Software Design Document

for

Traffic Monitoring with Machine Learning

Version 1.0 approved

Prepared by Hue Ngo, Javier Hernandez, Russell Carter,

Daniel Caceres, Vrezh Khalatyan, Gracie Zamora

LA Data Science Federation and LADOT

March 12, 2019

Los Angeles, California

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 System Overview 1
2. Design Considerations 2
2.1 Assumptions and Dependencies 2
2.2 General Constraints 2
2.3 Goals and Guidelines 3
2.4 Development Methods 4
3. Architectural Strategies 4
4. System Architecture 5
5. Polices and Tactics 6
5.1 Specific Product Used 6
5.2 Requirements Traceability 6
5.3 Testing the Software 6
5.4 Choice of Particular Algorithm 6
5.5 Interfaces for End-users 6
6. Detailed System Design 7
6.1 Data Preparation Engine 7
6.2 Data Analytics Engine 7
6.3 Data Visualization Engine 8
7. User Interface 9
7.1 Overview of User Interface 9
8. Requirements Validation and Verification 12
9. Glossary 12
10. References 13

Revision History

Name	Date	Reason For Changes	Version
Hue	11/14/18	Revised sections 1, 2, 4, 5, Appendix	1.0
Hue	11/18/18	Revised sections 2, 3, 5, 6, 8, 10, 11	1.0
Hue	12/1/18	Revised sections 6, 7, 9	1.0
Hue	2/24/19	Revised all sections for final draft	1.0

1. Introduction

The Traffic Monitoring with Machine Learning application is used to predict and visualize traffic conditions. The application utilizes machine learning fundamentals to find traffic propagation patterns and make predictions based on various features. The application is also developed as a web application to visualize.

1.1 Purpose

The purpose of this document is to explain in detail the design aspects of the many functions that the application will perform. The version being referred to is version 1.0.0 as there are no other revisions or updates that have been processed. There will be a complete overview of all aspects of the application, including the functions of the application, the operating requirements, and the database requirements. The document will inform readers as to what the application will do.

1.2 Document Conventions

The terms *product* and *application* will be used interchangeably to refer to the Traffic Monitoring with Machine Learning application. External links will be underlined and have blue color text. A list may be in the format of a table or bullet points. Illustrations may accompany some technical specifications for easy understanding and simplification. For clarification, any sections of this document that contain <work in progress> mean that the developers do not have enough progress to fill out the sections.

1.3 Intended Audience and Reading Suggestions

The main audience of the software requirements specifications document are developers, project managers, testers, marketing staffs, and researchers. It is suggested that if this is your first read through to quickly skim through the entire document to further your knowledge and the usage of the application. For developers, some main points to focus on can be the product functions and the database requirements. For project managers the user interface, functional and database requirements are good points to read over. Testers may go over the user documentation, the assumptions and dependencies, user interfaces, and performance requirements sections. The marketing staff and researchers may go over the user interface and performance requirements to start off with. But all should go over the user, hardware, software, and communications interfaces as well as the safety requirements of the application.

1.4 System Overview

The scope of this application is to model the road network using various geographic data. This road network is then transformed into a Network Linkage, which provides the available routes vehicles can travel on. The data from the Network Linkage may provide insightful propagation patterns of traffic

congestions, which is used in a machine learning algorithm to predict future traffic jams. These traffic jams are visualized in a web application.

2. Design Considerations

Listed are the various issues that need to be addressed before attempting to devise a complete design solution:

- To create a data pipeline that can process big data
- To understand the Waze data and other traffic information presented
- To develop a machine learning algorithm that predict traffic conditions based on geolocations and time units
- To preprocess data effectively

2.1 Assumptions and Dependencies

Due to time constraint, some of the features may be removed or modified. The application depends on the Waze data, so any delay in acquiring the Waze data may cause the application to be incomplete by the delivery date. The application requires secure HTTP connection to support the web component of the application. The responsiveness of the application user interface may vary, depending on the browsers and the screen or window size of the end-user. The application also depends on the connection of PostgreSQL that store the Waze data.

2.2 General Constraints

The list describes the global limitations or constraints that have a significant impact on the design of the system's software:

- Hardware or software environment
    - Python 3.6.x to be used in Data Preparation Engine and Data Analytics Engine.
    - Jupyter Notebook and Python 3.6.x interpreter as the environment to perform data preprocessing, data training, and data testing.
- End-user environment
    - A browser is required to view the visualizations.
    - Basic computer inputs and outputs shall be provided by the user such as mouse, keyboard, monitor screen, and desktop.
    - End-user must have valid accounts from Esri ArcGIS prior to using the application.
- Standards compliance
    - Data Visualization Engine shall follow the standards-compliance of World Wide Web.
- Interoperability requirements
    - Developers require credential to connect to PostgreSQL database that store Waze dataset.
- Data repository and distribution requirements

    - Waze traffic data is used to perform data analysis.
- Network communications
    - Require secure internet connections.
- Verification and validation requirements (testing)
    - Train dataset must be tested against Test dataset to test for accuracy of the machine learning model.

2.3 Goals and Guidelines

Listed are the goals, guidelines, principles, and priorities which embody the design of the system's software:

- The application should follow test-driven development.
- The application should emphasize on data analysis to make better predictions.
- The application's dependencies should use be simple to maintain.
- Deadline:
    - Delivery Date: August 13, 2018 - May 9, 2019
    - Milestone 1 – August 13, 2018
        - Senior Design Initial Meeting
            - Be assigned a project
            - Meet advisor and teammates
        - Launch Day
            - Meeting with liaison
            - Learn more about project details
        - Develop meeting schedules with group and advisor
        - Assign lead roles to team
        - Research different technologies for architecture
        - Work on data visualizations and incorporate ArcGIS API
        - Research ArcGIS workflow and database compatibility
        - Attend ArcGIS workshop
        - Come up with initial system architecture design
        - Work on SRS
    - Milestone 2: October 19, 2018
        - Finalize system architecture
        - Complete SRS
        - Receive subsets of Waze data
        - Perform data preprocessing of subsets of Waze dataset
        - Speak with project manager
        - Research GRNN for data analytics
        - Complete SDD
        - 1st Presentation
    - Milestone 3: February 24, 2019

        - Completed linkage network
        - Performed initial training of model
        - Successfully trained with GPU using CUDA and cuDNN
        - Fixed major bug that prevent model from learning
        - Revised and finalized SRS and SDD
        - Use alternative data source to fill in for missing values
        - Retrained model
        - Validate and measure model for accuracy
        - Create visualization of prediction
    - Final Delivery Date: May 10, 2019

2.4 Development Methods

Since the nature of this software is based on newly found research and has little support from external sources, the development and implementation of the software will follow that of a Test-driven development to ensure that each stage of the process perform as intended.

3. Architectural Strategies

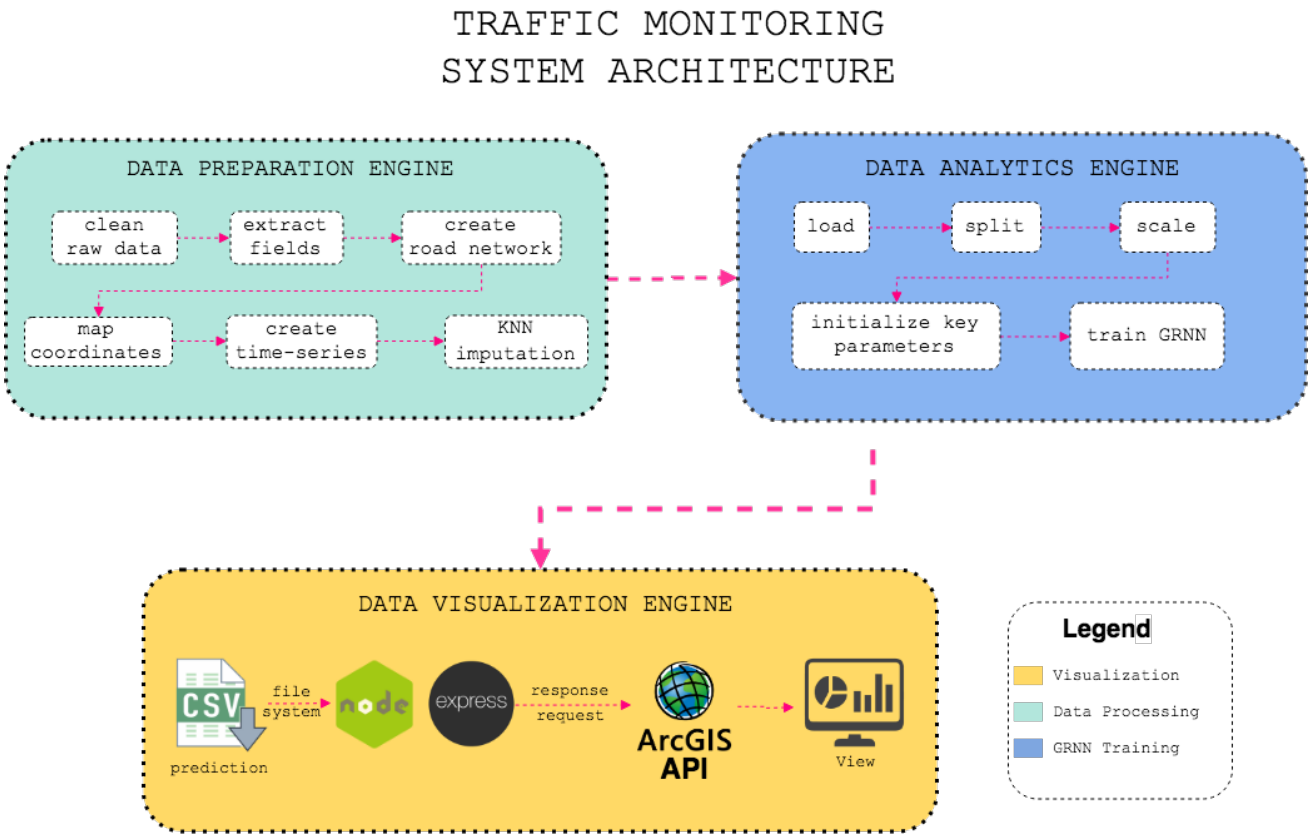
- Use of a product (programming language, database, library, etc.)
    - PostgreSQL database is used for storing output data
    - JavaScript and Typescript are used for Data Visualization Engine
    - Python is used for Data Preparation Engine and Data Analytics Engine
    - Pandas to manipulate and process data
- Reuse of existing software components to implement various parts or features of the system
    - The application is the first version, so there's no reuse of existing software components
- Plans for extending or enhancing the software
    - Dynamically update locations with traffic congestion
    - Display alternative routes
    - Allow users to filter traffic data for visualizing
    - Accuracy of machine learning algorithm is high
- User interface paradigms (or system input and output models)
    - A physical or virtual keyboard is required for users' input in text fields
    - A physical mouse is required for users' selections
- Hardware and/or software interface paradigms
    - Users' views will be updated with JavaScript Promise, Async, and Await
- Error detection and recovery
    - Error checking was done on the data being returned from the database, so that the views would accurately display the model.
- External databases and/or data storage management and persistence

- PostgreSQL is required to develop the database with the understanding that each of the members can read, write, and debug SQL queries when writing for the server-side.
- Database implementation is outside of the scope of this application. Liaison shall be responsible for database implementation and usage.

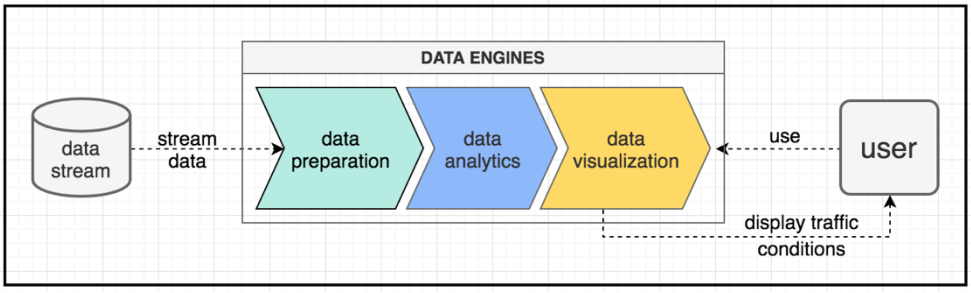
- Management of other resources
  - The database connection must be closed after use

4. System Architecture

Level 0: Data Engines Overview



Level 1: Engine Process Flow Overview



5. Polices and Tactics

The design tactics to implement the application includes first building the Data Preparation Engine that performs the data preprocessing. Second, building the Data Analytics Engine to extract and select features for training and testing the machine learning algorithm. And lastly, building the Data Visualization Engine to visualize the predicted output to display to users.

5.1 Specific Product Used

Jupyter Notebook and Python Interpreter shall be used to process the data provided. Python 3.6.x shall be used to develop both Data Preparation and Data Analytics Engines. Typescript shall be used to develop ArcGIS widgets and map feature layers for the user interface.

5.2 Requirements Traceability

The requirements of the application shall be traceable. The source code can be traced via GitHub version control.

5.3 Testing the Software

The Waze data shall be split to train and test the machine learning algorithm. Dummy data may be used in place to test widgets' functionalities if Waze data has not been preprocessed.

5.4 Choice of Particular Algorithm

The predictive model of this application is inspired by the Graph Recurrent Neural Network (GRNN) developed by Xiaoyu Wang, Cailian Chen, Yang Min, Jianping He, Bo Yang, and Yang Zhang from the Department of Automation from Shanghai Jiao Tong University. The application shall model the Network Linkage architecture from the features of the road network to gain insight on traffic congestion propagation pattern. These propagation patterns are used as input features for GRNN to mine and predict new propagation patterns.

5.5 Interfaces for End-users

The application shall provide end-users with a webpage to interface with the application. End-users may select data to filter what is displayed on the map. End-users may use the documentation from Esri ArcGIS to learn how to use the map feature layers and widgets.

6. Detailed System Design

The detailed engines described shall be the Data Preparation Engine, Data Analytics Engine, and Data Visualization Engine.

6.1 Data Preparation Engine

This engine shall preprocess data, fill in missing data fields using the most fitted missing-data imputation methods, organize data fields as features, and aggregate data to normalize and make sense of data.

6.1.1 Responsibilities

The flow of the data stream shall be determined by the Waze data providers. The stored data depends on the PostgreSQL configurations provided by the project administrator.

6.1.2 Constraints

There may be missing features and labels from the preprocessed data because the Waze data is inconsistent and flawed.

6.1.3 Compositions

The listed Jupyter Notebooks are the sub-modules that are used to preprocess the initial Waze datasets provided:

- AlertsDataCleanup.ipynb - provide data preprocessing for the alerts.csv
- IrregularitiesDataCleanup.ipynb - provide data preprocessing for irregularities.csv
- JamsDataCleanup.ipynb - provide data preprocessing for jams.csv

6.1.4 Resources

The listed resources are required for data preprocessing until the credential of the PostgreSQL connection is provided:

- alerts.csv
- irregularities.csv
- jams.csv

6.1.5 Interface/Exports

The output data shall be kept in the corresponding Jupyter Notebook of each dataset until the credential of the PostgreSQL connection is provided. Once the database connection is established, any input and output data shall be store in the PostgreSQL database.

6.2 Data Analytics Engine

This engine shall extract and select several features from the preprocessed data to the machine learning model. Part of the extracted and selected data shall be used to test the machine learning model for accuracy.

6.2.1 Responsibilities

This engine prepares processed data in the form that the machine learning algorithm can use. The output is the predicted traffic speed, type of alerts, and classification of the flow of traffic based on the date and time, and location.

6.2.2 Constraints

The output predicted may or may not precisely represent real-life traffic conditions because the machine learning algorithm is still in its early development. The date and time interval are averaged and rounded to the nearest interval to create the time-series necessary for the machine learning model.

6.3 Data Visualization Engine

This engine shall establish a HTTP connection and an API end-point by which the output data generated from the machine learning model from Data Analytics Engine is in the form of a JSON. The JSON shall be parsed from the back-end of the web application to be used for the front-end to display traffic data to end-user.

6.3.1 Responsibilities

The followings are constraints of Data Visualization Engine:

- The accuracy of the predicted traffic data depends on the machine learning model
- The smoothness of the visualization may be affected by the data stream, especially when traffic congestion is heavy

6.3.2 Constraints

The output predicted may or may not precisely represent real-life traffic conditions because the machine learning algorithm is still in its early development.

6.3.3 Compositions

The list below contains widget components for users to interface with the application:

- Zoom in and out to change the map visibility
- Display current location (require GPS to be enabled)
- Reference the Legend
- Select which data layer to be displayed
- Filter by date and time
- Search for a location in the map
- See summary of traffic in the form of charts
- Add new data to map
- Print the current map in different formats

6.3.4 Resources

The following data are used to visualize the traffic congestions.

- alerts.csv
- irregularities.csv
- jams.csv

6.3.5 Interface/Exports

The visualized data is displayed on a map layer created from ArcGIS JavaScript API.

7. User Interface

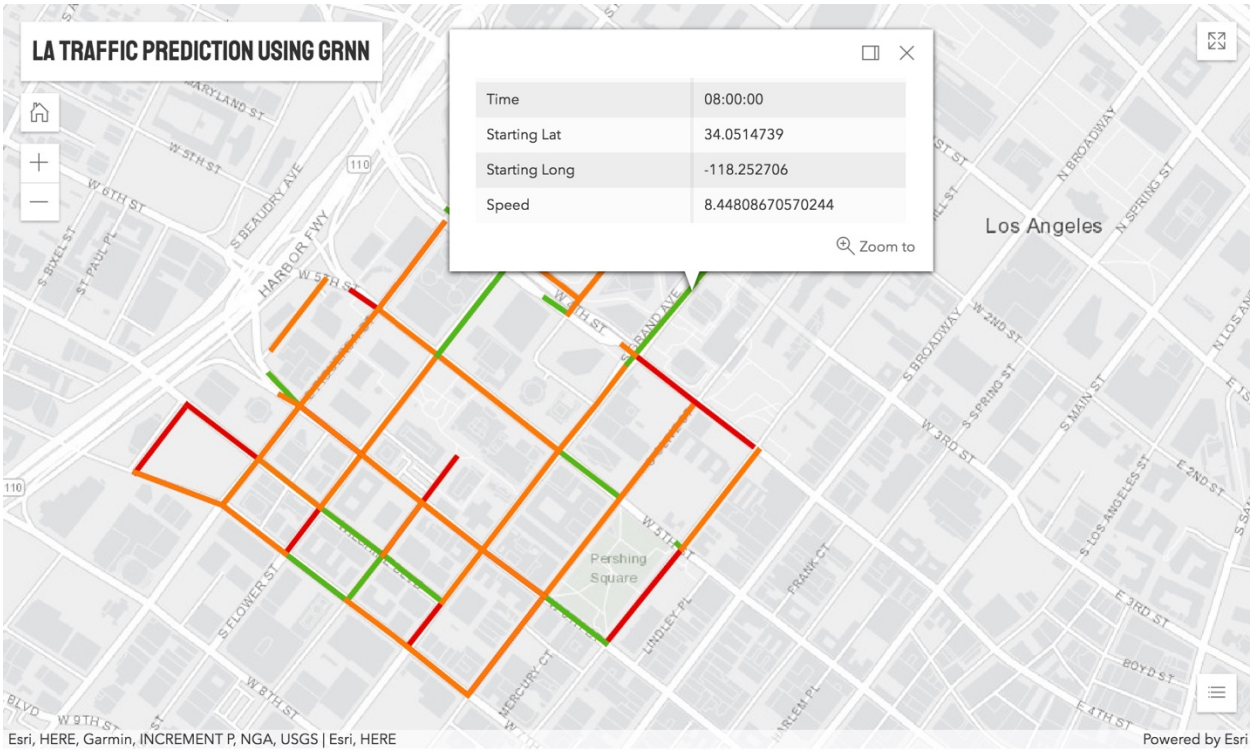
The user interface is designed and implemented as a single page web application using ESRI ArcGIS JavaScript API 4.11

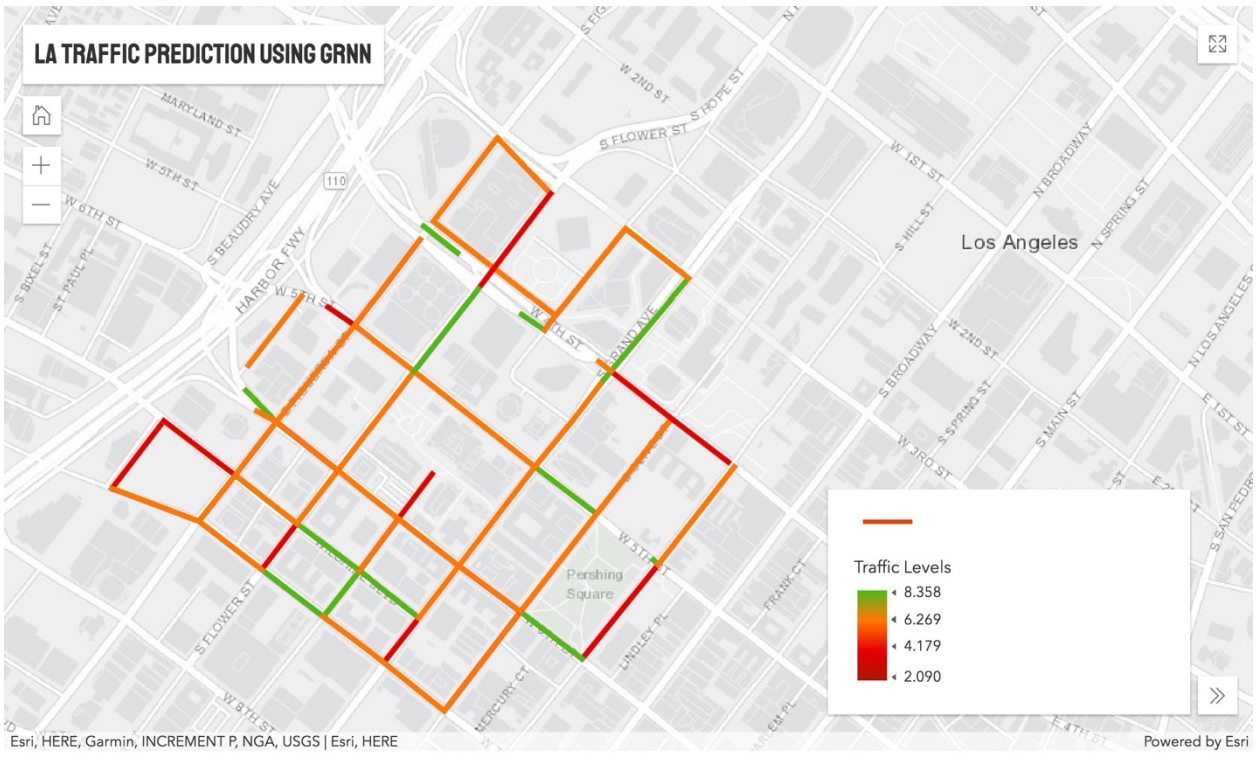
7.1 Overview of User Interface

From the user interface, users can:

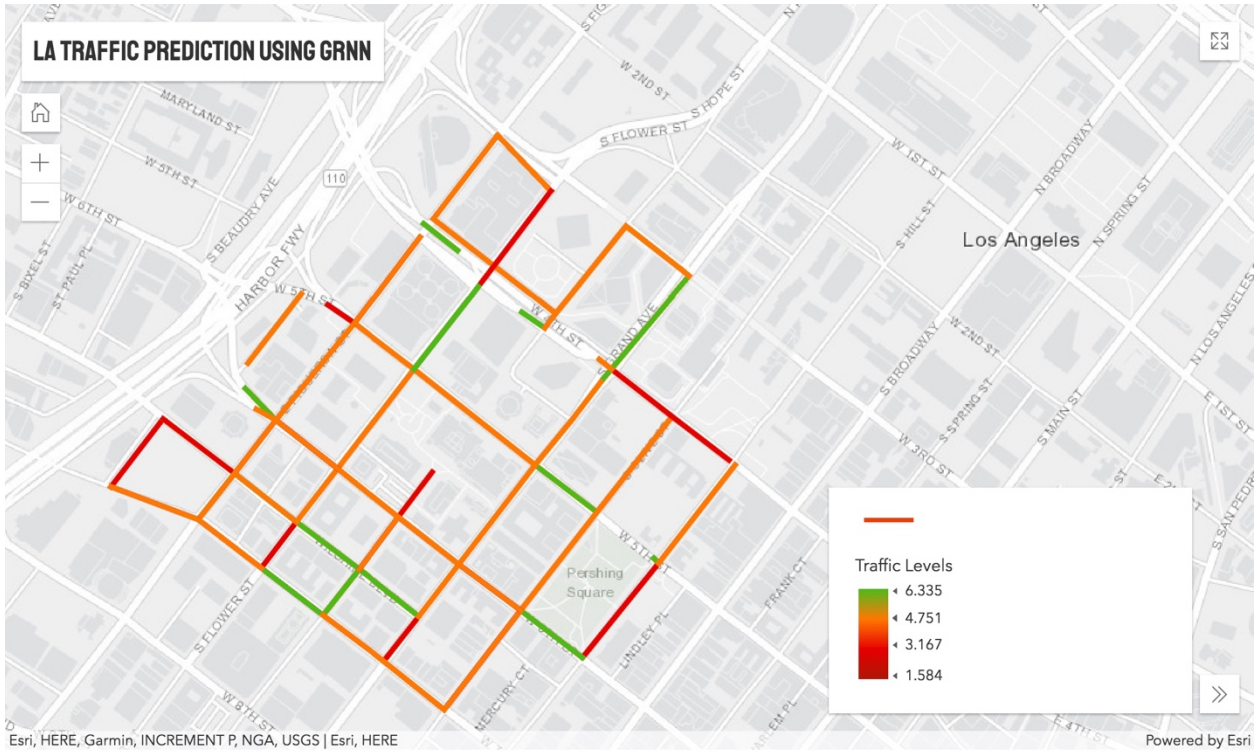
- Zoom in and out to change the map visibility
- Reference the Legend
- Filter by date and time using time slider
- Full screen mode
- Redirect to center of map
- Animate traffic through time


LA TRAFFIC PREDICTION USING GRNN
Los Angeles
Pershing Square
Esri, HERE, Garmin, INCREMENT P, NGA, USGS | Esri, HERE
Powered by Esri
07:00


LA TRAFFIC PREDICTION USING GRNN
Time 08:00:00
Starting Lat 34.0514739
Starting Long -118.252706
Speed 8.44808670570244
Zoom to
Los Angeles
Pershing Square
Esri, HERE, Garmin, INCREMENT P, NGA, USGS | Esri, HERE
Powered by Esri
08:00
Tuesday, December 12th 2017



07:30 Tuesday, December 12th 2017



07:00 Tuesday, December 12th 2017

8. Requirements Validation and Verification

The table displays the requirements of each engine, where it is satisfied in the project, and how the result is tested.

Requirement	Satisfy By	Testing Method
DPE-1: The engine shall average integer or decimal values.	Cleanup Jupyter Notebook	Test-driven development
DPE-2: The engine shall find the unique strings that exist in a column.	Cleanup Jupyter Notebook	Test-driven development
DPE-3: The engine shall replace empty fields with NULL values.	Cleanup Jupyter Notebook	Test-driven development
DPE-4: The engine shall reformat the column headers with values in all uppercase.	Cleanup Jupyter Notebook	Test-driven development
DPE-5: The engine shall replace NULL values in a non-id number column with the average of its sum.	Prediction Jupyter Notebook	Test-driven development
DAE-1: The engine shall group data based on time, day of the week, special events, and others to form the extracted features.	Prediction Jupyter Notebook	Test-driven development
DAE-2: The engine shall select specific features from the extracted features to train the predictive model.	Prediction Jupyter Notebook	Test-driven development
DAE-3: The engine shall train a model based on the dataset with the selected features.	Prediction Jupyter Notebook	Test-driven development
DAE-4: The engine shall test the trained model for accuracy.	Prediction Jupyter Notebook	Test-driven development
DVE-1: The engine shall retrieve data from the database.	Visualization Jupyter Notebook	Test-driven development
DVE-2: The engine shall visualize traffic data in a map as feature layers.	ArcGIS Online AppBuilder	Test-driven development
DVE-3: The engine shall summarize traffic data in forms of charts and graphs.	ArcGIS Online AppBuilder	Test-driven development
DVE-4: The engine shall visualize traffic data dynamically.	ArcGIS Online AppBuilder	Test-driven development
DVE-5: The engine shall query selected data per user requests via widgets.	Visualization Jupyter Notebook	Test-driven development

9. Glossary

The following table contains the terms and definitions to properly interpret the application.

Term	Definition
ArcGIS	Esri's all-in-one solution to work with geographic information.
CSS	Cascading Style Sheet is a style sheet that is used to describe the presentation of a markup language.

CSV	Comma Separated Values. File format that is used to store tabular data such as spreadsheets or databases.
DSF	Data Science Federation is a specialized department in the City of Los Angeles.
FE	Feature Extraction is a method to convert data into features usable in a model.
FS	Feature Selection is a method to remove unnecessary features in a dataset.
HTML	Hypertext Markup Language is the standard markup language for creating web pages.
JavaScript	A programming language that is heavily used for web applications.
LADOT	Los Angeles Department of Transportation is a specialized department of Los Angeles City.
Machine Learning	Predictive mathematical model used for making predictions.
MongoDB	A No-SQL database.
Operating System	The software that allows any computer to communicate, modify, and terminate any hardware and software communications based on end-users decisions.
Python	A general-purpose programming language that can also be used to program web application and data analytics application.
Pytorch	Machine learning library.
Runtime	The time when an application is executed.
SDD	Software Design Document is a document that specifies the design of the application on a technical and detailed level.
SRS	Software Requirement Specification is a document that specifies the functional and nonfunctional requirements of the application.

10. References

The references listed are used during the research phase to design the machine learning algorithm.

Daniel Wilson. Using Machine Learning to Predict Car Accident Risk. <https://medium.com/geoai/using-machine-learning-to-predict-car-accident-risk-4d92c91a7d57>

Danqing Kang, Yishen Lv, Yuan-yuan Chen. Short-term Traffic Flow Prediction with LSTM Recurrent Neural Network.

Xiaoyu Wang, Cailian Chen, Yang Min, Jianping He, Bo Yang, Yang Zhang. Efficient Metropolitan Traffic Prediction Based on Graph Recurrent Neural Network <https://arxiv.org/pdf/1811.00740.pdf>

Yusu Pan. Traffic Prediction Github. <https://github.com/corenel/traffic-prediction>