Chenyu Yuan

chenyuyu

12-746 Fall 2020

Traffic Forecast Prediction by ARIMA model and Neural Network

Final Report

October 15, 2020

1. Introduction

This project focus on doing exploratory data analysis from current traffic situation we had in the dataset and giving the problem of forecasting the future traffic trend with time series data. We know that in urban network, there are many interesting things need discussion. By datasets with daily observation of traffic volume and intersection monitoring station id, we can easily identify the traffic volume situation in different place, and their time period of heaviest traffic volumes. The project will discuss the traffic volumes in a different level from daily to monthly and yearly, in order to give specific conclusions to traffic volume in different intersection and show forecast in a future trend.

2. Background

2.1 Detailed about dataset

Our dataset can be divided into two parts. The first part is time series data with 38 columns. In this part, we can find time traffic volume information from hourly to daily in different station intersections. We also have 'fips_state_code' column to tell us more about state information. And there is another column called 'direction_of_travel_name' to tell us the direction of sensors. For each record, we have an special station ID to represent monitoring stations. In addition, we have a series of columns from 'traffic_volume_counted_after_0000_to_0100' to 'traffic_volume_counted_after_2300_to_2400'. These columns tell us how much traffic volume in each hourly period in detail.

Another part of dataset is station information. From this dataset, we can get where the sensor located, and the longitude and latitude location of station. Again, this dataset also have 'fips_state_code' column to tell us which state the monitoring station located. We also have a column called 'station_location' to tell us detailed information such as number and streets of the monitoring station. All above geographic information helps us to identify specific station when doing ARIMA model in further steps.

2.2 Detailed about exploratory data analysis

Basically, when talking about a dataset, we need several steps to deal with it. The standard process of data analytics pipeline is extraction, acquisition, cleaning, analysis and presentation. We first have already had extraction and acquisition steps because we find traffic volume dataset with different stations and time periods through Kaggle.com. Then we need to clean up the outliers in the dataset and do some analysis on this dataset. The final stage will be presentation, from which we need to do some data visualization and telling interesting findings behind the data.

A standard data analysis requires us to ask different kinds of questions with different characteristics. For example, we need to first have some descriptive questions, in order to summarise different columns in a wider level. Then we need to do some exploratory work to our dataset, such as find some interesting patterns and trends behind each column and make some connections between different factors. Finally, I will set up some model to make predictions for some specific part. For our dataset, my initial thinking is to demonstrate the trend of traffic volume from hourly, weekly, monthly to yearly.

2.3 Detailed about Autoregressive Intergrated Moving Average

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity. [1]

ARMA model can be regarded as two parts, AR model and MA model. AR model talks the relationship about current value with historical value. We can take this formula to describe AR model. [2]

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

c: constant

ε_t: assumed random error with normal distribution

And MA model talks the error of MA model:

$$X_t = \mu + \sum_{i=1}^{p} \theta_i \varepsilon_{t-i} + \varepsilon_t$$

μ: the average value of time series

$\theta_1, \theta_2, \ldots, \theta_q$ are parameters

And we can combined AR and MA model to ARMA model, from which we can get ARMA(φ, θ), the function of our time series model.

2.4 Detailed about Neural Network

Besides ARIMA model, Multiple layer perceptron, a kind of neural network are also very popular on doing time series prediction. In our project, we use 'tensorflow' package to set up multiple layer perceptron model. Basically, a standard multiple layer perceptron model should contain steps below:

1. Firstly, we should establish a sequential model in tensorflow as a frame of multiple layer perceptron, which allow us to add layers of neuron further.
2. Then we need to construct layers for our multiplayer perceptron model. Generally, a multilayer perceptron contains input layer, hidden layer and output layer. We can include deep learning neuron layer such as maxpooling layer, convolutional layer and full connected layer in hidden layer to optimize the prediction.
3. After completing the structure of model, we need to choose optimizer to evaluate our deep learning model. In this step, we need to select optimization function and loss function to evaluate the performance of our model.
4. In the step, we need to enter our training dataset, to train our model and modify if loss entropy or mean squared error doesn't reach requirement.
5. Prediction on test data.

3. Implementation

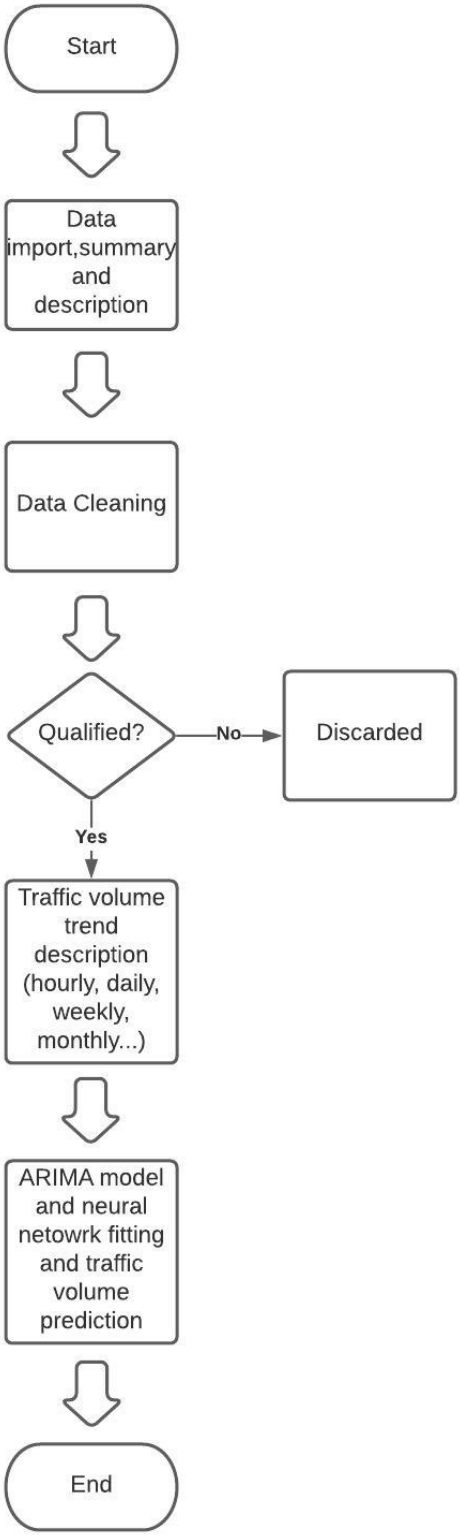

Figure 1 Flow chart of project

Here is the flow chart of my project. From the flow chart we can see the full process of data analysis of traffic volume dataset. First, I did traffic volume dataset summary and description in order to have a full version of my dataset. This process included an understand of how many columns and records we had in the dataset, and an initial statistical description table of traffic volume dataset. Based on the statistical description table I had, I found some outliers and unreasonable value in the dataset. So the next step was to do data cleaning, and excluded unnecessary value and abnormal records in the dataset. After that, I did trendlines of all Pennsylvania stations specifically in hourly, daily, weekly and monthly time scale. This step helped me to see how traffic volume changes by time to time in different time scale. And the last step of my project is to construct an ARIMA mathematical model. I will select some representative stations with time series data to fit the model, and give future prediction based on the data we have.

4. Conclusion

4.1 Data import, summary and description

Raw code:

1. Use pandas.read_csv to import dataset
2. Use shape, columns and describe function to summarize and describe the dataset.

Result:
Pd.shape: (7140391, 38)
We have 7140391 records and 38 columns.
Pd.columns:
'''

```
Index(['date', 'day_of_data', 'day_of_week', 'direction_of_travel',
       'direction_of_travel_name', 'fips_state_code',
       'functional_classification', 'functional_classification_name',
       'lane_of_travel', 'month_of_data', 'record_type', 'restrictions',
       'station_id', 'traffic_volume_counted_after_0000_to_0100',
       'traffic_volume_counted_after_0100_to_0200',
       'traffic_volume_counted_after_0200_to_0300',
       'traffic_volume_counted_after_0300_to_0400',
       'traffic_volume_counted_after_0400_to_0500',
       'traffic_volume_counted_after_0500_to_0600',
       'traffic_volume_counted_after_0600_to_0700',
       'traffic_volume_counted_after_0700_to_0800',
       'traffic_volume_counted_after_0800_to_0900',
       'traffic_volume_counted_after_0900_to_1000',
       'traffic_volume_counted_after_1000_to_1100',
       'traffic_volume_counted_after_1100_to_1200',
       'traffic_volume_counted_after_1200_to_1300',
       'traffic_volume_counted_after_1300_to_1400',
       'traffic_volume_counted_after_1400_to_1500',
       'traffic_volume_counted_after_1500_to_1600',
```

'traffic_volume_counted_after_1600_to_1700',
'traffic_volume_counted_after_1700_to_1800',
'traffic_volume_counted_after_1800_to_1900',
'traffic_volume_counted_after_1900_to_2000',
'traffic_volume_counted_after_2000_to_2100',
'traffic_volume_counted_after_2100_to_2200',
'traffic_volume_counted_after_2200_to_2300',
'traffic_volume_counted_after_2300_to_2400', 'year_of_data'],
dtype='object')

'''

Pd.describe:

Table 1 Descriptive statistical table of traffic volume dataset

	fips_state_code	lane_of_travel	month_of_data	record_type	restrictions	traffic_volume_counted_after_0000_to_0100
mean	29.669665	1.29167226	6.52026325	3		114.587836
std	16.7061221	1.08041933	3.45523441	0		281.849233
min	1	0	1	3		-1
25%	13	1	4	3		13
50%	30	1	7	3		42
75%	44	2	10	3		126
max	56	9	12	3		99999

	traffic_volume_counted_after_0100_to_0200	traffic_volume_counted_after_0200_to_0300	traffic_volume_counted_after_0300_to_0400	traffic_volume_counted_after_0400_to_0500	traffic_volume_counted_after_0500_to_0600	traffic_volume_counted_after_0600_to_0700
mean	78.7455848	66.2250301	70.1613823	117.185123	245.4065	433.430074
std	220.287474	210.264237	224.248302	322.708513	572.330122	835.907796
min	-1	-1	-1	-1	-1	-1
25%	8	7	7	12	28	56
50%	27	21	23	38	86	170
75%	85	70	75	118	241	457
max	80741	90017	90012	70560	78159	90020

	traffic_volume_counted_after_0700_to_0800	traffic_volume_counted_after_0800_to_0900	traffic_volume_counted_after_0900_to_1000	traffic_volume_counted_after_1000_to_1100	traffic_volume_counted_after_1100_to_1200	traffic_volume_counted_after_1200_to_1300
mean	583.379894	577.497634	560.06943	581.108328	618.413301	650.134612
std	998.494105	959.42167	891.730834	897.057229	937.17169	1001.83585
min	-1	-1	-1	-1	-1	-3061
25%	90	107	123	137	149	159
50%	264	285	303	332	362	386
75%	657	657	645	681	725	761
max	90187	99999	95300	99999	90200	99999

	traffic_volume_counted_after_1300_to_1400	traffic_volume_counted_after_1400_to_1500	traffic_volume_counted_after_1500_to_1600	traffic_volume_counted_after_1600_to_1700	traffic_volume_counted_after_1700_to_1800	traffic_volume_counted_after_1800_to_1900
mean	663.507518	700.982507	749.715969	777.043711	756.553646	617.332177

std	1024.51475	1092.23619	1143.31806	1173.93294	1172.11589	1061.54491
min	-1	-1	-1	-1	-1	-1
25%	162	170	183	186	174	131
50%	391	409	438	452	432	337
75%	777	822	891	934	907	722
max	94200	99999	99999	99999	99999	99999

	traffic_volume_counted_after_1900_to_2000	traffic_volume_counted_after_2000_to_2100	traffic_volume_counted_after_2100_to_2200	traffic_volume_counted_after_2200_to_2300	traffic_volume_counted_after_2300_to_2400	year_of_data
mean	479.375636	390.642581	327.47465	253.444707	179.829806	15
std	920.371139	829.027125	798.914633	728.407411	690.171309	0
min	-1	-1	-1	-1	-1	15
25%	95	72	54	36	22	15
50%	252	198	155	109	70	15
75%	551	447	367	278	193	15
max	99999	99999	99999	99999	99999	15

From table above, we can see that for each hourly traffic volume period column, we need to do some data cleaning work before analysis. Traffic volume that refer to -1 is ridiculous. In addition, hourly traffic volume with 99999 vehicles also seems impossible. We need to kick off this part from the total dataset.

4.2 Data Cleaning

According to transportation department of California, the maximum ideal lane capacity for a multilane highway segment is 2200 vehicles per hour[3] . In our dataset, the maximum lane in a road segment is 9 lanes. So I assume that the maximum capacity of a road segment is 19800 vehicles per hour. Based on this conclusion, I filter impossible traffic volume per hour in our dataset. Before filtering, there are 7140391 rows in my dataset. And there are 1997 impossible records have been eliminated.

4.3 Data Visualization

The final part is to describe the average trend of traffic flow in Pennsylvania as an example, to prove traffic flow is whether a periodic data or not. In this part, I will show my process in raw code below:
Raw code:

1. Select Pennsylvania data by filtering fips_state_code equals to 42.
2. Categorize different traffic volume with different functional system
3. Group by data in different time scale respectively (hourly, weekly, daily, monthly), and then calculate their average traffic volume in selected functional system
4. Using Matplotlib to draw line charts for each type of functional system in different time scale
5. Verify whether the traffic volume is periodic.

By using describe function in pandas module, we have a summary about what kinds of function system exist in Pennsylvania.
Functional system:
'Rural: Principal Arterial - Other'
'Urban: Principal Arterial - Interstate' 'Rural: Minor Arterial'
'Urban: Principal Arterial - Other Freeways or Expressways'
'Urban: Principal Arterial - Other' 'Urban: Minor Arterial'
'Rural: Principal Arterial - Interstate' 'Rural: Major Collector'
'Rural: Minor Collector' 'Urban: Collector' 'Rural: Local System'
'Urban: Local System'

From the descriptive information above, we choose two types of roads in urban and rural area respectively to draw the time series line charts. For both urban and rural area, I choose ‘Principal Arterial – Interstate’ and ‘Minor Arterial’ as representative examples.

After operation in Python, we got the four average time series line chart in Pennsylvania.

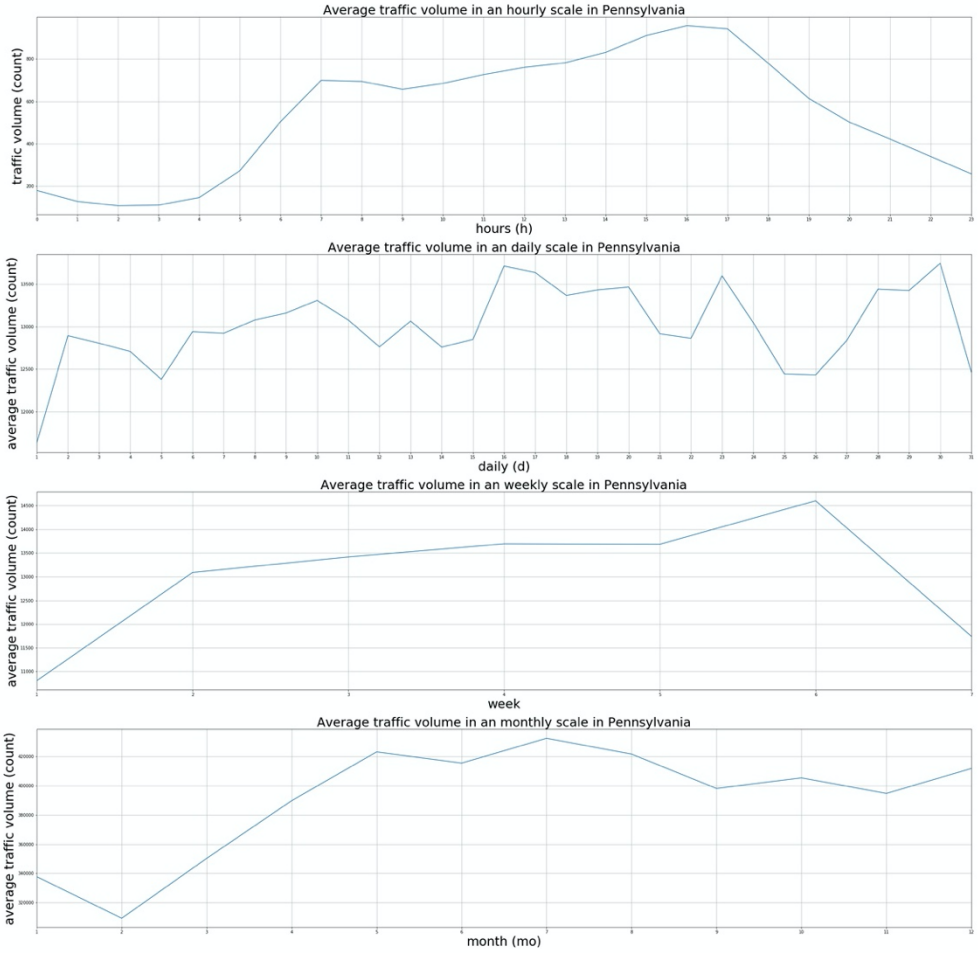

Fig 1. Average traffic volume in different time scale in Pennsylvania, functional system: Urban Principle Arterial: Interstate

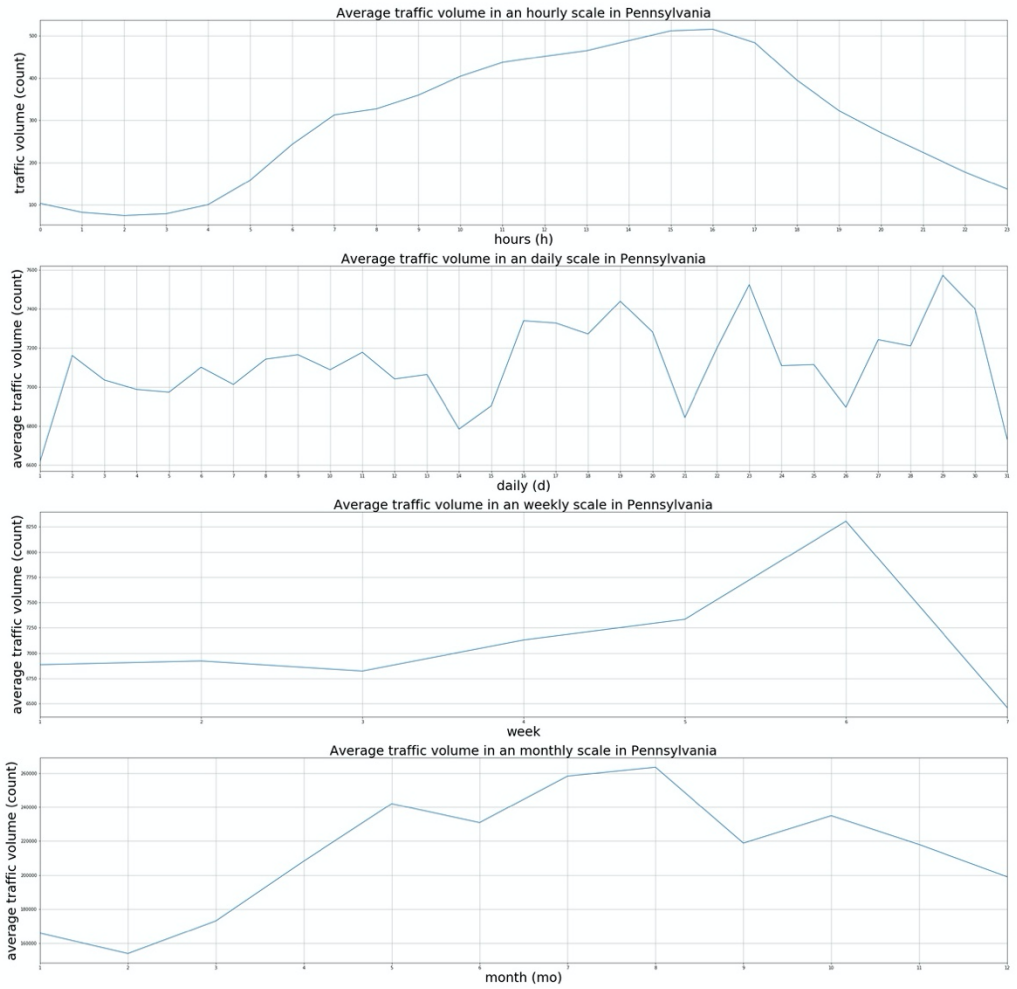

Fig 2. Average traffic volume in different time scale in Pennsylvania, functional system: Rural Principle Arterial: Interstate

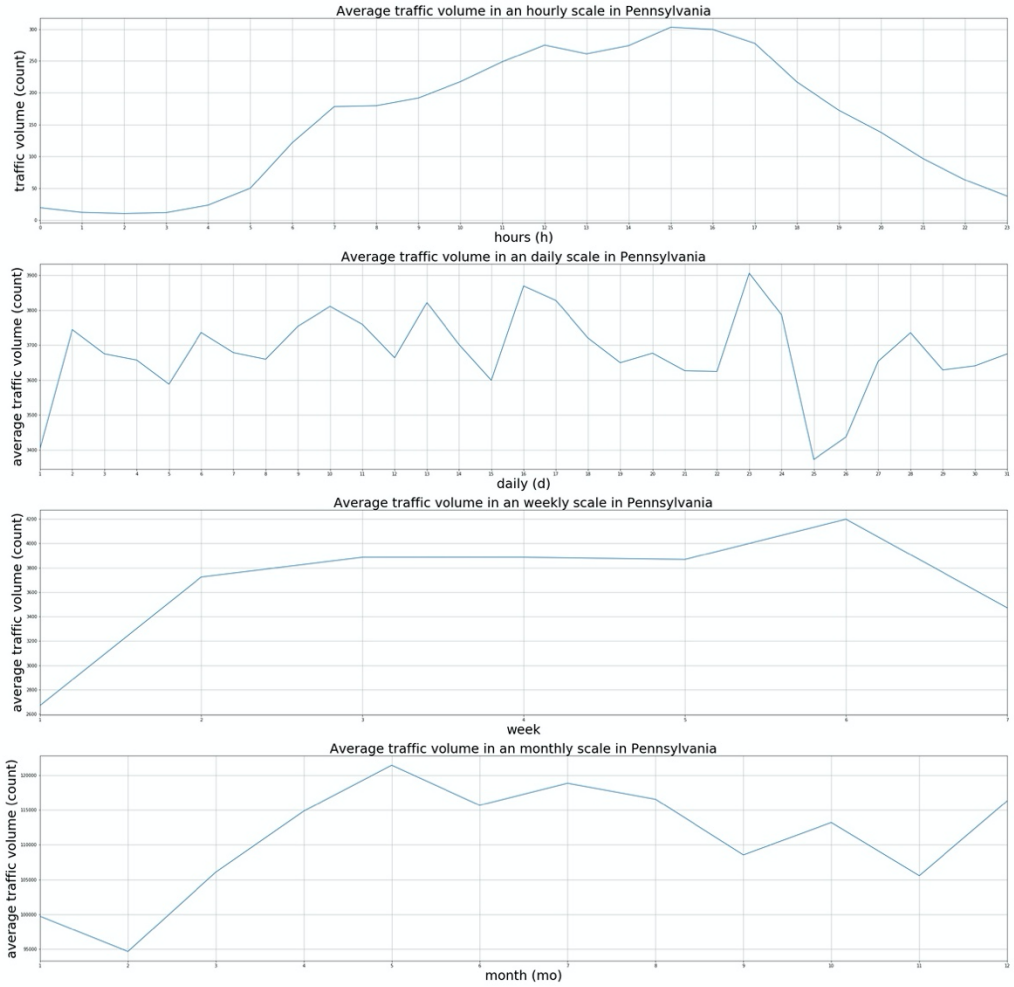

Fig 3. Average traffic volume in different time scale in Pennsylvania, functional system: Urban Minor Arterial

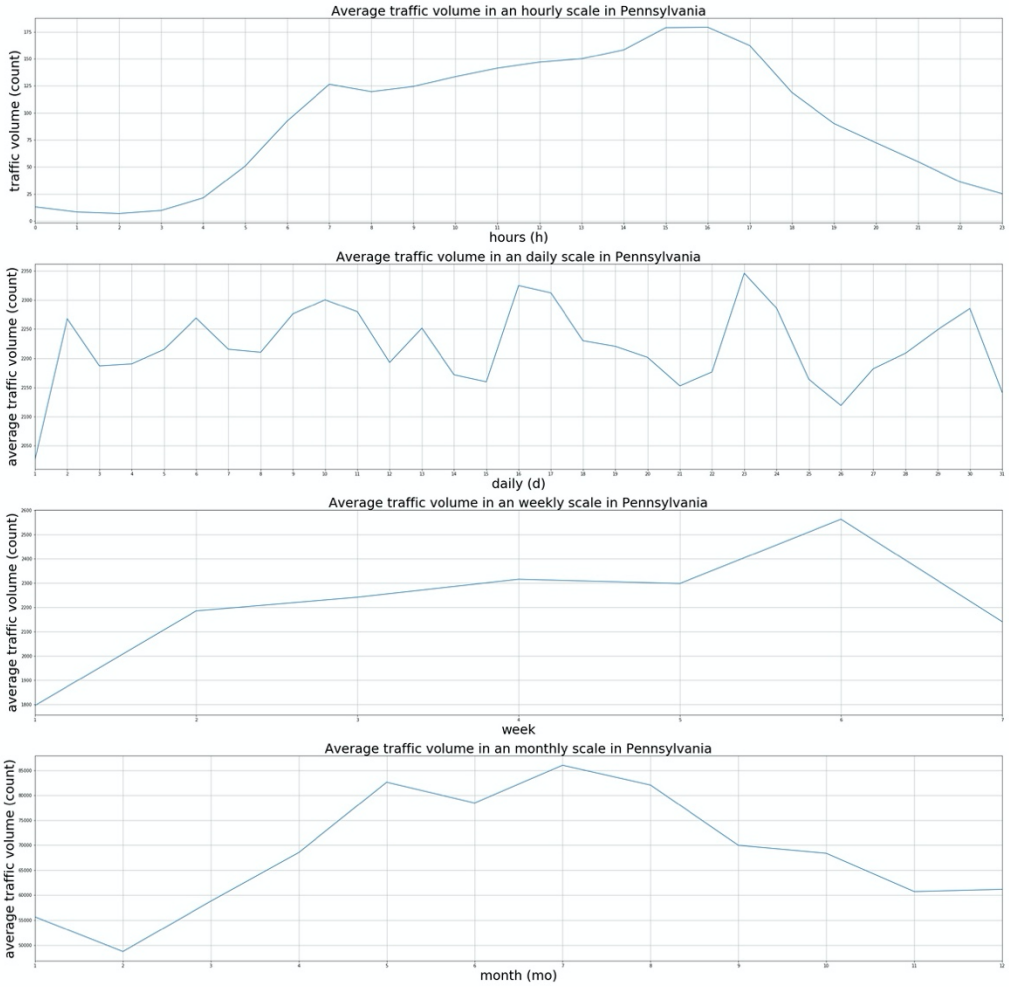

Fig 4. Average traffic volume in different time scale in Pennsylvania, functional system: Rural Minor Arterial

From the figures above, we can have a straight forward information that time series is periodic in Pennsylvania no matter what functional system it is. Why it is important to tell that our time series is periodic is that ARIMA model requires a periodic data to do prediction. So it is important to do data analysis to prove our intuiotion first. Besides periodic verification, we can also find other interesting information from the figures above. Firstly, the high peak of traffic volume happens from 7am – 8pm for urban roads such as interstate and minor arterial hourly. However, in rural area, it only happens from 2pm-8pm. The morning peak time is relatively mild than urban area. When it comes to weekly, we can find that Saturday is the busiest to for both rural and urban area weekly. And the busiest month will happen from April to June.

4.4 Model Prediction

Based on periodic data we have that proved by exploratory data analysis, I plan to use two different statistical models to predict the future trend of traffic, the first one is ARIMA model, which based on an moving trend time series and residual time series. In python we have package 'pmdarima' to complete this math model automatically. Another statistical model I used is neural network. I will discussion some hyperparameter such neurons and epoch times of neural network to find the model with best performance. In python, we have package 'tensorflow' to complete this model.

Before setting up statistical model, we need to use some data preprocessing step to convert our dataframe to a standard time series. In my code, I use 'datetime' and 'pandas' module to write some function, to convert our daily records into time series hourly records.

For time series, we need to split our 2015 whole year dataset into different train set and test set. For splitting part, I provide three training and test dataset to show the performance of models. Here is the details of each training and test dataset:
Training dataset 1: January 1st 00:00:00 – March 31st 23:59:59
Test dataset 1: April 1st 00:00:00 – April 30th 23:59:59
Training dataset 2: May 1st 00:00:00 – July 31st 23:59:59
Test dataset 2: Aug 1st 00:00:00 – Aug 31st 23:59:59
Training dataset 3: Sept 1st 00:00:00 – Nov 30th 23:59:59
Test dataset 3: Dec 1st 00:00:00 – Dec 31th 23:59:59

4.4.1 ARIMA Model

In ARIMA model, we use 'auto arima' function to automatically find the parameters for ARIMA model. The essential parameters we need are p, d and q. P means the number of autoregressive terms, d means the number of nonseasonal differences needed for stationarity, and q is the number of lagged forecast errors in the prediction equation.

Before we set up ARIMA model, another thing we need to verify is the stationarity of dataset. For this issue, we can use ADF test to verify the stationarity. If ADF test failed, first difference is applied to time series data in order to eliminate instationarity. The full name of ADF test is Augmented Dickey-Fuller test. The testing object of ADF test is unit root. If unit root exists in an ARIMA model, that means the error on residual series will not decrease as sample size increase. That means the effect of residual in the model is permanent. In our time series data, first ADF test failed so we choose to use first difference time series data to set up ARIMA model.

In my auto ARIMA model, I set up parameter for d is 1 because we need d=1 to keep time series stationary. For p and q, I search the best parameters from 1 to 5 to optimize the model based on their performance on MLE (Maximum likelihood estimation).

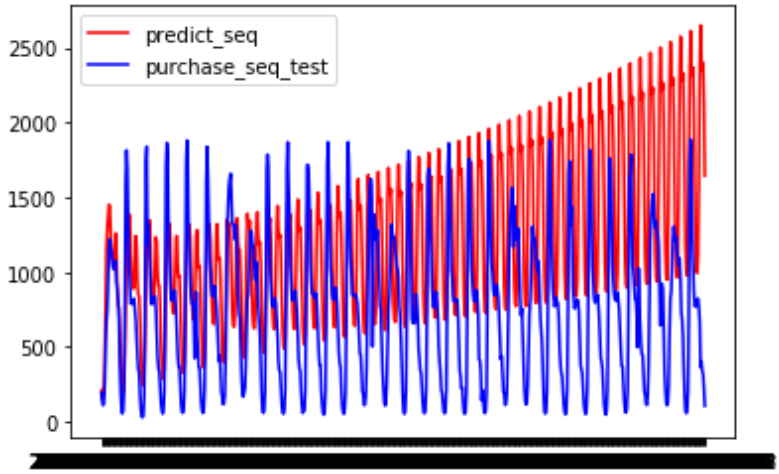

Fig 5. ARIMA model prediction for test 1 dataset

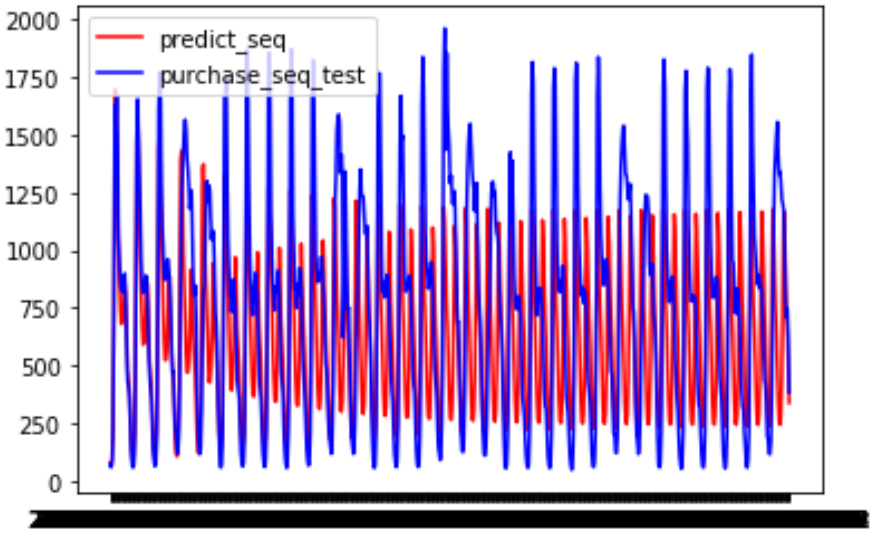

Fig 5. ARIMA model prediction for test 2 dataset

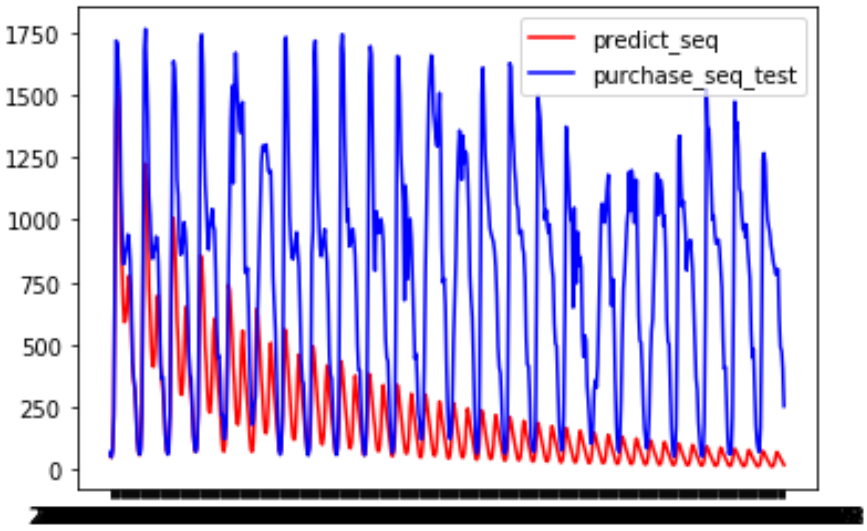

Fig 6. ARIMA model prediction for test 3 dataset

From the graph above, we can see that model 2 which uses training dataset 2 has the best performance among three models. However, ARIMA models for test data 1 and test data 3 don't have a good performance on predict future trend and show a relatively unrealistic trend.

4.4.2 neural network

Before we set up neural network models for our time series data, we still need to do some data preprocessing step to convert our time series data into a mapping data from previous time series to future prediction. In here, I constructed a 'split_sequence()' function to realize this function. In split_sequence function, we have three parameters, sequences, n_steps_in and n_steps_out. For parameter 'sequence', we can just enter our training dataset. For parameter 'n_steps_in', we need to modify how many time steps to be trained by our model. For parameter 'n_steps_out', we need to set up how many time steps we need to predict by our model. And there are two outputs in our function. The first one, 'X', should be our training dataset, we use multi-time-step time series dataset to train our neural network, and validate loss function on another output, 'y', to modify the parameters of neural network and reduce loss to improve model.

In my neural network, I used one 100 full-connected layer network structure with ReLU function as activation function to train out model. And the result of model look like below.

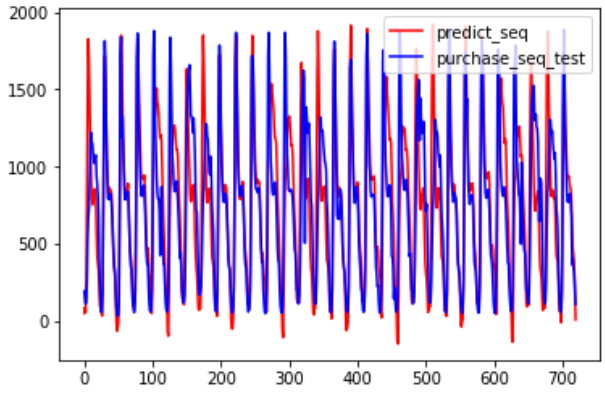

Fig 7. Neural Network Prediction with test 1 dataset

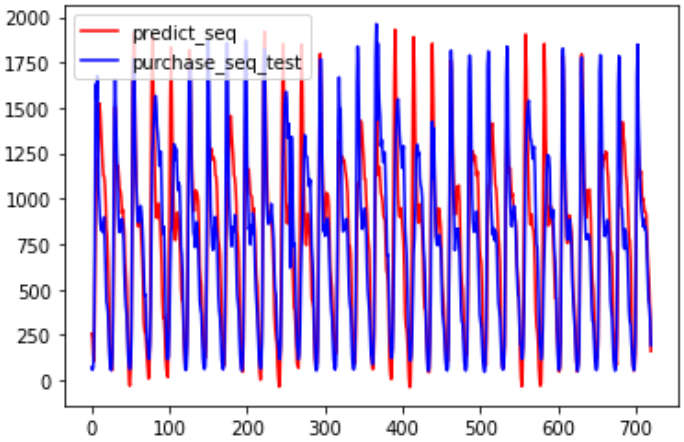

Fig 8. Neural Network Prediction with test 2 dataset

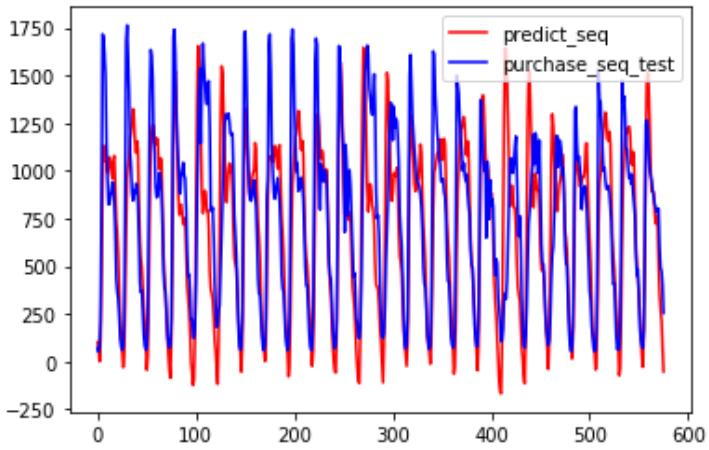

Fig 9. Neural Network Prediction with test 3 dataset

From figures above, we can see a better performance with neural networks than ARIMA model. And the trends and peak values are in relatively good fit than ARIMA model.

4.4.3 Prediction Evaluation

In this part, I compared RMSE with ARIMA model and neural network in order to show which statistical model has a better performance than another. I used RMSE to measure the accuracy on prediction on test data and compare both model in a table.

	test 1 data RMSE	test 2 data RMSE	test 3 data RMSE
ARIMA	744.13	428.34	702.81
Neural Network	332.15	326.52	318.05

From table above, we can see neural network have a higher accuracy than ARIMA. So we can use data from neural network to predict the traffic flow in our dataset.

5 Things I learned

In this project, I've learned many packages in Python. I use 'numpy' and 'pandas' to do calculation and dataset modification. I also learned how to use 'matplotlib' module to do data visualization by comparing the prediction value with true value. Besides that, I use 'pmdarima' package to set up ARIMA model and I use 'tensorflow' package to set up neural network model.

I learned how to modify the parameter of ARIMA model by using statistical test such as ADF test and Ljung-Box test. These statistical methods helps me to verify the accuracy and ARIMA model and improve my parameters and loss function. I also learned how to construct a standard multilayer perceptron by using tensorflow package, how to set up sequential model, how to add layers and how to modify hyperparameters of a neural network. I also reviewed some knowledge of stochastic gradient descent.

I have a hard time on selecting parameters with ARIMA model because my residual time series is not white noise. If I use manual method to select p, d and q for ARMA model, my prediction performance is very bad and not satisfy as expected. So I choose auto.arima function instead to solve this problem.

References

- 1. *Asteriou, Dimitros; Hall, Stephen G. (2011). "ARIMA Models and the Box–Jenkins Methodology". Applied Econometrics (Second ed.). Palgrave MacMillan. pp. 265–286.* ISBN 978-0-230-27182-1.
- 2. https://zh.wikipedia.org/wiki/ARMA模型
- 3. https://ccag.ca.gov/wp-content/uploads/2014/07/cmp_2005_Appendix_B.pdf