Cloud Computing

Project Report

Subway Delay Predictor

Bhavana Ramakrishna - br1525
Nirupama Suresh - ns3981
Shruthi Nemu Tandel - snt288

Abstract

Subway Delay Predictor is an intelligent mobile application built on a serverless platform. The application uses the historic weather and subway delay time data from the year 2015 to predict subway delay time based on the weather predicted for the next five days. The project focuses on creating a scalable architecture to accept input from the user, predict from a collection of data and store the results for further analysis on the AWS platform.

Contents

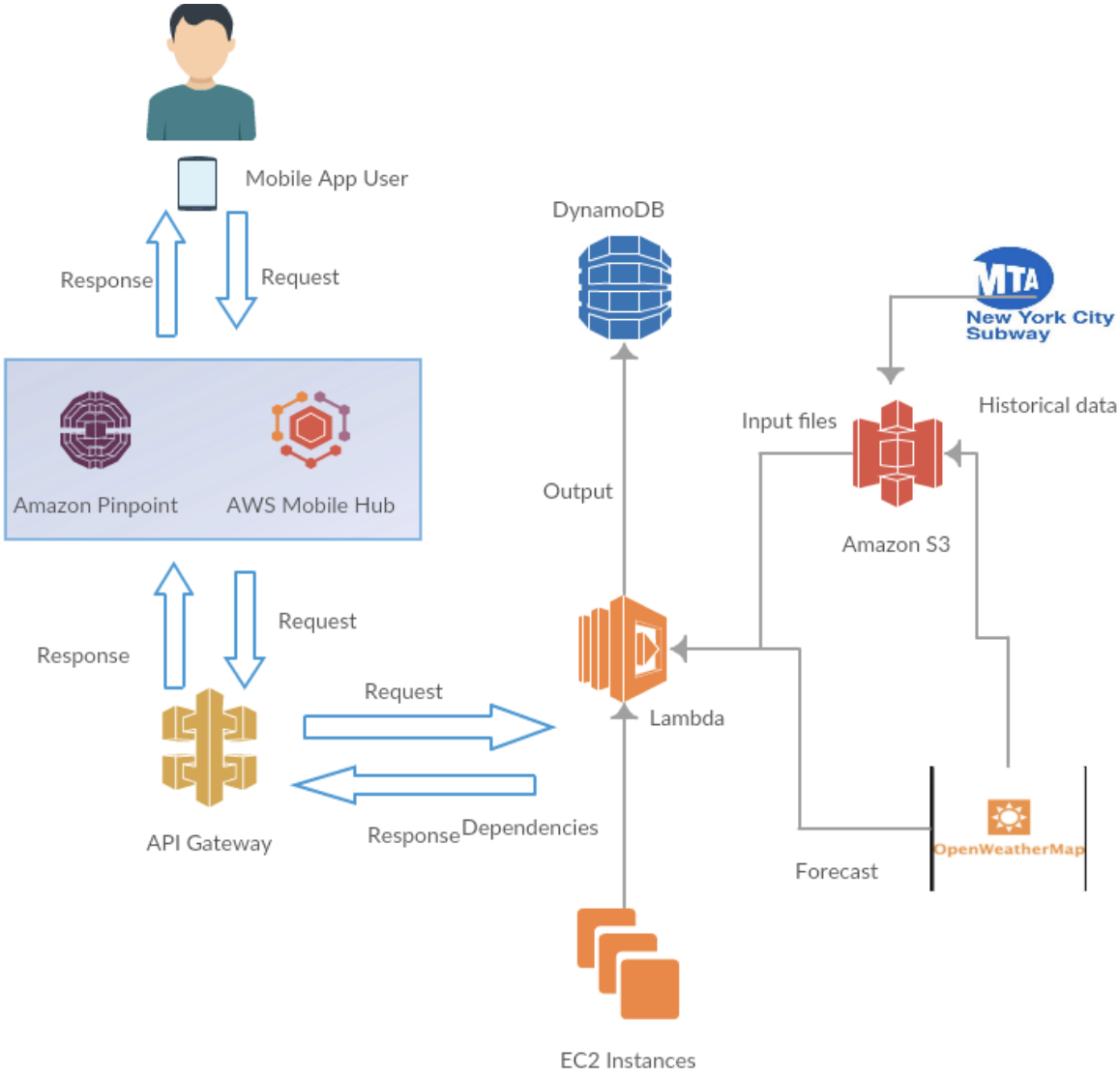
1. Introduction

If you commute on the New York city's subway on a regular basis, chances are you are affected by subway delays at some point in time. Although there are various reasons for the delays to be caused, there is one factor which affects the subway schedule almost everytime. Weather, the correlation between weather and the subway delay is high, especially during the winter. When cold or inclement weather is in the forecast, MTA services are subject to change as conditions demand. So, here is an application which aims to predict subway delays accurately on a daily basis depending on the weather forecast. This application can be used as a guide to planning your trip on NYC subways.

2. Methodology

2.1 Architecture

The entire application is built on AWS environment. Data collected from MTA New York City Subway and Open Weather Map are stored in Amazon S3 bucket. The user can send the request through an Android Mobile Application, which is deployed on AWS Mobile Hub. The request is pushed to a Lambda function when the request triggers the API Gateway. The application uses Linear Regression implemented using various libraries that are installed and compiled in EC2 instances, to predict the results which are sent back through the pipeline and also stored in DynamoDB for further analysis. Amazon Pinpoint is used to regularly push updates on a daily basis.



2.2 Data Collection

The project makes use of two sets of historical data to perform predictions. Firstly, MTA Subway Performance Data, a collection of data describing average delay time per month for various lines of the train starting January 2015, this set also includes the average number of passengers at the

subway station per day for every month. The second set of data gives historical information about weather conditions including temperature, precipitation and dew point starting January 2015 on a monthly basis.

2.3 Data Cleaning

The data collected are cleaned to arrive at the format required to perform predictions. The datasets are merged based on the common month column, giving us weather inputs and the corresponding subway delays. Due to the lack of subway delay data available, for the purpose of this project, we used the intuition from the correlation between monthly weather data and monthly delay data to be normalized to days, thus arriving at a single dataset containing maximum and minimum temperatures during a time frame and their respective average delay time for each subway line. Finally, this data is stored onto an S3 bucket in .csv format for further use.

2.4 Prediction

The prediction algorithm is implemented in an AWS Lambda function which will be executed when there is a request from the user. The data stored in S3 bucket is filtered based on the subway line as requested by the user, thus generating the training set. Weather forecast for the next five days is obtained through an API call to OpenWeatherMap API, this forms our testing set. To perform prediction, various libraries were used like Python Pandas, Numpy, and Scikit-learn which are installed and zipped along with the lambda function code in EC2 instance. The package was then uploaded to lambda function as a zip file through S3. This is considerably faster than the alternative of uploading the zip to Lambda directly. Machine Learning Linear Regression algorithm is used to predict the average delay time with 74% accuracy.

2.5 Output Storage

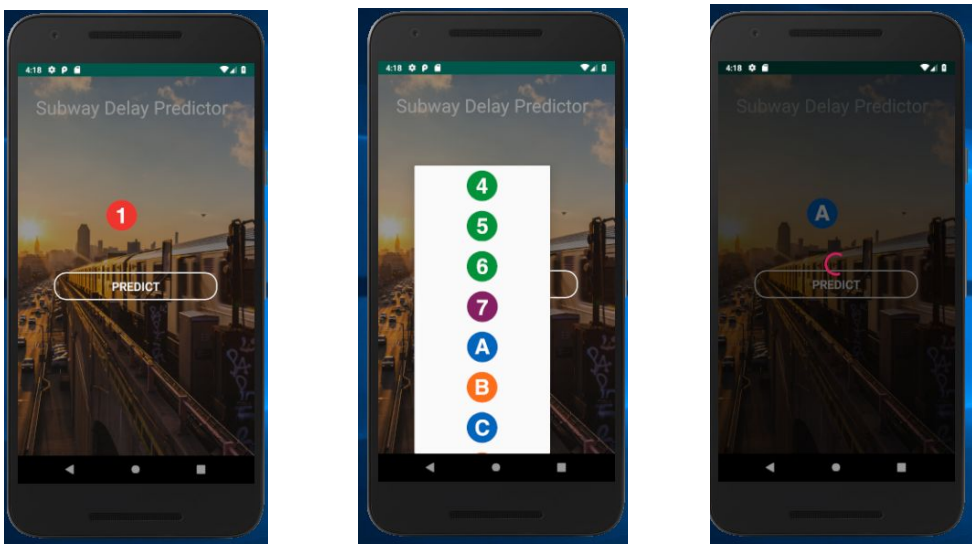
Once the output values are predicted, they are sent back on the pipeline through API Gateway to the front end. The output results are also stored in a NoSQL database, DynamoDB to determine the accuracy, trend and perform data analysis in the future.

2.6 Mobile Application

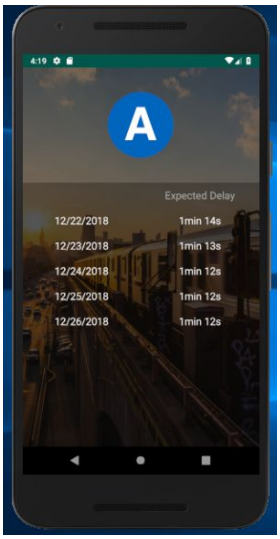
The front end interface is an android mobile application. It provides a simple interface for the user to select the subway route they are interested in and produces results for the next five days and their respective delays.

The user is presented with the first screen that mainly provides two functionalities:

1. Choose a subway route
2. Click predict to get the delays

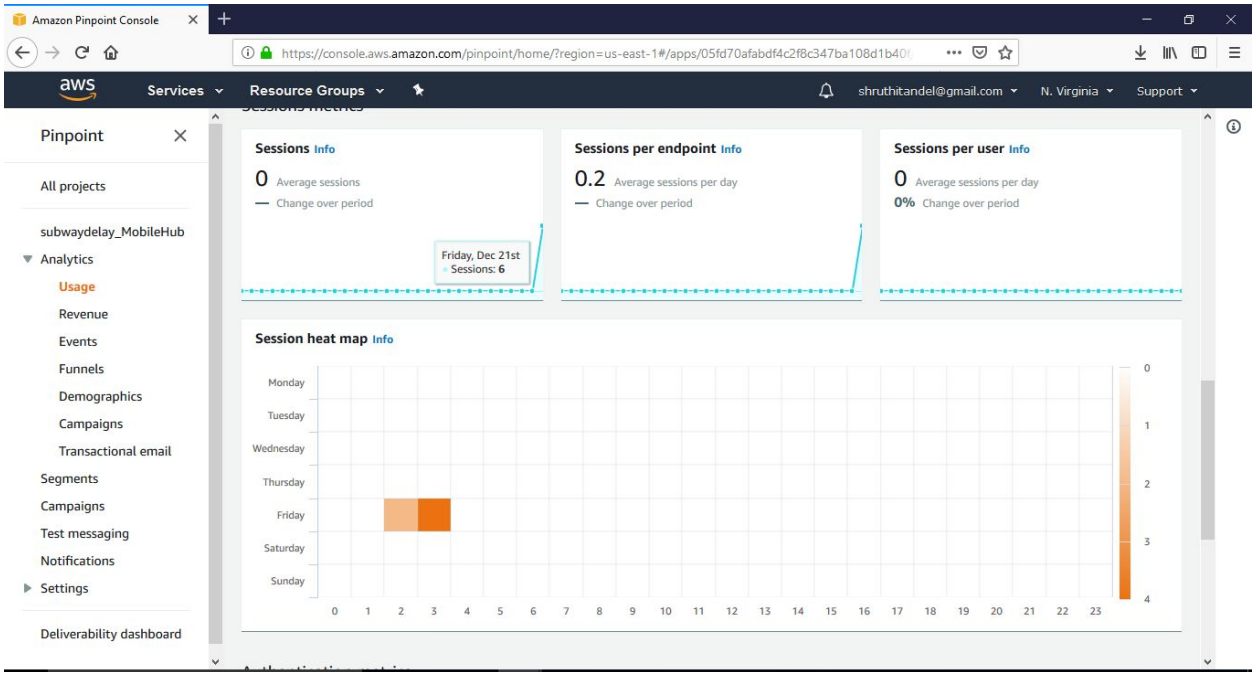


The next screen provides the results of delays over the next five days:



The application is also integrated with AWS Mobile Hub

AWS Mobile Hub has also been configured to monitor usage statistics of the number of sessions of prediction from users



3. Obstacles

Here are a few challenges that we faced while building this project.

Obtaining historical data for New York MTA subway delay was not very easy which is why we are using the data which gives the overall information in months.

Initially, we aimed to used Amazon Machine Learning which proved to predict results with more accuracy. However, every batch prediction takes at least 2 minutes to be executed, this could have been addressed by triggering the prediction algorithm every one hour or so. However, at this point, the priority was to have a real-time system which performs when a request is triggered. Another solution for this is to use AWS Sagemaker, however, since it charged us after a certain number of prediction, we did not continue with that.

Lastly, to implement predictive modeling, we had to import pandas and other machine learning libraries like sci-kit learn. We first tried creating a python virtual environment in our windows machine but we couldn't get rid of the python dependency package errors. This happens when the dependencies are installed via versions of Python which differ from the version AWS is running. So we then spun up an EC2 instance to resolve the issue.
In the future, we would like to overcome these obstacles by making use of various other resources to improve the performance.

4. Conclusion

To sum up our project, we successfully implemented a scalable, serverless, event-driven mobile application using various microservices. With the given data, we were able to arrive at nearly accurate predictions for 7 out of 10 inputs.

5. Future Work

In the future, we would like to focus on the following areas. We would like to subscribe to MTA, obtain more historical data as we need along with a forecast on an hourly basis and add to our database, DynamoDB in real-time. With more data, we would like to perform real-time validation in-order to improve the prediction accuracy.
We would also like to successfully implement the prediction algorithm in Amazon Machine Learning to improve the accuracy of the predictions.
We also plan on implementing push notification using Amazon Pinpoint which alerts the user about the train delay based on his travel trend.
Lastly, we like to use the information obtained based on predictions to perform some analysis on trends, changing weather and create alerts to take precautionary measures.

6. Code

The code and other resources for this project can be found on Github at https://github.com/BhavanaRamakrishna/Subway-Delay-Prediction

The code for the android app can be found on Github at https://github.com/nirupamasuresh/subway-delay