Financial time Series Project
Prediction of future in-app purchase
Spring 2017

Submitted By:
Cong Shi
Gunjan Batra
Yahui Guo
Yanchi Liu
Yueqing Zhang

1. Introduction

Recently, ubiquitous mobile phones, especially the smart phones, have brought users with convenience, independence, mobility and flexibility. With fast development of mobile devices, tremendous mobile applications (apps), especially commerce apps (e.g., amazon, Walmart), have been developed to be downloaded by mobile users. Thus, the Mobile-Commerce business is growing rapidly in recent years. Take Alibaba Group's great sale on Nov. 11, 2014 for example. It is said that the Gross Merchandise Volume (GMV) on mobile terminals accounts for 42.6% of total GMV. Access to the network via mobile terminals has great advantages over common PC and it can take place anytime anyplace. Thus, mobile network tends to possess richer background data, such as users' location information, regularity in their access time and etc. This is an obvious trend in the mobile e-commerce. It gives us more chances to predict and personalize the recommendation in such condition. To this end, in this project, we aim to predict future in-app purchase based on historical user login, purchase and mobile device information. This problem can be formalized as a classification problem, given the login behavior in previous days, we want to predict whether or not the user will make purchase in the next day. Specifically, we will first extract various features related to user purchase behavior from the historical data. Then, with the features extracted from all users, we develop a customer purchase prediction model with different classification methods. With the model learned from training set, we will classify the users by predicting whether they will have a purchasing behavior or not. And the results of different methods will be compared with a cross-validation method.

2. Data Description

We have three datasets with user historical login and purchase activities collected from a Chinese online store.

2.1.User IP:

This dataset provides the ip address of user, which can be used as the unique id of user.

117.136.40.197:China:Guangdong:Shenzhen
208.54.90.171:United States:Pennsylvania:Philadelphia
202.137.156.230:Laos:Vientiane:Vientiane
116.21.120.141:China:Guangdong:Guangzhou
117.136.46.53:China:Jiangsu Sheng:Suzhou

2.2.User Login:

This dataset provides the login records of users, which contains user id, login time, and ip address.

4232765,ylt,yltpp_10000version_,3,562fdf64c7ab752e82d0ab9ab7da61b0f3bed000,117.136.40.197,2015-11-01 00:00:04
5227480,ylt,yltpp_10000version_,3,fdf0b43e074217d8ae37e8984e5583218a62633a,208.54.90.171,2015-11-01 00:00:04
551633,ylt,yltpp_10000version_,3,68345c75c48fc905bc7aa1a434eb06a282d54b25,202.137.156.230,2015-11-01 00:00:05
977092,ylt,yltpp_10000version_,3,7234a075faa2e3ebd131956f9f92319867996b25,116.21.120.141,2015-11-01 00:00:05

1567953,ylt,yltpp_10000version_,3,e4c7800f612c13290a49bf4cc7dd3cd938be967a,117.136.46.53,2015-11-01 00:00:05

2.3.User Purchase:

This dataset provides the purchase records of users, which contains user id, purchase time, amount, and ip address.

2254513,ylt,yltpp_10000version_,3,fe11566ee2ff8ac4395aa89cb3dce08d20a510aa,1.202.176.72,2015-11-01 00:20:34,6,CNY,60,appstore,com.qtz.zlsg.60yb
1243187,ylt,yltpp_10000version_,3,cb33ec0230edd3e19adecc65b0d5a74cdb7970c1,117.136.46.49,2015-11-01 00:47:50,6,CNY,60,appstore,com.qtz.zlsg.60yb
1925078,ylt,yltpp_10000version_,3,517b3cb6efebb7728e97a636adfcc617838f547c,107.77.90.76,2015-11-01 00:30:41,6,CNY,60,appstore,com.qtz.zlsg.60yb
1592153,ylt,yltpp_10000version_,3,d62a07dbf40babc03d9ebc0e1fd14f921b71f5db,174.62.77.247,2015-11-01 00:52:03,648,CNY,7888,appstore,com.qtz.zlsg.7800yb
1441532,ylt,yltpp_10000version_,3,01820be931dacc7a5181b3277bcf3cdb1d6bcdd0,222.128.182.249,2015-11-01 00:21:44,6,CNY,60,appstore,com.qtz.zlsg.60yb

3. Methodology

Since label the goal of our project is to find whether the customers will purchase for the next day and we use label feature to represent the idea that label equals to 1 means the customer will purchase for the next day, while label equals to 0 means the customer won't purchase for the next day. Based on this main idea, we have to analyze the data and make predictions using classification Techniques. We plan to use four different classification methods for this task: Logistic Regression, SVM, and Ensemble Techniques such as Random Forest, and Boosting (Adaboost). Next, we briefly introduce these classification methods.

3.1.Logistic Regression

Binary logistic regression is used to estimate the response (or dependent) variable which is binary (dichotomous), using the predictor (or independent) variables (features) which could be continuous and/or categorical independent variables.

Logistic Regression is a non-linear regression technique that assumes that the expected probability of a dichotomous outcome is:

$$P = \frac{1}{1 + e^{-(\alpha+\beta_1 X_1+ \beta_2 X_2+\cdots)}}$$

where the X_i are variables with numeric values (if dichotomous, they are, for example, zero for false and one for true) and the βs are the regression coefficients which quantify their contribution to the probability. Using this model, we can do stepwise selections of the variables and the corresponding coefficients computed. In producing the LR equation, the maximum-likelihood ratio is used to determine the statistical significance of the variables. Logistic Regression has proven to be very robust in a number of domains and proves an effective way of estimating probabilities from dichotomous variables.

Pros:

- Convenient probability scores for observations
- Efficient implementations available across tools

- Multi-collinearity is not really an issue and can be countered with L2 regularization to an extent
- Wide spread industry comfort for logistic regression solutions

Cons:

- Doesn't perform well when feature space is too large
- Doesn't handle large number of categorical features/variables well
- Relies on transformations for non-linear features
- Relies on entire data

3.2.SVM (Support vector machine)

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features we have in the data set) with the value of each feature being the value of a particular coordinate. We have user ID, IP address, Country, City, pay time, amount, website and login as the features. Then, we perform classification by finding the hyper-plane hat differentiate the two classes very well (look at the below snapshot).

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

The best thing about support vector machines is that they rely on boundary cases to build the much-needed separating curve. Reliance on boundary cases also enables them to handle missing data for "obvious" cases. SVM can handle large feature spaces which makes them one of the favorite algorithms in text analysis which almost always results in huge number of features where logistic regression is not a very good choice.

Pros:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

Cons:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Logistic regression and support vector machines are closely linked. Both can be viewed as taking a probabilistic model and minimizing some cost associated with misclassification based on the likelihood ratio. This lets us analyze these classifiers in a decision theoretic framework which is exactly suitable for our project's framework. So now we can see the common place of these two techniques and understand why we use both in our project.

3.3.Random Forest

Random forest is like bootstrapping algorithm with Decision tree (CART) model. It combines the predictions made by multiple decision trees, where each tree is generated based on the values of

an independent set of random vectors. The random vectors are generated from a fixed probability distribution. For example, say, we have 1000 observation in the complete population with 10 variables. Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observations and 5 randomly chosen initial variables to build a CART model. It will repeat the process (say) 10 times and then make a final prediction on each observation. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction.

Pros:

- It produces a highly accurate classifier.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- It offers an experimental method for detecting variable interactions.

Cons:

- Random forests sometimes overfit for some datasets with noisy classification/regression tasks.
- Unlike decision trees, the classifications made by random forests are difficult for humans to interpret.

3.4.Boosting (AdaBoost)

Boosting is an iterative procedure that works by adaptively changing the distribution of training data so that base classifiers focus on the training samples that are hard to classify. At every iteration, it applies a classification algorithm to weighted versions of the training data and then takes a weighted majority vote of the sequence of classifiers thus produced. It adaptively changes the weight at the end of each boosting round. We are hopeful that this technique will further improve the performance of the model. Most common version of boosting is AdaBoost

Pros

- Fast
- Simple and easy to program
- No parameters to tune (except T)
- No prior knowledge needed about weak learner
- Provably effective given Weak Learning Assumption
- Versatile

Cons

- Weak classifiers too complex leads to overfitting.
- Weak classifiers too weak can lead to low margins, and can also lead to overfitting.
- AdaBoost is particularly vulnerable to uniform noise.

4. Feature Extraction

Feature extraction is an important step of the whole solution. To obtain the training dataset, we first transform the user behavior (including login and purchase) log into a dataset with user's ID and date as the primary key. This means we extract features for each user each day since we aim to predict if a user will purchase in the next day. To label a training instance, we use "1" to represent the user making a purchase, also called positive instance; and "0" to represent the user not making purchases, noted as negative instance. The feature extraction is about the patterns of user login and purchase individually, and the combination of the interaction patterns between them.

4.1. User features

To make the prediction personalized, it's very important to depict a user thoroughly. To do it well, we find a lot of features to depict a user, with emphasis on purchase power and user behavior habits. The features are summarized based on the overall behavior of one user. Basic aggregating the features of raw behavior logs has been assumed in many aspects. The raw count of the behaviors in a selected set of periods could provide the information about the pattern of the user. It is also a basic preparation of other complex features that will be used.

Average number of login for this user:

$$\text{Login}_{\text{avg}} = \frac{\sum \#daily\ login}{\#day}$$

Average number of purchase for this user:

$$\text{Pur}_{\text{avg}} = \frac{\sum \#daily\ purchase}{\#day}$$

Average purchase amount for this user:

$$\text{Amt}_{\text{avg}} = \frac{\sum daily\ purchase\ amount}{\#day}$$

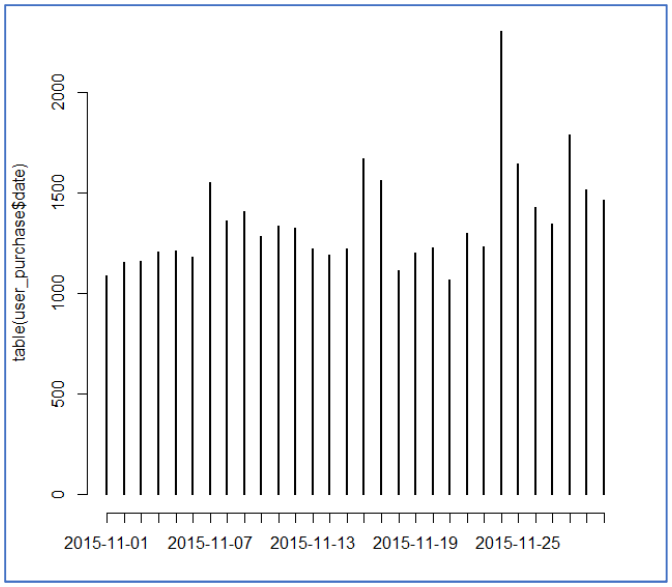
Conversion rate is one of the most valuable features in our model. For users, the behavior conversion rate is used to determine the probability to buy.

$$\text{CR} = \frac{\sum \#daily\ purchase}{\sum \#daily\ login}$$

Moreover, we can infer the user geographic information based on the ip address he/she login. Since one user may login at different places which indicate different ip addresses. For each user, we aggregate the ip address and use the most frequent one as the main ip address. Then we can obtain the country, city information based on the ip. We have the following features depict the geographic information of the users: country, province, and city.

4.2.Daily behavior features

Other than the general user features extracted, we also need to extract features that can depict the user daily behavior or recent behaviors. Since these features are important for the users to make purchase decision for the next day. From the data, we find that the purchase behavior for all the users exhibit a periodicity of 7 days (as shown in below figure). So here we choose a time span of 7 days to summarize user's recent activities. This means we cannot summarize user daily behavior before 2015-11-07 since we only have data from 2015-11-01.



We extract daily behavior features for each user as follows.

Number of login

$$\text{Login} = \#daily\ login$$

Number of purchase

$$\text{Pur} = \#daily\ purchase$$

Purchase amount

$$\text{Amt} = \#daily\ purchase\ amount$$

Average login in previous 7 days

$$\text{Login}_7 = \frac{\sum_{i-6}^{i} \#daily\ login}{7}$$

Average purchase in previous 7 days

$$\text{Pur}_7 = \frac{\sum_{i-6}^{i} \#daily\ purchase}{7}$$

Ratio of purchase after login

$$\text{RP}_7 = \frac{\sum_{i-6}^{i} \#daily\ purchase}{\sum_{i-6}^{i} \#daily\ login}$$

Number of days since last login

$$\text{Login}_{\text{diff}} = DateLogin_i - DateLogin_{i-1}$$

Number of days since last purchase

$$\text{Login}_{\text{diff}} = DatePurchase_i - DatePurchase_{i-1}$$

Finally, we end up with a training data set, which contains 772,214 rows with following 19 attributes, with 1 attribute as the label.

uid: user id
ip: ip address
date: date
login: number of login
purchase: number of purchase
amount: purchase amount

login_7: average login in previous 7 days
purchase_7: average purchase in previous 7 days
pur_aft_login_7: ratio of purchase after login
login_diff: number of days since last login
purchase_diff: number of days since last purchase
avg_login: average number of login for this user
avg_purchase: average number of purchase for this user
avg_amount: average purchase amount for this user
convrate: conversion rate, which is number of purchase/number of login
country: login country
province: login province
city: login city
label: label of purchase for the next day

5. Experiments and Results

In this part, we will present our application of Logistic Regression, SVM, Random Forest and Adaboost models in our dataset. We put the main R code and corresponding accuracies/results of each model here.

5.1.Logistic Regression model.

```
m_lr = glm(label~.-login_7-pur_aft_login_7, set.seed(123), data=train_data[,-c(1:3, 16:18)], family=binomial)
pred_lr = predict(m_lr, test_data)
pred_lr[pred_lr > .5] = 1
pred_lr[pred_lr <= .5] = 0
table(observed = test_data$label,predicted =pred_lr)
#             predicted
observed        0     1
      0     370742  1432
      1       6999  6934            TP = 6934/(1432+6934) = 0.8288309
accuracy = sum(diag(table(observed = test_data$label,predicted =pred_lr)))/nrow(test_data)
```

5.2.SVM model

Then we present SVM model's R code and accuracies/results here. We applied all four different kernels – linear, radial, sigmoid and polynomial.

```
idx <- sample(nrow(train_data), as.integer(nrow(train_data)*0.05))
obj = best.tune(svm, label~., set.seed(123), data = train_data[idx,-c(1:3, 16:18)], kernel = "linear")
m_svm <- svm(label~., set.seed(123), data=train_data[idx,-c(1:3, 16:18)], cost = 1, gamma = 0.08333333 , kernel = "linear")
pred_svm = predict(m_svm, test_data)
table(observed = test_data$label,predicted =pred_svm)
#             predicted
observed        0     1
      0       369961  2213
```

```
      1          5667   8266       TP = 8266/(2213+8266) = 0.7888157
accuracy = sum(diag(table(observed = test_data$label,predicted =pred_svm)))/nrow(test_data)

obj = best.tune(svm, label~., set.seed(123), data = train_data[idx,-c(1:3, 16:18)], kernel = "radial")
m_svm <- svm(label~., data=train_data[idx,-c(1:3, 16:18)], cost = 1, gamma = 0.08333333, kernel
= "radial")
pred_svm = predict(m_svm, test_data)
table(observed = test_data$label,predicted =pred_svm)
#             predicted
observed         0     1
      0     369998   2176
      1       5673   8260       TP = 8260/(2176+8260) = 0.7914910
accuracy = sum(diag(table(observed = test_data$label,predicted =pred_svm)))/nrow(test_data)

obj = best.tune(svm, label~., set.seed(123), data = train_data[idx,-c(1:3, 16:18)], kernel =
"sigmoid")
m_svm <- svm(label~., set.seed(123), data=train_data[idx,-c(1:3, 16:18)], cost = 1, gamma =
0.08333333, kernel = "sigmoid")
pred_svm = predict(m_svm, test_data)
table(observed = test_data$label,predicted =pred_svm)
#             predicted
observed         0     1
      0     359535  12639
      1      12214  1719       TP = 1719/(12639+1719) = 0.1197242
accuracy = sum(diag(table(observed = test_data$label,predicted =pred_svm)))/nrow(test_data)

obj = best.tune(svm, label~., set.seed(123), data = train_data[idx,-c(1:3, 16:18)], kernel =
"polynomial")
m_svm <- svm(label~., set.seed(123), data=train_data[idx,-c(1:3, 16:18)], cost = 1, gamma =
0.08333333, kernel = "polynomial")
pred_svm = predict(m_svm, test_data)
table(observed = test_data$label,predicted =pred_svm)
#             predicted
observed         0     1
      0     370213   1961
      1       6092   7841      TP = 7841/(1961+7841) = 0.7999388
accuracy = sum(diag(table(observed = test_data$label,predicted =pred_svm)))/nrow(test_data)
```

5.3.Random Forest

```
> m_rf <- randomForest(label ~.,data=train_data[,-c(1:3, 16:18)], importance=TRUE,ntree = 300)
> pred_rf = predict(m_rf, test_data)
> table(observed = test_data$label,predicted =pred_rf)
             predicted
observed        0     1
         0   370184   1990
         1   5461   8472
```

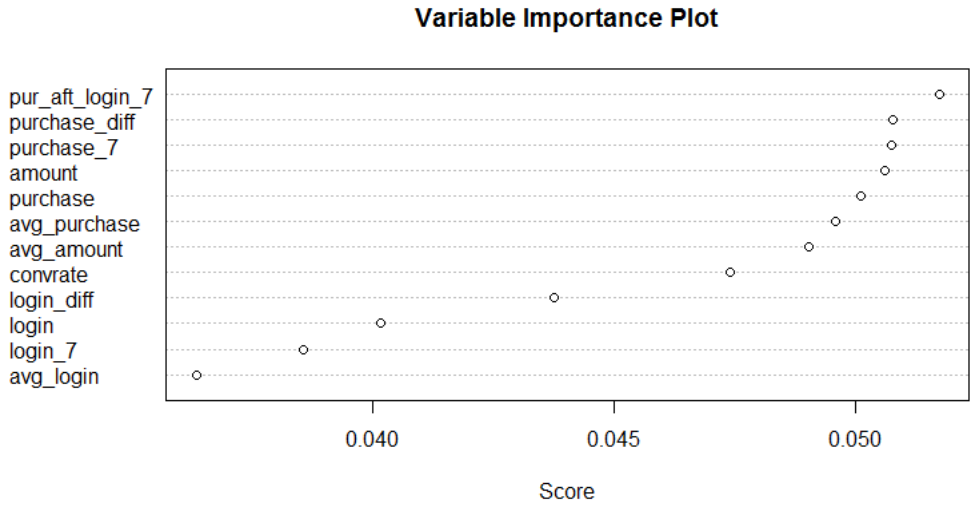
```
> accuracy = sum(diag(table(observed = test_data$label,predicted =pred_rf)))/nrow(test_data)
> accuracy
[1] 0.9807022
```

5.4. Adaboost

We applied Adaboost algorithm using 2 methods. Both give us same accuracy.

5.4.1. Using function Ada

```
> gen1 <- ada(label ~., data=train_data[,-c(1:3, 16:18)],type = "gentle", cont
rol = control, iter = 300)
> gen1 <- addtest(gen1, test_data[,-c(1:3, 16:18)], test_data$label)
> summary(gen1)
Call:
ada(label ~ ., data = train_data[, -c(1:3, 16:18)], type = "gentle",
    control = control, iter = 300)

Loss: exponential Method: gentle   Iteration: 300

Training Results

Accuracy: 0.988 Kappa: NA

Testing Results

Accuracy: 0.981 Kappa: NA
> varplot(gen1)
```




```
> pred_ab = predict(gen1, test_data)
> table(observed = test_data$label,predicted =pred_ab)
        predicted
observed      0      1
       0 370120   2054
       1   5432   8501
> accuracy = sum(diag(table(observed = test_data$label,predicted =pred_ab)))/n
row(test_data)
> accuracy
[1] 0.9806116
```

5.4.2. Using Function Boosting

```
library(adabag);
```

```
adadata=read.table("user_data.txt",sep=',',header=T,fill=T,quote="",na.strings=c("","None","NA
"))
adaboost<-
boosting(label~.,data=train_data[,c(1:3,16:18)],boos=TRUE,mfinal=2,coeflearn='Breiman')
pred_ab = predict(adaboost, test_data)
table(observed = test_data$label,predicted =pred_ab$class)
#              predicted
# observed      0      1
#        0    370560   1614
#        1      6091   7842
accuracy=sum(diag(table(observed = test_data$label,predicted =pred_ab$class)))/nrow(test_data)
accuracy
#[1] 0.9800444
summary(adaboost)
# Length Class   Mode
# formula         3 formula call
# trees           2 -none-  list
# weights         2 -none-  numeric
# votes      772214 -none-  numeric
# prob       772214 -none-  numeric
# class      386107 -none-  character
# importance     12 -none-  numeric
# terms           3 terms   call
# call            6 -none-  call
adaboost$trees
```

5.5. Compiled Results

The compiled results for all the models applied are as follows:

Model	**Accuracy**
Logistic Regression	0.9781641
SVM – kernel = “linear”	0.9795911
SVM – kernel = “radial”	0.980091
SVM – kernel = “sigmoid”	0.9350232
SVM – kernel = “polynomial”	0.9788193
Random Forest	0.9807022
AdaBoost	0.981

6. Analysis

Model	Accuracy	True Positive (TP)
Logistic Regression	0.9781641	0.8288309

SVM – kernel = “linear”	0.9795911	0.7888157
SVM – kernel = “radial”	0.9796714	0.7914910
SVM – kernel = “sigmoid”	0.9356318	0.1197242
SVM – kernel = “polynomial”	0.9791431	0.7999388

For Logistic Regression, we use whole training data which is the 50% of whole dataset to train model and use whole testing dataset which is the rest 50% of whole dataset for accuracy prediction. For SVM with four various kernels, we use 5% of training data to train model and use whole testing dataset for accuracy prediction. We use less training data because of the limitation of computational resources and consumption of time. Although we use less training data, the accuracy of SVM model is still satisfactory.
For Random Forest and AdaBoost, we use 50% data for training and 50% data for testing.

The final part is showing the analysis of Logistic Regression model and SVM model. As we can see, the accuracy of Logistic Regression model is higher than SVM model when using sigmoid kernel, but lower than SVM model when using linear kernel, radial kernel and polynomial kernel. Since the accuracy of Logistic Regression and SVM model are high, we can tell that Logistic Regression model and SVM model are all meaningful and reasonable to be used in our project for predicting label of purchase for the next day. Although the accuracy of model is important parameter to show us the performance of model, True Positive(TP) is the more important parameter we need to focus on and pay more attention. The reason is that our main goal is want to maximize the accuracy of predicting the user who actually purchase the item for the next day and our model exactly predict it just as the observation. Based on this criterion, we prefer to choose Logistic Regression model as the best model since the TP rate of Logistic Regression model is 0.8288309 which is the highest TP rate out of five models. And the rest of models can be ranked from the highest TP rate to the lowest TP rate.

We observe that the accuracy of Ensemble methods, bring improvement in the accuracy of the models as both Random Forest and Adaboost have higher accuracy than that of Logistic Regression and SVM. Boosting achieved an accuracy of 98.06% which is quite significant. We also plotted the variables and identified the following 4 variables to be the most important: -
1. pur_aft_login_7: ratio of purchase after login
2. purchase_diff: number of days since last purchase
3. purchase_7: average purchase in previous 7 day
4. amount: purchase amount

Random Forest also has an accuracy of 98.07%. We also calculated the Sensitivity, Specificity, Precision, Recall and F1 Measure for the 2 models and all of them were almost same. So, we can conclude that both the ensemble methods bring about an improvement in the classification models and should definitely be used.

7. Time Series Model for Company Prediction

Data explanation:

	V1	V2	V3	V4	V5	V6
1	1001019	2015-11-01	118.183.253.53	2	0	0
2	1001019	2015-11-02	27.224.71.144	5	0	0
3	1001019	2015-11-03	42.89.26.160	6	0	0
4	1001019	2015-11-04	42.89.26.160	4	0	0
5	1001019	2015-11-05	42.89.26.160	8	0	0
6	1001019	2015-11-06	106.38.252.44	5	0	0
7	1001019	2015-11-07	106.38.252.44	1	0	0
8	1001019	2015-11-08	61.178.58.42	5	0	0
9	1001019	2015-11-09	61.178.58.42	2	0	0
10	1001019	2015-11-10	61.178.58.42	4	0	0
11	1001019	2015-11-11	61.178.58.42	3	0	0
12	1001019	2015-11-12	117.34.96.138	6	0	0
13	1001019	2015-11-13	117.34.96.138	4	0	0
14	1001019	2015-11-14	118.194.237.5	6	0	0
15	1001019	2015-11-15	1.82.229.209	3	0	0
16	1001019	2015-11-16	1.82.229.209	3	0	0
17	1001019	2015-11-17	1.82.229.209	2	0	0
18	1001019	2015-11-18	1.82.229.209	6	0	0
19	1001019	2015-11-19	1.82.229.209	5	0	0
20	1001019	2015-11-20	106.38.252.53	2	0	0
21	1001019	2015-11-21	106.39.253.89	5	0	0
22	1001019	2015-11-22	106.39.253.89	7	0	0
23	1001019	2015-11-23	106.39.253.89	2	0	0
24	1001019	2015-11-24	101.254.209.38	3	0	0
25	1001019	2015-11-25	59.76.43.18	7	0	0
26	1001019	2015-11-26	59.76.43.18	1	0	0
27	1001019	2015-11-27	59.76.43.9	3	0	0
28	1001019	2015-11-28	59.76.43.17	8	0	0
29	1001019	2015-11-29	59.76.43.30	12	0	0
30	1001019	2015-11-30	59.76.43.30	2	0	0
31	1002710	2015-11-01	117.136.75.209	6	0	0
32	1002710	2015-11-02	117.136.75.146	3	0	0
33	1002710	2015-11-03	117.136.75.142	19	0	0
34	1002710	2015-11-04	117.136.75.167	4	0	0
35	1002710	2015-11-05	117.136.75.144	13	0	0
36	1002710	2015-11-06	117.136.75.177	8	0	0

Showing 1 to 36 of 2,324,220 entries

The data is retrieved from a Chinese online-shopping mobile app.
As showed in the previous chart, we have 2324220 rows of observations and 6 variables. Column 1 is user id, which is unique for each different app user. Column 2 is date, which range from 2015-11-01 to 2015-11-30. Column 3 is the most frequently used IP address by specific user on that day. Column 4,5,6 indicates the total login times, purchase times, and total money spent for each user

on each day accordingly. There are 77474 different users who have at least login onto the app once in November 2015.

```
> summary(data[,4])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   3.795   5.000 224.000
> summary(data[,5])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.0209  0.0000 32.0000
> summary(data[,6])
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   0.000    0.000    0.000    1.175    0.000 6486.000
```

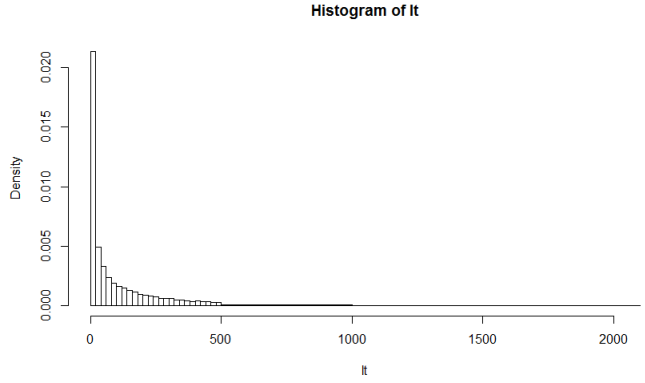
We can see that the maximum amount of login times for any user in any day is 224 times, the ave rage amount of login times for any user in any day is 3.795 times, the maximum amount of purch ase times for any user in any day is 32 times, and the maximum amount of total purchase amount in one day for any user in any day is 6486 CNY. The distribution of fifth and sixth column is hig hly right-skewed, which is why the median and 1st&3rd quantile value is zero for purchase times, and total money spent for each user on each day.

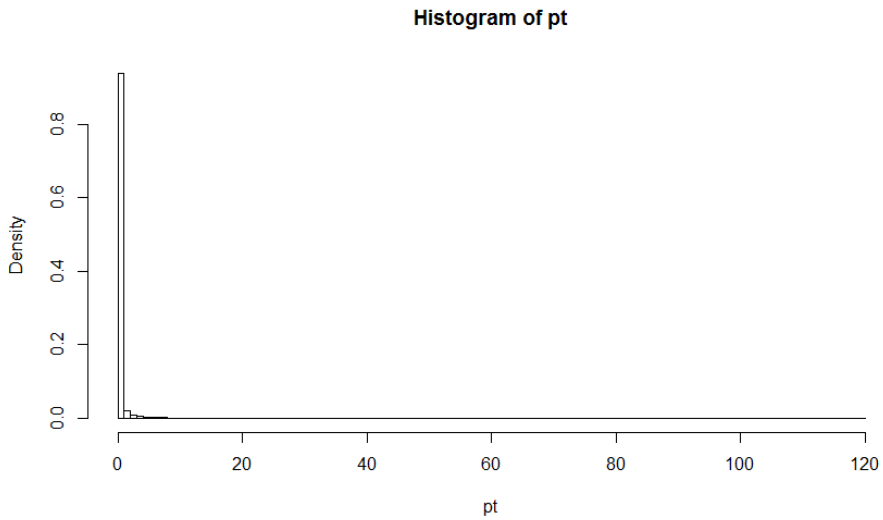
If we sum up the 30 days' activities for each user, we can get the statistics of total login times, pu rchase times, total purchase amount:

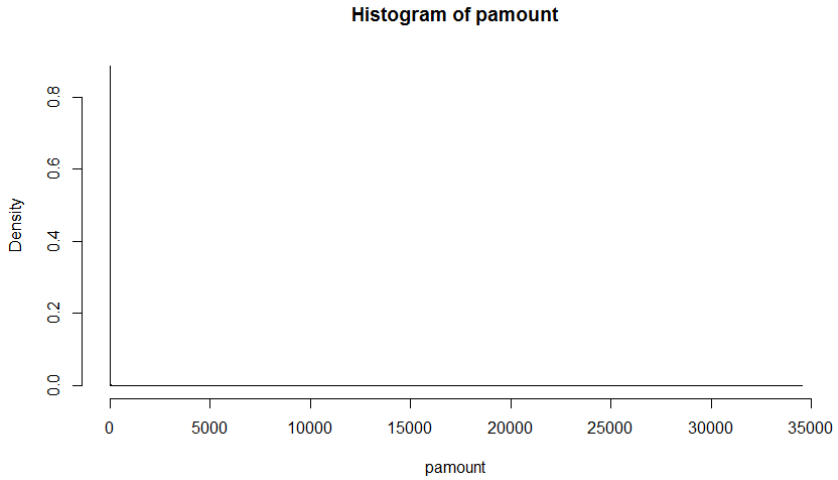
```
> summary(lt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     4.0    34.0   113.9   150.0  2063.0
> summary(pt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   0.627   0.000 118.000
> summary(pamount)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    0.00     0.00     0.00    35.25     0.00 34510.00
```

The statistics suggests that average user login onto the app 114 times on November, averaged 4 ti mes per day, average user spent 35.25 CNY in the app on November, averaged 1.175 CNY per d ay. All three vectors have median much lower than mean, which suggest highly right-skewed dist ribution. For purchase times and purchase amount, since the 3rd quantile is zero for both vectors, i t suggests that most of the users are window-shopping, and only a few users do purchase a lot.

Below is the distribution of total login times, purchase times, and total purchase amount in Nove mber 2015 for different users:



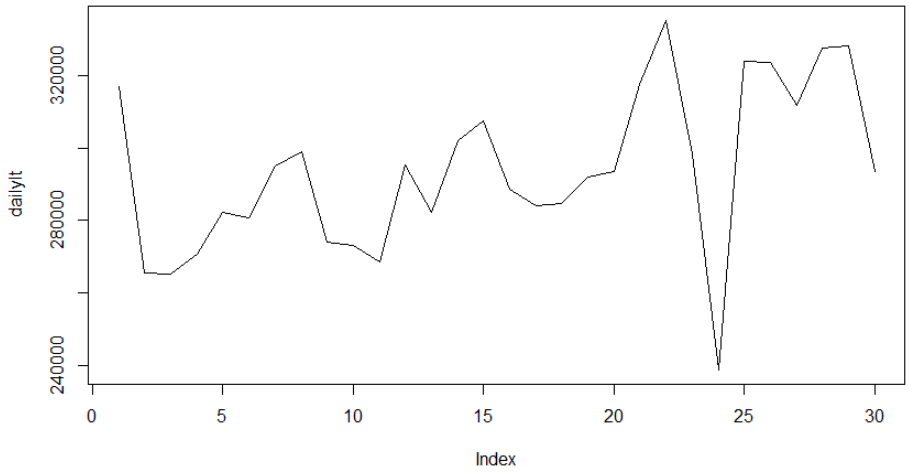




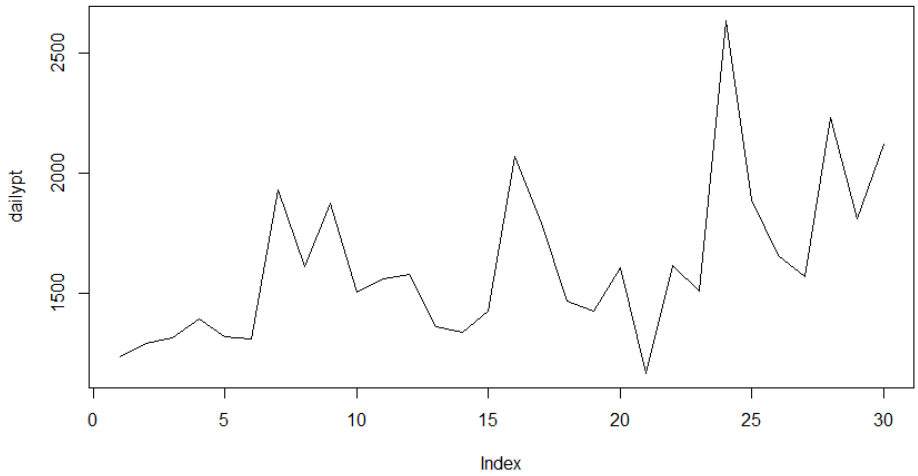
avlt: average login times per user per day
avpt: average purchase times per user per day
avpamount: average total purchase amount per user per day

```
> mdata<-cbind(avlt,avpt,avpamount)
> cov(mdata)
              avlt       avpt   avpamount
avlt      34.7938900 0.23187421  13.4905804
avpt       0.2318742 0.01344909   0.9631532
avpamount 13.4905804 0.96315318 186.4612606
```

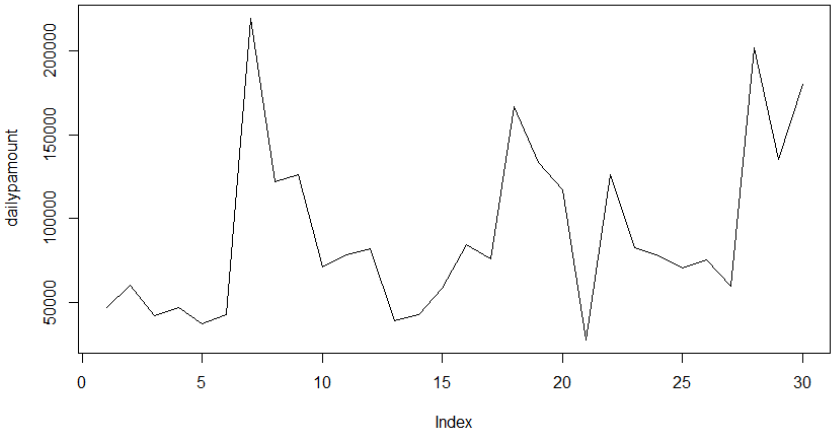
Above is the covariance matrix for three vectors.
If we sum up all users' activities for each day, we can get three crude time series for total login ti mes, purchase times, and total purchase amount for November.
Daily total login times:



Daily total purchase times:



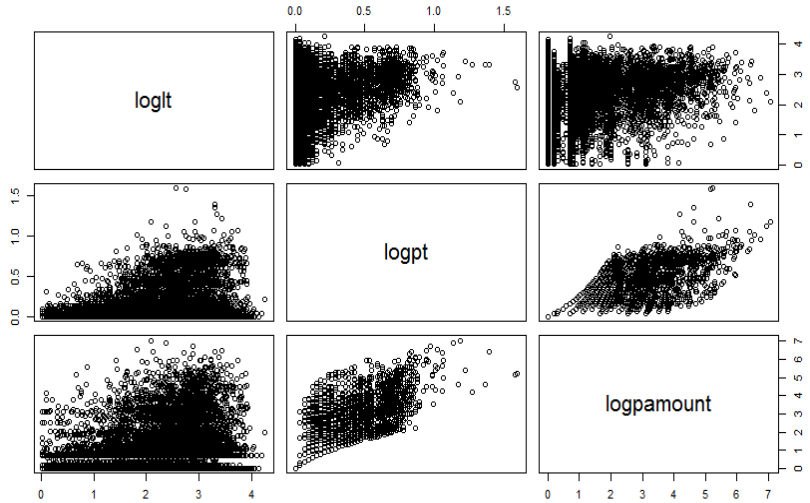
Daily total purchase amount:




```
> summary(dailylt)
```

Min. 1st Qu. Median Mean 3rd Qu. Max.

```
 238700  281100  293600  294000  310700  335300
> summary(dailypt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1165    1366    1564    1619    1806    2635
> summary(dailypamount)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  27540   49700   76710   91020  125100  219600

> logdata<-data.frame(loglt=log(avlt+1),
+                     logpt=log(avpt+1),
+                     logpamount=log(avpamount+1))
> pairs(logdata[,1:3])
```

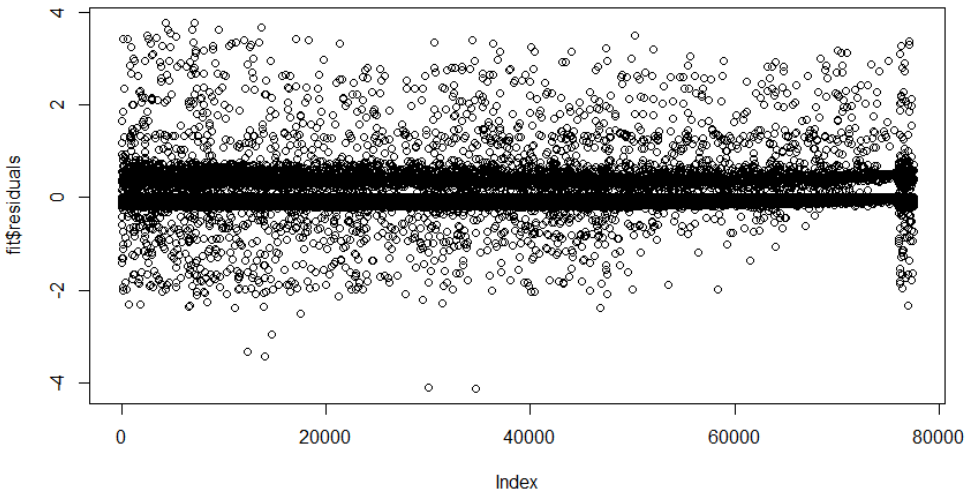


After log transformation, we can see there is a vague linear relationship between logpt and logpa mount, the reason is probably most users spent 6 CNY in one purchase times.

```
> fit<-lm(logpamount~loglt+logpt,data=logdata)
> summary(fit)

Call:
lm(formula = logpamount ~ loglt + logpt, data = logdata)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1252 -0.0778 -0.0157  0.0107  3.7681

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.01464    0.00141  -10.38   <2e-16 ***
loglt        0.06180    0.00104   59.40   <2e-16 ***
logpt        5.76454    0.01236  466.51   <2e-16 ***
---
Signif. codes:  0 ?**?0.001 ?*?0.01 ??0.05 ??0.1 ??1
```

Residual standard error: 0.2665 on 77471 degrees of freedom
Multiple R-squared: 0.7734, Adjusted R-squared: 0.7734
F-statistic: 1.322e+05 on 2 and 77471 DF, p-value: < 2.2e-16

> plot(fit$residuals)



We try to do a multiple linear regression to find the relationship between logpamount with loglt a nd logpt. The coefficients are significant and R-square is relatively high. However, the residue pl ot shows strong correlation between residues, so simple multiple regression won't work.

Conversion rate, which is number of purchase/number of login:

```
> cr=convrate(data[,4],data[,5])
> summary(cr)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.007634 0.045450 0.071430 0.119600 0.125000 7.000000
```

Purchase amount per purchase, which is total amount of purchase/number of purchase:

```
> app=convrate(data[,5],data[,6])
> summary(app)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   6.00    6.00    6.00   39.57   30.00  648.00
```

The median and 1st quantile equal to 6 means that more than half of purchase amount is 6 CNY p er purchase, distribution of app is right-skewed.

R code:

```
library(moments)
data=read.table("results.txt",header=F,sep = ",")
sumindividual=function(vec,t){
  l=length(vec)/t
  x=rep(0,l)
  for (i in 1:l){
    for (j in 1:t){
      x[i]=vec[t*(i-1)+j]+x[i]
```

```
    }
  }
  x
}
averagevec=function(vec,k){
  l=length(vec)
  x=rep(0,l)
  for (i in 1:l){
    x[i]=vec[i]/k
  }
  x
}
sumday=function(vec,t){
  l=length(vec)/t
  x=rep(0,t)
  for (i in 1:t){
    for(j in 1:l){
      x[i]=vec[t*(j-1)+i]+x[i]
    }
  }
  x
}
convrate=function(vec1,vec2){
  if (length(vec1)!=length(vec2)) stop("vectors' lengths do not match")
  l=length(vec1)
  y <- vector(mode="numeric", length=0)
  for (i in 1:l){
    if ((vec2[i]!=0)&(vec1[i]!=0)){
      x=vec2[i]/vec1[i]
      y=c(y,x)
    }
  }
  y
}
lt<-sumindividual(data[,4],30)
pt<-sumindividual(data[,5],30)
pamount<-sumindividual(data[,6],30)
avlt<-averagevec(lt,30)
avpt<-averagevec(pt,30)
avpamount<-averagevec(pamount,30)
dailylt<-sumday(data[,4],30)
dailypt<-sumday(data[,5],30)
dailypamount<-sumday(data[,6],30)
hist(lt,breaks = c(20*0:25, 1000,2100))
hist(pt,breaks = c(0:10,120))
hist(pamount,breaks=c(0:50,100,34510))
```

```
plot(dailylt,type="l")
plot(dailypt,type="l")
plot(dailypamount,type="l")
skewness(pt)
kurtosis(pt)
summary(data[,4])
summary(data[,5])
summary(data[,6])
summary(lt)
summary(pt)
summary(pamount)
summary(avlt)
summary(avpt)
summary(avpamount)
summary(dailylt)
summary(dailypt)
summary(dailypamount)
mdata<-cbind(avlt,avpt,avpamount)
cov(mdata)
logdata<-data.frame(loglt=log(avlt+1),
                    logpt=log(avpt+1),
                    logpamount=log(avpamount+1))
pairs(logdata[,1:3])
fit<-lm(logpamount~loglt+logpt,data=logdata)
summary(fit)
plot(fit$residuals)
cr=convrate(data[,4],data[,5])
summary(cr)
app=convrate(data[,5],data[,6])
summary(app)
```

8. Conclusion

In this project, we aim to predict future in-app purchase for each user based on historical user login, purchase and mobile device information. The paper focuses on the feature extraction and model ensemble. In the feature extraction section, we proposed a set of new derived features, which dramatically improve the overall performance of models. Then four different classification models are employed for the prediction task. Prediction results are evaluated in accuracy and all the models can achieve good performance. Other than that, we also developed a time series model to predict the company purchase as a whole.

9. Each Group Member's Contribution:

Our group consists of five members and their responsibilities are as follows.

1. Yanchi Liu: data preprocessing, feature extraction
2. Yueqing Zhang: data analysis with SVM and Logistic Regression
3. Gunjan Batra: data analysis with Random Forest and Adaboost
4. Yahui Guo: Statistics and Time Series Model
5. Cong Shi: Statistics and Time Series Model