National College of Ireland

Configuration Manual

MSc Research Project
Short term traffic flow prediction

Hongyi Yan
Student ID: 19207433

School of Computing
National College of Ireland

Supervisor: Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …Hongyi Yan……

Student ID: …19207433……

Programme: …short traffic flow prediction…… **Year:** …2021……..

Module: …machine learning……

Supervisor: …Jorge Basilio……

Submission Due Date: …08.16……

Project Title: …Comparison of machine learning in Intelligence Traffic System…

Word Count: …… **Page Count**…10……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: …Hongyi Yan……

Date: …08.16……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	√
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	√
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	√

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Hongyi Yan
19207433

1 Introduction

In fact, the operation background of the whole project comes from the intelligent traffic system. After studying the development of ITS, I found that traffic flow prediction is a very basic but important technology. And this technology has been updated and changed with the development of computer hardware. In the early stage, the principle of statistics was applied to estimate the peak of traffic, and then artificial regulation was carried out. Later, with the emergence of new technologies such as machine learning and deep learning, the accuracy of prediction also increased (Zhao et al; 2012). It can be said that this subject is energetic, and it is novel when new technologies appear.

The data is from Queensland data website, which is a very original traffic data. The number of cars passing and leaving in some sections over a period of time is recorded. The good thing is that it is counted every 15 minutes, which is very in line with the research standard (MA C et al; 2020).

As for the introduction of the working environment of this study, python programming language is selected, and anaconda is used as the working environment. Anaconda needs to be updated to a newer version because the old version does not provide corresponding services when using some LSTM packages later. In fact, after testing, Anaconda version 2.0 or above needs to be installed. In this way, some packages can be deployed normally and the code can run normally. In this step, the author has to reinstall the environment to realize the application of neural network. However, if only the seasonal demonstration experiment is realized, it can run without too high version.

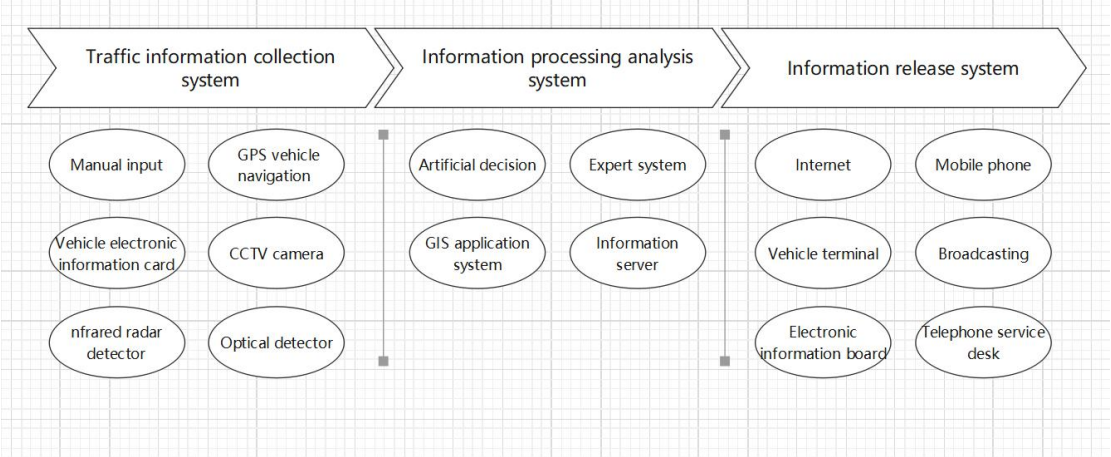


Figure 1 The whole ITS system

2 Preprocessing

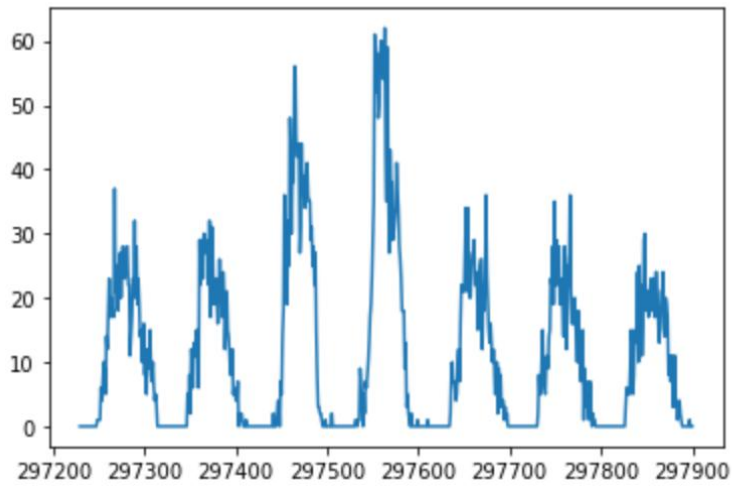
This step is in the same file with ARIMA. The most basic packages numpy and pandas are imported for data preprocessing. First read the data into our working environment, and then look at the table.

	SITE	Direction	DAY	TIME	C1	C2	C3	C4-13	Ped	Bike
0	2006CC1	Entry	9/05/2006	6:00:00	0	NaN	0.0	NaN	NaN	NaN
1	2006CC1	Entry	9/05/2006	6:15:00	0	NaN	0.0	NaN	NaN	NaN
2	2006CC1	Entry	9/05/2006	6:30:00	1	NaN	0.0	NaN	NaN	NaN
3	2006CC1	Entry	9/05/2006	6:45:00	1	NaN	0.0	NaN	NaN	NaN
4	2006CC1	Entry	9/05/2006	7:00:00	0	NaN	0.0	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
95	2006CC1	Exit	23/05/2006	9:45:00	2	NaN	0.0	NaN	NaN	NaN
96	2006CC2	Entry	9/05/2006	6:00:00	0	NaN	0.0	NaN	NaN	NaN
97	2006CC2	Entry	9/05/2006	6:15:00	0	NaN	0.0	NaN	NaN	NaN
98	2006CC2	Entry	9/05/2006	6:30:00	2	NaN	0.0	NaN	NaN	NaN
99	2006CC2	Entry	9/05/2006	6:45:00	1	NaN	0.0	NaN	NaN	NaN

Using the df.drop() function to delete all the lines which direction is leaving. We choose the number of vehicles entering the section as the traffic flow. In addition, because the value of C1 is the largest, that is, the change is the most obvious, therefore C1 column is selected as the time series for analysis.

The next step is to find special values. After visualizing the data as a line chart, it is found that the data fluctuates very irregularly for two days. The code and implementation are as follows.

```
plt.plot(df1.timeid, df1.C1)
#plt.xlim(0,2)
plt.figure(dpi=300,figsize=(24,8))
plt.show()
```



So use drop () to delete the data of March 14 and March 15. Then use the DF. ToCSV () function to export the data which has been processed. This will help you directly reference other models when you build them later.

The following is some exploratory analysis, which will not be explained too much in this paper, because it is analyzed in detail in the report.

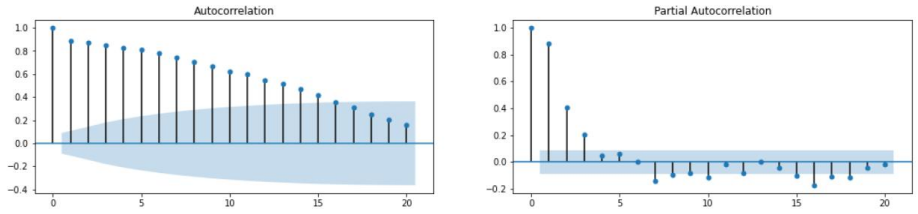
3 ARIMA

This step only needs to be carried out according to the submitted code. First, import the relevant statistical package. The code is as follows:

import statsmodels.api as sm;

Present the relevant ACF and PACF diagrams of the time series and judge the relevant AR (P) and MA (q) (D. cvetek et al; 2020)。

```
fig = plt.figure(figsize=(18,8))
ax1=fig.add_subplot(221)
fig = sm.graphics.tsa.plot_acf(data,lags=20,ax=ax1)
ax2 = fig.add_subplot(222)
fig = sm.graphics.tsa.plot_pacf(data,lags=20,ax=ax2)
plt.show()
```



Next, check the stability. Finally, my subjective judgment is diff = 1, P = [2,3], q = [1,2].

Here, in order to find the optimal solution of the model, several parameters are modeled separately. The ARIMA () function in the statsmodels package is used. Input the parameters judged above, and put it in the model. Summary() function is used to summarize the performance of the model. In the optimization stage, I choose AIC and BIC as the

optimization criteria. The value of the model should be as small as possible. Finally, after modeling several possible parameter models, select the best group. Therefore, the final model is as follows:

$$ARIMA(2, 1, 1)$$

Dep. Variable:	y	**No. Observations:**	480
Model:	SARIMAX(2, 1, 1)	**Log Likelihood**	368.061
Date:	Tue, 27 Jul 2021	**AIC**	-728.122
Time:	22:04:36	**BIC**	-711.435
Sample:	0	**HQIC**	-721.562
	- 480		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.2271	0.148	-1.535	0.125	-0.517	0.063
ar.L2	-0.0857	0.080	-1.070	0.284	-0.243	0.071
ma.L1	-0.3329	0.149	-2.242	0.025	-0.624	-0.042
sigma2	0.0126	0.001	23.793	0.000	0.012	0.014

Ljung-Box (Q):	64.74	**Jarque-Bera (JB):**	195.61
Prob(Q):	0.01	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.11	**Skew:**	0.62
Prob(H) (two-sided):	0.53	**Kurtosis:**	5.88

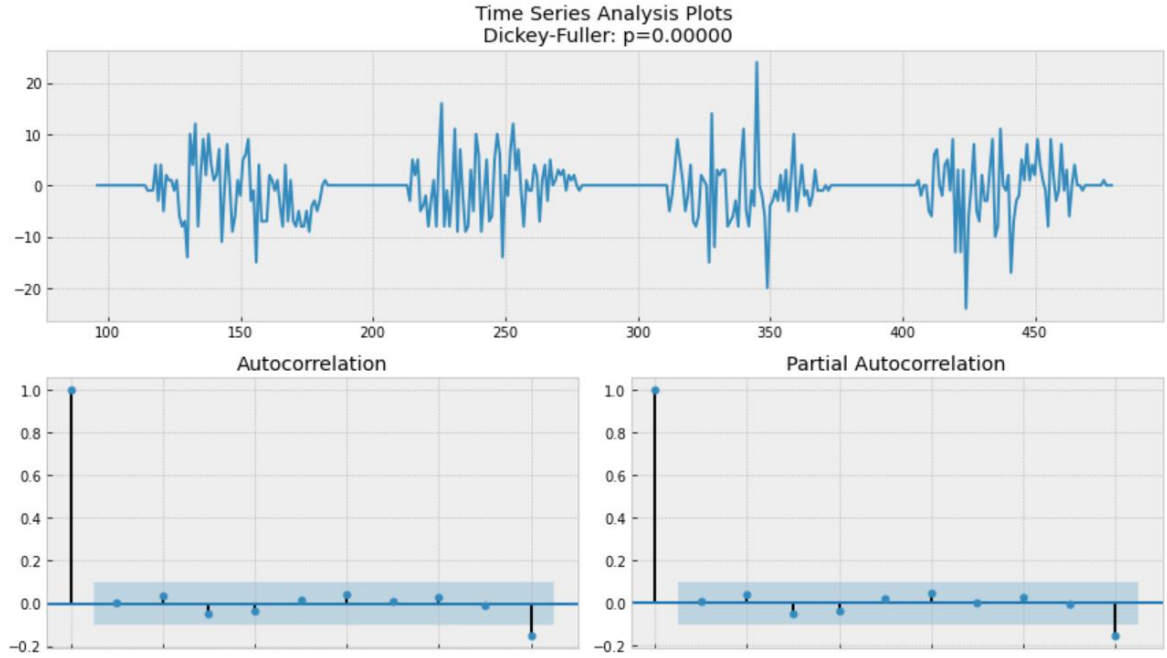
4 SARIMA

Here, first define the drawing function. The code is as follows:

```
def tsplot(y, lags=None, figsize=(12, 7), style='bmh'):
    """
        Plot time series, its ACF and PACF, calculate Dickey-Fuller test

        y - timeseries
        lags - how many lags to include in ACF, PACF calculation
    """

    if not isinstance(y, pd.Series):
        y = pd.Series(y)

    with plt.style.context(style):
        fig = plt.figure(figsize=figsize)
        layout = (2, 2)
        ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
        acf_ax = plt.subplot2grid(layout, (1, 0))
        pacf_ax = plt.subplot2grid(layout, (1, 1))

        y.plot(ax=ts_ax)
        p_value = sm.tsa.stattools.adfuller(y)[1]
        ts_ax.set_title('Time Series Analysis Plots\n Dickey-Fuller: p={0:.5f}'.format(p_value))
        smt.graphics.plot_acf(y, lags=lags, ax=acf_ax)
        smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax)
        plt.tight_layout()
```

This can bring all the graphs together in the subsequent ACF and PACF checks. When using, you only need to run in the format of tsplot (x, y). X is the time series to be modeled, and Y is the lags quantity which you need to view. This is very important, because in this subject, the seasonal value is too large, especially it is important to check the changes after one or two seasons. Therefore, in my experiment, first define the lags as 100, and then observe the change. As for my seasonal choice, it is not entirely determined by observing these two pictures. The seasonality is 96. This is because I regard a day as a season, and then a day is 24 hours, but I count it four times per hour. Therefore, a total of 96 times of data were counted in one day (U. ashwini et al; 2021).

```
# The seasonal difference
ts_diff = ts.C1 - ts.C1.shift(96)
tsplot(ts_diff[96:], lags=10)
```



The above figure, together with the code part, is the observation after incorporating the seasonality 96 as the difference value. Firstly, it can be seen that the stability has improved. And then by observing the long-time lag, I found that the seasonality has disappeared. Therefore, the series data with a seasonality of 96 can be modeled. The model parameter judgment process is introduced in the report, so only the code implementation is introduced here.

Next, adjust and build the model by calling the sarimax () function. The specific codes are as follows (Kang et al; 2021)：

```
sm.tsa.statespace.SARIMAX(data, order=(1, 1, 1),
                          seasonal_order=(1, 1, 1, 96)).fit(disp = -1)
```

Turning to the mathematical formula, we chose a new model.

The process of adjusting parameters is the same as that of ARIMA model. Finally, model. Summary () is used to summarize the model.

5 LSTM

Use the following code to allocate the data value and target value.look_Back is the number of dimensions characterized by several rows of data.

```
def creat_dataset(dataset,look_back):
    data_x = []
    data_y = []
    for i in range(len(dataset)-look_back):
        data_x.append(dataset[i:i+look_back])
        data_y.append(dataset[i+look_back])
    return np.asarray(data_x), np.asarray(data_y)
```

Finally, the value of Look_back is actually 2 as the feature dimension. Then, len () * 0.7 is used to obtain the division of training set and test set.

As for the data previously converted to ndarray, it needs to be converted to the tensor data which is required in pytorch. Use torch.from_ Numpy() function to convert it. This step is necessary, and non conversion will lead to data mismatch (Qu et al; 2020)。

The next step is the establishment of the model. First call the class RNN which is designed in the torch package, and then initialize it.

```
super(RNN,self).__init__() #Inheritance
```

Next, set the parameters of LSTM, define the dimension of the input eigenvalue as 2, and the two lstm are connected in series. Here I set six hidden layers. Of course, if your laptop is efficient, the value of the hidden layer can also be slightly adjusted.

Finally, there is a linear fitting layer. It receives a data dimension of 6 and outputs a dimension of 1.

Of course, the connection of the whole model needs dimension transformation to be connected together. Define a forward () to handle such problems, as follows.

```
def forward(self,x):
    x1,_ = self.lstm(x)
    a,b,c = x1.shape
    #Because the linear layer inputs two-dimensional data, the three-dimensional data x1
    #output by LSTM should be adjusted to two-dimensional data here,
    #and the final feature dimension cannot be changed
    out = self.out(x1.view(-1,c))
    #Because it is a cyclic neural network,
    #the two-dimensional out should be adjusted into three-dimensional data at the last
    #time, which can be recycled next time
    out1 = out.view(a,b,-1)
    return out1
```

The torch package optim.Adam () function is used to optimize the parameters of the model. The loss function is used as the optimization index here, and I choose a learning rate of 0.02.

The final training code and training process are as follows:

```
##### training
for i in range(1000):
    var_x = Variable(x_train).type(torch.FloatTensor)
    var_y = Variable(y_train).type(torch.FloatTensor)
    out = lstm(var_x)
    loss = loss_func(out, var_y)
    MAE  = mean_absolute_error(var_y.view(-1).detach().numpy(), out.view(-1).detach().numpy())
    RMSE = mean_squared_error(var_y.view(-1).detach().numpy(), out.view(-1).detach().numpy())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if (i + 1) % 100 == 0:
        print('Epoch:{}, Loss:{:.5f}, MAE:{:.3f}, RMSE:{:.3f}'.format(i + 1, loss.item(), MAE, RMSE))
torch.save(lstm.state_dict(), 'time_seq_parameter.pkl')
```

```
Epoch:100, Loss:0.01052, MAE:0.067, RMSE:0.011
Epoch:200, Loss:0.00968, MAE:0.065, RMSE:0.010
Epoch:300, Loss:0.00925, MAE:0.063, RMSE:0.009
Epoch:400, Loss:0.00896, MAE:0.063, RMSE:0.009
Epoch:500, Loss:0.00938, MAE:0.066, RMSE:0.009
Epoch:600, Loss:0.00837, MAE:0.061, RMSE:0.008
Epoch:700, Loss:0.00790, MAE:0.059, RMSE:0.008
Epoch:800, Loss:0.00765, MAE:0.059, RMSE:0.008
Epoch:900, Loss:0.01035, MAE:0.074, RMSE:0.010
Epoch:1000, Loss:0.00712, MAE:0.057, RMSE:0.007
```

RMSE: It is the square root of the ratio of the square of the difference between the predicted value and the real value to the number of predictions. The full name is Root Mean Square Error. In this research, it is used to measure the difference between the predicted value and the real value of the final model, which can be regarded as the prediction accuracy. The formula is as follows:

$$\mathrm{RMSE}(X,h)=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})^2}$$

Here, RMSE is used as the same measure of the three models. I designed it for my own time series model. By calculating the predicted value and the real value, the error of each time is calculated, and the prediction accuracy of the model is obtained. The code is as follows. Try_model.fittedvalues represents the predicted value and data represents the value of the original data.

```
def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())
rmse(try_model.fittedvalues, data)
```

Finally, compare the RMSE values of each model and draw a conclusion.

References

D. Cvetek, M. Muštra, N. Jelušić and B. Abramović, "Traffic Flow Forecasting at Micro-Locations in Urban Network using Bluetooth Detector," *2020 International Symposium ELMAR*, Zadar, Croatia, 2020, pp. 57-60, doi: 10.1109/ELMAR49956.2020.9219023.

H. -J. Kang, C. Kallas, M. -H. Park and J. Kim, "A Scalable Learning Model for Multi-seasonal Time Series Forecasting," *2021 International Conference on Electronics, Information, and Communication (ICEIC),* Jeju, Korea (South), 2021, pp. 1-4, doi: 10.1109/ICEIC51217.2021.9369826.

H. Qu, J. Li and Y. Zhang, "Long Short-term Memory Network Prediction Model Based on Fuzzy Time Series," *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*, Dalian, China, 2020, pp. 417-421, doi: 10.1109/ICAIIS49377.2020.9194902.

Ma C, Tan L, Xu X. Short-term Traffic Flow Prediction Based on Genetic Artificial Neural Network and Exponential Smoothing. *Promet - Traffic&Transportation*. 2020;32(6):747-60. DOI: 10.7307/ptt.v32i6.3360

U. Ashwini, K. Kalaivani, K. Ulagapriya and A. Saritha, "Time Series Analysis based Tamilnadu Monsoon Rainfall Prediction using Seasonal ARIMA," *2021 6th International Conference on Inventive Computation Technologies (ICICT),* Coimbatore, India, 2021, pp. 1293-1297, doi: 10.1109/ICICT50816.2021.9358615.

Zhao, Y. Dai and Z. Zhang, "Computational Intelligence in Urban Traffic Signal Control: A Survey," *in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 42, no. 4, pp. 485-494, July 2012, doi: 10.1109/TSMCC.2011.2161577.