Big Data Project: Prediction of Music Relistens

Jihoon Bae, Seunggeun Baek, Yujin Jang, Junhu Kang, and Nawon Kim

December 18, 2017

Abstract. In this report we apply three different models (Gradient Boosting, Neural Network, and Random Forest) on a prediction problem related to music relistens. We observe that Gradient Boosting provides the most accurate model among the three on this problem, and random forest shows the lowest overfitting rates. Neural network is the least accurate model as the categorical data is the predominant feature type of this problem. We edit and introduce features in order to acquire better accuracy. We observe the effects from changes of model hyperparameters and the network structures in case of the neural networks.

Keywords. Gradient boosting. Neural network. Random forest. Feature engineering.

1 Introduction

One of the major recent progresses of the computer science is the machine learning on big data. It is deeply related to the real life, and they have synergized with the applications of the artificial intelligence. Manipulation and analysis on the large amount of data using the machine learning algorithms allow people approach to problems which are difficult using legacy methods.

The rapid development of machine learning technology have attracted the managers in the business area. Companies can enhance their working process and analyze the behavior patterns of their customers using data they collected. From the data could have been treated meaninglessly, companies can profit by discovering its hidden values.

Our team tried to find a problem instance regarding to behavioral pattern of customers, to stand at the companies' point of view. We joined to the competition on Kaggle hosted by KKBox(a Taiwanese music hosting company), which providing enough dataset related to the customers.

1.1 Aim of the project

The information of the members, the songs, and the listens(as the train and the test data) are given in the dataset. We create a prediction model determining whether the listener would listen that song again within a month, from the data of listens. The prediction model can estimate the preference over new songs or artists, eventually provide recommendations to new users. We use three different models (Gradient Boosting, Neural Network, and Random Forest) for the prediction and compared their performances.

1.2 Applications on business area

For companies, customer specific solutions plays a core role to manage relationship with the customers, eventually leading to an increase of the profit.

In this project, predicting the probability of relistens leads to the extraction of individual preferences. The model also shows the trends of customers listening certain songs. This encourages the company enhance their music recommendation system and customized service plans. Through the model, the company can understand their customers and deploy strategic plans even for their latent customers.

2 Data analysis

A part of relational database on the company is given as the data to be analyzed. The CSV-encoded data files are provided for each relevant tables and columns in the database.

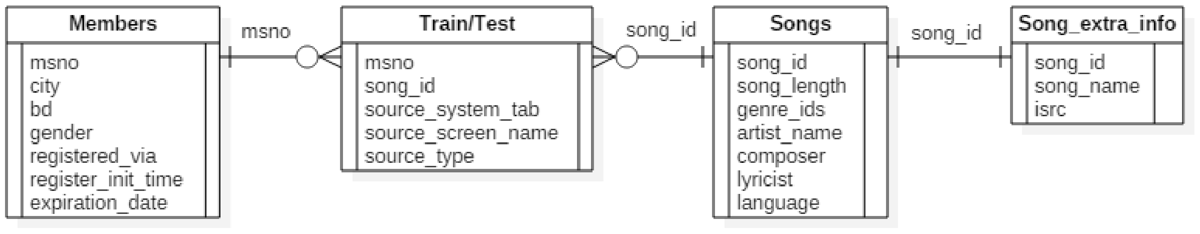


Figure 1: Entity-relation diagram of the given data

Data	Meaning	Format
msno	Member ID	Hashed Value
city	City category	Integer/Nominal
bd	Age, in years	Integer
gender	Gender	Binary
registered_via	Method of registration	Integer/Nominal
registered_init_time	Membership registration date	Date
expiration_date	Membership expiration date	Date
song_id	Song ID	Hashed Value
song_length	Length of the song, in milliseconds	Integer
genre_ids	Genre categories	(Integer/Nominal)*
artist_name	Name of the artists	(String)*
composer	Name of the composers	(String)*
lyricist	Name of the lyricists	(String)*
language	Dominant language	Integer/Nominal
song_name	Title of the song	String
isrc	ISRC serial number	String
source_system_tab	Tab name which the song was played	String/Nominal
source_screen_name	Screen name which the song was played	String/Nominal
source_type	Menu which the song was played	String/Nominal

Table 1: List of the features provided in the dataset

The dataset have several properties important for our analysis.

Categorical data. Most features of the data are categorical even they are represented as numbers. Specifically, they are nominal features rather than ordinal, as the numbers have no intrinsic orders.

Multiple values. Some features including the genre and the artists have multiple values separated by bars. This indicates that the original database is a denormalized database from the 1NF.

Incomplete member data. Member data including the gender, the age, and the city have much incomplete data. Especially, approximately 42% of the members did not provide their age information; their ages are represented as 0.

3 Neural network

Artificial Neural Network(NN) is a re-arising method of machine learning, which mimics the activity on brain cells (neurons). If the states of the input neurons x_i is given, the output is given by

$$output = f\left(\Sigma\left(w_i x_i\right) + b\right)$$

where w, b, f represents weights, bias, and an activation function, respectively. Weights and biases would be randomly set first, and improved throughout the training, by manually calculating back-propagation formula or just running some optimizations. Such neurons are linked to build a model; layers (groups of neurons) are used for the simplicity. Recently researchers developed some variants of NNs including Convolutional NN for image recognition and Recurrent NN for serial data analysis. However, as these models assume spatial and temporal locality of input layers, they are not appropriate for this project as the project analyzes separated features. In this project, we used layers of dense networks mainly. Some python libraries including Tensorflow and Theano aim to support neural networks. We used GPU implementation of Tensorflow for this project.

3.1 Categorical data preprocessing

Neurons only accept numbers as the level of activation, making the NN rather ineffective to deal with categorical data. To use NN with nominal data, we could have following approaches to convert them into numbers.

Binary feature. As the gender is a binary feature, it can be directly processed such as (male=1, female=0). Nulls can be mapped into 0.5, or the portion of males from gender-known members.

Dummy variables. Dummy variable(or indicator variable) is a binary variable having 1 or 0, representing whether some condition is met or not. To encode the categorical feature into dummy variables, one can introduce columns for each source categories, and encodes the value as 0 or 1 depend on whether the column indicates the value of source category.

ID	Genre
1	Pop
2	Rock
3	Pop | Electronic
4	(null)

→

ID	Genre_Pop	Genre_Rock	Genre_Electronic
1	1	0	0
2	0	1	0
3	1	0	1
4	0	0	0

Table 2: Example of a dummy variable conversion

The details how to deal with nulls(ID 4) and multiple values(ID 3) may vary by methods. If exactly one 1 exists in every records because of absence of null or multiple values, the table is called a complete disjunctive table(CDT). Making dummy variables is still one of the simplest methods to encode categorical variables and becomes the base of other categorical data processing methods. Large number of distinct categories may lead to the same number of features; one can use some workarounds such as retaining 'Top n' categories.

Multiple Correspondence Analysis(MCA). For multiple related categorical variables, MCA converts them to tuples of numbers, by applying PCA to their CDTs after having some transformations. If one visualizes the table using the numbers in the tuples as coordinates, he or she may notice some underlying structures and relationships between the categories.

ID	Source1	Source2	Source3
1	My Library	My Profile	My Profile more
2	Radio	Radio	Listen to
3	Playlist	Others Profile	Purchase

→

ID	F1	F2
1	0.88	0.79
2	0.22	0.17
3	0.65	0.23

Table 3: Example of a MCA conversion

Application. Note that nominal features other than sources and genres are not used in this analysis. The features include city, registered_via, language, and artists; we expect the performance rise is negligible if we add these features by converting them into numbers. We only created dummy variables for Top 32 genres and all source features to use in NNs.

3.2 Data transformation and normalization

Experts say that "In addition to the necessity of encoding categorical data, experience has shown that neural network training is usually more efficient when numeric x-data are scaled, or normalized, so that their magnitudes are relatively similar."[1] We applied some nonlinear transformations to reflect the property of data or remove skewness on the data, and adjusted the data to have similar scale (between 0 and 1).

Normalizer. Normalizer generates a nonlinear transformation mapping min_value to 0 and max_value to 1. The function should be an increasing function to obtain meaningful results.

```
def trimmer(x):
        return 0 if x<0 else 1 if x>1 else x
def normalizer(min_value, max_value, func, positive=False, unknown=0.5):
        return lambda x: unknown if math.isnan(float(x)) or (positive and x<=0) \
                else trimmer((func(float(x))-func(max_value))/(func(min_value)-func(max_value)))
```

Features	func	min_value	max_value	Notes
bd (age, yr)	Sqrt	9	49	Adjusting the mode(25) into 0.5
song_length (ms)	Log	32768	1048576	
member_duration (s)	Log	1	444873600	reg_init_time-expiration_date

Table 4: Code and List of the features preprocessed with the normalizer

Decayer. Decayer generates an exponential decay transformation when the current point(which mapped to 1) and the decaying rate(as the half-life) are given.

```
def decayer(current, halflife, unknown=0.5):
        return lambda x: unknown if math.isnan(float(x)) else 1 if float(x)>current \
                else math.pow(0.5,(current-float(x))/halflife)
```

Features	current	halflife	Notes
song_year (yr)	2017	10	Data from isrc
registration_init_time (s)	1488258000	100000000	current: Feb 28, 2017

Table 5: Code and List of the features preprocessed with the decayer

For example, as recent songs have more listens than older ones and the popularity of songs usually have exponential decay, it is reasonable that we preprocess the release years of the songs using a decaying function (we chose half-life 10 years in this case). The decaying function relatively spreads out the recent song, making the recent songs more distinguishable.

3.3 Network design

To examine effects of network modification, we set the different configurations of the networks. The baseline network is a 3-layered ReLU network with no dropouts. We changed the number of the neurons and the number of layers, changed the dropout rates, used the logistic function as the activation function, and split the inputs to follow different path on the network.

Aside from the shape of the networks, we modified iteration number of trains. The baseline network uses 50000 steps; we also tried 30000(Train-) and 100000(Train+) steps of trains.

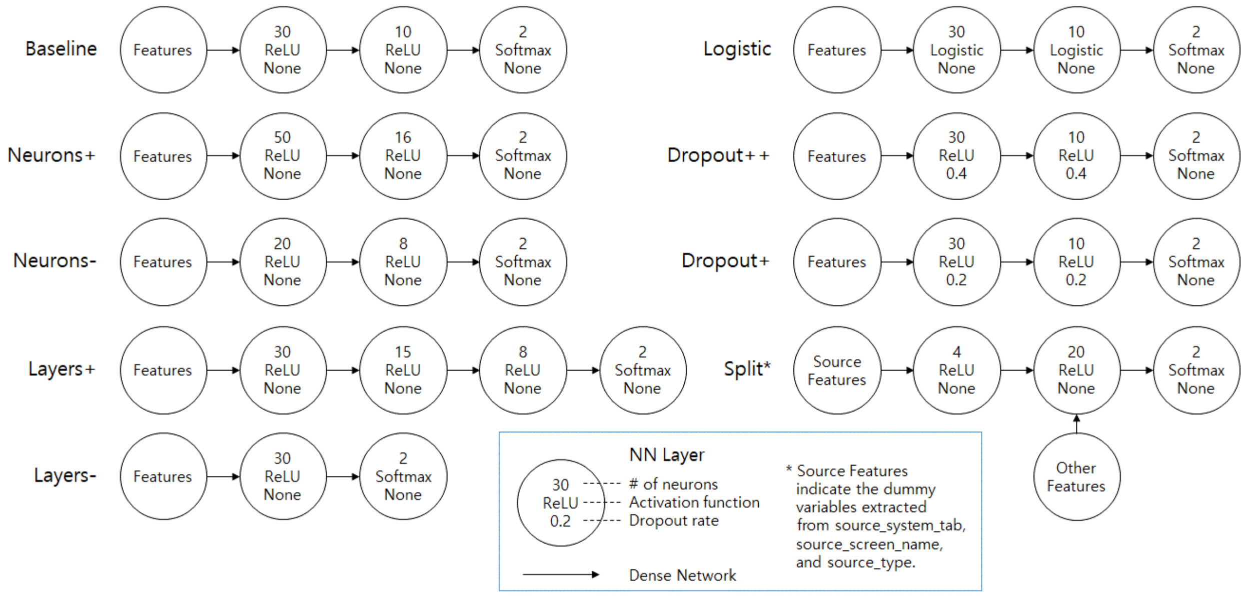
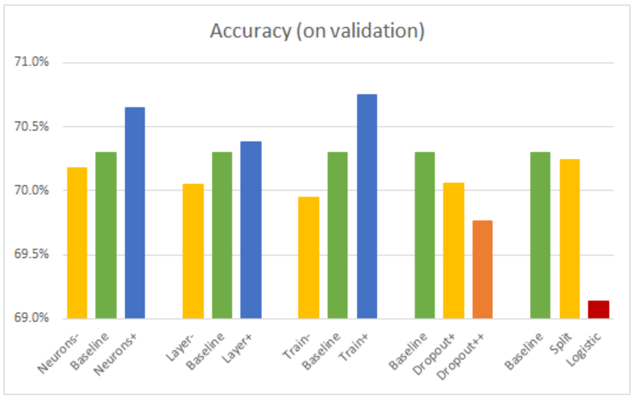


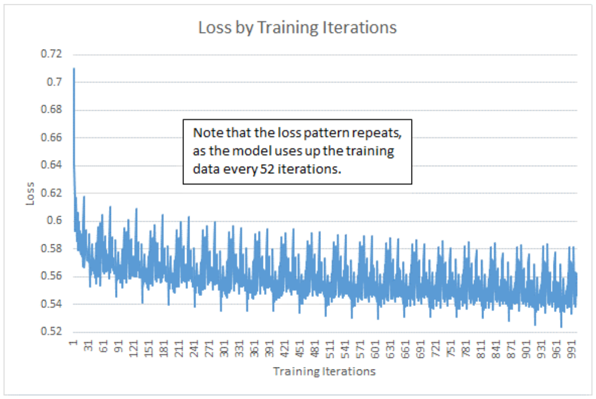
Figure 2: Network structures

3.4 Results

As we have expected from ineffectiveness of treating nominal data by NN, the general results are around 70% validation accuracy, which is far more inaccurate than the results of other models.



(a) Network performances in validation accuracy



(b) Loss improvement

Figure 3: Results of NN models

Still, we could observe tendency of accuracy by the model parameters including number of neurons and the number of layers. While addition of the neurons, the layers, or the training iterations improves the model as expected, the dropout technique seems to prevent the models to improve. When we split data and pass it to different locations of the network separately, the results are better than Neurons-(28 neurons) and Layers-(30 neurons) models while using only 24 neurons.

It is normal that the score from Kaggle uploads are lower than local validation. The difference roughly indicates how much the model overfits the data. Upload of results from Neurons+ model yields 62.172% score. Compared to validation accuracy 70.652%, the difference is roughly 8.5%p. Observing the losses during the iterations, we could see decrease of the losses like the following diagram of Train+ run.

3.5 Aggregation of multiple networks by voting

Voting is one of the simplest ensemble methods, letting multiple models to have multiple predictions and choosing the result which most of the models predicted. The voting procedure is more effective if the models have similar structures and produce different predictions due to their different parameters or initial conditions.[2]

We applied voting to 5 executions of single model (Baseline), and another 5 executions of different models (2 Baselines, Neurons+, Layers+, Split). It turns out that even the different networks produce similar predictions, observing that 0- and 5-voted records are dominant. We have 61.262% submission score for single model voting, and 61.983% for different model voting, which is not very different from the original baseline model.

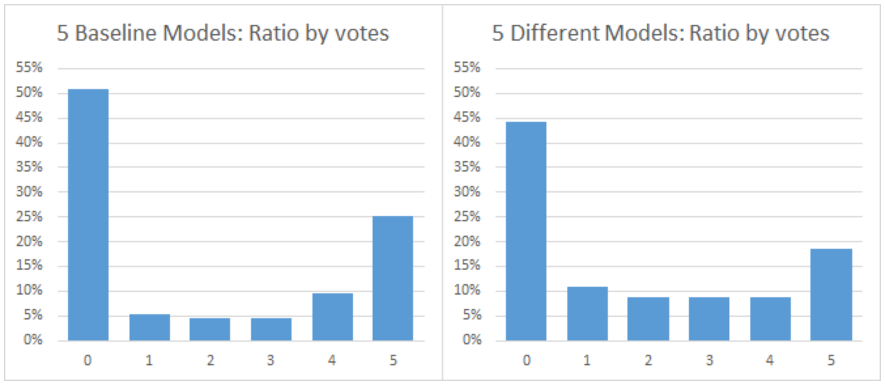
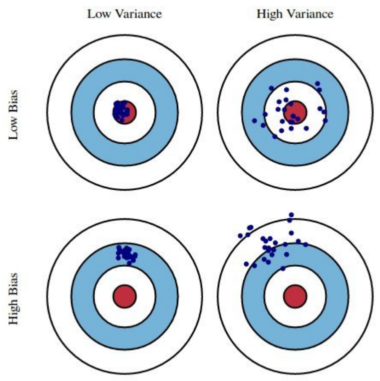


Figure 4: Distribution of vote results, indicating the number of models suggesting the target is 1

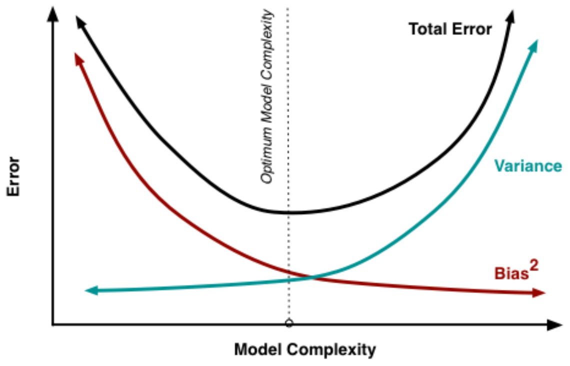
4 Random forest

4.1 Bagging

Random forest is a machine learning algorithm utilizing bagging(bootstrap aggregating) approach. Bagging is an ensemble method to have votes from individual models, which uses train data randomly sampled from the whole train dataset. Two measures, the bias and the variance, have tradeoff relationship despite they are together the cause of the learning error of a problem. Bagging can provide an ensemble of models that lowers both measures. To have an effective result, the underlying models should have high predictability when the stability is not concerned at all.



(a) Diagram of bias and variance[3, Fig.1]



(b) Bias and variance contributing to total error[3, Fig.6]

Figure 5: Bias-variance tradeoff[3]

4.2 Decision tree, as the underlying model

A random forest uses decision trees as the underlying model of bagging. Decision trees are quite unstable classifiers, but they can effectively deal with categorical data. Decision trees become less effective if too many features are given; this is because only one feature should be selected at the each node and the selected feature is more probably not the best feature. Fortunately, the effect on the number of features is quite negligible for less than 100 features.

As a random forest behaves like a black box, the flow of logic could be unexplanatory for humans different from the individual decision tree. However, it is easy to use because only 3 major hyperparameters are required. The number of decision trees, the depth of the trees, and the sampling rate for bagging should be provided.

4.3 Results

The random forests have around 70% validation accuracy, which is similar to the neural networks. However, the properties of bagging can lead the model to have less overfitting. The prediction accuracy is around 65%, which provides accuracy difference near 5%p which is the smallest among the models we used.

5 Gradient boosting

A gradient boosting model(GBM) is a group of gradient descent models ensembled with boosting method. Boosting is a serial ensemble method which gradually increases weight of misclassified data. Originally the gradient descent models modifies parameters by calculating the gradient of the loss function through partial derivation, multiplied by the specified learning rate α.

$$\mathbf{v}_{new} = \mathbf{v}_{old} - \alpha \bigtriangledown Loss\left(\mathbf{v}\right)|_{\mathbf{v}_{old}}$$

GBM applies the gradient of the loss function for the whole parameters used in partially aggregated model learned so far. GBM requires the learning rate, the loss function, and the parameters for the boosting(such as the number of models) as its hyperparameters.

5.1 Data preprocessing and feature engineering

For the null or invalid values, they are replaced into average value of the feature in case of numeric features(including song_length), or a category indicating the absence of the values in case of categorical feature. The data types of each of the feature columns are stated explicitly.

Feature engineering is a preprocessing procedure which extracts new features can be considered meaningful, from the existing features. We extracted some temporal aspects of the features separately, such as year/month/day of date features and release year from the ISRC codes. The numbers of song plays by songs or artists are added. The length of the songs are categorized by equally ranged group using the half of the average length as the size of the ranges.

Steps	Introduced Feature	Prediction accuracy
1	Using features only occuring in the provided data	62.0%
2	Stating the explicit types of the columns	62.8%
3	Extracting release years of the songs from ISRC	64.2%
4	Separating dates into year, month, and day	65.6%
5	Adding the number of artists, composers, and lyricists	66.8%
6	Categorizing the song length	67.6%
7	Adding the numbers of song plays by songs	68.3%
8	Adding the numbers of song plays by artists	68.7%

Table 6: List of the features provided in the dataset

5.2 Results

We used LightGBM library provided by Microsoft. As we introduced features from the feature engineering procedures, the prediction accuracy continues to improve if the features are effective.

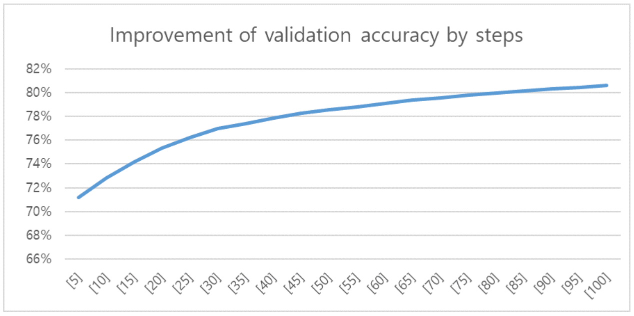


Figure 6: Improvement of validation accuracy throughout the training

The final model(learning rate 0.2, 100 boostings) produces 68.71% prediction accuracy, and its corresponding validation accuracy is 80.6% for the best model. Though the difference(12%p) which is the largest among the models shows large degree of overfitting, the prediction accuracy is the best result we could obtain so far. Adding more features lead to the memory overflow error in the laptops, limiting the number of features we could actually use.

6 Conclusion

We attempted to produce a solution for prediction problem using the three(Random Forest, Gradient Boostring, Neural Network) machine learning models. We could have following accuracy results.

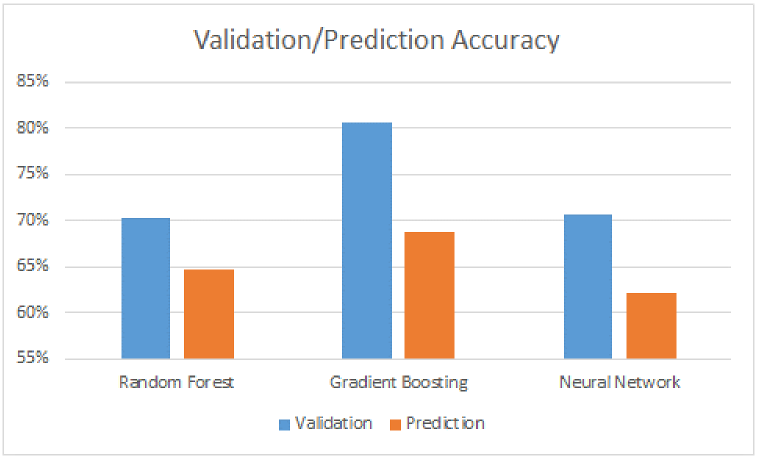


Figure 7: Validation and prediction accuracy of each models

Comparing these models one another, GBM has the best prediction accuracy. Random forest shows the lowest overfitting(difference of the two accuracy values) rates. Neural Network is the least accurate model as the categorical data is the predominant feature type of this problem.

All members in our team started the project without the background knowledge about machine learning. Though it is not the best result in the competition, this was a fine opportunity to deal with various kinds of data and models. We learned the underlying theories and details of models, and got intuition about data relationship and preferred analysis method by trials and errors.

It is interesting that we analyzed the data used in actual business area, while maintaining competitive relationship with the other teams in the competition. This kind of experience would be a substantial help when we advance into the actual business area in the future.

Appendix A Project timetable

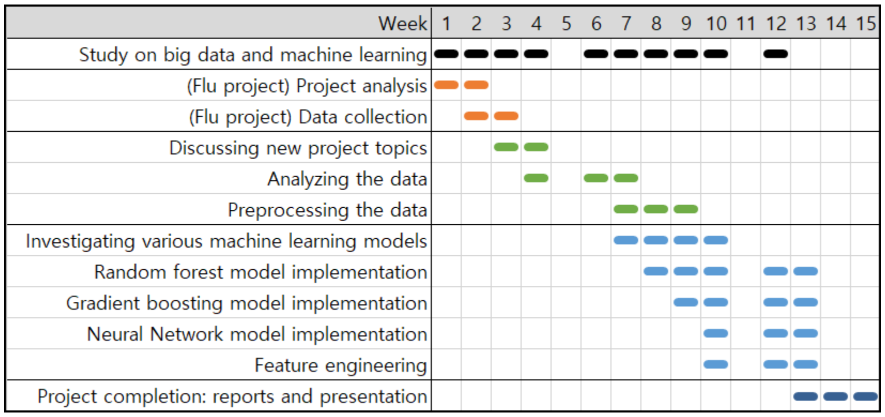


Figure 8: Gantt chart of the works, in weekly basis

Appendix B Work contribution

Our team has a democratic team structure providing equal relationship among the five teammates. The team is driven by ourselves; we decided our own goals under some advice from Assoc. Prof. John A. Springer. Though we had no firm work distributions except for each models, we had helped one another through suggestions based on goals we discussed in each weeks. The following describes the major contributions of the each team members.

J.Bae and J.Kang together concentrated on Gradient boosting and feature engineering, producing the best result among our trials. S.Baek focused on Neural networks while also concerning team management and some data preprocessing. Y.Jang aimed on Random forests and tried to seek and crawl external relevant data earlier in the project. N.Kim played the key role on designs of the documents and the presentation, also putting her effort on literature search.

The final presentation is mainly done by J.Kang and N.Kim. J.Bae, S.Baek, and Y.Jang contributed to the contents of the report. S.Baek wrote this report in English refining the original report in Korean.

Appendix C Notes on project topic change

The flu spreading model is a major research accomplishment of the project advisor, Prof. John A. Springer. The first topic we selected was to apply the flu spreading model, in Korea, using various sources of the data.

We had made attempt to collect the necessary data including weather data, medical data, and the social media data. Weather data could be easily acquired through Korea Meteorological Administration. For the social media data, the users rarely allow their location to be posted; some media do not provide their API public. Especially for the medical data, no suitable data is available to use. A cohort sample with a million anonymized patients is not only expensive, but also not a real-time data. Offices of education monitor the students with flu and upload their report online every week; it is the data with best temporal granularity, but the data is in document form and only students are monitored. There are no data sources with the spatial granularity finer than the province level.

Meanwhile, National Health Insurance Service(NHIS) in Korea already have prediction models for various diseases, provided to the public through 'Public health Alarm Service'. Though flu is not directly included in the service, the service predicts cold. As NHIS have its own real-time medical data we cannot obtain, it is capable to provide accurate predictions, different from our team. We decided to find a new project topics which data is available to use.

Acknowledgement. This project is a result of Korea Software Square 2017 Capstone 8 program organized by IITP and Purdue University. We appreciate to the project advisor Prof. John A. Springer and the program manager Prof. Eric T. Matson, making our projects to be more successful. We also appreciate to IITP and each universities the team members are belonging to, for the support and the funding to this program.

References

[1] J. McCaffrey, "How to standardize data for neural networks." https://visualstudiomagazine.com/articles/2014/01/01/how-to-standardize-data-for-neural-networks.aspx, Jan 2014. Accessed on 2017-12-02.

[2] E. Alpaydin, "Multiple neural networks and weighted voting," in *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pp. 29–32, IEEE, 1992.

[3] S. Fortmann-roe, "Understanding the bias-variance tradeoff." http://scott.fortmann-roe.com/docs/BiasVariance.html, Jun 2012. Accessed on 2017-12-10.

[4] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] R. C. Steorts, "Bagging and random forests." http://www2.stat.duke.edu/ rcs46/lectures_2015/random-forest/slides_lecture15.pdf, Mar 2014. Accessed on 2017-12-01.

[6] Microsoft Corporation, https://lightgbm.readthedocs.io/en/latest/, *LightGBM's documentations*, 2017.