Amazon Employee Access Control System

Shijian Tang, Jiang Han and Yue Zhang
Department of Electrical Engineering
Email: sjtang@stanford.edu

***Abstract*—In this work, based on the history data of 2010-2011 from Amazon Inc., we build up a system which aims to take place of resource administrators at Amazon. Our analysis shows that the given dataset is highly imbalanced with categorical values. Thus in the preprocessing step, we tried different sampling methods, feature selection as well as one hot encoding to make the data more suitable for prediction. In the prediction step, initially we tried single models which are suitable for categorical data like Naive Bayes, K-Nearest Neighbors (K-NN) and Decision Tree. Then due to the performance limitation, we further applied ensemble methods like random forest and gradient boosting. In addition, with one-hot encoding which transforms the categorical data into binary values, we are able to apply linear classifier and obtain satisfied performance. Finally, we ensemble three best prediction results from random forest, gradient boosting and logistic regression (with encoded data), and improved Area Under Curve (AUC) from initial 0.7338 (Decision Tree) to 0.8903.**

Keywords*—Imbalanced classification, categorical feature, one hot encoding, random forest, gradient boosting.

I. Introduction

An employee may need to apply for different resources during his or her career at the company. For giants like Google and Amazon, due to their highly complicated employee and resource situations, the application review process is generally done by different human administrators. In this project, based on the history data of 2010 to 2011 done by human administrators at Amazon Inc., we aim to build up an employee access control system, which automatically approve or reject employee's resource application.

Determining resource access privileges of employees is a popular real-world challenge for many giant companies like Amazon. When employees start to work, they first need to know what kinds of resources of the company they are or they are not supposed to get access to. The resources maybe very diverse, like computing resource and storage resource. It is supposed that employees fulfilling the functions of the same or similar roles should access the same or similar resources. Here we have the data set from a knowledgeable supervisor who takes time to manually grant the employee requests for resource access. For the purpose of saving money and time to allocate resources for the coming-and-going employees, we built up a predication model that automatically determine resource access privileges of employees.

In this problem, the data set is extremely imbalanced, where one of the classes ACTION = 0 has a significant less number of occurrences than the other ACTION = 1 (5-95 percent). Due to most of the learning algorithms design principle of minimizing the overall error rate to which the minority class contributes less, they always perform poorly in problems with imbalanced data set.

This paper can be organized as follows. We will first describe and preprocess the data set in sections II and III. Then, we will briefly introduce several types of models such as single classifiers described as well as ensemble models in section IV. In section V, we will present the prediction results for these models. At last we will summarize our work and draw conclusions in section VI.

II. Data Description

The data comes from Amazon Inc. collected from 2010-2011 (published on Kaggle platform) with training set of 32769 samples and testing of 58922 samples. As shown in table I, each of the data samples has one label attribute called "ACTION", where value "1" indicates this application is approved and "0" indicates rejection. In addition, each samples has eight features, which basically indicates different role or group of one employee at Amazon.

TABLE I. Data Feature Description

Feature Name	Feature Meaning
ACTION	"1": approved; "0": rejected.
RESOURCE	Resource ID
MGR_ID	ID of the employee's manager
ROLE_ROLLUP_1	Company role category ID1 (e.g. US Engineering)
ROLE_ROLLUP_2	Company role category ID2 (US Retail)
ROLE_DEPTNAME	Department description
ROLE_TITLE	Business title description
ROLE_FAMILY_DESC	Role family extended description
ROLE_FAMILY	Role family description (e.g. Retail Manager)
ROLE_CODE	Unique ID for each company role (e.g. Manager)

As for the evaluation metric, Receiver Operating Characteristic (ROC) curve is used to summarize classifier performance over tradeoffs between true positive and false positive error rates. And we use Area Under the ROC Curve (AUC) as a useful performance metric for imbalance classing problems.

III. Data Preprocessing

In this section, we did three parts of work, which are: balancing the dataset, one-hot encoding and feature selection.

A. Imbalanced dataset

As we described above, the given dataset is extremely imbalanced with the number of one class significantly lower than the other. In real life, many issues can be described as imbalanced classification problem, such as medical diagnose, text categorization, online resources management and so on.

Recently, the main approaches to solve the imbalanced classification is trying to balance the distribution between

TABLE II. ONE-HOT ENCODING EXAMPLE

Category Data	One-hot encoded eata
119433	001
118321	010
118278	100

TABLE III. FEATURE FREQUENCY VARIANCE

Feature Index	F_1	F_2	F_3	F_4	F_5
Variance(*10^6)	0.0003	0.0001	3.5665	0.3191	0.0137
Feature Index	F_6	F_7	F_8	F_9	
Variance(*10^6)	0.1244	0.0220	2.0396	0.1244	

minority and majority classes in training set [1],[2] to make the dataset suitable for standard machine learning models. These techniques includes oversampling, undersampling [1] and Synthetic Minority Over-sampling Technique (SMOTE) [2].

1) Random oversampling and undersampling: Oversampling typically refers to balance the data distribution by sampling the minority class data with replacement. On the other hand, undersampling changes the distribution of data by randomly removing the data in majority class. Although the performance will be improved by the above sampling methods, the shortcomings of oversampling and undersampling are obvious. Oversampling will result in overfitting, and undersampling may loss the importance information from dataset.

2) SMOTE: SMOTE is an oversampling method, which will generate synthetic training samples [2] instead of removing or duplicating raw data. The basic idea of SMOTE is that for each minority class sample, we create a synthetic example from some of the k nearest neighbors of that sample. Based on the number of new samples we need, we will randomly choose some neighbors among all the nearest neighbors. This process can be interpreted as choosing a random point in the line between two feature vectors as our new samples.

B. One-hot encoding

Since the original features are discrete category values which indicate different types. We can not directly apply linear classifier models on this kind of data. In order to apply linear classifier on this data, we need to use one-hot encoding.

One-hot encoding refers to bits that only have one single active 1 while all remaining bits are inactive 0s. In the given Amazon data, one feature may include multiple discrete categorical values. In order to apply linear classifier, it is necessary to separate those values with only one active at a time. Table II shows the example with one hot encoding on a feature with three categories, which will be encoded into "001", "010" and "100" separately. We see that this encoding method will expand the feature space from one to five. Sine most of the sample values are "0"s, we use sparse matrix to represent the newly encoded feature space.

C. Feature selection

Removing features with low variance is a common approach in feature selection [3]. In this work, we calculate the frequency variance of each feature based on Eq. (1):

$$Var(F_i) = \sum_{s_j} \frac{freq(s_j)}{sum} \left(freq(s_j) - \frac{sum}{|F_{i,uniq}|} \right)^2 \qquad (1)$$

where $s_j \in F_{i,uniq}$, and F_i is the i_{th} feature and $F_{i,uniq}$ is the set with all the unique feature values of F_i. Function $freq(s_j)$ indicates the frequency of s_j. $|F_{i,uniq}|$ is the size of $F_{i,uniq}$, i.e. the unique value number of feature F_i. In addition, $sum = \sum_{s_j} freq(s_j)$, which is the frequency sum of each s_j.

Table III shows the analysis result of frequency variance of all nine features, from which we can see that feature 1 and feature 2 have obviously small variance. Hence, we remove those two features in the further analysis steps.

IV. PREDICTIONS MODELS

A. Single classifier

In this section, we briefly introduce some single classification models we applied for initial testing. Here, "single" corresponds to the "Ensembling" in the next section. They are commonly used, thus we do not introduce the details here.

1) Naive Bayes: In this project problem, we treat all the features to be mutually independent. This "naive" independence assumption allows us to apply Naive Bayes algorithm in the given categorical data. Also, laplace smoothing is applied to those features never seen in the training. Thus, one sample is labelled with 1 when $P(1|F_1...F_N) > P(0|F_1...F_N)$ and vice versa.

2) K-NN: K-NN algorithm is also suitable for categorical data. It will label testing samples based on their nearest K neighbors in the training set. Since the given data is categorical, we use hamming distance instead of educlidean distance. Smaller hamming distance between test sample and specific training sample indicates they two are closer in the space. Besides, we combine the labels of K neighbors based on their hamming distance, closer neighbor will own higher influence.

3) Decision Tree: Decision tree builds classification models in the form of a tree structure. It breaks down a dataset into smaller subsets based on its features while an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A leaf node represents a classification or decision.

4) Logistic Regression: Logistic regression is a commonly used linear classifier, which is simple but relatively efficient.

B. Ensemble Methods

Ensemble method is a type of supervised learning, which employs a set of classifiers and their decisions are combined in certain way. Compared with single classifier, in the ensemble model, the output (decisions) depends on the vote of all the individual classifiers. Generally, the ensemble model containing bagging is also known as bootstrap aggregating [4] and boosting [6].

The main idea of bagging is that we generate several new training data set by randomly uniform sampling the original data set with replacement. Then use each new dataset to train a model, and the output equals to the dominate vote by all the trained models. Typically, we choose decision tree as each training model in many practical problems.

Boosting is an alternative ensemble method in machine learning. Compared with bagging, the basic rule of boosting is that by combining a set of weak learner properly, we can obtain a strong learner. The boosting algorithms are typically performed iteratively [6]. In each iteration, we add a new weak learner to the set of learners.

Among several boosting algorithms, the adaptive boosting (Ada Boosting) [6] and gradient boosting [7] are most popular.

In the subsection IV-B1, we will briefly describe the random forest method, which is developed based on bagging and will be adopted in our paper. Then, in subsection IV-B2, we will introduce the gradient boosting model in detail. We employ the above two models to solve our problem.

1) Random Forest: Random forest [5] combines the methods of bagging with random subspace. The main idea is that we build a set of decision trees not only depends on the random sampling of training data, but also randomly selecting the features when we building each tree. In prediction, we will takes all the decisions made by each tree into account and select the majority result as our prediction. In [5], each tree in the forest can be generated in the following three rules: First, randomly choose samples with replacement from original data set to form a new data set (same as bagging). Second, randomly choose a subset of features for each tree. And we split the nodes during building the tree based on this subset of features. The last is building the tree without pruning.

In [5], it reveals the fact that random forest model is able to prevent overfitting. That is as the increase of tree number, the generalization error will converge to an upper bound,

$$P_e \leq \rho(1-s^2)/s^2 \tag{2}$$

where P_e is the generalization error. ρ as the mean value of correlations and s is defined as the classifier set strength.

Based on Eq. (2) we find that the performance of random forest depends on the correlation between each tree. The performance will degrades as the increase of correlations. And the author of [5] also shows that the correlation of trees is mainly determined by the number of the features we selected when we split the nodes. The less number of features we choose, the trees will become more uncorrelated, and then the generalization error will decrease. The authors suggest that, we choose the $\sqrt{N}$ as the number of features we selected to build the tree, with N as the number of features in training dataset.

2) Gradient Boosting: The gradient boosting [7] is a type of boosting algorithms. Boosting process can be described as $F_i = F_{i-1} + \gamma_i h_i(x; a_i)$ where F_i is the learner in the i th iteration which may be a poor learner at this step. γ_i is a parameter to weight the new estimator $h_i(x; a_i)$, and a_i is the parameter in function h. In gradient boosting, the parameters γ_i and a_i are determined by minimizing the cost functions in current iteration,

$$(\gamma_i, a_i) = \arg\min_{\gamma, a} \sum_{j=1}^{m} L(y_j, F_{i-1}(x_j) + \gamma h_i(x_j; a)) \tag{3}$$

where x_j, y_j are the j th training samples with $j = 1, \cdots, m$. $h(x; a)$ is a simple functions added in the i th iteration with a as the parameters to be determined. $L(y_j, F_{i-1}(x_j) + \gamma h_i(x_j; a))$ is the cost function between the output y_j and the model in the i th iteration which is $F_i(x_j)$.

Solving Eq. (3) consists of two steps. The first step is determining a_i follows the expression as

$$a_i = \arg\min_{a_i, \rho} \sum_{j=1}^{m} r_{i,j} - \rho h(x_j; a) \tag{4}$$

with $r_{j,i}$ is the pseudo residuals with details in [7].

After finding the current estimator $h(x; a_i)$, we can further solve γ_i by,

$$\gamma_i = \arg\min_{\gamma} \sum_{j=1}^{m} L(y_j, F_{i-1}(x_j) + \gamma h_i(x_j; a_i)) \tag{5}$$

From Eq. (5), [7] points out that in the case of cost function is convex, the gradient boosting is actually equivalent to gradient descent in function space.

Based on the description above, the most significant parameters in gradient boosting are the selection of weak learners as well as the cost function. There are several candidates for cost function such as least squares and exponential functions. For week learner, the most popular choice is decision tree, where the gradient boosting model is often called gradient boosted decision trees.

In the following section, we will discuss how to implement the gradient boosting to solve our problem.

V. Prediction Result

In this section, we will show the simulation results with different prediction models. Firstly, we try single predication models which can be applied directly on categorical data like Naive Bayes, K-NN and decision tree. Secondly, since we found out that single classifier does not well on the given data, we further try ensemble models including random forest and gradient boosting based on decision tree. In addition, we show the AUC performance of logistic regression with one hot encoded data, which verifies the advantage of one hot encoding. In the final, we combine the three best prediction results from random forest, gradient boosting and one-hot encoding based logistic regression to obtain further performance improvement.

A. Single Model with Categorical Data

In this subsection, we present the prediction results based on different single models. In addition, different dataset with raw data, under-sampling, over-sampling and SMOTE are also taken into consideration for comparision.

Table IV shows the prediction results of Naive Bayes, KNN and Decision Tree. We can see that since the data size is big enough, the difference between training and test AUC is generally quite small. For different sampling method, we see that direct prediction on raw data can not work well due to the imbalanced data type. While under-sampling and SMOTE generally work better than over-sampling method, which mathes the conclusion drawn by [1].

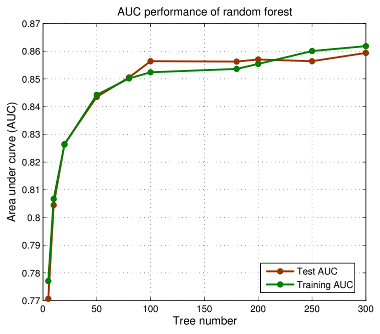

Fig. 1. The training and testing AUC for various tree numbers.

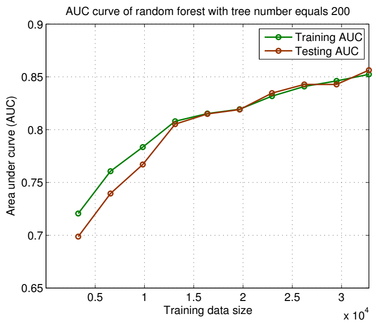

Fig. 2. The training and testing AUC for various training data set with tree number as 200.

Another important information conveyed by Table IV is that since the best test AUC performance is around 0.73 here, we may conclude that one single prediction model can not work well on the given problem and dataset. This makes us to think about using some upgraded ensembel method.

TABLE IV. AUC of Naive Bayes, K-NN and Decision Tree

	Raw Data	Under-sampling	Over-sampling	SMOTE
		Naive Bayes		
Training AUC	0.6186	0.6926	0.6497	0.7037
Test AUC	0.6027	0.6893	0.6512	0.7019
		K-NN		
Training AUC	0.5971	0.6469	0.6205	0.6590
Test AUC	0.5731	0.6365	0.6109	0.6620
		Decision Tree		
Training AUC	0.6538	0.7283	0.6870	0.7323
Test AUC	0.6235	0.6963	0.6687	0.7338

B. Ensemble model

1) Random forest: First, we will adopt the random forest model for prediction. As suggested in [5], the number of features we random selected for each tree is the root square of the total number of features. We use CART trees without prune, which is also implied in [5].

First we choose different number of trees to build our model. The training and testing AUC for different tree numbers from 10 to 300 are depicted in Fig. 1. The AUC for training data is the average AUC calculated from 10-fold cross validation from training data.

Fig. 1 demonstrates that as the increase of tree number, the performance of testing AUC gradually converges and does not degrade too much as the tree number increases, i.e., the overfit can be prevented. These results match with the conclusion we described in subsection IV-B1.

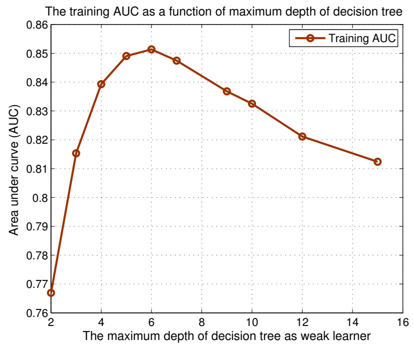

Fig. 3. The training AUC as a function of maximum depth of decision tree with 500 iterations.

From Fig. 1, we can observe that 200 is a good choice for the tree numbers, which will not degrade the AUC performance compared with the cases of less trees. On the other hand, it is also not necessary to choose too large number of trees. The time complexity will increase but the performance will change sightly. Therefore we choose the tree number as 200 in our further prediction.

In Fig. 2, we plot the learning curve characterized by AUC versus the size of training data set. For the training AUC, we use 10-fold cross validation to calculate the average AUC for each size of training data. The learning curve implies that as we enlarge the training data set, the training and testing AUC converge to an expected value 0.8558. This means that random forest is a good model for this problem and no overfitting yields.

2) Gradient boosting: In this subsection, we will discuss the results of gradient boosting. We choose the decision tree as weak learner. Recall the discussion in subsection IV-B2, we should select the parameters in the tree to specify the weak learner. Typically there does not exit a general method to determine the parameters such as maximum depth and minimum sample splits in the decision tree. They depends on the practical problems. In [7], the authors suggest that the depth of tree is more important than other parameters.

Therefore, we go through several maximum depth from 2 to 15 over 500 iterations. Note that we have test that for this specific data, after 500 iterations, the performance of model will converge. The training AUC based on 10-fold cross validation for each maximum depth has been plotted in Fig. 3. Fig. 3 indicates that the optimal depth locates at 6. Next, we consider the training and testing AUC based on the weak learner we just found. By increasing the size of training data, the training AUC based on 10-fold cross validation and the testing AUC are illustrated in Fig. 4. Similar with other models we have implemented. The AUC curves will converge to an expected value (AUC=0.8514 for full training data set).

Finally, let us compare the results from two ensemble models: random forest and gradient boosting. We find that for large training data size, the performance of random forest are competitive with gradient boosting, with testing AUC as

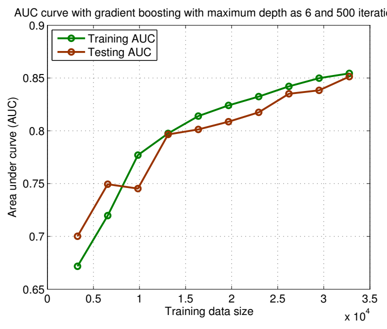

Fig. 4. The training and testing AUC with gradient boosting with maximum depth as 6 and 500 iterations.

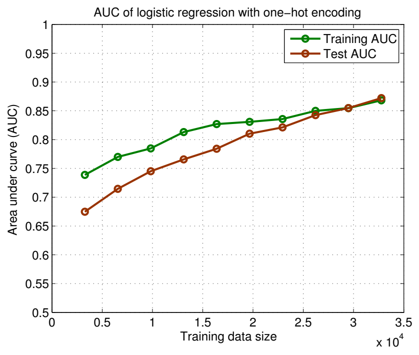

Fig. 5. AUC of logistic regression with one-hot encoding

0.8558 for random forest and 0.8514 for gradient boosting. This conclusion matches the experimental results from [5] where the authors conclude that the performance of random forest will be competitive as boosting. However, from the view of practical problems, the random forest will beat the gradient boosting in some aspects. First, the random forest is more flexible since we do not need to select the weak learner which will cost additional time in boosting. Second, the random forest runs much faster than gradient boosting especially when weak learner depth is large. Therefore, to make our model more flexible and efficient, random forest is a better choice.

C. Linear classifier with one-hot encoding

In section III, we mentioned that linear classifier can not be applied directly to the raw data due to the categorical feature values (with AUC for SVM around 0.5). In this subsection, we apply one-hot encoding to transform the raw feature into binary feature values and thus expand the feature space. After one-hot encoding, with only one unique feature value active in each column, we can apply linear classifier in new feature space.

Here, we choose the logistic regression to fit the model. Fig. 5 shows the prediction result of logistic regression with one-hot encoding. Different data size is chosen here to show the difference between training and test AUC performance. From Fig.5, we can see that after one-hot encoding, the best test AUC value can achieve 0.8721, which shows the great advantage of one-hot encoding. Also, note that with training data size increase training and test AUC performance converges to very close values.

D. Further Ensemble

Until now, we have three relatively better prediction results, which are: random forest (test AUC = 0.8558 with tree number of 100), gradient boosting (test AUC = 0.8514) and logistic regression with one-hot encoding (test AUC = 0.8721). In this subsection, we will ensemble those different prediction models for further performance imporvement.

Roughly consider the above three models are mutually independent, and since they have similar AUC performance, let's roughly take their test error as ε. For three classifiers with binary classification, their ensemble model will make incorrect prediction only when more than two of them are incorrect.

$$\varepsilon_{ensemble} = C_3^2\varepsilon^2(1-\varepsilon) + C_3^3\varepsilon^3 = 3\varepsilon^2 - 2\varepsilon^3 \qquad (6)$$

here, $\varepsilon_{ensemble}$ is the prediction error rate of ensemble model. Solve inequation $\varepsilon_{ensemble} > \varepsilon$ and get $\varepsilon > 0.5$. Thus by ensembling those three models, we generally get better result.

In this last step, we combine the prediction results of random forest, gradient boosting and logistic regression with one-hot encoding. By getting the weighted (training AUC as the weight) average value of the three models, we finally improve the test AUC result to 0.8903.

VI. Conclusion and Future Work

In this work, based on the history data from Amazon Inc. We built up an automatic system to review employee's application on resources. In the preprocessing, low variance feature remove method is used for feature selection. We also applied different sampling methods and one-hot encoding to make the data balanced and suitable for linear classifier. In the prediction step, we tried single models on categorical data, verified one-hot encoding with logistic regression. After that, we also applied ensemble methods like random forest, gradient boosting and best three ensemble to further improve AUC. The AUC performance finally achieves 0.8903 from the initial 0.7338 by single decision tree. In the future, we plan to apply feature grouping strategy, use more linear models with one-hot encoding data and try different boosting models for performance improvement.

References

[1] N. Japkowicz. "Class imbalance: Are we focusing on the right issue," in Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets, 2003.

[2] N. Chawla, L. Hall, K. Bowyer, and W. Kegelmeyer. "SMOTE: Synthetic Minority Oversampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," Journal of Machine Learning Research, vol. 3, pp. 1157-1182, 2003.

[4] L. Breiman, "Bagging predictors," Machine Learning, vol. 26, no. 2, pp. 123-140, 1996

[5] L. Breiman. "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[6] M. Collins, R. Schapire and Y. Singer, "Logistic regression, AdaBoost and Bregman distances," Machine Learning, vol. 48, no. 1-3, pp. 253-285, 2002.

[7] J. Friedman, "Stochastic gradient boosting," Computational Statistics & Data Analysis, vol. 38, no. 4 pp. 367-378, 2002.