Boston 311 City Service Request

Final Project Report

Advance Data Science

Group -6

NORTHEASTERN UNIVERSITY BOSTON

MASSACHUSSETS

APRIL 2016

Table of contents

SUMMARY: Boston 311 city service is an initiative taken by government to keep the city clean by providing facilities to residents to report any kind of issues they face in their surroundings, so that government server can help them to keep their surroundings clean. Non-emergency **311** call systems, used increasingly in U.S. cities, offer a number of advantages. They give citizens a quick, convenient way to kvetch about problems in their neighborhoods, and get a response. They enable city governments to identify patterns and address issues proactively. BOS: 311 enables real time collaboration with citizens, 'deputizing' mobile users to become the city's eyes and ears. Citizens report potholes, graffiti, and other issues from anywhere in the city using their mobile phones.

311 service was first implemented on 2 October, 1996 in Baltimore County.

311 service for Boston was stared on August 11, 2015.

The 311 service is part of an ongoing effort by the City of Boston to make city government effective and more responsive to the needs of residents, its business owners, and visitors. Residents of Boston need to know only two numbers to access all City of Boston services:

- 9-1-1 -> For emergency only
- 3-1-1 -> for all non-emergency and city government services.

Preparedness:

Information Sharing: 311 systems can represent the first line of communication between citizens and their government. Citizens can call 311 systems to receive up-to-date information on impending emergencies, such as hurricanes, receive guidance on how to shelter in place, where to locate sandbags, and learn about expected flood zones, evacuation routes, and locations of the nearest evacuation centers. 311 call takers can also provide controls on information 'rumor mills' that can often accompany emergency situations.

Planning: Conference call participants reported incorporating the use of 311 systems into their emergency operations plans and/or COOPs.

Response:

Public safety support: During Hurricane Rita, Hurricane Ike, and then again in the aftermath of Hurricane Katrina, Houston 311 systems relieved 911 systems from handling nonemergency calls, resulting in first responders remaining free to handle life-threatening situations. Houston credited its 311 system with providing police and fire department personnel the necessary time they needed to block off flooded roads so that drivers would not accidentally drive through them.

24/7 Government access. In their discussion of the *I-35W Bridge Collapse* of August 1, 2007, Minneapolis participants noted that the incident occurred at 6:05 p.m., a time when the majority of government service agencies were closed. By default, the 311 Center became the gathering point for media and citizen requests for information. The 311 Center's television monitors also allowed staff to collect real-time information on emergency operations, and anticipate questions they were likely to receive, and quickly develop and disseminate responses to the public

Recovery:

Public safety support: Orange County Sheriff's Office staff used their secure laptops to access the 311 system and electronically report the nature and location of a nonemergency incident directly to the 311 call system. For example, when downed trees blocked routes used for emergency vehicles, officers could log in a 311 request for tree removal from their laptops, which would then be sent to the Public Works field operations center. Public Works would dispatch a vehicle immediately and remove the tree, allowing emergency vehicles to enter the area and provide public safety services to persons in need.

Housing: When Hurricane Katrina left thousands of citizens from neighboring jurisdictions homeless, Houston 311 was the number to call for help with housing issues. 311 became a housing hot line for persons either wanting to temporarily relocate to Houston or wanting to make Houston their permanent home and needed information on how to establish residency.

OVERVIEW:

This document covers the cleaning of datasets of Boston 311 city service, and technical aspects of using Azure Machine Learning (AML), Microsoft Visual Studio for using the restful API generated in AML for the implementation and deployment of model. The document consists of two main parts:

1. ***Data Mining***: Data mining is defined as extracting the information from a huge set of data. In other words we can say that data mining is mining the knowledge from data.
2. ***Data mining engine*** is very essential to the data mining system. It consists of a set of functional modules that perform the following functions
    - Characterization
    - Association and Correlation Analysis
    - Prediction
    - Outlier analysis
    - Evolution analysis
3. ***Data Integration:*** Data Integration is a data preprocessing technique that merges the data from multiple heterogeneous data sources into a coherent data store. Data integration may involve inconsistent data and therefore needs data cleaning.
4. ***Data Cleaning:*** Data cleaning is a technique that is applied to remove the noisy data and correct the inconsistencies in data. Data cleaning involves transformations to correct the wrong data. Data cleaning is performed as a data preprocessing step while preparing the data for a data warehouse.
5. ***Data selection:*** Data Selection is the process where data relevant to the analysis task are retrieved from the database. Sometimes data transformation and consolidation are performed before the data selection process.
6. ***Data Transformation:*** In this step, data is transformed or consolidated into forms appropriate for mining, by performing summary or aggregation operations.
7. ***User Interface:*** User interface is the module of data mining system that helps the communication between users and the data mining system. User Interface allows the following functionalities

Technical Tools that were used in the implementation of web service from raw data is as follow:

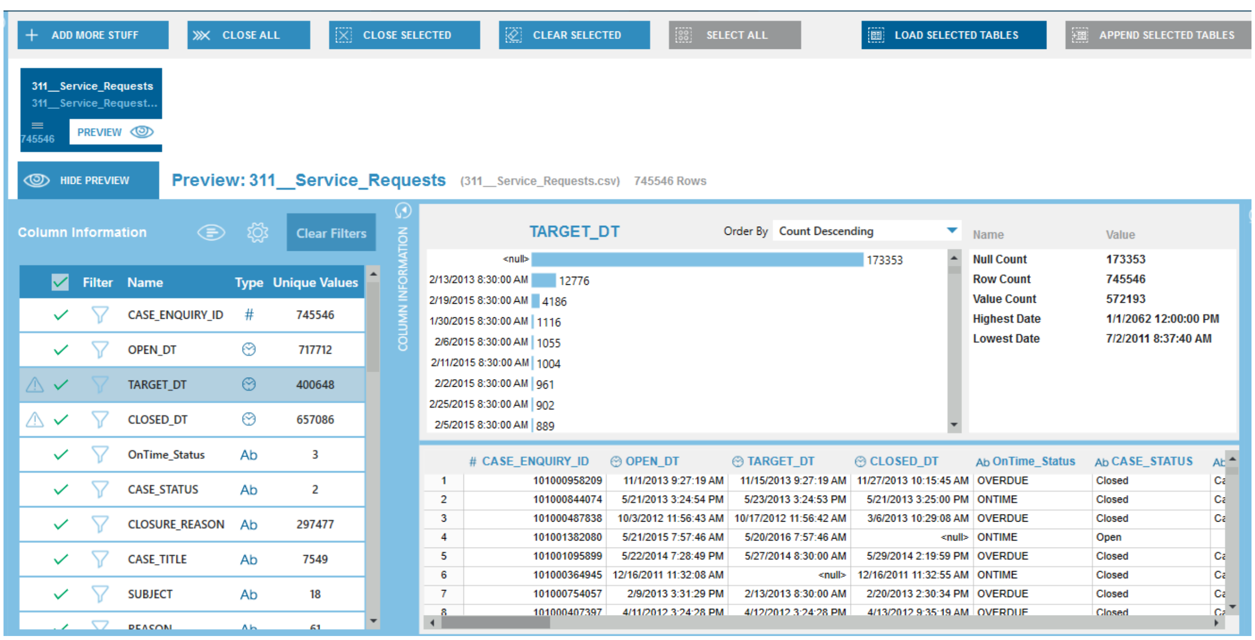
1) R -> Data Cleaning
2) DataWatch ->Data Cleaning
3) Tableau -> Visualization
4) Power BI -> Visualization
5) Azure Machine Learning -> Predictive Model and restful API generation
6) Microsoft Visual Studio -> Deploying website
7) Html, CSS, Bootstrap-> Front End
8) Node js & AWS-> Utilization of restful API for Prediction

DATA PREPARATION:

Data-Preprocessing:

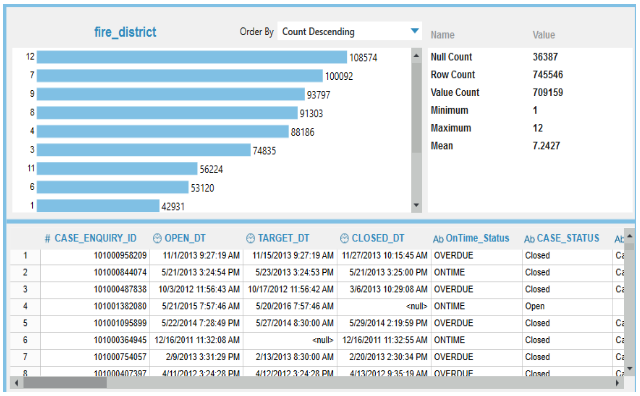
The Boston 311 City Service data is available on boston.gov website. We have around seven lakhs records in data set. The attributes in data sets are case id no, department, location, reason, subject, geo location , opening date, target date, closing date, case status, and on time status of requests.

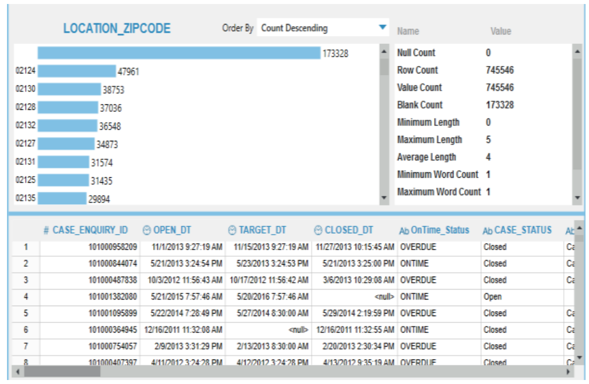
Summary of data: We have used Data watch and R for pre-processing of data. Below image shows the overall summary of data and glimpse of how data looks once uploaded in data watch.

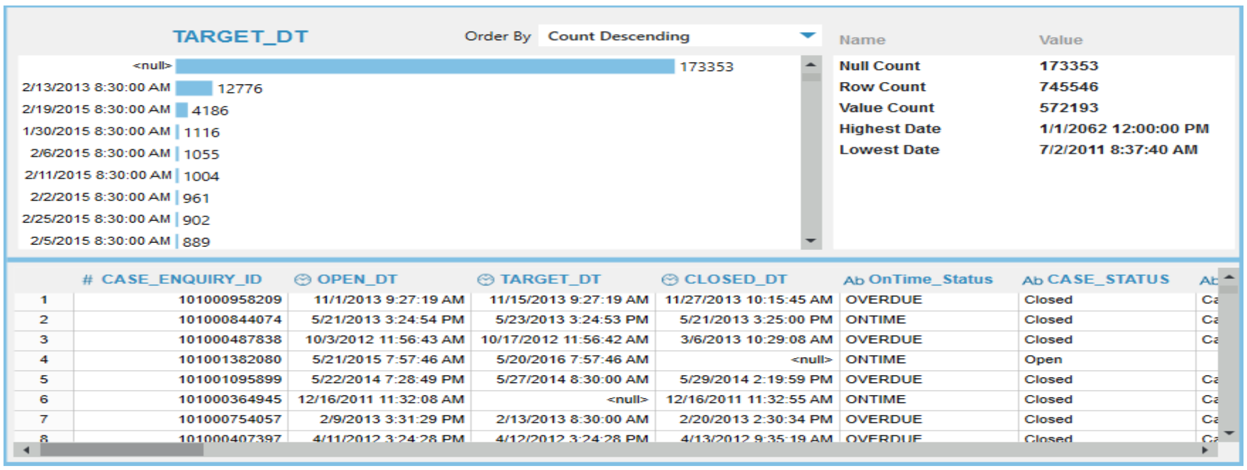


Once the data is uploaded we can see all the attributes and their data type and unique count. Also it is very easy to check the null or blank values count by just clicking on the attribute from left plane.

Below is summary of our null and blank and total row count and unique count of attributes.

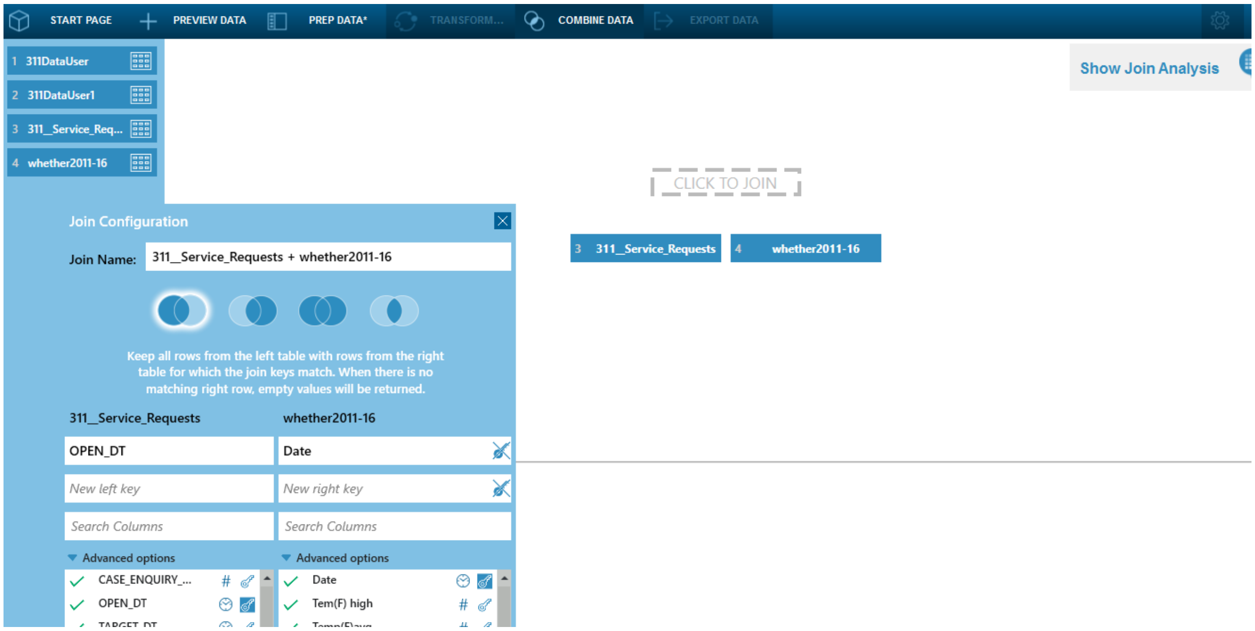






Also we have merged the weather data to existing data by using data watch data preparation studio. Steps involved were

- Upload both the data files
- Select both the files and click on the load data file option
- Select combine option at the top
- Drag your files
- Choose a relation between file(common join key)
- Click ok



We also used **R studio** for data preparation.

- Below is the screen shot of r code for extracting weekday or weekend from date for prediction.

```
### find whether the day is a weekday or a weekend

df$openweekday <- weekdays(df$OPEN_DT)
df$openweekdaynum <- ifelse(df$openweekday == 'Saturday' | df$openweekday == 'Sunday',1,0)
```

- Below code gives us date, month and year column

```
### find the date, month, year of the open date column
df$opendate <- format(df$OPEN_DT,'%d')
df$openmonth <- format(df$OPEN_DT,'%m')
df$openyear <- format(df$OPEN_DT,'%Y')
```

- Below code is used to replace the null values in zip code from available same combination of street name

```
### replacing the zip code

for (i in 1:664430){
  if (df$neighborhood[i]=="Charlestown" & is.na(df$LOCATION_ZIPCODE[i])) {
    df$LOCATION_ZIPCODE[i] <-  2129
  }
}
for (i in 1:664430){
  if (df$neighborhood[i]=="Allston / Brighton" & is.na(df$LOCATION_ZIPCODE[i])) {
    df$LOCATION_ZIPCODE[i] <-  2135
  }
}
for (i in 1:664430){
  if (df$neighborhood[i]=="Back Bay" & is.na(df$LOCATION_ZIPCODE[i])) {
    df$LOCATION_ZIPCODE[i] <-  2116
  }
}
```

- Replaces the null values for following departmets

```
### remove rows where the fire districh, police district, city district council and pwd district is null

sum(is.na(df$fire_district) & is.na(df$police_district)& is.na(df$pwd_district) & is.na(df$city_council_distr

df <- df[!with(df,is.na(df$fire_district) & is.na(df$police_district)& is.na(df$pwd_district) & is.na(df$city
```

- Removed all invalid cases which will our prediction

```
### remove all the invalid cases

df$CLOSURE_REASON <- sub(".*invalid.*","NA",ignore.case = T,df$CLOSURE_REASON)

index <- with(df, which(df$CLOSURE_REASON=="NA", arr.ind=TRUE))

## creat a new column which will tell wheather a case is invalid or valid

df$flag <- ifelse(df$CLOSURE_REASON == "NA",1,0 )

## replacing outliers

df <-  df[!is.na(df$daysSolved),]

df <-  df[!is.na(df$neighborhood),]
```

- Replaced the 0 values in days solved column which indicates that issue was resolved in few hours because we are predicting days and not hours.

```r
#replacing the days SOLVED if its 0 replace with 1

df$daysSolved <- ifelse(df$daysSolved == 0,1,df$daysSolved)
```

- Calculates the days solved and days targeted from closed date and target date column

```r
#calculating days solved and days targetted
df$daysSolved <- as.Date(df$CLOSED_DT)-as.Date(df$OPEN_DT)
df$daysTargeted <- as.Date(df$TARGET_DT)-as.Date(df$OPEN_DT)
```

```r
# changing the format of the open, target and closed date

df$OPEN_DT <- as.Date(as.POSIXct(df$OPEN_DT,format =  "%m/%d/%y %H:%M"))
df$TARGET_DT <- as.Date(as.POSIXct(df$TARGET_DT,format =  "%m/%d/%y %H:%M"))
df$CLOSED_DT <- as.Date(as.POSIXct(df$CLOSED_DT,format =  "%m/%d/%y %H:%M"))
```

DATA ANALYSIS:

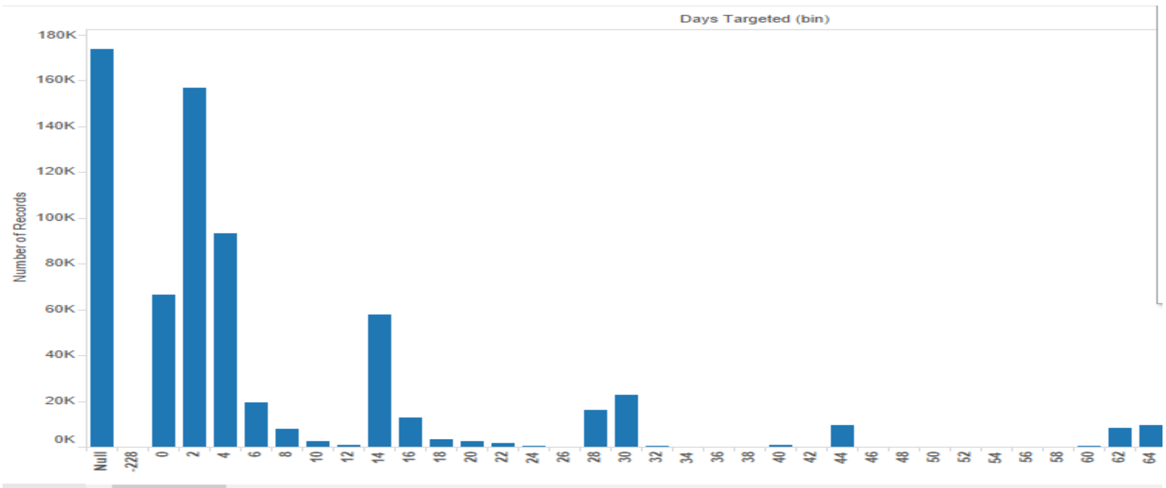
Data Exploration: The total number of records in dataset were **745,546** out of which few were invalid records and some were null ad blanks.

Summary of Invalid Records

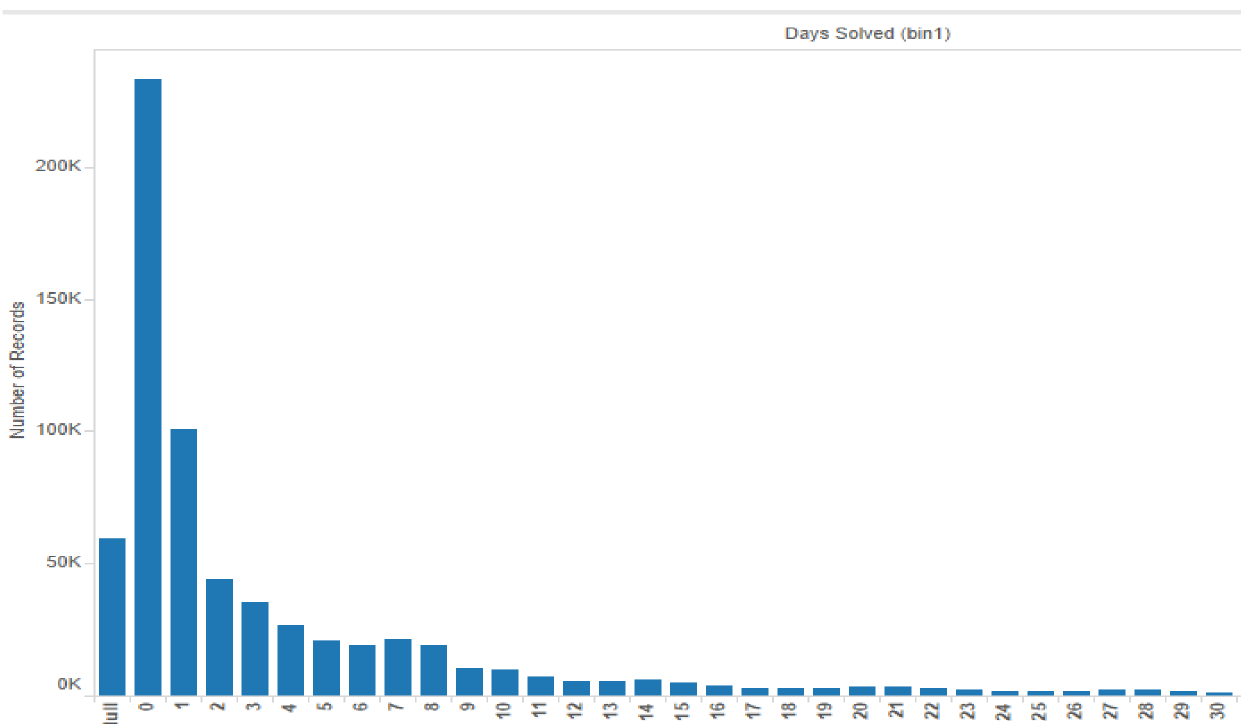
invalid count	
invalid	✗ 17,933
Valid	✓ 727,613

CLOSURE_REASON
Case Closed Case Invalid
Case Closed Case Invalid computer diffic
Case Closed Case Invalid
Case Closed Case Invalid
Case Closed Case Invalid Accidental dup
Case Closed Case Invalid Tree replanted
Case Closed Case Invalid e-form didn't a
Case Closed Case Invalid
Case Closed Case Invalid no informatior
Case Closed Case Invalid nothing at 58 ar
Case Closed Case Invalid
Case Closed Case Invalid duplicate
Case Closed Case Invalid
Case Closed Case Invalid
Case Closed Case Invalid
Case Closed. Closed date : 2015-08-27 10
Case Closed Case Invalid
Case Closed Case Invalid wrong dept
Case Closed Case Invalid
Case Closed. Closed date : 2015-09-19 11
Case Closed Case Invalid wrong intersec

Anomaly Detection: Weird numbers were found in targeted day's prediction and days solved. Below charts describes the days targeted in initial dataset. The zero indicates that, a particular request was solved within few hours, which we will be replacing with 1 because we have not considered hourly data. Also we have null values in target column.



Below charts describes the range of days solved. We have created bin of one for this chart. The null values tell us that there are few cases which are still open and not closed so we have null.



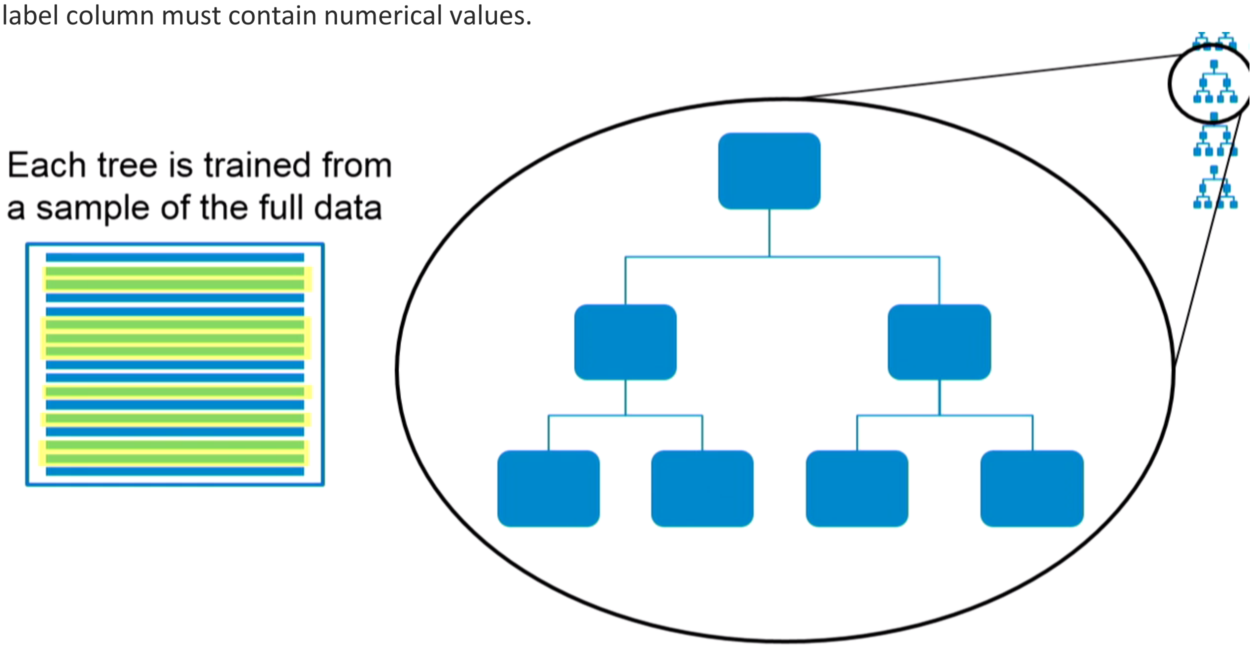
The attributes such as fire district, pwd district, police department and city council district were found missing for few records. We did analysis and noticed the pattern for rest of the departments for respective missing department.

For example: The image below is one of the observed pattern for missing values of fire district pattern

fire_district	pwd_district	city_co	police_
	10A	8	D4
	10A	8	D4
	10A	8	D4
	10A	8	D4

Data Modelling:

Boosted Decision Tree Overview: You can 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.



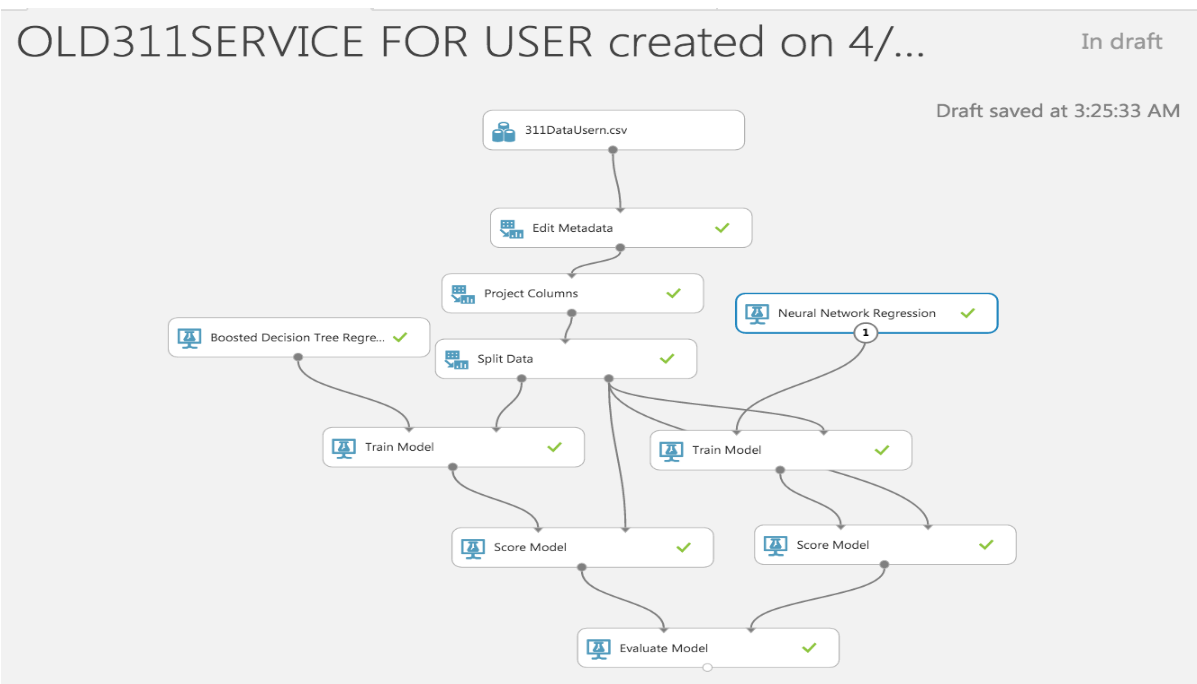
Neural Network Regression: You can use the **Neural Network Regression** module to create a regression model using a customizable neural network algorithm.
Although neural networks are widely known for use in deep learning and modeling complex problems such as image recognition, they are easily adapted to regression problems. Any class of statistical models can be termed a neural network if they use adaptive weights and can approximate non-linear functions of their inputs. Thus neural network regression is suited to problems where a more traditional regression model cannot fit a solution.

Neural network regression is a supervised learning method, and therefore requires a *tagged dataset*, which includes a label column. Because a regression model predicts a numerical value, the label column must be a numerical data type.

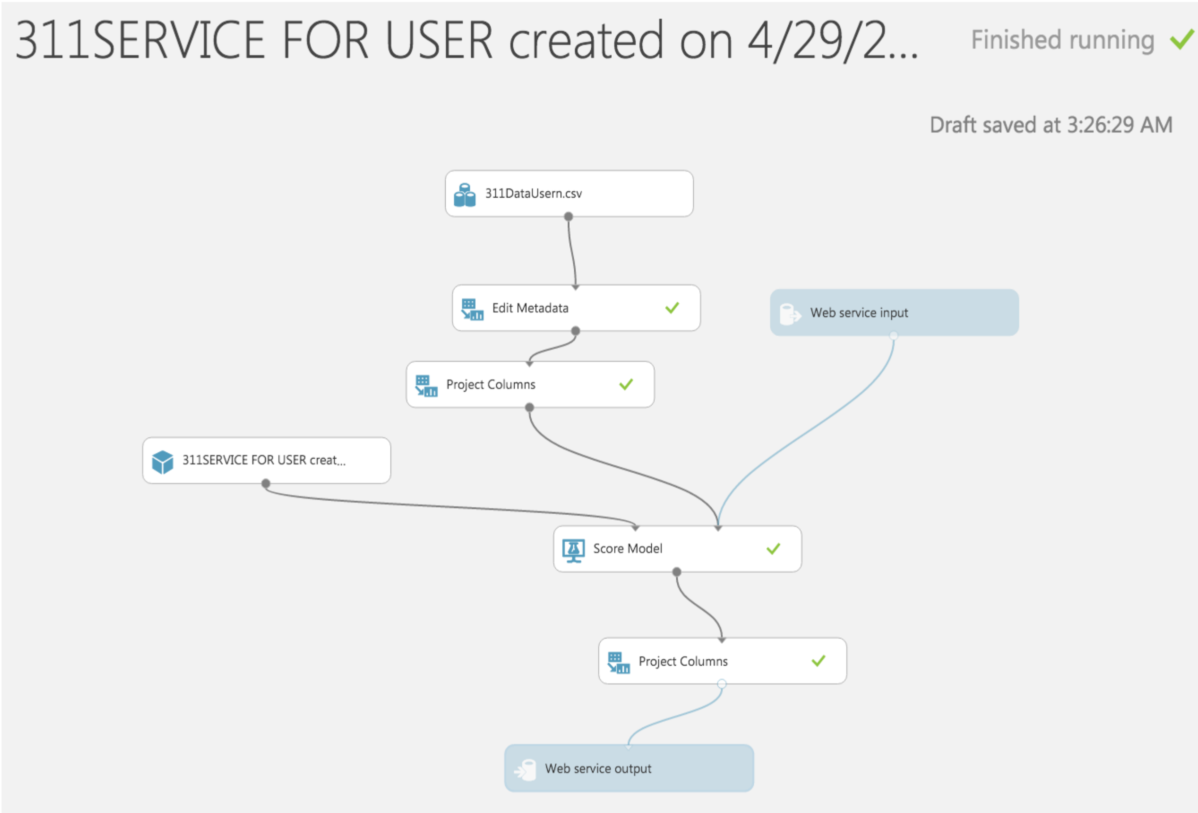
Microsoft Azure:

We created the model to predict the number of days required to solve a particular issue in the given location. We used the Boston 311 service request data which has been cleaned in R and Data watch tool. Firstly we determined the dependent Variable for our model. We did the feature selection by calculating the correlation as well as by using the feather selection component given in Azure. We ran our model using Boosted Decision Regression and Neural Network algorithm. We observed that we got better accuracy and less error by using Decision Regression algorithm. Also we have created two model the one which predicts the number of days as per the input parameters given by the user and second which predicts the number of days to solve the bulk request raised by different user to the 311 service provider. The second model is basically designed for 311 service provider to help them predict the days required to solve the issue so that then ca work efficiently and accurately.

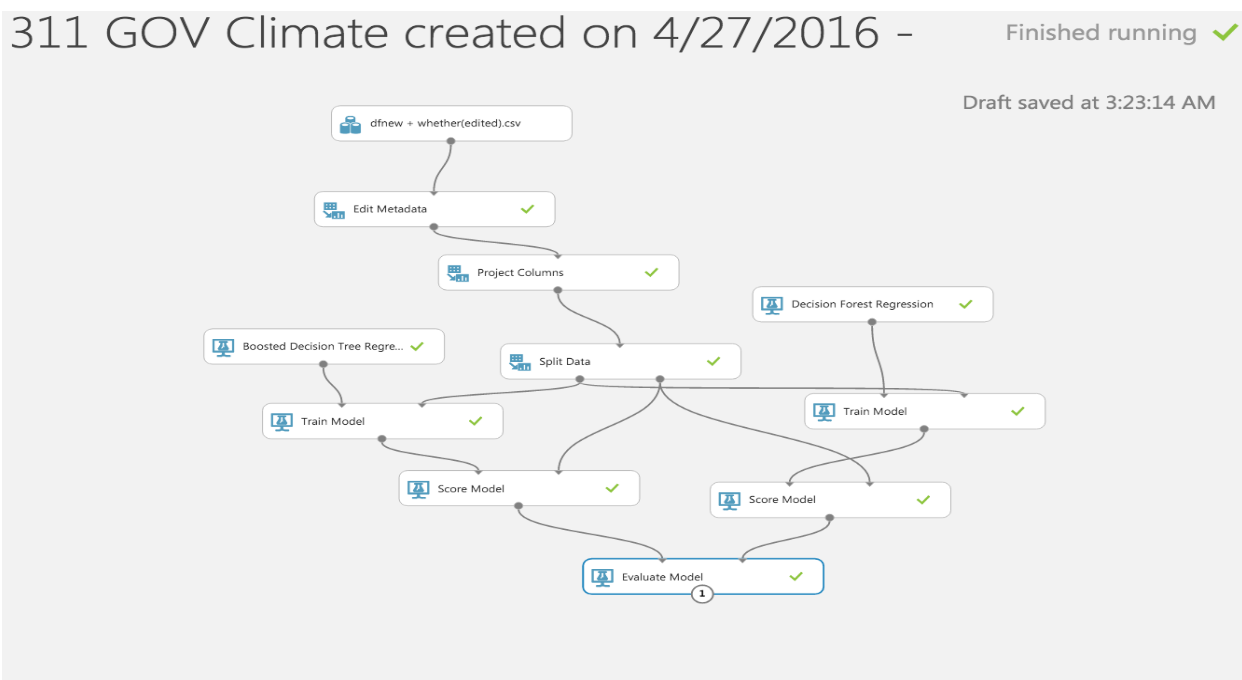
Microsoft Azure Model 1: For Boston residents



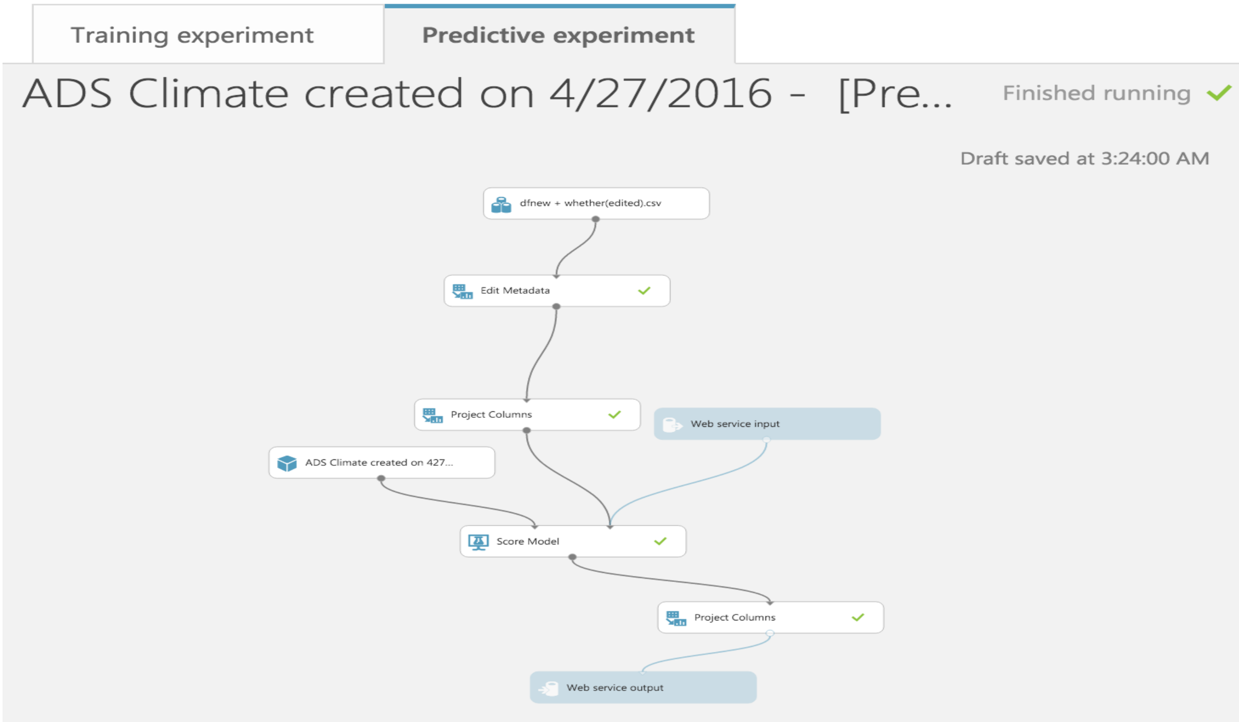
Web Deployment



Microsoft Azure Model 2: For Government



Web Deploymennt



Metrics Report:

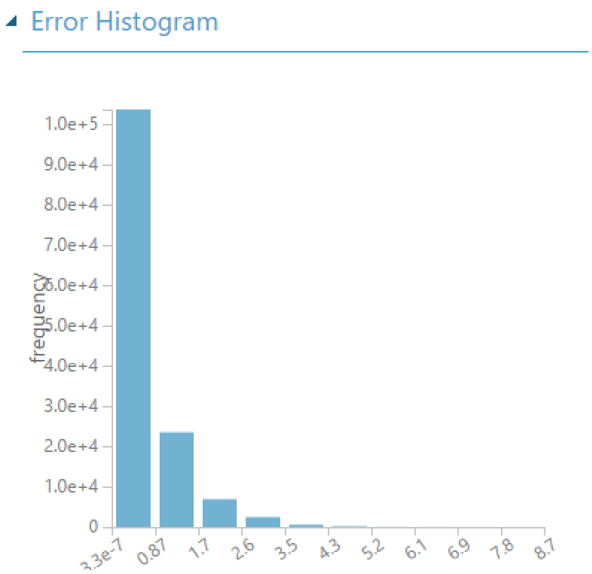
Metrics

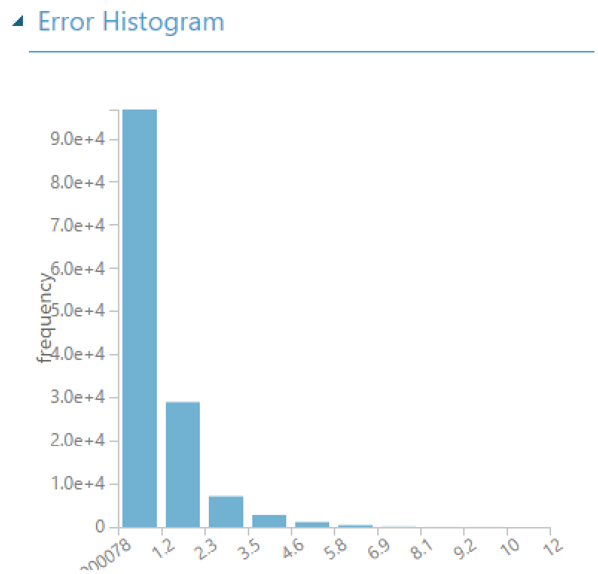
Mean Absolute Error	0.62911
Root Mean Squared Error	0.931092
Relative Absolute Error	0.353136
Relative Squared Error	0.164799
Coefficient of Determination	0.835201

Metrics

Mean Absolute Error	0.982262
Root Mean Squared Error	1.385386
Relative Absolute Error	0.55137
Relative Squared Error	0.364848
Coefficient of Determination	0.635152

Error Histogram:





Prediction:

311SERVICE FOR USER created on 4/29/2... › Score Model › Scored dataset

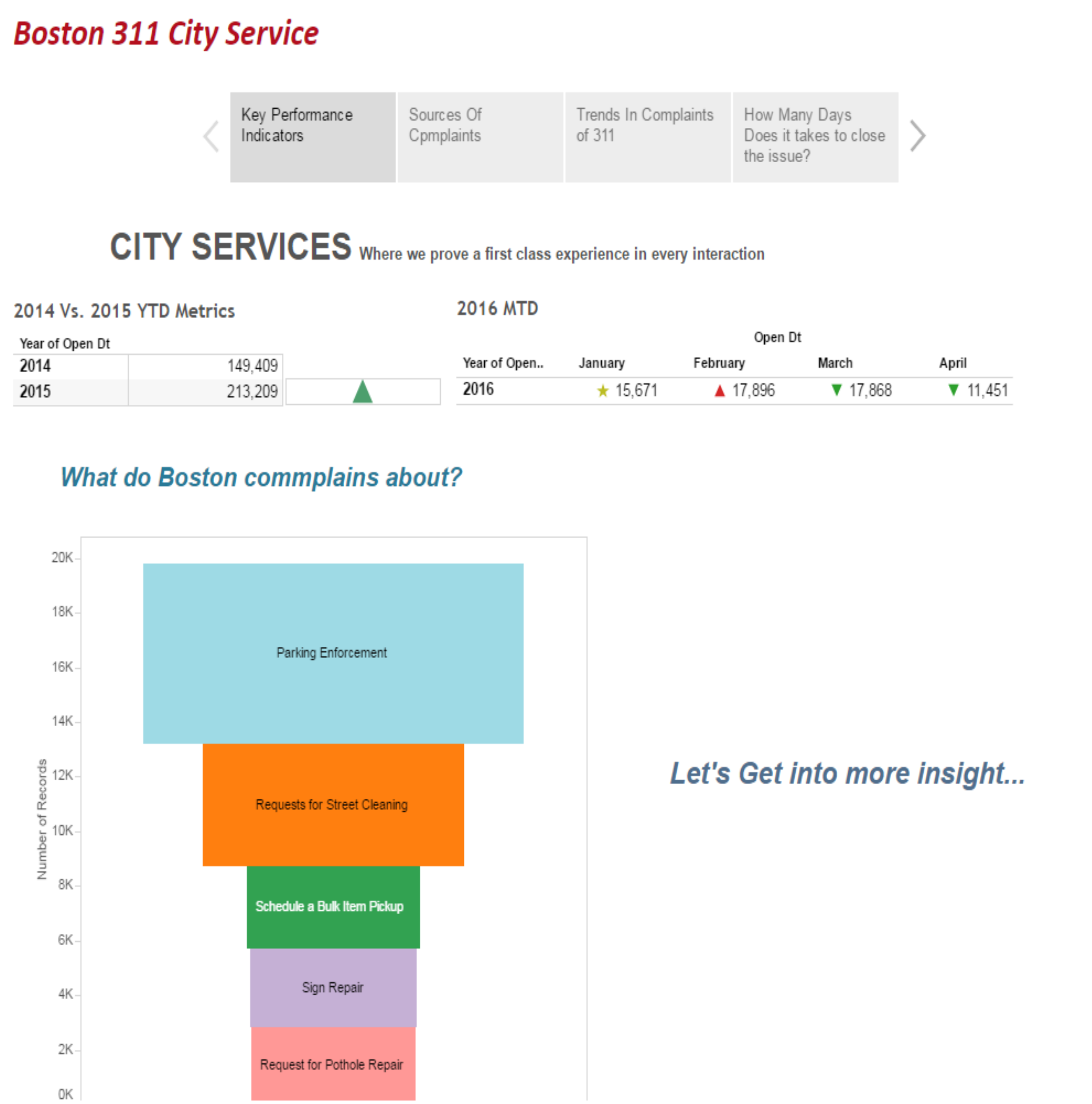
rows	columns
332215	11

daysSolved	flag	opendate	openmonth	openyear	openweekdaynum	Scored Labels
1	0	9	11	2014	1	7.0329
1	0	19	11	2012	0	4.2273
1	0	30	7	2013	0	2.6093
4	0	10	7	2015	0	6.1157
1	0	6	11	2013	0	3.9674

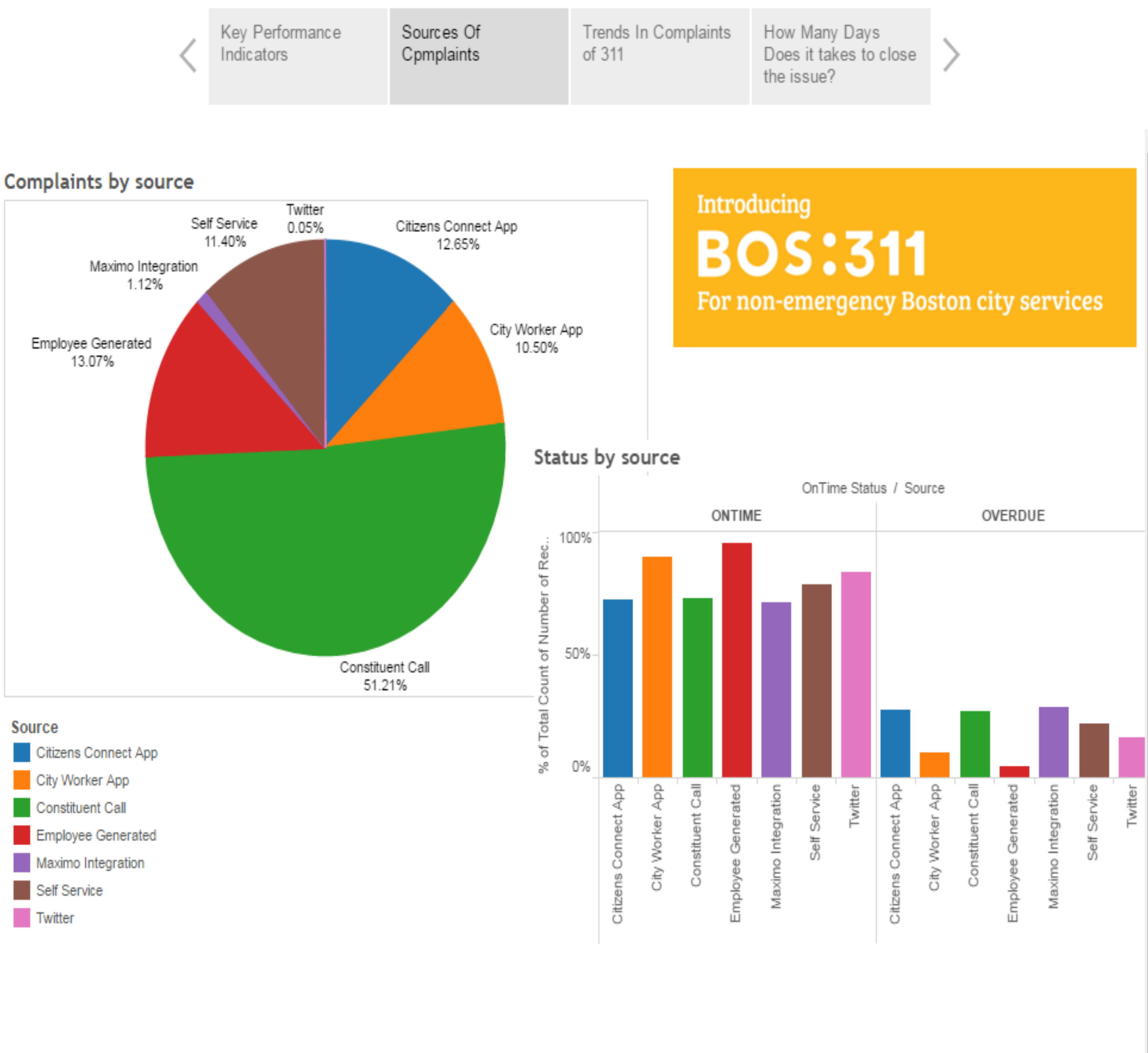
DATA VISUALIZATION:

Using Data to Manage a City:

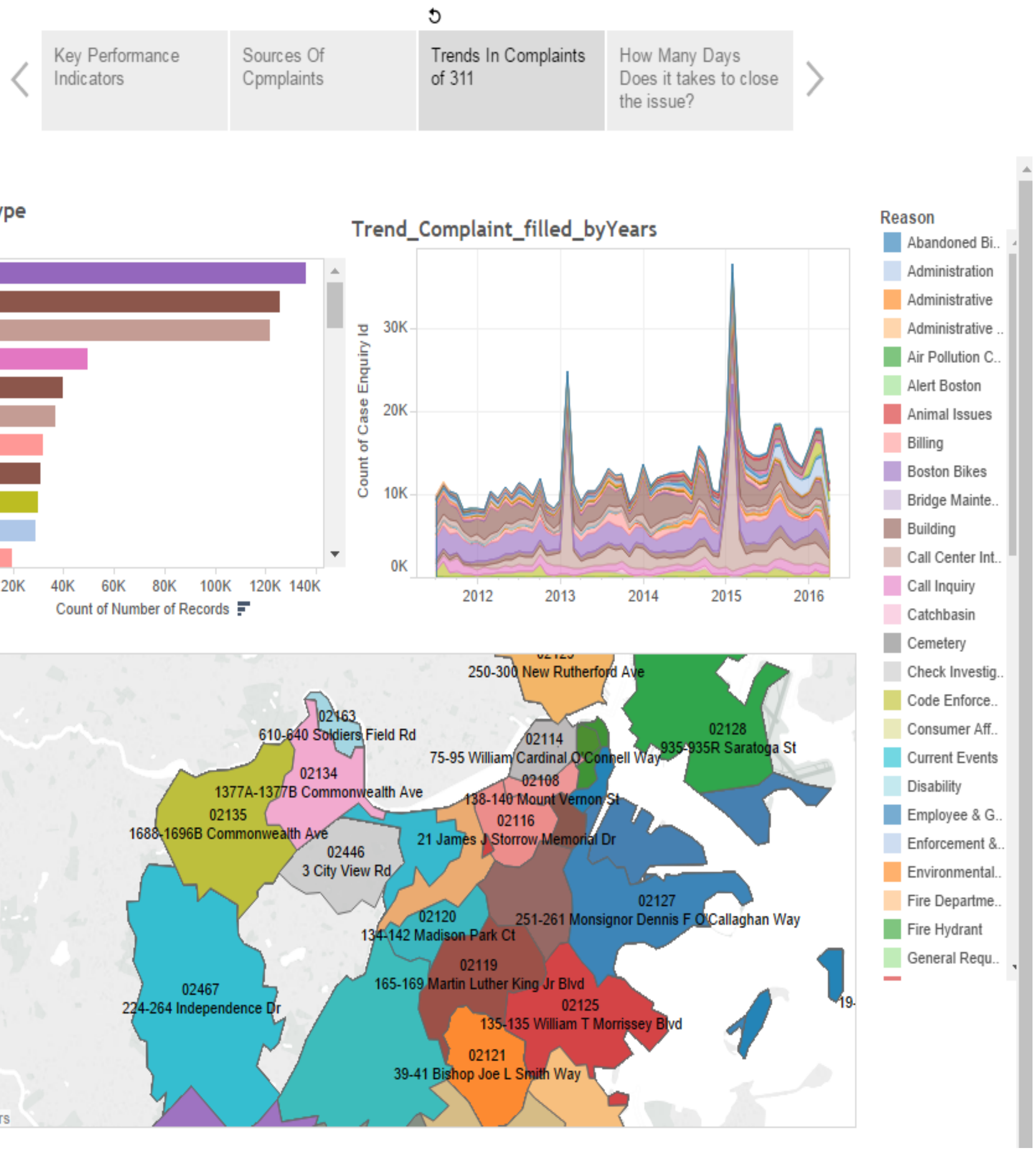
Every day, Mayor and his management track daily actions and advancement on major strategic objectives. The team uses a set of large dashboards, to envision what the City is doing to realize these objectives. Check out the dashboards below to see what the Mayor is seeing.

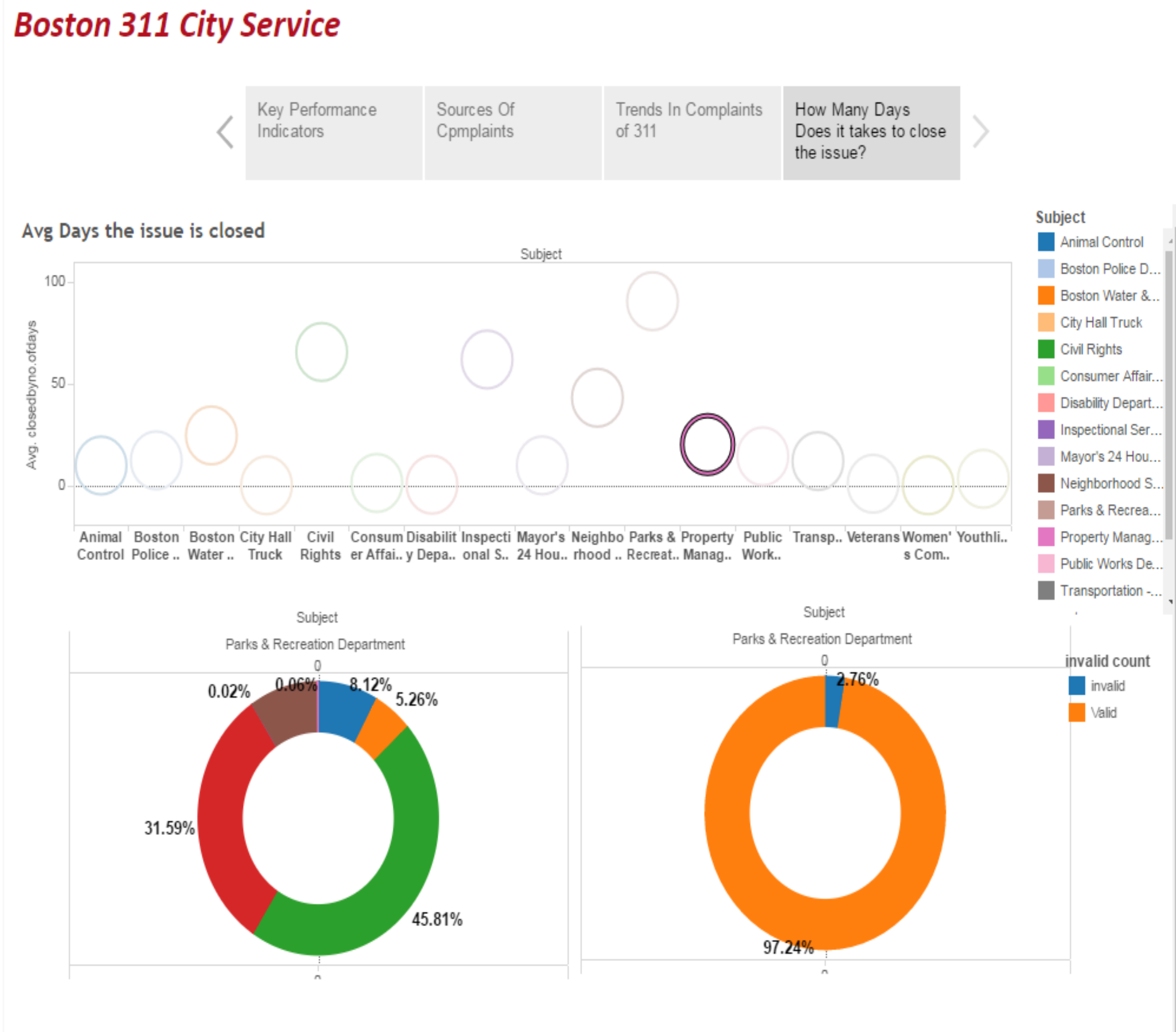


Boston 311 City Service



Boston 311 City Service



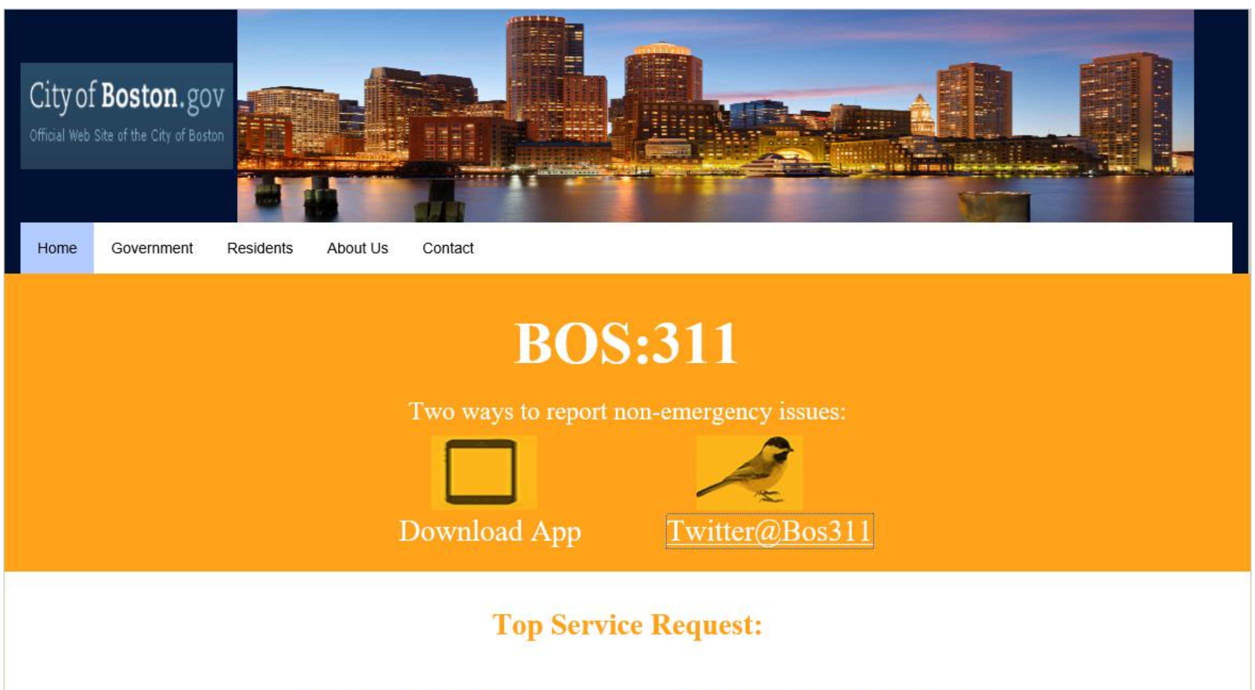

Boston 311 City Service
Key Performance Indicators
Sources Of Cpmplaints
Trends In Complaints of 311
How Many Days Does it takes to close the issue?
Avg Days the issue is closed
Subject
Avg. closedbyno.ofdays
100
50
0
Animal Control
Boston Police ..
Boston Water ..
City Hall Truck
Civil Rights
Consumer Affai..
Disability Depa..
Inspectional S..
Mayor's 24 Hou..
Neighborhood ..
Parks & Recreat..
Property Manag..
Public Work..
Transp..
Veterans
Women's Com..
Youthli..
Subject
Animal Control
Boston Police D...
Boston Water &...
City Hall Truck
Civil Rights
Consumer Affair...
Disability Depart...
Inspectional Ser...
Mayor's 24 Hou...
Neighborhood S...
Parks & Recrea...
Property Manag...
Public Works De...
Transportation -...
Subject
Parks & Recreation Department
0
0.02%
0.06%
8.12%
5.26%
31.59%
45.81%
Subject
Parks & Recreation Department
0
2.76%
97.24%
invalid count
invalid
Valid

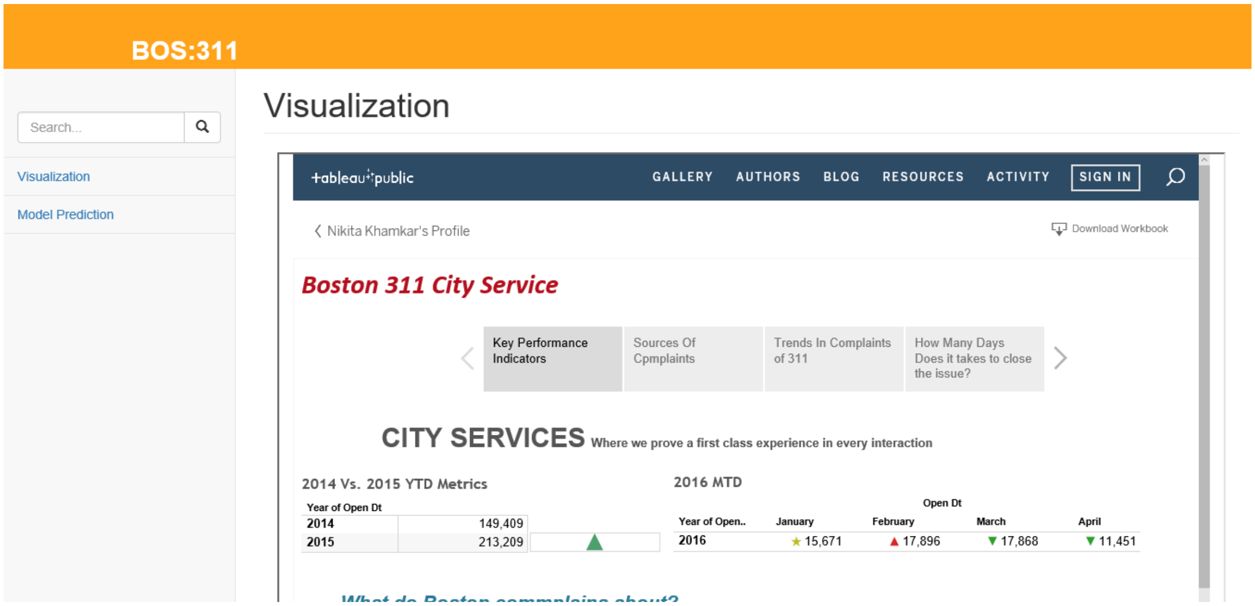
USER INTERFACE:

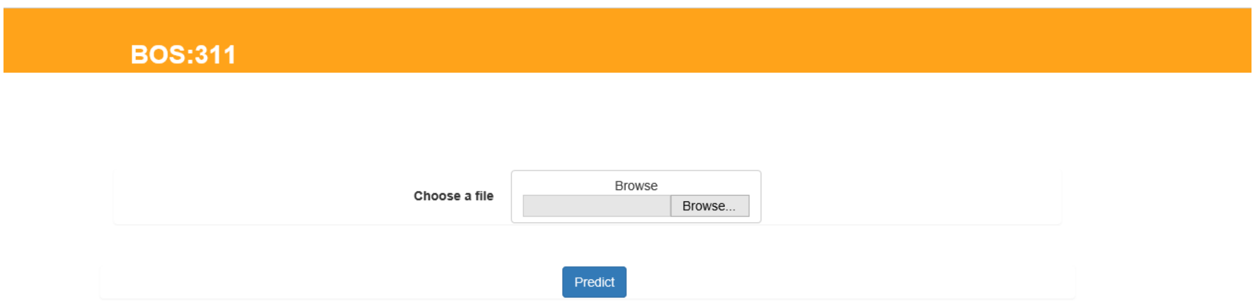
User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

UI Description: We have used different technologies to develop a responsive UI. This UI can be used by the 311 service provider as well as by the resident of Boston. This application gives them a platform to raise their request as well as know when that particular request will be completed. Apart from this we made sure that this application ca be used by 311 service providers as well to predict the days they would take to resolve the bulk issues depending on various factors like (temperature & humidity of the city, Reason and type of the request, Location, Department that needs to handle the particular issue)

Technologies used: HTML5, CSS3, Bootstrap




BOS:311
Search...
Visualization
Model Prediction
Visualization
+ableau public
GALLERY
AUTHORS
BLOG
RESOURCES
ACTIVITY
SIGN IN
Nikita Khamkar's Profile
Download Workbook
Boston 311 City Service
Key Performance Indicators
Sources Of Cpmplaints
Trends In Complaints of 311
How Many Days Does it takes to close the issue?
CITY SERVICES Where we prove a first class experience in every interaction
2014 Vs. 2015 YTD Metrics
Year of Open Dt
2014 149,409
2015 213,209
2016 MTD
Open Dt
Year of Open.. January February March April
2016 15,671 17,896 17,868 11,451


BOS:311
Choose a file
Browse
Browse...
Predict


BOS:311
Download the App
Access non-emergency city services and submit cases straight from your phone.
GET IT ON Google play
Available on the App Store

BACKEND INTEGRATION:

NodeJs is an interface between the user interface and MS Azure. It routes the client request to the server request. We have also used Python which is used to convert the user data into Azure compatible and required input data. The website has been hosted in AWS EC2 instance and below is the URL to connect to the website.

Challenges Faced:

1. NodeJs and python Integration
2. Handling NodeJs asynchronous calls

References : Stack flow site, NodeJs APIs

AWS: Amazon web service where one can create its own instance which is platform independent using Ubuntu 14.02 operating system. Referred site **Digital OLcean** to solve the problems that I faced during creating an instance in AWS.

Live Data Streaming:

Data-Ingestion:

Endpoint - https://data.cityofboston.gov/api/views/awu8-dc52/rows.json

Authentication – Not required

Data Format – JSON

JSON Parser – Jackson

Language – Java

Tool – NetBeans

Cloud Service – Microsoft Azure

Data Ingestion Engine – Azure EventHub

Data Streaming Engine – Azure Stream Analytics

Database – Azure SQL

Libraries Used - azure-eventhubs-0.6.5.jar

gson-1.7.1.jar

proton-j-0.12.1.jar

geronimo-jms_1.1_spec-1.0.jar

qpid-amqp-1-0-client-jms-0.24.jar

qpid-jms-client-0.1.0.jar

1. Data Creation

- Historical Data

Application - BostonCity311__Response_Dump

Historical data is pulled by hitting the endpoint,
The JSON response is parsed and dumped into a file.

Response from end point


```
, [ 3071168, "7F264256-630C-453B-A270-40DDFDE30C61", 3071168, 1452105066, "525314", 1452105066, "525314", null, "101000438913", "2012-06-20T07:51:46", "2012-06-22T07:51:46", "2012-
06-21T06:04:04", "ONTIME", "Closed", "Case Closed Case Resolved ASSIGNED TO CIU", "Utility Call-In", "Public Works Department", "Highway Maintenance", "Utility Call-In",
"PWDx_Utility Call-In", "PWDx", null, null, "74 Medford St  Charlestown  MA  02129", "3", "1A", "1", "A15", "Charlestown", "2", "Ward 2", "0202", "E", "74 Medford St", "02129",
"Address", "273745", "42.3789", "-71.0552", "Constituent Call", [ "{\"address\":\"\",\"city\":\"\",\"state\":\"\",\"zip\":\"\"}", "42.3789", "-71.0552", null, false ] ]
, [ 3071169, "87086A80-1515-4D9F-A9EB-85F119A63915", 3071169, 1452105066, "525314", 1452105066, "525314", null, "101000315873", "2011-08-18T11:51:44", null, "2011-08-18T11:51:44",
"ONTIME", "Closed", "Case Closed Internal Case Performed by Highway Maintenance crew.", "Pothole Repair (Internal)", "Public Works Department", "Highway Maintenance", "Pothole Repair
(Internal)", "PWDx_District 1B: North End", "PWDx", null, null, "46 Hull St  Boston  MA  02113", "3", "1B", "1", "A1", "Downtown / Financial District", "3", "Ward 3", "0302", "R3",
"46 Hull St", "02113", "Address", "75931", "42.3669", "-71.0562", "Employee Generated", [ "{\"address\":\"\",\"city\":\"\",\"state\":\"\",\"zip\":\"\"}", "42.3669", "-71.0562", null,
false ] ]
```

Jackson Parser Output

```
[
3071168
7F264256-630C-453B-A270-40DDFDE30C61
3071168
1452105066
525314
1452105066
525314
null
101000438913
2012-06-20T07:51:46
2012-06-22T07:51:46
2012-06-21T06:04:04
ONTIME
Closed
Case Closed Case Resolved ASSIGNED TO CIU
Utility Call-In
Public Works Department
Highway Maintenance
Utility Call-In
PWDx_Utility Call-In
PWDx
null
null
74 Medford St  Charlestown  MA  02129
3
1A
1
A15
Charlestown
2
Ward 2
0202
E
74 Medford St
02129
Address
273745
42.3789
-71.0552
Constituent Call
[
{"address":"","city":"","state":"","zip":""}
42.3789
-71.0552
null
false
]
]
```

Application - BostonCity311_Historical_Azure

Each Jackson parsed response above is modelled to a **CityData** object

CityData Object

```
String CASE_ENQUIRY_ID;
String OPEN_DT ;
String TARGET_DT ;
String CLOSED_DT ;
String OnTime_Status ;
String CASE_STATUS ;
String CLOSURE_REASON ;
String CASE_TITLE ;
String SUBJECT ;
String REASON ;
String TYPEE ;
String QUEUEE ;
String Department ;
String SubmittedPhoto ;
String ClosedPhoto ;
String Location ;
String fire_district ;
String pwd_district ;
String city_council_district ;
String police_district ;
String neighborhood ;
String neighborhood_services_district ;
String ward ;
String precinct ;
String land_usage ;
String LOCATION_STREET_NAME ;
String LOCATION_ZIPCODE ;
String Property_Type ;
String Property_ID ;
String LATITUDE ;
String LONGITUDE ;
String Source ;
String Geocoded_Location ;
```

- Per day 311 Requests

  Application - BostonCity311_CurrentDate_Azure

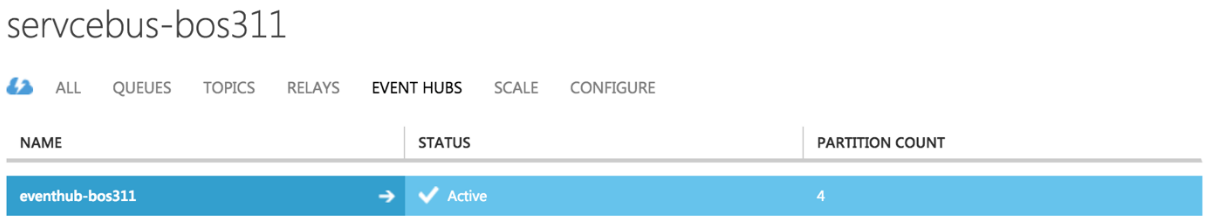
  Jackson Parser Output is modelled to a **CityData** object only if the current date (date when the application is executed) is equal to the value of OPEN_DT key of each 311 requests.

2. Streaming Data to Azure Cloud and store in Azure SQL

- Data Ingestion Engine – EventHub
  The Azure EventHub requires eventhub input data in a JSON and UTF-8 format.
  Each **CityData** object of 311 request is converted into JSON format by using goolge's gson parser and then sent to the eventhub.

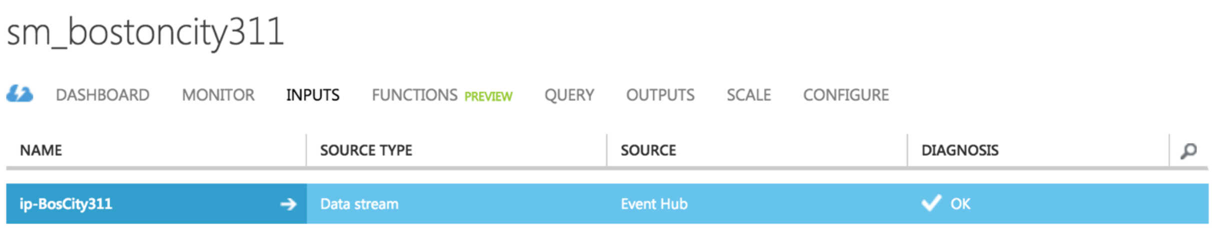

```
{"CASE_ENQUIRY_ID":"101000438913","OPEN_DT":"2012-06-20T07:51:46","TARGET_DT":"2012-06-22T07:51:46","CLOSED_DT":"2012-
06-21T06:04:04","OnTime_Status":"ONTIME","CASE_STATUS":"Closed","CLOSURE_REASON":"Case Closed Case Resolved ASSIGNED TO CIU","CASE_TITLE":"Utility Call-In","SUBJECT":"Public
Works Department","REASON":"Highway Maintenance","TYPEE":"Utility Call-In","QUEUEE":"PWDx_Utility Call-
In","Department":"PWDx","SubmittedPhoto":"null","ClosedPhoto":"null","Location":"74 Medford St  Charlestown  MA  02129","fire_district":"3","pwd_district":"1A","city_council_dis
trict":"1","police_district":"A15","neighborhood":"Charlestown","neighborhood_services_district":"2","ward":"Ward
2","precinct":"0202","land_usage":"E","LOCATION_STREET_NAME":"74 Medford
St","LOCATION_ZIPCODE":"02129","Property_Type":"Address","Property_ID":"273745","LATITUDE":"42.3789","LONGITUDE":"-71.0552","Source":"Constituent
Call","Geocoded_Location":"(42.3789,-71.0552)"}
```

EventHub - eventhub-bos311



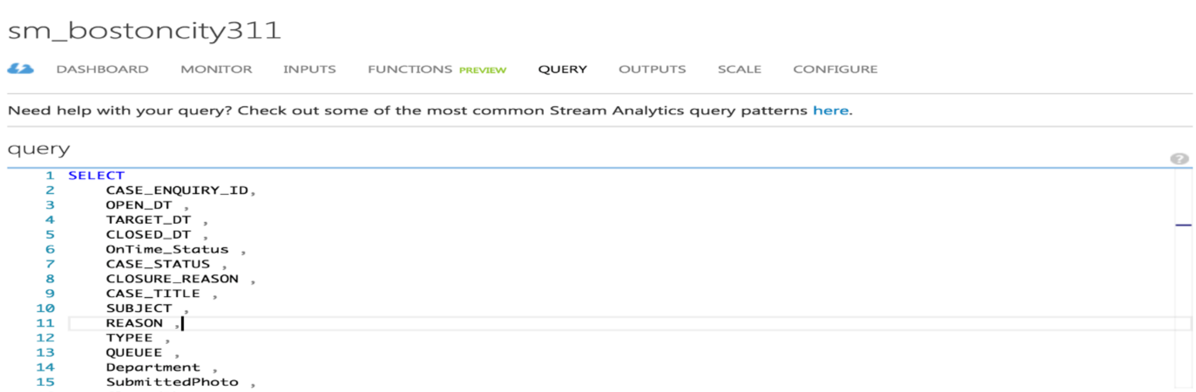
- Data Streaming Engine

  The data streaming engine takes input from eventhub and stores the output from eventhub into SQL database.



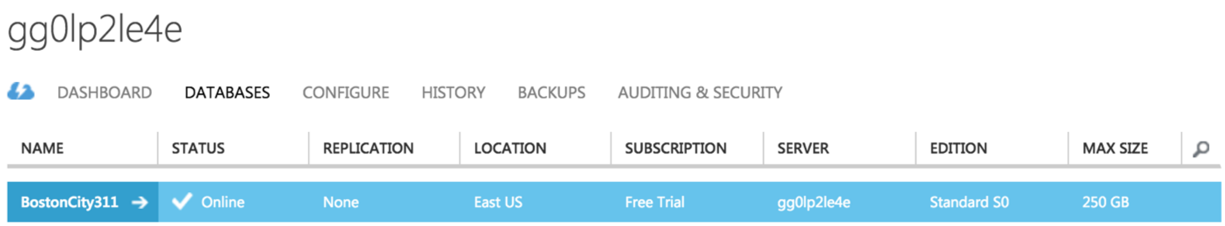
The transition between input and output of Stream Engine is determined by the Query parameter.

The query given below captures the value of all the keys of eventhub JSON data and stores them into SQL database.

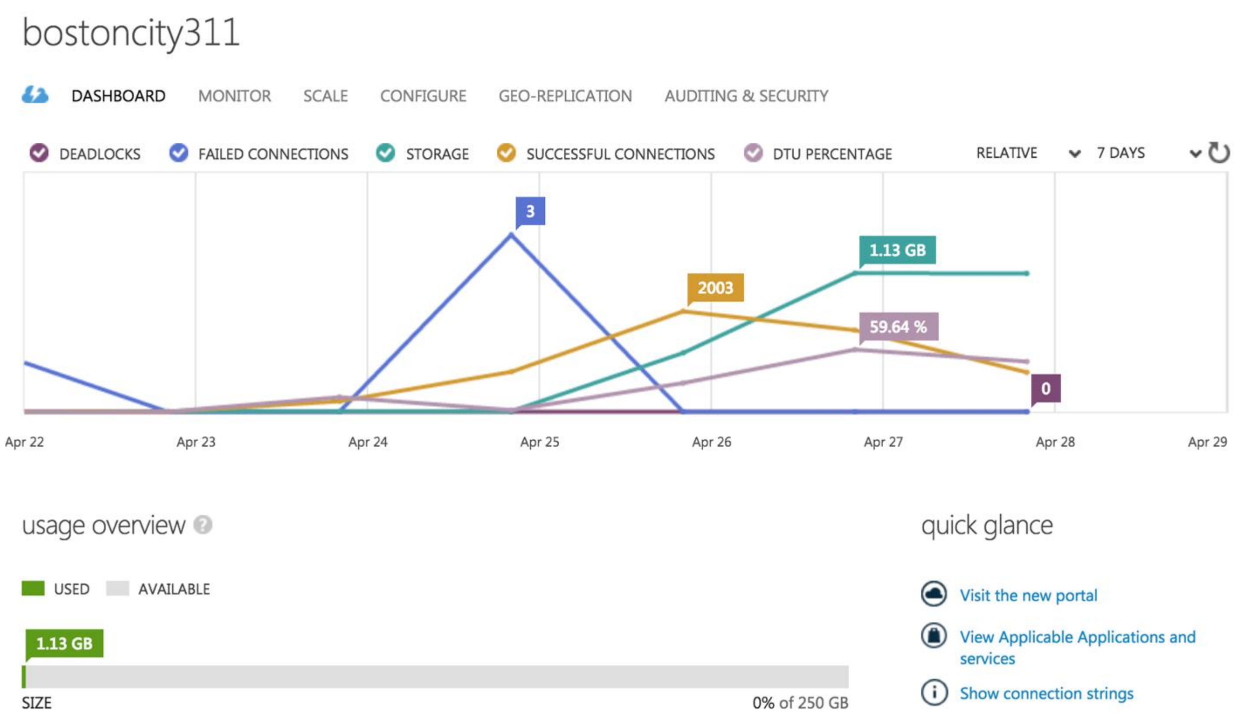


- SQL Database
  A SQL server **gg0lp2le4e** is created and a database **BostonCity311** is created inside the server.

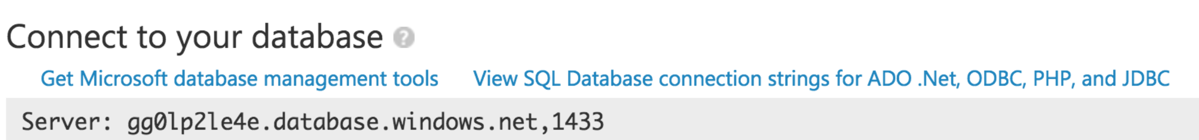
SQL Server



SQL Database



Connection String



CONCLUSION:

- From our website we can predict the number of days the issue will be solved by using AWS Framework for integrating the restful API generated from Azure Machine Learning

- We have developed a user friendly website giving business an ability to raise a request. Also view the data insights for better business decision.

.