Introduction to Data Science

NIST Project :- Cleaning 2

Introduction

As per the details of the project, We have employed the three techniques for calculating the predicted flow values. These three techniques are :-

1. Using Linear regression on nearby rows to predict flow

2. Using weighted sum of flow values of nearby timestamps-rows to predict flow

3. Using original probability density distribution

IMPORTANT NOTES :-

a. Since the context of the current folder was ambiguous so we are considering that the output txt files will be created in the same folder as of the bash script. This is the home directory for the project submission.

CONTENT INDEX :-

Results and Observations :-

We can observe below a chunk of data from flow file “1160”. The first image shows the actual flow values and the second image shows the predicted flow values. This file had a lot of erroneous values in the form of very high flow like 255. These flows had a very small probability density assigned to them and we utilized the three methods of Predicted flow calculation and Confidence Calculation to predict their values.

1. As can be seen below 255 is reduced to 0. This is a significant change where probability density helps us in removing the outlier values and predicting a flow which is comparable to the flow in other lanes.
2. The predicted flow values for non-outlier flow values can be also be observed. For e.g in row index-31, The flow values of (8, 4, 3) are normalized to (6.61, 5.27, 9.57). This depict that our approach successfully predicts values based on the three approaches and doesn’t deviate from the actual values by a huge margin, affirming the correctness of our results.

ACTUAL FLOW VALUES :-

25	5	4	255
26	10	9	255
27	3	6	255
28	8	8	255
29	11	6	4
30	5	5	4
31	8	4	3
32	7	7	3
33	14	9	4
34	5	5	4

PREDICTED FLOW VALUES :-

25	6.66440313589	6.47225821148	0.0
26	10.9097058168	9.31492234573	0.0
27	7.49987020866	6.90298830794	0.0
28	8.47339492431	8.13409507666	0.0
29	8.20889973182	5.89134775429	8.67535693127
30	6.67870505096	4.86801819034	9.83382431109
31	6.61797433425	5.27591037148	9.57826646578
32	7.01754393353	5.94311214706	4.26441279902
33	7.17766170826	7.18732216322	9.32647164509
34	4.71056706444	4.87248890717	8.2238608146

Performance Evaluation

In a similar manner, We can observe below a chunk of data from flow file “3532” zone. The first image shows the actual flow values and the second image shows the predicted flow values. We observed that the heaviest dataset i.e the 3532 took around 30 mins to complete on our local instance. In our opinion, This is pretty remarkable considering the fact that the combined size of the dataset exceeds 2 GB and has nearly 20,000,000 rows. We are maintaining the precision of the results upto 8 digits which increase the size of intermediate results to over 5GB. This is the main reason for the showcased time. If we are allowed to drop this precision to a lower value, then we would be able to considerably improve the performance execution of the methods. This will also lead to a drop in fize size as you may notice that our current zip file size exceeds 450MB.

ACTUAL AND PREDICTED VALUES ON HEAVIEST DATASET (ZONE-3532)

ACTUAL FLOW VALUES :-

20	16	25	30	24	30	31
21	18	25	24	22	26	36
22	14	20	22	32	16	15
23	16	27	28	23	25	30
24	18	24	29	17	23	15
25	12	23	24	22	21	20
26	12	18	21	20	20	29
27	17	21	29	23	23	22
28	15	18	23	24	19	13
29	12	18	26	20	23	16
30	16	17	21	24	43	18
31	15	18	23	26	18	15
32	11	23	27	20	17	25
33	17	23	24	25	18	24
34	15	28	21	28	31	23
35	17	18	22	24	24	19

PREDICTED FLOW VALUES :-

20	16.46348932317724	22.38268702281132	24.237069955378136	25.682694177880226	21.955479452054792	24.537333379496946
21	17.4321594198711	24.21140251564025	26.88952481099853	23.869232176104283	22.959213214030086	23.09090909090909
22	15.344821514949071	20.670000034342856	23.06125719776507	21.491940201346107	19.93917147515571	16.59599001104359
23	16.361125836190382	22.93203883495145	24.626262626262626	24.150510867928013	25.27025333390291	20.462962962962962
24	15.050773048052974	21.86016542122203	22.57543103448276	21.61114865254792	22.013251875910846	20.337958959598886
25	14.654706420504427	21.51265118834259	23.814896752270784	21.847120767299952	21.641783020679174	20.969850910264725
26	14.43222437067912	19.23116233511137	22.611966602982267	20.72665522515976	20.551946268772443	17.5
27	15.276537727208176	21.00746592927707	23.51556253358949	23.475444846204454	22.096652536566143	21.919546353733416
28	13.66695649530534	18.660180460127442	21.075450512936264	22.112954270698037	19.283743032605603	16.52863091985175
29	12.556547624116275	18.786860709660584	24.708427067016217	21.05151421304365	24.717024867626623	17.897766922103273
30	14.805925443096731	21.04676763148202	24.312945166702686	22.60116161677526	18.764391061073546	20.024634560754407
31	13.931904375107253	19.130774787115925	22.19385429984921	22.266060323215882	19.04005857564009	17.395650354217732
32	13.798276107859984	20.756465724298533	22.9245341389461 7	21.24656880537124	21.4040458723094	21.212970861853247
33	15.934401131203805	21.914999843554405	23.627529283831176	23.916352172518128	20.35981678386709	21.79359620646125
34	15.713281495703493	21.355946044575	24.430590073831862	24.06002412355381	19.169399269815568	22.553438506463753
35	15.291155404926055	19.972129588871802	22.119275736694053	23.218924884171557	22.032913058923697	21.277777777777786

SPLITTING THE DATA “MAKING CODE GENERIC”

The main challenge for us in this Lab was to make the code Generic. It was very important that the code works for any number of columns and is able to split it properly. Prior to splitting and combining the data we are reading the number of columns in file. Therefore,

THE CODE WORKS FOR **N** NUMBER OF COLUMNS IN ANY CONFIGURATION

You can notice below the techniques we have employed to make the code generic :-

```
def merge_columns(df, flow, flow_cols, occupancy_cols, prob_cols,
speed_cols):
    df_arr = []
    for i in range(0, len(flow.columns), 1):
        df_arr.append(pd.DataFrame(
            {
                "flow": df[flow_cols[i]],

def setup_columns(flow):
    columns = []
    columns_flow = []
    columns_occupancy = []
    columns_speed = []
    columns_prob = []
    for i in range(1, len(flow.columns) + 1, 1):
        columns_flow.append("flow" + str(i))
        columns_occupancy.append("occupancy" + str(i))
        columns_speed.append("speed" + str(i))
        columns_prob.append("prob" + str(i))
```

Method #1

Using Linear regression on nearby rows to predict flow

"Using the data of nearby lanes that provide useful information about flow measurements we try to predict the flow. That is, If we see that the flow of the current lane is significantly more than the the flow of other adjacent lanes then we know that it could be an outlier as in a practical scenario vehicles tend to automatically balance the load on different lanes.We use this relationship to predict the correct flow measurement, as given by a linear regression model:

Predicted(flow) = a * Nearby_Measured(flow) + b

where (a,b) are LR model parameters. We calculate the confidence of this prediction using the probability density of the nearby measurement, as:

Confidence(flow) = Prob_Desensity(Nearby(flow, speed, occupancy) "

Using this approach, where we predict the flow of a given detector using the detectors from other lanes, we tend to bring down the variance in the flows by a huge margin. If there are **n** lanes, n > 1, we get the predicted flow for each lane from other n-1 lanes. For ex., If a three lane road originally had the following flow values (8, 14, 255), the predicted value from the regression model is approximately (11,13,18) which seems much more plausible. This normalizes the extreme flow values and can be used as a possible data cleaning technique

The data which can be seen below, shows a small output for the Linear regression run on "1160" zone.

As can be seen in the image, the Linear Regression is trying to normalize the row data based on the flow values of the adjacent columns.

Steps Used in Linear Regression :-

TRAINING

We are training on the n-1 lanes i.e all the lanes except the one lane for which we need to predict the flow values.

OUTPUT

We output the normalized flow values in the columns "flow1_predict", "flow2_predict" and "flow3_predict".

Improvements Possible :-

We can try to train the Linear regression model on a subset of flow values which have a probability density greater than a threshold for e.g :- 0.01. This way we can avoid training on erroneous data and increase the efficiency of prediction model.

.

	flow1	flow1_predict	flow2	flow2_predict	flow3	flow3_predict
27	11	6.310883	6	7.469865	4	19.026071
28	5	5.523103	5	4.421795	4	15.693601
29	8	4.714598	4	5.927437	3	16.427183
30	7	7.077938	7	5.419425	3	17.840705
31	14	8.674223	9	8.993899	4	22.246727
32	5	5.523103	5	4.421795	4	15.693601
33	6	6.227983	6	4.856233	0	16.767153

Method #2

Using weighted sum of flow values of nearby timestamps-rows to predict flow

> "Measurements between consecutive time intervals are usually similar. Thus, we can predict the flow measurement by average of preceding and following time intervals, as given by:
>
> Predicted(flow) = w1*Preceding_Measured(flow) + w2*Following_Measured(flow)
>
> where (w1,w2) are weights between (0,1) such that w1+w2=1. We can calculate the (w1,w2) as:
>
> w1 = c1/(c1+c2), w2 = 1-w1
>
> where c1 is probability density of preceding measurement (calculated in Lab 9), similar for c2:
> c1 = Prob_Desensity(Preceding(flow, speed, occupancy))
> Finally, the confidence of this prediction can be estimated as: Confidence(flow) = min(c1,c2)."

The way we segregate data for the observations is very unique. We don't rely on just the preceding and the following value. To normalize any outlier which can dramatically affect the value of flow, we take into consideration two preceding and two following values such that they lie between the specified range of 150 sec. The threshold of 150 secs is used to highlight the fact that we can be sure of traffic conditions remaining the same under this range. Any value over this range, cannot be taken for calculation as an extreme value may present a valid flow. For e.g :- a traffic congestion or an accident might occur which can drastically skew the correct data if a very large range is taken.

SOMETHING NEW !!!!

For the calculation we consider two chunks of data

1. Current Timestamp - Preceding timestamp < 150 secs
   We take upto 2 values instead of just one for the preceding density and the preceding flow. This help us in normalizing outliers in the flow and density. This can be seen in the image below highlighted in red. The index 7 and 8 fall in the 150 sec timeframe and help in normalizing the flow for index 9.

2. Following Timestamp - Current Timestamp < 150 secs
   Similar as above we take 2 values for following density and the following flow instead of just one. This can be seen in the image below highlighted in green. The index 10 and 11 fall in 150 sec timeframe and help in normalizing the flow for index 9.

	detector	flow	occupancy	probability	speed	timestamp	Expected2
5	x2	21.0	18.0	0.000029	48.600000	2009-08-19T15:11:15	31.410646
6	x2	30.0	27.0	0.000004	33.000000	2009-08-19T15:12:15	30.196833
7	x2	30.0	26.0	0.000004	39.000000	2009-08-19T15:13:15	29.514184
8	x2	24.0	20.0	0.000057	46.800000	2009-08-19T15:14:15	30.717608
9	x2	27.0	21.0	0.000071	57.600000	2009-08-19T15:15:14	31.990196
10	x2	24.0	19.0	0.000063	60.600000	2009-08-19T15:16:14	32.333333
11	x2	22.0	19.0	0.000013	63.000000	2009-08-19T15:17:15	30.474114
12	x2	30.0	20.0	0.000023	41.600000	2009-08-19T15:18:14	30.110169
13	x2	27.0	23.0	0.000028	59.400000	2009-08-19T15:19:15	29.497245
14	x2	27.0	21.0	0.000037	60.600000	2009-08-19T15:20:15	28.970260

Thus, we can predict the flow measurement by normalizing the preceding and following time intervals by their probability densities.

FORMULA's USED IN CODE

W1 = totalDensityPreceding / (totalDensityPreceding + totalDensityFollowing)

W2 = 1-W1

averageFlow = W1 * (totalFlowPreceding/2) + W2 * (totalFlowFollowing/2)

bestConfidenceValue = min(totalDensityPreceding/2,totalDensityFollowing/2);

Method #3

Using the measurements directly.

> Most of the measurements are correct, so keeping all flow measurements unchanged might not lead to a result that is too bad. In this case, we simply predict a correct flow value with the measured flow value, as:
>
> Predicted(flow) = Measured(flow) The confidence of this prediction can be estimated by probability density of this point (calculated in Lab 9):

There are some erroneous measurement, some very extreme values ranging from -10 to 290, which are considered by this approach, but since we are using an ensemble learning method for predicting the correct flow values, the effects of these outliers are mitigated either by the other two weighted methods or by the very low probability density values corresponding to these outliers.

COMBINING RESULTS AND EXTRAPOLATION

We have described three methods to predict correct flow values. Each method outputs a predicted flow value, as well as confidence score of this prediction. We can thus merge multiple predictions by:

$$\text{Merged(flow)} = w1 * \text{Predicted\_1(flow)} + \ldots + w3 * \text{Predicted\_3(flow)}$$

where (w1,w2,w3) are weights between (0,1) such that w1+w2+w3=1. The w1 can be defined as (similar to w2 and w2):

$$w1 = c1/(c1+c2+c3)$$

where c1 is confidence of the flow prediction given by method 1