Introduction to Data Science

NIST Project :- Prediction

BY ABHISHEK NIGAM AND ABHISEK MOHANTY
NIST-19

Introduction

Given event information in the past years, we are asked to predict future event occurrence.
We are interested in 6 different types of events (though there are more in the
given data), they are:
1. Accidents and Incidents (A).
2. Roadwork (R).
3. Precipitation (P).
4. Device Status (D).
5. Obstruction (O).
6. Traffic Conditions (T).

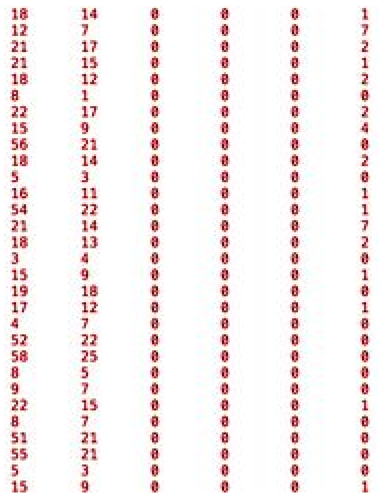
CONTENT INDEX :-

a. RESULTS AND OBSERVATIONS/ PERFORMANCE EVALUATION
b. ALGORITHM USED AND MODIFICATIONS TO THE GIVEN ALGORITHM
c. METHODS USED AND THEIR DESCRIPTION
    i. METHOD 1 : - Linear regression
    ii. METHOD 2 :- Ridge Regression
    iii. METHOD 3 :- Polynomial Ridge Regression
    iv. METHOD 4 :- SVM Regression
    v. METHOD 5 :- Neural Networks
d. COMBINING RESULTS AND EXTRAPOLATION
e. CONCLUSION

Results and Observations (Predicted Values)

In this section we compare the predicted values of different regression models which we employed to compare our prediction results. Below we present a random sample of 60 geo-boxes from our prediction results and we try to draw a comparison of results based on the values predicted by these models.

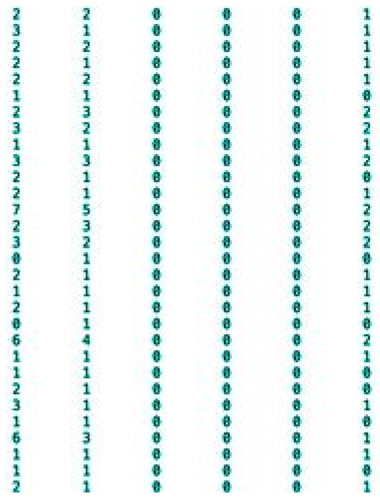
Linear Regression



18	14	0	0	0	1
12	7	0	0	0	7
21	17	0	0	0	2
21	15	0	0	0	1
18	12	0	0	0	2
8	1	0	0	0	0
22	17	0	0	0	2
15	9	0	0	0	4
56	21	0	0	0	0
18	14	0	0	0	2
5	3	0	0	0	0
16	11	0	0	0	1
54	22	0	0	0	1
21	14	0	0	0	7
18	13	0	0	0	2
3	4	0	0	0	0
15	9	0	0	0	1
19	18	0	0	0	0
17	12	0	0	0	1
4	7	0	0	0	0
52	22	0	0	0	0
58	25	0	0	0	0
8	5	0	0	0	0
9	7	0	0	0	0
22	15	0	0	0	1
8	7	0	0	0	0
51	21	0	0	0	0
55	21	0	0	0	0
5	3	0	0	0	0
15	9	0	0	0	1

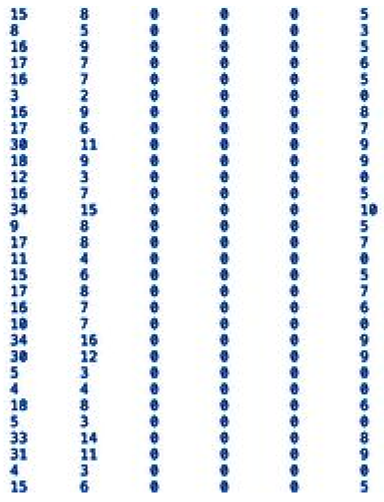
Ridge Regression

_



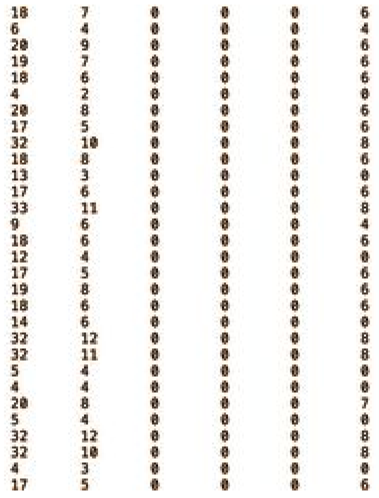
2	2	0	0	0	1
3	1	0	0	0	1
2	2	0	0	0	1
2	1	0	0	0	1
2	2	0	0	0	1
1	1	0	0	0	0
2	3	0	0	0	2
3	2	0	0	0	2
1	1	0	0	0	1
3	3	0	0	0	2
2	1	0	0	0	0
2	1	0	0	0	1
7	5	0	0	0	2
2	3	0	0	0	2
3	2	0	0	0	2
0	1	0	0	0	0
2	1	0	0	0	1
1	1	0	0	0	1
2	1	0	0	0	1
0	1	0	0	0	0
6	4	0	0	0	2
1	1	0	0	0	1
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	1
1	1	0	0	0	0
6	3	0	0	0	1
1	1	0	0	0	1
1	1	0	0	0	0
2	1	0	0	0	1

Polynomial Ridge Regression



15	8	0	0	0	5
8	5	0	0	0	3
16	9	0	0	0	5
17	7	0	0	0	6
16	7	0	0	0	5
3	2	0	0	0	0
16	9	0	0	0	8
17	6	0	0	0	7
30	11	0	0	0	9
18	9	0	0	0	9
12	3	0	0	0	0
16	7	0	0	0	5
34	15	0	0	0	10
9	8	0	0	0	5
17	8	0	0	0	7
11	4	0	0	0	0
15	6	0	0	0	5
17	8	0	0	0	7
16	7	0	0	0	6
10	7	0	0	0	0
34	16	0	0	0	9
30	12	0	0	0	9
5	3	0	0	0	0
4	4	0	0	0	0
18	8	0	0	0	6
5	3	0	0	0	0
33	14	0	0	0	8
31	11	0	0	0	9
4	3	0	0	0	0
15	6	0	0	0	5

SVM Regression



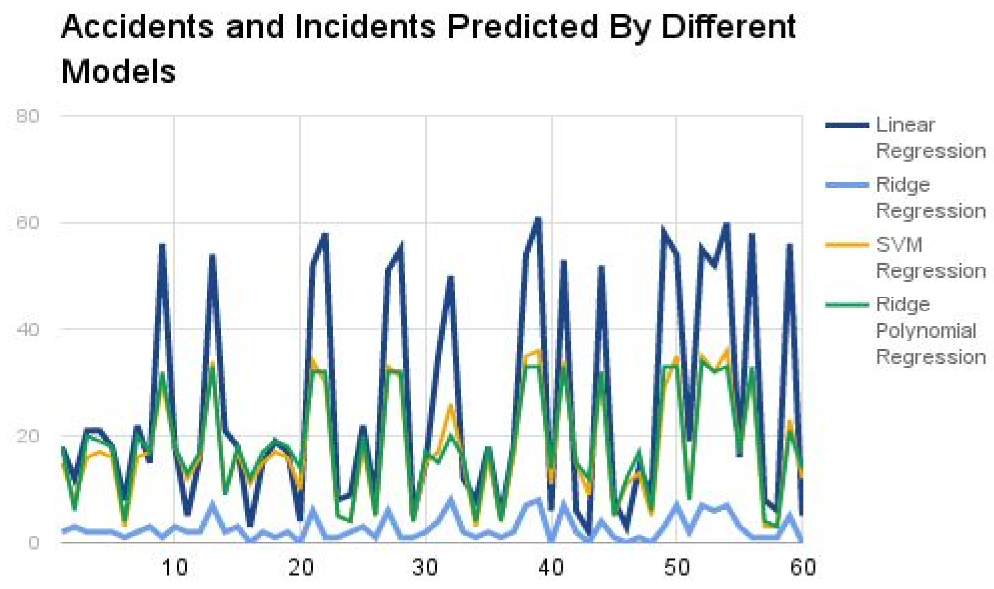
18	7	0	0	0	6
6	4	0	0	0	4
20	9	0	0	0	6
19	7	0	0	0	6
18	6	0	0	0	6
4	2	0	0	0	0
20	8	0	0	0	6
17	5	0	0	0	6
32	10	0	0	0	8
18	8	0	0	0	6
13	3	0	0	0	0
17	6	0	0	0	6
33	11	0	0	0	8
9	6	0	0	0	4
18	6	0	0	0	6
12	4	0	0	0	0
17	5	0	0	0	6
19	8	0	0	0	6
18	6	0	0	0	6
14	6	0	0	0	0
32	12	0	0	0	8
32	11	0	0	0	8
5	4	0	0	0	0
4	4	0	0	0	0
20	8	0	0	0	7
5	4	0	0	0	0
32	12	0	0	0	8
32	10	0	0	0	8
4	3	0	0	0	0
17	5	0	0	0	6

ACCIDENTS AND INCIDENTS :-

As can be noticed in this case, accidents and Incidents take up a fair share of the overall incidents being predicted by by our model. On a random sample of 60 geo-boxes, we have plotted below the number of predicted accidents. These have been segregated by all the models which we have used for prediction.

Interesting Observations :-

a. Ridge Linear Regression overfits more compared to all other models. The values predicted by ridge regression had a high mean square error value which suggests that the model didn't predict the values correctly for our data(Most probably due to overfitting).
b. Linear regression has the highest predicted values and we noticed that the values predicted by the Linear regression had the lowest Mean Square error for the overall data implying that the values were very close to the optimum values.



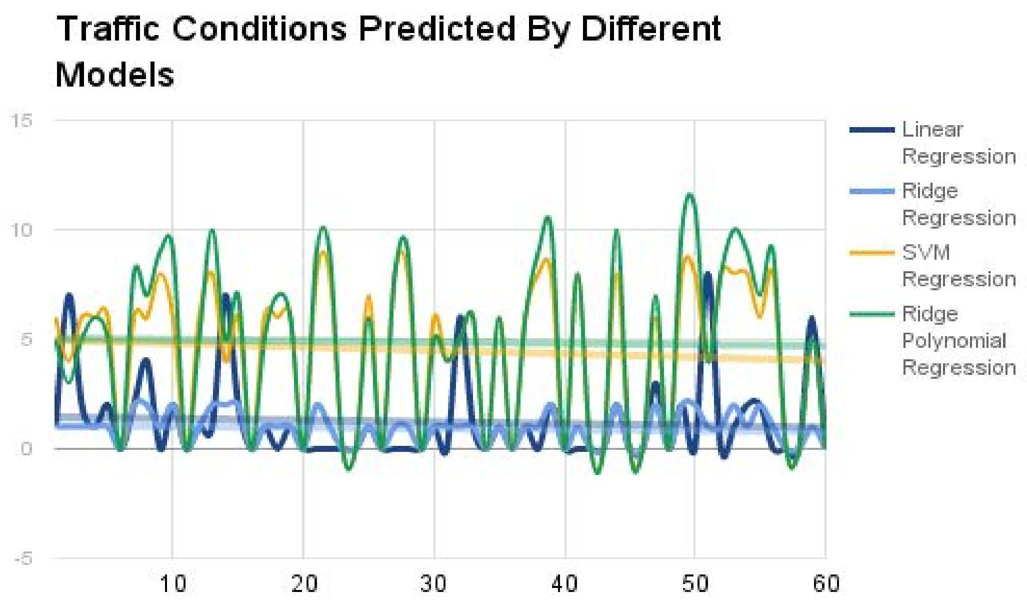
TRAFFIC CONDITIONS:-

As can be noticed in the case of traffic conditions, we have plotted a graph of the predictions done on the same random sample of 60 geo-boxes.

Interesting Observations :-

a. One observation particularly intriguing was that for Polynomial Ridge regression the values went below the zero value. Now, since the predicted values cannot be below zero we made necessary changes to the code to avoid these. This indicated a negative trend in the curve, pointing towards the fact that there have been better traffic management practices by the city planners.
b. As in previous scenario, We noticed that the values predicted by the Linear regression had the lowest Mean Square error for the overall data implying that the values were very close to the optimum values.
c. As can be noticed, there are four lines horizontally bisecting the graph. These are the **Trend Lines** for the graph and indicate the trend of prediction by different models.

-

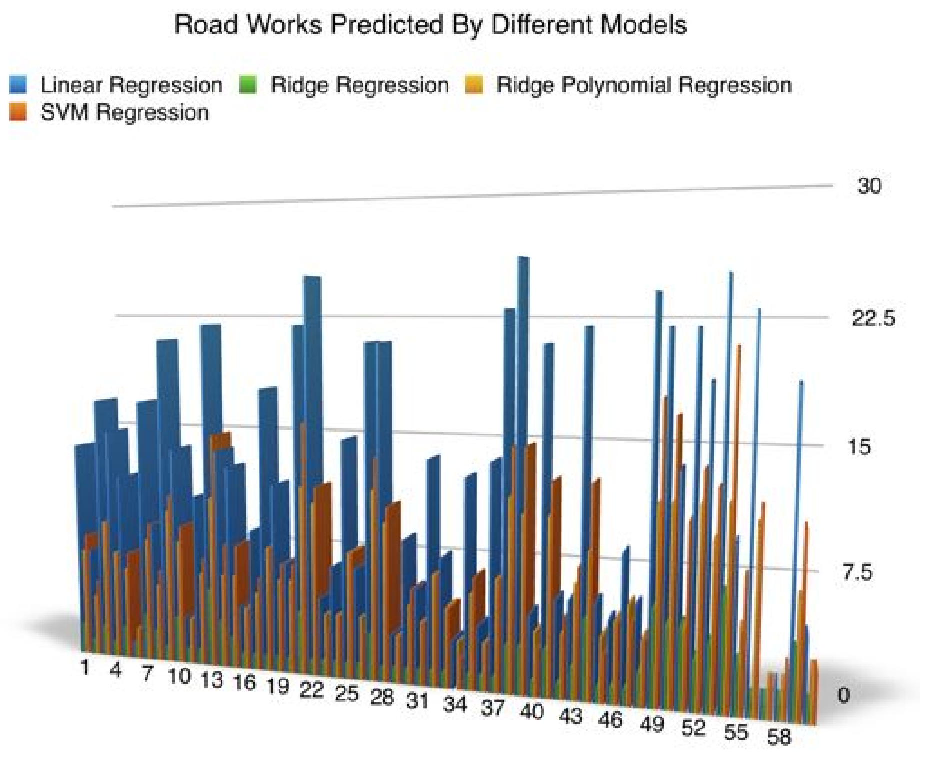


ROADWORKS (3D-PLOT):-

As can be noticed in the case of Roadworks, we have plotted a graph of the predictions done on the same random sample of 60 geo-boxes.

Interesting Observations :-

a. This 3D Plot shows in the comparison, that the highest number is predicted by the Linear regression followed by SVM Regression and then Ridge polynomial regression. Interestingly, The mean square error in the case of roadworks was minimum for Ridge Polynomial Regression. This shows that the Ridge Polynomial Regression suited best for this observation.



Precipitation, Device Status, Obstruction :-

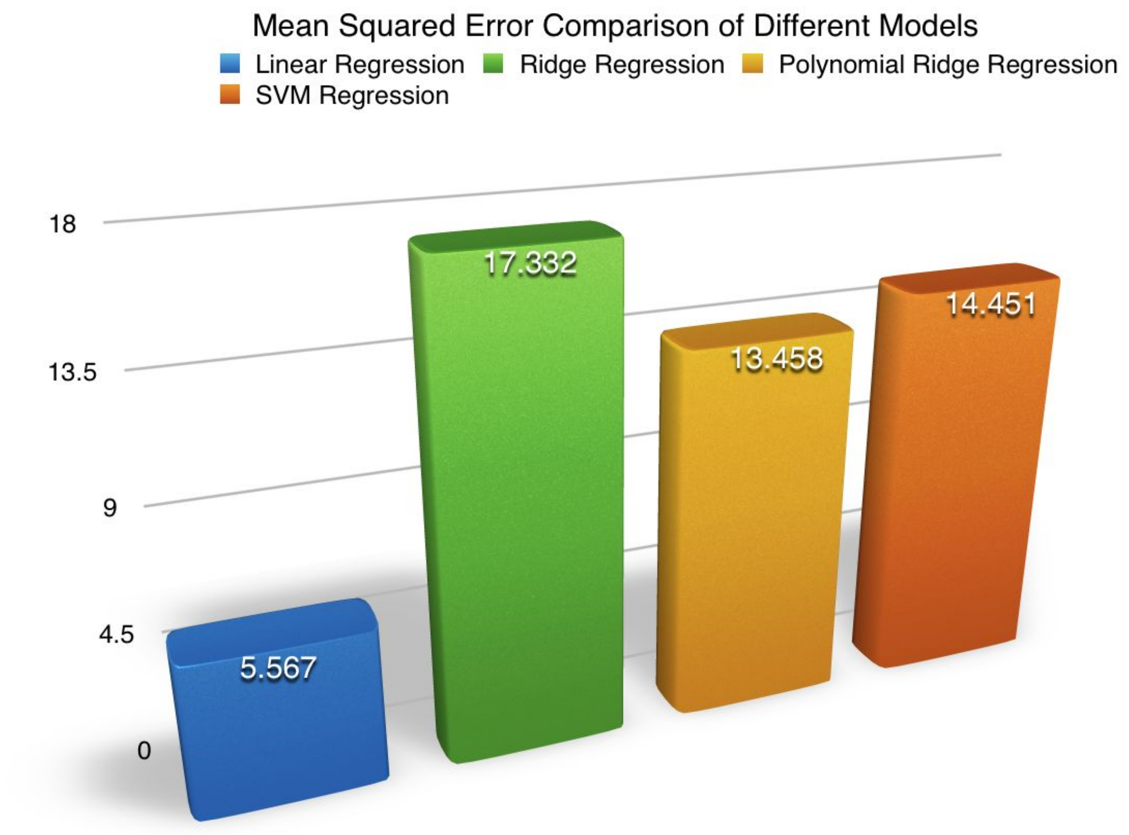
We have clubbed the result for these three observations into a single slide as we don't have significant data in our random sample for these three events. As these three events occur at highly random intervals. There are some interesting observations which we can make about these events :-

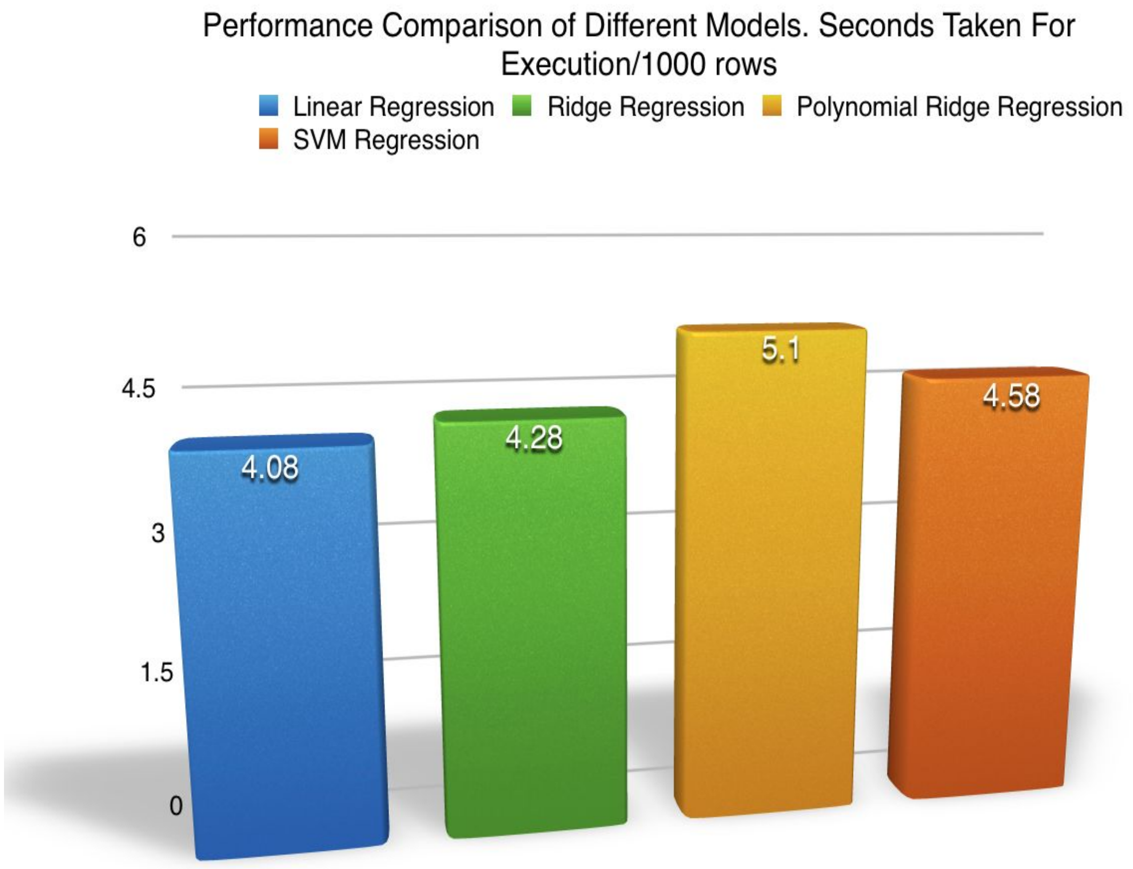
Interesting Observations :-

a. Polynomial Ridge Regression gives the lowest mean squared error followed by Linear regression.

Performance Evaluation

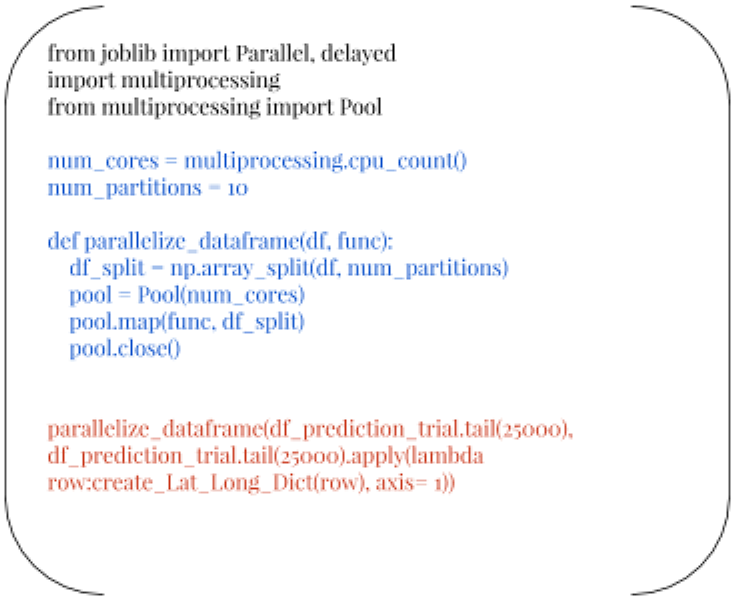
The performance evaluation of the four regression models gives us interesting insight into how they perform. We have the following data of the different models.





In general, we observed that while running the regression for all the models at the same time and predicting the results for each of them, **we were able to complete the prediction of all ~30,00,000 records within 2-3 hours**. Apart from normal optimization techniques, we also employed **Parallel Processing using the python library.** The code that we employed for multiprocessing can be found below.

Also, We observed that the multiprocessing enhanced the runtime of our algorithm by almost two times.



```
from joblib import Parallel, delayed
import multiprocessing
from multiprocessing import Pool

num_cores = multiprocessing.cpu_count()
num_partitions = 10

def parallelize_dataframe(df, func):
   df_split = np.array_split(df, num_partitions)
   pool = Pool(num_cores)
   pool.map(func, df_split)
   pool.close()


parallelize_dataframe(df_prediction_trial.tail(25000),
df_prediction_trial.tail(25000).apply(lambda
row:create_Lat_Long_Dict(row), axis= 1))
```

The major bottleneck in performance which we faced was while running the code for Neural networks. It took the algorithm ~15 mins to train the neural network and output the predictions for 10 rows. Since, this would have taken a long time, we didn't run it on the complete dataset.

Models Used and Changes proposed to the existing algorithm

Algorithm Implementation:

The algorithm specified in class considered only year and count as the parameters for the Regression models. One drawback of this model can be the task of predicting the number of accidents per month which must be computed using the formula :

$$Accidents\ per\ month \ = \ Total\ Accidents\ in\ year / 12$$

This does not make much sense in the context of the prediction problem as monthly accidents can be affected by a lot of factors. One of them is the fact that since Washington has some of the chilliest winters with a lot of snowfall around, **the number of accidents in the winter months December to February could be greater than the summer months**. **Hence, we have used both Year and Month as our features in training the model.**

Machine Learning:

Linear Regression: Linear Regression seems to be the best technique for this kind of a problem since the amount of data points per box is not a huge number and the data won't be scattered by a huge margin. It is therefore better to use a linear model, or a basic polynomial model so that the model does not overfit.

Kernel Ridge Regression: We have used Ridge Regression here for two main reasons. First, the total number of training data per box is too low and Ridge Regression is helpful in such scenarios, Secondly, we wanted to try out different kernels and compare the Mean Square Errors. We have used Poly and RBF kernels in our submission but the MSE are comparably higher than Linear Regression.

Support Vector Regression: The third algorithm we have used is the Support Vector Regression using the default RBF kernel.

1 Hidden Layer Neural Network Regression: Though the amount of data is not enough for training a MLP, we have also implemented a 1 Hidden Layer neural network model in Keras as we wanted to see how it compares with the other models. We couldn’t run it on the entire dataset as it takes an extremely long time (on CPU; GPU will do a lot better). But on a subset of data, we observed that it has high MSE which makes sense since a 1H layer network will definitely overfit the data.

Conclusion and Improvements:

Even though the regression model has the lowest MSE among all the models, we feel that a **Linear Regression model by itself would not give the perfect prediction since the training data is too less and the model does not generalize properly. Using only Kernelized models won't work either as they are overfitting and hence have higher MSE.** *We could probably use an ensemble method that uses both Linear Regression and one of the Kernel Models/Neural Net model (We would prefer Kernel Models over Neural Networks as the time taken by the kernel models is much much lower and not much difference in the performance) to get the final prediction.*

$$Final\ Model = c1 * LinearRegression + c2 * (Kernelized\ Ridge\ Regression\ or\ SVR)$$

where c1 and c2 are the weights of the models.

Neural Network

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor

reg_iter = 1
total_error_nn = []
def neural_net_model(x_train, y_train, x_test, y_test, x_predict):
    if len(x_train) < 2 or len(x_test) == 0 or (len(x_train) != len(y_train)) or (len(x_test) != 
len(y_test)) :
        return np.zeros(2)

    x_train = np.asmatrix(x_train)
    y_train = np.transpose(np.asmatrix(y_train))
    x_test = np.asmatrix(x_test)
    y_test = np.transpose(np.asmatrix(y_test))
    x_predict = np.asmatrix(x_predict)

    model = Sequential()
    model.add(Dense(13, input_dim=2, init='normal', activation='relu'))
    model.add(Dense(6, init='normal', activation='relu'))
    model.add(Dense(1, init='normal'))

    # Compile model
    model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
    model.fit(x_train, y_train, nb_epoch=1, batch_size=1024,verbose=0)
    predicted_nn = model.predict(x_test)

    error_nn = np.mean(np.asarray(predicted_nn - y_test) ** 2, axis=1)
    total_error_nn.append(error_nn)
    return model.predict(x_predict)
```