Taxi data prediction(time series) with Recurrent Neural Network vs General Approaching Ensemble method

Data set:https://www.kaggle.com/c/pkdd-15-taxi-trip-time-prediction-ii

I (Chan Yao Chun/詹耀鈞) work with 徐煒員(Assistant researcher of NTU computer science department) and 江威庭(NTUST student)

- **Data handling abstract:**

The data set is 1GB to 2GB, but the space is mainly occupied by POLYLINE attribute, so we make the strategy that we select which data we want to use by month or hour, after that, we put the data set, especially whole the POLYLINE, which we will not used back to the disk for saving the memory space. Thus, the data in the memory will be less than 100 or 200MB. In addition, we do not need to transfer the POLYLINE to json format of whole the data, so it will be a lot of more efficient.

The data is a real world record for the taxi information, it provides some basic information like time and the point (longitude and latitude) in the trajectory, the point is recorded by GPS every 15 second, and the taxi id, etc.

- **Data exploration:**

The critical thing in the data exploration is that we find the testing data's distribution only contain a part of distribution, like the hour only from 2~17 in a day, also some of the larger value or rough speed bigger than 160km/hr is not in the testing data, and the position's latitude and longitude also have same result. Thus, we make some decision in our feature engineering process.

There are some interesting things, like one of drivers nearly dominate the number of trip, it has 29 trip in a day. Also, some of the position is popular for calling the taxi, like the train station, national theater, etc.

- **Main part of feature engineering:**

We mainly extend from the homework of one member(Chan Yao Chun/詹耀鈞), but using more sophisticated method. Also, we restrict the data's feature range. This is because, we find the evidence that the important feature, even target value, is more likely have outlier in it. Taking the travel time for an example, the travel time computed by the points number has many outlier. Moreover, some of important features, like start point's latitude, start point's longitude, etc, are shift during different month. Thus, we decide to clean out the data and select some useful feature. Finally, we decide to:

1. restrict the position in the range which is almost same in the testing data
2. only using the hour present in the testing data.
3. drop out the Original Call, because it has many different values, up to 6000, however in testing data set, the number of different value is less than 100, and it has more than 2/3 missing values, also in the feature importance plot it has low importance. Therefore, we drop it.
4. In some kind of model, like RNN, we found that if we do not use Original Stand, the performance will be better, no matter we use dummy variable or other method. Since that, we drop it.
5. Absolute difference between start point and end point's latitude/ longitude.
6. Add the distance between start point/end point and the central point of the city.
7. Add the features indicate that the trip is pass through the main train station or not.
8. Add the features indicate that the trip is pass through the airport or not.

9. Add the features indicate that the trip is pass through some hot spot(for tourist) or not.

ps. in 7, 8,9 the error is within 100 meter to the longitude and latitude of the point on google map.

- **How to validation:**

In the common part of our model's validation, we choose to use walk forward validation. That is, sort the data by time first and use the history data to predict future data. The reason we choose this method is because we think the except the geographical features, some of time related feature is the importance feature for model to learn, even in the Decision tree's feature importance plot can prove this ,so base on time will let the validation be more robust, also it's quite intuitive to use historical data to predict future result. Base on these reasons, the walk forward validation can help us see the model perform bad or good on which time.

The detail is that we make every months to be a time frame and choose two months to train and predict the next one month, but how many months use for training is base on each members choice, because we do not have same computation device, also the model is lot of different between each other.

- **RNN model:**

As the plot shows (**Figure pre-1)** , there is some outlier which may make the model recognize it as a trend that the travel time will be bigger, so after the data cleaning, the RNN model become more stable.

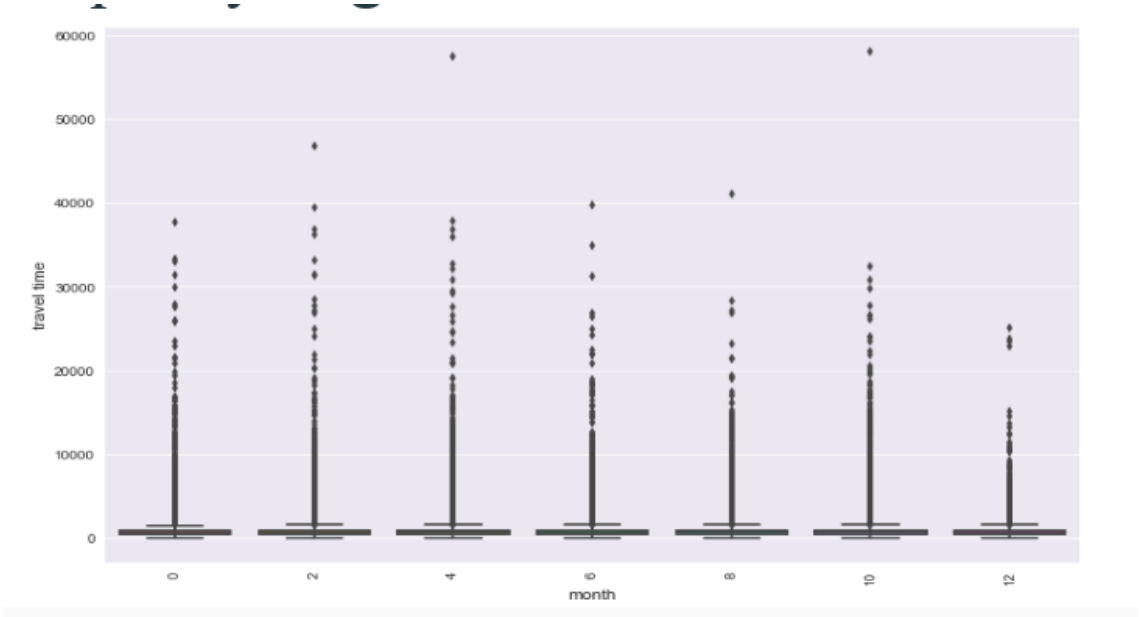


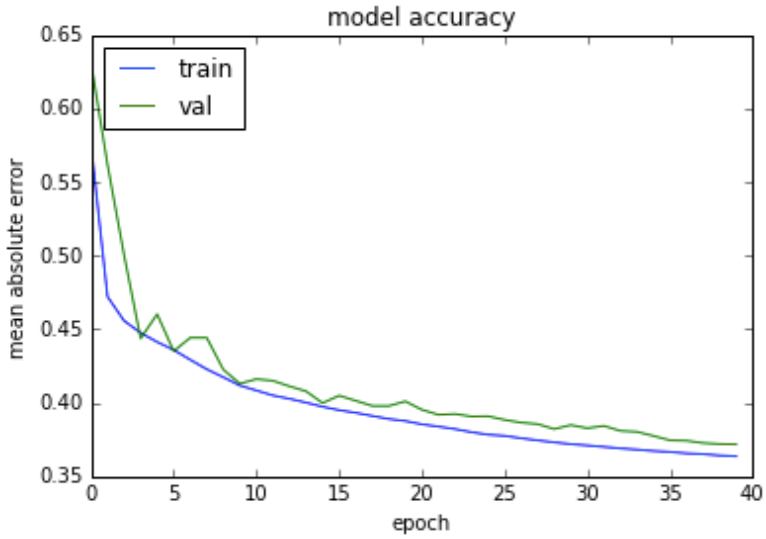
Figure pre-1. The target value distribution for each month used for validation.

The part of RNN model selection and validation, we extend from the homework3(Chan Yao Chun/ 詹耀鈞), the basic two layers GRU, but the result at the first is not meet our expectation. Thus, we decide to make the new structure of the RNN model, and we try to use bidirectional GRU and LSTM for building the best RNN model structure. After some experiments, we found that compare to the original two layer stacking GRU, the bidirectional RNN model is really stable and outstanding performance, according some reference materials, the bidirectional RNN have the ability to get the future and pass information, actually is positive and negative time direction of the data [**reference** 1 and 2].

Moreover, we use different normalized method to see which is suitable for the model. Finally we establish the best RNN mode : LSTM of 128 X 128 X 64 X64 X32 (the hidden layer's number of node, from first hidden layer to fifth hidden layer), the last layer is a linear function artificial neuron to averaging or say summarize the output of bidirectional LSTM , and we combine it with standardized scaler. It also has significant improvement compare to original L2 normalizer. Why standard scaler work out? we think that maybe it is because the data's distribution is very similar to Gaussian distribution after the data cleaning, but the value is not small to the model, so replace the mean with 0 and scaling data with their standard deviation improve the performance, also the target value is already transformed to Gaussian distribution (From the homework3 of Chan Yao Chun/詹耀鈞). Yet, using robust scaler can also get a lot improvement, though it can not beat the standard scaler, and its std is larger than any normalizer, it means that robust scaler may break some of the distribution of the data after we do the data cleaning.

For more specific of our model selection process, we use some tips in the Work forward validation to save time of training RNN model. First , using half of data, odd months in the data (if it's counted from zero, then the data represent to be even months), totally seven months for validation. Second, we set the epochs to be 40 but with early stopping mechanism, so the model will stop training if the performance not been improved in the limited time(we set to be 5), and it saves a little time for the model selection, at the same time, we can be more likely to see the real power of good RNN model. Another thing needs to be mentioned is that the bidirectional model can really "survive" at the early stopping mechanism ,and training again and again, keep improve in few steps even when the model is overfitting **(Figure 1)**, but it spends more time than common RNN model.

Yet, compare to the original GRU unit model, the LSTM unit is slower, because it has more number of gates than GRU(2 vs 3), also it is more sensitive to the data than GRU. Also, the bidirectional structure needs double number of nodes, one for negative direction and other for positive direction ,so the trade off of the model is a lot of time!!



At the final training, we did not use the huge data set to feed RNN model. Actually, one of our member write a code to create sampling machine [Appendix c. 1], and it can produce the data similar to testing data according to the time and location.

Figure 1 . The learning curve of bidirectional RNN when it's overfitting, reaches to nearly full times of epochs (we set it to be 40).

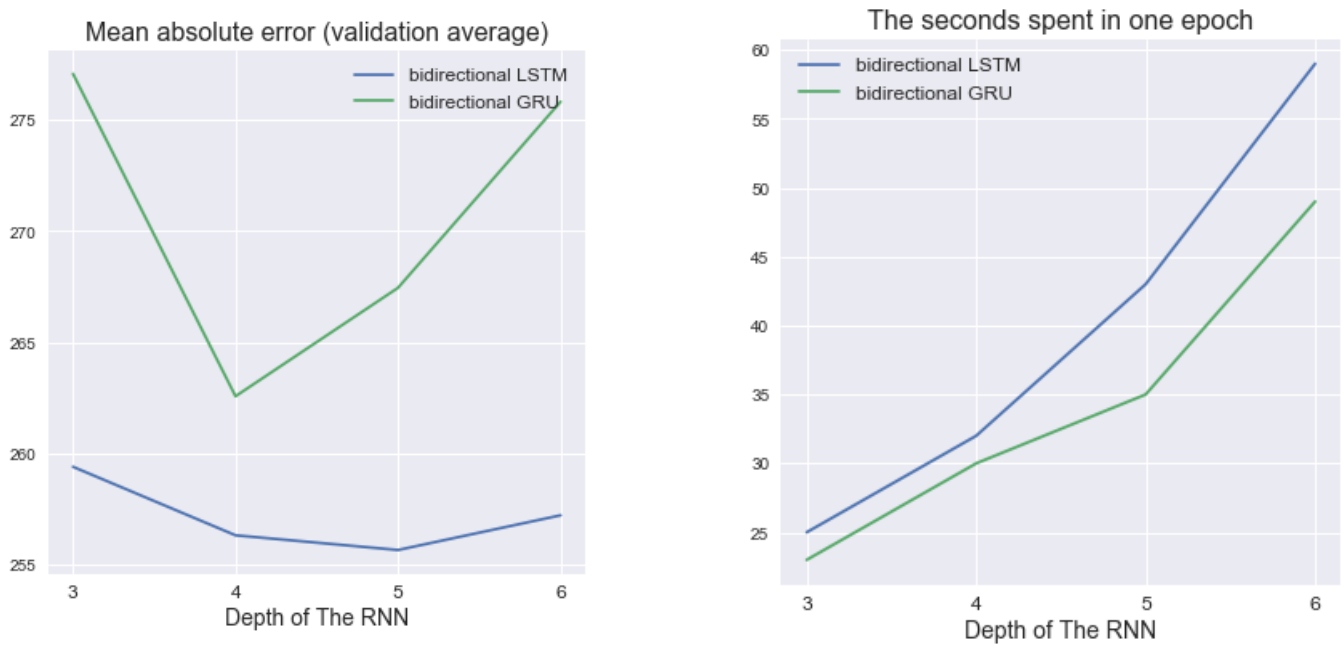


Figure 1-2 . The comparison plot for MAE and the time of one epoch between different depth of the bidirectional LSTM/GRU model .

- **Radom forest ensemble model:**

a.Decision tree

Random forest ensemble the "strong" decision tree model, so we plan to find the best decision tree model at first. After some of validations to find the best decision tree, we set max_depth=15, min_samples_split=0.001, min_samples_leaf=10, and limited those three parameters because we are aware of the overfitting problem, an the result is 0.52787.

There is some conclusions about those results, first, Max_depth larger than 15 didn't influence the result much although the tree depth is 29. This means that we do not have to let out tree grow too deep. We only need the appropriate tree depth, then we can get good result. Second, Min_samples_leaf and min_samples_split limit the tree in some common aspects. However, in our experiments, Min_samples_leaf didn't influence the result much because min_samples_split took charge of the model.

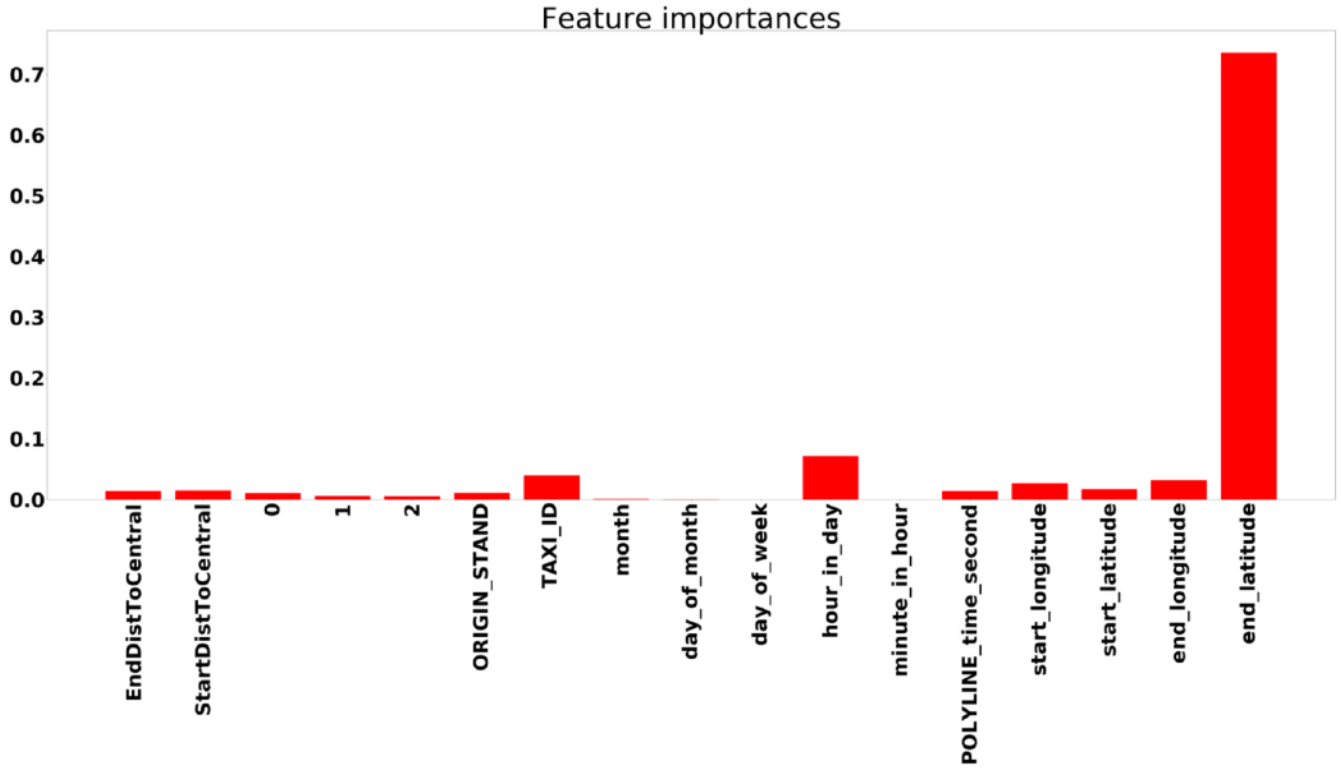
For more of data exploration, we make some plot of decision tree **(Figure 2 and 3)** for helping us .We can found out that "end_latitude" is the most important. And "hour_In_day" and "TAXI_ID" are also important. Also, the distance between the start point and end point is the critical feature for decision tree to make the decision stump. On the other hand, we decide to drop out "ORIGIN_CALL" because of the low importance, many missing value and big different between testing and trying data. In contrast, " ORIGIN_STAND" is included in tree plot, so in tree base model we still use this feature.

Further, in walk forward validation, we discovered that in month=13 (2014/1), the error is the highest. We think that it is probably because the activity of celebrating New Year's Eve. People during the New Year's Eve perhaps have trips different than trips in their ordinary lives.

b.start to ensemble

In the end, we start to ensemble the best Decision tree to be the Random forest model. We started to try different number of estimator, and set oob_score=True to let the model see the out of bag samples. Therefore, the ensemble model could be more robust.

In our experiments, we got that the more number of estimators(tree), the better the result will it be, but the trade off is larger number of estimators is that it took a much longer time, also the computation power would be a problem.



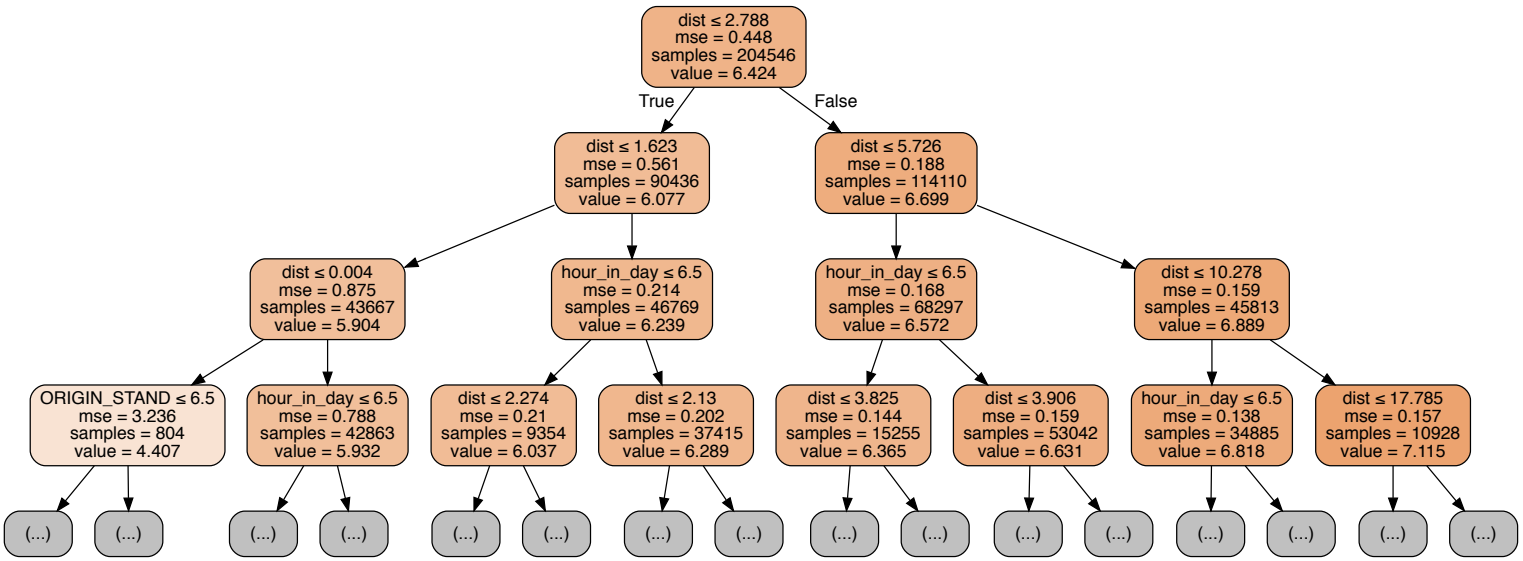


Figure 2 and 3 . The feature importance (Gini importance) plot of decision tree and Decision tree plot [**reference** 3]

- **Gradient Boosting:**

Gradient Boosting aggregated several weak decision tree model into a forest-like model, and this kind of model can boost the performance of weak regressor. Also, it is hard to find the suitable "weak" decision tree. Thus, we start to train the model directly.As usual, the max depths of decision tree is a principle parameter, it control the maximum depth of the tree. The deeper the decision tree is, the better the accuracy is. Yet, it's more easily that the model overfit on training data set, and we choose 9 max depth of decision tree to be the best model, with 100 estimators.

The validation method here is also as previous method, and the figure (**Figure 4)** below are the result of MAE value with 4 different depth and max depth with 9 is the best model, but in Dec the model of depth 9 is slightly weaker than the model of depth 7. The reason is that the extra branching of decision tree is not improved for the data of month=12 (2014/6), also some of overfitting problem will happen.

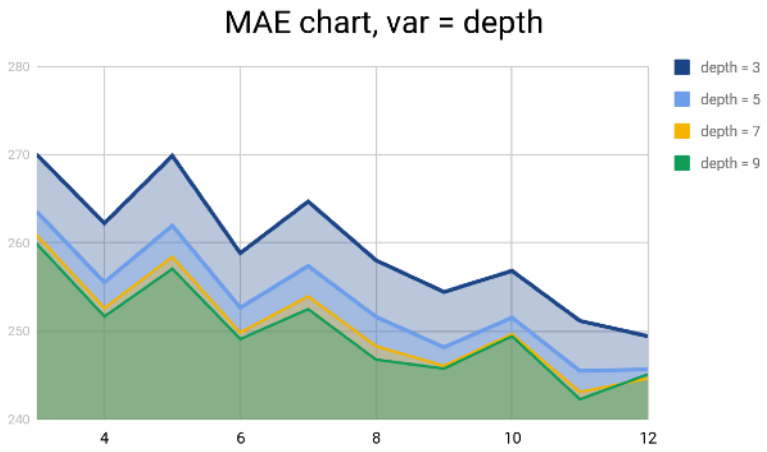


Figure 4 . Gradient boosting different max depth and its MAE value, **x axis** is which month (we transfer it to be 0~12, in real world is 2013/7 ~ 2014/6)

- **Compare to state of art and result discussion:**

In the part of feature engineering, we can combine the first one solution's feature to improve the performance, the first one winner using a lot of special features like the angle of position or some point's data of the last part of trajectory. On the contrary, we take a lot more part on model building and selection, and even using the RNN model that is rare in this kind of competition (most of it is used for NLP).

Also, we find that when we do the final testing to the Kaggle, the performance is influenced by the adding some bias. Furthermore, when we test the model, the result of actual MAE is not change a lot, but RMSLE is worse, it means that the model have "under estimate" problem. That is, the testing data "deliberately" put some large travel time data, and it make the model be punishing by these value when we evaluate the data by RMSLE criterion. Thus, one challenge for the model is that the model needs to precisely "recognize" the sample with large travel time and do the right prediction, not just predict the model with most frequent value.

The one other improvement is that we do is add constant positive bias value because the test data seems that missing several GPS points, and the actual time is greater than the estimation value of our model. Plus, it really makes sense that the extra time should be added to the travel time, because the GPS is likely to lose some record of the whole path, and the taxi may have some situation make

the travel time to be longer(customers spend time to find coins to pay the fee, etc). This bias evidence is just as above observation, though the inference is different.

- **Final comparison:**

The Deep Neural network related model is time consumed, but if we use it at the right place and "right kind" of structure the power of model is very significant, as we see the models without bias. However, the Random Forest and Gradient boosting may not beat it, but actually the result is very similar, also they do not need to take a lot of time to find out the structure that fit the expectation, simply tune the parameters can see the power. Thus, both of them save a lot of time.

Appendix:

a.The performance on Kaggle leaderboard, RMSLE:

(1).Bidirectional LSTM model:
private:0.75536

(2).Bidirectional LSTM model with bias:
private:0.68709

(3).Random forest:
private:0.79683

(4)Gradient Boosting:
private: 0.98003

(5)Gradient Boosting with bias:
private:0.66368

(6)Just counting the point in testing data:
private:1.30529 **It shows that the data is missing

b.Reference:

1. Bidirectional RNN, Mike Schuster and Kuldip K. Paliwal, Member, IEEE,1997
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.331.9441&rep=rep1&type=pdf
2. Stanford CS224D Deep Learning for NLP, class notes on Spring 2015
https://cs224d.stanford.edu/lecture_notes/LectureNotes4.pdf
3. Random forest detail explanation and Gini importance(Berkely Breimen)
https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#features
4.Taxi trip prediction first one solution
https://github.com/hochthom/kaggle-taxi-ii/blob/master/doc/Taxi_II_Winning_Solution.pdf

c. Some of code(provided by Chan Yao Chun/詹耀鈞):

1.Sampling machine to training the RNN at final **&&** **2.**RNN model and validation code
https://github.com/JasonEricZhan/Machine-learning-kaggle-/tree/master