**STAT 652 REPORT**

1. **INTRODUCTION**

This report is focussed on prediction of flight delay in year 2013 by doing predictive modelling using R packages and functions. Knowing reasons for flight delay is important to prevent future delays,

leading to better business and ease of travel in future.

This report will describe the various stages involved in predictive modelling: exploratory data analysis feature analysis on data, data wrangling, applying various regression models, applying hyperparameter tuning for model performance, limitations of approaches tried and conclusions and results. There will be several plots and graphs that help visualize data and inferences.

1. **DATASET**

The report uses nycflights13 package and gives information about all flights departing in 2013. The data is combined from four datasets for this package namely:

Flights

Weather

Airports

Planes

The dataset has 200000 rows with 43 predictor variables.

**EXPLORATORY DATA ANALYSIS:**

**Data Pre-processing:** This involves cleaning of data and conversion of variables.

1. Conversion to factor variables:

We convert all character variables to factor variables.

1. Handling Missing data:

On getting summary of our dataset we find that predictors in plane dataset have 15% of data missing.

First, we remove columns with more than 5 % data missing via rule of thumb.

We removed the missing values as replacing them with mean or median can introduce bias leading to less accuracy. Hence, we completely omit those rows. Now, we are left with 184316 rows and 33 columns.

**Distribution of output variable:**

Below are the plots of that show the variation of output variable(dep\_delay) with different predictors.

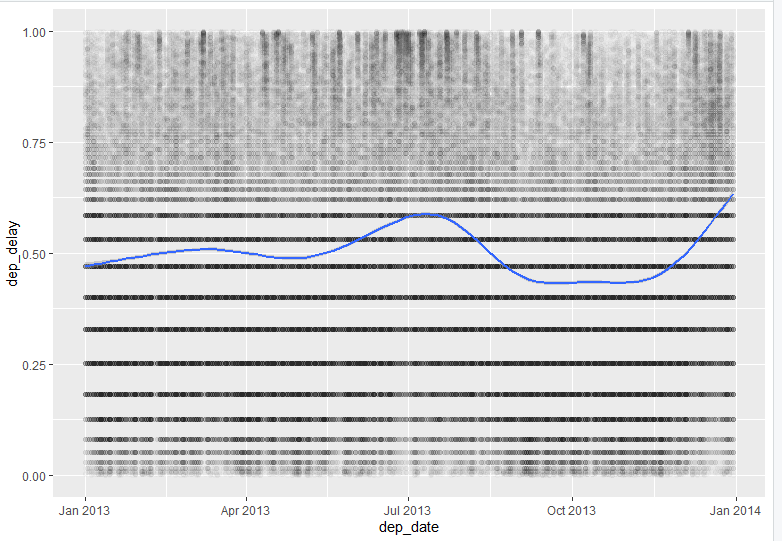


Fig: Non linear trend between dep\_delay and dep\_date

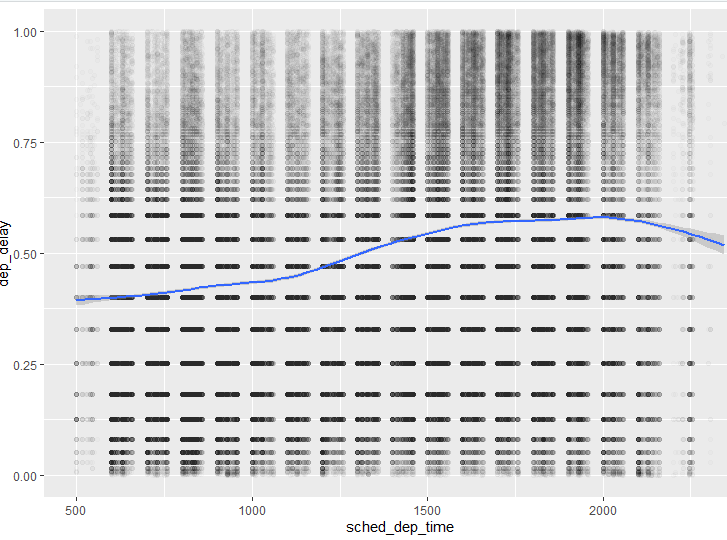


Fig: Non linear trend between dep\_delay and sched\_dep\_time

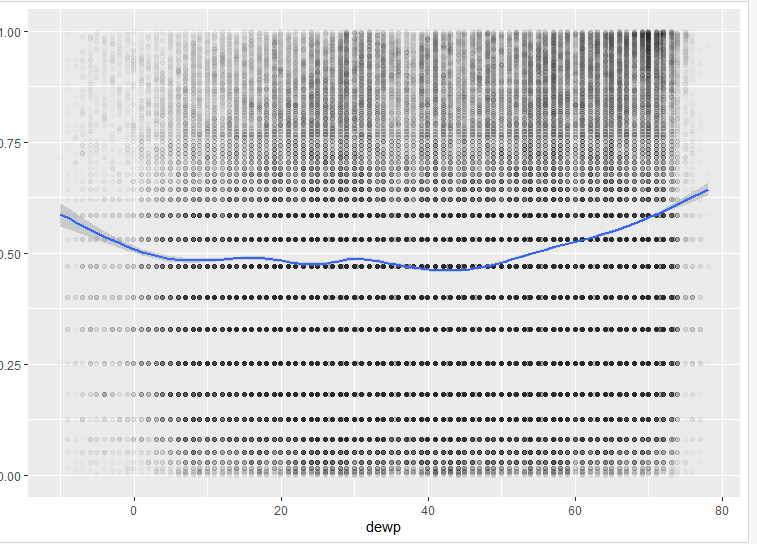


Fig: Non linear trend between dep\_delay and dewp

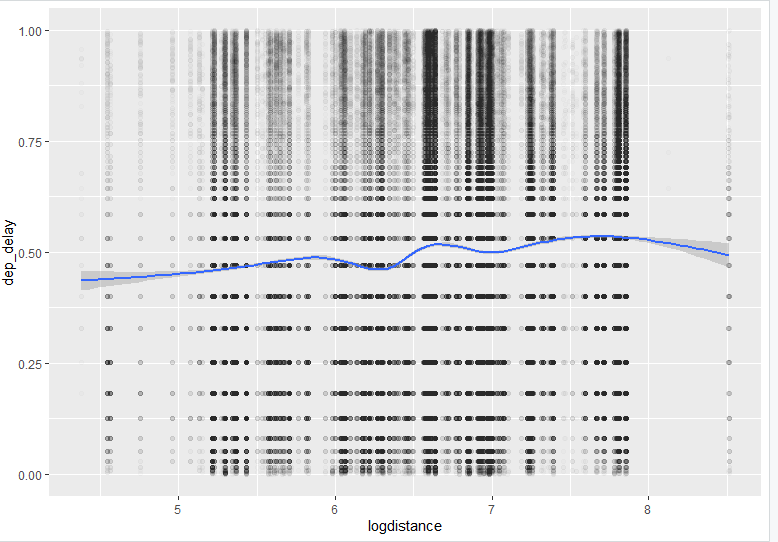


Fig: Non linear trend between dep\_delay and dep\_date

**Summary:** We observe non linear relationship between dep\_delay and other numeric variables like sched\_dep\_time, logdistance, dewp, dep\_date.

**Splitting the data:** We split the dataset into training (two-thirds of total) and validation (one-third of total) to apply algorithms and evaluate mean squared error.

**Data Wrangling:** This step involves feature analysis and engineering.

1. **Output Variable analysis**: We see that dep\_delay is highly rightly skewed when summarizing by date, airline, carrier. We apply rank to normalize the delay to get empirical quantiles which are comparable when getting values on test data.
2. **Feature engineering:** We see that we can construct date with columns year, month and day. We convert precipitation to just values 0 and 1 on a threshold of 0. We converted distance and alt to log(distance) and log(alt) for highly skewed distribution to less skewed.
3. **Feature Analysis:** The predictors arr\_time, dep\_time and arr\_delay are not related to our output variable dep\_delay as they give information after the delay has happened. Hence, we remove them. Other features like tail\_num, name and flight\_num do not give any relevant information as these are just numbers or name. Moreover, we remove redundant features like hour, minute (depicted by sched\_dep\_time) and time\_hour(same as sched\_dep\_time) alongwith tz, dst and dest( depicted by variable tzone).Now, we are left with 17 columns.
4. **METHODS**

From EDA, we found non linearity in data and hence I implemented following methods that deal with non-linearity. Brief description of these methods is given below:

1. **Decision Tree Regression** – It builds model in the form of tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches, each representing values for the attribute tested. Leaf node represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. This method works on both numerical and categorical data.
2. **GAM**- Generalized additive models (GAMs) provide a general framework for generalized additive model extending a standard linear model by allowing non-linear functions of each of the variables, while maintaining additivity. Just like linear models, GAMs additivity can be applied with both quantitative and qualitative responses.
3. **Gradient Boosting**- Boosting is a sequential process; i.e., trees are grown using the information from a previously grown tree one after the other. GBMs build an ensemble of shallow and weak successive trees with each tree learning and improving on the previous. When combined, these many weak successive trees produce a powerful “committee” that are often hard to beat with other algorithms.
4. **XG BOOST:** XGBoost is a scalable and accurate implementation of gradient boosting machines and it pushes the limits of computing power for boosted trees algorithms. The implementation of XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement with addition of regularization parameters.

I took Gradient Boosting from textbook and ran on train and validation set with its default hyperparameters as follows

1. Ntrees: integer specifying total number of trees to fit
2. Shrinkage: It is learning rate or step size reduction. It is applied to each tree in expansion.
3. Interaction\_depth: It is the number of splits to perform on a tree.

With default hyperparameter values of ntrees=1000, shrinkage=0.01 and interaction\_depth=1, the value

of MSE(0.07208569) on validation is a bit high than the one from GAM(0.07066494).

So, I used **hyper parameter tuning method** to find good choice of parameters to model.

1. **RESULTS**

The least mse on validation set is obtained with ntrees=2000 and shrinkage =0.1 with interaction\_depth=3.

The best test result for GBM is obtained is: 0.06347855

The variable importance plot and relative importance of each variable is as follows:

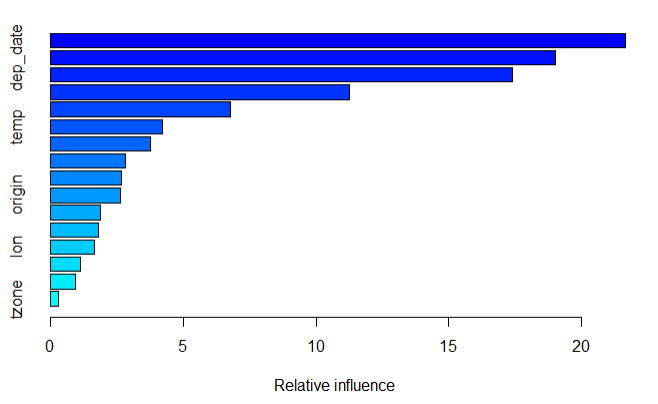


Fig: Relative Influence of predictors on dep\_delay

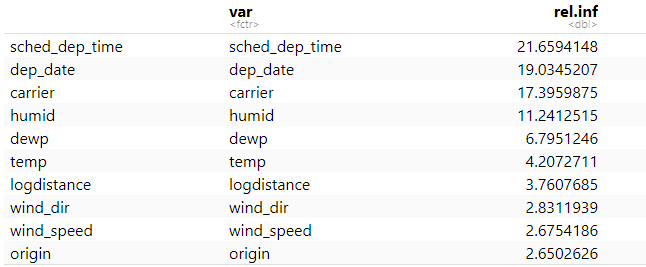


Fig : Values of relative importance of predictor values on dep\_delay

We see that sched\_dep\_time, dep\_date, carrier, humid, dewp, and temp are most important variables in determining dep\_delay.

Other models I tried are as follows and following table shows the validation error obtained.

|  |  |
| --- | --- |
| **MODELS** | **VALIDATION ERROR** |
| DECISION TREE | 0.07628095 |
| GENERALIZED LINEAR MODEL | 0.07066494 |
| XGBOOST | 0.06364874 |
| GRADIENT BOOST | 0.06383010 |

The MSE of decision tree is highest, followed by Generalized Linear Model. The validation result of XGBoost and GBM (after tuning) are comparable.

1. **CONCLUSIONS AND DISCUSSIONS**

**Summary:**

We predict departure delays of NYC flights in year 2013. We perform various methods like data wrangling, EDA, modelling, tuning and comparison to get the results.

All models give good performance on validation set except for tree which gives quite large validation test error.

From results, we conclude boosting is done properly by selecting appropriate tuning parameters such as shrinkage parameter, the number of splits we want and the number of trees, then it can generalize really well and convert a weak learner to strong learner. Ensembling techniques are really well and tend to outperform a single learner which is prone to either overfitting or underfitting or generate thousands or hundreds of them, then combine them to produce a better and stronger model.

**Limitations**

Gradient Boosting generates one tree at a time. Hence it is slower than XGBoost or Random Forest and takes large computation time. Also, finding optimum parameter values of shrinkage, interaction\_depth and ntrees is also a challenge.

One of the methods that did no run on this dataset is Random forest when ntree=100 due to heavy computation. However, it ran when ntree=5. Thus, a large number of trees makes the algorithm slow for prediction.

**Future work:**

We can use XGBoost, which is both computationally fast and accurate and for default parameters give comparable results as GBM.

1. **References:**
2. An Introduction To Statistical Learning with Applications in R (ISLR Sixth Printing)
3. <https://www.wikipedia.org/>
4. <https://github.com/SFUStatgen/SFUStat452>