ECE657A Project Report

Lei Yao(20658122), Jun Zhao(20683461), and Chongyi Zhou(20657803)

University of Waterloo, Waterloo, ON N2L 3G1, Canada

1 Introduction

Our project is based on the Alibaba Tianchi Big Data competition "Customer Flow Forecasts on Koubei.com". Our main goal is to predict the customer flow per day during the period of 14 days (11.01.2016-11.14.2016) for 2000 shops. It is a typical regression problem with time series data.

Based on the literature review on some well-recognized research papers that are related to our project, we select three popular methods for solving the current problem, including XGBoost, Neural Network, and SVM. Particularly, it was shown that there were 17 out of 29 winning teams in Kaggle competitions using XGBoost model in 2015. Also, the other two methods have also shown good performance in similar problems.

1.1 Competition background

Koubei is a Chinese Online-to-Offline (O2O) platform, which provides customers with a variety of shops' basic information like level of grading, category, and other customers' comments, etc. Customers who have logged in this platform can make payment for different shopping items after viewing relevant shop information. Koubei also aims to offer sales forecast services for every shop on this platform, which helps shops optimize their cash management, reduce unnecessary cost, and improve user experience.

In this competition, customer flow is defined as "number of customers making payment via Alipay in a shop during a particular period of time". With all the payment and browsing history in the given period from 07.01.2015 to 10.31.2016, we are supposed to predict the customer flow of each of the 2000 shops for the following 14 days.

1.2 Dataset description and evaluation metrics

Provided data: Generally, we are provided with three kinds of data[4] (requiring registration before downloading the data). One contains shop information data (Table 1), with ten fields and 2000 records which correspond to 2000 shops. The other two kinds of data includes payment log (Table 2) and browsing log (Table 3) information. The former contains 69674110 records, and the latter contains 5556715 records. Both these two kinds of data are collected during the period from 07.01.2015 to 10.31.2016 (except 2015.12.12). All provided data is of string type, and the prediction data is of integer type. Data files are in *csv* format with UTF-8 encoding.

Table 1. Shop Information Data

Field	Description
shop_id	Shop id
city_name	City name (in Chinese)
location_id	Neighbor shops have same location id
per_pay	Average pay (larger number indicates higher average pay)
score	Shop score given by users (larger number indicates higher score)
comment_cnt	Users comment number (larger number indicates more comments)
shop_level	Shop level given by KouBei (larger number indicates higher level)
cate_1_name	First level category name (in Chinese)
cate_2_name	Second level category name (in Chinese)
cate_3_name	Third level category name (in Chinese)

Table 2. Users Pay Behavior

Field	Description
user_id	user id
Shop_id	shop id (related to shop_info)
time_stamp	pay time

Table 3. Users View Behavior

Field	Description
user_id	user id
Shop_id	shop id (related to shop_info)
time_stamp	view time

External data: Considering other factors that may influence shop customer flow, we also collected weather and air quality information for each city (Table 4).

Table 4. Weather and air quality information

Field	Description
weather	cloudy, sunny, rainy, etc.
Max_temp	maximum temperature
Min_temp	minimum temperature
air quality	PM 2.5

Official evaluation metrics: The official evaluation metrics provided by the competition will be used as the evaluation metrics as follows:

$$L = \frac{1}{nT} \sum_{i}^{n} \sum_{t}^{T} \left| \frac{c_{it} - c_{it}^{g}}{c_{it} + c_{it}^{g}} \right| \tag{1}$$

where c_{it} represents the forecast customer flow of shop i in day t; c_{it}^{g} represents the ground truth customer flow of shop i in day t.

2 Literature review

In this section, literature review on the selected methods and their application to similar problems are presented as follows.

2.1 XGBoost

XGBoost[5] is a scalable machine learning system for tree boosting, which is one technique that shines in many applications. This algorithm has been widely recognized in Kaggle machine learning competitions. Among the 29 challenge winning solutions published at Kaggles blog during 2015, 17 solutions used XGBoost. Furthermore, XGBoost gives state-of-the-art results on a wide range of problems including store sales prediction, which is closely related to our project. This system is based on the gradient tree boosting algorithm and it can support the distributed processing frameworks Hadoop, Spark, and Flink.

As to the idea behind this algorithm, it is the boosting technique. The basic idea of boosting is to combine multiple weak models to produce a powerful ensemble. Boosting algorithm[6] allows fitting many weak classifiers to reweighted versions of the training data and classifying final examples by majority voting (Figure 1). When using decision tree as a weak estimator, the Generalized Boosted Model (GBM) reduces to the Gradient Boosted Tree(GBT) which can work with different loss functions (regression, classification, risk modeling etc.)[2]. Finally, XGBoost (Extreme Gradient Boosting) is a more regularized and scalable version of Gradient Boosted Tree with good bias-variance (simple-predictive) trade-off "out of box" and great computation speed.

2.2 Neural network

Artificial neural networks (ANNs)[9] have been widely used in time series forecasting. Several distinguishing features of ANNs make them valuable and attractive for a forecasting task. First, ANNs are data-driven self-adaptive methods. They learn from examples and capture subtle functional relationships among the data even if the underlying relationships are unknown or hard to describe. Second, ANNs can often correctly infer the unseen part even if the sample data contain noisy information. Third, ANNs have more general and flexible functional forms than the traditional statistical methods. Multi-layer perceptron (MLP) is

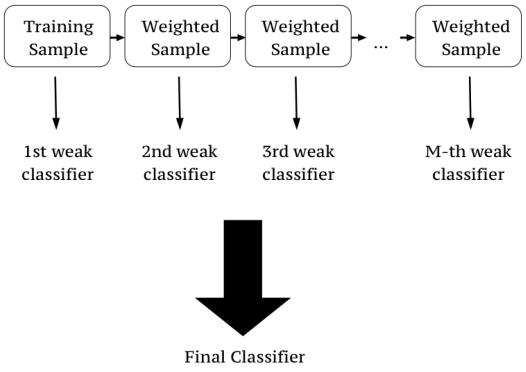

Fig. 1. Basic idea of boosting technique[2].

a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs, this is a supervised learning algorithm, which is good at learning non-liner models and time series prediction.

2.3 SVM

SVM[7] is to minimize the structural risk instead of the usual empirical risk by minimizing an upper bound of the generalization error, and it obtains an excellent generalization performance. More formally, it constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space. It often happens that the sets to discriminate are not linearly separable in that space regression and time series prediction, so there are some kernel functions to map original space into higher-dimensional space then make sets linearly separable. Experiment results show that SVM model has better performance over others in the field of regression and time series prediction.

2.4 Ensemble methods

The goal of ensemble methods[3] is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability or robustness over a single estimator. Ensemble methods mainly includes two distinguished families: averaging methods and boosting methods. Since the latter has been reviewed in the previous *XGBoost* section, we will focus on reviewing the averaging methods in the current subsection.

In averaging methods[3], the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

Particularly, in averaging algorithms, bagging methods[3] form a class of algorithms which build several instances of a black-box estimator on random subsets of the original training set and then aggregate their individual predictions to form a final prediction, which provides a very simple way to improve with respect to a single model without making it necessary to adapt the underlying base algorithm.

In our project, we combine both averaging and boosting methods to further reduce the estimation variance and optimize the prediction accuracy.

3 Approach

Our approach to the current challenge consists of four steps as follows.

3.1 Data preprocessing

Motivation

1. Some fields of the dataset are incomplete, there are some lacking attribute values, some fields may contain errors or outliers and there may be some inconsistencies.
2. The given dataset only contains information of transactions during a certain period, our model needs the customer flow of each shop for each training day.
3. Some features are nominal, we need to label them by using numerical values. For example, the weather for a day could be cloudy.
4. For SVM and Neural Network algorithm, features should be normalized, since these models are sensitive to feature scaling.

Methods

1. We fill in the missing data and inconsistent data of a shop by averaging available feature values of the same shop; We remove noisy data and outliers by using the 25 percentile and 75 percentile of that feature.
2. Multithreading technique is used in calculation of the customer flow for each shop. We create a thread pool of 16 threads to compute the customer flow, which can finally improve the computation efficiency by 3 times.
3. We label nominal feature by hardcoding. For example, weather is labeled into five levels. The larger the value of the weather label is, the worse the weather is.
4. We use the tool in *sklearn* called "StandardScaler" for standardization.

3.2 Feature extraction

We have mainly extracted three types of features, they are "Basic information", "Temporal information" and "Recent data and current trends". Basic information is independent of time period and only related to the shop information

under consideration. In contrast to the basic information, temporal information is independent of the shop information under consideration and only related to the given time period. Recent data and current trends information are related to both the shop information and the time period.

Basic information: Our goal is to predict each shop's customer flow, we need to gather some basic information for all shops. Each shop has its own information, including the number of comments, the average review score, and its category, etc (Table 5). This information could be important features of a shop. For example, shops which are located in big cities tend to have larger customer flow than shops in small cities do.

Table 5. Basic Information

Field	Description
ShopId	Shop id
City	City code
PerPay	Average Pay
Score	Shop score given by users
CommentCnt	Users comment number
ShopLevel	Shop level given by KouBei
Category	Shop category code

Temporal information (Table 6): Apart from the date information, we believe that holiday information is also an important factor for the prediction of custom flow. We categorize each day into three holiday levels (working day, weekend, national holiday). Furthermore, the number of holidays in the last week, this week and future week are also included as important features. Additionally, we also take into account the weather information as temporal information.

Recent data and current trends (Table 7):

1. To create features on recent data, we select the customer flow data of specific shop for each day in the train set. Then for each record, we use the data during the last period with respect to the date of that record (last week, last two week, last month, and last two month) as the recent history, and we compute the measure of centrality (mean value) and the measures of spread (standard deviation, skewness, and kurtosis) of recent customer flow history as the features of recent data.
2. For current trend data, we generate two groups of features. The first one ('trend_mean') is the mean values of the customer flow of the same days of week during last period (last week, last two week, last month, and last two month). As for the second group ('trend_estimated'), we train a *sklearn* Ridge

Table 6. Temporal Information

Field	Description
Year	Year of this sample (2016)
Month	Month of this sample (8 - 11)
Day	Day of this sample (1 - 30/31)
dayOfWeek	Monday - Sunday (1 - 7)
Holiday	Holiday Level (0 - 2)
numHolidayCur	Number of holidays in current week
numHolidayLast	Number of holidays in last week
numHolidayNext	Number of holidays in next week
Max_temperature	Maximum temperature (e.g. 25)
Min_temperature	Minimum temperature (e.g. 10)
Weather	level (1 - 5)
AirQuality	pm 2.5

regressor with the customer flow data during the last period and estimate the custom flow for the current day as the trend feature.

Table 7. Recent Data & Current Trends

Field	Description
lastPeriod_mean	Mean of last period's customer flow
lastPeriod_std	Standard deviation of last period's customer flow
lastPeriod_skew	Skew of last period's customer flow
lastPeriod_kusrtosis	Kusrtosis of last period's customer flow
trend_mean	Mean customer flow of same days of week during last period
trend_estimated	Estimated customer flow based on last period's regression (Ridge)

3.3 Modeling

After extracting the features, we begin to construct the train set and validation set for our models. Based on the experience in the Kaggle competition "Rossmann Store Sales"[8], we use the preprocessed data of the last 14 days of the train data (from 10.18.2016 to 10.31.2016) as the validation set, and the preprocessed data for the 14 days from 11.01.2016 to 11.14.2016 as the final testing set.

Assuming that short term factors have much more significant influence on the future custom flow, we choose to use the data of previous 28 days (train set) to train our models and then make prediction of the custom flow for the following 14 days. In our project, our train set contains 56000 (2000×28) samples and

labels and the validation set has 28000 (2000×14) samples and labels. Also, the testing set contains 28000 (2000×14) samples. All sets are ordered by "ShopID" and date values("Year", "Month", and "Day").

For each single model, we use the train set of all features for training and evaluate the prediction results by calculating the official loss value with the validation set. We perform multiple calculations to select the best parameters for each single model, which is presented in Table 8. By comparing the loss values of three selected models and the base line method (KNN regressor), we select the best performing single model to ensemble for further optimization in the next step.

Table 8. Parameter selection for each single model

KNN	'n_neighbors' = 20
SVM	'C' = 10, 'gamma' = 0.005, 'epsilon' = 0.005
Neural Network	'hidden_layer_sizes' = (100, 1000, ...) (totally 26 hidden layers), 'alpha' = 0.04, 'max_itr' = 1000, 'early_stopping' = TRUE
XGBoost	'max_depth' = 5, 'earning_rate' = 0.02, 'num_rounds' = 1000

3.4 Ensembling

In order to further improve the prediction accuracy of single models, averaging methods will be applied to the best performing single model. Based on the experimental results (Table 9), we select the XGBoost single model with least modeling time (including traing and testing time) and lowest validation error for final ensembling.

With consideration of the ensembling method in the Kaggle competition "Rossmann Store Sales"[8], we build random subsets of 1/3 of the features using the Random Subspaces Bagging method. Then, we run over 500 random models and systematically calculate the validation error on every pair-ensemble of models (totally $500 \times (500 - 1)/2$ pair-ensembles).

From the best validated model pair, we obtain two best ensembling subsets of features. Based on these two subsets of features, we use the prediction of the ensemble model with the testing set for the final submission in the competition.

4 Implementation

In our project, all code is written in *Python*. We use *Jupyter Notebook* as our IDE, which is an open-source web application that allows creating and sharing documents that contain live code, equations, visualizations and explanatory text[1].

All code is stored in two folders: *featureExtraction* and *models*. The former contains the scripts that preprocess the original data, and the latter contains the scripts for modeling.

4.1 Dependencies

The following Python packages are required to run the code:

- *featureExtraction*: os, numpy, pandas, datetime, scipy, sklearn, multiprocessing.
- *models*: numpy, pandas, xgboost, sklearn, datetime

In our project, we use *sklearn.neural_network.MLPRegressor* as the Neural Network model and *sklearn.svm.SVR* as the SVM model.

4.2 Data structure

All original data is stored in the folder *dataset*, and all preprocessed data is put in the folder *preprocess*. The ensembling data is stored in the folder *combination*. The final submission *csv* prediction results are stored in the folder *prediction*.

4.3 Program structures

In order to generate the solution, we first run the code in the *featureExtraction* folder to generate the preprocessed data. The order of execution for feature engineering is described as follows:

1. *timeHistoryData.ipynb*: transfer the original transaction data to time history data of custom flow;
2. *featureExtraction_*.ipynb* (including basicInfo, currentTrend, recentData, recentDataView, temporalInfo, and weather): generate three types of feature data;
3. *featureEnsemble.ipynb*: ensemble all preprocessed features into ensemble data files for final training, validation and testing.

After generating feature data, the scripts in the *models* folder can be executed to select best parameters for each single model and train the ensemble model for final prediction submission. The general program execution structure is shown in Figure 2.

5 Evaluation

The evaluation of our models consists of two steps:

- **Validation for single models**: in this step, we compare the performance of the selected single models with the base line method (KNN regressor) and select the best performing single model in terms of validation error and modeling time (including training and testing time) for further ensembling.
- **Official evaluation of the final ensemble model**: for the final submission in the competition, we use the selected best performing model to construct the ensemble model, then we submit the prediction of this ensemble model with the testing set for official evaluation.

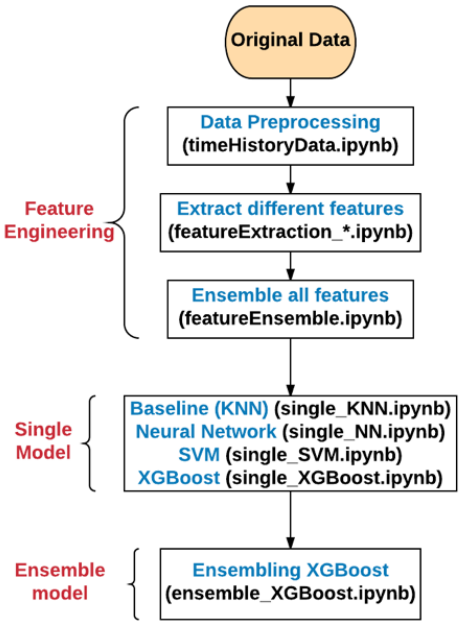

Fig. 2. Program structure

5.1 Validation for single models

Performance evaluation of the selected models in comparison with base line methods (KNN regressor) is presented in Table 9. It is clear that the XGBoost model has the lowest validation error and best modeling time, which is suitable to perform large number of calculations. Therefore, the XGBoost model is selected for further ensembling, which reduces model estimation variance.

Table 9. Evaluation of the performance for each single model

Model	Validation error	Modeling time
KNN	0.130	4 min
SVM	0.113	10 min
Neural Network	0.112	58 sec
XGBoost	0.103	31 sec

5.2 Official evaluation

By ensembling the best performing single model (XGBoost) using the bagging method, the loss value based on the official evaluation metrics in the compe-

tition can be further improved to **0.0928** (Table 10). Our final ranking in the competition is top 10% among more than 4000 teams. The official evaluation performance of different models is illustrated in Figure 3.

Table 10. Comparison between single model and ensemble model for XGBoost method

XGBoost	Single model	Ensemble model
Evaluation (loss)	0.101	0.0928

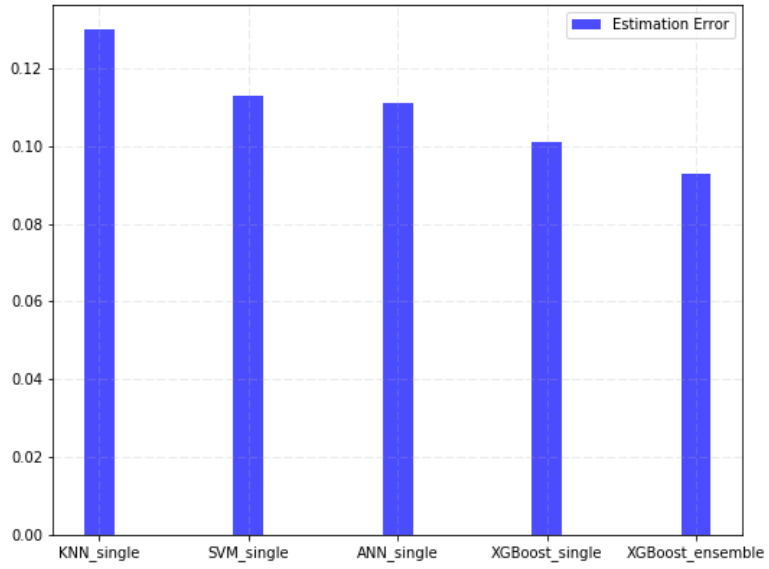

Fig. 3. Official evaluation (loss)

6 Discussion

6.1 Method suitability

In terms of official evaluation, all three selected models perform better than the base line method (KNN regressor), which supports the good performance of the selected models in the field of regression and time series prediction.

Particularly, the XGBoost model presents greatest computation speed and best estimation error among all methods because of application of boosting technique and scalable end-to-end tree system. Additionally, XGBoost can adaptively

handle missing value and unnormalized data. It is a highly effective and robust machine learning method suitable for supervised learning problems.

As for SVM and Neural Network models, they have similar performance in terms of prediction accuracy. However, SVM costs much more time for training and testing due to mapping inputs into high-dimensional feature spaces when data dimension is relatively high. Neural Network performs better in terms of modeling time, whereas we notice that design of the network structure is biased by designer's ideas and the results are comparatively difficult to interpret.

Additionally, we apply the bagging method to our best single model, which can further reduce the estimation error by 8%. This method can be used as a general and simple way to reduce the variance of a base model (e.g. XGBoost model) by introducing randomization into its construction procedure. It also provides a way to reduce overfitting and improve prediction accuracy with respect to a single model.

6.2 Parameter effect

In this subsection, the parameter effect for each method on the trade-off in performance will be discussed as follows.

SVM When applying SVR to prediction problems, the performance heavily depends on the setting of free meta-parameters of SVM:

- 'C': penalty parameter C of the error term.
- 'gamma': kernel coefficient for RBF kernel SVR
- 'epsilon': epsilon in the epsilon-SVR model

'C' trades off misclassification of training examples against simplicity of the decision surface. If it is too large, we have a high penalty for nonseparable points and we may store too many support vectors and overfit the model. If it is too small, it may lead to underfitting. 'gamma' can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. 'epsilon' has an effect on the smoothness of the SVRs response and it affects the number of support vectors, so both the complexity and the generalization capability of the network depend on its value.

Neural Network A few parameters have significant impact on the performance of Neural Network model, there are:

- 'hidden_layer_sizes': it is a tuple, the ith element represents the number of neurons in the ith hidden layer.
- 'activation': activation function for each hidden layer.
- 'solver': the solver for weight optimization.
- 'alpha': L2 penalty (regularization term) parameter.
- 'max_iter': Maximum number of iterations.

– 'early_stopping': if it is true, the model stops training when validation score is not improving.

For 'activation function', four kinds of function were tested when tuning parameters, which includes 'identity', 'logistic', 'tanh' and 'relu'. The function 'relu' performs better than other activation functions. We use 'adam' for parameter 'solver', it is a stochastic gradient-based optimizer, which is recently proposed and works pretty well on relatively large datasets in terms of both training time and validation score. Parameter 'alpha' combats overfitting by constraining the size of the weights. Increasing 'alpha' can fix high variance, which is a sign of overfitting; while decreasing 'alpha' may fix high bias, which is a sign of underfitting. Hence, a suitable 'alpha' has a big influence on our model. Parameter 'max_iter' is tuned with 'early_stopping' equals true, we notice that the maximum number of iteration has relatively unimportant influence on our model, since the model stops training when its performance is not improving any more. One of the most important parameters for neural network model is 'hidden_layer_size', we first tried a small number of layers with about 100 neurons for each layer, the prediction accuracy was not so satisfying. Then we added more layers, and the accuracy got better. However, when the number of layers reached a certain value (e.g. 25), the prediction accuracy stops improving.

XGBoost The main parameters to adjust when using this method are 'max_depth', 'learning_rate', and 'num_rounds'. Their definitions are as follows:

– 'max_depth': maximum tree depth for base learners.
– 'learning_rate': boosting learning rate, which shrinks the feature weights to make the boosting process more conservative.
– 'num_rounds': number of boosting iterations.

Based on our experience, increasing 'max_depth' value makes the model more complex and costs more time for training. 'learning_rate' controls the step size shrinkage used in update to prevents overfitting. 'num_rounds' controls the number of rounds for boosting, and the estimation error tends to be stable as the number of boosting rounds grows more than 800. Thus, in order to obtain satisfactory balance between prediction accuracy and time performance, slightly higher value (e.g. 'num_rounds' = 1000) can be set on 'num_rounds' with relatively low values for 'max_depth' and 'learning_rate' to avoid overfitting.

References

1. Project jupyter, http://jupyter.org/index.html
2. Boosting - wisdom of the crowd (theory) (2017), https://github.com/ParrotPrediction/docker-course-xgboost/blob/master/notebooks/2.%20Basics/2.1%20Boosting%20-%20wisdom%20of%20the%20crowd%20(theory).ipynb
3. Ensemble methods: scikit-learn 0.18.1 documentation (2017), http://scikit-learn.org/stable/modules/ensemble.html
4. Ijcai-17 customer flow forecasts on koubei.com: Description and data (2017), https://tianchi.aliyun.com/competition/information.htm?spm=5176.100067.5678.2.Bf7XNE&raceId=231591
5. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. CoRR abs/1603.02754 (2016), http://arxiv.org/abs/1603.02754
6. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: icml. vol. 96, pp. 148–156 (1996)
7. Gao, C., Bompard, E., Napoli, R., Cheng, H.: Price forecast in the competitive electricity market by support vector machine. Physica A 382 (2007), http://www.sciencedirect.com/science/article/pii/S0378437107003251
8. Jacobusse, G.: Winning model documentation: describing my solution for the kaggle competition "rossmann store salse" (2015), https://kaggle2.blob.core.windows.net/forum-message-attachments/102102/3454/Rossmann_nr1_doc.pdf
9. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks:: The state of the art. International Journal of Forecasting 14(1), 35 – 62 (1998), http://www.sciencedirect.com/science/article/pii/S0169207097000447