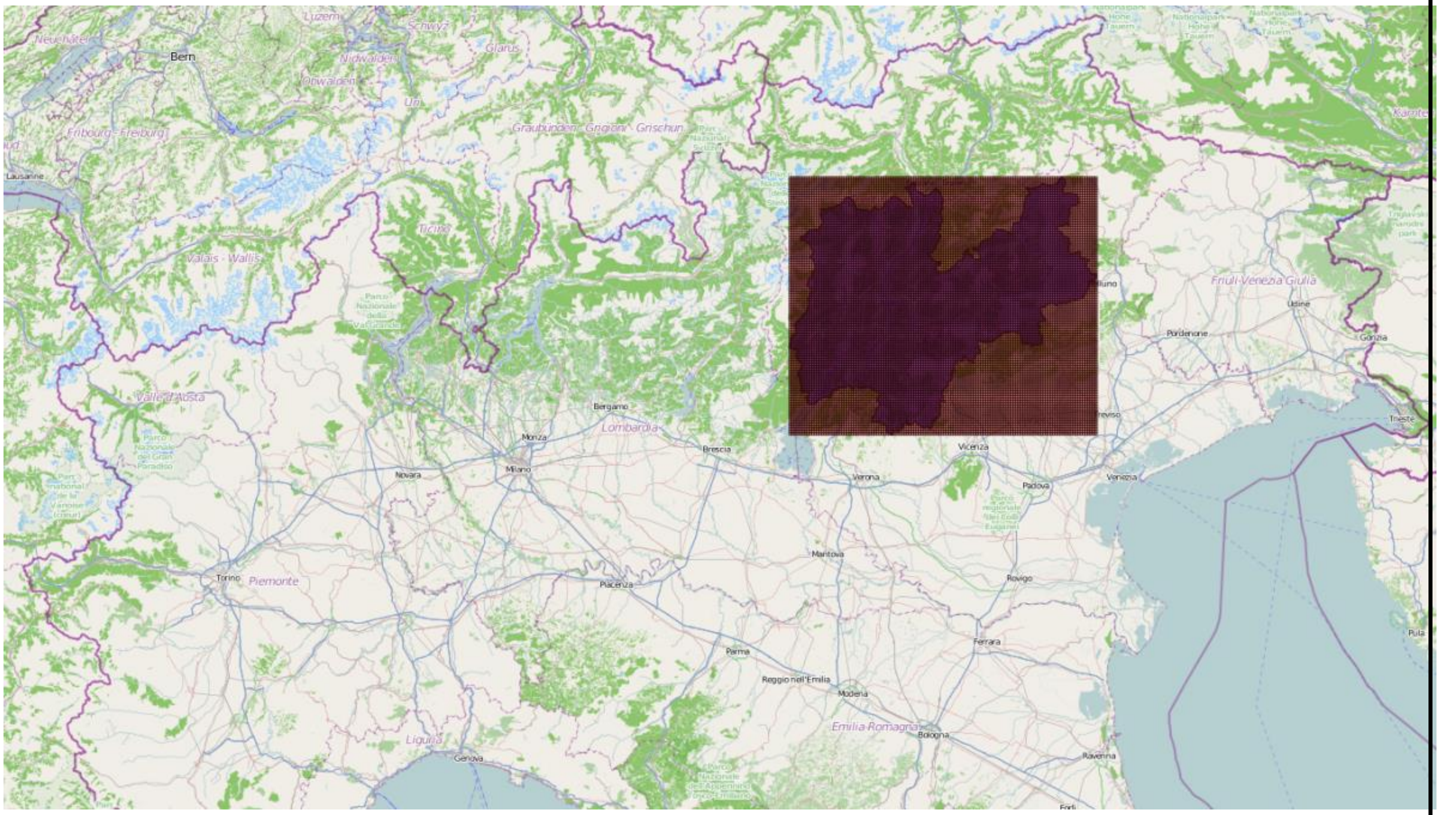
$[x_3,y_3]$ = [11.940320242153627, 45.651198648022117 ***]***

$[x_4,y_4]$ = [10.439443587087794, 45.679983268451274 ***]***

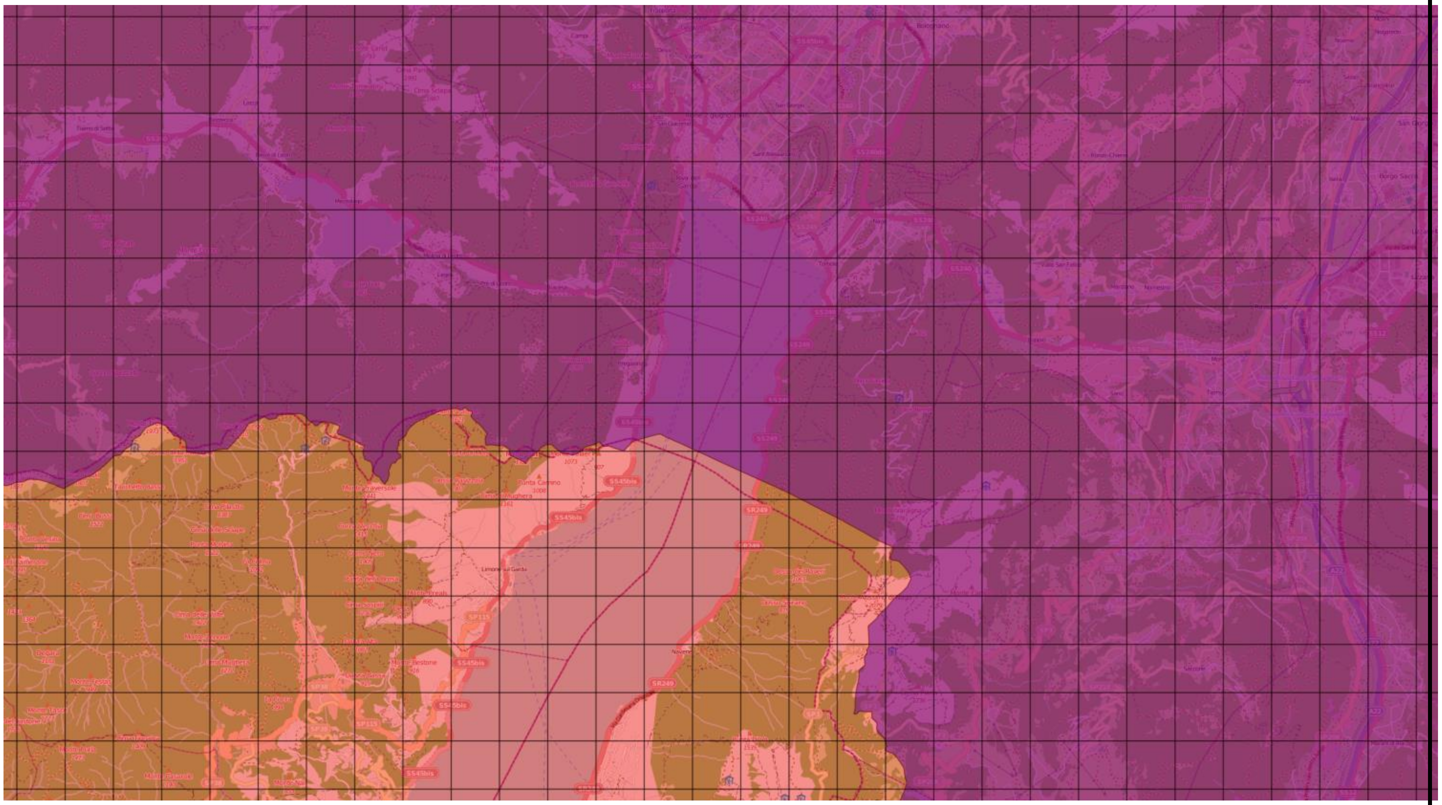
d: 1000 m

As previously shown squares are numbered with ids. The square id numbering starts from the bottom left corner of the grid and grows till its right top corner.

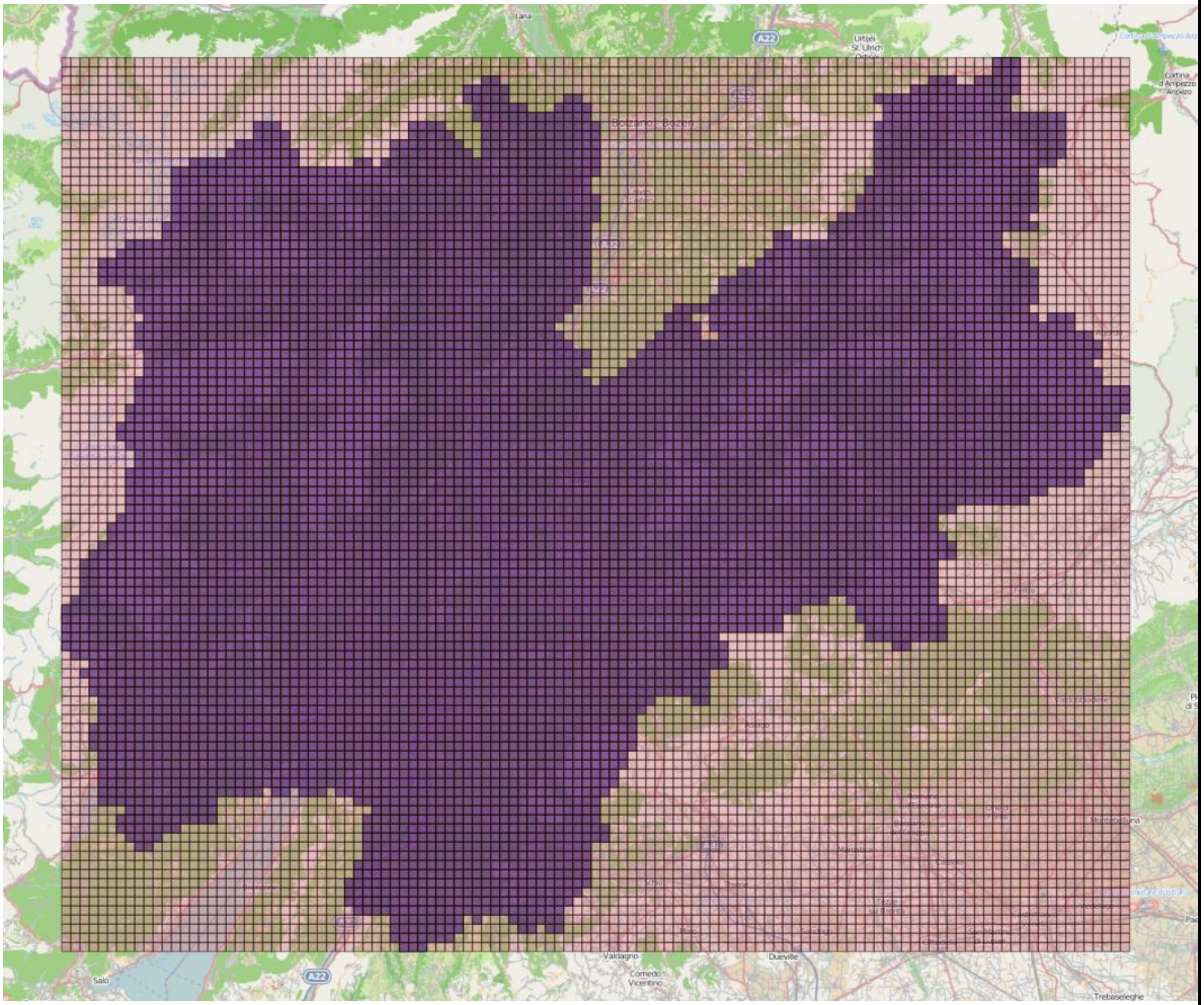
The portion of territory covered by the grid is shown in the following picture where the grid (red-gray) is overlaid to the northern part of Italy. The Trentino province is highlighted in violet.



In order to give an idea about the spatial granularity of the grid we provide a zoomed view of the above figure which shows the northern tip of the Lake Garda.



The geoJSON file available in the resources section and the API exclusively provide information for the squares overlapping the Trentino province. These squares are those highlighted in the following picture.


Bolzano - Bozen
A22
Valdagno
Dueville
Salò
Trebaseleghe

1.2.3 Dataset Schema:

This dataset serves as measure of the level of interaction between the users and the mobile phone network.

● ***Square id:*** identification string of a given square of Trentino GRID; *TYPE: numeric*

● ***Time Interval:*** start interval time expressed in milliseconds. The end interval time can be obtained by adding 600,000 milliseconds (10 min) to this value; *TYPE: numeric*

● ***SMS-in activity:*** activity proportional to the amount of received SMSs inside a given Square id and during a given Time interval. The SMSs are sent from the nation identified by the Country code; *TYPE: numeric*

● ***SMS-out activity***: activity proportional to the amount of sent SMSs inside a given Square id during a given Time interval. The SMSs are received in the nation identified by the Country code; *TYPE: numeric*

● ***Call-in activity***: activity proportional to the amount of received calls inside the Square id during a given Time interval. The calls are issued from the nation identified by the Country code; *TYPE: numeric*

● ***Call-out activity:*** activity proportional to the amount of issued calls inside a given Square id during a given Time interval. The calls are received in the nation identified by the Country code; *TYPE: numeric*

● ***Internet traffic activity***: number of CDRs generated inside a given Square id during a given Time interval. The Internet traffic is initiated from the nation identified by the Country code;

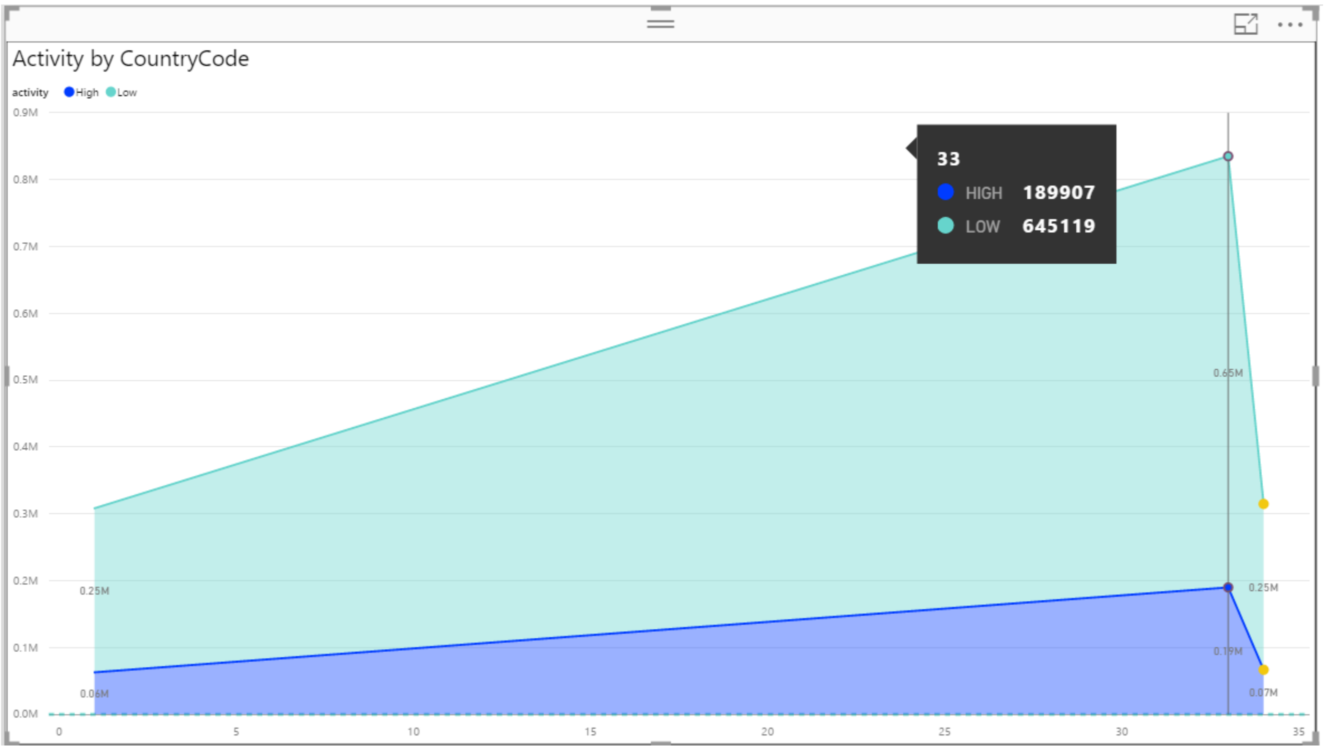
● ***Country code***: the phone country code of the nation. *TYPE: numeric*

1.3 Power BI Analysis:

We have used the cleansed data to visualize and draw patterns about the mobile network activities in the Province of Trentino.

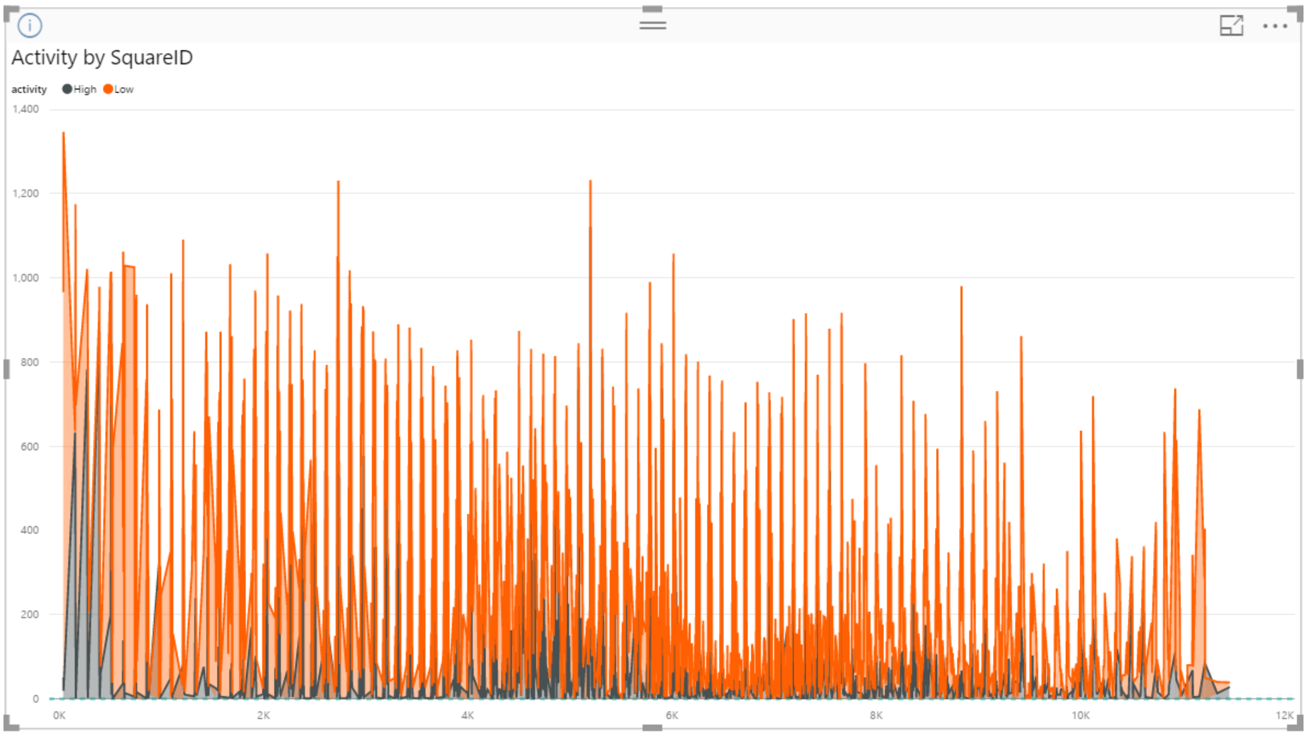
Graph1: Activity by Country Code

- The below graph shows that sms, call and internet activities are high between Trento and France (country code 33) than Trento and Spain (country code 34) or Trento and United State of America (country code 1).



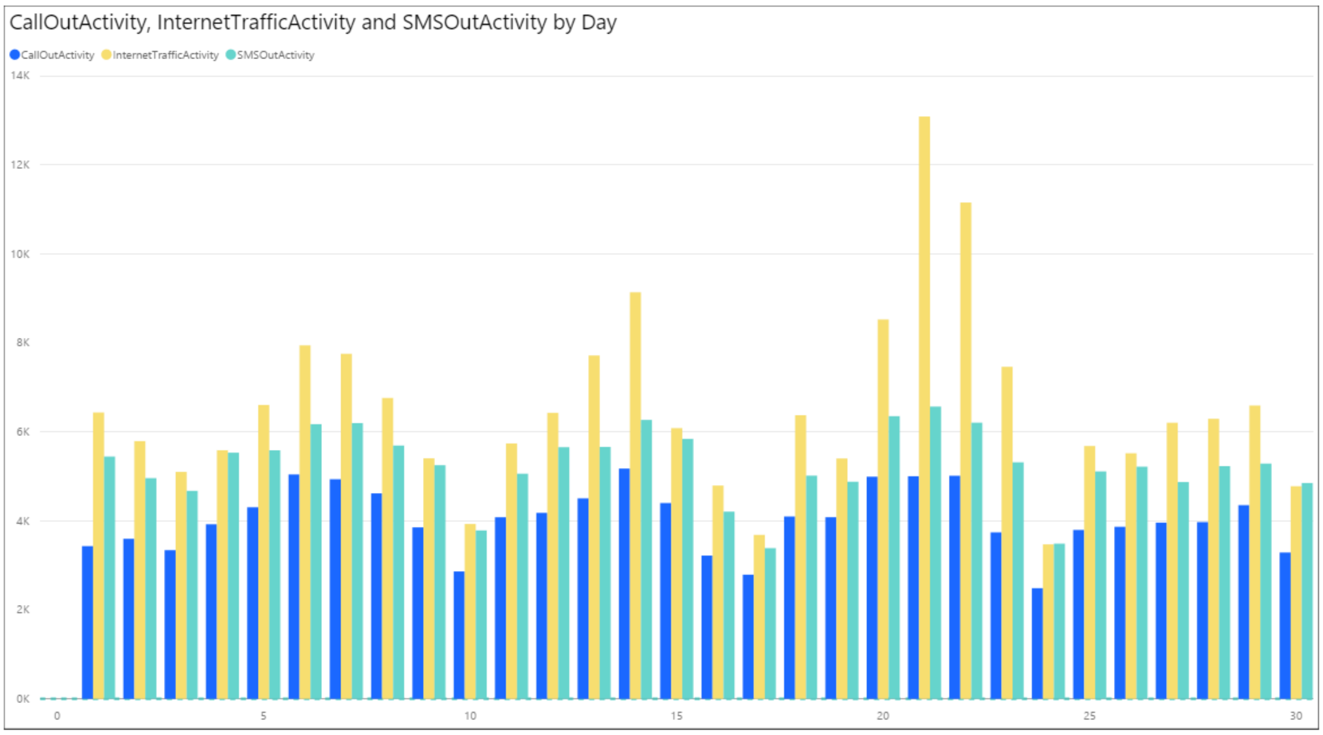
Graph2: Activity by Square ID

- The below graph shows that sms, call and internet activities varying between square IDs. Orange colour represents low activities and grey represents high activities.



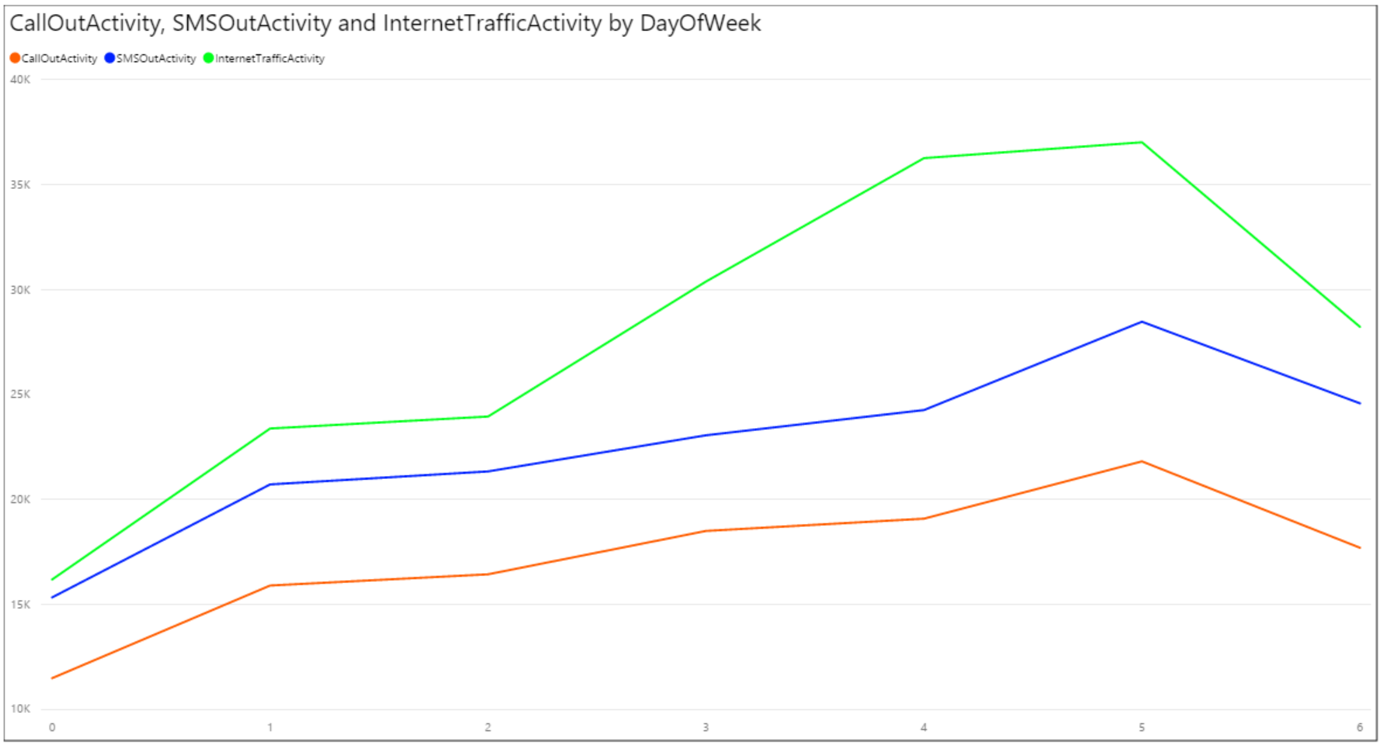
Graph3: Call out activity, Internet traffic activity and SMS out activity by the day

- The below graph shows that sms out, call out and internet activities by the day. With 21st of November having a high internet traffic activity. But this can't be generalized as we have limited data of just one month.



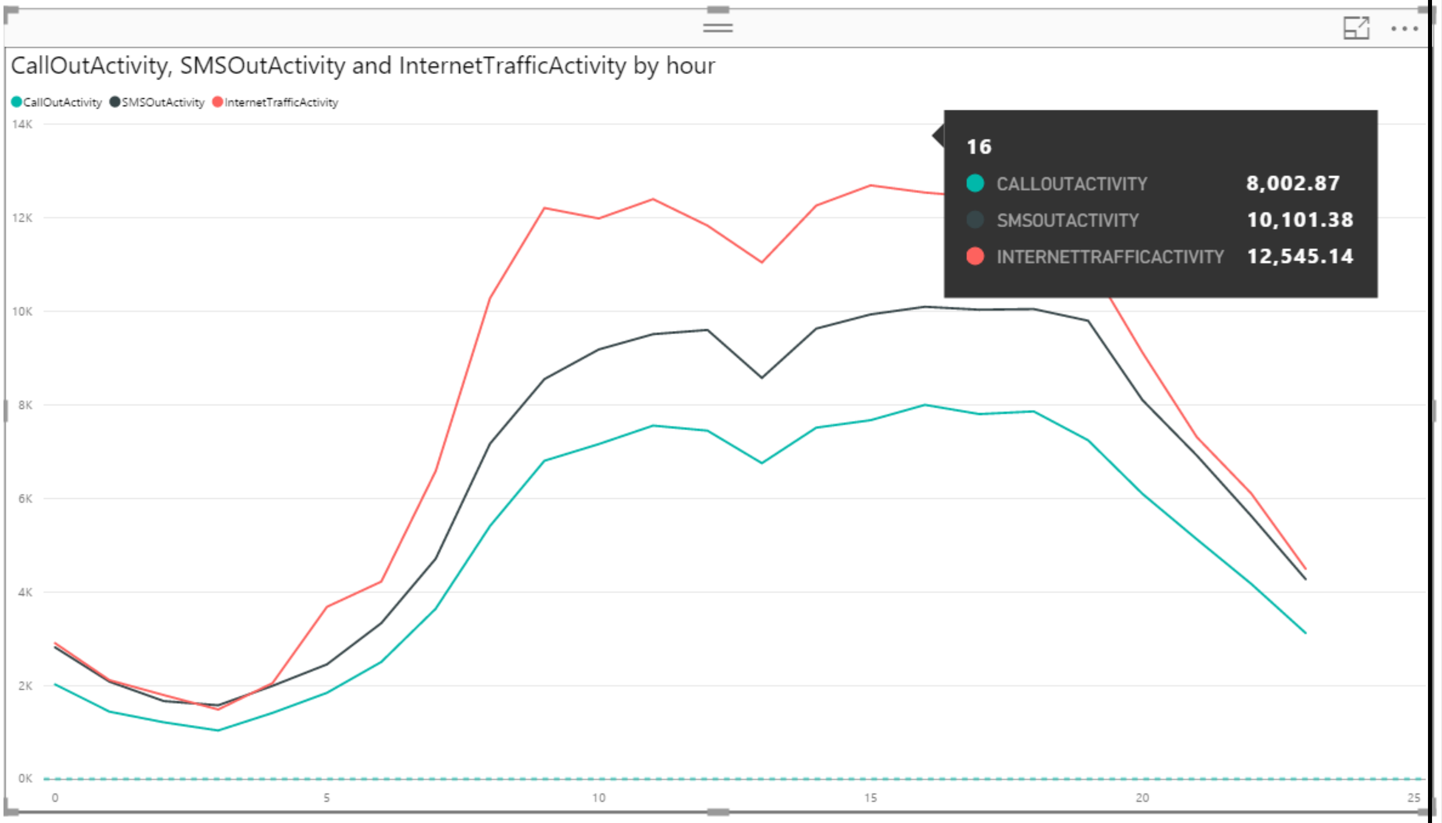
Graph4: Call out activity, Internet traffic activity and SMS out activity by dayOfWeek

- The below graph shows Call out activity, Internet traffic activity and SMS out activity by dayOfWeek. As shown in the graph, there is a rise as the week proceeds with Friday (dayOfWeek -> 5) recording the highest activity values.



Graph5: Call out activity, Internet traffic activity and SMS out activity by hour

- The below graph shows Call out activity, Internet traffic activity and SMS out activity by hour. As shown in the graph, there is a rise as the day proceeds but a sudden dip at 13 hours and then a rise again.



1.4 Data Cleansing:

Data Cleansing is divided into two parts:

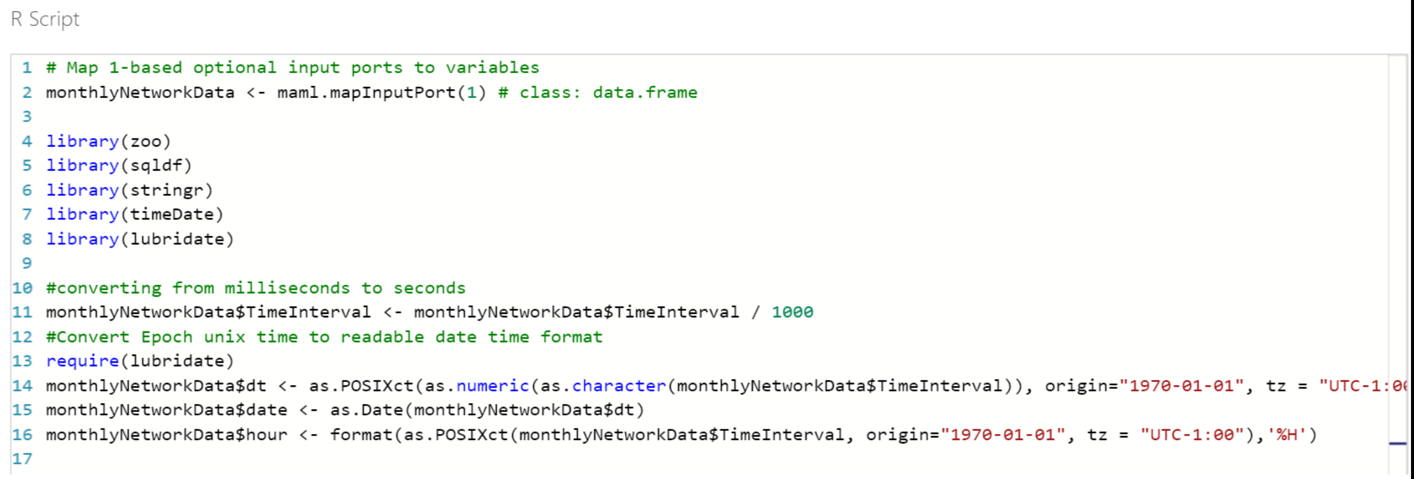
R studio:

- Read the daily network data text file.
- Consolidate them to form a monthly network data for November.
- Filter out data with country code 1,33 and 34 (1 represents United states of America, 33 represents France and 34 represents Spain).
- Merge the column names with data.
- Output a csv file with this consolidated data and name it - monthlyNetworkData.csv

```r
#Set working directory
setwd("C:/Users/amitha.murali/Desktop/Assignment3")

monthlyNetworkData <- vector()
for (i in 1:30) {

  #padding with Zero
  j <- str_pad(i,2,pad="0")

  #Read daily network data
  fileName <- paste ("sms-call-internet-tn-2013-12-",j,".txt",sep="",collapse ="")
  dailyNetworkData <- read.table(fileName,sep="\t")

  dailyNetworkData <- read.table("sms-call-internet-tn-2013-12-01.txt",sep="\t")

  #Filtered the daily data to fetch data for certain country(US, France, Spain)
  toMatch <- c(1,33,34)
  dailyFilteredData <- dailyNetworkData[which(dailyNetworkData[,3] %in% toMatch), ]

  #combine daily data to create monthly data
  monthlyNetworkData <- rbind(monthlyNetworkData,dailyFilteredData)

}
```

Azure ML Studio:

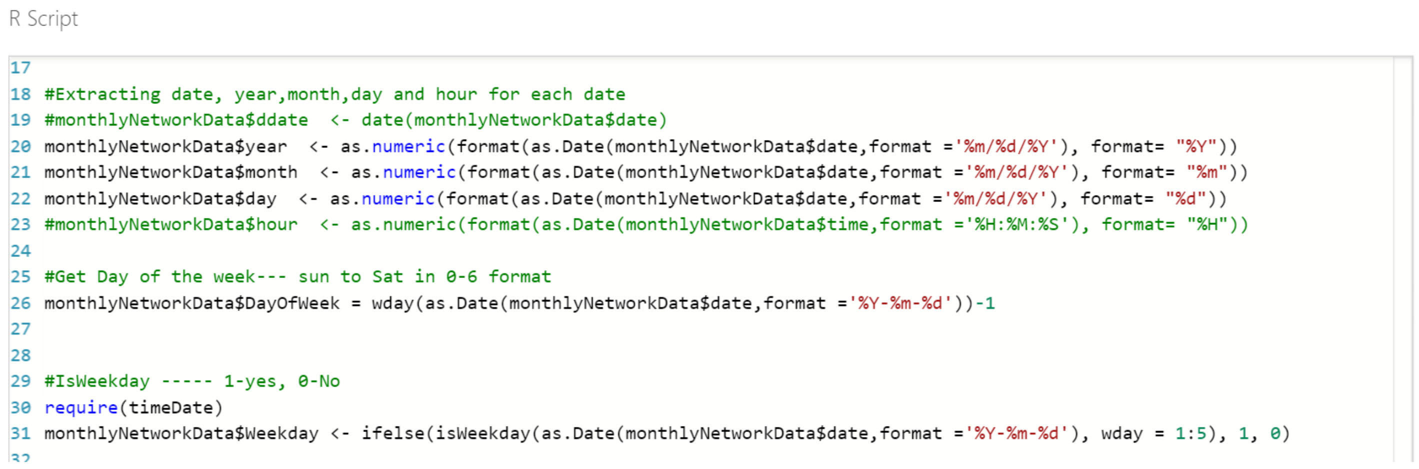
- Read the consolidated csv file.
- Convert Epoch unix time to readable date time format. This has been elaborated further in the below section.



R Script

```r
# Map 1-based optional input ports to variables
monthlyNetworkData <- maml.mapInputPort(1) # class: data.frame

library(zoo)
library(sqldf)
library(stringr)
library(timeDate)
library(lubridate)

#converting from milliseconds to seconds
monthlyNetworkData$TimeInterval <- monthlyNetworkData$TimeInterval / 1000
#Convert Epoch unix time to readable date time format
require(lubridate)
monthlyNetworkData$dt <- as.POSIXct(as.numeric(as.character(monthlyNetworkData$TimeInterval)), origin="1970-01-01", tz = "UTC-1:0
monthlyNetworkData$date <- as.Date(monthlyNetworkData$dt)
monthlyNetworkData$hour <- format(as.POSIXct(monthlyNetworkData$TimeInterval, origin="1970-01-01", tz = "UTC-1:00"),'%H')

```

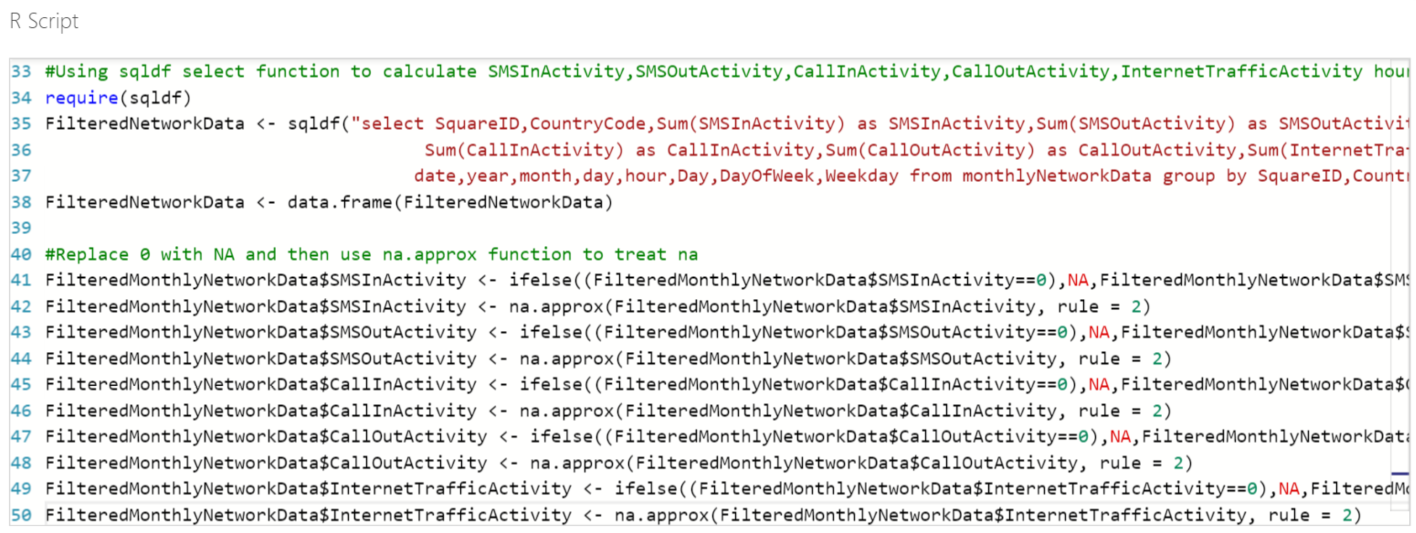
- Extract date, year, month, day and hour values for each date
- Create a column for Day of the week by assuming sun to Sat is represented in 0-6 format
- Create a column for IsWeekday by assuming 1-yes, 0-No



R Script

```r

#Extracting date, year,month,day and hour for each date
#monthlyNetworkData$ddate  <- date(monthlyNetworkData$date)
monthlyNetworkData$year  <- as.numeric(format(as.Date(monthlyNetworkData$date,format ='%m/%d/%Y'), format= "%Y"))
monthlyNetworkData$month  <- as.numeric(format(as.Date(monthlyNetworkData$date,format ='%m/%d/%Y'), format= "%m"))
monthlyNetworkData$day  <- as.numeric(format(as.Date(monthlyNetworkData$date,format ='%m/%d/%Y'), format= "%d"))
#monthlyNetworkData$hour  <- as.numeric(format(as.Date(monthlyNetworkData$time,format ='%H:%M:%S'), format= "%H"))

#Get Day of the week--- sun to Sat in 0-6 format
monthlyNetworkData$DayOfWeek = wday(as.Date(monthlyNetworkData$date,format ='%Y-%m-%d'))-1


#IsWeekday ----- 1-yes, 0-No
require(timeDate)
monthlyNetworkData$Weekday <- ifelse(isWeekday(as.Date(monthlyNetworkData$date,format ='%Y-%m-%d'), wday = 1:5), 1, 0)
```

- Use sqldf select function from sqldf package to calculate SMSInActivity, SMSOutActivity, CallInActivity, CallOutActivity, InternetTrafficActivity hourly data.
- Treating the NA --- use na.approx function from zoo package to treat NA



R Script

```
#Using sqldf select function to calculate SMSInActivity,SMSOutActivity,CallInActivity,CallOutActivity,InternetTrafficActivity hou
require(sqldf)
FilteredNetworkData <- sqldf("select SquareID,CountryCode,Sum(SMSInActivity) as SMSInActivity,Sum(SMSOutActivity) as SMSOutActivi
                                   Sum(CallInActivity) as CallInActivity,Sum(CallOutActivity) as CallOutActivity,Sum(InternetTra
                                  date,year,month,day,hour,Day,DayOfWeek,Weekday from monthlyNetworkData group by SquareID,Count
FilteredNetworkData <- data.frame(FilteredNetworkData)

#Replace 0 with NA and then use na.approx function to treat na
FilteredMonthlyNetworkData$SMSInActivity <- ifelse((FilteredMonthlyNetworkData$SMSInActivity==0),NA,FilteredMonthlyNetworkData$SMS
FilteredMonthlyNetworkData$SMSInActivity <- na.approx(FilteredMonthlyNetworkData$SMSInActivity, rule = 2)
FilteredMonthlyNetworkData$SMSOutActivity <- ifelse((FilteredMonthlyNetworkData$SMSOutActivity==0),NA,FilteredMonthlyNetworkData$
FilteredMonthlyNetworkData$SMSOutActivity <- na.approx(FilteredMonthlyNetworkData$SMSOutActivity, rule = 2)
FilteredMonthlyNetworkData$CallInActivity <- ifelse((FilteredMonthlyNetworkData$CallInActivity==0),NA,FilteredMonthlyNetworkData$
FilteredMonthlyNetworkData$CallInActivity <- na.approx(FilteredMonthlyNetworkData$CallInActivity, rule = 2)
FilteredMonthlyNetworkData$CallOutActivity <- ifelse((FilteredMonthlyNetworkData$CallOutActivity==0),NA,FilteredMonthlyNetworkDat
FilteredMonthlyNetworkData$CallOutActivity <- na.approx(FilteredMonthlyNetworkData$CallOutActivity, rule = 2)
FilteredMonthlyNetworkData$InternetTrafficActivity <- ifelse((FilteredMonthlyNetworkData$InternetTrafficActivity==0),NA,FilteredM
FilteredMonthlyNetworkData$InternetTrafficActivity <- na.approx(FilteredMonthlyNetworkData$InternetTrafficActivity, rule = 2)
```

- Output a csv file with this consolidated data and name it - FilteredMonthlyNetworkData.csv

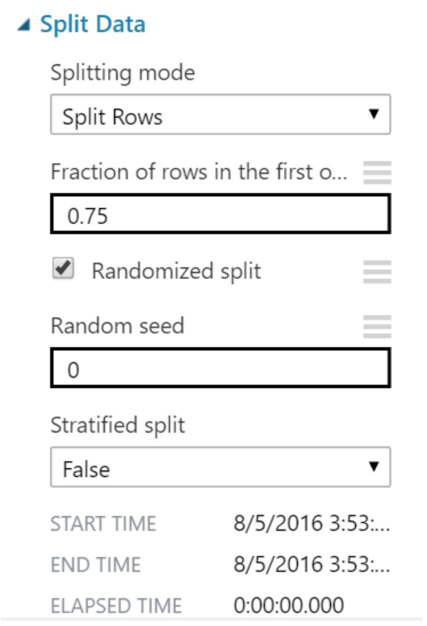
What is epoch time?

The **Unix epoch** (or **Unix time** or **POSIX time** or **Unix timestamp**) is the number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT), not counting leap seconds (in ISO 8601: 1970-01-01T00:00:00Z). Literally speaking the epoch is Unix time 0 (midnight 1/1/1970), but 'epoch' is often used as a synonym for 'Unix time'. Many Unix systems store epoch dates as a signed 32-bit integer, which might cause problems on January 19, 2038 (known as the Year 2038 problem or Y2038).

Human readable time	**Seconds**
1 hour	3600 seconds
1 day	86400 seconds
1 week	604800 seconds
1 month (30.44 days)	2629743 seconds
1 year (365.24 days)	31556926 seconds

Azure Modules:

- **Split Data -** To split a dataset into two equal parts, just add the Split Data module after the dataset without no other changes. By default, the module splits the dataset in two equal parts. For data with an odd number of rows, the second output gets the remainder.


Split Data

Splitting mode: Split Rows

Fraction of rows in the first o...: 0.75

☑ Randomized split

Random seed: 0

Stratified split: False

START TIME	8/5/2016 3:53:...
END TIME	8/5/2016 3:53:...
ELAPSED TIME	0:00:00.000

- **Train Model** - Training a classification or regression model is a kind of *supervised machine learning*. That means you must provide a dataset that contains historical data from which to learn patterns. The data should contain both the outcome you are trying to predict, and related factors (variables). The machine learning model uses the data to extract statistical patterns and build a model.

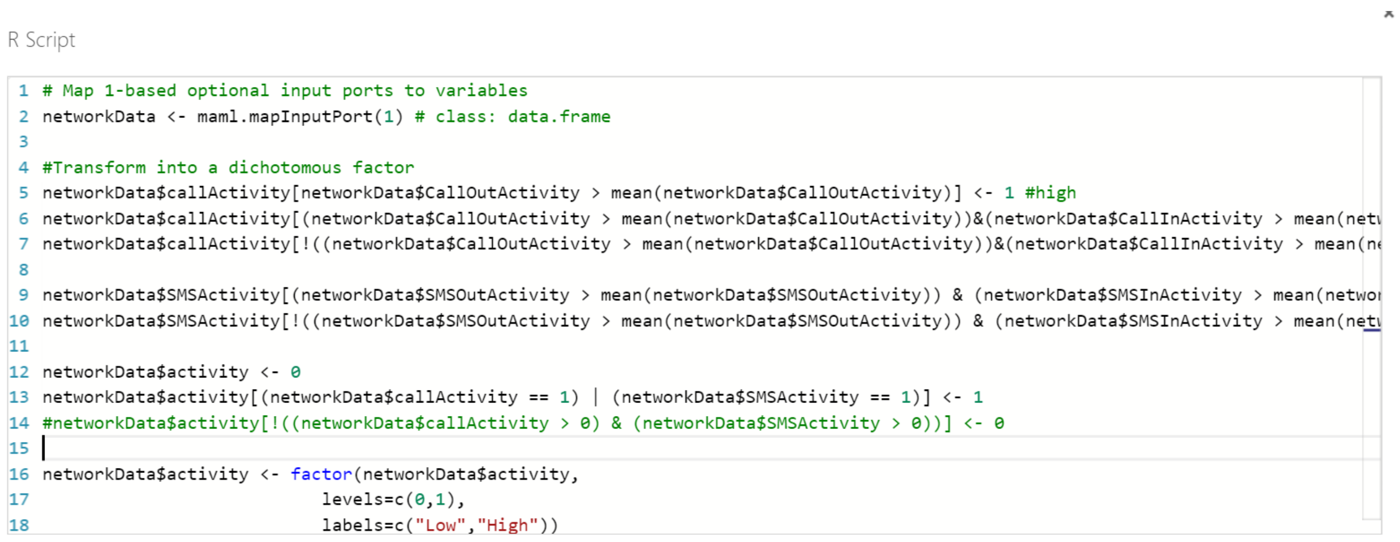
  When you configure Train Model**,** you must also connect an already configured model, such as a regression algorithm, decision tree model, or another machine learning module.

- **Score Model -** Score Model is used to generate predictions using a trained classification or regression model. The predicted value can be in many different formats, depending on the model and your input data: If you are using a classification model to create the scores, Score Model outputs a predicted value for the class, as well as the probability of the predicted value. For regression models, Score Model generates just the predicted numeric value.

- **Evaluate Model** - Evaluate Model is used to measure the accuracy of a trained classification model or regression model. You provide a dataset containing scores generated from a trained model, and the Evaluate Model module computes a set of industry-standard evaluation metrics. The metrics returned by Evaluate Model depend on the type of model that you are evaluating

1.5 Classification Models

We build classification model to classify the given squareId/cellId into high and low activity area. Sms, call and internet activities are monitored and using the following business logic, we classified data into high n low activity.

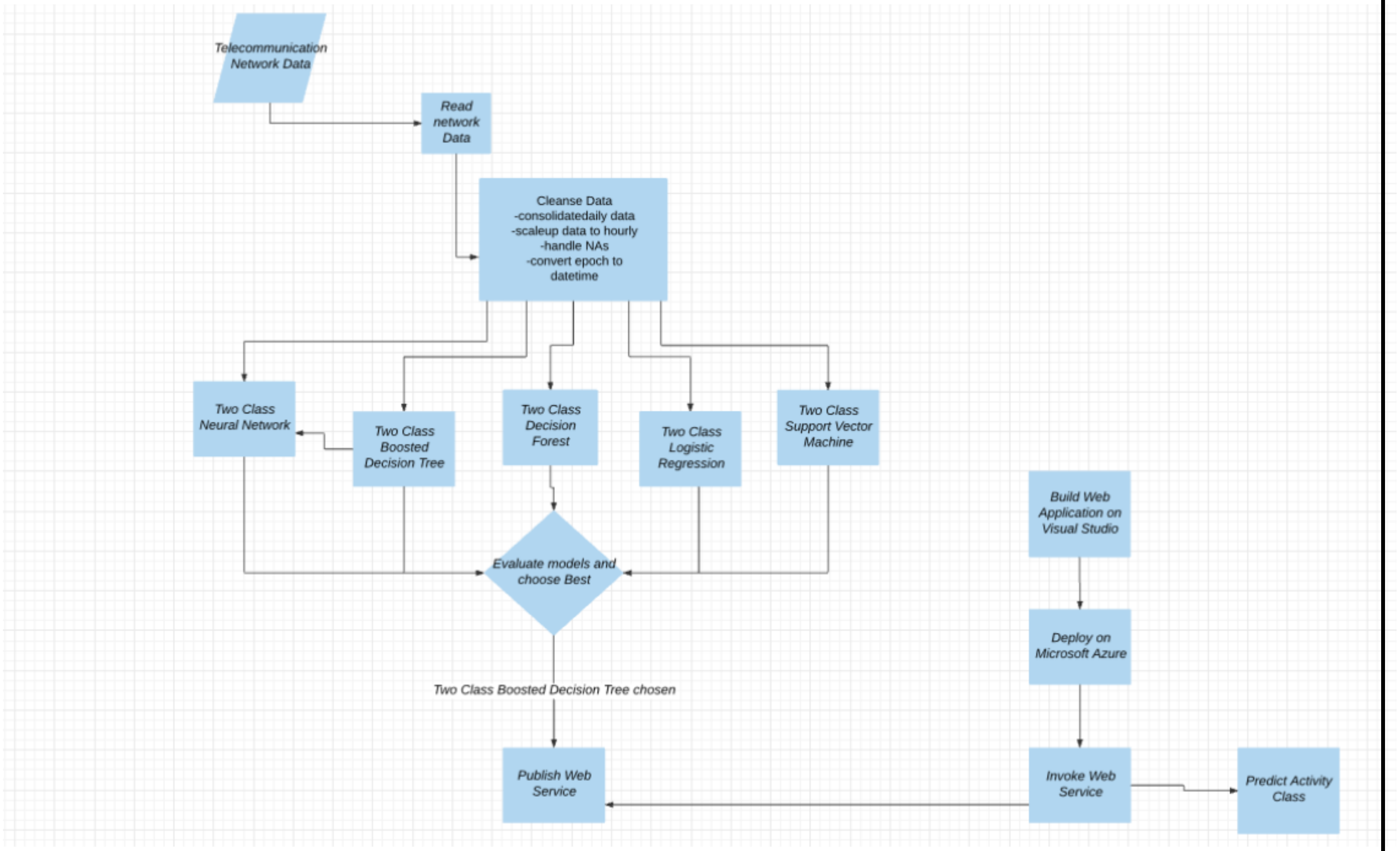


```
# Map 1-based optional input ports to variables
networkData <- maml.mapInputPort(1) # class: data.frame

#Transform into a dichotomous factor
networkData$callActivity[networkData$CallOutActivity > mean(networkData$CallOutActivity)] <- 1 #high
networkData$callActivity[(networkData$CallOutActivity > mean(networkData$CallOutActivity))&(networkData$CallInActivity > mean(netw
networkData$callActivity[!((networkData$CallOutActivity > mean(networkData$CallOutActivity))&(networkData$CallInActivity > mean(ne

networkData$SMSActivity[(networkData$SMSOutActivity > mean(networkData$SMSOutActivity)) & (networkData$SMSInActivity > mean(networ
networkData$SMSActivity[!((networkData$SMSOutActivity > mean(networkData$SMSOutActivity)) & (networkData$SMSInActivity > mean(netw

networkData$activity <- 0
networkData$activity[(networkData$callActivity == 1) | (networkData$SMSActivity == 1)] <- 1
#networkData$activity[!((networkData$callActivity > 0) & (networkData$SMSActivity > 0))] <- 0

networkData$activity <- factor(networkData$activity,
                               levels=c(0,1),
                               labels=c("Low","High"))
```

1.5.1Overall Design

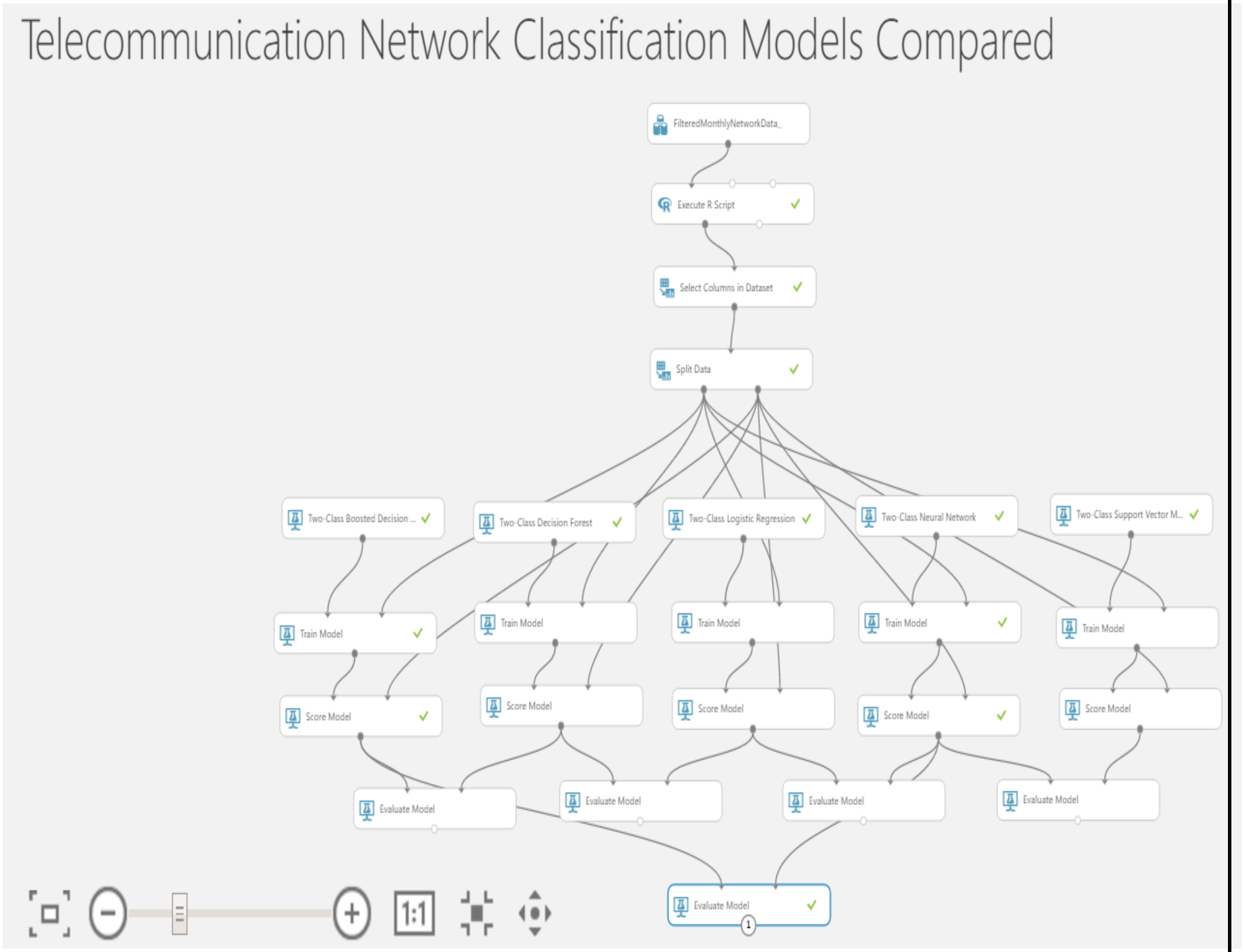
- Read the telecommunication network data.
- Cleanse the data as explained in the data cleansing section.
- Implement two class neural network, two class boosted decision tree, two class decision forest, two class logistic regression, two class support vector machine. Parameters used in each model is elaborated in the further module below.
- Compare the models through the roc curve, root mean square error, mean absolute error parameters and choose the best model.
- Publish the best classification model created for this data as a web service.

- Build a web application using visual studio and deploy it on Microsoft Azure.
- This web application is used as an interface to invoke the web service expose through Azure.
- Predictions are performed through this web application to predict if the given area is going to be high or low activity area.
-


Telecommunication Network Data
Read network Data
Cleanse Data -consolidatedaily data -scaleup data to hourly -handle NAs -convert epoch to datetime
Two Class Neural Network
Two Class Boosted Decision Tree
Two Class Decision Forest
Two Class Logistic Regression
Two Class Support Vector Machine
Evaluate models and choose Best
Two Class Boosted Decision Tree chosen
Publish Web Service
Build Web Application on Visual Studio
Deploy on Microsoft Azure
Invoke Web Service
Predict Activity Class

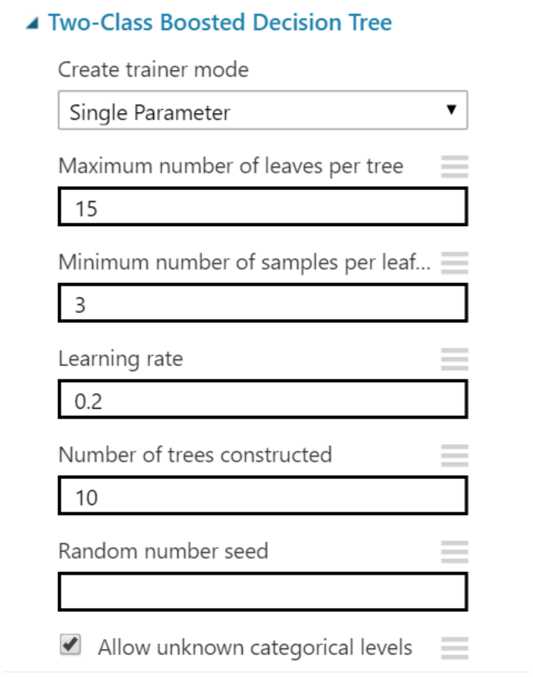
1.5.2 Azure models

Below is the telecommunication network classification model created for classifying the cell id as high activity and low activity area.



1.5.3 Two class Boosted Decision Tree

A boosted decision tree is an ensemble learning method in which the second tree corrects for the errors of the first tree, the third tree corrects for the errors of the first and second trees, and so forth. Predictions are based on the entire ensemble of trees together that makes the prediction. Generally, when properly configured, boosted decision trees are the easiest methods with which to get top performance on a wide variety of machine learning tasks. However, they are also one of the more memory-intensive learners, and the current implementation holds everything in memory; therefore, a boosted decision tree model might not be able to process the very large datasets that some linear learners can handle.

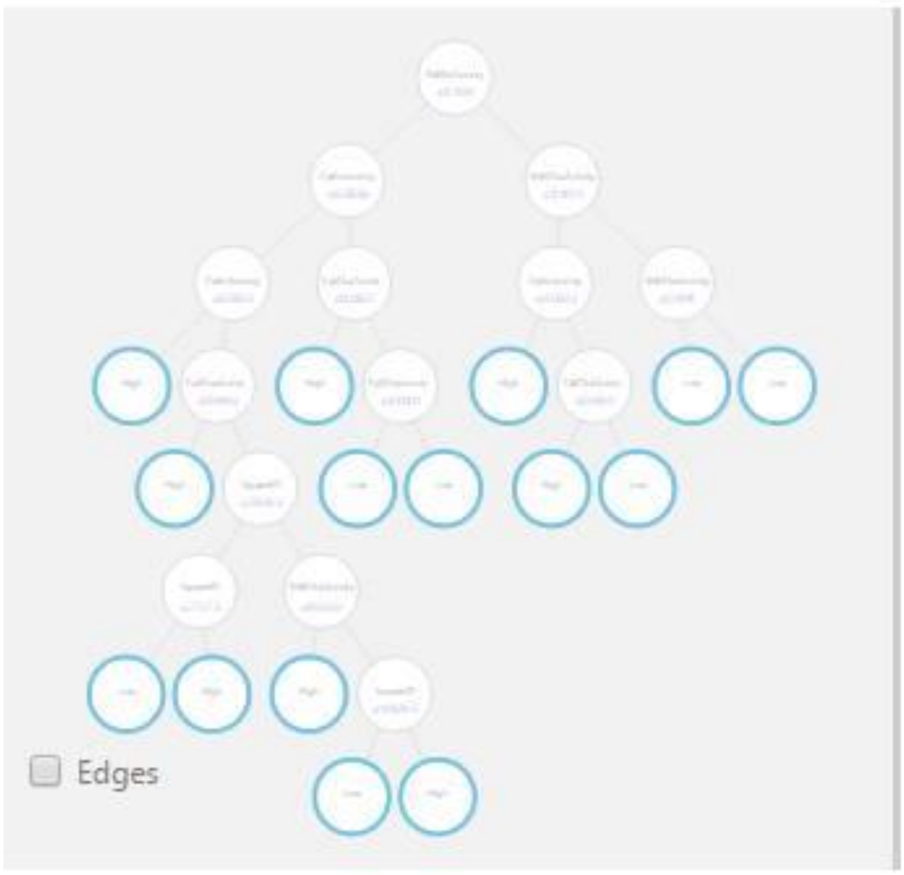


Two-Class Boosted Decision Tree

Create trainer mode	Single Parameter
Maximum number of leaves per tree	15
Minimum number of samples per leaf...	3
Learning rate	0.2
Number of trees constructed	10
Random number seed	

✓ Allow unknown categorical levels

Classification Tree



ROC curve

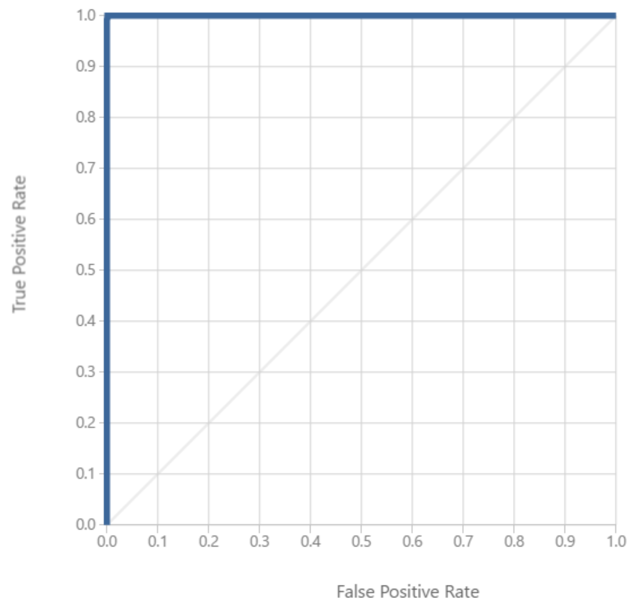
The ROC curve plots the pairs {sensitivity, 1-specificity} as the cutoff value increases from 0 and 1

• **Sensitivity** (also called the **true positive rate**, or the **recall** in some fields) measures the proportion of positives that are correctly identified (e.g., the percentage of sick people who are correctly identified as having the condition).

• **Specificity** (also called the **true negative rate**) measures the proportion of negatives that are correctly identified

as such (e.g., the percentage of healthy people who are correctly identified as not having the condition).

• Better performance is reflected by curves that are closer to the top left corner

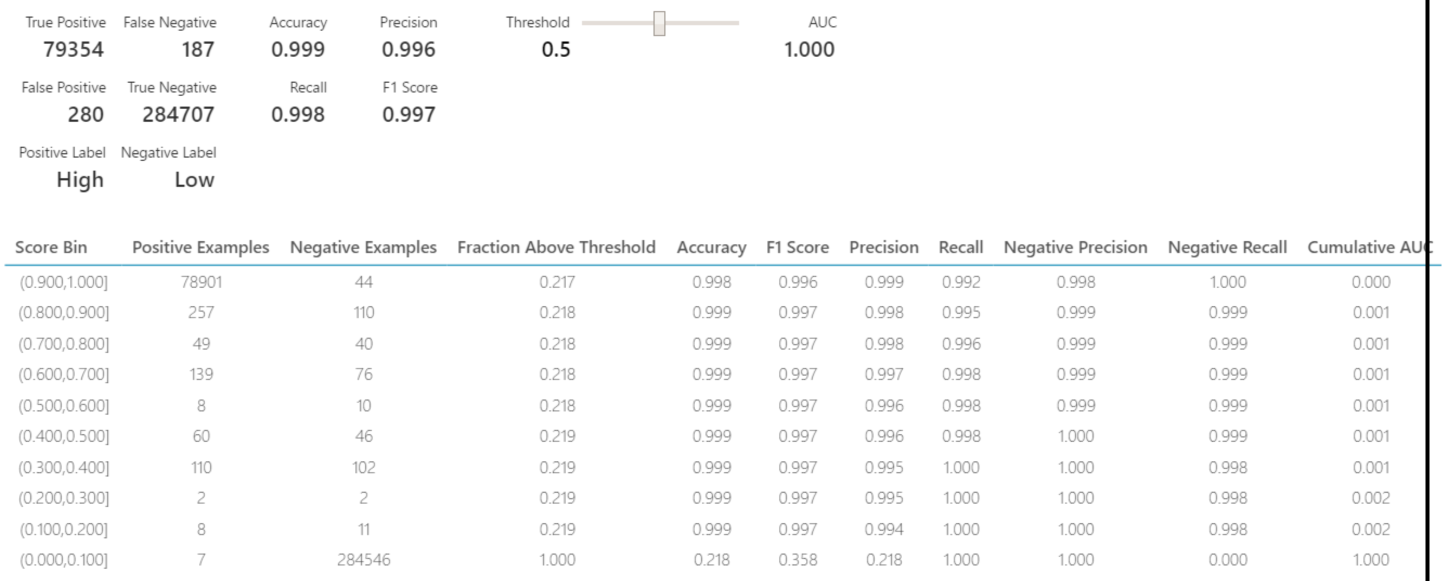


Confusion Matrix and other performance metrics :

The following statistics are shown for our model:

- **Mean Absolute Error** (MAE): The average of absolute errors (an *error* is the difference between the predicted value and the actual value).
- **Root Mean Squared Error** (RMSE): The square root of the average of squared errors of predictions made on the test dataset.
- **Relative Absolute Error**: The average of absolute errors relative to the absolute difference between actual values and the average of all actual values.
- **Relative Squared Error**: The average of squared errors relative to the squared difference between the actual values and the average of all actual values.
- **Coefficient of Determination**: Also known as the **R squared value**, this is a statistical metric indicating how well a model fits the data.

For each of the error statistics, smaller is better. A smaller value indicates that the predictions more closely match the actual values. For **Coefficient of Determination**, the closer its value is to one (1.0), the better the predictions.

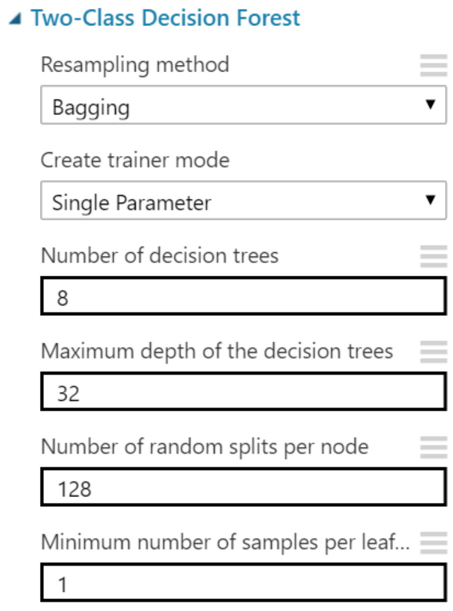


True Positive	False Negative	Accuracy	Precision	Threshold	AUC
79354	187	0.999	0.996	0.5	1.000
False Positive	**True Negative**	**Recall**	**F1 Score**		
280	284707	0.998	0.997		
Positive Label	**Negative Label**				
High	Low				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AU
(0.900,1.000]	78901	44	0.217	0.998	0.996	0.999	0.992	0.998	1.000	0.000
(0.800,0.900]	257	110	0.218	0.999	0.997	0.998	0.995	0.999	0.999	0.001
(0.700,0.800]	49	40	0.218	0.999	0.997	0.998	0.996	0.999	0.999	0.001
(0.600,0.700]	139	76	0.218	0.999	0.997	0.997	0.998	0.999	0.999	0.001
(0.500,0.600]	8	10	0.218	0.999	0.997	0.996	0.998	0.999	0.999	0.001
(0.400,0.500]	60	46	0.219	0.999	0.997	0.996	0.998	1.000	0.999	0.001
(0.300,0.400]	110	102	0.219	0.999	0.997	0.995	1.000	1.000	0.998	0.001
(0.200,0.300]	2	2	0.219	0.999	0.997	0.995	1.000	1.000	0.998	0.002
(0.100,0.200]	8	11	0.219	0.999	0.997	0.994	1.000	1.000	0.998	0.002
(0.000,0.100]	7	284546	1.000	0.218	0.358	0.218	1.000	1.000	0.000	1.000

1.5.4 Two class Decision Forest

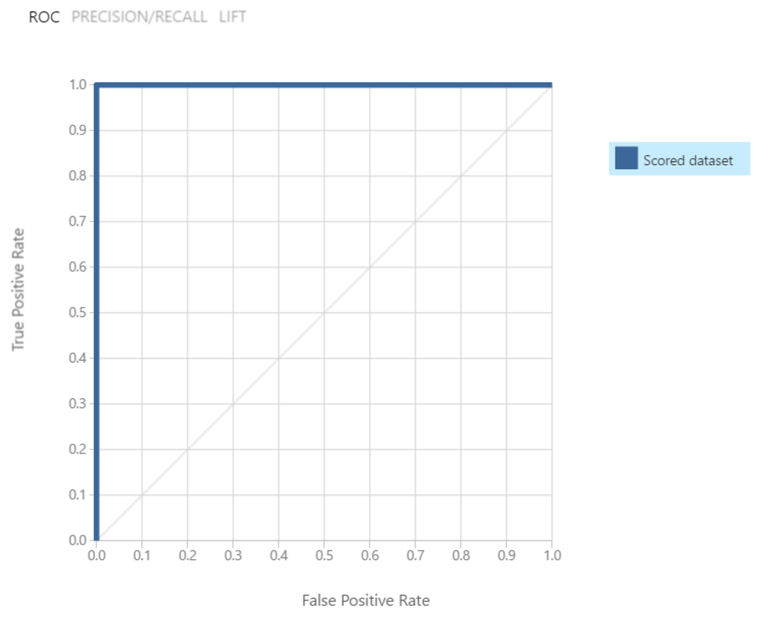
The decision forest algorithm is an ensemble learning method for classification. The algorithm works by building multiple decision trees and then voting on the most popular output class. Voting is a form of aggregation, in which each tree in a classification decision forest outputs a non-normalized frequency histogram of labels. The aggregation process sums these histograms and normalizes the result to get the "probabilities" for each label. The trees that have high prediction confidence will have a greater weight in the final decision of the ensemble.



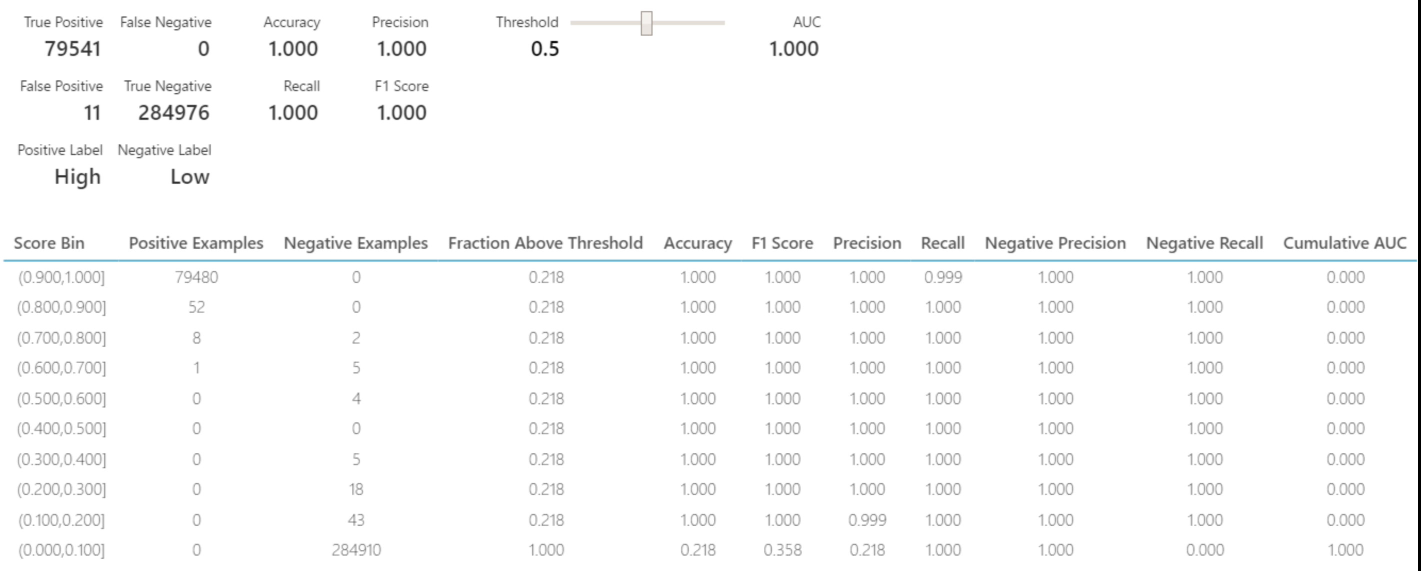
Two-Class Decision Forest

Resampling method	Bagging
Create trainer mode	Single Parameter
Number of decision trees	8
Maximum depth of the decision trees	32
Number of random splits per node	128
Minimum number of samples per leaf...	1

ROC Curve:



Confusion Matrix and other performance metrics :



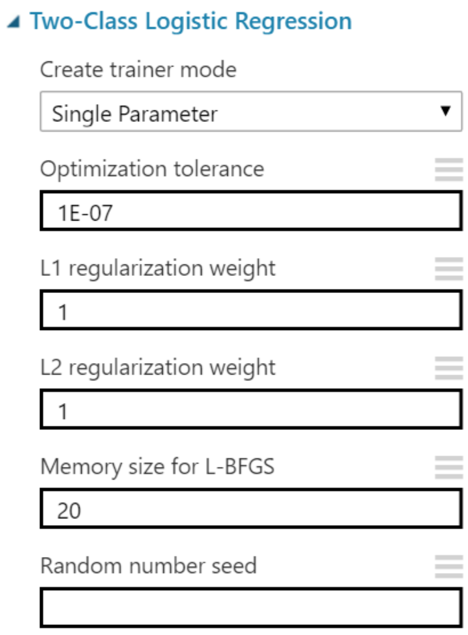
True Positive 79541	False Negative 0	Accuracy 1.000	Precision 1.000	Threshold 0.5	AUC 1.000
False Positive 11	True Negative 284976	Recall 1.000	F1 Score 1.000		
Positive Label High	Negative Label Low				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	79480	0	0.218	1.000	1.000	1.000	0.999	1.000	1.000	0.000
(0.800,0.900]	52	0	0.218	1.000	1.000	1.000	1.000	1.000	1.000	0.000
(0.700,0.800]	8	2	0.218	1.000	1.000	1.000	1.000	1.000	1.000	0.000
(0.600,0.700]	1	5	0.218	1.000	1.000	1.000	1.000	1.000	1.000	0.000
(0.500,0.600]	0	4	0.218	1.000	1.000	1.000	1.000	1.000	1.000	0.000
(0.400,0.500]	0	0	0.218	1.000	1.000	1.000	1.000	1.000	1.000	0.000
(0.300,0.400]	0	5	0.218	1.000	1.000	1.000	1.000	1.000	1.000	0.000
(0.200,0.300]	0	18	0.218	1.000	1.000	1.000	1.000	1.000	1.000	0.000
(0.100,0.200]	0	43	0.218	1.000	1.000	0.999	1.000	1.000	1.000	0.000
(0.000,0.100]	0	284910	1.000	0.218	0.358	0.218	1.000	1.000	0.000	1.000

1.5.5 Two class Logistic Regression

Logistic regression is a well-known method in statistics that is used to predict the probability of an outcome, and is especially popular for classification tasks. The algorithm predicts the probability of occurrence of an event by fitting data to a logistic function.

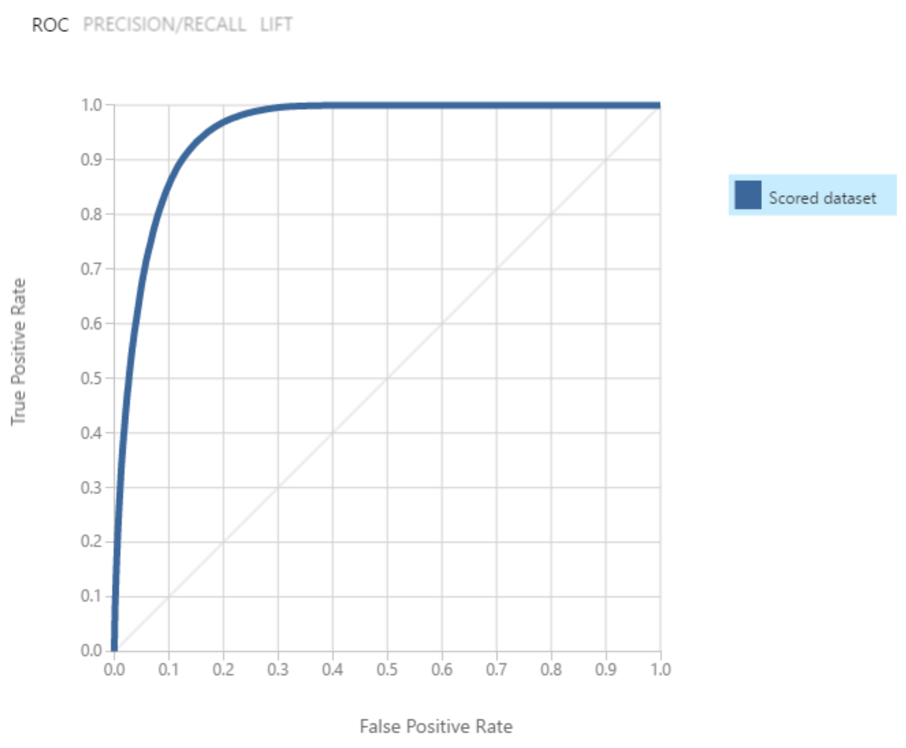
Logistic regression requires numeric variables. Therefore, when you use categorical columns as variable, Azure Machine Learning converts the values to an indicator array internally.



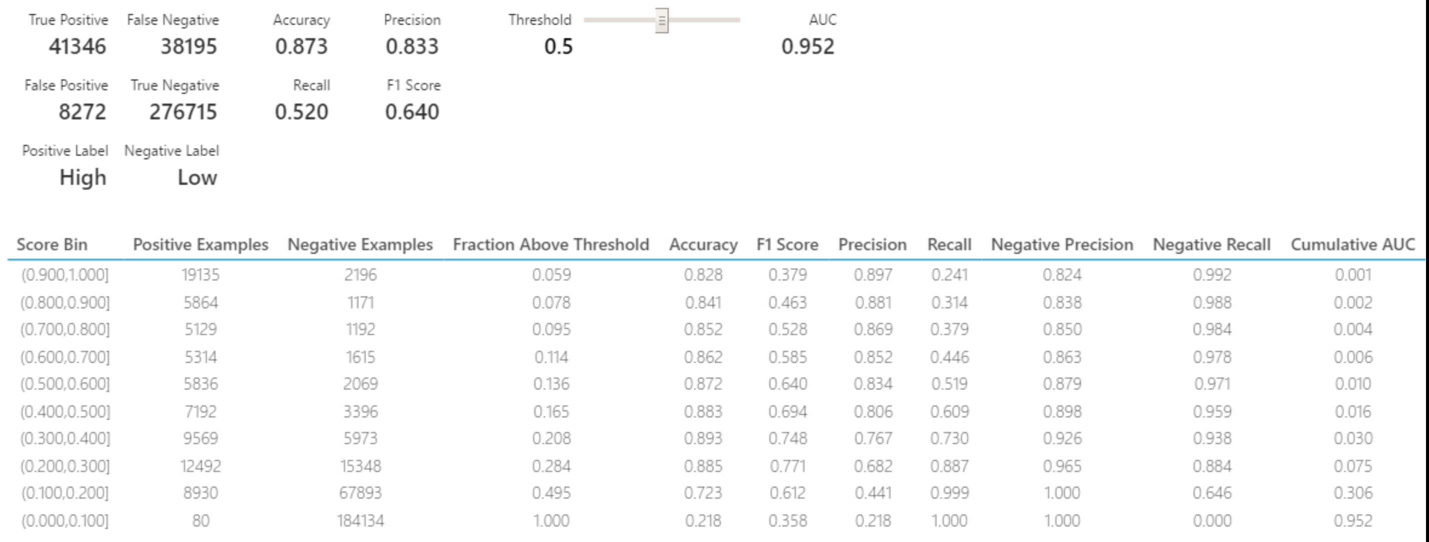
Two-Class Logistic Regression

Create trainer mode	Single Parameter
Optimization tolerance	1E-07
L1 regularization weight	1
L2 regularization weight	1
Memory size for L-BFGS	20
Random number seed	

ROC Curve:



Confusion Matrix and other performance metrics :



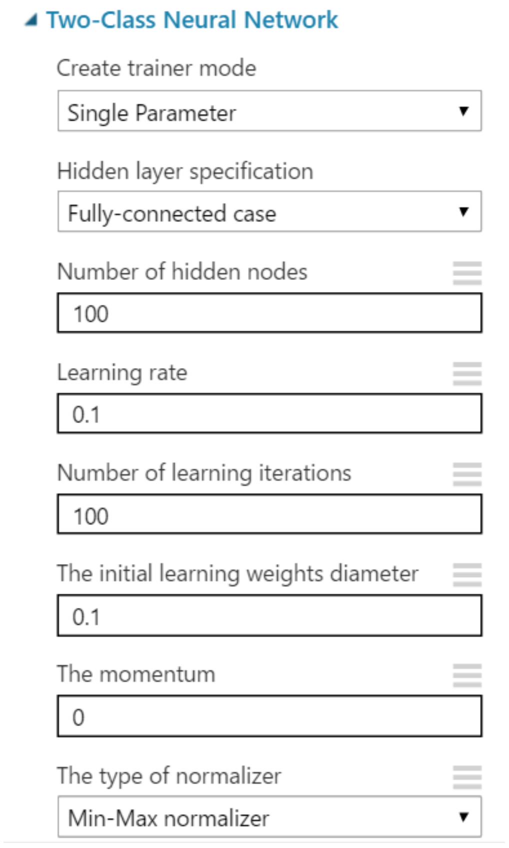
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
41346	38195	0.873	0.833	0.5	0.952
False Positive	True Negative	Recall	F1 Score		
8272	276715	0.520	0.640		
Positive Label	Negative Label				
High	Low				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	19135	2196	0.059	0.828	0.379	0.897	0.241	0.824	0.992	0.001
(0.800,0.900]	5864	1171	0.078	0.841	0.463	0.881	0.314	0.838	0.988	0.002
(0.700,0.800]	5129	1192	0.095	0.852	0.528	0.869	0.379	0.850	0.984	0.004
(0.600,0.700]	5314	1615	0.114	0.862	0.585	0.852	0.446	0.863	0.978	0.006
(0.500,0.600]	5836	2069	0.136	0.872	0.640	0.834	0.519	0.879	0.971	0.010
(0.400,0.500]	7192	3396	0.165	0.883	0.694	0.806	0.609	0.898	0.959	0.016
(0.300,0.400]	9569	5973	0.208	0.893	0.748	0.767	0.730	0.926	0.938	0.030
(0.200,0.300]	12492	15348	0.284	0.885	0.771	0.682	0.887	0.965	0.884	0.075
(0.100,0.200]	8930	67893	0.495	0.723	0.612	0.441	0.999	1.000	0.646	0.306
(0.000,0.100]	80	184134	1.000	0.218	0.358	0.218	1.000	1.000	0.000	0.952

1.5.6 Two class Neural network

A neural network is a set of interconnected layers, in which the inputs lead to outputs by a series of weighted edges and nodes. The weights on the edges are learned when training the neural network on the input data. The direction of the graph proceeds from the inputs through the hidden layer, with all nodes of the graph connected by the weighted edges to nodes in the next layer. Most predictive tasks can be accomplished easily with only one or a few hidden layers.

Recent research has shown that deep neural networks (DNN) can be very effective in complex tasks such as image or speech recognition, in which successive layers are used to model increasing levels of semantic depth. To compute the output of the network for any given input, a value is calculated for each node in the hidden layers and in the output layer. For each node, the value is set by calculating the weighted sum of the values of the nodes in the previous layer and applying an activation function to that weighted sum


Two-Class Neural Network

Create trainer mode
Single Parameter

Hidden layer specification
Fully-connected case

Number of hidden nodes
100

Learning rate
0.1

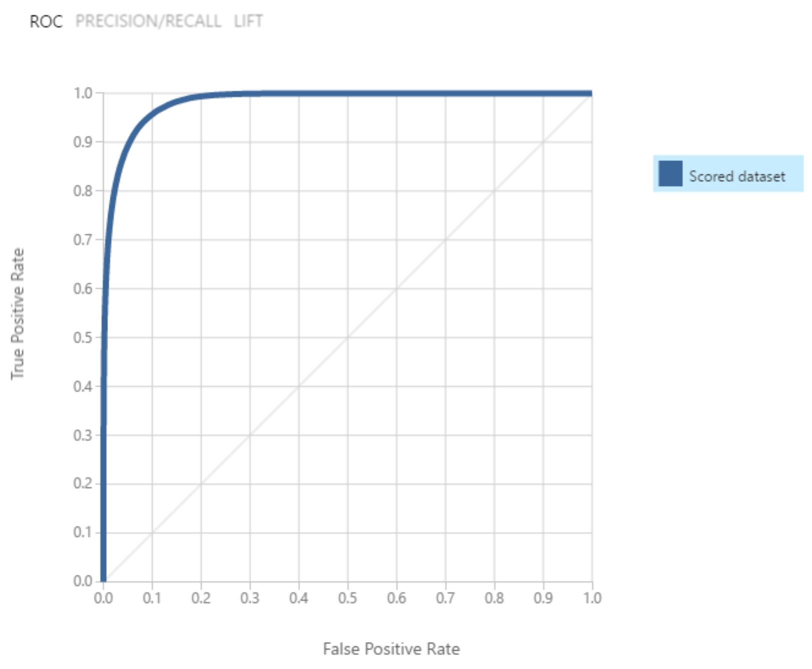
Number of learning iterations
100

The initial learning weights diameter
0.1

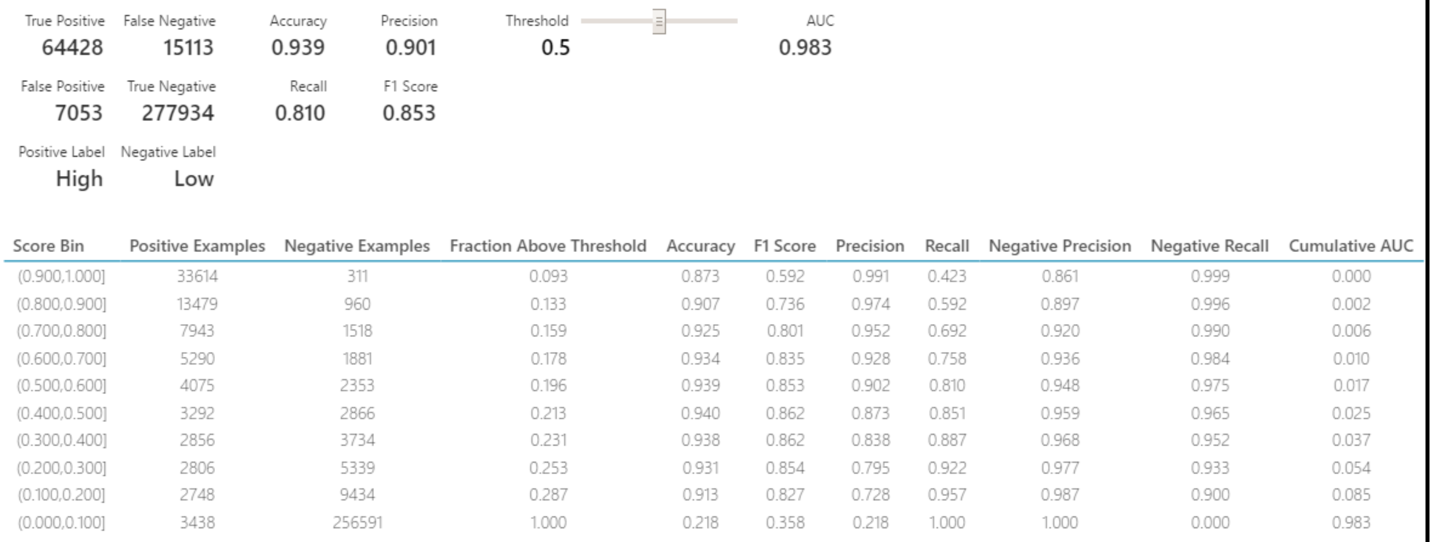
The momentum
0

The type of normalizer
Min-Max normalizer

ROC Curve:



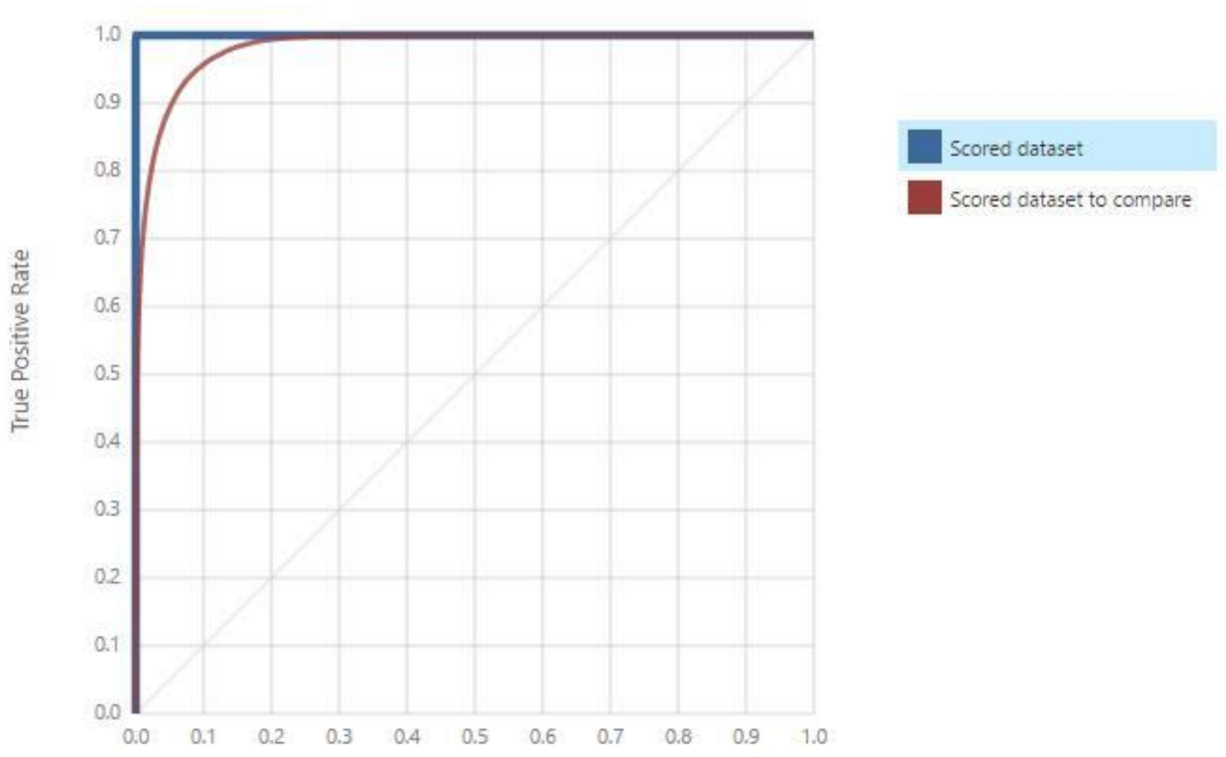
Confusion Matrix and other performance metrics :



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
64428	15113	0.939	0.901	0.5	0.983
False Positive	True Negative	Recall	F1 Score		
7053	277934	0.810	0.853		
Positive Label	Negative Label				
High	Low				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	33614	311	0.093	0.873	0.592	0.991	0.423	0.861	0.999	0.000
(0.800,0.900]	13479	960	0.133	0.907	0.736	0.974	0.592	0.897	0.996	0.002
(0.700,0.800]	7943	1518	0.159	0.925	0.801	0.952	0.692	0.920	0.990	0.006
(0.600,0.700]	5290	1881	0.178	0.934	0.835	0.928	0.758	0.936	0.984	0.010
(0.500,0.600]	4075	2353	0.196	0.939	0.853	0.902	0.810	0.948	0.975	0.017
(0.400,0.500]	3292	2866	0.213	0.940	0.862	0.873	0.851	0.959	0.965	0.025
(0.300,0.400]	2856	3734	0.231	0.938	0.862	0.838	0.887	0.968	0.952	0.037
(0.200,0.300]	2806	5339	0.253	0.931	0.854	0.795	0.922	0.977	0.933	0.054
(0.100,0.200]	2748	9434	0.287	0.913	0.827	0.728	0.957	0.987	0.900	0.085
(0.000,0.100]	3438	256591	1.000	0.218	0.358	0.218	1.000	1.000	0.000	0.983

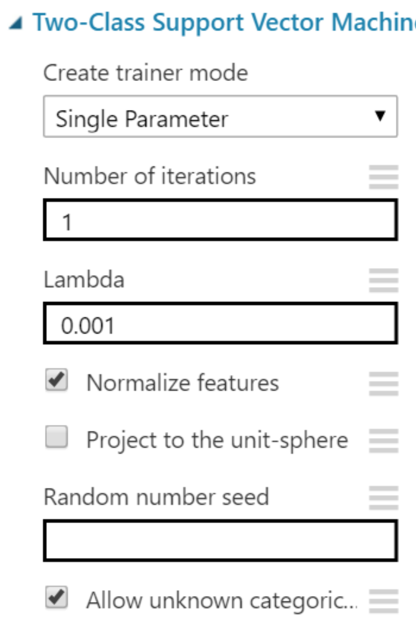
- Comparison between Two class Boosted Decision Tree and Neural Network Classification Algorithm



Blue – Two class Boosted Decision Tree
Red – Neural Network Classification

1.5.7 Two class Support Vector Machine

Given a set of training examples labeled as belonging to one of two classes, the SVM algorithm assigns new examples into one category or the other. The examples are represented as points in space, and they are mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. The feature space that contains the training examples is sometimes called a hyperplane, and it may have many dimensions.



◢ Two-Class Support Vector Machine

Create trainer mode

Single Parameter

Number of iterations

1

Lambda

0.001

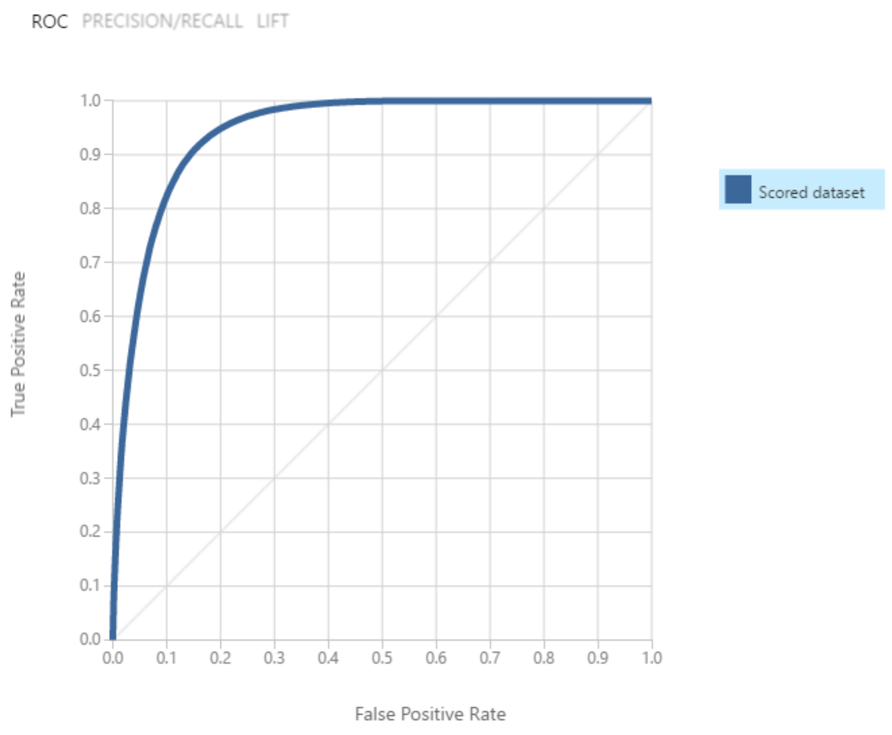
✓ Normalize features

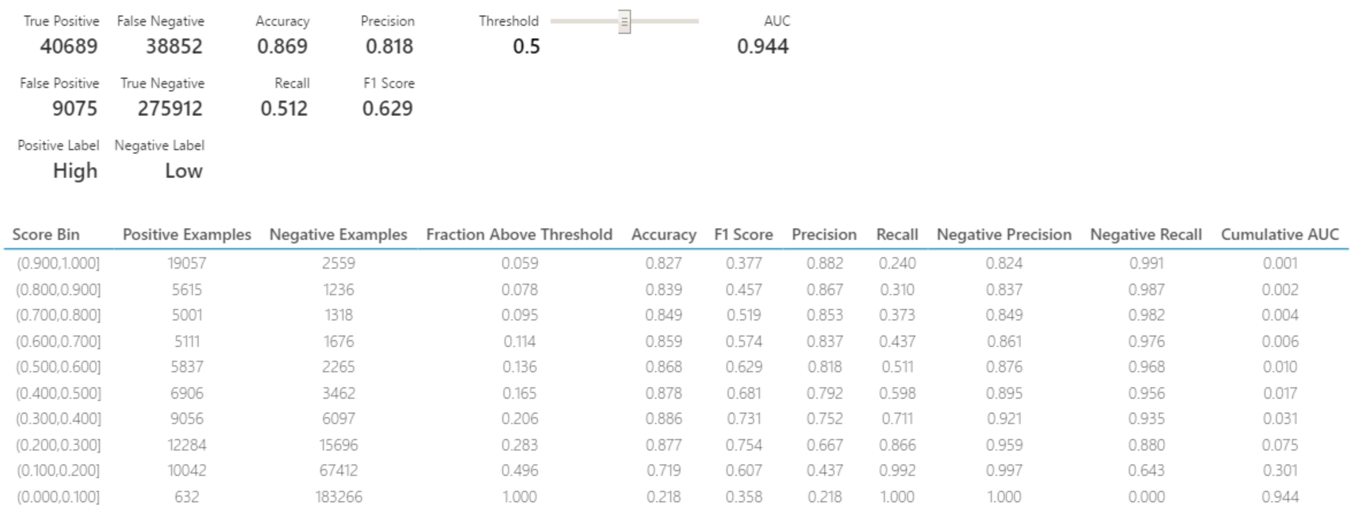
☐ Project to the unit-sphere

Random number seed

✓ Allow unknown categoric...

ROC curve:

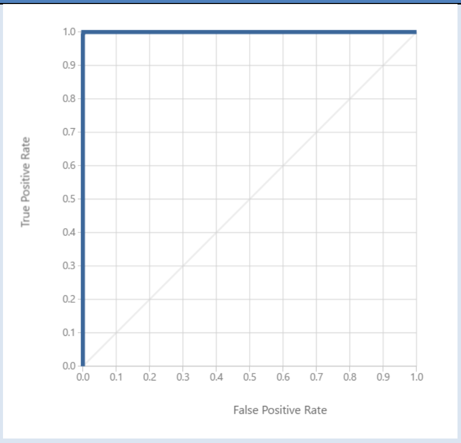
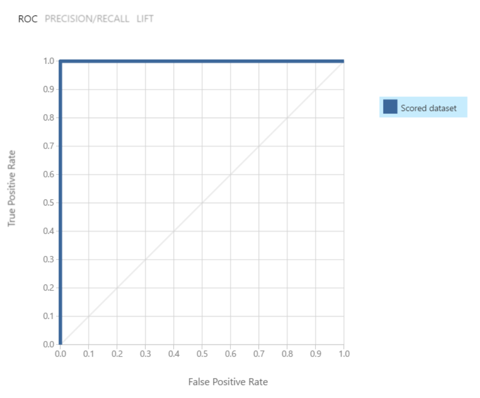


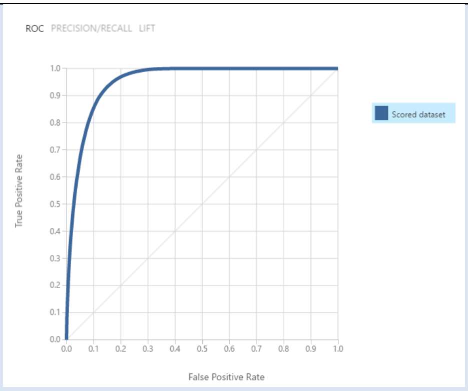
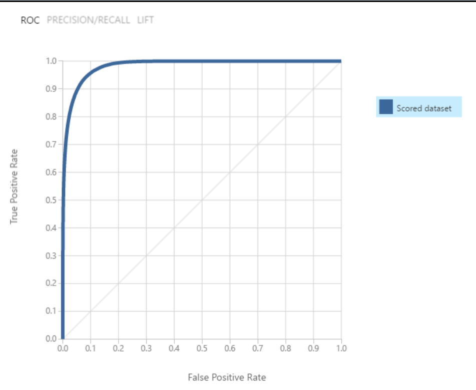
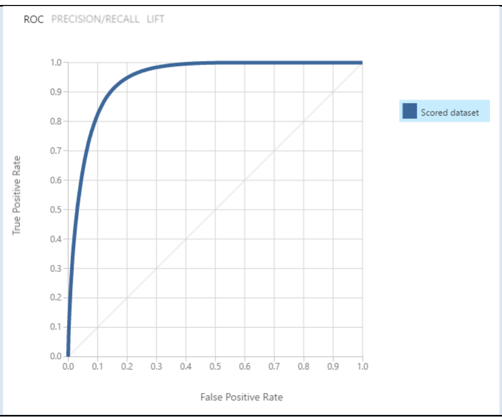


True Positive	False Negative	Accuracy	Precision	Threshold	AUC
40689	38852	0.869	0.818	0.5	0.944
False Positive	**True Negative**	**Recall**	**F1 Score**		
9075	275912	0.512	0.629		
Positive Label	**Negative Label**				
High	Low				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	19057	2559	0.059	0.827	0.377	0.882	0.240	0.824	0.991	0.001
(0.800,0.900]	5615	1236	0.078	0.839	0.457	0.867	0.310	0.837	0.987	0.002
(0.700,0.800]	5001	1318	0.095	0.849	0.519	0.853	0.373	0.849	0.982	0.004
(0.600,0.700]	5111	1676	0.114	0.859	0.574	0.837	0.437	0.861	0.976	0.006
(0.500,0.600]	5837	2265	0.136	0.868	0.629	0.818	0.511	0.876	0.968	0.010
(0.400,0.500]	6906	3462	0.165	0.878	0.681	0.792	0.598	0.895	0.956	0.017
(0.300,0.400]	9056	6097	0.206	0.886	0.731	0.752	0.711	0.921	0.935	0.031
(0.200,0.300]	12284	15696	0.283	0.877	0.754	0.667	0.866	0.959	0.880	0.075
(0.100,0.200]	10042	67412	0.496	0.719	0.607	0.437	0.992	0.997	0.643	0.301
(0.000,0.100]	632	183266	1.000	0.218	0.358	0.218	1.000	1.000	0.000	0.944

1.5.8 Classification model comparison

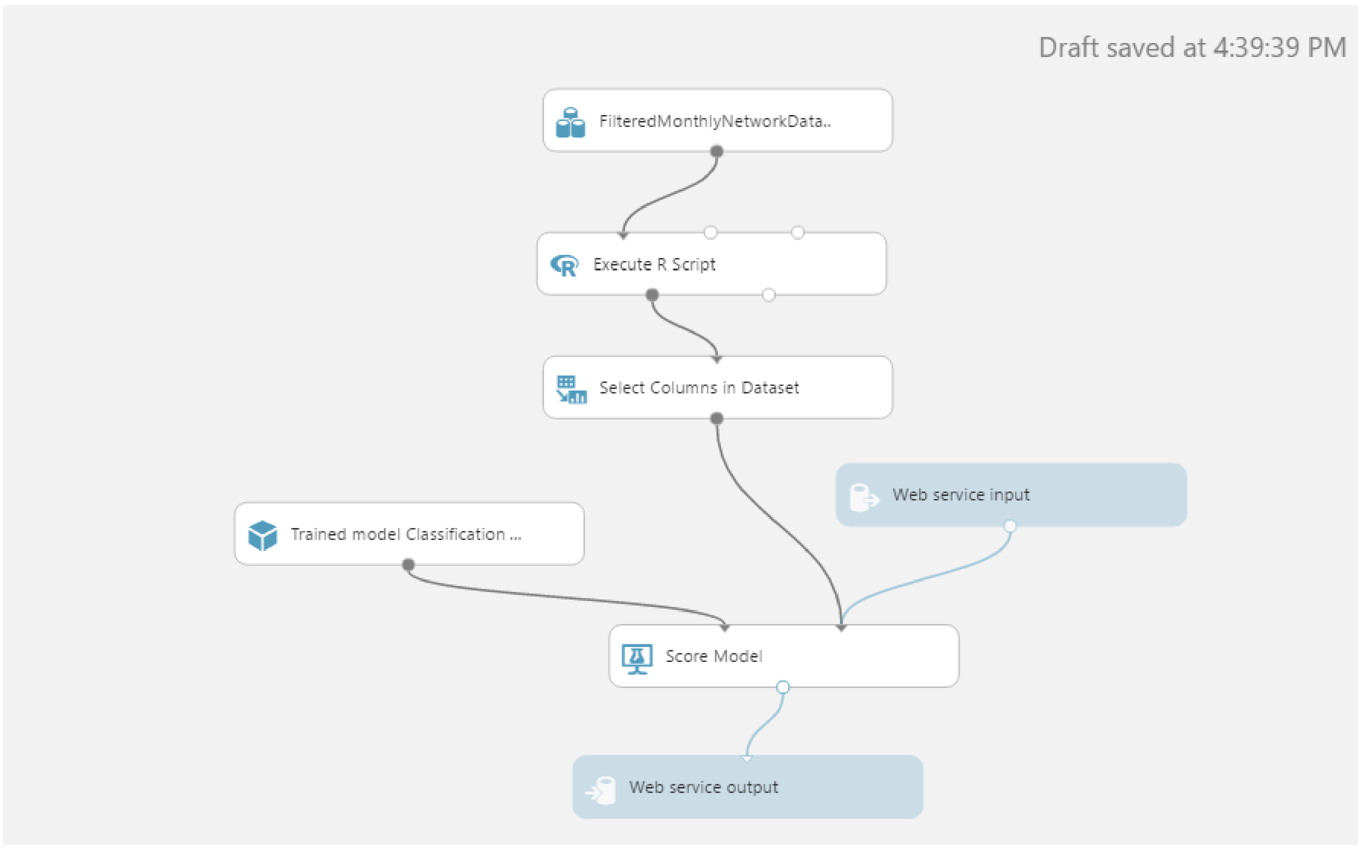
	ROC curve	Performance Metrics
Two class Boosted Decision Tree		Accuracy: 0.999 Precision: 0.996
Two class Decision Forest		Accuracy: 1.000 Precision: 1.000

Two class Logistic Regression		Accuracy: 0.873 Precision: 0.833
Two class Neural network		Accuracy: 0.939 Precision: 0.901
Two class Support Vector Machine		Accuracy: 0.869 Precision: 0.818

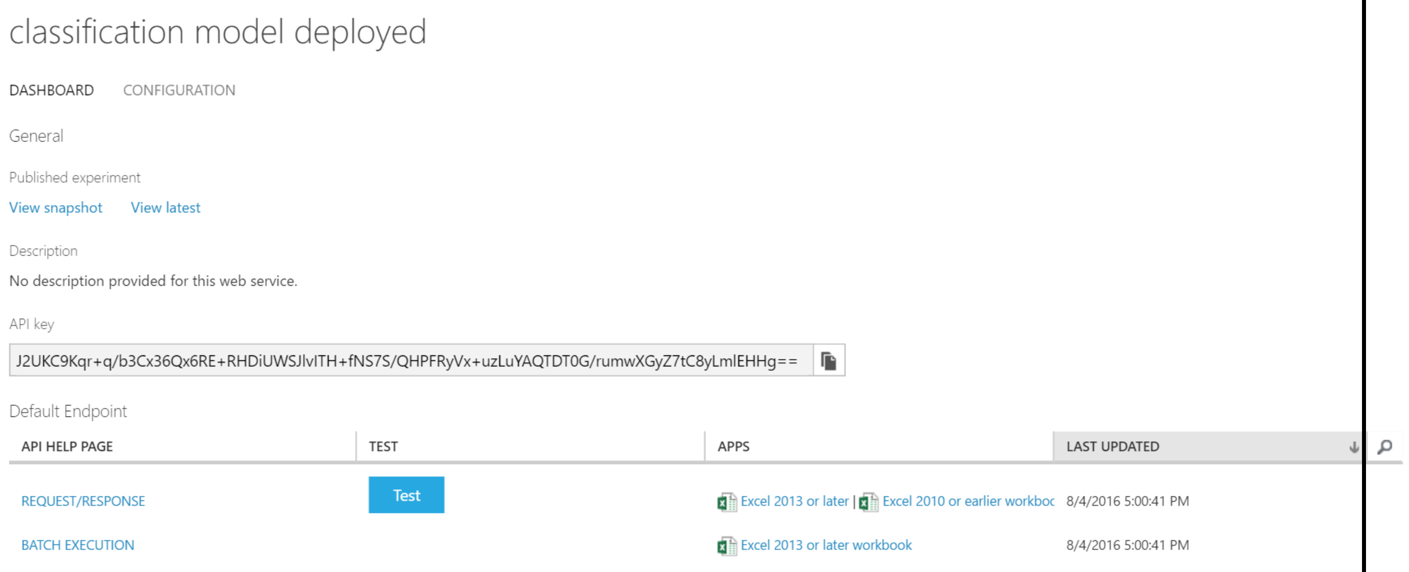
Conclusion: Thus from above table it's clear that the best model among the classification models is Two class Boosted Decision Tree as it has high accuracy rate at 99.9%. Also, the Area Under Curve (AUC) is highest in Two class boosted decision tree and Two class decision forest ROC graphs but as the accuracy and precision is a perfect 1 it might be over fitted and hence we pick two class decision tree.

1.5.9 Web Service

- Once the classification model is ready, we set up **Web Service**.
- The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**).
- Modules that were used for training are removed. Specifically:
    - Two-Class Boosted Decision Tree
    - Train Model
    - Split Data
- Then we added the saved trained model back into the experiment.
- **Web service input** and **Web service output** modules are added.


Draft saved at 4:39:39 PM
FilteredMonthlyNetworkData..
Execute R Script
Select Columns in Dataset
Web service input
Trained model Classification ...
Score Model
Web service output

- Now run the model and publish the web service.



classification model deployed

DASHBOARD CONFIGURATION

General

Published experiment

View snapshot View latest

Description

No description provided for this web service.

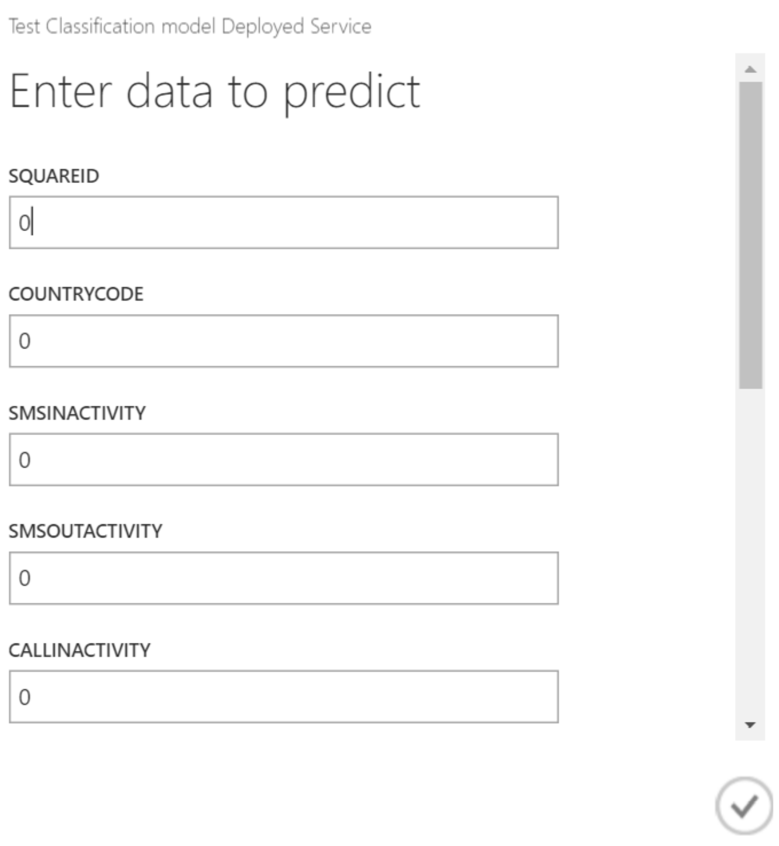
API key

J2UKC9Kqr+q/b3Cx36Qx6RE+RHDiUWSJlvITH+fNS7S/QHPFRyVx+uzLuYAQTDT0G/rumwXGyZ7tC8yLmlEHHg==

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test	Excel 2013 or later | Excel 2010 or earlier workboc	8/4/2016 5:00:41 PM
BATCH EXECUTION		Excel 2013 or later workbook	8/4/2016 5:00:41 PM

- On running the web service, we get the following form which can be used to invoke the web service and do prediction.



Test Classification model Deployed Service

Enter data to predict

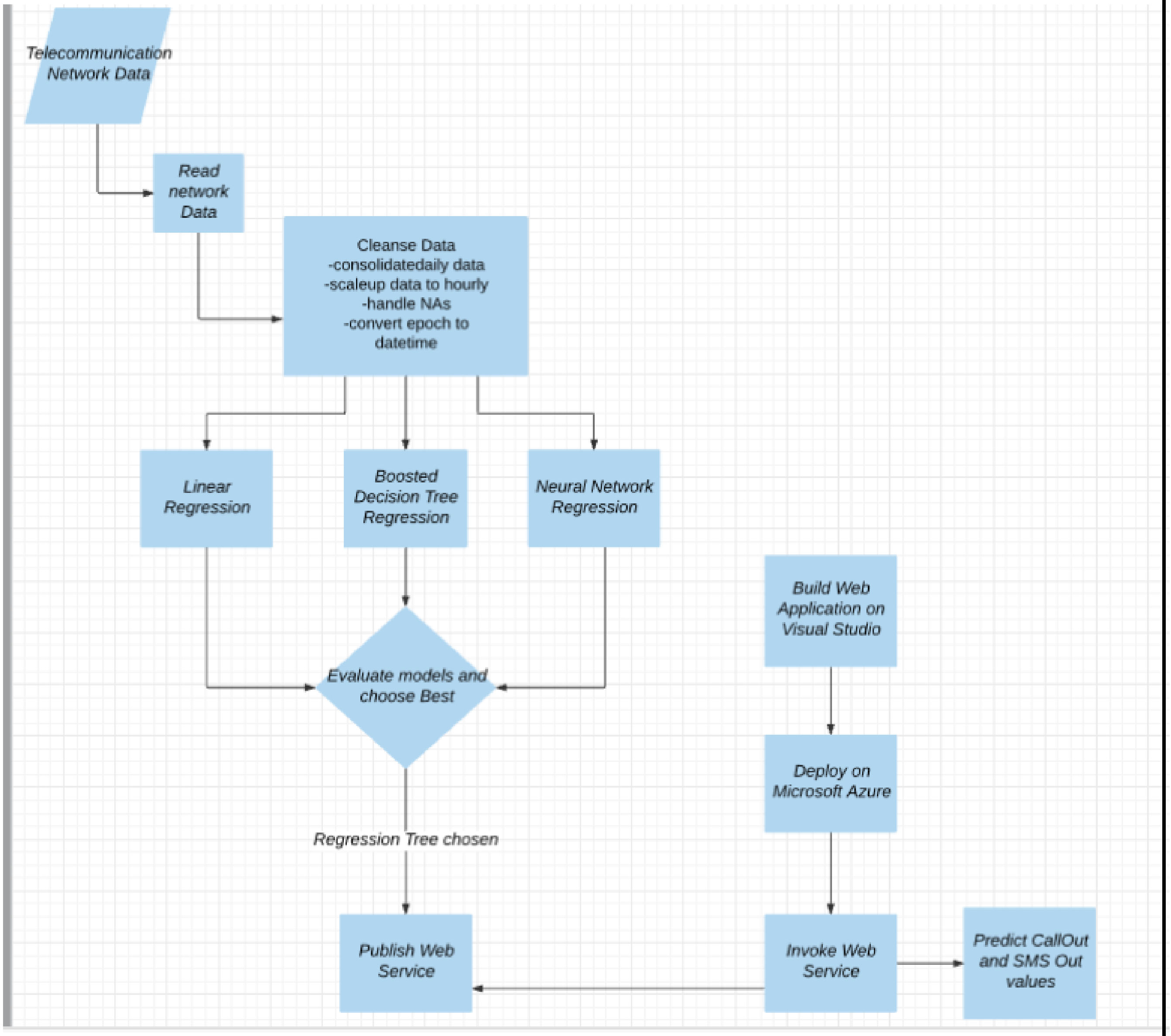
SQUAREID	0
COUNTRYCODE	0
SMSINACTIVITY	0
SMSOUTACTIVITY	0
CALLINACTIVITY	0

1.6 Regression Models

We build regression model to predict the call out activity but this can be scaled to predict internet activity and sms out activity in the future.

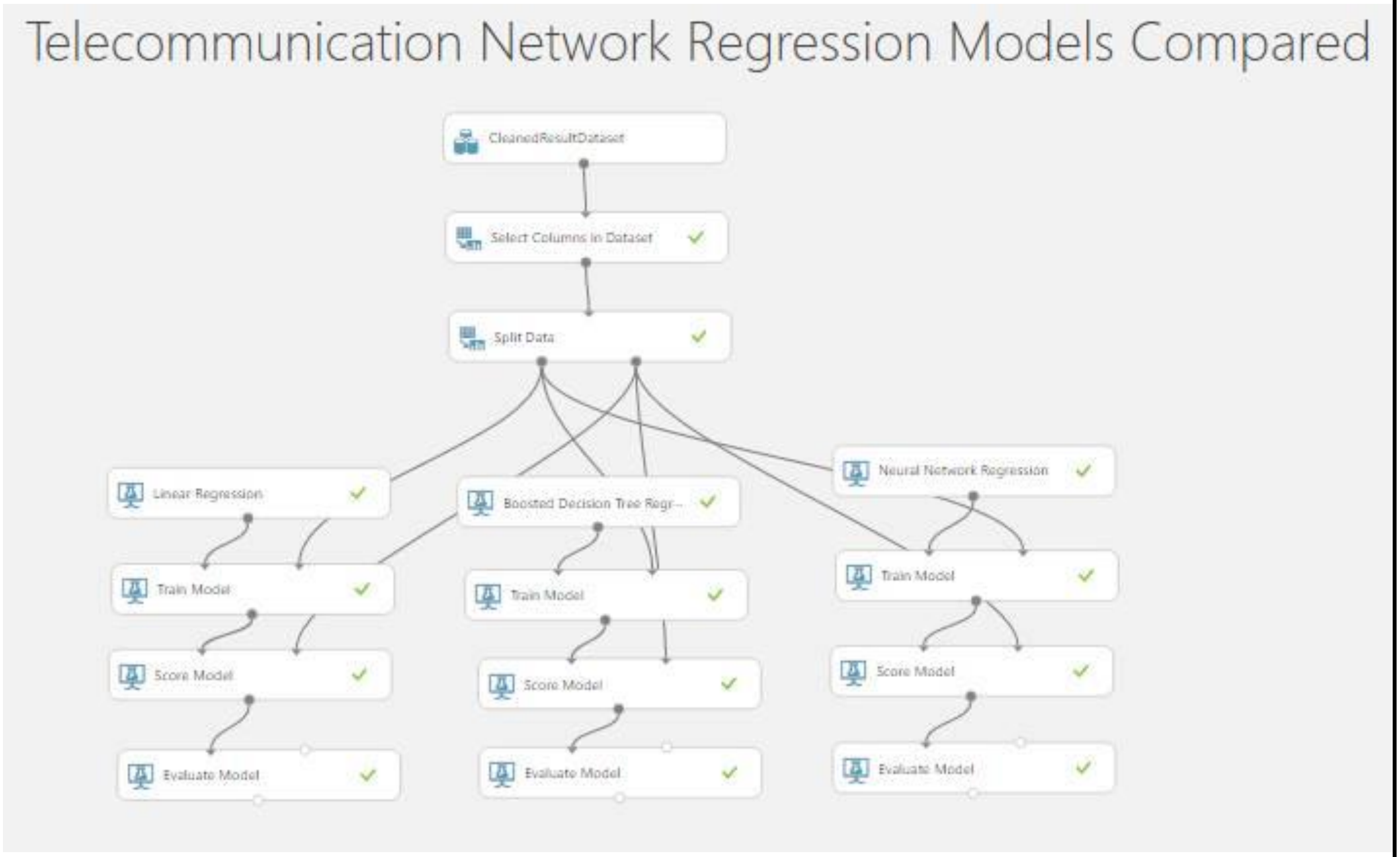
1.6.1Overall Design

- Read the telecommunication network data.
- Cleanse the data as explained in the data cleansing section.
- Implement Linear Regression, Boosted Decision Tree Regression and Neural Network regression. Parameters used in each model is elaborated in the further module below.
- Compare the models through the coefficient of determination, root mean square error, mean absolute error parameters and choose the best model.
- Publish the best classification model created for this data as a web service.
- Build a web application using visual studio and deploy it on Microsoft Azure.
- This web application is used as an interface to invoke the web service expose through Azure.
- Call Out activity can be predicted using this model.


Telecommunication Network Data
Read network Data
Cleanse Data -consolidatedaily data -scaleup data to hourly -handle NAs -convert epoch to datetime
Linear Regression
Boosted Decision Tree Regression
Neural Network Regression
Evaluate models and choose Best
Regression Tree chosen
Publish Web Service
Build Web Application on Visual Studio
Deploy on Microsoft Azure
Invoke Web Service
Predict CallOut and SMS Out values

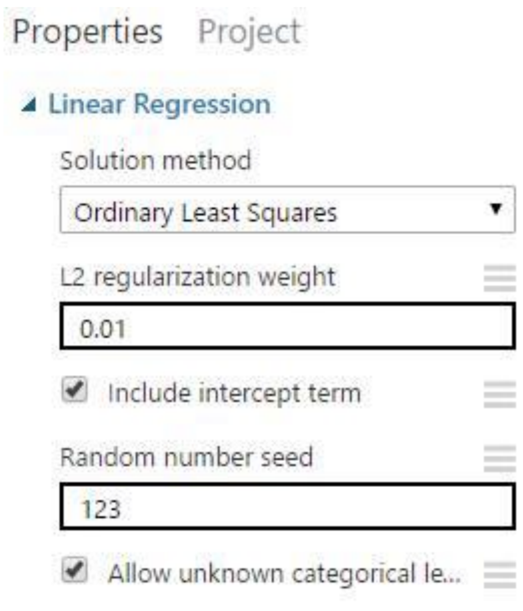
1.6.2 Azure models

Below is the telecommunication network regression model created for predicting call out activity.



1.6.3 Linear Regression

use the Linear Regression module to create a linear regression model using either the ordinary least squares method or the online gradient descent method. Regression is a machine learning used to predict a numeric outcome. Linear regression attempts to establish a linear relationship between independent variables and an outcome variable, or dependent variable, that is also numeric.


Properties Project

Linear Regression

Solution method: Ordinary Least Squares

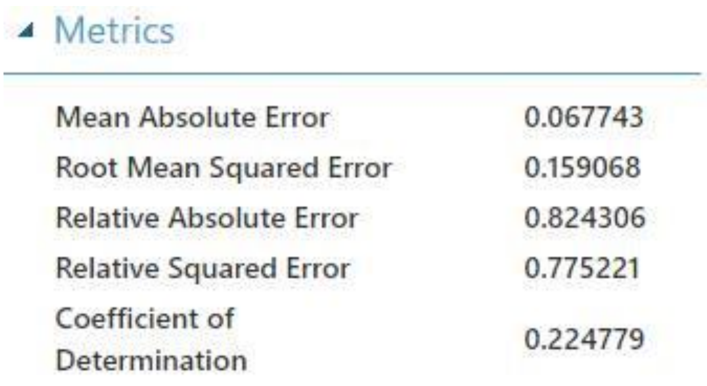
L2 regularization weight: 0.01

✓ Include intercept term

Random number seed: 123

✓ Allow unknown categorical le...

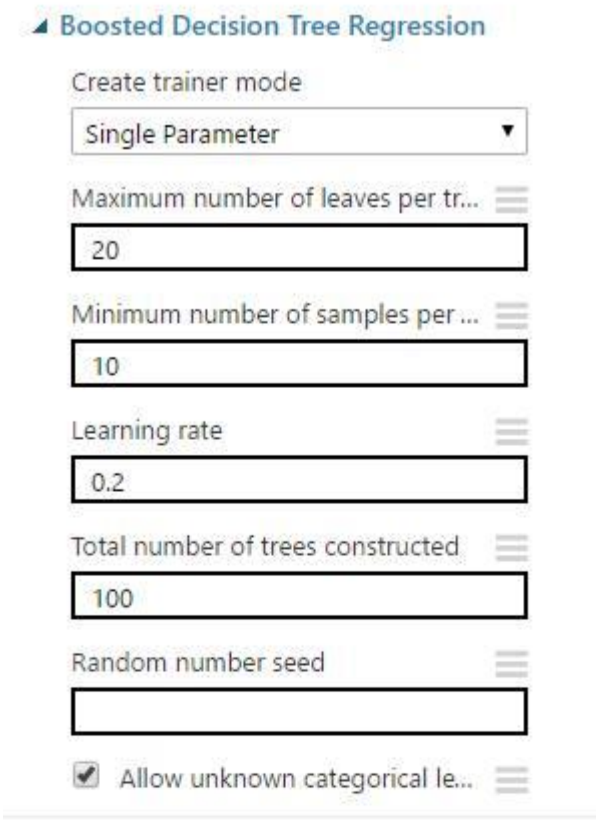
Performance metrics :


Metrics

Mean Absolute Error	0.067743
Root Mean Squared Error	0.159068
Relative Absolute Error	0.824306
Relative Squared Error	0.775221
Coefficient of Determination	0.224779

1.6.4 Boosted Decision Tree Regression

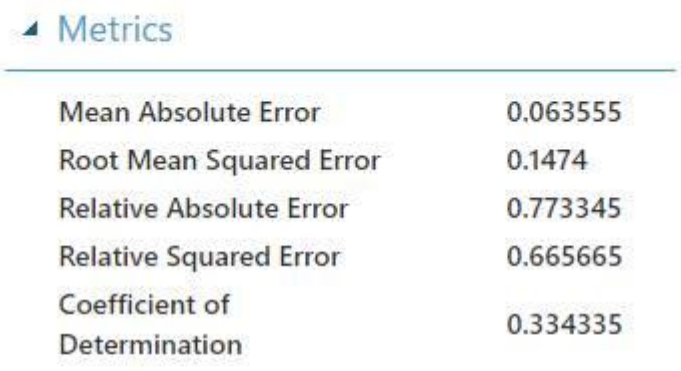
use the Boosted Decision Tree Regression module to create an ensemble of regression trees using boosting. Boosting means that each tree is dependent on prior trees, and learns by fitting the residual of the trees that preceded it. Thus, boosting in a decision tree ensemble tends to improve accuracy with some small risk of less coverage. This regression method is a supervised learning method, and therefore requires a labeled dataset. The label column must contain numerical values.



◢ Boosted Decision Tree Regression

Create trainer mode	Single Parameter
Maximum number of leaves per tr...	20
Minimum number of samples per ...	10
Learning rate	0.2
Total number of trees constructed	100
Random number seed	
✓ Allow unknown categorical le...	

Performance metrics :



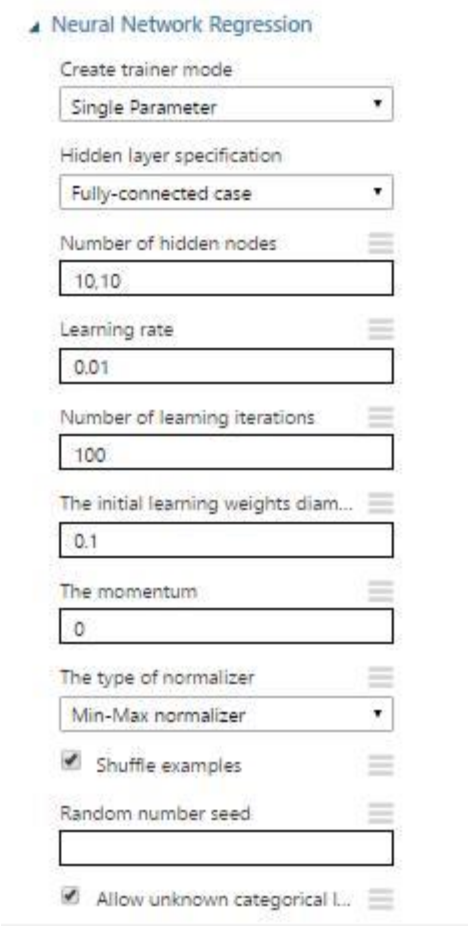
◢ Metrics

Mean Absolute Error	0.063555
Root Mean Squared Error	0.1474
Relative Absolute Error	0.773345
Relative Squared Error	0.665665
Coefficient of Determination	0.334335

1.6.5 Neural Network Regression

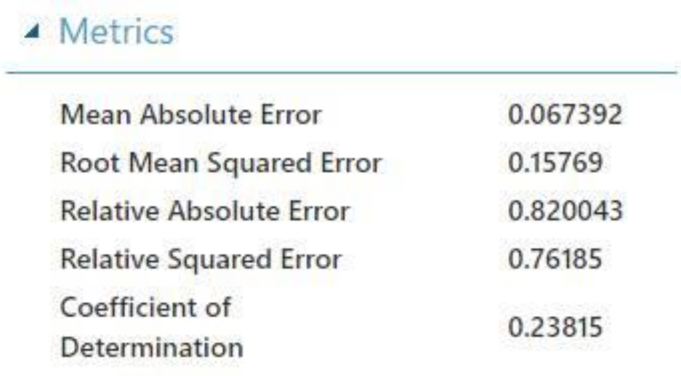
Logistic regression is a well-known method in statistics that is used to predict the probability of an outcome, and is especially popular for classification tasks. The algorithm predicts the probability of occurrence of an event by fitting data to a logistic function.

Logistic regression requires numeric variables. Therefore, when you use categorical columns as variable, Azure Machine Learning converts the values to an indicator array internally.


Neural Network Regression

Create trainer mode	Single Parameter
Hidden layer specification	Fully-connected case
Number of hidden nodes	10,10
Learning rate	0.01
Number of learning iterations	100
The initial learning weights diam...	0.1
The momentum	0
The type of normalizer	Min-Max normalizer
Shuffle examples	✓
Random number seed	
Allow unknown categorical l...	✓

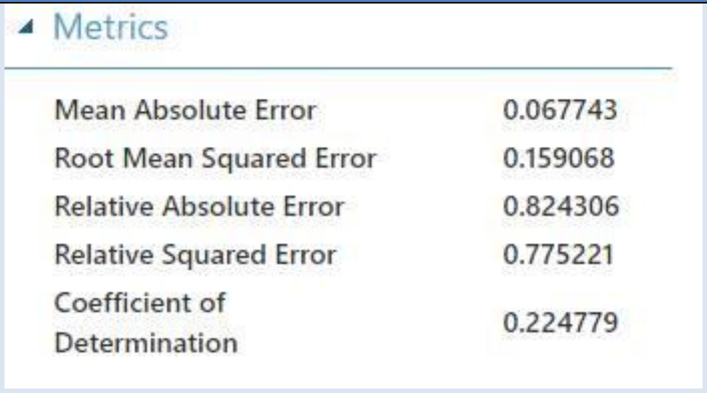
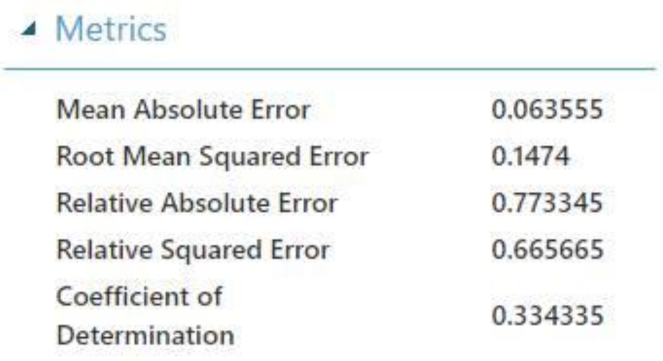
Performance metrics :

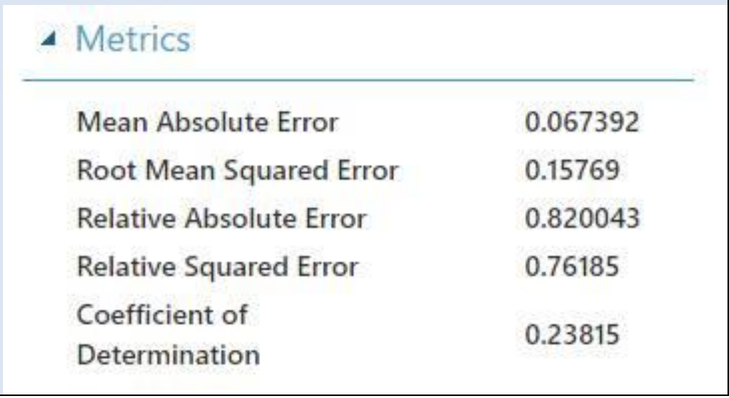


Metrics

Mean Absolute Error	0.067392
Root Mean Squared Error	0.15769
Relative Absolute Error	0.820043
Relative Squared Error	0.76185
Coefficient of Determination	0.23815

1.6.6 Regression model comparison

	ROC curve
Linear Regression	 Metrics Mean Absolute Error 0.067743 Root Mean Squared Error 0.159068 Relative Absolute Error 0.824306 Relative Squared Error 0.775221 Coefficient of Determination 0.224779
Boosted Decision Tree Regression	 Metrics Mean Absolute Error 0.063555 Root Mean Squared Error 0.1474 Relative Absolute Error 0.773345 Relative Squared Error 0.665665 Coefficient of Determination 0.334335

Neural Network Regression	

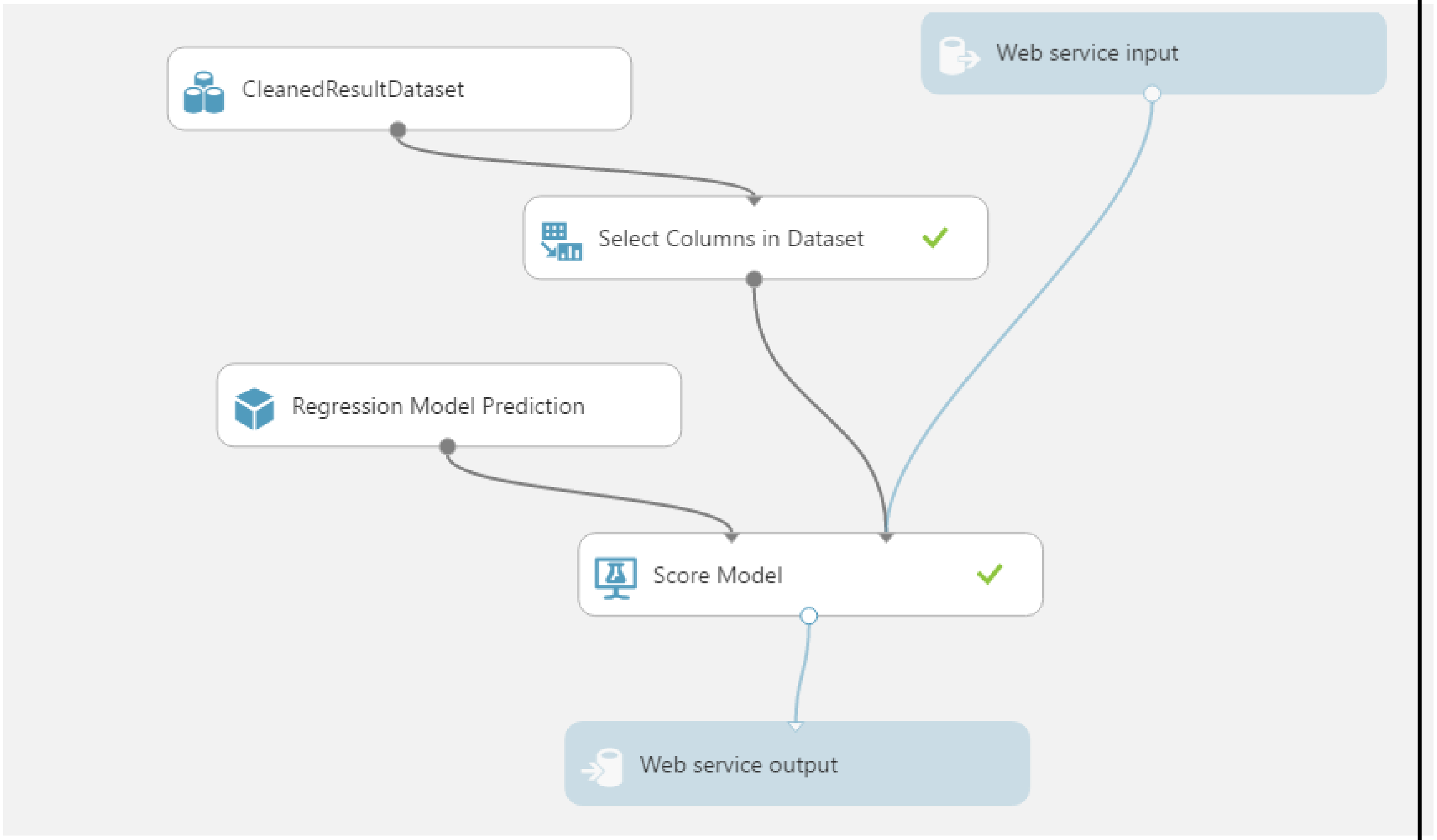
Metrics

Mean Absolute Error	0.067392
Root Mean Squared Error	0.15769
Relative Absolute Error	0.820043
Relative Squared Error	0.76185
Coefficient of Determination	0.23815

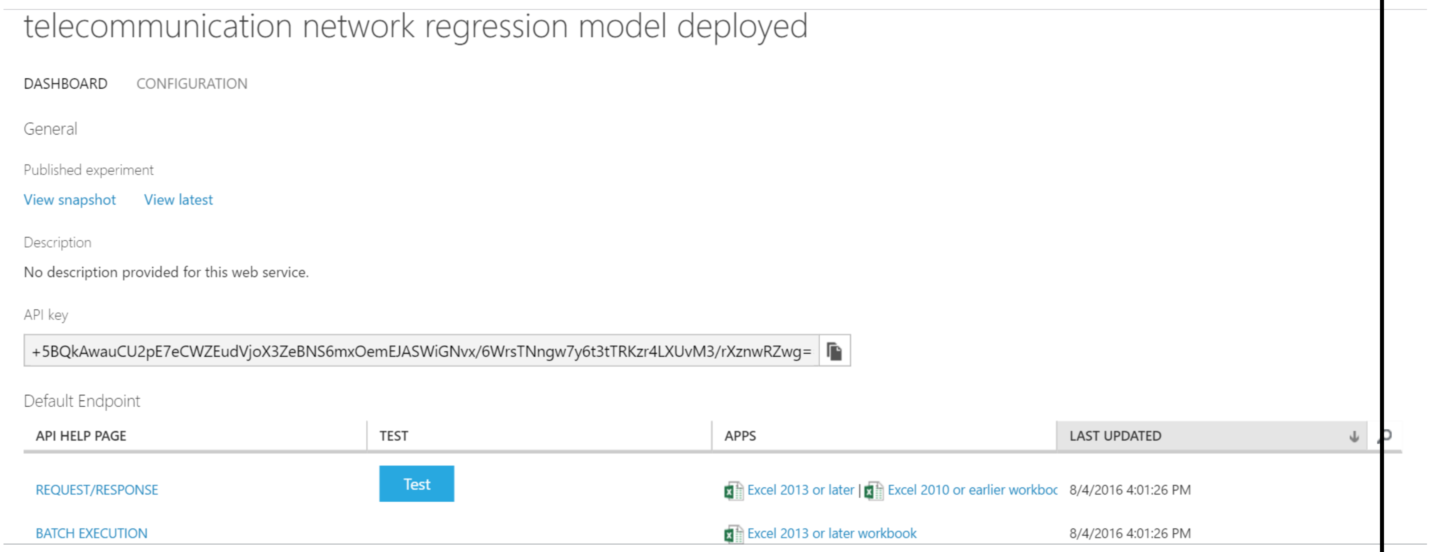
Conclusion: Thus from above table it's clear that the best model among the regression models is Boosted Decision Tree Regression as it has the highest coefficient of determination i.e. 33%. Also, mean absolute error and root mean squared error is least of all.

1.6.7 Web Service

- Once the regression model is ready, we set up **Web Service**.
- The model we trained is saved as a single **Trained Model** module into the module palette to the left of the experiment canvas (you can find it under **Trained Models**).
- Modules that were used for training are removed. Specifically:
    - Two-Class Boosted Decision Tree
    - Train Model
    - Split Data
- Then we added the saved trained model back into the experiment.
- **Web service input** and **Web service output** modules are added.



➢ Now run the model and publish the web service.


telecommunication network regression model deployed

DASHBOARD CONFIGURATION

General

Published experiment

View snapshot View latest

Description

No description provided for this web service.

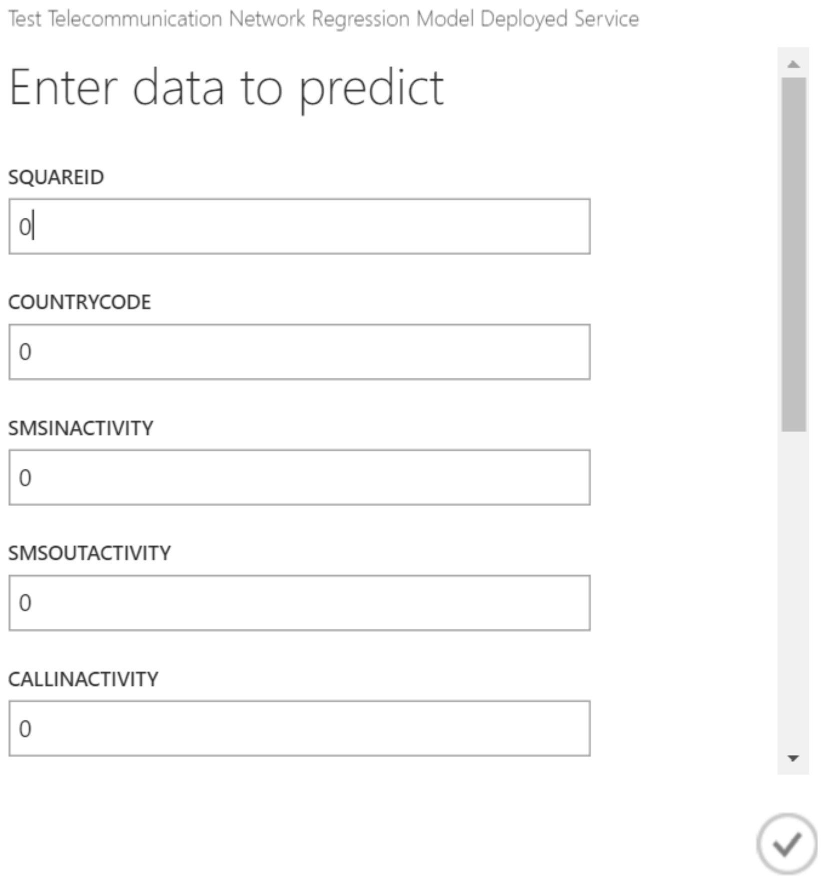
API key

+5BQkAwauCU2pE7eCWZEudVjoX3ZeBNS6mxOemEJASWiGNvx/6WrsTNngw7y6t3tTRKzr4LXUvM3/rXznwRZwg=

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test	Excel 2013 or later | Excel 2010 or earlier workboc	8/4/2016 4:01:26 PM
BATCH EXECUTION		Excel 2013 or later workbook	8/4/2016 4:01:26 PM

- On running the web service, we get the following form which can be used to invoke the web service and do prediction.



Test Telecommunication Network Regression Model Deployed Service

Enter data to predict

SQUAREID	0
COUNTRYCODE	0
SMSINACTIVITY	0
SMSOUTACTIVITY	0
CALLINACTIVITY	0

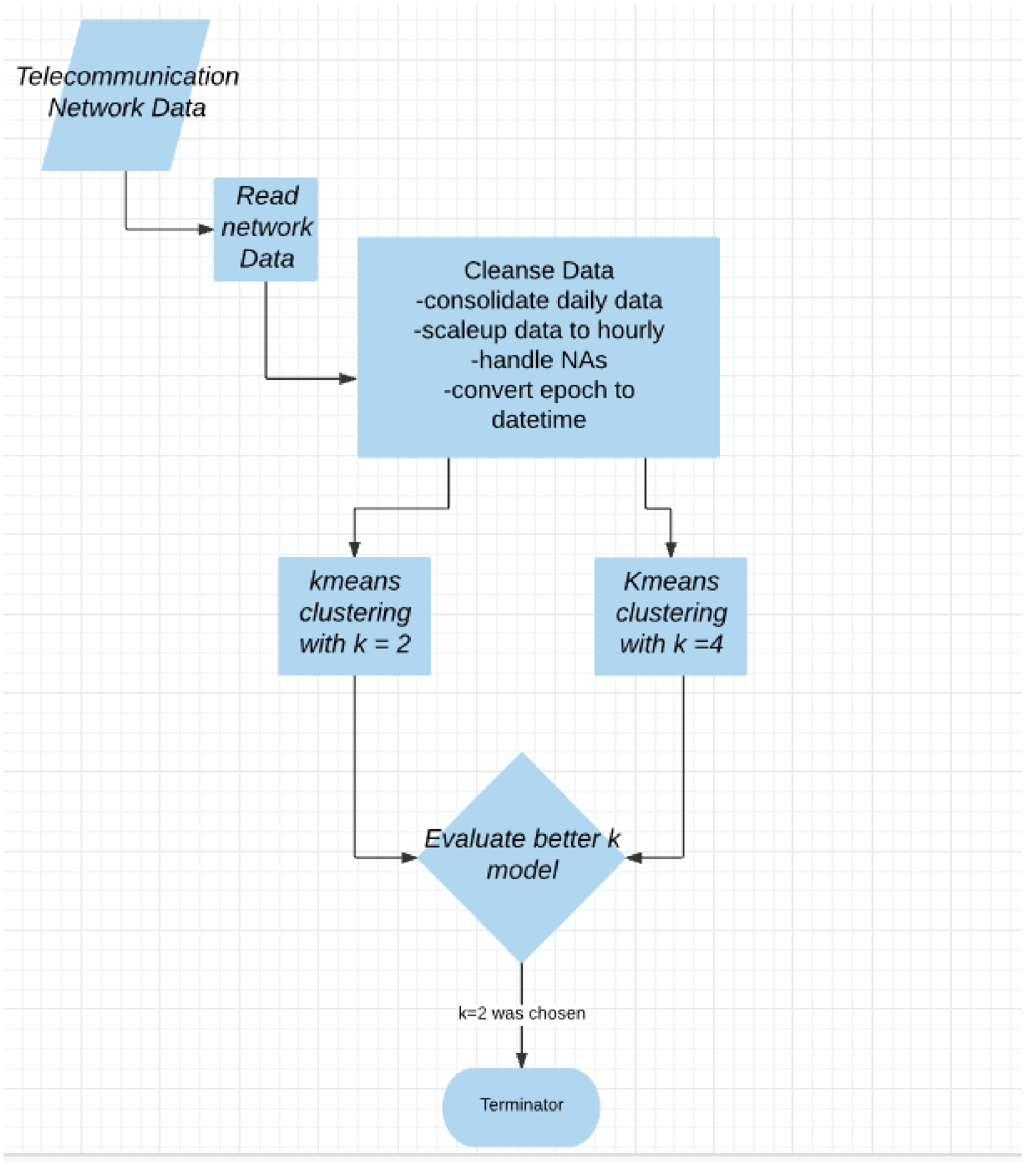
1.7 Clustering

Unsupervised learning methods are known as methods deal with finding patterns or classes of unlabeled data objects. In other word in such datasets there is no continuous or discrete responds associated with observations.

•Methods include of clustering (partitioning and hierarchical), P.C.A, association rules mining and Kernel density clustering.

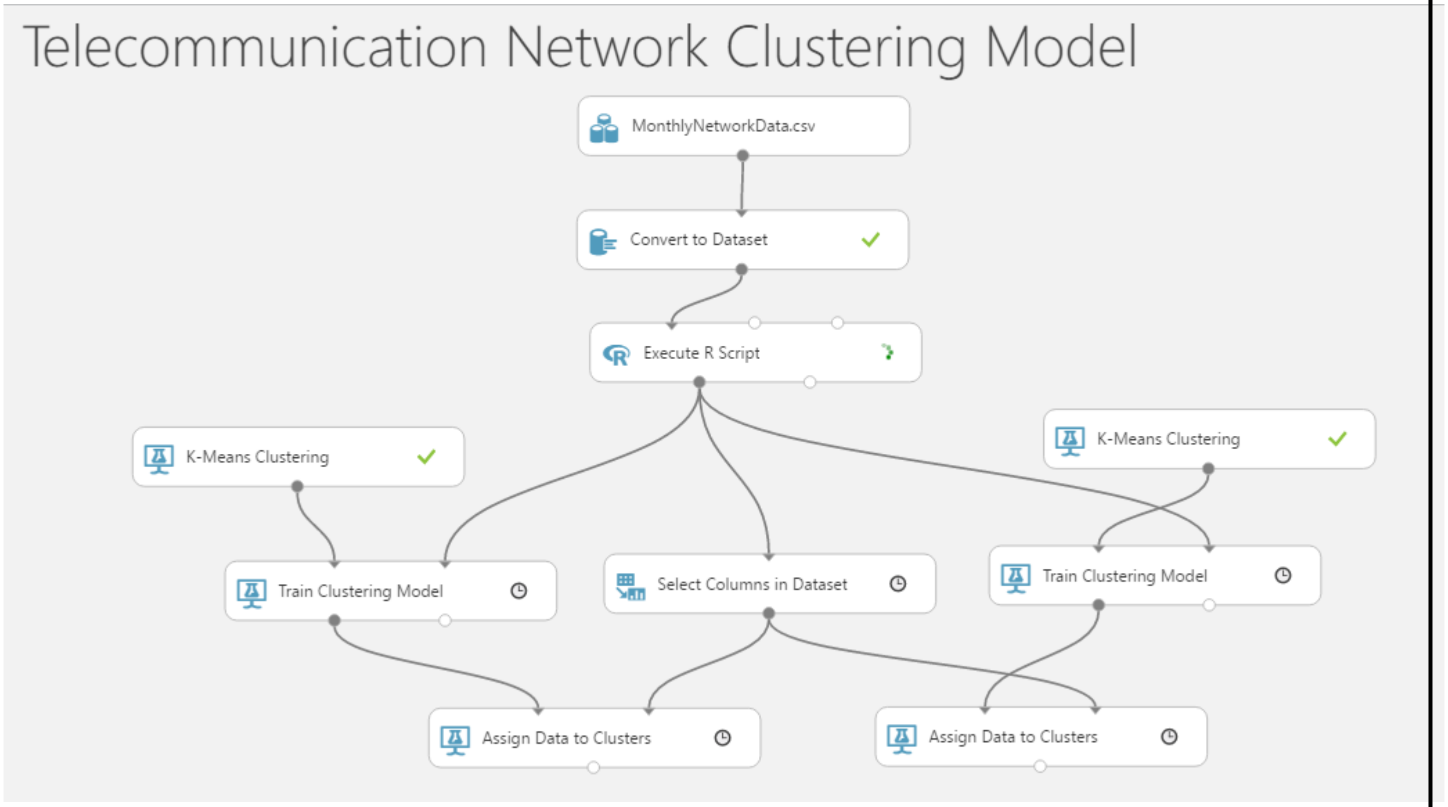
1.7.1Overall Design

- ❖ Read the telecommunication network data.
- ❖ Cleanse the data as explained in the data cleansing section.
- ❖ Implement K-means clustering. We found the best K value using x-means algorithm in Rapid miner tool.


Telecommunication Network Data
Read network Data
Cleanse Data -consolidate daily data -scaleup data to hourly -handle NAs -convert epoch to datetime
kmeans clustering with k = 2
Kmeans clustering with k =4
Evaluate better k model
k=2 was chosen
Terminator

1.7.2 Azure models

Below is the telecommunication network clustering model created.

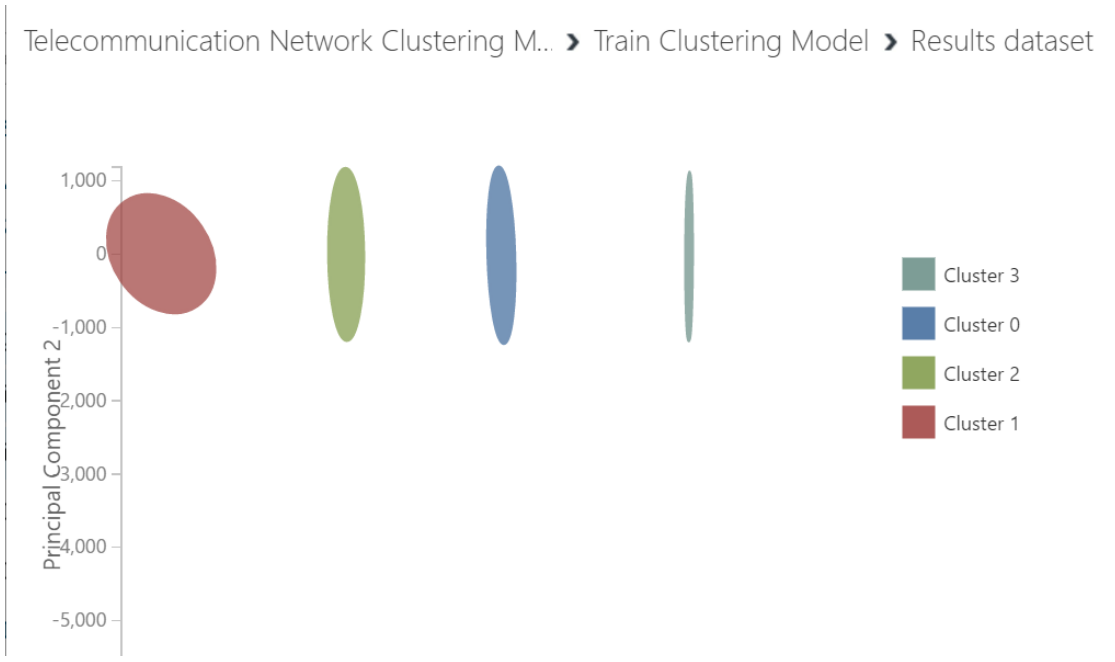


1.7.3 K-means Clustering

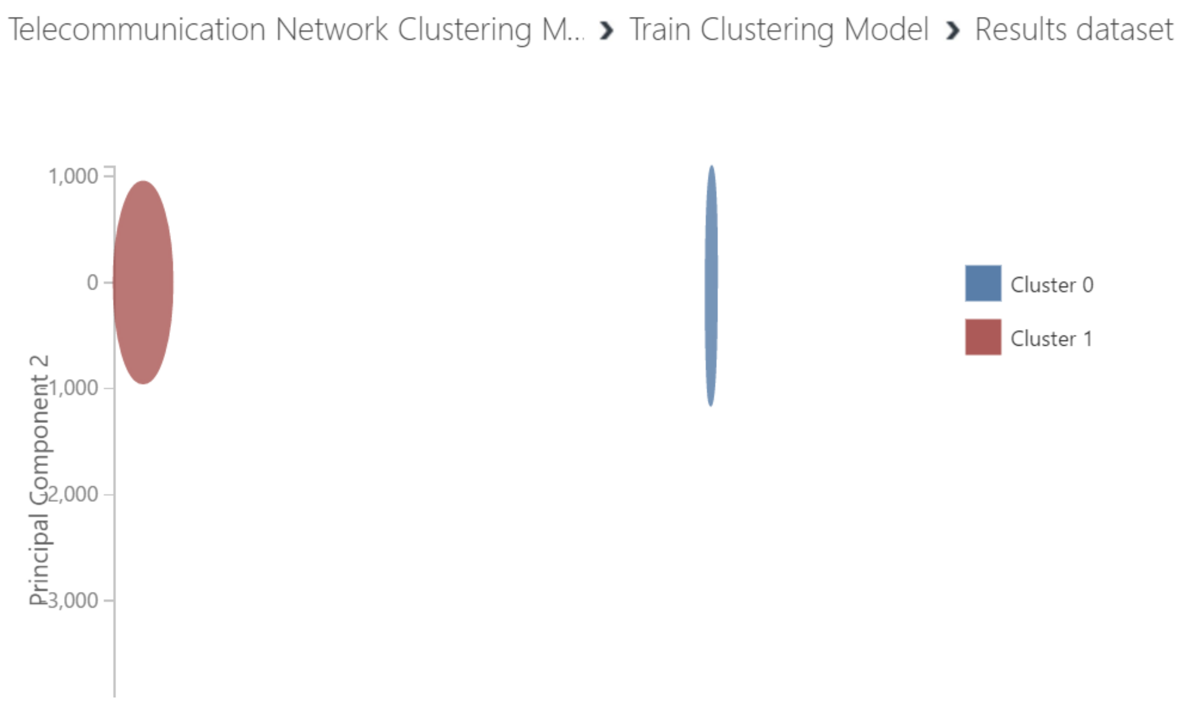
K-means is one of the simplest and the best known unsupervised learning algorithms, and can be used for a variety of machine learning tasks, such as detecting abnormal data, clustering of text documents, and analysis of a dataset prior to using other classification or regression methods.

Because the K-means algorithm is an unsupervised learning method, the data you use to train the model does not need a label column. In other words, you don’t need to know any of the cluster categories in advance; the algorithm will find possible categories based solely on the data.

K-mean clustering with k = 2



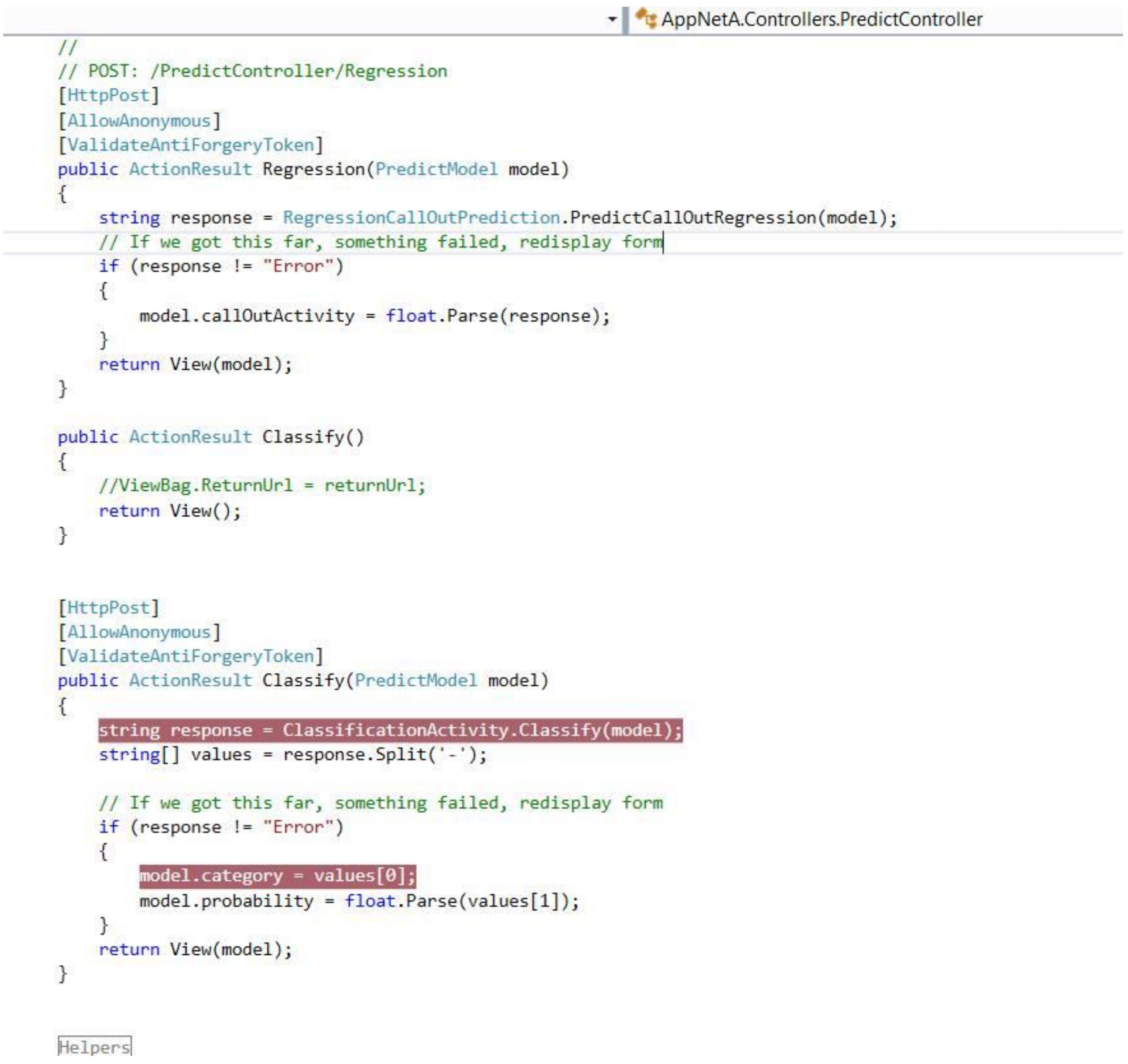
K-mean clustering with k = 2



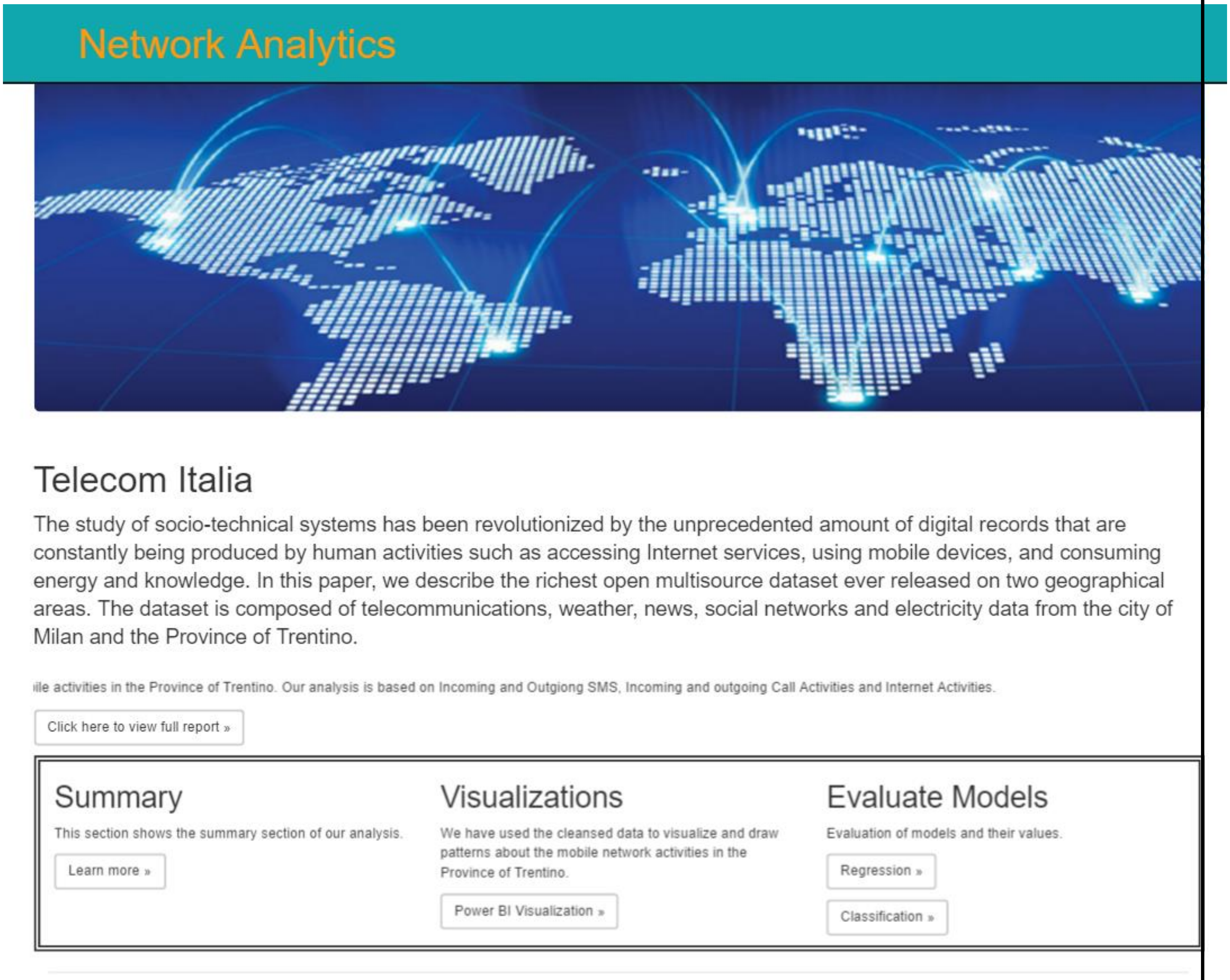
1.8 Web Application

Link: *http://networktelecomprediction.azurewebsites.net/*

We created a MVC ASP.net application on visual studio and then deployed the web application on Microsoft Azure.



```
//
// POST: /PredictController/Regression
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Regression(PredictModel model)
{
    string response = RegressionCallOutPrediction.PredictCallOutRegression(model);
    // If we got this far, something failed, redisplay form
    if (response != "Error")
    {
        model.callOutActivity = float.Parse(response);
    }
    return View(model);
}

public ActionResult Classify()
{
    //ViewBag.ReturnUrl = returnUrl;
    return View();
}


[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Classify(PredictModel model)
{
    string response = ClassificationActivity.Classify(model);
    string[] values = response.Split('-');

    // If we got this far, something failed, redisplay form
    if (response != "Error")
    {
        model.category = values[0];
        model.probability = float.Parse(values[1]);
    }
    return View(model);
}


Helpers
```



Network Analytics

Telecom Italia

The study of socio-technical systems has been revolutionized by the unprecedented amount of digital records that are constantly being produced by human activities such as accessing Internet services, using mobile devices, and consuming energy and knowledge. In this paper, we describe the richest open multisource dataset ever released on two geographical areas. The dataset is composed of telecommunications, weather, news, social networks and electricity data from the city of Milan and the Province of Trentino.

ile activities in the Province of Trentino. Our analysis is based on Incoming and Outgiong SMS, Incoming and outgoing Call Activities and Internet Activities.

Click here to view full report »

Summary

This section shows the summary section of our analysis.

Learn more »

Visualizations

We have used the cleansed data to visualize and draw patterns about the mobile network activities in the Province of Trentino.

Power BI Visualization »

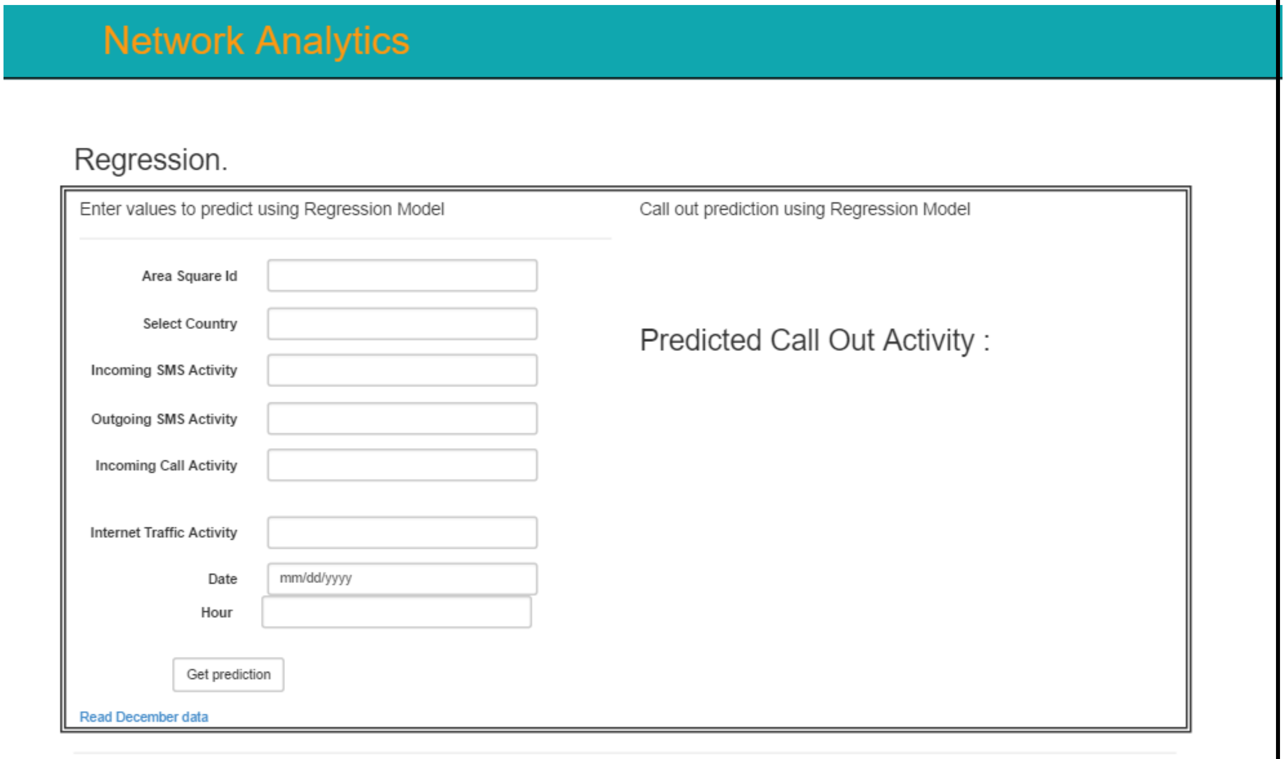
Evaluate Models

Evaluation of models and their values.

Regression »

Classification »

- Predict Call out activity through the regression model.



Network Analytics

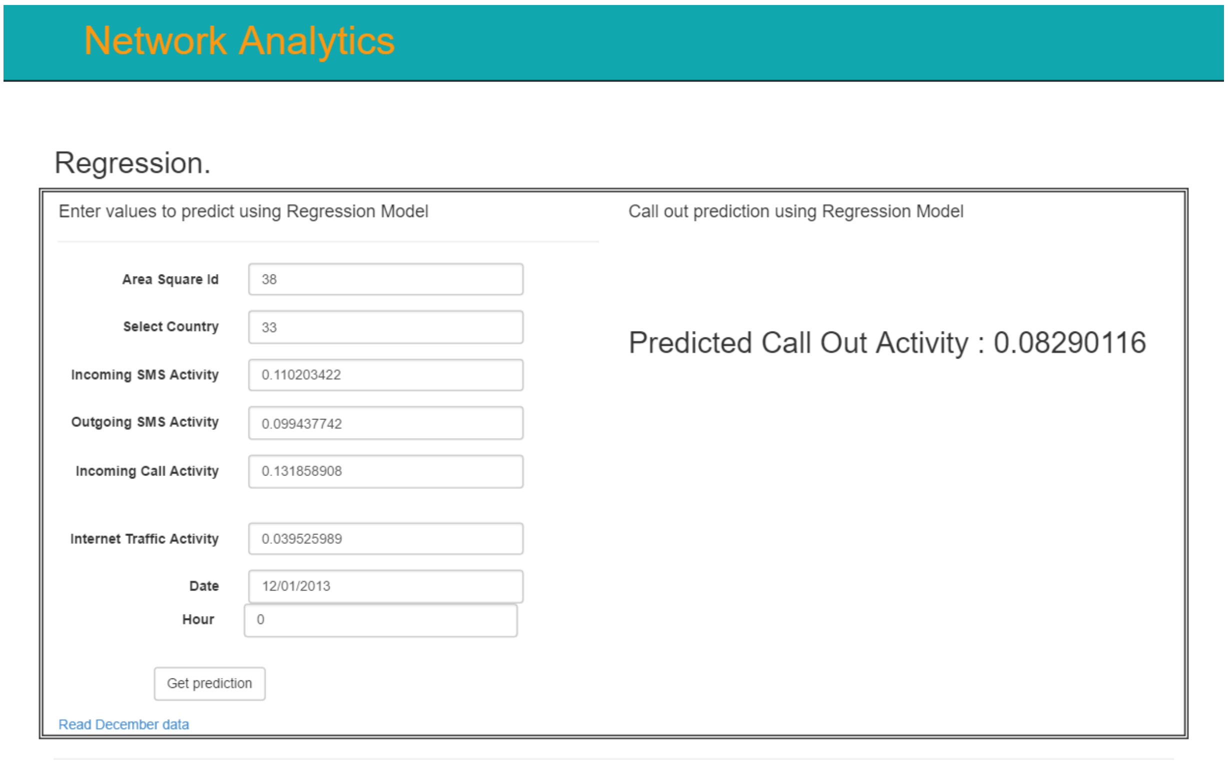
Regression.

Enter values to predict using Regression Model

Area Square Id	
Select Country	
Incoming SMS Activity	
Outgoing SMS Activity	
Incoming Call Activity	
Internet Traffic Activity	
Date	mm/dd/yyyy
Hour	

Call out prediction using Regression Model

Predicted Call Out Activity :



Network Analytics

Regression.

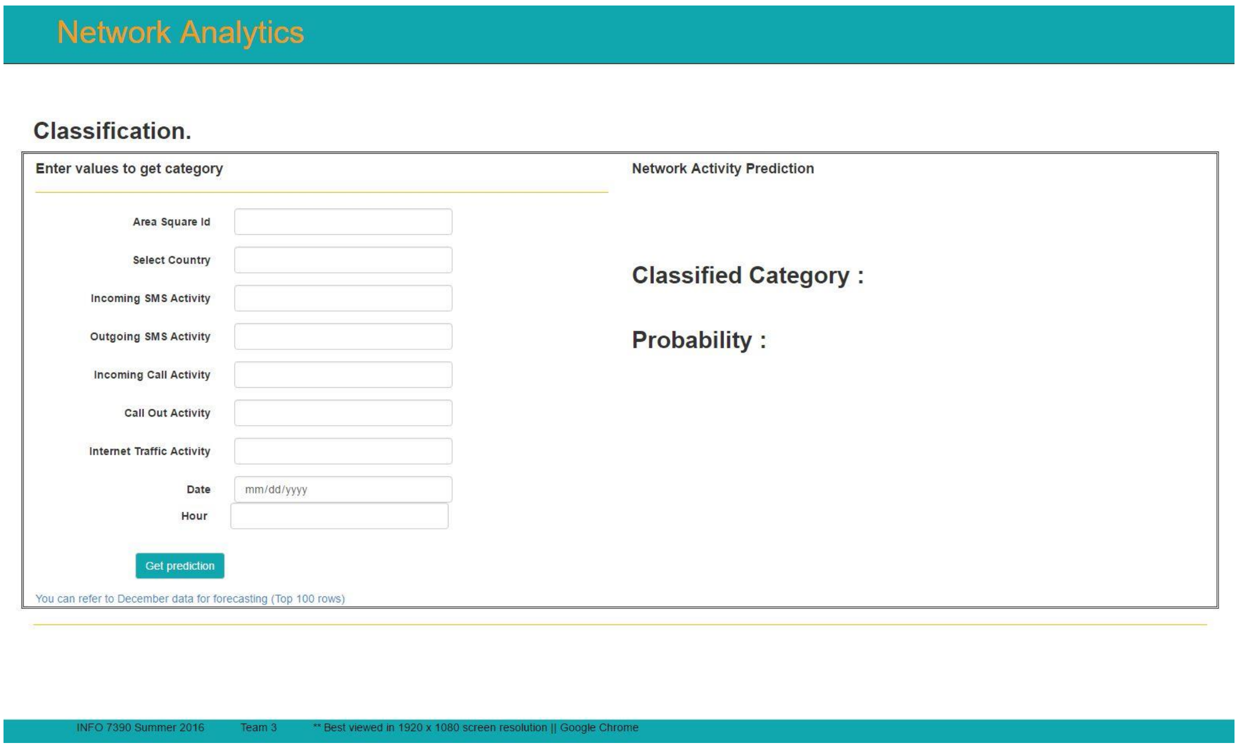
Enter values to predict using Regression Model

Area Square Id	38
Select Country	33
Incoming SMS Activity	0.110203422
Outgoing SMS Activity	0.099437742
Incoming Call Activity	0.131858908
Internet Traffic Activity	0.039525989
Date	12/01/2013
Hour	0

Call out prediction using Regression Model

Predicted Call Out Activity : 0.08290116

❖ Predict high/low activity through the Classification model.


Network Analytics

Classification.

Enter values to get category

Area Square Id	
Select Country	
Incoming SMS Activity	
Outgoing SMS Activity	
Incoming Call Activity	
Call Out Activity	
Internet Traffic Activity	
Date	mm/dd/yyyy
Hour	

Get prediction

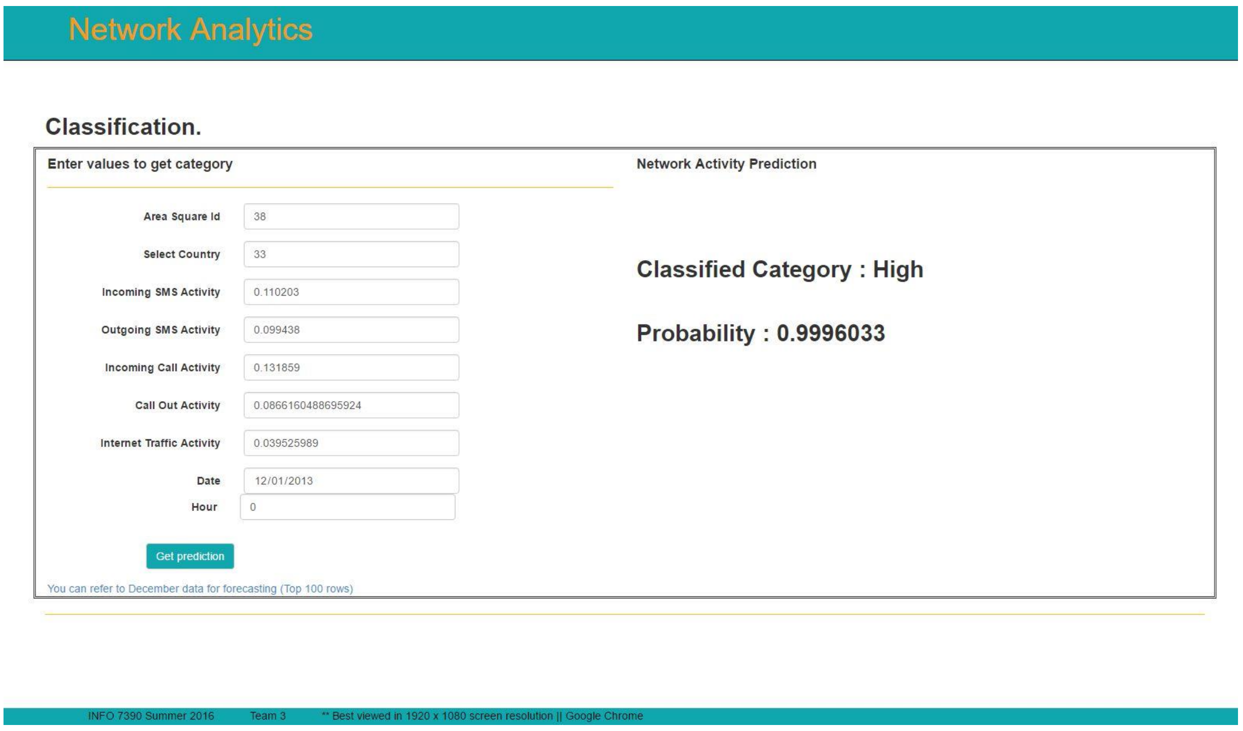
You can refer to December data for forecasting (Top 100 rows)

Network Activity Prediction

Classified Category :

Probability :

INFO 7390 Summer 2016 Team 3 ** Best viewed in 1920 x 1080 screen resolution || Google Chrome


Network Analytics

Classification.

Enter values to get category

Area Square Id	38
Select Country	33
Incoming SMS Activity	0.110203
Outgoing SMS Activity	0.099438
Incoming Call Activity	0.131859
Call Out Activity	0.0866160488695924
Internet Traffic Activity	0.039525989
Date	12/01/2013
Hour	0

Get prediction

You can refer to December data for forecasting (Top 100 rows)

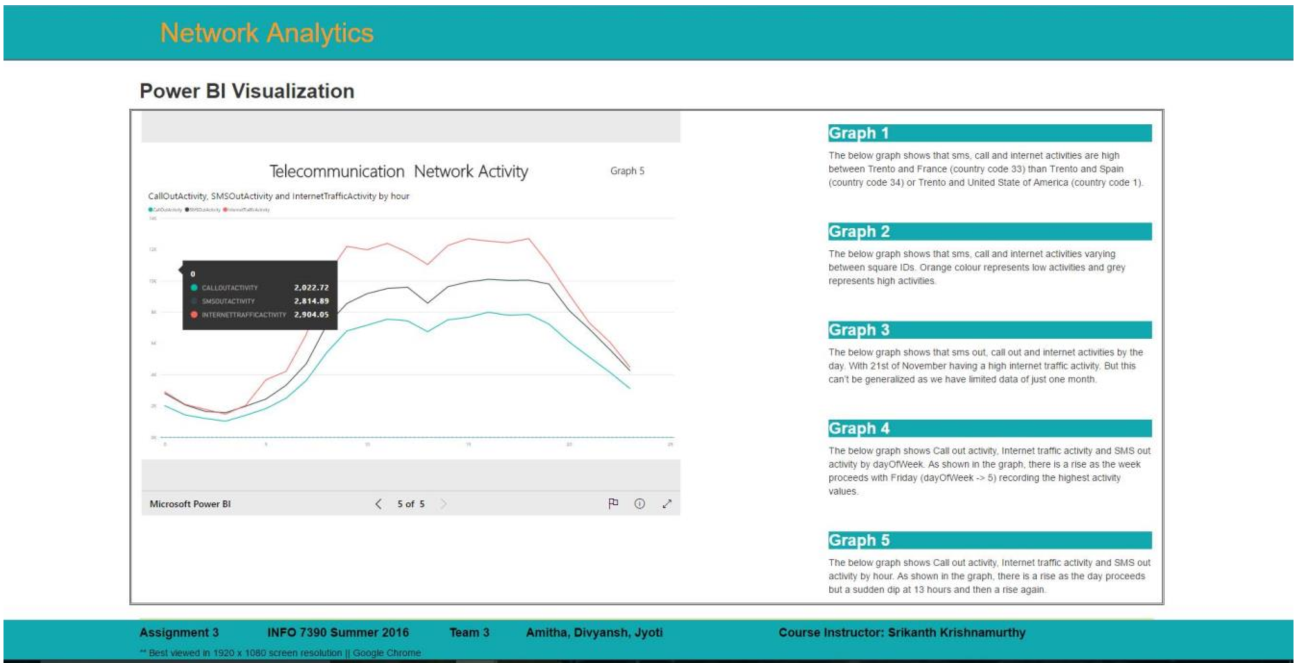
Network Activity Prediction

Classified Category : High

Probability : 0.9996033

INFO 7390 Summer 2016 Team 3 ** Best viewed in 1920 x 1080 screen resolution || Google Chrome

❖ PowerBI Integration with Web Application



Network Analytics

Power BI Visualization

Telecommunication Network Activity

Graph 5

CallOutActivity, SMSOutActivity and InternetTrafficActivity by hour

0
CALLOUTACTIVITY 2,022.72
SMSOUTACTIVITY 2,814.89
INTERNETTRAFFICACTIVITY 2,904.05

Microsoft Power BI 5 of 5

Graph 1

The below graph shows that sms, call and internet activities are high between Trento and France (country code 33) than Trento and Spain (country code 34) or Trento and United State of America (country code 1).

Graph 2

The below graph shows that sms, call and internet activities varying between square IDs. Orange colour represents low activities and grey represents high activities.

Graph 3

The below graph shows that sms out, call out and internet activities by the day. With 21st of November having a high internet traffic activity. But this can't be generalized as we have limited data of just one month.

Graph 4

The below graph shows Call out activity, Internet traffic activity and SMS out activity by dayOfWeek. As shown in the graph, there is a rise as the week proceeds with Friday (dayOfWeek -> 5) recording the highest activity values.

Graph 5

The below graph shows Call out activity, Internet traffic activity and SMS out activity by hour. As shown in the graph, there is a rise as the day proceeds but a sudden dip at 13 hours and then a rise again.

Assignment 3 INFO 7390 Summer 2016 Team 3 Amitha, Divyansh, Jyoti Course Instructor: Srikanth Krishnamurthy

** Best viewed in 1920 x 1080 screen resolution || Google Chrome

1.9 References

1. Telecommunications - SMS, Call, Internet – TN
   https://dandelion.eu/datagems/SpazioDati/telecom-sms-call-internet-tn/resource/

2. Trentino Grid
   https://dandelion.eu/datagems/SpazioDati/trentino-grid/description/

3. "X-means: extending *K*-means with efficient estimation of the number of clusters" by Dan Pelleg and Andrew Moore in ICML 2000.