TalkingData Mobile User Demographics

Qi Qi
CS:5430 Machine Learning
University of Iowa
qi-qi@uiowa.edu

Yujian Zhou
CS:5430 Machine Learning
University of Iowa
yujian-zhou@uiowa.edu

Qianhang Sun
CS:5430 Machine Learning
University of Iowa
qianhang-sun@uiowa.edu

Abstract

With one month hard work, we ranked 25th on the public leaderboard in the competition of "TalkingData Mobile User Demographics" held by kaggle. We add the experiments into project proposal to integration all the works we have done to finish the machine learning project.

1 Introduction

In this project, we are going to finish a Kaggle competition from scratch, called "TalkingData Mobile User Demographics". The goal of this competition is predicting the demographics of a user (gender and age) based on their app usage, geolocation, and mobile device properties to help millions of developers and brand advertisers around the world pursue data-driven marketing efforts which are relevant to their users and catered to their preferences. There two main aspects we are going to working on to kick the leaderboard: 1) Feature Engineering 2) Model Design. We introduce the existing methods both for feature selection and model building in Section 2 and our proposed methods will be analyzed in Section 3.

2 Existing Approaches

In feature engineering part, for one deviceid (key, the unique id of a mobile phone) may install lots of apps, and those apps belongs to different labels. We need to make use of the technique of bag-of-words technique in document representation, which means one app id(app label) occupy a feature space with one dimension for each term, and we call it bag-of-apps and bag-of-labels. It has two potential deficiency to our problems: One potential problem is that with the bag-of-words teachnique is that it does not explicitly express the similarity between related concepts. However, the category of apps already given, which can be read as the concept in concept of words[4]. PCA, low variance filter and high correlation filter are often used, but neither of can add the prior knowledge of the data[3].

XGBoost, a scalable machine learning system for tree boosting[] proposed by Tianqi Chen, which has ben widely recognized in machine learning competitions and data mining changllenges. A survey conducted on the the challenges hosted by the machine learning competition site Kaggle shows that among the 29 challenge winning solutions, 17 solutions used XGBoost. Even though the great success in Kaggle, it still exists a drawback that it can not make full use of all the useful information that contained in the feature space. This drawback is exacerbated especially when the feature space is extremely large. In this competition, our feature space is more than 20000 dimensions, which means the XGBoost itself can not perform well in this competition.

3 Proposed Algorithms

The evaluation metrics that used in our project is logistic loss. The better model we get, the lower logistic loss. So we are going to improve our models from three aspects. First of all, to make full use of the features, we will **design a problem-specified similarity by adding prior knowledge**. This methods has been mentioned in [5] . Secondly, the idea of ensemble learning is to employ multiple learners and combine their predictions, which has been widely used in machine learning model design. Adaboost and Xgboost[1] are two most wide used tree-based boosting models. However, both of them are model-based ensembles, and cannot overcome the deficiencies of only using part of the features. So we are going to **design a weighted-feature ensemble models based on the xgboost** to improve the performance of our model. Lastly, since deep learning show its dominating performance on image classification and text analysis recently[6], we will **design a deep neural network** to solve our problem, which also served as a comparison with our weighted-feature ensemble models. It is worth to mention that all of those improvements are based on the baseline model we design, which also time consuming.

4 Data Preparation

4.1 Data Description

According to `https://www.kaggle.com/c/talkingdata-mobile-user-demographics/data`, the data is collected from TalkingData SDK integrated within mobile apps TalkingData serves under the service term between TalkingData and mobile app developers. There are seven tables which can be connected using primary key, like device_id and event_id, in our dataset.

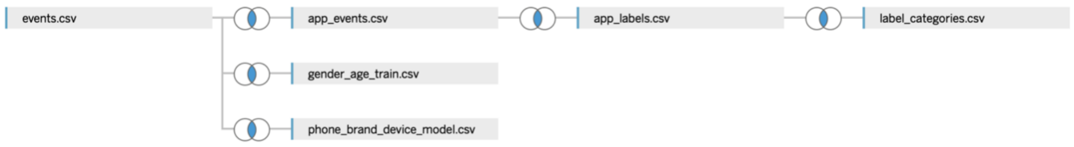
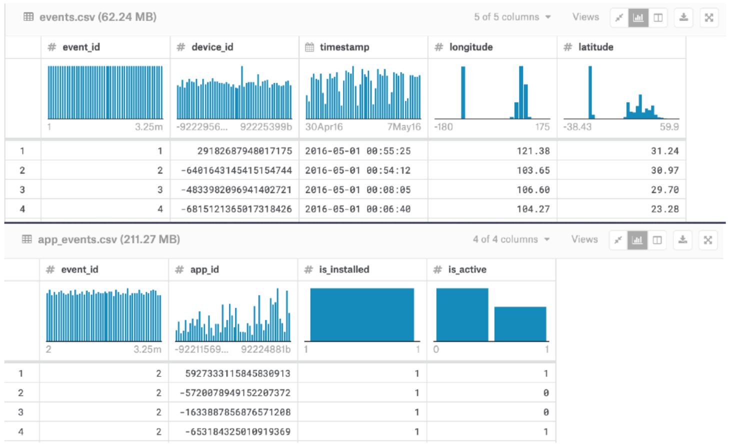


Figure 1: Table Connection

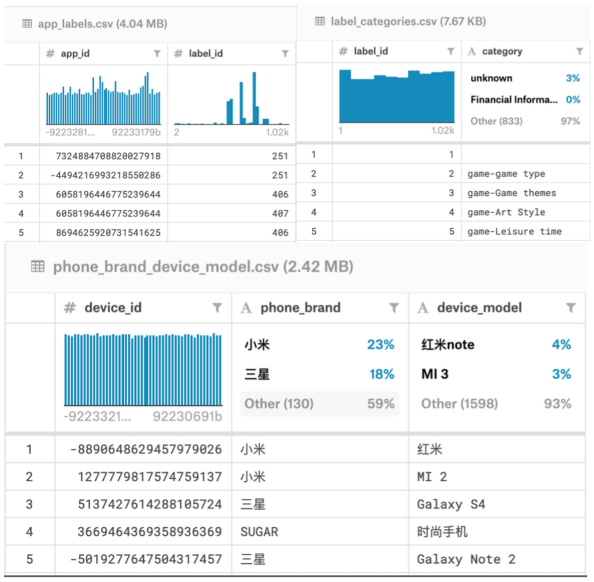
In Figure 2, Event.csv describes where (longitude and latitude) and when (timestamp) an event happens at what device (device_id). App_events.csv describes an app is whether installed and active or not in each event.



	# event_id	# device_id	timestamp	# longitude	# latitude
1	1	29182687948017175	2016-05-01 00:55:25	121.38	31.24
2	2	-6401643145415154744	2016-05-01 00:54:12	103.65	30.97
3	3	-4833982096941402721	2016-05-01 00:08:05	106.60	29.70
4	4	-6815121365017318426	2016-05-01 00:06:40	104.27	23.28

	# event_id	# app_id	# is_installed	# is_active
1	2	5927333115845830913	1	1
2	2	-5720078949152207372	1	0
3	2	-1633887856876571208	1	0
4	2	-653184325010919369	1	1

Figure 2: events and app_ events source file



app_labels.csv (4.04 MB)

	app_id	label_id
1	7324884708820027918	251
2	-4494216993218550286	251
3	6058196446775239644	406
4	6058196446775239644	407
5	8694625920731541625	406

label_categories.csv (7.67 KB)

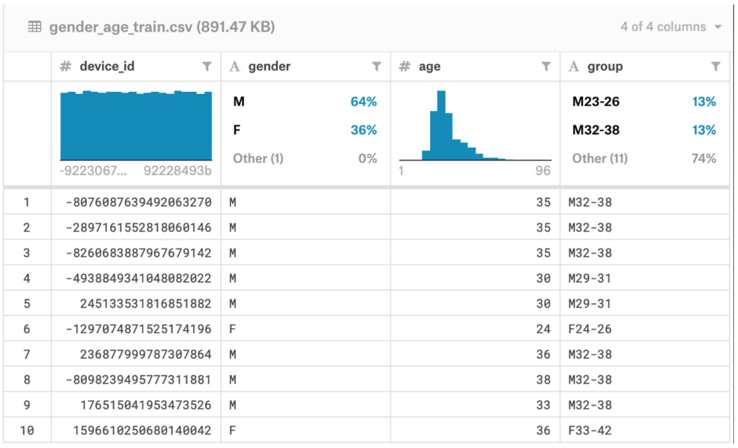
	label_id	category
		unknown 3%
		Financial Informa... 0%
		Other (833) 97%
1	1	
2	2	game-game type
3	3	game-Game themes
4	4	game-Art Style
5	5	game-Leisure time

phone_brand_device_model.csv (2.42 MB)

	device_id	phone_brand	device_model
		小米 23%	红米note 4%
		三星 18%	MI 3 3%
		Other (130) 59%	Other (1598) 93%
1	-8890648629457979026	小米	红米
2	1277779817574759137	小米	MI 2
3	5137427614288105724	三星	Galaxy S4
4	3669464369358936369	SUGAR	时尚手机
5	-5019277647504317457	三星	Galaxy Note 2

Figure 3: app_labels.csv, label_categories.csv and phone_brand_device_model.csv

In Figure 3, App_ labels.csv describes apps and their labels, whose id can be used to join with label_ categories. Label_ categories.csv describes apps' labels and their categories in text. Phone_ brand_ device_ model.csv describes the brand and models of a certain device.



gender_age_train.csv (891.47 KB)

	device_id	gender	age	group
		M 64%		M23-26 13%
		F 36%		M32-38 13%
		Other (1) 0%		Other (11) 74%
1	-8076087639492063270	M	35	M32-38
2	-2897161552818060146	M	35	M32-38
3	-8260683887967679142	M	35	M32-38
4	-4938849341048082022	M	30	M29-31
5	245133531816851882	M	30	M29-31
6	-1297074871525174196	F	24	F24-26
7	236877999787307864	M	36	M32-38
8	-8098239495777311881	M	38	M32-38
9	176515041953473526	M	33	M32-38
10	1596610250680140042	F	36	F33-42

Figure 4: Training Set

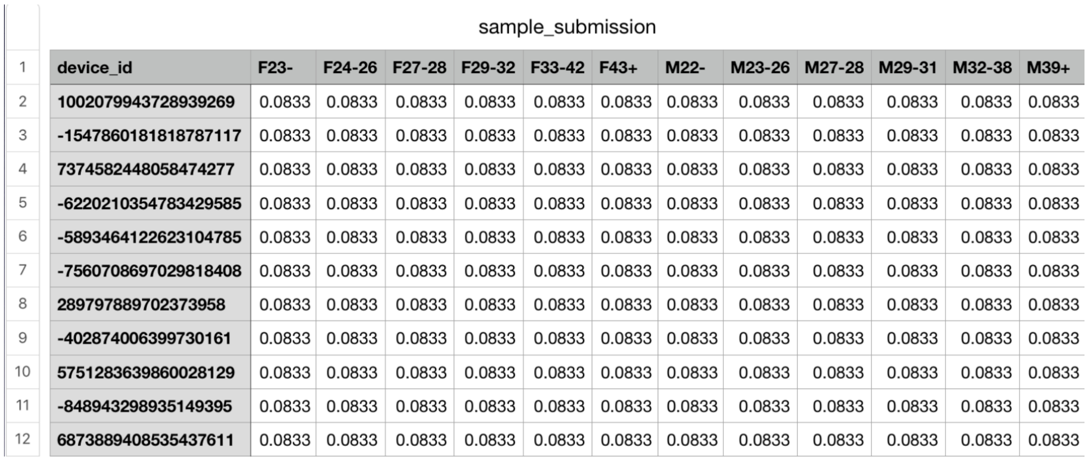
As for the training set, it shows device_id, corresponding with gender and age. If a person is male and 35 years old, he should belong to the group of M32-38. Finally, we need to submit the prediction in a form of probability distribution as shown below in Figure 5

4.2 Evaluation of Model

The competition evaluate the result using the multi-class logarithmic loss. Each device has been labeled with one true class. For each device, in $result.csv$ file, there will be a set of predicted probabilities (one for each class). The formula is then become:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{M} y_{ij}\log(p_{ij}) \qquad (1)$$

where N is the number of devices in the test set, M is the number of class labels, log is the natural logarithm, y_{ij} is 1 if device i belongs to class j and 0 otherwise, and $p_{ij}$is the predicted probability



sample_submission

1	device_id	F23-	F24-26	F27-28	F29-32	F33-42	F43+	M22-	M23-26	M27-28	M29-31	M32-38	M39+
2	1002079943728939269	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
3	-1547860181818787117	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
4	7374582448058474277	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
5	-6220210354783429585	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
6	-5893464122623104785	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
7	-7560708697029818408	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
8	289797889702373958	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
9	-402874006399730161	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
10	5751283639860028129	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
11	-848943298935149395	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833
12	6873889408535437611	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833	0.0833

Figure 5: Submission Example

that observation i belongs to class j. For result, the probabilities for a given device are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). However, they need to be in the range of [0, 1]. In order to avoid the extremes of the log function, predicted probabilities are replaced with $\max(\min(p, 1 - 10^{15}), 10^{15})$.

4.3 Data Visualization

Data visualization is our first step to visualize and analyze the connection among different features. The application we used here is tableau public. It is introduced from the website: `https://public.tableau.com/en-us/s/`, Tableau Public is a free software that can allow anyone to connect to a spreadsheet or file and create interactive data visualizations for the web. Our aim is to have a basic justification that which features are of great importance and which are not.

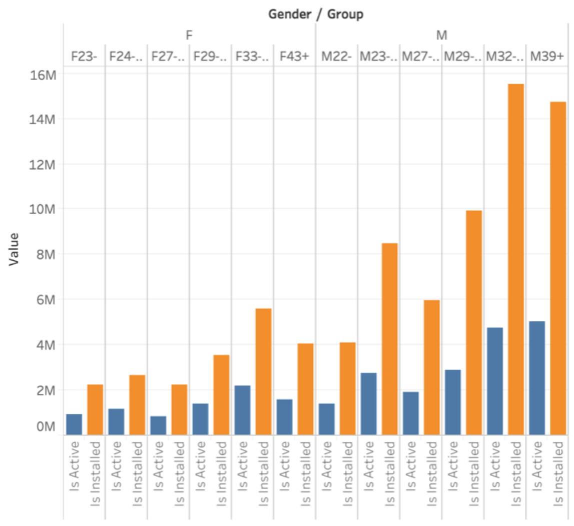
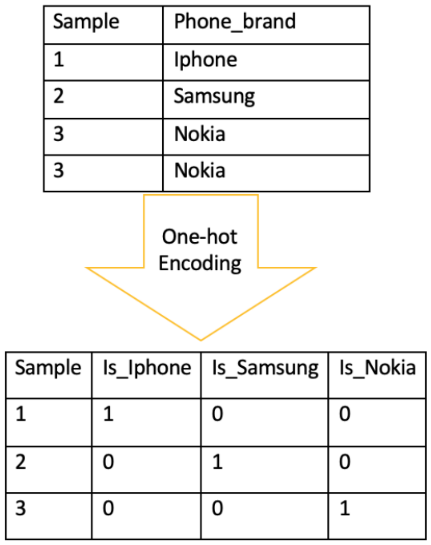


Figure 6: Submission Example

After trying different combinations of features, we plotted a graph as shown in Figure 6. The x-axis represents apps which are active or installed in different groups. Y-axis represents the the amount of different apps in different groups. The information we can get from this graph is: group M32-38 has the most app installation and group M39+ has the most app activation. This might imply if a user's device has quite a lot of apps installation and activation, he might be predicted as a male and with the age of greater than 32.

4.4 Data Preprocessing

One-hot encoding is a well-known data preprocessing technique nowadays. According to ..., it is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.



Sample	Phone_brand
1	Iphone
2	Samsung
3	Nokia
3	Nokia

Sample	Is_Iphone	Is_Samsung	Is_Nokia
1	1	0	0
2	0	1	0
3	0	0	1

Figure 7: Submission Example

5 CNN Model

Deep learning shows its dominating performance on image classification and text analysis recently. The Convolutional Neural Network(here as CNN), is a very common tool widely used in Deep Learning. Thus we designed a deep neural network to solve our problem, which also served as a comparison with our weighted-feature ensemble models. Here We Choose CNN as for our model mainly because the linear model (logistic regression) we did in the first week performance very bad and have a result about 700 + ranking in Kaggle.

5.1 Input Data Format

Generally, CNNs work well with data that has a spatial relationship. The CNN input is traditionally two-dimensional. but can also be changed to be one-dimensional, allowing it to develop an internal representation of a one-dimensional sequence. In order to prepare to build the model, we need to do the feature selection and format engineering first. After loading all the 7 data file we do the feature format work based on mainly four important features: phone brand,device model, installed apps, app labels. Then we concatenate the four types of features using the result of format work. Before we split xtrain dataset into train set and devset, we need to deal with y labels , here we use the one-hot encoding. And then use pad_sequences() method to transform out data in to nf x nf format that can be used in CNN.

5.2 Structure of the Module

we have 5 layers in our module, each of them has their only duties which are list below:
Input and output layer.

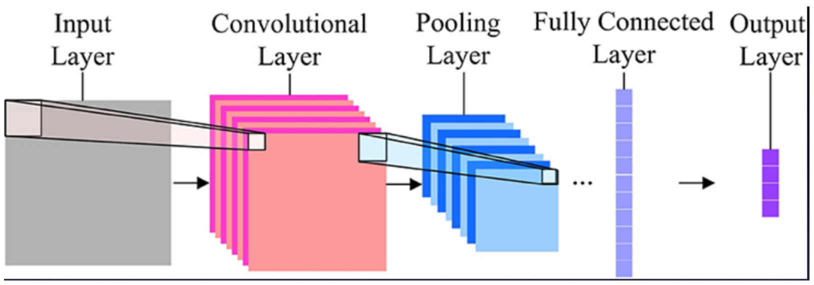
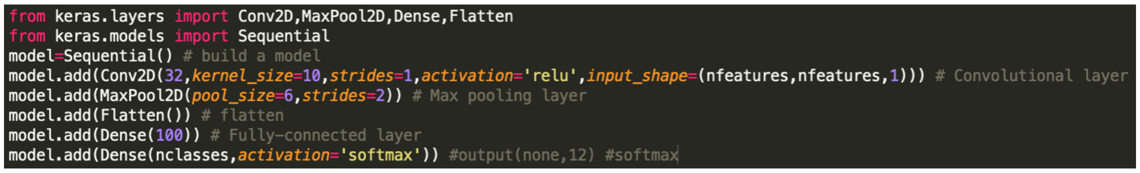


Figure 8: CNN Structure

Convolutional layer: Use Convolution kernel to extract and map features
Pool layer: Subsample, reduce the spatial size of the representation to reduce the amount of parameters and computation in the network.
Fully-connected layer: Refit at the end of CNN, prevent feature information missing, cross-entropy, Adam optimizer.

5.3 Implementation and Result

Interface that used in implementation: We use the keras as the main frame in our CNN implement. Keras is a high-level API to build and train deep learning models. It's used for fast prototyping, advanced research, and production, with three key advantages. The first one here is user friendly. Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors. The second one is modular and composable. Keras models are made by connecting configurable building blocks together, with few restrictions. Last but not least, it is easy to extend. Write custom building blocks to express new ideas for research. Create new layers, loss functions, and develop state-of-the-art models. The implement detail shown below:



```
from keras.layers import Conv2D,MaxPool2D,Dense,Flatten
from keras.models import Sequential
model=Sequential() # build a model
model.add(Conv2D(32,kernel_size=10,strides=1,activation='relu',input_shape=(nfeatures,nfeatures,1))) # Convolutional layer
model.add(MaxPool2D(pool_size=6,strides=2)) # Max pooling layer
model.add(Flatten()) # flatten
model.add(Dense(100)) # Fully-connected layer
model.add(Dense(nclasses,activation='softmax')) #output(none,12) #softmax
```

Figure 9: Implementation

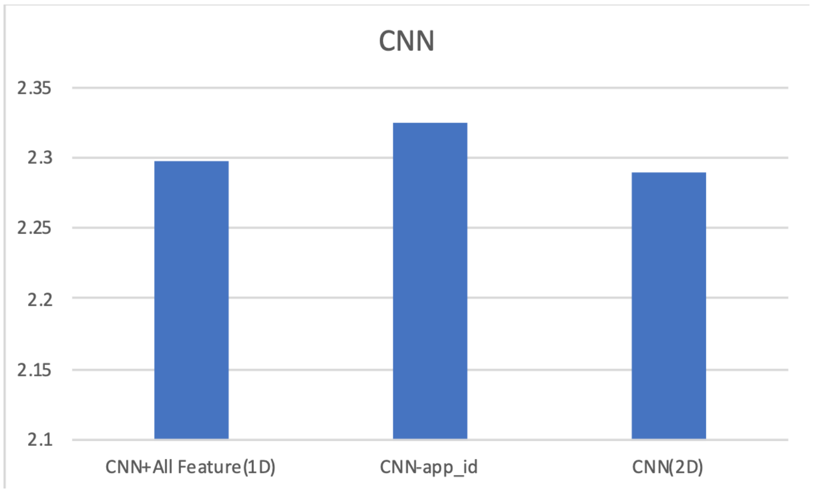


Figure 10: Result

Figure 10 shows the result of CNN. To get better score, first, we have tried to remove one of the features, which helps the score increase. However, It is just a way to get higher scores in kaggle so therefore we can surely remove other features to see if the performance is better or not. Kernel number and size, pooling size and the activation function(softmax now) will result in different evaluations. Here we have used the kernel size as 10,pool size as 6 and softmax as activation function, cross-entropy and Adam optimize. If we modify these parameters, then the result might be better.

We also have tried to used the Conv1D but not Conv2D so that we do not have to reshape and padding our data. And we have found that the result is not becoming better a lot. However, we believe it is relate to other parameters also so we can not say which one is better based on just the same result. It is not comparable. Other optimizing skills are available. There are many ways such as PCA and increase diversity, combine neural network with model stacking. Also we can integrate app_id, label_id, and category information, we will use network embedding to construct new features.

6 XGBoost and its Ensemble

Firstly, we use a single xgboost model as a based line for our competition and compare it with CNN method. The most important parameters of a xgboost model is including number of trees, the depth for each tree, the type of regularizer in the objective function, and the learning rate for the optimize group. After roughly tune these parameters by making use of grid search, we compare our prediction results with CNN models. At this step, different feature selection methods also have been tested to show that same model would have different performance with different features.

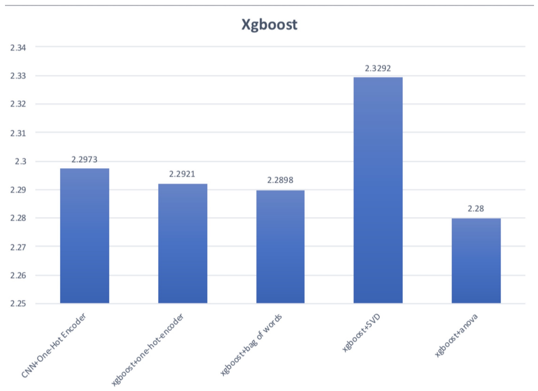


Figure 11: Experiment Results of Single Xgboost

The first two column in the figure 6 shows the effectiveness of the xgboost method. The last four column shows that features are important to model performance. After showing , we try to overcome its underlying weaknesses that can not make use of all the information of the features as we mentioned in section 2.

6.1 Feature Importance Extraction

During the competition, using the $get\_fscore()$ function of xgboost interface to return the importance features that have been used during the training process of xgboost model. It is shown that xgboost model only make use 5000 out of more than 20000 features in the feature space. This result either caused by the other 15000 dimension features are totally useless for this task or the xgboost cannot make use of all the information during its training process. In practice, the later condition is often the case since the data features are extracted according to informations that relative to the problem.

Furthermore, we plot the top 20 important features when building the tree. The x-axis represents the number of times a feature is used to split data across all the trees, which is the metric that evaluate the importance of a feature in the $get\_fscore()$ function. Thus, the larger the returned value of a feature. The y-axis is the dimension of the important features. We can see that the most important feature is the 21459 dimension feature in the training dataset and it is a dominating feature to this

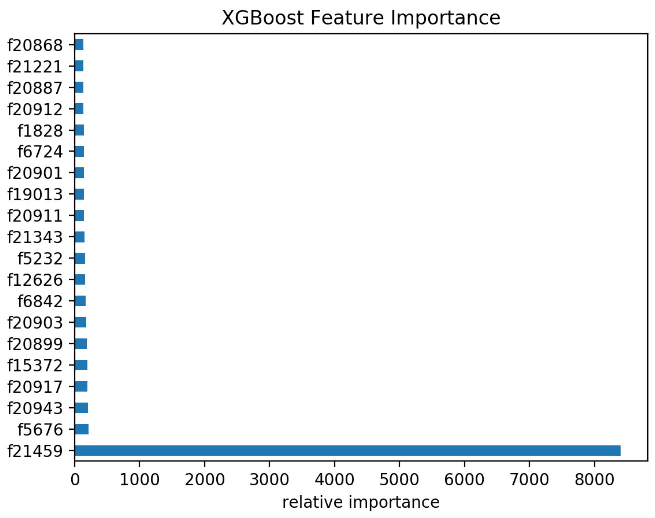


Figure 12: Feature Importance

problem since it almost 80 times important than the second important feature. This means we can make use this feature to get a better prediction results in the future.

6.2 Weighted Model Ensemble

To make use of the other 15000 dimension trivial features that haven't been used in a single xgboost model. We first split those trivial features from the important features both for the training dataset and testing dataset. Then further randomly split the trivial features into 10 sub-feature space again, as showed in Figure 6.2. After that, we have 11 different training dataset and testing dataset, so that we can train 11 different xgboost models to make better predictions.

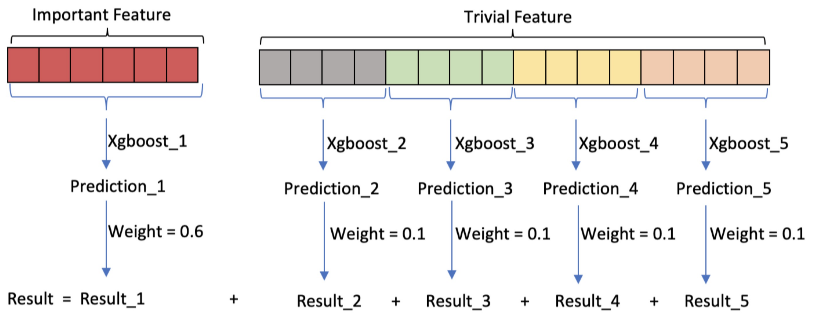


Figure 13: Weighted Model Ensemble

However, it is the fact that the model trained by important features would have better predictions. In order to make sure the important feature model contribute more to the prediction, we set different weights to different models instead of taking averages as often used in model ensembles. In our experiments, we set the weight of important model prediction as 0.7, and other 10 weights of trivial models predictions are 0.03.

6.3 Two-layer Tree Model Stacking

Model stacking is another technique that often used in model ensembles[]. Stacking means the results of a previous mode can be used as the input of the following one. In our competition, a two-layer tree

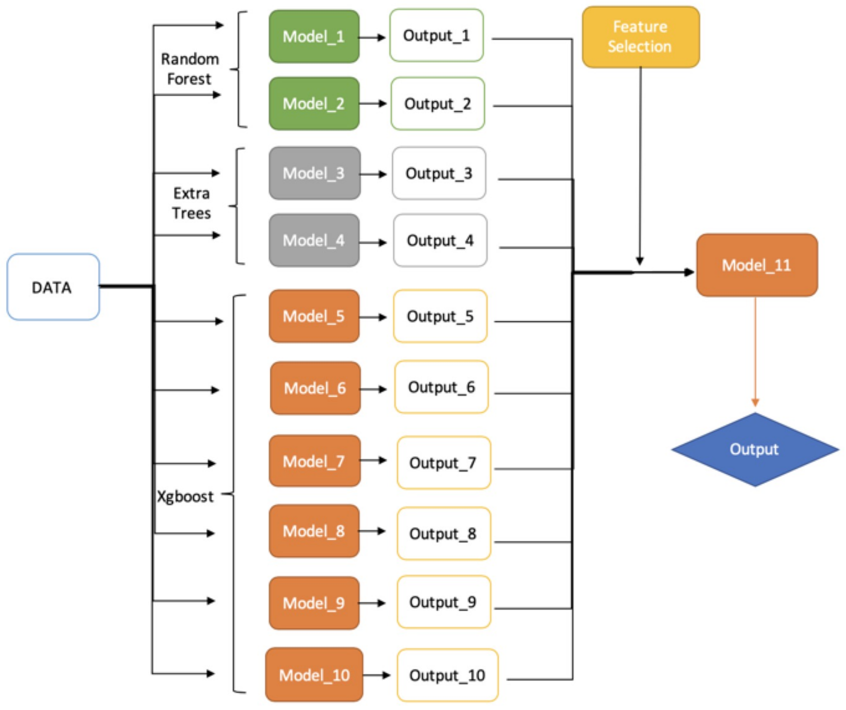


Figure 14: Two-layer Tree Model Stacking

model stacking model has been designed. In the first layer, we use ten tree models, which including 2 random forests, 2 extra class tree models, and 6 xgboost models. For each model, we use10-fold CV. Since the predictions on the evaluation data need to be imported into the second layer model, so we need to concatenate to all the predictions on each validation dataset as the output of each model in the first layer. Then we concatenate all the outputs of models in the column direction as the final output of the first layer models. Our task is a 12-classification task, so feature dimension of the output of the first model layer is $12 * 10 = 120$.

In the second layer, in order to improve the performance the second layer model, we add the dominant feature into the input datasets. Furthermore, 10 models will be trained using 10-CV, but only make use of the models whose $logloss$ smaller than the threshold (2.1) make predictions.

6.4 Experiment Results of Ensemble Models

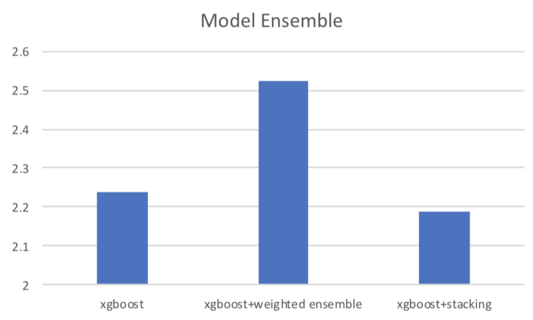


Figure 15: Model Ensemble Results

Figure 7 shows the results of different ensemble models compared with xgboost. It is interesting to see that the weighted-ensemble model didn't achieve the performance we expected to be. It is even worse than the single xgboost. It can be caused by lots of reasons, such as the parameter of different xgboost models are not well tuned, or the weights for different models isn't appropriate. We will save it as a future work to improve the performance of the weighted ensemble models. But the two-layer model have the best performance after **tuning all day**, which help us kick the leaderboard in the end.

7 Conclusion and Future Work

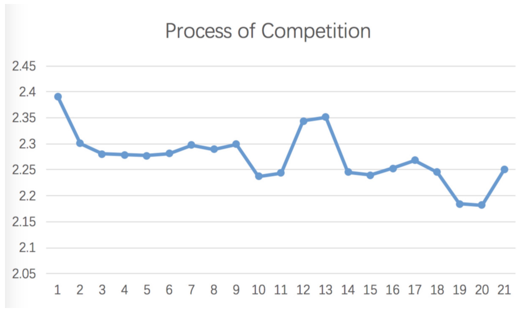


Figure 16: Score Timeline

The process of finishing this competition is not as easy as we expected, we met lots of difficulties, such as how to deal with large amount of data, the model always not work well when we built it. To better tuning the model, we learned lots of knowledge about tree boosting and convolutional neural networks. Figure 6 is the score timeline of our part submitted. The fluctuation of the line implies that we didn't improve our results all the time, which also means that the method which should work in theory doesn't works well in practice. Other interesting direction of improve the performance of our model is using feature embedding to find the hidden pattern of the feature space.

Reference

[1] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.
[2] Safavian, S. Rasoul, and David Landgrebe. "A survey of decision tree classifier methodology." IEEE transactions on systems, man, and cybernetics 21.3 (1991): 660-674.
[3] Jolliffe, Ian. Principal component analysis. Springer Berlin Heidelberg, 2011.
[4] Wallach, Hanna M. "Topic modeling: beyond bag-of-words." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.
[5] Yang, Yiming, and Jan O. Pedersen. "A comparative study on feature selection in text categorization." Icml. Vol. 97. No. 412-420. 1997.
[6] Dos Santos, Cicero, and Maira Gatti. "Deep convolutional neural networks for sentiment analysis of short texts." Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. 2014.
[7] Menard, Scott. Applied logistic regression analysis. Vol. 106. Sage, 2002.