Prediction of Call Arrival Times and Rates

Kaavish Report
presented to the academic faculty
by

Musabbir Abdul Majeed	musabbir.majeed
Syeda Saleha Raza	saleha.raza
Waqar Saleem	waqar.saleem
Abdul Samad	abdul.samad

In partial fulfillment of the requirements for
Bachelor of Science
Computer Science

Dhanani School of Science and Engineering

Habib University
Spring 2020

Prediction of Call Arrival Times and Rates

This Kaavish project was supervised by:

Mamoon ul Haq Haqqi	Syeda Saleha Raza	Zeehasham Rasheed
Lead Data Scientist	Faculty of CS	Faculty of CS
Afiniti	Habib University	Habib University

Approved by the Faculty of Computer Science on ______________________.

Dedication

For the love using of applied Artificial Intelligence to facilitate the call center industry using novel and innovative solutions.

Acknowledgements

We want to thank the faculty of Computer Science department from Habib University who allowed us to solve a real-world problem using innovative approaches. Additionally, we want to dedicate an honorable mention to **Dr. Syeda Saleha Raza** and **Dr. Zeehasham Rasheed** for their unmatched and unparalleled supervision and dedicated commitment for the project.

We also want to thank **Afiniti Software Solutions (Pvt.) Ltd.** for their continuous support and confident trust in us to develop state-of-the-art solutions for the problem under discussion.

Abstract

In an industry which requires precise allocation of resources, scheduling mechanisms play an integral role. Accurate scheduling offers benefits and satisfaction to businesses as the business leaders can safely control their service quality as well as the finances. Examples of such businesses include food delivery (such as Food Panda, Eat Mubarak, Home Chef, etc.), ride-hailing services (such as Careem, Uber, Bykea, Siayara, etc.), call centers, etc. All these industries can revolutionize customer experiences if provided with proper analytics which in turn allow them to perform scheduling operations effectively and precisely.

In this project, we plan to solve a pre-requisite problem for scheduling human resources in a **call center industry**. In a traditional call center, under-staffing leads to effecting services whereas over-staffing generates a high financial overhead. This problem can be solved/minimized with efficient prediction mechanisms which can be used to estimate call arrival times and rates.

In this project, we plan to solve the problems of arrival time for next call, determining the category of predicted call and predicting the count of calls expected to come in a given time-interval.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem Statement

This section explains the problem we are solving and our motivation to do so.

In an industry which requires precise allocation of resources, scheduling mechanisms play an integral role. Accurate scheduling offers benefits and satisfaction to businesses as the business leaders can safely control their service quality as well as the finances. Examples of such businesses include food delivery (such as Food Panda, Eat Mubarak, Home Chef, etc.), ride-hailing services (such as Careem, Uber, Bykea, Siayara, etc.), call centers, etc. All these industries can revolutionize customer experiences if provided with proper analytics which in turn allow them to perform scheduling operations effectively and precisely.

In this project, we plan to solve a pre-requisite problem for scheduling human resources in a **call center industry**. In a traditional call center, under-staffing leads to effecting services whereas over-staffing generates a high financial overhead. This problem can be solved/minimized with efficient prediction mechanisms which can be used to estimate call arrival times and rates.

We divide our problem set into three separate modules, stated below:

Module 1

When can we expect the next call to come in?

Module 2

Amongst well-defined possible categories, which type of call will it be?

Module 3

Within a defined interval, how many calls can we expect from a particular category of callers?

The problem is to devise a set of algorithms that is able to answer the above questions within a **reasonable degree of accuracy** which will be further clarified and finalized after exhaustive literature review.

The full specifications of any answer to the above problem, therefore, requires the following elements:

- One or more metrics that allow us to objectively measure how an algorithm or strategy performs in answering the above questions.
- A set of algorithms that is able to answer the above questions.
- A list of data elements/inputs that are required by the algorithms.

1.2 Proposed Solution

This section gives a summary of our proposed solution, i.e. how does it solve the problem? An overview of the system is provided. A detailed description of each module of the system is presented later in Chapter 1.

After a comprehensive review of industry trends and previous researches on the problem it became evident that most of the approaches deal with the following subset of our problem:

Within a defined interval, how many calls can we expect from a particular category of callers? (Module 3)

Very little work targets the following subset of our problem statements:

When can we expect the next call to come in? (Module 1)

and,

Amongst well-defined possible categories, which type of call will it be? (Module 2)

Furthermore, the problem of predicting call types or particular category of callers has not been addressed so extensively in the literature. Afiniti desires to upgrade its services by addressing the above problem and incorporating this innovative solution. Hence we claim this project to be novel.

Existing Solutions

The existing solutions found so far for each module are listed below:

Module 1

There are no convincing solutions found for this module. Since it is a time-series problem, the use of time-dependent approaches is inevitable. Very little work which was found did not apply time-series approaches on this module and hence convincing results were not produced.

Module 2

This module seems be a conventional classification problem with independent existence but in actual it is not that simple. Many classification approaches are available for independently existing datasets. In our case, this problem derives from Module 1 and its dataset is dependent on the results produced from Module 1. No convincing solutions were found for this specific case.

Module 3

An extensive literature for a problem close to Module 3 was found but it focuses on the problem statement *Within a defined interval, how many calls can we expect?* instead of *Within a defined interval, how many calls can we expect* ***from a particular category of callers****?* The available solutions are listed below:

- K – Nearest Neighbor Algorithm
- Mixed Models/Bayesian Gaussian Model
- Multiplicative Model
- Additive Model
- Seasonal ARIMA and Regression with ARIMA Errors

Proposed Solutions

Some, out of multiple, solutions we propose for each module are listed below:

Module 1

- Sort First Approach

Module 2

- Chained Approach

Module 3

We propose the use of existing solutions for this module by dividing the dataset into n parts such that n is the number of categories from Module 2 and all the entries in each part belong to the same category. After that, we individually compute the results for all n categories and analyze them. Moreover, following other approaches can also be used as a sub-part to develop new solutions.

- Post Estimate Technique
- Averaging Approaches
- Dynamic Regression Models
- Exponential Smoothing
- Holt's Linear Model
- Holt-Winter Trend and Seasonality Model
- Regression Analysis
- Multiple Temporal Aggregation
- Neural Networks

For detailed explanation about these solutions, please see https://urlzs.com/LyA6B.

1.3 Intended User

This section outlines the target users of this system. The different types of users in our user base and their interaction with the system are described briefly.

The users of our system are the management and administration personnel from a call center. They interact with our system to view the prediction results and then make appropriate decision for the scheduling and management of caller agents in the call center.

1.4 Key Challenges

This section mentions the key challenges that we foresee in this project and possible ways to address them.

Following is the list of resources we require in order to smoothly carry out this project and are available.

- Dataset (live dataset provided by Afiniti).
- Efficient data manipulation libraries.
- Efficient analysis and plotting libraries.

Following is the list of resources we require in order to smoothly carry out this project and are NOT available.

- Fast-paced computer machine.
- GPU.
- Miscellaneous expenses (PKR 10,000/- which includes frequent trips to Afiniti).

We request the availability of above resources as soon as they can be arranged. If not, it can serve as a bottle-neck for various approaches we plan to solve the problem with. Once notified, we can prioritize the approaches which do not extensively involve above mentioned resources accordingly. ETA we expect for above resources is max 2 months.

2. Literature Review

This chapter presents the current state of the art in the domain and talks about other similar work that has been done in this area. It also establishes the novelty of our work by highlighting the differences between the existing work and our work.

We will keep updating this chapter as our research proceeds and we come across more work related to our problem.

The project is divided into three modules in total; *Call Time Predictor*, *Call Category Predictor* and *Call Count Predictor*. Considering the fact that the nature of problems addressed in all three modules is different, the process of literature review for them is separate and independent.

Call Time Predictor

For the given problem of predicting the time for next call, there is no vast literature available to solve the problem. The work which we do adds to the novelty of this research problem.

A Poisson Process is a model for a series of discrete event where the average time between events is known, but the exact timing of events is random. The arrival of an event is independent of the event before. For example, suppose we own a website which our content delivery network (CDN) tells us goes down on average once per 60 days, but one failure doesn't affect the probability of the next. All we know is the average time between failures. Poisson process modeling is one of the potential ways to explore in the domain call arrival process. [**1**] discuss a similar kind of an approach but based on the call arrival count rather than the call arrival rate itself. Another proposed approach for predicting the starting time of the next unobserved activity is to model activities occurring at a variable rate using a Log-Gaussian Cox Process (LGCP) and learn the rate function from the training data. Then the starting time is predicted using importance sampling algorithm [**14**]. [**8**] redefines the problem at a personalised level for intelligent phones instead of call centers and proposes probabalistic models based on the paramters of *caller's behavior* and *reciprocity.*

Soon after the recent popularity of Long Short-Term Memory networks, the domain of time-series prediction captured the highlights of its efficiency and started proposing time-series models based on LSTMs. In a recent paper published by Uber, they used LSTM network to solve the problem of predicting the time for next ride. With the use of LSTM model, they minimised the risk of the contribution of exogenous variables while training. Earlier, instead of using bayesian deep networks with LSTM cells, Uber used probabalistic time-series models which had its own demerits especially in cases of frequent training and eliminating exogenous variables [**15**].

Holt-Winters method introduces the concept of seasonality and trends in the time-series models to map the data according to its underlying trends. [**10**] discusses the use of Holt-Winters method as a call prediciton model for frequent and periodic callers. Some papers discuss about solving a similar kind of a problem by using queuing theory. In order to apply queuing theory on such problems, it is necessary to decompose the problem into three fundamental concepts: arrival, customer patience and service durations [**13**].

Call Category Predictor

The literature review for this module is under process. This module will be handled in Sprint 3 and hence proper review will be written and updated by then.

Call Count Predictor

For the given problem of predicting the time for next call, there is significant literature available to solve the problem. The researchers in the domain have vastly contributed to solving the problem.

Bayesian forecasting of inhomogeneous poisson process contributes to solving the problem of call count with in a given interval of time using correlation and autoregressive techniques. [**1**]. In the presence of intraday and interday dependence in call arrival rates, standard time series models may be applied for forecasting call arrivals, for example autoregressive integrated moving average (ARIMA) models and exponential smoothing [**2**]. A doubly stochastic Poisson process can be viewed as a two-step randomization: a stochastic process (for the arrival rate) is used to generate another stochastic process (for the call arrival process) by representing its intensity [**2**].

[**4**] discusses Point Forecast and Distributional Forecast approaches of predicting the call count with in a given time interval. Moreover a bayesian gaussian model, multiplicative model and additive model can be effectively used for forecasting as well. A modulated Poisson process model [**5**] to describe and analyze arrival data to a call center. The attractive feature of this model is that it takes into account both covariate and time effects on the call volume intensity and in so doing enables us to assess the effectiveness of different advertising strategies along with predicting the arrival patterns. A Bayesian analysis of the model is developed and an extension of the model is presented to describe potential heterogeneity in arrival patterns. The proposed model and the methodology are implemented using real call center arrival data.

The use of nearest neighbour algorithm for forecasting call arrivals to call centers can also contribute to solving the question under discussion. The algorithm does not require an underlying model for the arrival rates and it can be applied to historical data without pre-processing it. The nearest neighbour algorithm with the Pearson correlation distance function is also able to take correlation structures, that are usually found in call center data, into account. [**6**].

In [**11**], the methods considered include seasonal Autoregressive Integrated Moving Average (ARIMA) and regression with ARIMA errors. Two models out of different candidate models are identified through statistical inferences of the model parameters, diagnostic checking and fit indices by using the training set for the original series and logged series. Forecasting performances of the models are evaluated with one-step ahead forecasts by using the test set and it is concluded that suggested models adequately predict hourly calls of the call centre.

Moreover, the approaches discussed for module 1 can be altered and tweaked in a way that they contribute to solving this problem of call arrival count instead call arrival rate.

References

1. Weinberg, J., Brown L.D. and Stroud J.R. *Bayesian Forecasting of an Inhomogeneous Poisson Process with Applications to Call Center Data*, 2006.

2. Ibrahim R., Ye H., L'Ecuyer P. and Shen H. *Modeling and forecasting call center arrivals: A literature survey and a case study*, 2016.

3. Ibrahim R., Ye H., L'Ecuyer P. and Shen H. *On the modeling and forecasting of call center arrivals*, 2015.

4. Shen H. and Huang J.Z. *Interday Forecasting and Intraday Updating of Call Center Arrivals.*

5. Soyer R. and Tarimcilar M.M. *Modeling and Analysis of Call Center Arrival Data: A Bayesian Approach.*

6. Bhulai S, W.H. Kan and Marchiori E. *Nearest Neighbour Algorithms for Forecasting Call Arrivals in Call Centers.*

7. Ibrahim R., L'Ecuyer P., Regnard N. and Shen H. *On The Modeling and Forecasting of Call Center Arrivals*, 2012.

8. Phithakkitnukoon S. and Dantu R. *Predicting Calls – New Service for an Intelligent Phone.*

9. Koen van den Bergh. *Predicting Call Arrivals in Call Centers*, 2006.

10. Sennaroglu B. and Polat G. *Time Series Forecasting for a Call Centre*, 2017.

11. Noiman S.A., Feigin P.D. and Mandelbaum A. *Workload Forecasting For a Call Center: Methodolgy and a Case Study*, 2009.

12. Brown L., Gans N, Mandelbaum A., Sakov A., Shen H., Zeltyn S. and Linda Zhao. *Statistical Analysis of a Telephone Call Center*, 2013.

13. Mahmud T., Hasan M., Chakraborty A., and Chowdhury A.R. *A Poisson Process Model for Activity Forecasting*, 2016.

14. Zhu L. and Laptev N. *Deep and Confident Prediction for Time Series at Uber*, 2017.

15. Zhu L. and Laptev N. *Deep and Confident Prediction for Time Series at Uber*, 2017.

3. Software Requirement Specification (SRS)

This chapter provides detailed specifications of the system under development.

3.1 Functional Requirements

This section describes each function/feature provided by our system. These functions are logically grouped into modules based on their purpose (as per our system).

- **Module 1:**
  This module contains functions and features to develop a part of our core engine and answer the following question from the scope of this project. *When can we expect the next call to come in?*
  - The system should be able to entail a set of prediction algorithms for Module 1.
  - The system should be able to take data as input from any third party (user/module/API/library).
  - The system should be able to output a continuous 'call duration' quantity for each prediction algorithm of Module 1.
  - The system should be able to output an accuracy/validation score against each prediction algorithm of Module 1.
  - The system should be able to provide a detailed documentation for each prediction algorithm of Module 1.
- **Module 2:**
  This module contains functions and features to develop a part of our core engine

and answer the following question from the scope of this project. *Amongst well-defined possible categories, which type of call will it be?*

  - The system should be able to entail a set of prediction algorithms for Module 2.
  - The system should be able to take data as input from any third party (user/module/API/library).
  - The system should be able to output a discrete 'call category' for each prediction algorithm of Module 2.
  - The system should be able to output an accuracy/validation score against each prediction algorithm of Module 2.
  - The system should be able to provide a detailed documentation for each prediction algorithm of Module 2.

- **Module 3:**
  This module contains functions and features to develop a part of our core engine and answer the following question from the scope of this project. *Within a defined interval, how many calls can we expect from a particular category of callers?*
  - The system should be able to entail a set of prediction algorithms for Module 3.
  - The system should be able to take data as input from any third party (user/module/API/library).
  - The system should be able to output a continuous 'call count' quantity for each prediction algorithm of Module 3.
  - The system should be able to output an accuracy/validation score against each prediction algorithm of Module 3.
  - The system should be able to provide a detailed documentation for each prediction algorithm of Module 3.

- **Module 4:**
  This module contains functions and features to implement the dashboard component and manage user interactivity/engagement in this system.
  - The system should be able to support 'login' and 'register' functionalities.
  - The system should be able to take data as CSV from the user.

  - The system should be able to display insights from data to the user.
    * The system should be able to display numeric insights to the user.
    * The system should be able to display graphical insights to the user.
    * The system should be able to display Exploratory Data Analysis (EDA) to the user.
  - The system should be able to allow the user to select analysis type (between Module 1, Module 2 and Module 3).
  - The system should be able to maintain a history of records on which analysis operations were performed.
  - The system should be able to ask the user to select a particular algorithm for analysis.
  - The system should be able to show the results of analysis on the uploaded data to the user.
  - The system should be able to show the accuracy/validation scores of all the analysis performed on the data.

3.2 Non-functional Requirements

This sections mentions the specific non-functional requirements of our system. These generally address performance, scalability and deployment.

- **Performance**
  - The system should be able to maintain a response time of 1000 ms at max for API requests.
  - The system should be able to maintain a response time of 10000 ms at max for model training requests.
- **Scalability**
  - The system should be modular. There should be as much reusable components and no code replication should be appreciated.
  - The system should follow proper design and architectural patterns (where necessary).
- **Deployment**

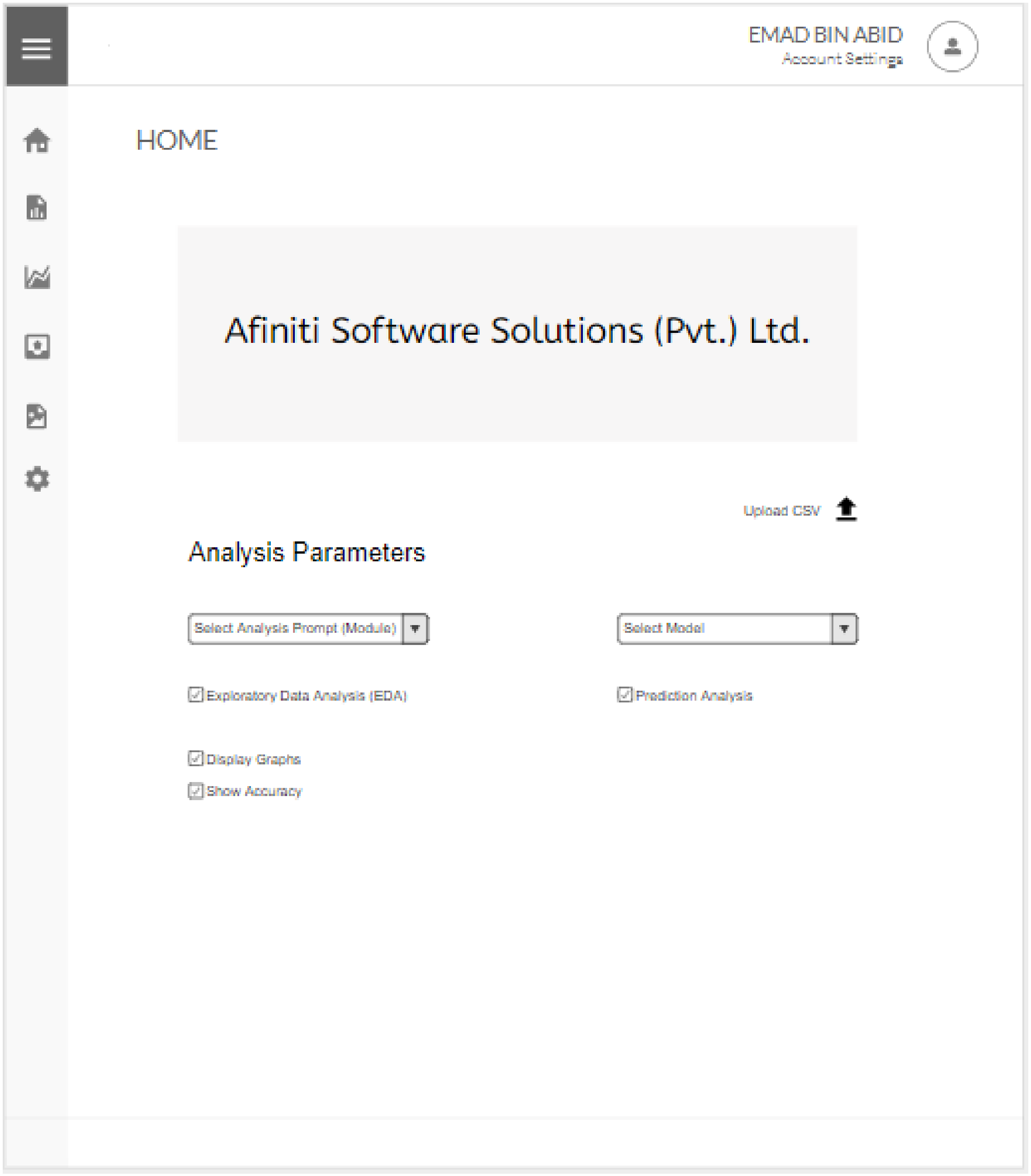
- The system should be easily and efficiently deployable in client's environment with minimum command usage.
- The deployment package should be less in size. ".whl" files for deployment are preferred.

3.3 External Interfaces

3.3.1 User Interfaces

This section includes our mockup screens and briefly explains them. The interface, overall, has three screens; **Home**, **EDA Analysis**, **PA Analysis**. Please note that the dashboard and its related screens and operations are subject to change in any sprint as per the client's requirements.

Screen 1

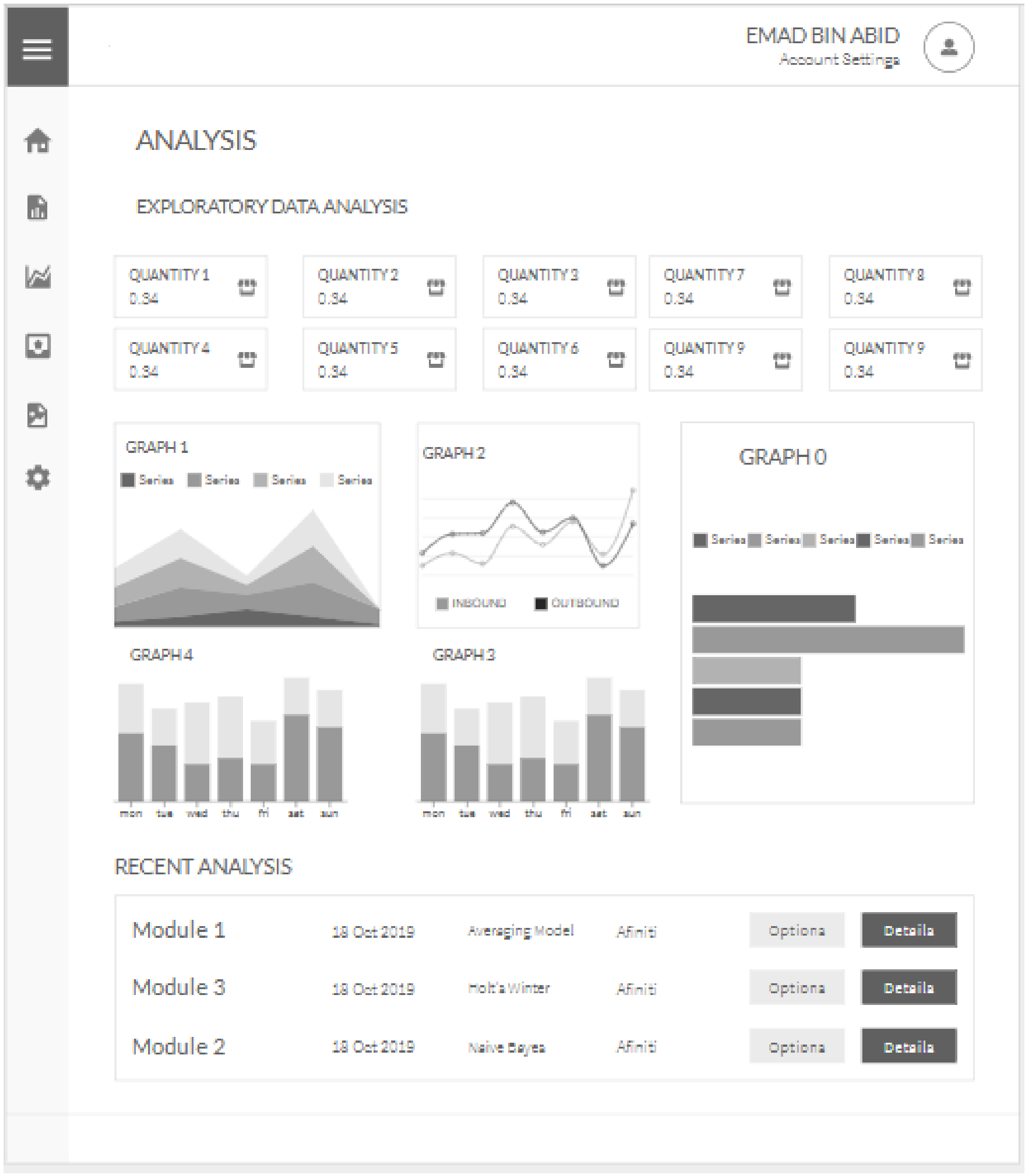


Home Page of Dashboard

The Home Page briefly greets the user by showing the company to which the user belongs. It then has various input expectations from the user which are discussed as follows.

- **Upload CSV:** The user is able to upload a CSV file on which the analysis needs to be done. There will be some validation checks on the uploaded CSV and the user will be notified of its acceptance or rejection (with error messages) accordingly.
- **Select Analysis Prompt (Module):** This dropdown allows the user to select the type of analysis i.e. Module 1, Module 2 or Module 3.
- **Select Model:** This dropdown allows the user to select the specific model for analysis purposes.
- **Exploratory Data Analysis (EDA):** The checkbox allows the user to view (or not view) the EDA of the uploaded data.
- **Prediction Analysis:** The checkbox allows the user to view (or not view) the PA of the uploaded data.
- **Display Graphs:** The checkbox allows the user to view (or not view) the graphs of EDA and PA analyses.
- **Show Accuracy:** The checkbox allows the user to view (or not view) the accuracy or validation scores of the model for which the analysis is performed.

Screen 2

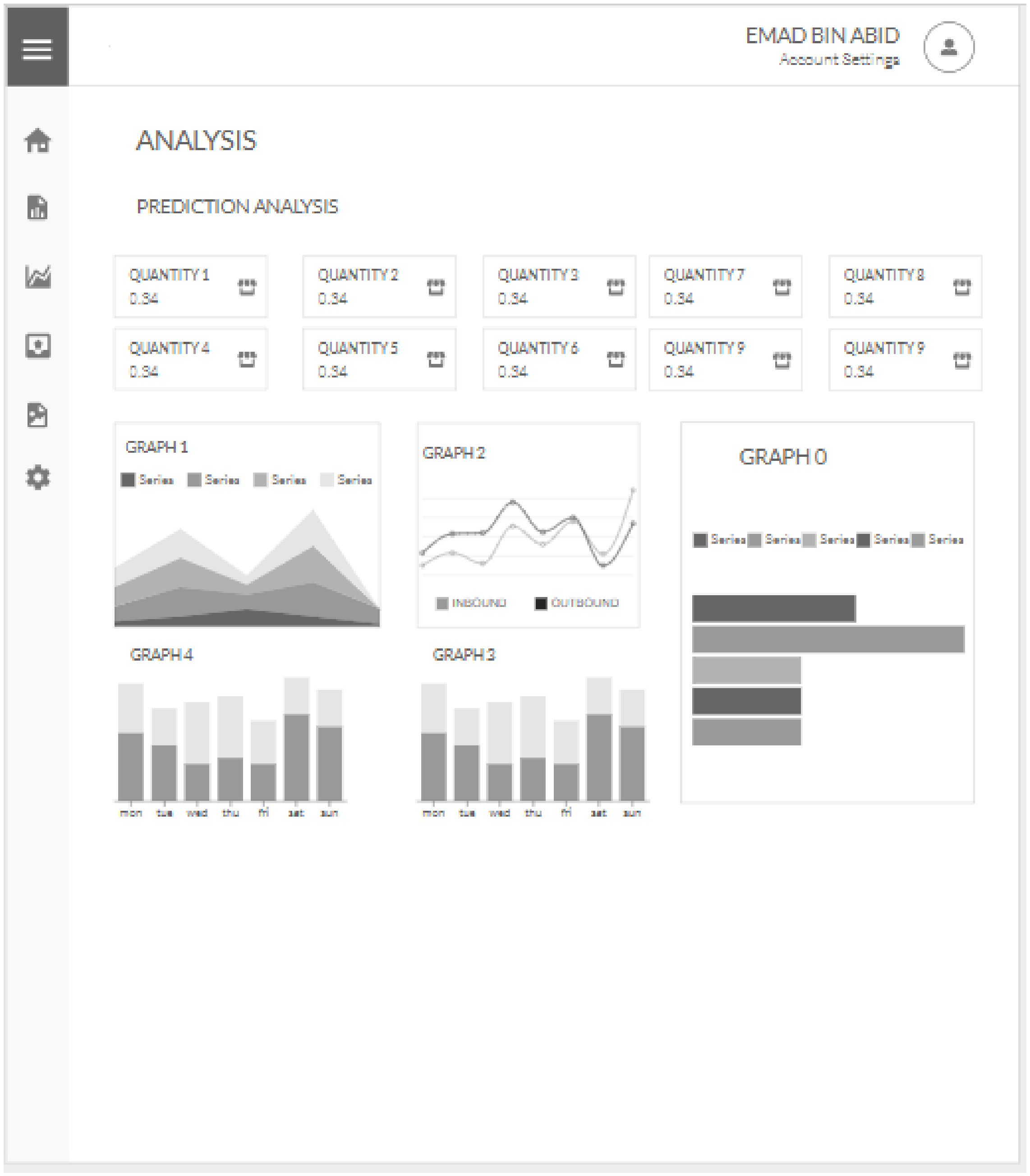


Module 1	18 Oct 2019	Averaging Model	Afiniti	Options	Details
Module 3	18 Oct 2019	Holt's Winter	Afiniti	Options	Details
Module 2	18 Oct 2019	Naive Bayes	Afiniti	Options	Details

Analysis Page (EDA) of Dashboard

The EDA Analysis Page shows the results of analysis on the basis of user selected modules, models and other preferences. It generally displays some numeric quantities and graphs to show the exploratory analysis on the data which is provided by the user. The names and types of quantities and graphs are anonymous for now and will be decided after a review meeting with the client.

Screen 3



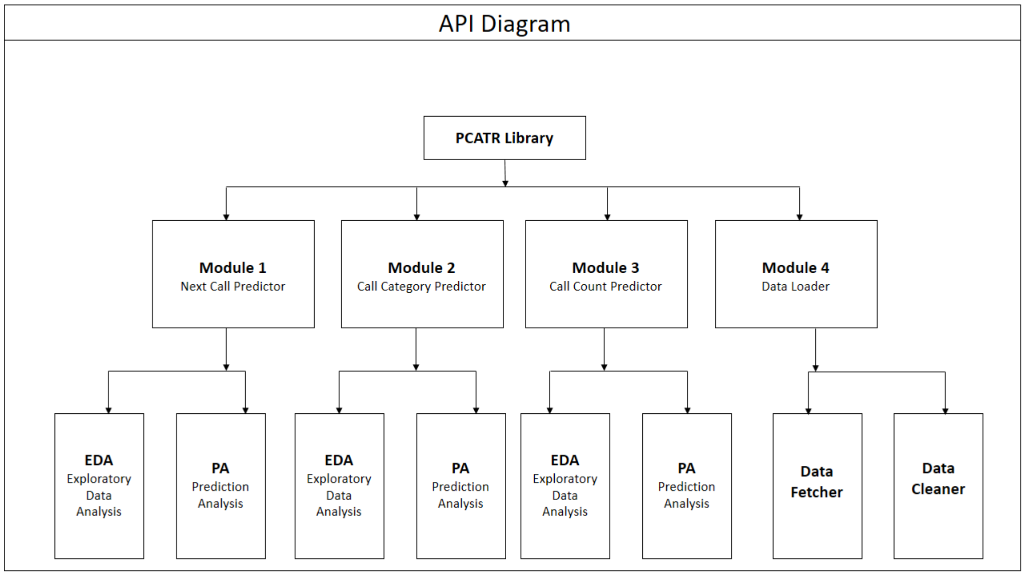
Analysis Page (PA) of Dashboard

The PA Analysis Page shows the results of analysis on the basis of user selected modules, models and other preferences. It generally displays some numeric quantities and graphs to show the prediction analysis on the data which is provided by the user. The names and types of quantities and graphs are anonymous for now and will be decided after a review meeting with the client.

3.3.2 Application Program Interface (API)

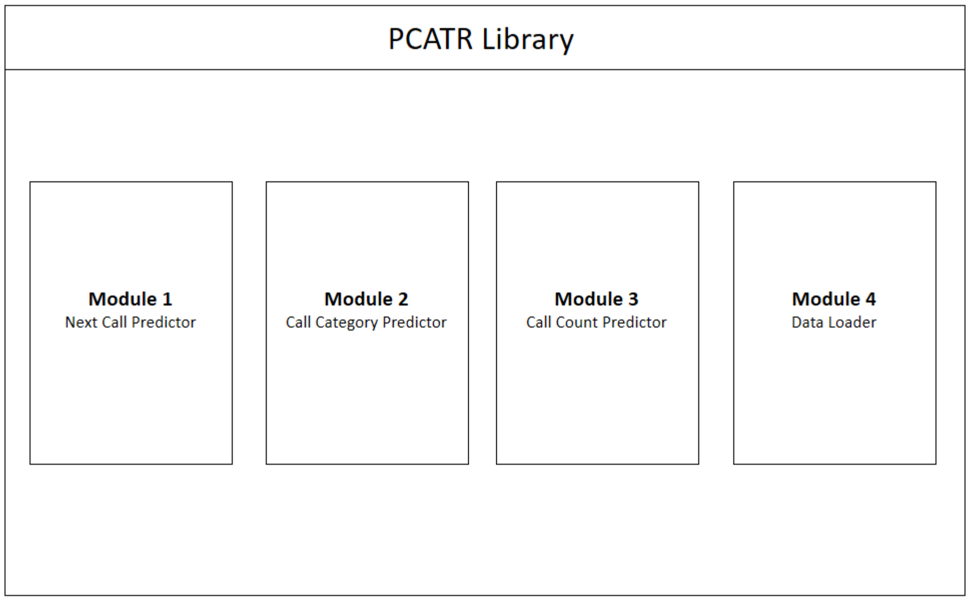
This section describes the library or API interface to our system.

The main library to this system is divided into four modules to cater Module 1, Module 2, Module 3 and data loading/manipulation operations. The API/library flow diagram of our system is given below followed by a brief documentation of each entailed module.

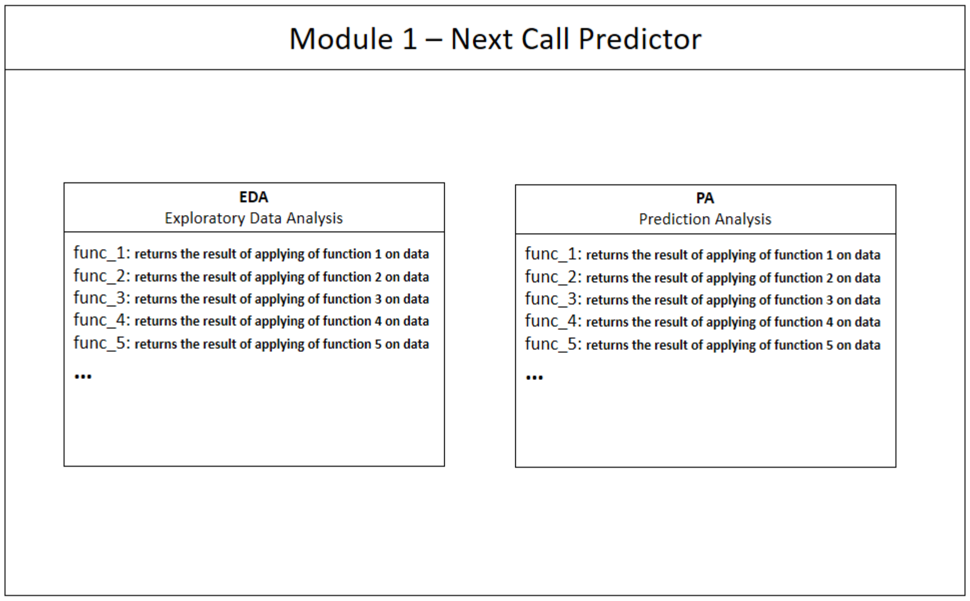


API Diagram of System

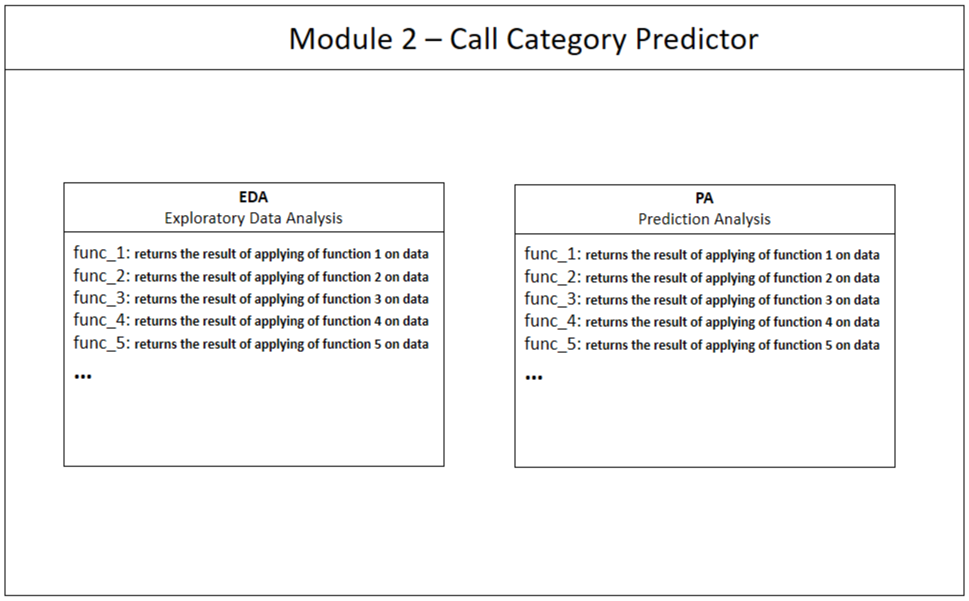
Following is the zoomed in version of system explaining each module and its functionality separately.



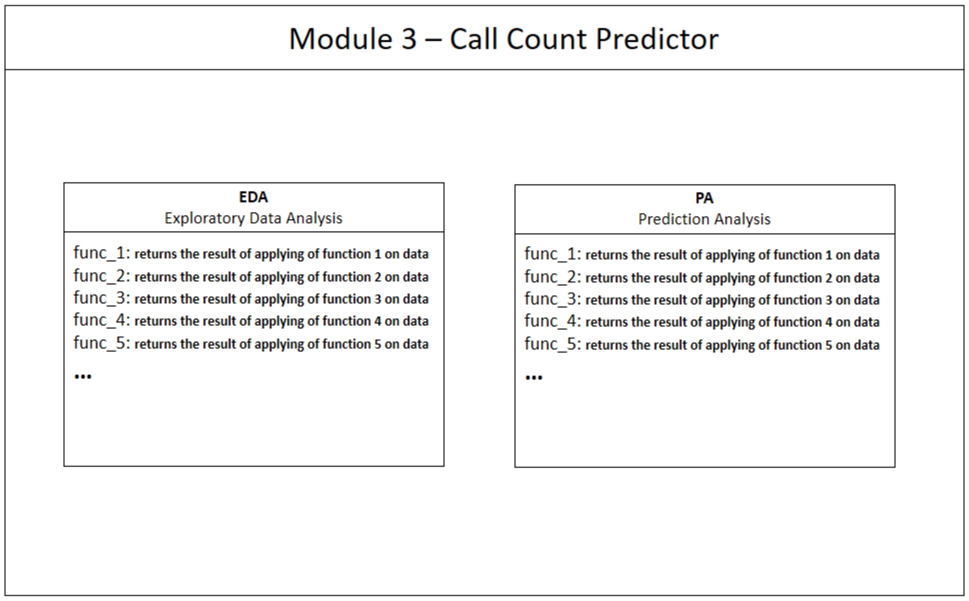
Higher Level Library Structure


Module 1 – Next Call Predictor
EDA
Exploratory Data Analysis
func_1: returns the result of applying of function 1 on data
func_2: returns the result of applying of function 2 on data
func_3: returns the result of applying of function 3 on data
func_4: returns the result of applying of function 4 on data
func_5: returns the result of applying of function 5 on data
...
PA
Prediction Analysis
func_1: returns the result of applying of function 1 on data
func_2: returns the result of applying of function 2 on data
func_3: returns the result of applying of function 3 on data
func_4: returns the result of applying of function 4 on data
func_5: returns the result of applying of function 5 on data
...

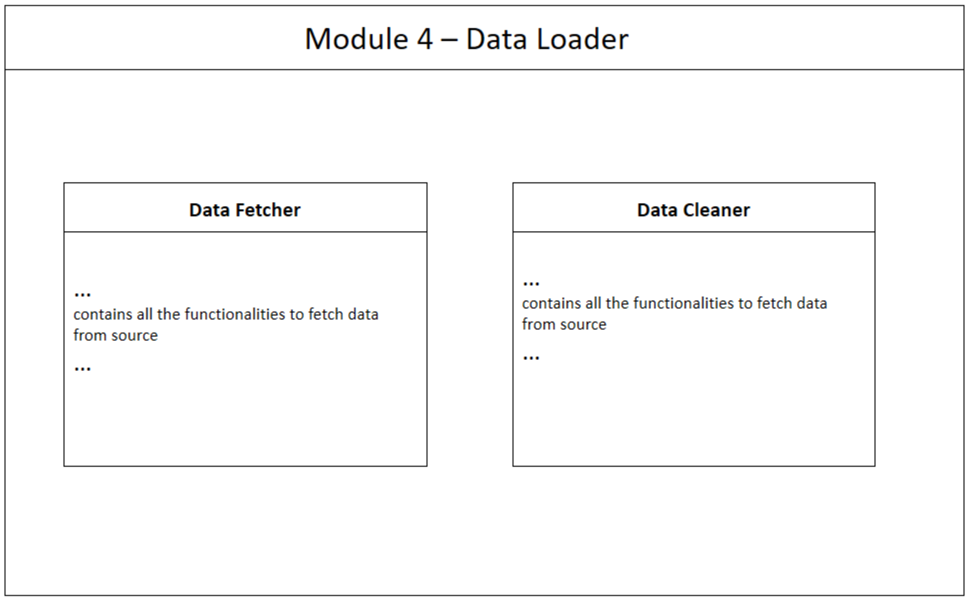
Next Call Predictor


Module 2 – Call Category Predictor
EDA
Exploratory Data Analysis
func_1: returns the result of applying of function 1 on data
func_2: returns the result of applying of function 2 on data
func_3: returns the result of applying of function 3 on data
func_4: returns the result of applying of function 4 on data
func_5: returns the result of applying of function 5 on data
...
PA
Prediction Analysis
func_1: returns the result of applying of function 1 on data
func_2: returns the result of applying of function 2 on data
func_3: returns the result of applying of function 3 on data
func_4: returns the result of applying of function 4 on data
func_5: returns the result of applying of function 5 on data
...

Call Category Predictor


Module 3 – Call Count Predictor
EDA
Exploratory Data Analysis
func_1: returns the result of applying of function 1 on data
func_2: returns the result of applying of function 2 on data
func_3: returns the result of applying of function 3 on data
func_4: returns the result of applying of function 4 on data
func_5: returns the result of applying of function 5 on data
...
PA
Prediction Analysis
func_1: returns the result of applying of function 1 on data
func_2: returns the result of applying of function 2 on data
func_3: returns the result of applying of function 3 on data
func_4: returns the result of applying of function 4 on data
func_5: returns the result of applying of function 5 on data
...

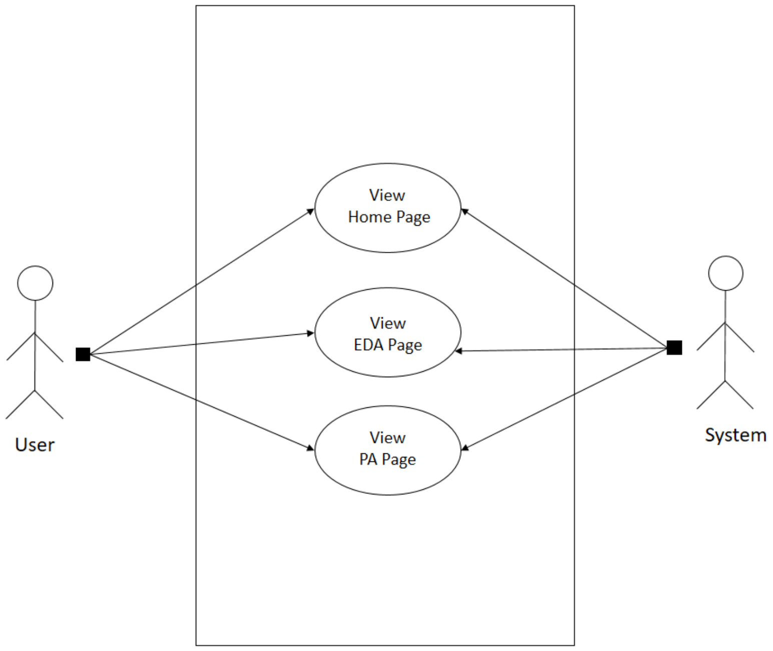
Call Count Predictor

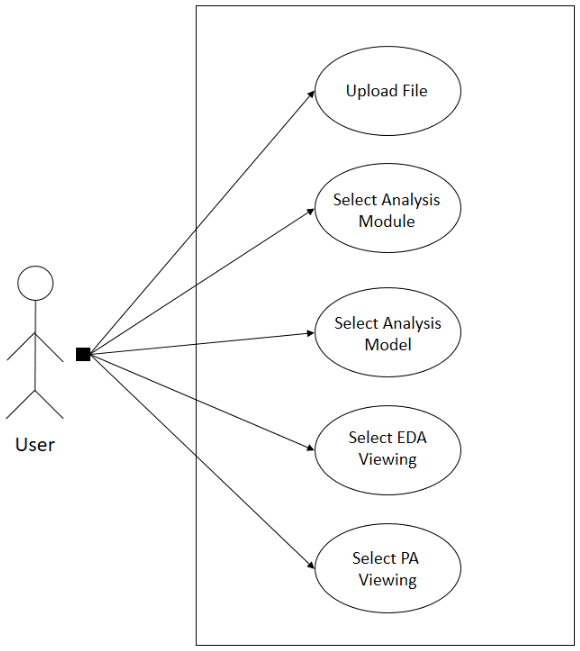

Module 4 – Data Loader
Data Fetcher
...
contains all the functionalities to fetch data from source
...
Data Cleaner
...
contains all the functionalities to fetch data from source
...

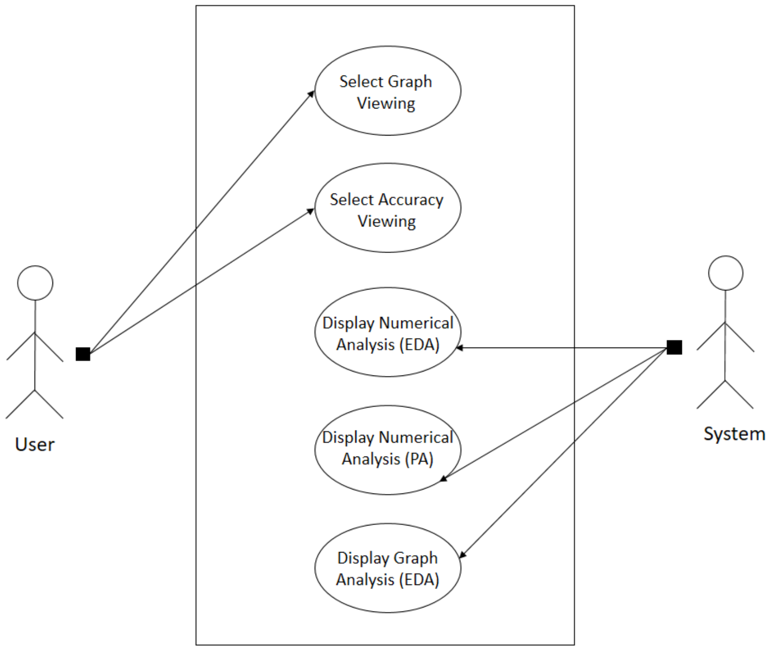
Data Loader

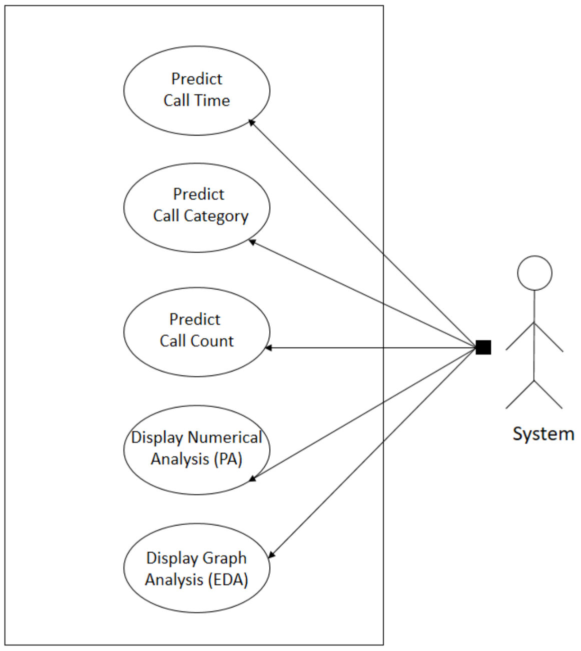
3.4 Use Cases

This section presents detailed use cases of our system.


View Home Page
View EDA Page
View PA Page
User
System


Upload File
Select Analysis Module
Select Analysis Model
Select EDA Viewing
Select PA Viewing
User

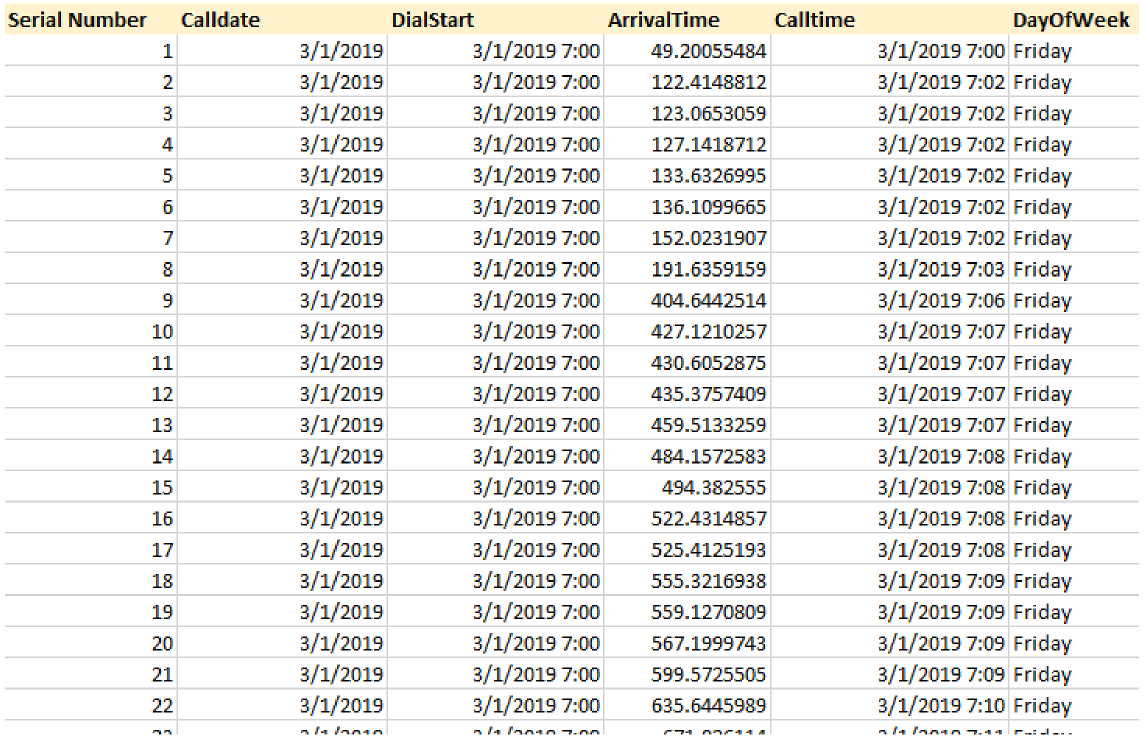

Select Graph Viewing
Select Accuracy Viewing
Display Numerical Analysis (EDA)
Display Numerical Analysis (PA)
Display Graph Analysis (EDA)
User
System


Predict Call Time
Predict Call Category
Predict Call Count
Display Numerical Analysis (PA)
Display Graph Analysis (EDA)
System

3.5 Datasets

This section describes the specific datasets used to build our system. The live datasets are provided by Afiniti for training purposes. The main dataset is divided into three parts in order to address the three core modules separately. As received from client in first (current) phase, a snapshot of the dataset for Module 1 is also included. Further details, when needed, are presented in the appendix.

- **Dataset 1:**
  This dataset will be used to train the models for the query being addressed in Module 1. Further specifications are given below.
  - A time-series dataset.
  - Class label is a continuous quantity i.e. call duration.
  - The dataset features will be used to train the model to predict the number of seconds after which the next call is expected to come.
- **Dataset 2:**
  This dataset will be used to train the models for the query being addressed in Module 2. Further specifications are given below.
  - A time-series dataset.
  - Class label is a discrete quantity i.e. call category.
  - The dataset features will be used to train the model to predict the category of call that was predicted in Module 1.
- **Dataset 3:**
  This dataset will be used to train the models for the query being addressed in Module 3. Further specifications are given below.
  - A time-series dataset.
  - Class label is a continuous quantity i.e. number of calls, but, can be modeled as discrete classes for various models after literature review for Module 3.
  - The dataset features will be used to train the model to predict the number of calls we can expect from a particular category of callers.

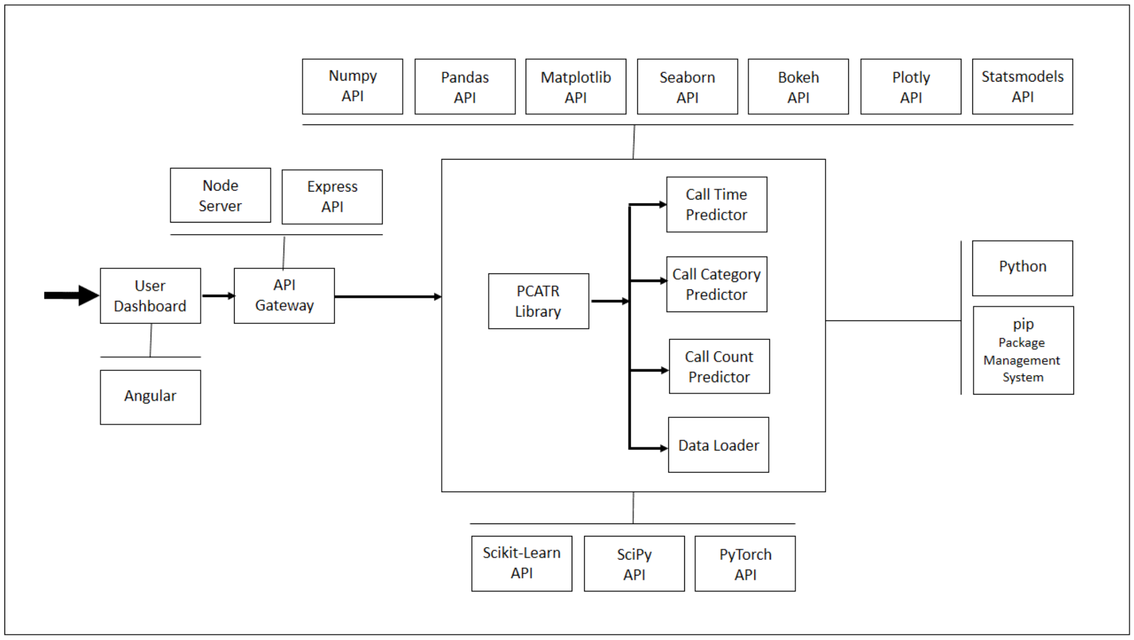


Serial Number	Calldate	DialStart	ArrivalTime	Calltime	DayOfWeek
1	3/1/2019	3/1/2019 7:00	49.20055484	3/1/2019 7:00	Friday
2	3/1/2019	3/1/2019 7:00	122.4148812	3/1/2019 7:02	Friday
3	3/1/2019	3/1/2019 7:00	123.0653059	3/1/2019 7:02	Friday
4	3/1/2019	3/1/2019 7:00	127.1418712	3/1/2019 7:02	Friday
5	3/1/2019	3/1/2019 7:00	133.6326995	3/1/2019 7:02	Friday
6	3/1/2019	3/1/2019 7:00	136.1099665	3/1/2019 7:02	Friday
7	3/1/2019	3/1/2019 7:00	152.0231907	3/1/2019 7:02	Friday
8	3/1/2019	3/1/2019 7:00	191.6359159	3/1/2019 7:03	Friday
9	3/1/2019	3/1/2019 7:00	404.6442514	3/1/2019 7:06	Friday
10	3/1/2019	3/1/2019 7:00	427.1210257	3/1/2019 7:07	Friday
11	3/1/2019	3/1/2019 7:00	430.6052875	3/1/2019 7:07	Friday
12	3/1/2019	3/1/2019 7:00	435.3757409	3/1/2019 7:07	Friday
13	3/1/2019	3/1/2019 7:00	459.5133259	3/1/2019 7:07	Friday
14	3/1/2019	3/1/2019 7:00	484.1572583	3/1/2019 7:08	Friday
15	3/1/2019	3/1/2019 7:00	494.382555	3/1/2019 7:08	Friday
16	3/1/2019	3/1/2019 7:00	522.4314857	3/1/2019 7:08	Friday
17	3/1/2019	3/1/2019 7:00	525.4125193	3/1/2019 7:08	Friday
18	3/1/2019	3/1/2019 7:00	555.3216938	3/1/2019 7:09	Friday
19	3/1/2019	3/1/2019 7:00	559.1270809	3/1/2019 7:09	Friday
20	3/1/2019	3/1/2019 7:00	567.1999743	3/1/2019 7:09	Friday
21	3/1/2019	3/1/2019 7:00	599.5725505	3/1/2019 7:09	Friday
22	3/1/2019	3/1/2019 7:00	635.6445989	3/1/2019 7:10	Friday

Snapshot of Module 1's dataset provided by Afiniti

3.6 System Diagram

This diagram gives a high-level view of the different components of our system and the interactions between them. Each component and the particular tools/technologies/libraries used to build it are described.



Higher Level System Diagram

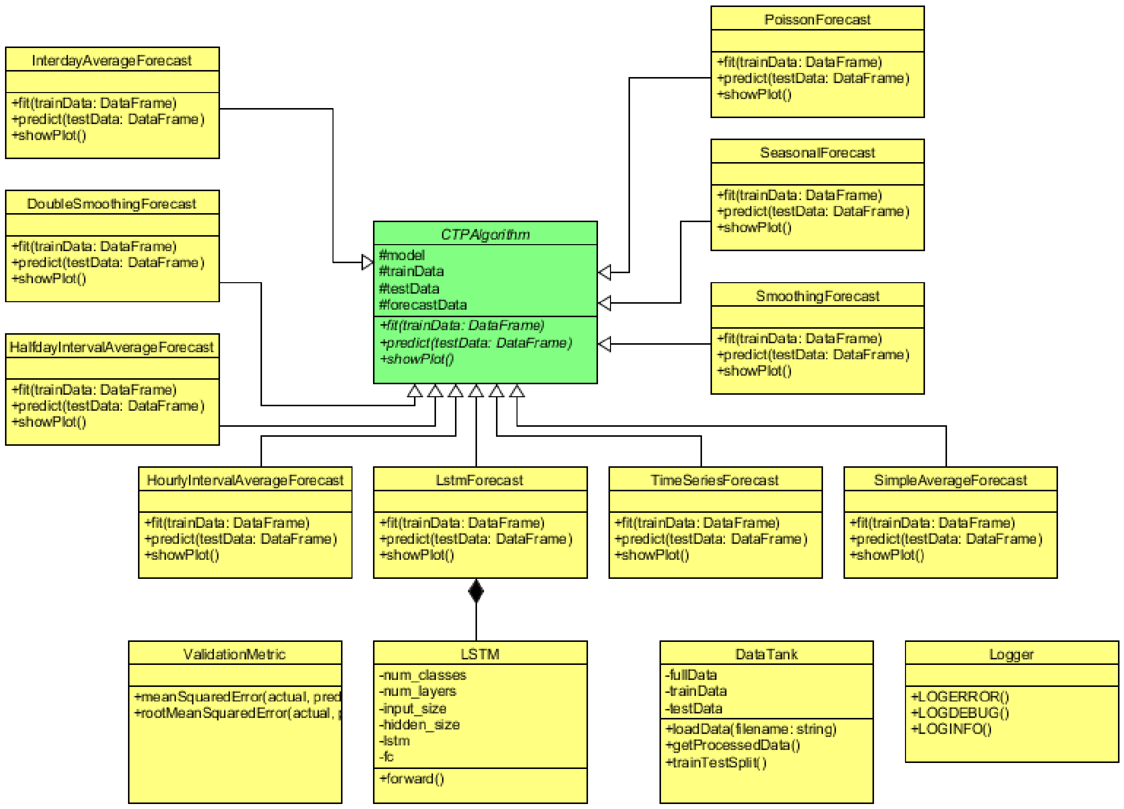
4. Software Design Specification (SDS)

This chapter provides important artifacts related to design of our project.

4.1 Software Design

This section presents the UML class diagram and gives a brief description of each class in our system. Attributes and methods of each class and relationship among classes are clearly presented.

In module 1 of our system, we present *CTPAlgorithm* which is an abstract class. It contains three virtual functions which must be implemented by child classes; namely *fit()*, *predict()* and *showPlot()*. The core algorithm classes, *InterdayAverageForecast*, *DoubleSmootingForecast*, *HalfdayIntervalAverageForecast*, *HourlyIntervalAverageForecast*, *LstmForecast*, *TimeSeriesForecast*, *SimpleAverageForecast*, *SmootingForecast*, *PoissonForecast* and *SeasonalForecast* inherit from the base *CTPAlgorithm* class. A class named *LSTM* is implemented to support algorithmic operations in *LstmForecast* class. In addition to that *ValidationMetric*, *Logger* and *DataTank* classes are implemented for validating results, logging and data preprocessing operations respectively.


InterdayAverageForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
DoubleSmoothingForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
HalfdayIntervalAverageForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
CTPAlgorithm
#model
#trainData
#testData
#forecastData
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
PoissonForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
SeasonalForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
SmoothingForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
HourlyIntervalAverageForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
LstmForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
TimeSeriesForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
SimpleAverageForecast
+fit(trainData: DataFrame)
+predict(testData: DataFrame)
+showPlot()
ValidationMetric
+meanSquaredError(actual, pred
+rootMeanSquaredError(actual, p
LSTM
-num_classes
-num_layers
-input_size
-hidden_size
-lstm
-fc
+forward()
DataTank
-fullData
-trainData
-testData
+loadData(filename: string)
+getProcessedData()
+trainTestSplit()
Logger
+LOGERROR()
+LOGDEBUG()
+LOGINFO()

4.2 Technical Details

Our project does not have persistent data so we have no ERD. Instead we explain here the technical details of the algorithms we use. These include the inputs and the outputs, how and where these algorothms fit in our tool chain, the techniques used in these algorithms, etc.

For detailed review, please see **pcatr-docs-v0.0.1.pdf** here

5. Experiments and Results

Module 1

When can we expect the next call to come in?

Holt Winters

Holt-Winters is a model of time series behavior. Forecasting always requires a model, and Holt-Winters is a way to model three aspects of the time series: a typical value (average), a slope (trend) over time, and a cyclical repeating pattern (seasonality). Holt-Winters uses exponential smoothing to encode lots of values from the past and use them to predict "typical" values for the present and future.

The three aspects of the time series behavior—value, trend, and seasonality—are expressed as three types of exponential smoothing, so Holt-Winters is called triple exponential smoothing. The model predicts a current or future value by computing the combined effects of these three influences. The model requires several parameters: one for each smoothing (α, β, γ), the length of a season, and the number of periods in a season.

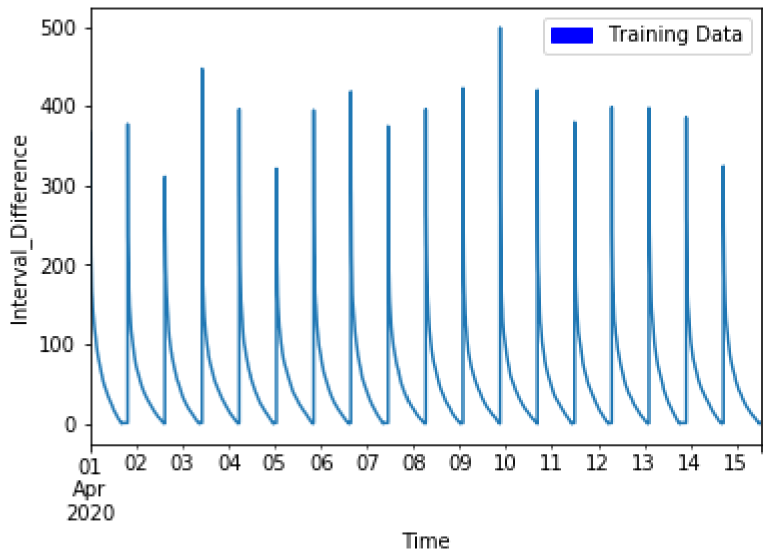
Seasonality can be confusing. A season is a fixed length of time that contains the full repetition. Within the season, there are periods, which is the granularity of prediction. If you want to model a value for every hour of every day within a week, your season is 168 hours long and your period is 1 hour.

The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations — one for the level $\boldsymbol{l}$, one for trend b_t and one for the seasonal component denoted by s_t, with smoothing parameters α, β and γ.

$$
\begin{aligned}
\text{level} \quad L_t &= \alpha(y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1}); \\
\text{trend} \quad b_t &= \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}, \\
\text{seasonal} \quad S_t &= \gamma(y_t - L_t) + (1-\gamma)S_{t-s} \\
\text{forecast} \; F_{t+k} &= L_t + kb_t + S_{t+k-s},
\end{aligned}
$$

Modelling our Problem We know several things about the data that we have. We know that the time durations between successive calls on any day are randomly distributed. This means that there is no pattern within the data that time series method can rely on to predict future events.

However, that does not mean that time series would not work at all. We know that our data given its exponential distribution has to follow a certain pattern, which is to say, that the number of calls decrease with the passage of time. This is to say that if we were to sort our random data for any given day we would obtain a pattern that follows the exponential distribution of data. Following is an example, of when we took all of our Mondays in our dataset and sorted them:



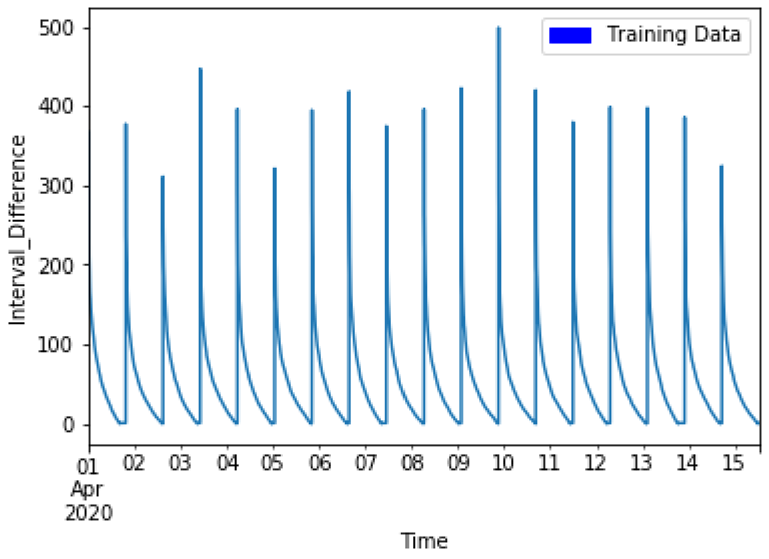
This means that while there is no sequential pattern in the data, we can figure out a pattern if we arrange our data to be sorted in order to obtain the exponential distribution. Since we know that each day has a different exponential distribution, then we can focus on applying time series methods of each day separately and then focus on understanding the exponential pattern by learning the smoothing parameters so that we can predict a distribution, based on how the past data behaves, for the next day.

But before we go and see the actual model in itself, including its performance, we need to go over through some other nuances. Time series methods, when seasonality is concerned, requires discrete and fixed times. In other words, if we are analyzing 12 hours of a day, then all of Mondays must have ONLY 12 datapoints that talk about the 12 hours of a day. This is due to the computation of seasonality. If I am trying to predict a value at 12 AM of the day, then I need to observe the values of the past days at 12 AM. If I tried to predict the value of 1 PM for a day, then I won't be able to because 1 PM might not exist in the previous days as part of the dataset and therefore, I will not be able to compute the seasonality.

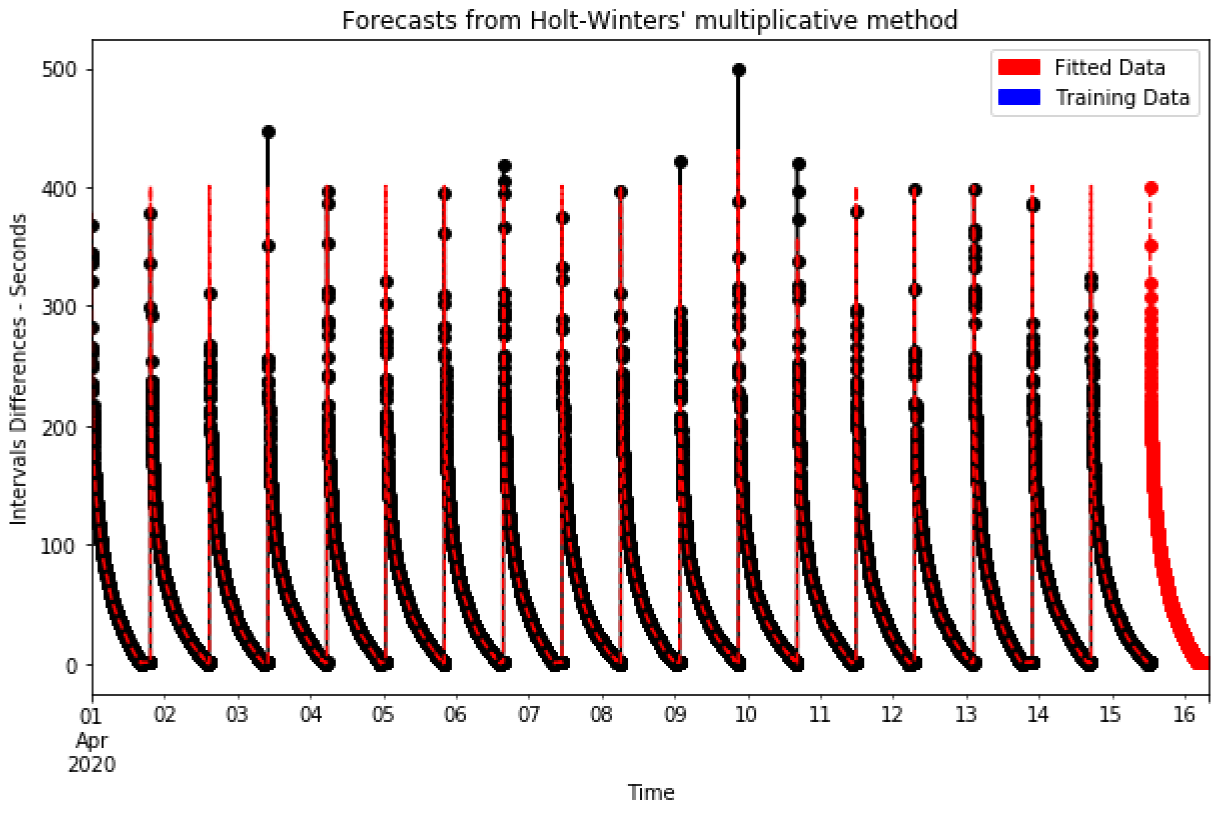
Similarly, the number of calls which means the number of total datapoints of duration between successive calls are not constant. In a certain week, a day might have received X calls but in another week, a day might have received X+Y calls. Therefore, in order to make the total number of datapoints the same so that we can compute the seasonality, we took the maximum number of calls received in all instance of that particular day and modified all of our days, by artificially adding calls, to have the reach up to the maximum number of calls. Hence, all Mondays now have X datapoints in form of total calls or total durations between successive calls. This is potentially corrupting out dataset because we are changing the parameter that defines the distribution of the data in itself. However, to evaluate effectively, we'll need to compare the predicted results with the actual ones.

Now, we take a particular day and apply time series on it. We consider Monday. There are a total of 17 Mondays in our dataset. We will be using 16 of them as our test dataset. Each Monday receives a different number of calls but the deviation between these number is small. The maximum number of datapoints on any Monday amounts to 3857 calls/total durations. We top of all the Monday to contain these maximum datapoints by adding additional data into them. Additional data in our case is just the addition of 1s until the difference is covered. This effectively means that we artificially added more calls into each day but all of which have a duration of 1 second between successive calls.

The graph below shows a set of Mondays that we'll be applying time series on. Each new rise indicates a new instance of Monday.



Following is the graph when we applied time series to instances of Monday:



The graph above the application of Holt Winters on our day and a production of a distribution for the next day, as a result of the analysis of the past instance. The red dots are the predicted datapoints. The black datapoints are the datapoints upon which time series ran and learnt the smoothing parameters. The red lines are the distributions produced by the smoothing parameters over the test dataset.

Just as we did with Monday, we applied time series methods on each of our days. Then for our predicted distribution of the next Monday, ideally the mean of our predicted durations should match exactly that of the training distributions. Following are the means that we obtained for each of the day and a comparison between the actual and predicted values:

Stat	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Actual Means	15.01	17.04	19.90	21.09	22.14	40.22	50.04
Predicted Mean	14.58	18.77	20.47	21.58	46.09	16.93	48.6

From the results, we can see that, with the exception of Saturday, the means from the actual and predicted data are quite close. There is not a significant enough difference in them to alter the distributions on a disastrous level.

Bootstrapping

Simulation methods in which the distribution to be sampled from is determined from data are called bootstrap methods. Bootstrap is a powerful, computer-based method for statistical inference without relying on too many assumption. The basic idea of bootstrap is make inference about a estimate(such as sample mean) for a population parameter - theta - (such as population mean) on sample data. It is a resampling method by independently sampling with replacement from an existing sample data with same sample size n, and performing inference among these resampled data.

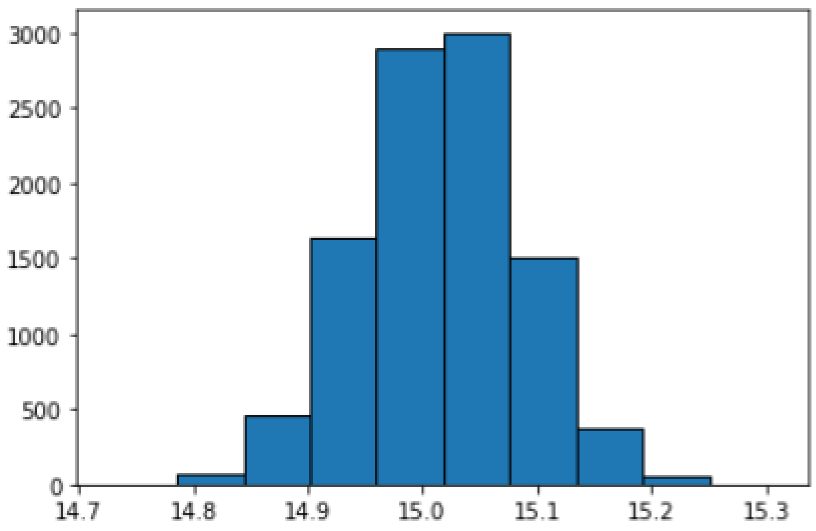
Our bootstrapping function takes in the dataFrame as input along with day of week (otherwise it runs on the whole data), number of times it needs to resample. In our bootstrapping approach we make a list which has a length of 90 percent of data that is being fed in. Then we sample with replacement and calculate the mean. We repeat this process 10000 times. As per the central limit theorem the sample means would form a normal distribution. Our function outputs the list of sample means, the mean of the data being fed in, a list of seed values that were put into the resample function, the lenght of samples and the data that was inputted in the function.

Furthermore, we calculate the mean of sample means which we call the bootstrap mean. Then we calculate the difference between the mean of arrival differences of the data and the bootstrap mean to see how much the values differ.

For each day of the week we calculate the mean of sample means which we call bootstrap mean. Then we calculate the difference between the simple mean (of arrival differences) and the bootstrap mean:

Labels	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Sample Mean	15.015320520847958	17.04326839939892	19.90720098119531	21.089445326852633	22.148069853519115	39.80605238829787	50.03980967819629
Bootstrap Mean	15.015856365878586	17.043415994368996	19.90689372352281	21.09044311952291	22.14717803630309	39.80510127204907	50.0411711558025
Difference	-0.0005358450306278684	-0.00014759497007688083	0.0003072576724996168	-0.0009977926702759987	0.0008918172160257143	0.0009511162487996216	-0.00136147760620986

Then we plot the sample means of a day, such as Monday, to determine that the central limit theorem is working.

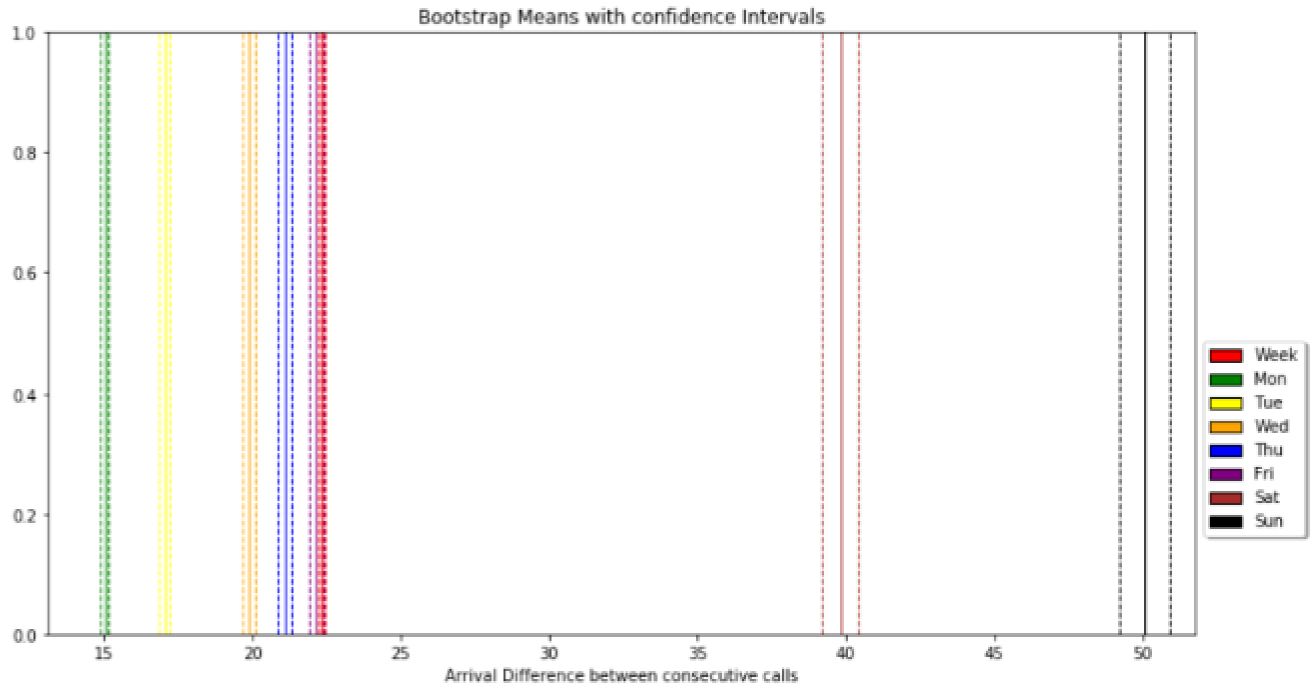


Moving on, we calculate the sample means of the whole dataset i.e. containing all days of week. We then calculate the bootstrap mean and then finally we calculate the difference between the simple mean and bootstrap mean.

The purpose of calculating the mean of the whole dataset was to see if there is a difference between the means of individual days and the mean of the whole dataset.

Furthermore, by calculating the mean of the whole dataset we could use it to model the days which have same means as the mean of this whole dataset. So instead of finding seven parameters for the exponential random variables we can use lesser number of parameters when there is an overlap. In accordance with our calculations, the bootstrap mean of the whole dataset turns out to be: **22.347058536955526** which is approximately the same as the mean of the dataset: **22.347058536955526**

We calculate the confidence intervals for our bootstrapping. We use the approach mentioned in the chapter 5 of the book. Statistics for Engineers and Scientists by William Nivadi. We sort the data and then calculate the mean of values from 250-260 for the lower confidence interval and calculate the mean of 9750-9760 value for the upper confidence interval. Plotting the bootstrap Means of each day and the bootstrap Mean of the whole dataset along with confidence intervals. If there is an overlap between confidence intervals we can say that the difference between those two particular means is not statistically significant. From our plots we can see the mean of the week overlaps with Friday hence the difference is not statistically significant. However, since there is only one overlap so we still need seven parameters or seven expected values to make an exponential distribution for each day of the week. Hence, we do not perform any hypothesis testing.



Following are the seven bootstrap means or our expected values.

Stat	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Week
B-Means	15.01	17.04	19.90	21.09	22.14	391.80	50.04	22.34

We can calculate the rate parameter by the following formula lambda = 1/E[x]:

Stat	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Week
Lambdas	2.71	3.99	1.50	1.19	2.84	3.52	3.01	2.68

Survival Analysis

Survival Analysis is used to estimate the lifespan of a particular population under study. It is also called 'Time to Event' Analysis as the goal is to estimate the time for an individual or a group of individuals to experience an event of interest. This time estimate is the duration between birth and death events. Survival Analysis was originally developed and used by Medical Researchers and Data Analysts to measure the lifetimes of a certain population. But, over the years, it has been used in various other applications such as predicting churning customers/employees, estimation of the lifetime of a Machine, etc. The birth event can be thought of as the time of a customer starts their membership with a company, and the death event can be considered as the customer leaving the company. Next, we also need to talk about one of the fundamental objects in survival analysis, the **Survival Function**.

Let **T** be a (possibly infinite, but always non-negative) random lifetime taken from the population under study. For example, the amount of time a couple is married. Or the time it takes a user to enter a webpage (an infinite time if they never do). The survival function - $\mathbf{S(t)}$ - of a population is defined as:

$$S(t) = P(T > t) \tag{5.1}$$

Simply, the survival function defines the probability the death event has not occurred yet at time t, or equivalently, the probability of surviving past time **t**.

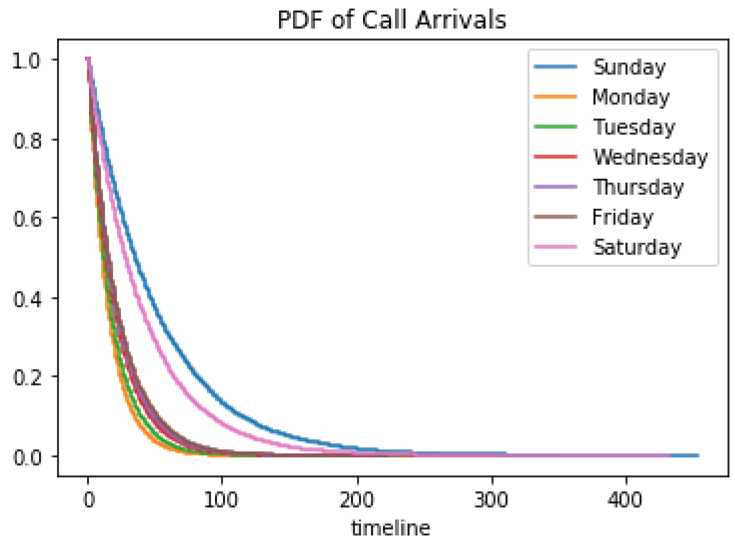
Modelling the Call Problem

Now, we need to model our problem for Survival Analysis. First, we begin by deciding how to represent our call in a life and death scenario. We decide that the death of a call occurs when a call is received. This means that all the calls are, initially, placed at a reference line, also known as their point of birth, and these calls die after an interval, in other words, after a call is received. Therefore, Survival Analysis estimates the survival or the death (aka call arrival) of the call.

Secondly, we decide upon the survival function that we will be using to estimate the probability of a call surviving with the passage of time. For that, we utilize Kaplan Meir Estimator to form a Probability Density Functions, which is a conditional distribution. The Kaplan-Meier estimate is the simplest way of computing the survival over time in spite of all these difficulties associated with subjects or situations. The survival curve can be created assuming various situations. It involves computing of probabilities of occurrence of event at a certain point of time and multiplying these successive probabilities by any earlier computed probabilities to get the final estimate. Its advantage of KMS lies in its essence of being a non-parametric estimator. Parametric Estimators assume that the data follows a certain distribution and utilizies the parameters of the distribution to obtain results. KMS, instead, form a PDF by directly computing conditional probabilities from the data.

Through our exploratory data analysis, we know that there are 7 lambdas for each day, which is to say that there is an exponential distribution describing the intervals between our calls for each day. We also know that any Exponential distribution can be described using its parameter, lambda.

Survival Analysis allows us to learn the probability distributions describing the **probability of a call arriving during or after a specific interval for each day**, which in-turn can be **used to obtain other statistics such as lambdas** that describes the average interval between two successive call, or in other words, our exponential distribution. Following are the PDFs obtained for each day:



Additionally, the mean survival time would be the area under the curve. Therefore:

Day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Mean	15.51	17.54	20.40	21.58	22.63	40.22	50.35

Module 2

Amongst well-defined categories, what type of call is it?

Generalized Linear Models

In these models, the response variable y_i is assumed to follow an exponential family distribution with mean μ_i which is assumed to be some (often nonlinear) function of $x_i^T\beta$. The covariates affect the distribution of y_i only through the linear combination

$x_i^T\beta$.The generalized linear models (GLMs) are a broad class of models that include linear regression, ANOVA, Poisson regression, log-linear models etc.The table below provides a good summary of GLMs: **There are three components to any GLM::**

- Random Component – refers to the probability distribution of the response variable (Y); e.g. normal distribution for Y in the linear regression, or binomial distribution for Y in the binary logistic regression. Also called a noise model or error model. How is random error added to the prediction that comes out of the link function?
- Systematic Component - specifies the explanatory variables $(X_1, X_2, ...X_k)$ in the model, more specifically their linear combination in creating the so called linear predictor; e.g., $\beta_0 + \beta_1 x_1 + \beta_2 x_2$ as we have seen in a linear regression, or as we will see in a logistic regression in this lesson.
- Link Function, $\eta org(\mu)$ - specifies the link between random and systematic components. It says how the expected value of the response relates to the linear predictor of explanatory variables.

Assumptions:

- The data $Y_1, Y_2, ..., Y_n$ are independently distributed, i.e., cases are independent.
- The dependent variable Yi does NOT need to be normally distributed, but it typically assumes a distribution from an exponential family (e.g. binomial, Poisson, multinomial, normal,...)
- GLM does NOT assume a linear relationship between the dependent variable and the independent variables, but it does assume linear relationship between the transformed response in terms of the link function and the explanatory variables. Independent (explanatory) variables can be even the power terms or some other nonlinear transformations of the original independent variables.
- The homogeneity of variance does NOT need to be satisfied. In fact, it is not even possible in many cases given the model structure, and overdispersion (when the observed variance is larger than what the model assumes) maybe present.
- Errors need to be independent but NOT normally distributed.

- It uses maximum likelihood estimation (MLE) rather than ordinary least squares (OLS) to estimate the parameters, and thus relies on large-sample approximations.

- Goodness-of-fit measures rely on sufficiently large samples, where a heuristic rule is that not more than 20 percent of the expected cells counts are less than 5.

Logistic Regression

Since our response variable has two categories we have used logistic regression. Binary Logistic Regression models how binary response variable Y depends on a set of k explanatory variables, $X = (X_1, X_2, ...X_k)$. $logit(\pi) = log(\frac{\pi}{1-\pi}) = \beta_0 + \beta * x_i + ... + \beta_0 + \beta * x_k$ which models the log odds of probability of "success" as a function of explanatory variables.

Random component: The distribution of Y is assumed to be Binomial(n, η), where η is a probability of "success".

Systematic component: X's are explanatory variables (can be continuous, discrete, or both) and are linear in the parameters, e.g., $\beta_0 + \beta * x_i .... + \beta_0 + \beta * x_k$. Again, transformation of the X's themselves are allowed like in linear regression; this holds for any GLM.

Link function:

- Logit link: $logit(\pi) = log(\frac{\pi}{1-\pi})$

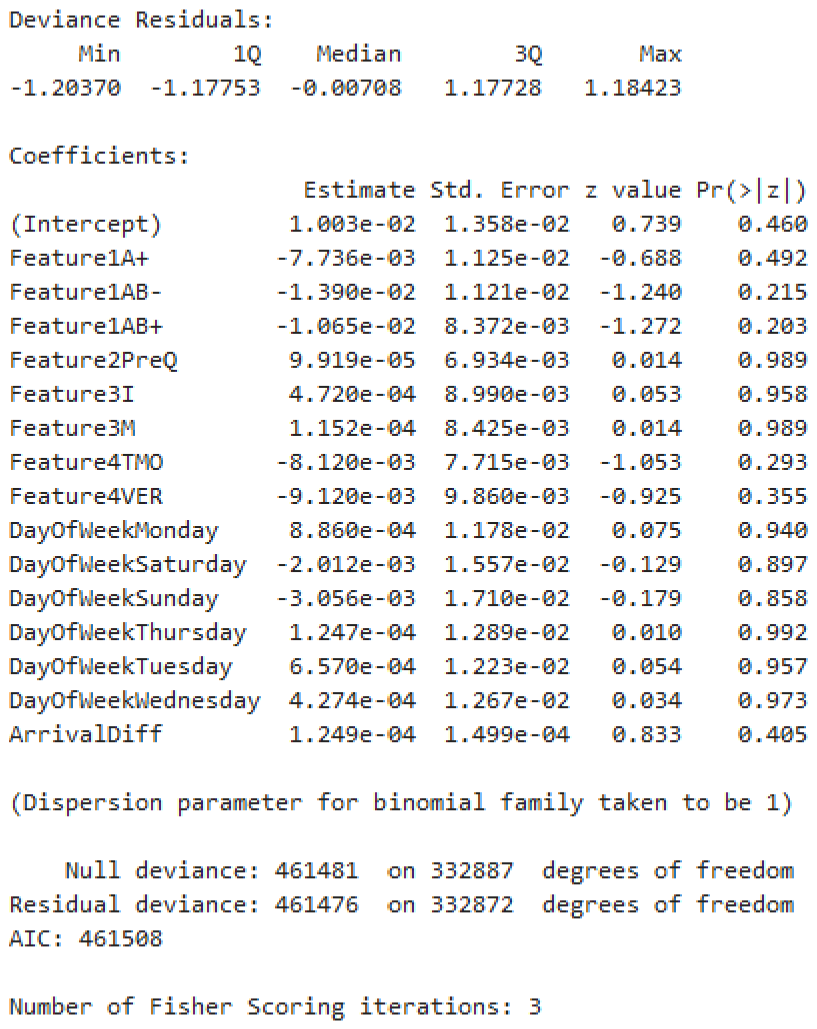
- More generally, the logit link models the log odds of the mean, and the mean here is η. Binary logistic regression models are also known as logit models when the predictors are all categorical.

Modelling our Problem

We first fitted a glm with all of the explanatory variables included. Then we used the dredge function to make models of all possible combinations of 2 from those features. Since we have 6 features then all possible combinations will be $2^6 = 64$. In order to simplify things a little, we also made another model which did not contain ArrivalDifferences since our problem specifically asks for predicting the call category of the next call. Then we used the dredge functions to make models from all possible combinations of 2 from those features. In this case since we removed Arrival Differences we have 5 features so all possible combinations will be $2^5 = 32$. Furthermore, it wasn't correlated with the response variable as per the Cramer's V test. In order to simplify things we also made model without ArrivalDifferences.

Training our Model

Next we trained our Logistic Regression Model for the given data. However, before we discuss the actual results, we need to improve our model further. How do we do that? By further analyzing our trained model and identifying the features that are important. Therefore, we first obtained the summary statistics to help us in understanding the model better which is the following:



```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.20370  -1.17753  -0.00708   1.17728   1.18423

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         1.003e-02  1.358e-02   0.739    0.460
Feature1A+         -7.736e-03  1.125e-02  -0.688    0.492
Feature1AB-        -1.390e-02  1.121e-02  -1.240    0.215
Feature1AB+        -1.065e-02  8.372e-03  -1.272    0.203
Feature2PreQ        9.919e-05  6.934e-03   0.014    0.989
Feature3I           4.720e-04  8.990e-03   0.053    0.958
Feature3M           1.152e-04  8.425e-03   0.014    0.989
Feature4TMO        -8.120e-03  7.715e-03  -1.053    0.293
Feature4VER        -9.120e-03  9.860e-03  -0.925    0.355
DayOfWeekMonday     8.860e-04  1.178e-02   0.075    0.940
DayOfWeekSaturday  -2.012e-03  1.557e-02  -0.129    0.897
DayOfWeekSunday    -3.056e-03  1.710e-02  -0.179    0.858
DayOfWeekThursday   1.247e-04  1.289e-02   0.010    0.992
DayOfWeekTuesday    6.570e-04  1.223e-02   0.054    0.957
DayOfWeekWednesday  4.274e-04  1.267e-02   0.034    0.973
ArrivalDiff         1.249e-04  1.499e-04   0.833    0.405

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 461481  on 332887  degrees of freedom
Residual deviance: 461476  on 332872  degrees of freedom
AIC: 461508

Number of Fisher Scoring iterations: 3
```

The summary statistics helps us in understanding the model better by providing us with the following information:

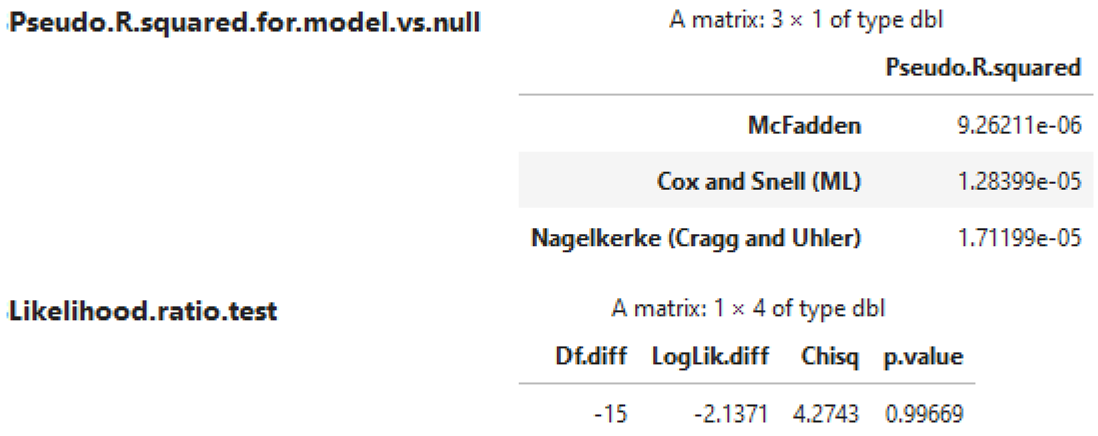
- Distribution of the deviance residuals

- Intercept and slope estimates along with the standard error, z-value, and p-value
- AIC value
- Residual Deviance (for fitted model) and Null Deviance (for intercept model)

The coefficient table shows that none of the explanatory variables have significant influence ($p - values > 0.05$) on CallCategory.The difference between Null deviance and Residual deviance tells us that the model is a good fit. Greater the difference better the model. Null deviance is the value when you only have intercept in your equation with no variables and Residual deviance is the value when you are taking all the variables into account. It makes sense to consider the model good if that difference is big enough. The difference in our case doesn't seem to be very large.

Additionally, we also applied two other tests:

- Pseudo R-Squared and Likelihood Ratio Test



Pseudo.R.squared.for.model.vs.null

A matrix: 3 × 1 of type dbl

	Pseudo.R.squared
McFadden	9.26211e-06
Cox and Snell (ML)	1.28399e-05
Nagelkerke (Cragg and Uhler)	1.71199e-05

Likelihood.ratio.test

A matrix: 1 × 4 of type dbl

Df.diff	LogLik.diff	Chisq	p.value
-15	-2.1371	4.2743	0.99669

  –
  – The McFadden Pseudo R-squared value is the commonly reported metric for binary logistic regression model fit. The table result showed that the McFadden Pseudo R-squared value is $9.26211e - 06$, which indicates the model has no predictive ability, although the likelihood value for the current model will be (it is always) larger than the likelihood of the null model, it will not be much greater. Therefore it will be close to zero, as we would hope.

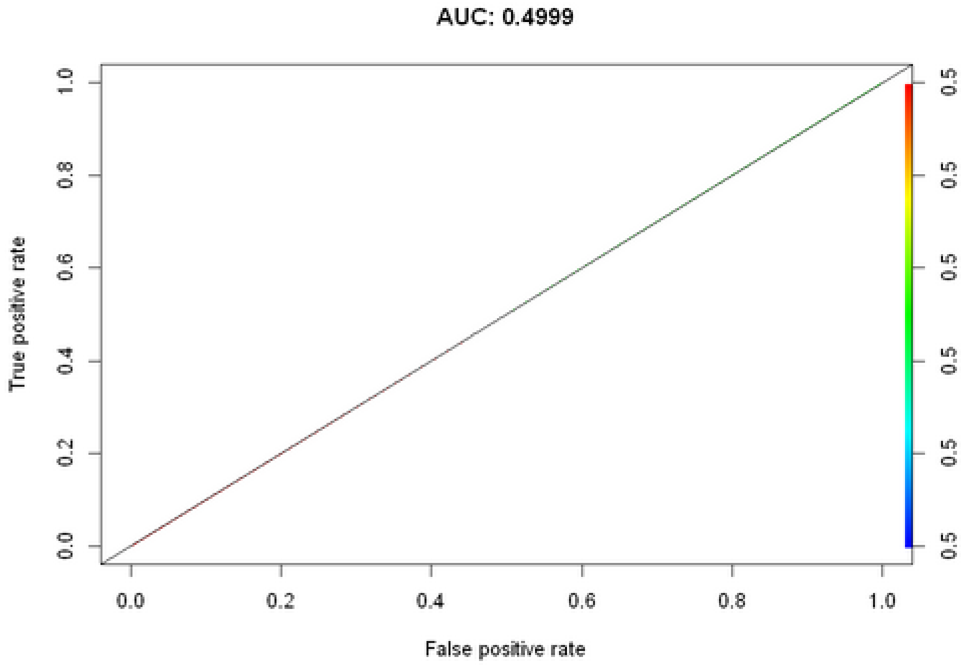
    Additionally, the table provides a Likelihood ratio test. Likelihood Ratio test (often termed as LR test) is a goodness of fit test used to compare between two models; the null model and the final model. The null hypothesis is that the smaller model is the “best” model; It is rejected when the test statistic is large. In other words, if the null hypothesis is rejected, then the larger model is a significant improvement over the smaller one. The test revealed that the Log-Likelihood difference between intercept only model (null model) and model fitted with all independent variables is $2.1371 and p - value 0.99$ which shows the improvement is also not significant as $(p - value > 0.05)$ and we do not reject the null hypothesis.

- AIC - Akaike Information Criteria
  - It is analogous to adjusted R^2 and is the measure of fit which penalizes model for the number of independent variables. We always prefer a model with minimum AIC value. Since from the table it shows that the null model is the best model we pick the model within two AIC values

Using the above tests, we learnt that the best performing model with the minimal number of features is one where the independent variables is only Feature 2. As can be seen from the AIC test, it holds the minimum value.

Prediction on Test Dataset

ROC stands for Receiver Operating Characteristic. It explains the model’s performance by evaluating True Positive Rate vs False Positive Rate. The area under the ROC Curve is an index of accuracy. Higher the area under the curve, better the prediction power of the model. AUC of a perfect predictive model equals 1.



From the ROC plot we can see that the value of AUC is 0.499.

Learning a Bayesian Network Structure from data

This section will be about obtaining a Bayesian network, given a set of sample data. Learning a Bayesian network can be split into two problems:

- **Structure learning:** Given a set of data samples, estimate a DAG that captures the dependencies between the variables . Structure Learning is mainly classified intoo:
  - score-based structure learning
  - constraint-based structure learning
- **Parameter learning**: Given a set of data samples and a DAG (Directed Acyclic Graph) that captures the dependencies between the variables, estimate the (conditional) probability distributions of the individual variables.

For our problem, we will be working with **Score Based Learning**. Let's discuss that in itself a bit before talking about it in the context of our case.

Score-based Structure Learning

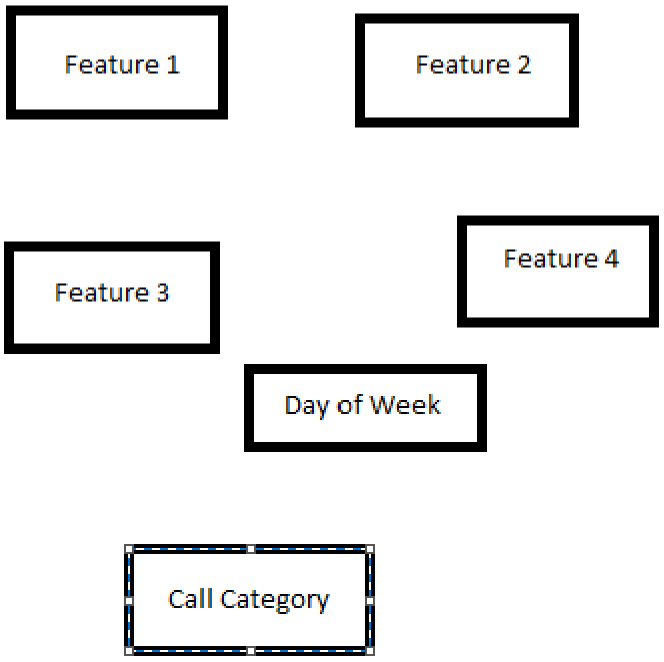
This approach construes model selection as an optimization task. It has two building blocks:

- A scoring function that maps models to a numerical score, based on how well they fit to a given data set.
- A search strategy to traverse the search space of possible models M and select a model with optimal score.

Scoring functions: Commonly used scores to measure the fit between model and data are Bayesian Dirichlet scores such as BDeu or K2 and the Bayesian Information Criterion (BIC, also called MDL).

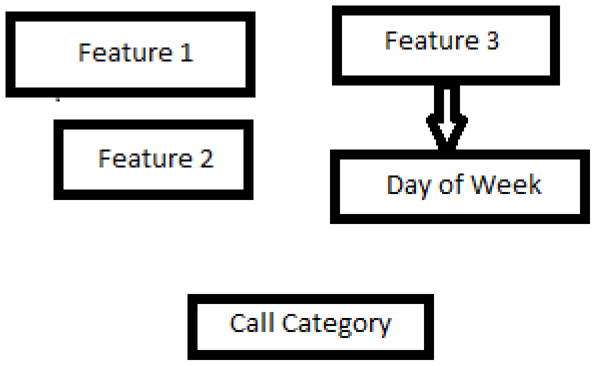
Search strategies: The search space of DAGs is super-exponential in the number of variables and the above scoring functions allow for local maxima. The first property makes exhaustive search intractable for all but very small networks, the second prohibits efficient local optimization algorithms to always find the optimal structure. Thus, identifiying the ideal structure is often not tractable. Despite these bad news, heuristic search strategies often yields good results. Take for instance, **HillClimbSearch**. It implements a greedy local search that starts from the DAG 'start' , our refereince node is a graph with all nodes but no edges or conditional dependencies, and proceeds by iteratively performing single-edge manipulations that maximally increase the score. The search terminates once a local maximum is found. By Single-Edge manipulations, we refer to legal operations that can be performed on the network. Operations such as addition, deletion or modification of an edge between two nodes.

For our problem, we used BicScore as our Scoring functions. The BIC/MDL score ("Bayesian Information Criterion", also "Minimal Descriptive Length") is a log-likelihood score with an additional penalty for network complexity, to avoid overfitting.As for the search space strategy, we employed HillClimbSearch.



To start off with, above is the reference graph that is created from our features and call category. First, the Scoring function calculates the log probability of this Directed Acyclic Graph to determine how good the graph is classifying Call Categories. Then, as the next step, the search space function finds a better directed graph that represents our problem by performing legal operations on the graph such as the addition or deletion of an edge. As a result of either of the two actions, a new graph is formed whose fitness is again determined through the Scoring functions. If the current graph is better than the old one, our algorithms chooses this as the most optimal graph up til now and repeats the process until it cannot find another directed graph that has a better score.

As a result, the optima directed graph determined by our algorithm comes out to be:



According to the above graph, our output - Call Category - is independent of any other features. In other words, it is either random or there are no patterns within the current features that affect our output label.

For the purpose of ensuring the reliability of the process and results, we also carried out a conditional independence test between the output label, call category, and our features. Dependencies in the data can be identified using chi2 conditional independence tests. The chi-squared statistic is a single number that tells you how much difference exists between your observed counts and the counts you would expect if there were no relationship at all in the population.

The Chi-squared test confirms the results acheived through Score-Based Structure Learning. It states that there is no conditional relationship between the output label oand and any of the features. It also states that there are no conditional relationships between the feature themselves, with the exception of Feature 1 and Day of Week.

Parameter Learning

Having learnt the DAG that captures dependencies between the variables, now using the data samples we need to estimate the (conditional) probability distributions of the individual variables. We did this through **Maximum Likelihood Estimation** and **Bayesian Parameter Estimation**.

Maximum Likelihood Estimation

A natural estimate is to simply use the *relative frequencies*, with which the variable states have occured. Hence, let's assume a scenario where I have two types of calls: HR and Business based calls. I receive a total of 10 calls where 7 of these calls are HR based calls, then according to the concept of relative frequences, MLE will overfit the data, where it estimate that in any scenario 70 percent of the calls are HR based calls. In the long run, this might actually turn out to not be the case. This 70 percent could be because we didn't have enough data. For our problem, given our data, MLE estimated that:

```
+-----------------+----------+
| Feature2(PostQ) | 0.507065 |
+-----------------+----------+
| Feature2(PreQ)  | 0.492935 |
+-----------------+----------+
+-------------+----------+
| Feature3(F) | 0.291573 |
+-------------+----------+
| Feature3(I) | 0.303369 |
+-------------+----------+
| Feature3(M) | 0.405058 |
+-------------+----------+
+---------------+----------+
| Feature4(SPR) | 0.370019 |
+---------------+----------+
| Feature4(TMO) | 0.44436  |
+---------------+----------+
| Feature4(VER) | 0.185621 |
+---------------+----------+
```

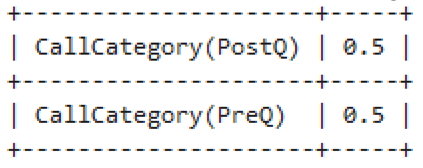
```
+---------------------+-----+
| CallCategory(PostQ) | 0.5 |
+---------------------+-----+
| CallCategory(PreQ)  | 0.5 |
+---------------------+-----+
+---------------+----------+
| Feature1(A+)  | 0.142063 |
+---------------+----------+
| Feature1(A-)  | 0.285841 |
+---------------+----------+
| Feature1(AB+) | 0.428357 |
+---------------+----------+
| Feature1(AB-) | 0.143739 |
+---------------+----------+
```

From the estimations, we can see that at any point in time when a call is received, there is a 50 percent probability that the call category can be either PostQ or PreQ.

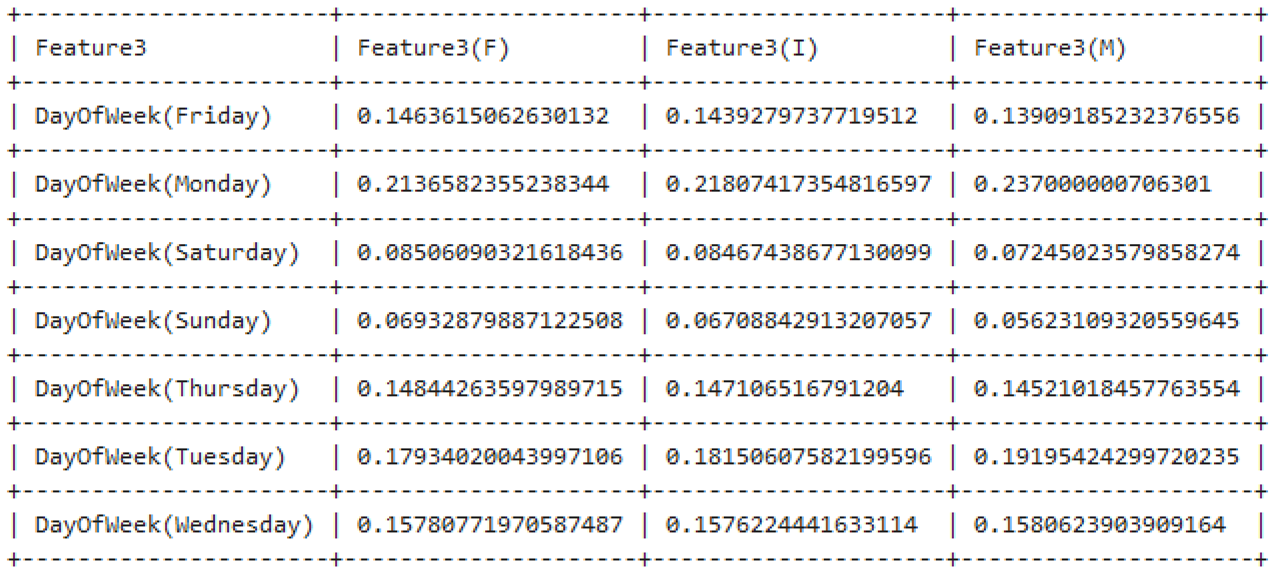
Bayesian Parameter Estimation

The Bayesian Parameter Estimator starts with already existing prior CPDs (Conditional Probability Distributions), that express our beliefs about the variables *before* the data was observed. Those "priors" are then updated, using the state counts from the observed data. One can think of the priors as consisting in pseudo state counts, that are added to the actual counts before normalization. A very simple prior is the so-called K2 prior, which simply adds 1 to the count of every single state.Another is BDeu (Bayesian Dirichlet equivalent uniform prior). For BDeu we need to specify an equivalent sample size N and then the pseudo-counts are the equivalent of having observed 'N' uniform samples of each variable.

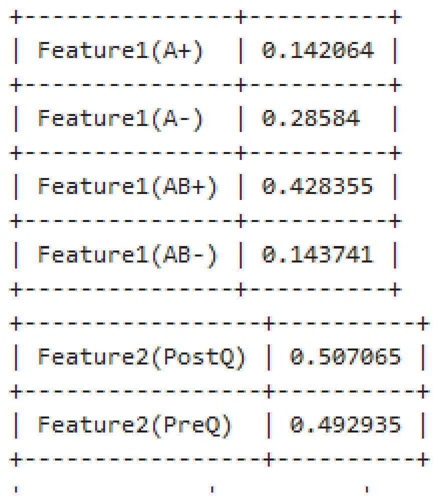
We used BDeu as our prior for Bayesian Parameter Estimation and obtained the below results. Note, that the estimator learns of a conditional probability distribution between Feature 3 and Day of Week but as for the call category probabilities, due to the absence of any conditional dependencies, the estimations for it are the same as that estimated by the MLE. Again, this is because there are no conditional dependences, and hence, in any case, regardless of what estimator we use, the probabilities of the Call Category will always be a result of the relative frequencies.



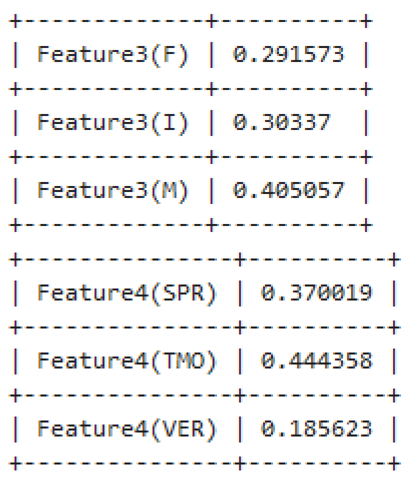
```
+---------------------+-----+
| CallCategory(PostQ) | 0.5 |
+---------------------+-----+
| CallCategory(PreQ)  | 0.5 |
+---------------------+-----+
```



```
+----------------------+---------------------+---------------------+---------------------+
| Feature3             | Feature3(F)         | Feature3(I)         | Feature3(M)         |
+----------------------+---------------------+---------------------+---------------------+
| DayOfWeek(Friday)    | 0.1463615062630132  | 0.1439279737719512  | 0.13909185232376556 |
+----------------------+---------------------+---------------------+---------------------+
| DayOfWeek(Monday)    | 0.2136582355238344  | 0.21807417354816597 | 0.237000000706301   |
+----------------------+---------------------+---------------------+---------------------+
| DayOfWeek(Saturday)  | 0.08506090321618436 | 0.08467438677130099 | 0.07245023579858274 |
+----------------------+---------------------+---------------------+---------------------+
| DayOfWeek(Sunday)    | 0.06932879887122508 | 0.06708842913207057 | 0.05623109320559645 |
+----------------------+---------------------+---------------------+---------------------+
| DayOfWeek(Thursday)  | 0.14844263597989715 | 0.147106516791204   | 0.14521018457763554 |
+----------------------+---------------------+---------------------+---------------------+
| DayOfWeek(Tuesday)   | 0.17934020043997106 | 0.18150607582199596 | 0.19195424299720235 |
+----------------------+---------------------+---------------------+---------------------+
| DayOfWeek(Wednesday) | 0.15780771970587487 | 0.1576224441633114  | 0.1580623903909164  |
+----------------------+---------------------+---------------------+---------------------+
```



```
+---------------+----------+
| Feature1(A+)  | 0.142064 |
+---------------+----------+
| Feature1(A-)  | 0.28584  |
+---------------+----------+
| Feature1(AB+) | 0.428355 |
+---------------+----------+
| Feature1(AB-) | 0.143741 |
+---------------+----------+
+-----------------+----------+
| Feature2(PostQ) | 0.507065 |
+-----------------+----------+
| Feature2(PreQ)  | 0.492935 |
+-----------------+----------+
```



```
+-------------+----------+
| Feature3(F) | 0.291573 |
+-------------+----------+
| Feature3(I) | 0.30337  |
+-------------+----------+
| Feature3(M) | 0.405057 |
+-------------+----------+
+---------------+----------+
| Feature4(SPR) | 0.370019 |
+---------------+----------+
| Feature4(TMO) | 0.444358 |
+---------------+----------+
| Feature4(VER) | 0.185623 |
+---------------+----------+
```

Module 3

Within a well-defined interval, how many total calls can we expect from a particular category of callers?

Interactions in Regression

For the third module, we'll be focusing on the concept of Interactions in Regression Analysis to classify our output variable. In regression, an interaction effect exists when the effect of an independent variable on a dependent variable changes, depending on the value(s) of one or more other independent variables. In any regression equation, an interaction effect is represented as the product of two or more independent variables. For example, here is a typical regression equation without an interaction: $y = b_0 + b_1 * X_1 + b_2 * X_2$ where y is the predicted value of a dependent variable, X_1 and X_2 are independent variables, and $b_0, b_1, and b_2$ are regression coefficients. Here is the same regression equation with an interaction:

$$y = b_0 + b_1 * X_1 + b_2 * X_2 + b_3 * X_1 * X_2$$

Here, b_3 is a regression coefficient, and $X_1 * X_2$ is the interaction. The interaction between X_1 and X_2 is called a two-way interaction, because it is the interaction between two independent variables. Higher-order interactions are possible, as illustrated by the three-way interaction in the following equation:

$$y = b_0+b_1*X_1+b_2*X_2+b_3*X_3+b_4*X_1*X_2+b_5*X_1*X_3+b_6*X_2*X_3+b_7*X_1*X_2*X_3$$

Analysts usually steer clear of higher-order interactions, like $X_1X_2X_3$, since they can be hard to interpret.

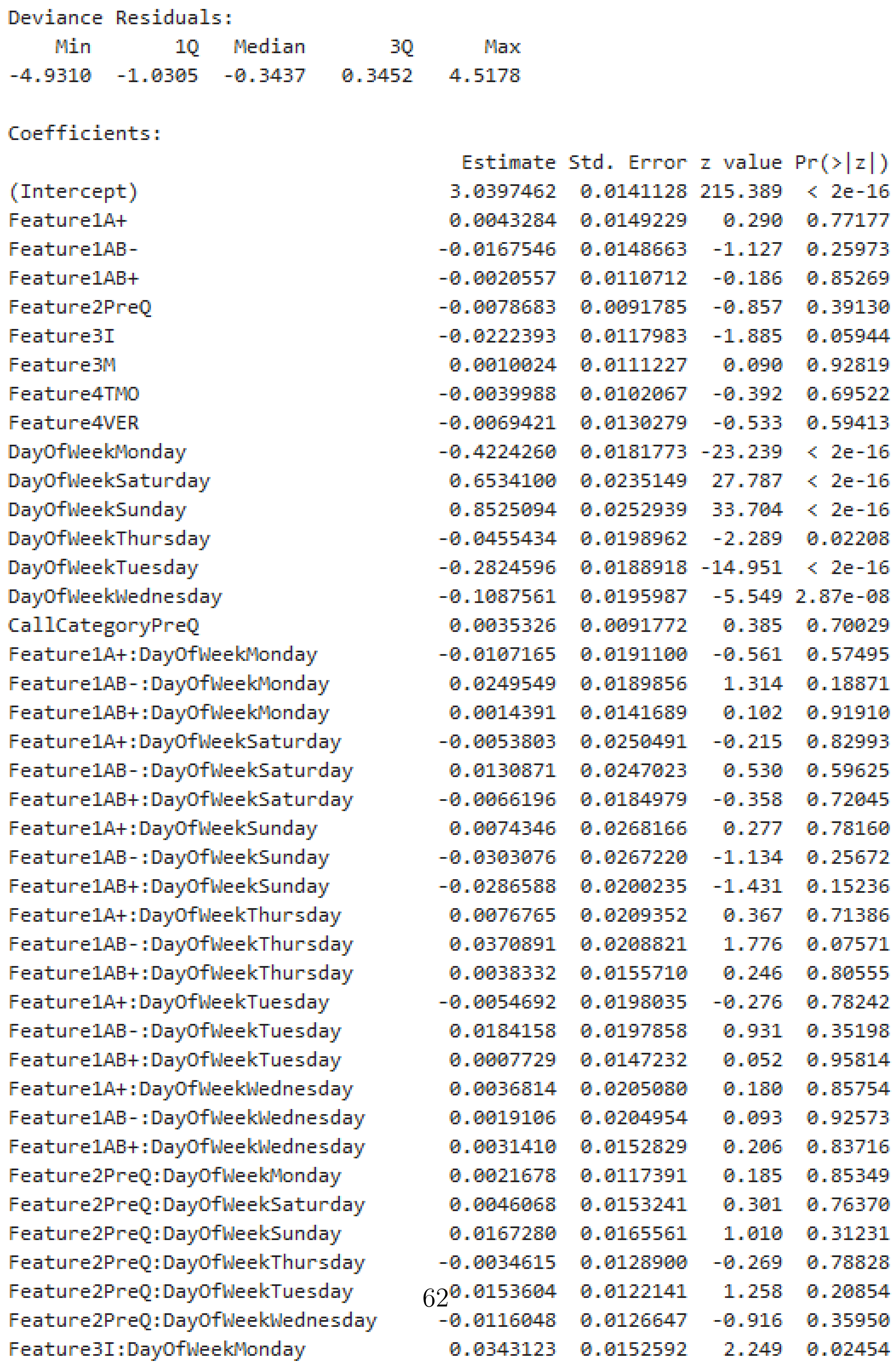
For this particular module we are looking into interactions as well to see if we can find complex relationships such that based on the features or their interactions we are able to predict the number of calls of a particular category in an interval.

Modelling our Problem

This time our output variable is the Interval difference/duration between successive call categories. Our intention is to predict the duration given a combination of features. Since, we are dealing with continous values and an exponential distribution, we'll be using ***Gamma GLMs***. The gamma glm has choice of three link functions the ones that are normally used are log and inverse we will create two models using each link function and then pick the one which gives a better fit. We first fit a glm with all of the explanatory variables included. Then we used the dredge function to make models of all possible combinations of those features. Since we have 6 features then all possible combinations will be $2^6 = 64$ because there are two possibilities for each feature either it could be in the model or either it cannot. We have different choices of link function so we try log link and inverse link function and choose the model which fits better. Moreover, the data has calls with zero ArrivalDiff the gamma glm doesn't accept zero entries hence we replace those zeros with very small value 0.0001.

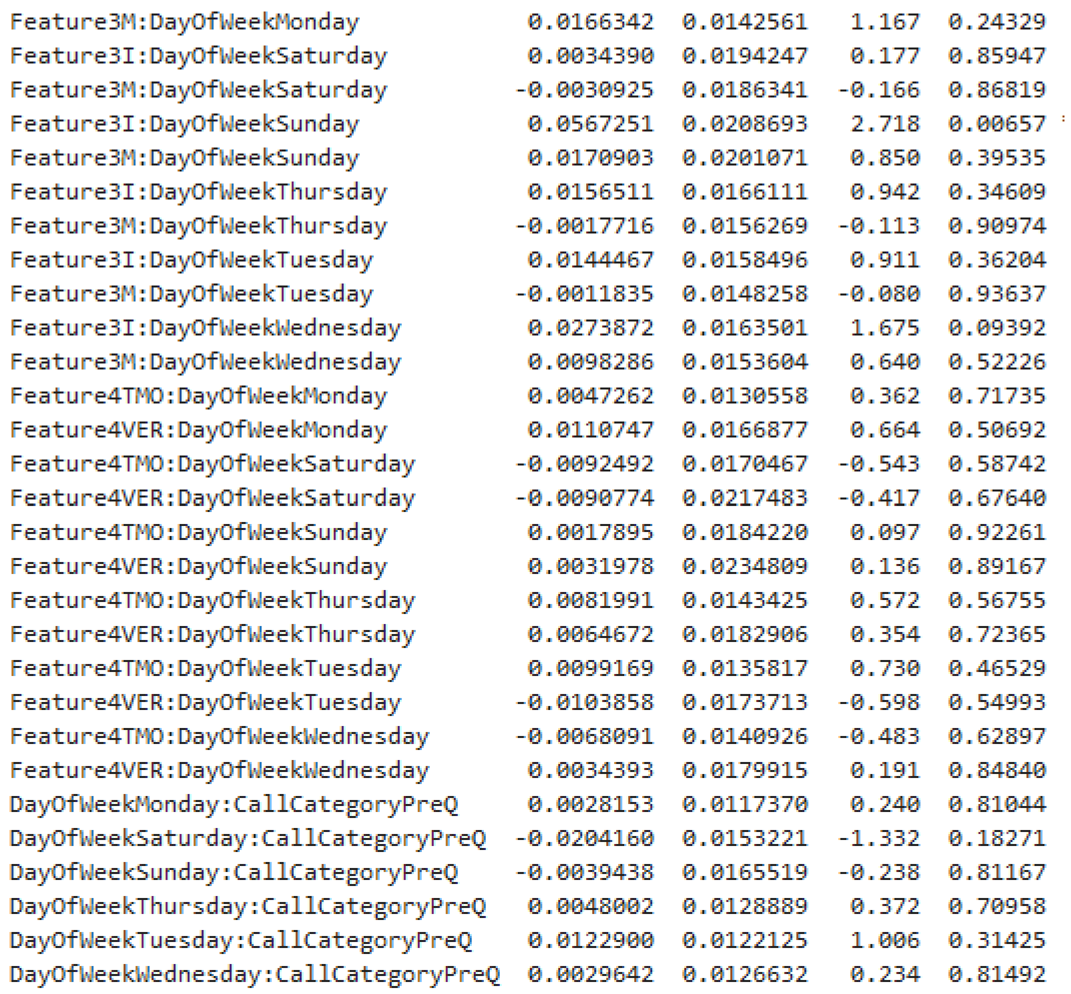
Training our Model

Next we trained our GLM for the given data. However, before we discuss the actual results, we need to improve our model further. How do we do that? By further analyzing our trained model and identifying the features that are important. Therefore, we first obtained the summary statistics to help us in understanding the model better which is the following:



```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.9310  -1.0305  -0.3437   0.3452   4.5178

Coefficients:
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                          3.0397462  0.0141128 215.389  < 2e-16
Feature1A+                           0.0043284  0.0149229   0.290  0.77177
Feature1AB-                         -0.0167546  0.0148663  -1.127  0.25973
Feature1AB+                         -0.0020557  0.0110712  -0.186  0.85269
Feature2PreQ                        -0.0078683  0.0091785  -0.857  0.39130
Feature3I                           -0.0222393  0.0117983  -1.885  0.05944
Feature3M                            0.0010024  0.0111227   0.090  0.92819
Feature4TMO                         -0.0039988  0.0102067  -0.392  0.69522
Feature4VER                         -0.0069421  0.0130279  -0.533  0.59413
DayOfWeekMonday                     -0.4224260  0.0181773 -23.239  < 2e-16
DayOfWeekSaturday                    0.6534100  0.0235149  27.787  < 2e-16
DayOfWeekSunday                      0.8525094  0.0252939  33.704  < 2e-16
DayOfWeekThursday                   -0.0455434  0.0198962  -2.289  0.02208
DayOfWeekTuesday                    -0.2824596  0.0188918 -14.951  < 2e-16
DayOfWeekWednesday                  -0.1087561  0.0195987  -5.549 2.87e-08
CallCategoryPreQ                     0.0035326  0.0091772   0.385  0.70029
Feature1A+:DayOfWeekMonday          -0.0107165  0.0191100  -0.561  0.57495
Feature1AB-:DayOfWeekMonday          0.0249549  0.0189856   1.314  0.18871
Feature1AB+:DayOfWeekMonday          0.0014391  0.0141689   0.102  0.91910
Feature1A+:DayOfWeekSaturday        -0.0053803  0.0250491  -0.215  0.82993
Feature1AB-:DayOfWeekSaturday        0.0130871  0.0247023   0.530  0.59625
Feature1AB+:DayOfWeekSaturday       -0.0066196  0.0184979  -0.358  0.72045
Feature1A+:DayOfWeekSunday           0.0074346  0.0268166   0.277  0.78160
Feature1AB-:DayOfWeekSunday         -0.0303076  0.0267220  -1.134  0.25672
Feature1AB+:DayOfWeekSunday         -0.0286588  0.0200235  -1.431  0.15236
Feature1A+:DayOfWeekThursday         0.0076765  0.0209352   0.367  0.71386
Feature1AB-:DayOfWeekThursday        0.0370891  0.0208821   1.776  0.07571
Feature1AB+:DayOfWeekThursday        0.0038332  0.0155710   0.246  0.80555
Feature1A+:DayOfWeekTuesday         -0.0054692  0.0198035  -0.276  0.78242
Feature1AB-:DayOfWeekTuesday         0.0184158  0.0197858   0.931  0.35198
Feature1AB+:DayOfWeekTuesday         0.0007729  0.0147232   0.052  0.95814
Feature1A+:DayOfWeekWednesday        0.0036814  0.0205080   0.180  0.85754
Feature1AB-:DayOfWeekWednesday       0.0019106  0.0204954   0.093  0.92573
Feature1AB+:DayOfWeekWednesday       0.0031410  0.0152829   0.206  0.83716
Feature2PreQ:DayOfWeekMonday         0.0021678  0.0117391   0.185  0.85349
Feature2PreQ:DayOfWeekSaturday       0.0046068  0.0153241   0.301  0.76370
Feature2PreQ:DayOfWeekSunday         0.0167280  0.0165561   1.010  0.31231
Feature2PreQ:DayOfWeekThursday      -0.0034615  0.0128900  -0.269  0.78828
Feature2PreQ:DayOfWeekTuesday        0.0153604  0.0122141   1.258  0.20854
Feature2PreQ:DayOfWeekWednesday     -0.0116048  0.0126647  -0.916  0.35950
Feature3I:DayOfWeekMonday            0.0343123  0.0152592   2.249  0.02454
```



```
Feature3M:DayOfWeekMonday             0.0166342  0.0142561   1.167  0.24329
Feature3I:DayOfWeekSaturday           0.0034390  0.0194247   0.177  0.85947
Feature3M:DayOfWeekSaturday          -0.0030925  0.0186341  -0.166  0.86819
Feature3I:DayOfWeekSunday             0.0567251  0.0208693   2.718  0.00657 :
Feature3M:DayOfWeekSunday             0.0170903  0.0201071   0.850  0.39535
Feature3I:DayOfWeekThursday           0.0156511  0.0166111   0.942  0.34609
Feature3M:DayOfWeekThursday          -0.0017716  0.0156269  -0.113  0.90974
Feature3I:DayOfWeekTuesday            0.0144467  0.0158496   0.911  0.36204
Feature3M:DayOfWeekTuesday           -0.0011835  0.0148258  -0.080  0.93637
Feature3I:DayOfWeekWednesday          0.0273872  0.0163501   1.675  0.09392
Feature3M:DayOfWeekWednesday          0.0098286  0.0153604   0.640  0.52226
Feature4TMO:DayOfWeekMonday           0.0047262  0.0130558   0.362  0.71735
Feature4VER:DayOfWeekMonday           0.0110747  0.0166877   0.664  0.50692
Feature4TMO:DayOfWeekSaturday        -0.0092492  0.0170467  -0.543  0.58742
Feature4VER:DayOfWeekSaturday        -0.0090774  0.0217483  -0.417  0.67640
Feature4TMO:DayOfWeekSunday           0.0017895  0.0184220   0.097  0.92261
Feature4VER:DayOfWeekSunday           0.0031978  0.0234809   0.136  0.89167
Feature4TMO:DayOfWeekThursday         0.0081991  0.0143425   0.572  0.56755
Feature4VER:DayOfWeekThursday         0.0064672  0.0182906   0.354  0.72365
Feature4TMO:DayOfWeekTuesday          0.0099169  0.0135817   0.730  0.46529
Feature4VER:DayOfWeekTuesday         -0.0103858  0.0173713  -0.598  0.54993
Feature4TMO:DayOfWeekWednesday       -0.0068091  0.0140926  -0.483  0.62897
Feature4VER:DayOfWeekWednesday        0.0034393  0.0179915   0.191  0.84840
DayOfWeekMonday:CallCategoryPreQ      0.0028153  0.0117370   0.240  0.81044
DayOfWeekSaturday:CallCategoryPreQ   -0.0204160  0.0153221  -1.332  0.18271
DayOfWeekSunday:CallCategoryPreQ     -0.0039438  0.0165519  -0.238  0.81167
DayOfWeekThursday:CallCategoryPreQ    0.0048002  0.0128889   0.372  0.70958
DayOfWeekTuesday:CallCategoryPreQ     0.0122900  0.0122125   1.006  0.31425
DayOfWeekWednesday:CallCategoryPreQ   0.0029642  0.0126632   0.234  0.81492
```

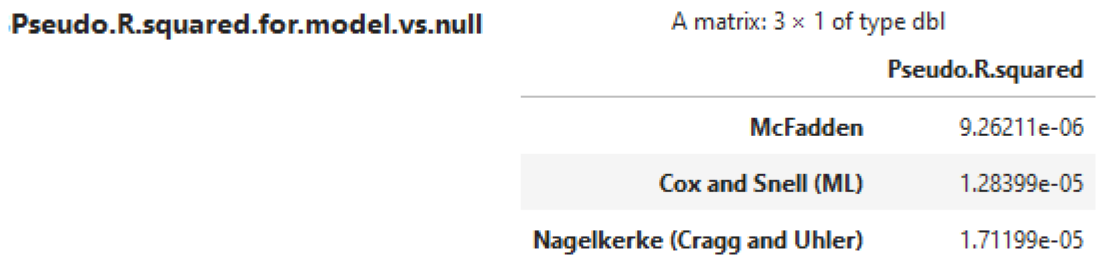
The summary statistics helps us in understanding the model better by providing us with the following information:

- Distribution of the deviance residuals
- Intercept and slope estimates along with the standard error, z-value, and p-value
- AIC value
- Residual Deviance (for fitted model) and Null Deviance (for intercept model)

The coefficient table shows that none of the explanatory variables have significant influence (p-values ¿ 0.05) on CallCategory.The difference between Null deviance and

Residual deviance tells us that the model is a good fit. Greater the difference better the model. Null deviance is the value when you only have intercept in your equation with no variables and Residual deviance is the value when you are taking all the variables into account. It makes sense to consider the model good if that difference is big enough. The difference in our case doesn't seem to be very large. Additionally, we also applied two other tests:

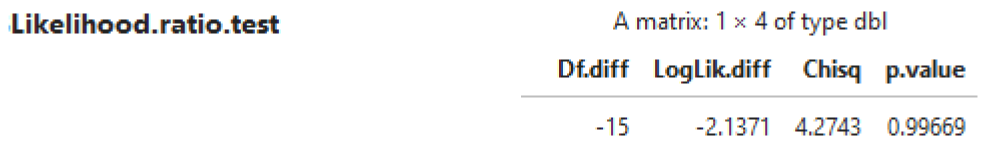
- Pseudo R-Squared and Likelihood Ratio Test



Pseudo.R.squared.for.model.vs.null

A matrix: 3 × 1 of type dbl

	Pseudo.R.squared
McFadden	9.26211e-06
Cox and Snell (ML)	1.28399e-05
Nagelkerke (Cragg and Uhler)	1.71199e-05



Likelihood.ratio.test

A matrix: 1 × 4 of type dbl

Df.diff	**LogLik.diff**	**Chisq**	**p.value**
-15	-2.1371	4.2743	0.99669

  –

  – The McFadden Pseudo R-squared value is the commonly reported metric for binary logistic regression model fit. The table result showed that the McFadden Pseudo R-squared value is $9.26211e-06$, which indicates the model has no predictive ability, although the likelihood value for the current model will be (it is always) larger than the likelihood of the null model, it will not be much greater. Therefore it will be close to zero, as we would hope.

    Additionally, the table provides a Likelihood ratio test. Likelihood Ratio test (often termed as LR test) is a goodness of fit test used to compare between two models; the null model and the final model. The null hypothesis is that the smaller model is the "best" model; It is rejected when the test statistic is large. In other words, if the null hypothesis is rejected, then the larger model is a significant improvement over the smaller one. The test revealed that the Log-Likelihood difference between intercept only model (null model) and model fitted with all independent variables is -18970 and p-value 0 which shows the improvement is significant as $(p - value < 0.05)$ and we reject the null hypothesis.

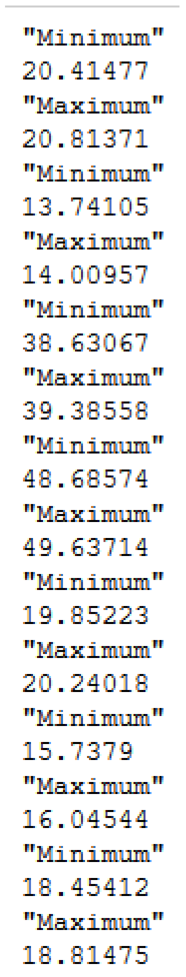
- AIC - Akaike Information Criteria
    - It is analogous to adjusted R^2 and is the measure of fit which penalizes model for the number of independent variables. We always prefer a model with minimum AIC value. Since from the table it shows that the null model is the best model we pick the model within two AIC values

The model which has Feature2 and Call Category have the minimum AIC value. Hence, we choose these features for our final model. None of the interactions came out to be important.

Prediction Model

As per the above configurations, we choose and trained a model.We ran the model on both our test and train dataset. On Test Dataset, we obtained an RMSE of approximately 23 and on Training Dataset, an RMSE of approximately 22.

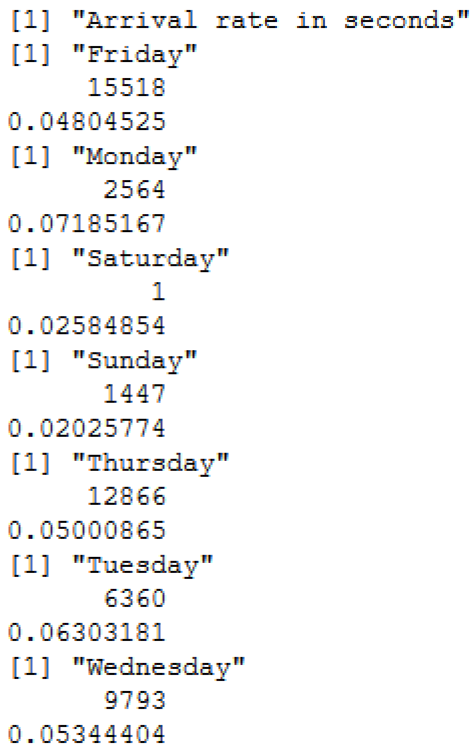
Now we calculate the range of predictions for each day of the week to see if there is a significant difference. We obtained the following values:



```
"Minimum"
20.41477
"Maximum"
20.81371
"Minimum"
13.74105
"Maximum"
14.00957
"Minimum"
38.63067
"Maximum"
39.38558
"Minimum"
48.68574
"Maximum"
49.63714
"Minimum"
19.85223
"Maximum"
20.24018
"Minimum"
15.7379
"Maximum"
16.04544
"Minimum"
18.45412
"Maximum"
18.81475
```

Now the minimum and maximum values of the predictions for each day of the week are very close. Hence we can directly calculate arrival rate for each day by picking any row from the dataset containing that day of week because essentially our GLM is giving us the expected value of the response variable with respect to the explanatory variables. In other words our GLM is giving us the mean arrival difference when we give the values of explanatory variables as input.

So now we pick a random row containg each day of the week from the dataset and get the expected arrival difference for that day since we already saw that for each day of week for any combination of other variables we were roughly getting the same arrival difference. Now we can calculate the arrival rate by: $1/(E[Y|X_i])$ where $E[Y|X_i]$ is the arrival difference, as shown below:



```
[1] "Arrival rate in seconds"
[1] "Friday"
     15518
0.04804525
[1] "Monday"
      2564
0.07185167
[1] "Saturday"
         1
0.02584854
[1] "Sunday"
      1447
0.02025774
[1] "Thursday"
     12866
0.05000865
[1] "Tuesday"
      6360
0.06303181
[1] "Wednesday"
      9793
0.05344404
```

6. Future Work

The results obtained from module are satisfactory in the sense that they are a result of data analysis and statistics techniques. However, although the results are correct, the data that was provided had features such that they were uniformly distributed and hence, there were no correlations in the data between our output variable and features.

Therefore, much better outcomes could be obtained if we work on a more improved dataset in terms of both its amount and the correlation that it possess.

Secondly, another future work revolves around the creation of an automation system that takes in the data in itself through a client interface, performs data analysis and on a fixed set of outcomes, determines the type of model that needs to be applied.

Appendix A. More Math

Here, we describe the background math for the techniques used in the text.

Appendix B. Data

Here is a dump of our 2TB data set. Enjoy!

Appendix C. Code

Our code can be found at: https://github.com/EmadBinAbid/pcatr