

Architecture Design

Inter-State Traffic Volume Prediction

Revision Number: 1.0

Last date of Revision: 06/01/2024

Document Version Control

Date Issued	Version	Description	Author
06/01/2024	1	Initial Architecture- V1.0	Ajith Kumar V HariHara Sudan R

Abstract

Machine Learning is a category of algorithms that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build models and employ algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. These models can be applied in different areas and trained to match the expectations so that accurate steps can be taken to predict real world scenarios. The goal of this project is to build a prediction model using machine learning techniques and to use a template to document the end-to-end stages. We're trying to forecast the value of a continuous variable with the Metro Interstate Traffic Volume dataset, which is a regression issue and deploy in AWS.

1. Introduction

1.1. What is Architecture Design?

The goal of Architecture Design (AD) or a low-level design document is to give the internal design of the actual program code for the `Bike Share Prediction System`. AD describes the class diagrams with the methods and relation between classes and program specification. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Architecture Design(AD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software, architecture, source code, and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work. And the complete workflow.

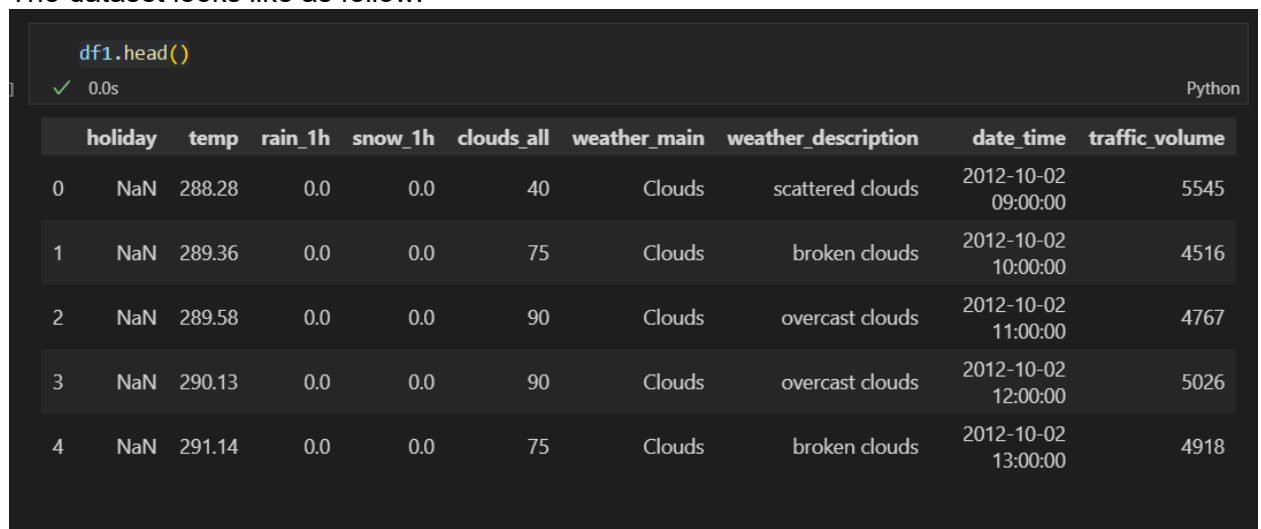
1.3. Constraints

We are predicting only for Metro Interstate area for which the data is provided.

2. Technical Specification

2.1. Dataset

The dataset looks like as follow:



	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	date_time	traffic_volume
0	NaN	288.28	0.0	0.0	40	Clouds	scattered clouds	2012-10-02 09:00:00	5545
1	NaN	289.36	0.0	0.0	75	Clouds	broken clouds	2012-10-02 10:00:00	4516
2	NaN	289.58	0.0	0.0	90	Clouds	overcast clouds	2012-10-02 11:00:00	4767
3	NaN	290.13	0.0	0.0	90	Clouds	overcast clouds	2012-10-02 12:00:00	5026
4	NaN	291.14	0.0	0.0	75	Clouds	broken clouds	2012-10-02 13:00:00	4918

The dataset snapshot provides valuable insights into weather conditions and traffic volume at specific date and time points. Notably, the "holiday" column indicates the absence of holidays for the initial entries, suggesting routine, non-holiday periods. Weather details reveal a moderate temperature range in Kelvin, with recorded rainfall or snowfall during the observed hours. The "clouds_all" column suggests varying levels of cloud coverage. The "weather_main" consistently categorizes the weather as "Clouds," and the accompanying "weather_description" offers additional details such as "scattered clouds," "broken clouds," "overcast clouds." etc The "date_time" column provides precise timestamps in the format "YYYY-MM-DD HH:MM:SS." Traffic volume, recorded in the "traffic_volume" column. This initial analysis sets the stage for further exploration, enabling a deeper understanding of patterns and correlations within the dataset.

```
df1.info()
[6] ✓ 0.0s Python
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48204 entries, 0 to 48203
Data columns (total 9 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   holiday              61 non-null     object
1   temp                 48204 non-null  float64
2   rain_1h              48204 non-null  float64
3   snow_1h              48204 non-null  float64
4   clouds_all           48204 non-null  int64
5   weather_main         48204 non-null  object
6   weather_description  48204 non-null  object
7   date_time            48204 non-null  object
8   traffic_volume       48204 non-null  int64
dtypes: float64(3), int64(2), object(4)
memory usage: 3.3+ MB
```

The dataset consists of various data types from integers to floating points and to object as shown in above Fig

```
df1.describe()
[5] ✓ 0.1s Python
temp      rain_1h    snow_1h    clouds_all  traffic_volume
count  48204.000000  48204.000000  48204.000000  48204.000000  48204.000000
mean    281.205870    0.334264      0.000222      49.362231     3259.818355
std      13.338232    44.789133      0.008168      39.015750     1986.860670
min       0.000000    0.000000      0.000000      0.000000      0.000000
25%      272.160000    0.000000      0.000000      1.000000     1193.000000
50%      282.450000    0.000000      0.000000     64.000000     3380.000000
75%      291.806000    0.000000      0.000000     90.000000     4933.000000
max      310.070000   9831.300000      0.510000    100.000000     7280.000000
```

The data reveals key statistics pertaining to temperature, rainfall, snowfall, cloud coverage, and traffic volume. The mean temperature is approximately 281.21 Kelvin, with an unusual minimum of 0 Kelvin that may indicate missing or erroneous data. Rainfall on average is low (mean = 0.33), but there's a notably high maximum value of

9831.3, suggesting potential outliers. Snowfall is minimal (mean = 0.000222). Cloud coverage averages around 49.36%, ranging from 0% to 100%. The traffic volume shows an average of 3259.82, ranging from 0 to 7280. These insights provide a foundational understanding of the dataset, highlighting temperature variations, precipitation levels, and their potential influence on traffic volume. Further analysis and visualization can offer more nuanced insights into patterns and relationships within the data.

2.2. Logging

Logging is like keeping a diary for our program. At every step, from bringing in the data, creating the model, training it, to making it live for use, we've made sure to write down what's happening. This way, if anything goes wrong or we need to understand how things are working, we can look at the log, just like reading through a diary to see what happened when. It helps us keep track of everything the program does.

2.3. Database

Database is used to store the dataset details only and we are fetching it using cassandra database driver and token provided.

2.4. Deployment

For hosting the project we will be using AWS EC2 and AWS ECR

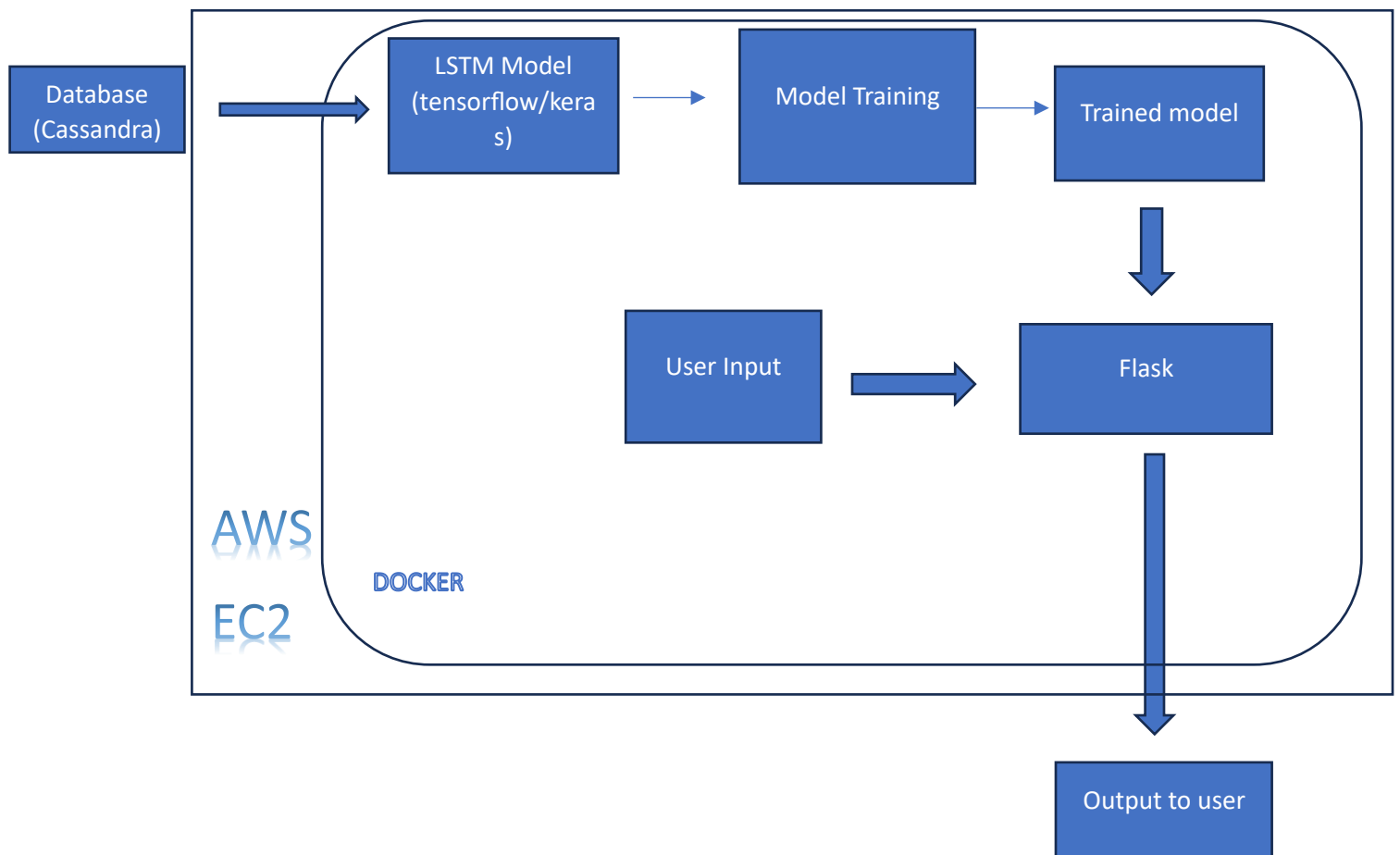
3. Technology Stack

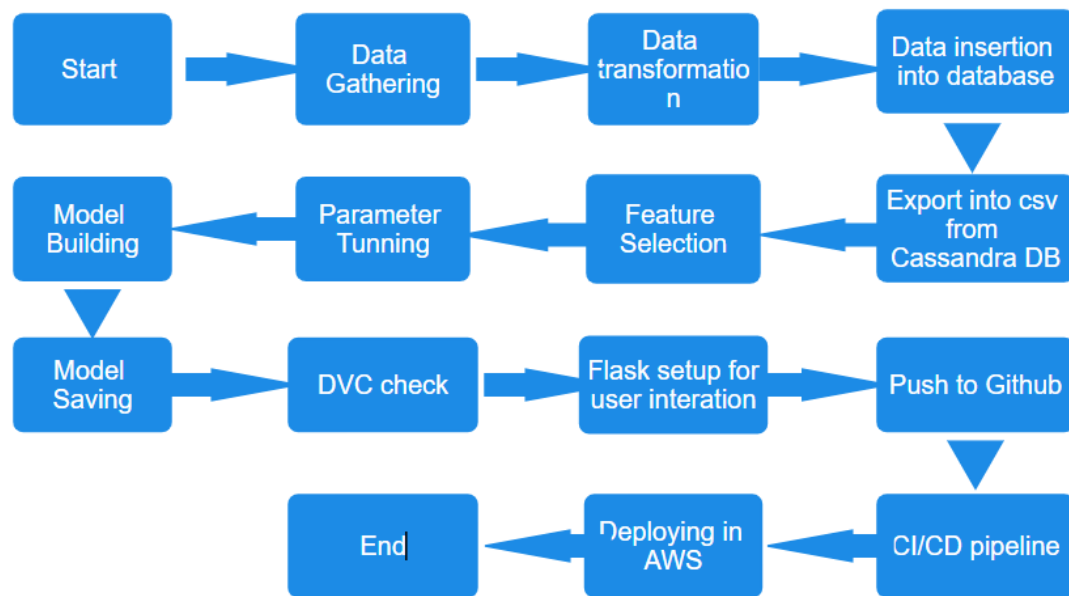
Front End	HTML/JavaScript
Backend	Python/Flask, Tensorflow/Keras
Database	Cassandra DB
CI/CD	Github CI/CD
Deployment	AWS EC2, AWS ECR

4. Proposed Solution

We will be using LSTM model for our regression task to predict the future traffic. The user will fill the required feature as input and will get results through web application. The system will get features and it will be passed into the backend where the features will be validated and preprocessed and then it will be passed to LSTM model to predict the final outcome. MLOps pipeline has also been added to the application will a single click user can fetch data from database, create a model, train it based on hyperparameters, evaluate it and use it in production.

5. Architecture





5.1. Data Gathering

Data Source: <https://archive.ics.uci.edu/dataset/492/metro+interstate+traffic+volume>
 Train and Test data are splitted in ratio of 80:20 using random seed during training and validating.

5.2. Data Transformation

Before sending data into database, data transformation is required so that data are converted into such form which can be easily interpreted by the model. Here, weather description, weather main were in textual form so they are modified into numeric form using one hot encoding and then datetime were in string format so they are splitted into year,month,date and hour so that it can be interpreted easily by the model.

5.3. Database insertion

Dataset is inserted into cloud database and here Cassandra DB is used which was specified as a requirement

5.4. Export as 'CSV' from database

From database we are making a CQL query using necessary drivers and tokens and the data is stored into CSV file. Now this CSV file is used for further processing.

5.5. Feature Selection

Necessary features like holiday data,time, rain fall amount, snow fall amount and other weather parameters are selected for processing. Inclusion of these key features aims to improve the accuracy and reliability of our predictive models ensuring a comprehensive consideration of factors influencing traffic volume.

5.6. Model Building

After doing all kinds of operations mentioned above our focus turns to the pivotal stage of model building.Following a thorough exploration and refinement of feature selection and parameter tuning we have opted for LSTM model. This choice is based on the fact

that LSTM is good in time series prediction. The constructed model comprises of two layers of LSTM which can accept a feature of (11,1) and two dense layers enhancing models capacity for capturing relationships within the data.

5.7. Parameter Tuning

Parameters like learning rate and epoch are chosen so that the loss converges. This ensures that our predictive models achieve optimal performance striking a balance between computational efficiency and the precision required for accurate traffic volume predictions.

5.8. Model Training

In the process of training our chosen LSTM model, we discovered that running it for 200 cycles (epochs) with groups of 500 data points (batch size) resulted in the validation loss settling down. This means the model became good at making predictions, providing us with reliable forecasts for traffic volume.

5.9. Model Saving

Model is saved using .h5 format using tf library

5.10 Flask Setup for user interaction

Setting up Flask for user interaction involves a few steps. First, after saving the model, we initiate the process of building an API using Flask. This creates a web application. Now, whenever a user enters data, it gets checked to make sure it's valid, and then it's sent to the model for predicting traffic volume. This way, users can easily interact with and get predictions from the model through the web application.

5.11 DVC Check

To make sure we're not doing the same thing over and over, we've included a Data Version Check (DVC check). This helps us avoid unnecessary repetition of a specific stage in our process. If we've already done something once, the check ensures we don't do it again, saving time and resources.

5.12 Github

The whole project is pushed into github repository and CI/CD pipeline is being used with AWS as host.

5.13 Deployment

We arranged things so our project could run on the internet using Amazon Web Services (AWS). Specifically, we used AWS ECR to store our software and AWS EC2 to run it. This setup allows our system to be accessible and functional in the cloud.

6. User Input/ Output Workflow

