NIST IAD

DATA SCIENCE EVALUATION

PILOT REPORT

GROUP 30
AKASH AGARWAL (UFID 59577179)
NATASHA MANDAL (UFID 79475372)

NIST IAD DSE Pilot Evaluation Plan

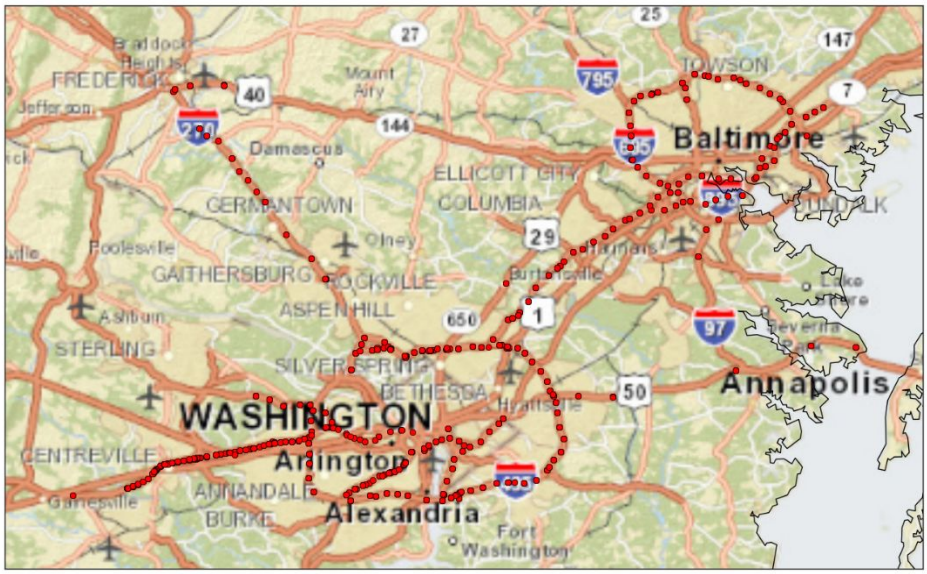
Out of the four outlined tasks we participated in the following two tasks:

- Cleaning: Detecting and correcting errors, omissions, and inconsistencies in flow values of detectors.
- Prediction: Predicting the different types of events which may occur within bounding boxes in the given time intervals.

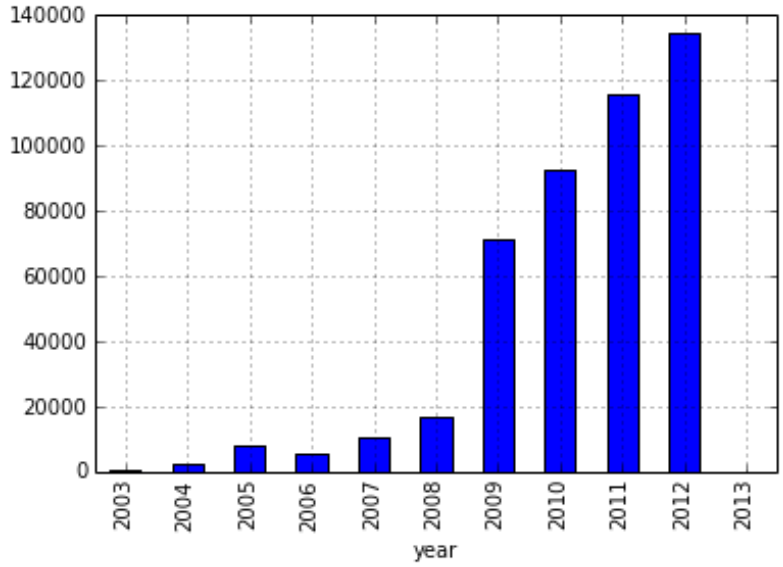
OVERVIEW AND EXPLORATORY ANALYSIS OF NIST DATA:

To get an idea about the data, we performed some visualizations on the data:

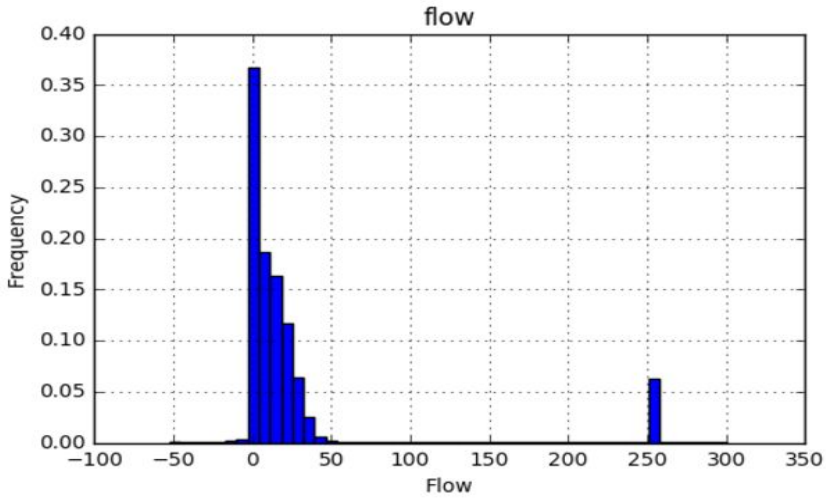
- We plotted the latitudes and longitudes of the detectors from detector_lane_inventory.tsv.

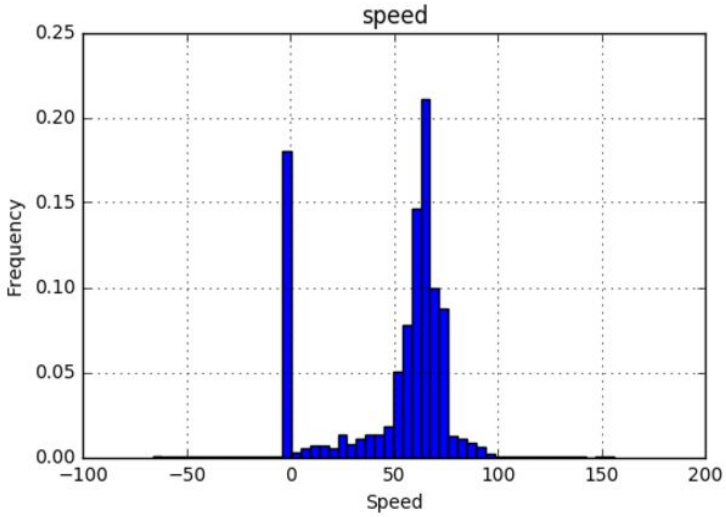


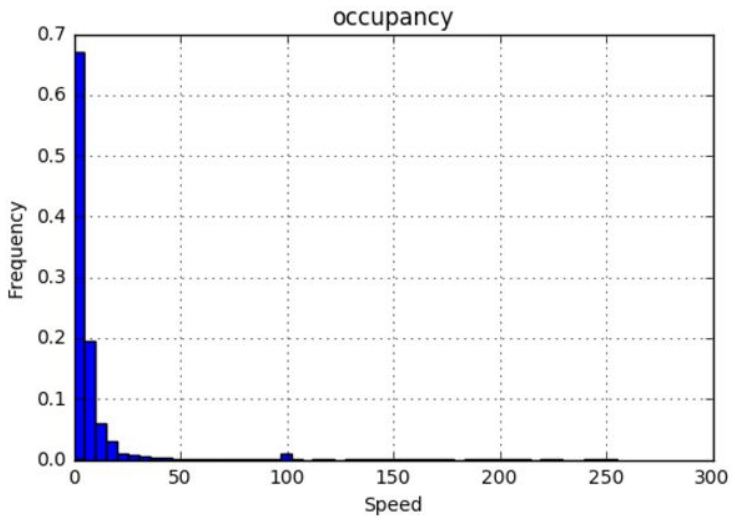
- We created a bar chart for the number of events occurring across the years. We took the data from events_train_holdout.tsv.



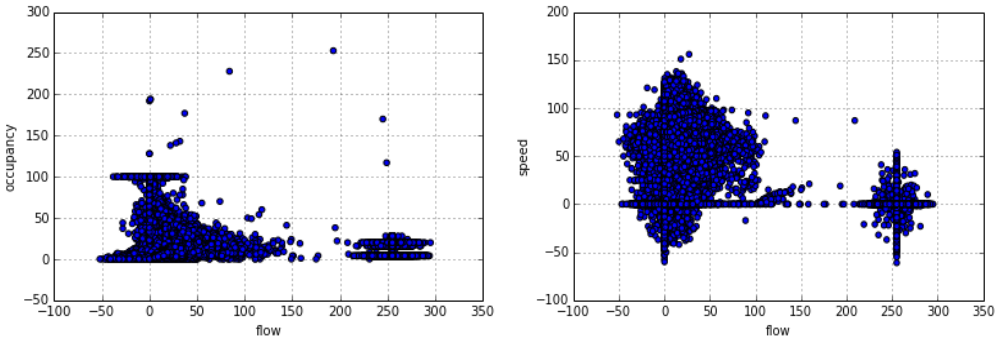
- We plotted histograms for the flow, speed and occupancy for the file cleaning_test_06_09.tsv. This helped us see how the values for flow, speed and occupancy were distributed in that file.

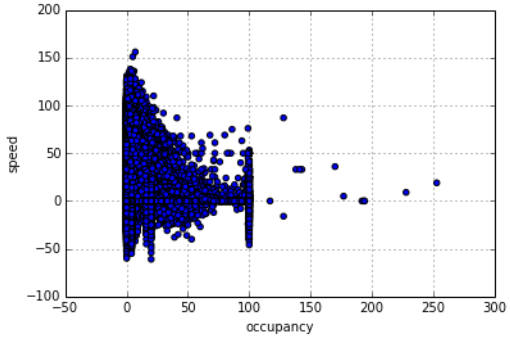

flow
Frequency
Flow
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−100
−50
0
50
100
150
200
250
300
350


speed
Frequency
Speed
0.25
0.20
0.15
0.10
0.05
0.00
−100
−50
0
50
100
150
200



- We made some additional visualizations, of which the most useful were:





- Some of the main observations which we made based on the visualization were:

- We noticed that most of these detectors are in Washington.
- Some of the flow values were negative.
- Many records were present with values of 255.
- Most of the occupancy values were between 0 and 100, but there were some outliers.
- Some of the speed values were negative.
- The number of events which had occurred across the years had an increasing trend.

CLEANING TASK - 1:

In this task we were supposed to find the probabilities of correctness for the vectors (of flow, speed and occupancy) for the five given zones. Our first intuition was that the vectors, in which the variables had extreme values, should be given a lower probability than the vectors with less extreme values. Although all the variables didn't follow a normal distribution, we thought that using a normal distribution would help in assigning a lower probability to vectors with 'extreme' values.

In our case, the normal distribution needed to be computed across multiple variables. A multivariate normal distribution is the extension of the 1-dimensional normal distribution problem to multiple dimensions. Thus, we used a multivariate normal distribution to compute the probability distributions.

We calculated the mean vector and covariances of the zone's vectors. We gave the vectors, the mean vector and the covariances as parameters to the multivaraie_normal.pdf() function (present in the scipy.stats package). This function gave us the probability of each of the vectors.

OBSERVATIONS:

The following observations were made about the probabilities which were predicted:

- Most of the vectors with extreme values (when compared to the mean) were given a low probability (ex. the vectors with flow values as 255 got a low probability).
- Some of the vectors which should have gotten a low probability didn't get a low probability. (ex. The vectors with negative flow, speed or occupancy values should have gotten a much lower probability). These cases had to handled manually.
- The vectors which were closer to the mean vector got a higher probability than the vectors which were far away from the mean.

RESULTS:

- Based on the how much the vectors varied from the mean vector, we used multivariate_normal.pdf() to compute the probability of that vector being correct.
- For certain cases (like handling negative flow, speed and occupancy values) a probability of 0 had to be assigned manually.
- The same algorithm could be extended across all the zones.

- All the zones took less than 2 minutes to run since the function multivariate_normal.pdf() operated on the whole dataframe to give the probabilities.

CLEANING TASK - 2:

In this task, we were supposed to correct the flow values based on the probabilities of the vectors (flow, speed, occupancy). For the flow values with low probabilities, there were several approaches we could consider for correcting the flow values:

- Spatial: There is a likelihood that the surrounding lanes might have similar traffic and that the flow values of those detectors might be similar to what the flow value could be. We used the function linregress() from the scipy.stats package to compute a linear regression for the nearby lanes (for the lanes with two nearby lanes we took the average value of the two neighboring lanes) to derive a flow value. The confidence [C1] level was based on the probability of the nearby lanes which were considered for the linear regression.
- Temporal: There is a likelihood that the flow values of the preceding and following timestamps might give an indication of what the actual flow value might be. We assigned weights w1 [= c1/(c1+c2)] and w2 [= c2/(c1+c2)] to the preceding row's flow value (whose probability is c1) and the following row's flow value (whose probability is c2). Based on the weights we calculated the new flow value. We

assigned the confidence [C2] to be the minimum of the probabilities of the preceding row and following row.

- Original: The original flow value might be correct. We assigned the confidence [C3] to be the original probability of that row.
- We merged the multiple methods by assigning weights to each of the methods. [W1 = C1/(C1+C2+C3) etc.]. Based on the weights, we gave a higher priority to those methods which have a higher probability value.
- Since most of the flow values are correct, we considered to correct only extreme cases (and those flow values which were above a certain threshold - based on the histogram we had plotted for flow values).

OBSERVATIONS:

The following observations were made about the probabilities which were corrected:

- For the nearby timestamp method, if a detector wasn't working for a long period of time, very low confidence results for this method was obtained.
- For the nearby lane method, erroneous values often contained very high values, so the predicted value also tended to be high (since it was a linear regression).
- We assumed that we didn't need to correct all the values since most of the flow values are correct and that we could correct values for some extreme cases. Because of this, some erroneous flow values might be missed.

RESULTS:

- For certain cases, the values predicted by all the methods were very low. In such cases, we thought we could replace it with the median flow value.
- The same algorithm could be extended across all the zones.
- It took very less time for the linregress() function to run since it could operate on the whole dataframe and the whole script ran in less than 2 minutes for all the zones.
- We made a manual correction for some cases:
    - If occupancy was 100 and speed was 0, we made the flow as 0
    - If occupancy was 0 and speed was 0, we made the flow as 0

PREDICTION TASK:

In the prediction task we predicted the counts of 6 different types of events that would take place within the time intervals for bounding boxes.

We used the following two methods for doing our prediction:

- Ordinary Least Squares (OLS)-

We had noticed that there was lesser event data during the initial years (when compared to the later years). Multiple reasons could be possible for this. One of the reasons could be that the sensor's detection capabilities increased over time. If that is the case, then

it can be assumed that there might be linear trends for the number of events of different types occurring within bounding boxes. Because of that, we thought of using Ordinary Least Squares method. A linear regression is formulated as below:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + ... + w_p x_p$$

The target variable is assumed to be a linear combination of the input variables. Here, $\hat{y}$ is the predicted value, $w = (w_1, ..., w_p)$ is the coefficient and w_0 is the intercept. In OLS, the coefficients are chosen to minimize the sum of squares between the observed responses in the dataset and the responses predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_w ||Xw - y||_2{}^2$$

- K Nearest Neighbor Regression (KNN)-

KNeighborsRegressor implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. Uniform weights can be assigned to the neighbors being considered. Alternatively, weights can be assigned based on an inverse of the distance of the neighbors. One disadvantage of this approach is that future predictions for all years after the last year being considered, tends to be the same.

For extending the prediction task, we thought we could implement some more methods for predicting the flow values. We wanted to see if different methods could lead to a better prediction. We thought of the following methods:

- Bayesian Ridge Regression (BR)-

Bayesian Regression techniques can be used to include regularization parameters which is tuned based on the data at hand. To obtain a fully probabilistic model, the output y is assumed to be Gaussian distributed around Xw:

$$p(y|X, w, \alpha) = \mathcal{N}(y|Xw, \alpha)$$

In Bayesian Ridge, the prior for the parameter w is given by a spherical Gaussian:

$$p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}\mathbf{I_p})$$

The priors over α and λ are gamma distributions. The advantages of Bayesian Regression techniques are that:

- It adapts to the data.
- Regularization parameters can be included.
- Compared to OLS, the coefficient weights are slightly shifted towards 0, which stabilises them.

- Lasso LARS -

It consists of a linear model trained with ℓ_1 prior as regularizer. The objective function to minimize is:

$$\min_w \frac{1}{2n_{samples}}||Xw - y||_2^2 + \alpha||w||_1$$

The lasso estimate solves the minimization of the least-squares penalty with $\alpha||w||_1$ added, where α is a constant and $||w||_1$ is the ℓ_1-norm of the parameter vector.

LARS (Least Angle Regression) can be used to find the Lasso estimate. This yields the exact solution, which is piecewise linear as a function of the norm of its coefficients.

- Support Vector Machine Regression (SVR)-

Given training data an optimal hyperplane is computed which categorizes the data. The goal should be to find the line passing as far as possible from all points. Thus it tries to assign the hyperplane that gives the largest minimum distance to the training examples. It needs to consider only a subset of the points which is an advantage. We used the linear kernel for the SVR function.

We grouped the data frame by year while calculating the counts which gave us the event frequency for a given year. We noticed that the time intervals for each of the bounding boxes varied. While predicting the number of events which would occur in that time interval, we need to scale the prediction for that year based on the time interval. For dealing with this, we calculated the difference between the start and end times in hours.

$Predicted\_value\_year$: value predicted for the year by the model

8760 : Number of hours in a year (with 365 days)

$Difference\_hours$: difference between the start and end fields for the bounding box

$Predicted\_value$: predicted value for the bounding box within the time interval

$$Predicted\_value / Difference\_hours = Predicted\_value\_year / 8760$$

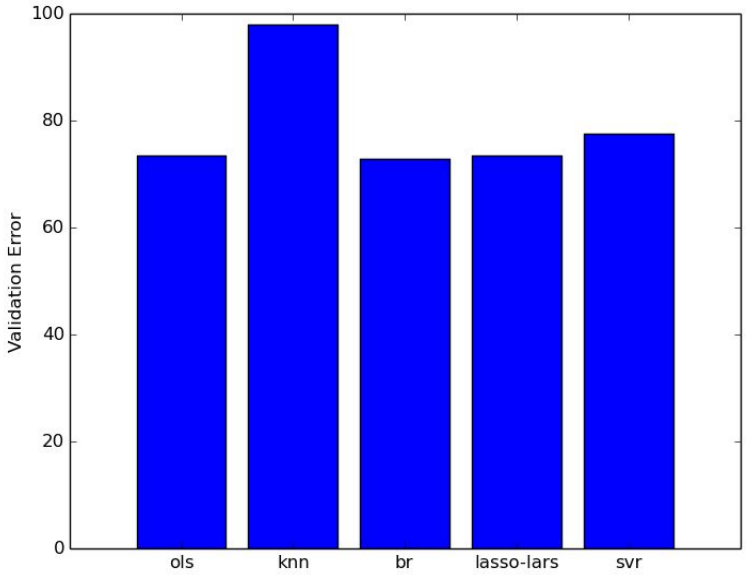
$$\Rightarrow Predicted\_value = (Difference\_hours * Predicted\_value\_year) / 8760$$

OBSERVATIONS:

- The final OLS model validation error is around 73.5.
- The final KNN model validation error is around 97.97.
- The final BR model validation error is around 72.73.
- The final Lasso LARS model validation error is around 73.35.
- The final SVR model validation error is around 77.47.

RESULTS:

- The Bayesian Ridge Regression provided the least validation error out of the five methods we had evaluated.
- In this task, we predicted the flow values based on the OLS method
- We scaled the prediction based on the difference of the timestamp in hours.



Some of the main lessons which we learnt while carrying out these tasks were:

- It is always good to get an idea about the data (by using visualizations) before we try to perform any tasks on it. It helps in determining which approach might be appropriate for the problem.
- Within Pandas, carrying out dataframe operations was faster than performing operations on each of the rows.
- Parallelizing the task might lead to improvements in speed.
- The data might have some outliers and it might be good to focus more on the outliers (for example, since most of the flow values were correct, it might not be a good idea to correct all the flow values).
- There may be some cases we may need to handle manually.
- Although the validation error might be greater for a method, it might give better results for prediction.