Predict Employees' Computer Access Needs in Company

Xin Zhou & Wenqi Xiang
Email: xzhou15,wenqi@stanford.edu
1.Department of Computer Science 2.Department of Electrical Engineering

***Abstract*—When an employee is hired by a company, given his/her job role, model can be built to predict an employee's access needs. This auto-access model tries to minimize the human labor and time for granting or revoking employee access. The input to our algorithm is information about the employee's role at the time of approval. We then use Logistic Regression, Naive Bayes and Support Vector Machine (SVM) models to output a predicted ACTION for a specific resource. While, due to the unique characters of our data set, with only 6% samples labelled as 0, simply applying models like Logistic Regression, Naive Bayes and SVM can not get a satisfactory prediction on this unbalanced dataset. Alternatively, we use the One-hot Encoder to preprocess the data before applying different models and achieve a much higher accuracy (around 87%) in labelling computer access needs.**

Keywords*—Unbalanced Prediction, Logistic Regression, SVM, Naive Bayes, One-hot Encoder, Cross Validation, Greedy Feature Selection.

I. Introduction

When an employee starts work at a company, he or she first needs to get access to computer resources, like applications and web portals, to fulfill their role. During their daily work, employees may encounter roadblocks when trying to get access they need but not granted. (e.g. not able to log into a reporting portal). In order to overcome access grant obstacles, it takes time to manually grant the needed access. Especially when employees move throughout a company, regranting their access needs wastes a nontrivial amount of time and money. To solve this problem, it's important to build a model on historical data to automatically determine an employee's access needs, which minimizing the human involvement required to grant or revoke employee access. Our project uses different machine learning models to achieve this goal and compare the performance among these models.

II. Related Work

The unbalance problem can cause suboptimal classification performance[5], which is pervasive and prevalent in many applications including risk management, text classification, and medical diagnosis/monitoring. In recent years, lots of researchers have made efforts to solve this problem both at the data and algorithmic levels. At the data level, different forms of resampling are applied. Batista et. al [6] compared different sampling strategies and their combinations. They present that it is applicable to combine focused over and under sampling. On the other hand, using recognition-based instead of discrimination-based learning, adjusting the costs of the various classes and changing the probabilistic estimation at the tree leaf in the decision tree are all good ways related to algorithmic level. B. Zadrozny and C. Elkan[7] gave a more flexible and direct method to the treatment of different parts of the decision space by adjusting the probabilistic estimation within cost-sensitive learning. Additionally, more accurate measures such as Receiver Operating Characteristic curves, Cost Curves[8] [9] [10] [11] and g-means metric[3] are applied. What's more, existing feature selection methods are not very applicable for unbalanced data sets. Instead, Zheng and Srihari[12] proposed a new feature selection approach, which separately selects features for positive and negative classes and then combines them.

III. Dataset and Features

A. Original Dataset

The data from Kaggle website (https://www.kaggle.com) consists of real historical data collected from 2010 & 2011. There are around 30000 samples in our data set. We use 70% of them for training, and the rest for testing. Employees are manually allowed or denied access to resources over time. That is to say, Action (y) is binary, to be 1 if the resource was approved or 0 if the resource was not. For each employee, there are 9 non-negative numeric features, which are:

1. # resource	2. # manager_ID	3. #role_rollup_1
4. # role_rollup_2	5. # department_name	6. #role_title
7. # role_family_description	8. # role_family	9. #role_code

For instance, employee A has the following information,{action, resource, manager_ID, role_rollup_1, role_rollup_2, department_name, role_title, role_family_description, role_family, role_code} is {1, 39353, 85475, 117961, 118300, 123472, 117905, 117906, 290919, 117908}, 1 means that the employee can use resource 39353. Other numbers stand for information about the employee.

There are many copies for each feature, which is corresponding to the real world. For example, two employees may have different role_title, but both are in the same department, then they have the same department_name.

B. Unbalanced Dataset

Unbalanced dataset means that, in a classification problem, instances of some classes are extremely more than others. In our dataset, there are 96% of samples labelled to be 1, and only 4% of them labelled to be 0. The unbalanced datasets make classifiers generally perform poorly on prediction because they are based on training examples and output the simplest hypothesis that best fits the data[1], that is to say, the simplest hypothesis is often the one that classifies almost all instances as positive. What's more, for our project, the negative instances can be treated as noise and then be ignored completely by the classifier. A popular approach towards solving these problems is to bias the classifier, which pays more attention to the negative instances. This can be done, for instance, by increasing the penalty associated with misclassifying the negative class relative to the positive class. Another approach is to preprocess the data by oversampling the majority class or undersampling the minority class and then to create a balanced dataset. For our project, these data preprocessing methods can not provide a satisfactory performance, so we apply another data preprocessing approach which significantly improve the train and test accuracy.

C. One-hot Encoder

Due to pretty low percentage of samples labelled as 0, building model on the original dataset will get very high and constant test accuracy (around 96%), but it's useless because it never makes a prediction to be labelled as 0. To tackle this problem, One-hot Encoder is applied to do the data preprocessing before using any models.

One-hot Encoder generates a sparse matrix for each feature. For example, all values for a certain feature may be 3, 2, 3, 1, 1, 10, 6, 6,..., and the largest number is 10. Our feature vector is $[3 \;\; 2 \;\; 3 \;\; 1 \;\; 1 \;\; 10 \;\; 6 \;\; 6...]^T$, and the number of columns for its sparse matrix would be equal to the value of the largest number, namely 10. we get its sparse matrix as follows:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ :::: & & & & & & & & & \end{bmatrix} \tag{1}$$

Many machine learning algorithms learn a single weight per feature. Assume a dataset only contains a single categorical feature "department", with values "Marketing", "Services" and "Inventory". Like our dataset, without loss of generality, they are encoded as 0, 1 and 2. It then has a weight w for this feature in a linear classifier, which will make a prediction based on $w * x + b > 0$, or equivalently $w * x < b$.

The problem now is that the weight w cannot encode a three-way choice. The three possible values of $w * x$ are 0, w and $2w$. Either these three all lead to the same prediction or "Services" and "Marketing" lead to the same prediction, or "Marketing" and "Inventory" have the same prediction. It's no possible for the model to learn that "Services" and "Inventory" should be given the same label, with "Marketing" the odd one out.

One-hot Encoder effectively blows up the feature space to three features in this case, and each of them will get their own weights, so the decision function is now $w[Marketing] * x[Marketing] + w[Services] * x[Services] + w[Inventory] * x[Inventory] < b$, where all the x's are booleans. In this space, such a linear function can express any sum/disjunction of the possibilities (e.g. " Marketing or Services ", which might be a predictor for someone having access 1).

After One-hot Encoder, we apply Logistic Regression, Naive Bayes and SVM to make the prediction.

IV. METHODS

A. Feature Section: Greedy Feature Selection

It is reasonable to suppose that there is only a small number of features that are relevant. To avoid the potential overfitting problem for our prediction, we can use forward search combined with a certain cross validation to do the feature selection. While for our learning task, we decide to use another approach of feature selection.

The dataset includes 9 features, and each of them corresponds to different resources, departments, and etc to an employee. In Reality, different features have inner relationship to each other. The next step is to find the optimal feature combination to build the model. We use a greedy feature selection algorithm to achieve this goal. In the greedy feature selection algorithm, for each iteration of prediction, we apply the cross validation algorithm to the training data, where 90% of data examples are for training and the rest are for testing, to evaluate the current feature sets. Firstly, we choose each of these 9 features to do prediction and estimate the test accuracy over ten times. Then, we compare the 9 test accuracies and choose the feature whose test accuracy is the best one. This chosen feature is combined with the other eight features one by one to do the prediction again. These 8 pairs of features are paralleled to 8 prediction accuracies and again we choose the pair corresponding to the highest accuracy. Similarly, we use this chosen pair to combine with the rest 7 features to get the combination of three features. Eventually, we can get the best combination of features.

B. Cross Validation

We use K-fold cross validation during each iteration of the greedy feature selection algorithm, where K = 10. The process is as follows: Firstly, we randomly split the dataset into k disjoint subsets of m/k training examples each. Lets name these subsets as $S_1, \ldots, S_k$. Secondly, for each model M_i, we evaluate it as follows: For j = 1, . . . , k, train the model M_i on $S_1 \bigcup S_j \bigcup S_{j+1} \bigcup S_k$ (i.e., train on all the data except S_j) to get some hypothesis h_{ij}. Then, test the hypothesis h_{ij} on S_j to get $\hat{\varepsilon}_{S_j}(h_{ij})$. The estimated generalization error of model M_i is then calculated as the average of the $\hat{\varepsilon}_{S_j}(h_{ij})$s (averaged over j). Finally, pick the model with the lowest estimated generalization error, and retrain that model on the entire training set S. The resulting hypothesis is then output as our final answer.

C. Logistic Regression

This project is a classification problem, and we use Logistic Regression to build the model. We choose sigmoid function as the hypothesis.

$$y = h_\theta(x) = g\left(\theta^T x\right), \theta, x \epsilon R^9 \qquad (2)$$

Given the Logistic Regression model, we use Newtons method to fix θ. The updated rule for θ is:

$$\theta := \theta - H^{-1} \bigtriangledown_\theta \, l(\theta) \qquad (3)$$

Here, $\bigtriangledown_\theta l(\theta)$ is the the vector of partial derivatives of $l(\theta)$. H is Hessian matrix, an n-by-n matrix (if we include the intercept term, H will be an (n + 1)-by-(n + 1) matrix. where,

$$\mathrm{H}_{ij} = \frac{\partial^2 l(\theta)}{\partial\theta_i \partial\theta_j} \qquad (4)$$

D. SVM

We use SVM to make the prediction. This is implemented by skilearn.svm.SVC, a library offered online at http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. This implements an SVM using a linear kernel. The optimization is as follows:

$$\min_{\gamma w b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^m \xi_i$$

$$\text{s.t.} y^i \left(w^T x^{(i)} + b\right) \geq 1 - \xi_i, \text{ i=1,2,...,m}$$

$$\xi_i \geq 0 \text{ i=1,2,...,m} \qquad (5)$$

E. Naive Bayes

In our prediction task, inputs are discrete-valued and the prediction is a classification problem. So, we can apply Naive Bayes. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes theorem with Naive Bayes (NB) assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_i through x_n Naive Bayes theorem can be stated as follows:

$$P(y|x_1, x_2\cdots, x_n) = \frac{P(y)\,P(x_1, x_2\cdots, x_n|y)}{P(x_1, x_2\cdots, x_n)} \qquad (6)$$

Using the Naive Independence Assumption that:

$$P(x_i|y, x_1, \cdots x_{i-1}, x_{i+1}\cdots, x_n) = P(x_i|y) \qquad (7)$$

for all i, this relationship is simplified to:

$$P(y|x_1, \cdots, x_n) = \frac{P(y)\prod_{i=1}^n P(x_i|y)}{P(x_1, \cdots, x_n)} \qquad (8)$$

Since $P(x_1, \cdots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y|x_1, x_2\cdots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \qquad (9)$$

So, we can get that:

$$\hat{y} = \operatorname{argmax}_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{10}$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i|y)$; the former is then the relative frequency of class y in the training set. The different Naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$

In spite of their apparently over-simplified assumptions, Naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.

Bernoulli NB implements the Naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions, and it requires samples to be represented as binary-valued feature vectors. The decision rule for Bernoulli Naive Bayes is based on:

$$P(x_i|y) = P(i|y)\, x_i + (1 - P(i|y))(1 - x_i) \tag{11}$$

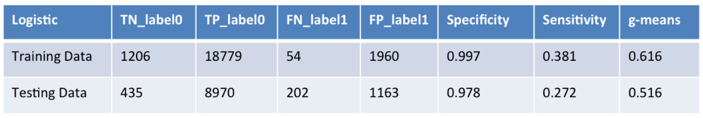
V. Experiments/Results/Discussion

Using accuracy as a metric to evaluate different classifiers on highly unbalanced dataset is virtually useless. For example, an unbalanced dataset with 94 samples to labelled 1 (y=1) and 6 samples to be labelled 0 (y = 0), a classifier that classifies everything to be 1 will be 94% accurate, but it will be completely useless as a classifier since it never really make a prediction. To use a more reasonable evaluation on the performance of various models, two metrics are introduced, the sensitivity and the specificity. For our prediction task, sensitivity can be defined as the accuracy on the negative instances (y=0) (true negatives / (true negatives + false positives)), while specificity can be defined as the accuracy on the positive instances (y=1) (true positives / (true positives + false negatives)). Kubat et al [3] suggested the g-means metric defined as:

$$g = \sqrt{acc^{+}.acc^{-}} \quad [4] \tag{12}$$

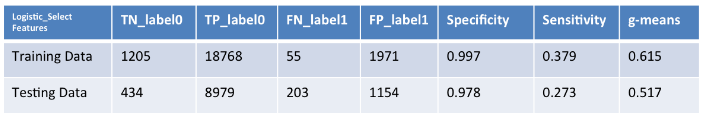
Here, acc^{+} is sensitivity and acc^{-} is specificity. This metric has been used by several researchers for evaluating classifiers on unbalanced datasets[3,4]. We will also use this metric to evaluate our classifiers and compare the performances of Logistic Regression, Naive Bayes and SVM.

After using One-hot Encoder to preprocess our unbalanced dataset, Logistic Regression model is applied to test the specificity, sensitivity and g-means. The table results for Logistic regression model in Figure 1 show seven metrics for both the training data and the testing data: true negative (TN), true positive (TP), false negative (FN), false positive (FP), the specificity, the sensitivity, and the g-means.



Logistic	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1206	18779	54	1960	0.997	0.381	0.616
Testing Data	435	8970	202	1163	0.978	0.272	0.516

Fig. 1. Table Results for Logistic Regression



Logistic_Select Features	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1205	18768	55	1971	0.997	0.379	0.615
Testing Data	434	8979	203	1154	0.978	0.273	0.517

Fig. 2. Table Results for Logistic Regression with combined good features

As we see, the specificity is pretty high (above 0.97) for both the training dataset and the testing dataset, but the sensitivity is much lower (around 0.27 for the testing dataset) since the number of false positive is much larger than the number of true negative based on our unbalanced dataset. Even though in this case, accuracy for the true negative is improved to 100% by the model, the sensitivity can only achieve to 0.35 for the testing dataset.

The dataset contains 9 features and each feature corresponds to the employees' relative information in the company, so the mutual relationship among these features can influence the computer access. To optimize forward feature selection algorithm, we use a greedy feature search algorithm which searches a single good feature, then a combination of two good features, and so on. Furthermore, for each iteration, we use the cross validation algorithm to evaluate the current feature sets. The table results using Logistic regression with the good features combination for the same seven metrics is shown in Figure 2. It indicates after using greedy feature search algorithm, it slightly improves the performance.

We also apply Naive Bayes model to our preprocessed dataset, and the table results for the same seven metrics are shown in Figure 3 and Figure 4. The g-means is decreased when only applying Naive Bayes model compared to Logistic Regression Model, but increased when using Naive Bayes model with combined good features.

To further evaluate the performance of our models, we also plot the accuracy for both the large sample set (y=1), small sample set (y=0), and the overall accuracy. These three accuracy metrics for both Logistic and Naive

Naïve Bayes	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1035	19781	225	958	0.989	0.519	0.716
Testing Data	346	9208	291	925	0.969	0.272	0.513

Fig. 3. Table Results for Naive Bayes

NB_Select Features	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1060	20086	200	653	0.990	0.619	0.782
Testing Data	329	9392	308	761	0.968	0.302	0.541

Fig. 4. Table Results for Naive Bayes with combined good features

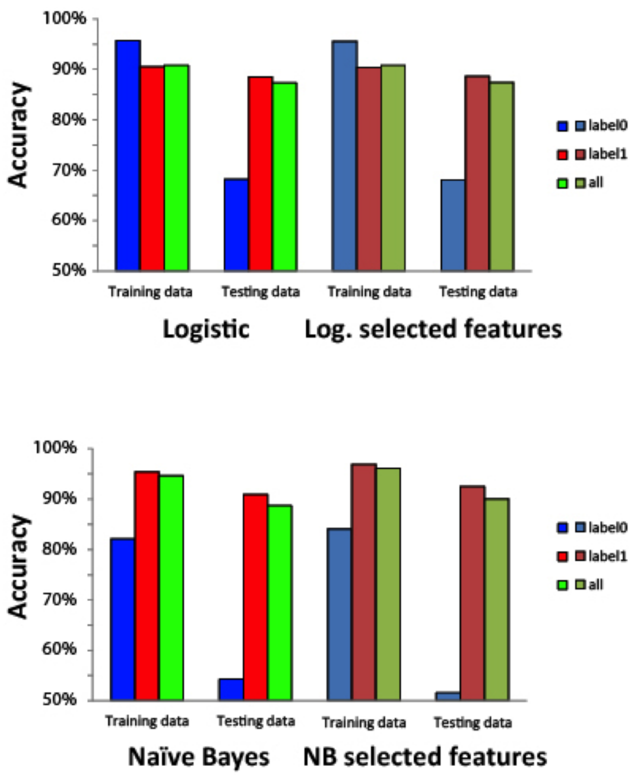


Fig. 5. Accuracy Results for Logistic regression and Naive Bayes

Bayes models are shown in Figure 5. As we see, when using Logistic Regression model, the training accuracy for the large or small sample set is pretty high (above 90%), and the test accuracy for the large sample set is around 88%, but only around 68% for the small sample set. The overall test accuracy is around 87%.

Compared to test accuracy results in Logistic Regression model, test accuracy for the small sample set (around 54%) is much lower when using Naive Bayes model even though the test accuracy for the large sample set is slightly increased to 91%.

Lastly, we apply the SVM model to our preprocessed dataset, the performance including g-means and the three

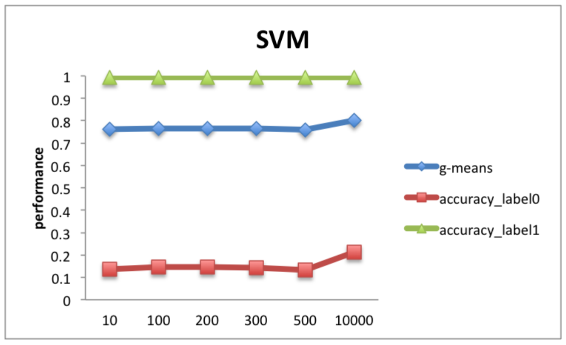


Fig. 6. Performance for SVM

accuracy metrics are show in Figure 6. The x axis shows different C regularization parameters we choose for the model. It shows the g-means is increased up to 0.76 which is higher then the previous two models, but the test accuracy for the small sample set is decreased to 12%-21% and the test accuracy for the large sample set is almost achieved to 100%. The kernel For this Figure is Radial Basis Function (RBF), and we also test other kernels which generate similar results (not shown here).

VI. Conclusion/Future Work

Simply applying models like Logistic Regression, Naive Bayes and SVM can only achieve low test accuracies (in our project, they only predict 1 or randomly predict 0 or 1 that is shown in the milestone report) in an unbalanced set and that the use of On-hot encoder to preprocess dataset can greatly improve their performance. From the above results, we can conclude that the best overall test accuracy we can achieve is around 94% when using the SVM model (g-means 0.76 is also the highest in this case), but the best test accuracy for the small sample set is only around 21%. To balance the test accuracy between the large and small sample sets, the Logistic Regression model with combination of good features give us the best prediction. The test accuracy for the small and large sample set is around 68% and 88% respectively. The overall test accuracy is achieved to 87%. To further improve the test accuracy, the future work we can do is combining two/three/more features together and hash them to be a single new feature. In this way, it considers some features together to make a prediction which indicates in reality, some information together about the employee is better to predict the computer access.

References

[1] Akbani R, Kwek S, Japkowicz N. Applying support vector machines to imbalanced datasets[M]//Machine Learning: ECML 2004. Springer Berlin Heidelberg, 2004: 39-50.

[2] Aha, D. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal Man-Machine Studies, 36, 267-287.

[3] Kubat, M. Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One- Sided Selection. Proceedings of the 14th International Conference on Machine Learning.

[4] Kubat, M., Holte, R. Matwin, S. (1997). Learning when Negative Examples Abound. In Proceedings of ECML-97, 9th European Conference on Machine Learning.

[5] Chawla N V, Japkowicz N, Kotcz A. Editorial: special issue on learning from imbalanced data sets[J]. ACM Sigkdd Explorations Newsletter, 2004, 6(1): 1-6.

[6] Batista G E, Prati R C, Monard M C. A study of the behavior of several methods for balancing machine learning training data[J]. ACM Sigkdd Explorations Newsletter, 2004, 6(1): 20-29.

[7] Zadrozny B, Elkan C. Learning and making decisions when costs and probabilities are both unknown[C]//Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001: 204-213.

[8] Provost F, Fawcett T. Robust classification for imprecise environments[J]. Machine learning, 2001, 42(3): 203-231.

[9] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. http://www.hpl.hp.com/personal/Tom Fawcett/papers/ index.html, 2003.

[10] C. Drummond and R. Holte. Explicitly representing ex- pected cost: An alternative to ROC representation. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 198207, 2001.

[11] J. Furnkranz and P. Flach. An analysis of rule evalua- tion metrics. In Proceedings of the Twentieth Interna- tional Conference on Machine Learning, pages 202209, 2003.

[12] Z. Zheng, X. Wu, and R. Srihari. Feature selection for text categorization on imbalanced data. SIGKDD Ex- plorations, 6(1):8089, 2004.