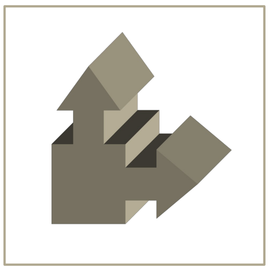
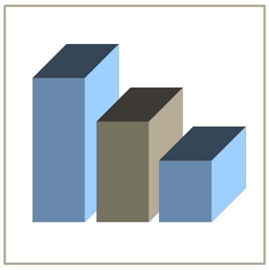



# Amazon.com Employee Access Challenge

**Author : Parag Shah**

MAT 460 Topics in Statistics
Data Mining

## Contents

# Introduction

## Background

The aim of this project is to predict employees' access needs given his or her job role in company. This project was started by amazon and hosted on competition site kaggle.com.

## Project Statement

When an employee at any company starts work, they first need to obtain the computer access necessary to fulfill their role. This access may allow an employee to read/manipulate resources through various applications or web portals. It is assumed that employees fulfilling the functions of a given role will access the same or similar resources. It is often the case that employees figure out the access they need as they encounter roadblocks during their daily work (e.g. not able to log into a reporting portal). A knowledgeable supervisor then takes time to manually grant the needed access in order to overcome access obstacles. As employees move throughout a company, this access discovery/recovery cycle wastes a nontrivial amount of time and money.

There is a considerable amount of data regarding an employee's role within an organization and the resources to which they have access. Given the data related to current employees and their provisioned access, models can be built that automatically determine access privileges as employees enter and leave roles within a company. These auto-access models seek to minimize the human involvement required to grant or revoke employee access.

## Objective

The objective of this competition is to build a model, learned using historical data, that will determine an employee's access needs, such that manual access transactions (grants and revokes) are minimized as the employee's attributes change over time. The model will take an employee's role information and a resource code and will return whether or not access should be granted.

## Data Description

The data consists of real historical data collected by amazon between 2010 and 2011. Employees were manually allowed or denied access to resource over time.

The data was divided in two sets, training and testing. Training set has 32769 instances and testing set has 58921 instances. Each instance has ACTION (target variable), RESOURCE and information about employee's role at the time of approval. There are 9 attributes associated with employee and 1 target variable.

| Variable Name | Description |
|---|---|
| ACTION | ACTION is 1 if the resource was approved, 0 otherwise |
| RESOURCE | An ID for each resource |
| MGR_ID | The EMPLOYEE ID of the manager of the current EMPLOYEE ID record; an employee may have only one manager at a time |
| ROLE_ROLLUP_1 | Company role grouping category id 1 (e.g. US Engineering) |
| ROLE_ROLLUP_2 | Company role grouping category id 2 (e.g. US Retail) |
| ROLE_DEPTNAME | Company role department description (e.g. Retail) |
| ROLE_TITLE | Company role business title description (e.g. Senior Engineering Retail Manager) |
| ROLE_FAMILY_DESC | Company role family extended description (e.g. Retail Manager, Software Engineering) |
| ROLE_FAMILY | Company role family description (e.g. Retail Manager) |
| ROLE_CODE | Company role code; this code is unique to each role (e.g. Manager) |

Here ACTION is target variable which is binary and has two levels, 0 and 1. Thus task is to build model using above data to correctly classify and predict the ACTION taken on resource.

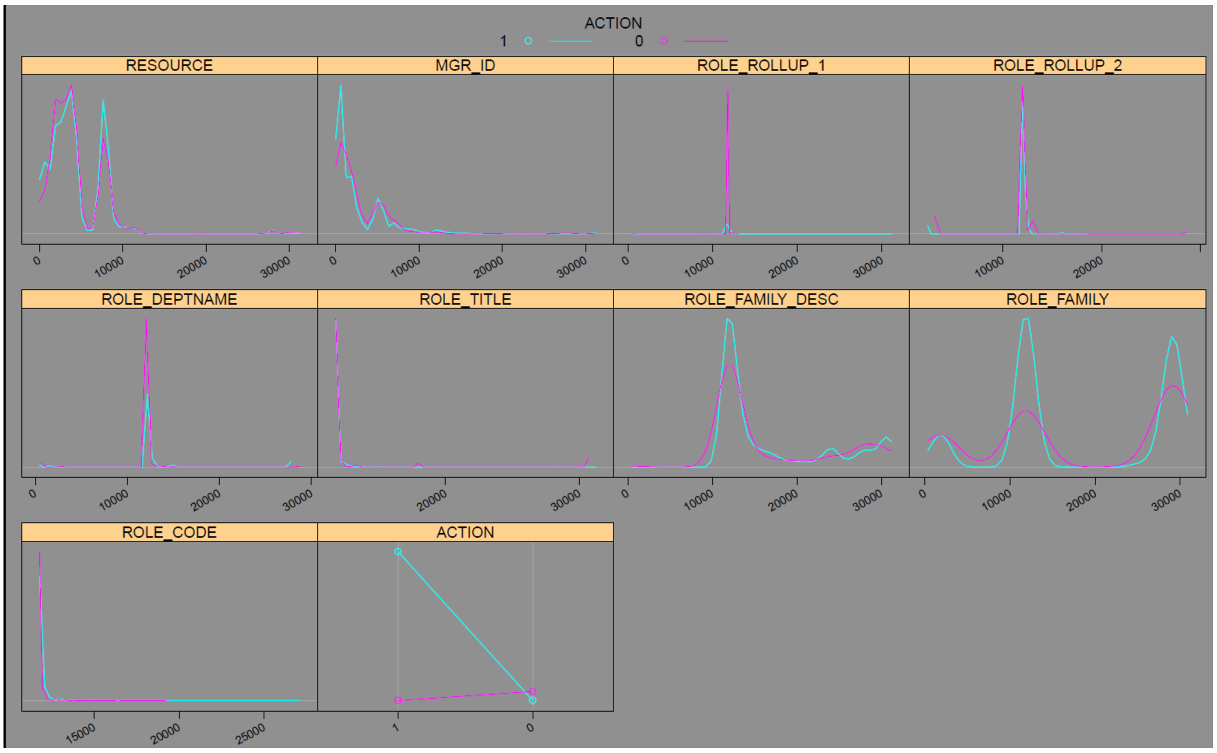
# Descriptive Statistics

No of Unique instances and Median of attributes:

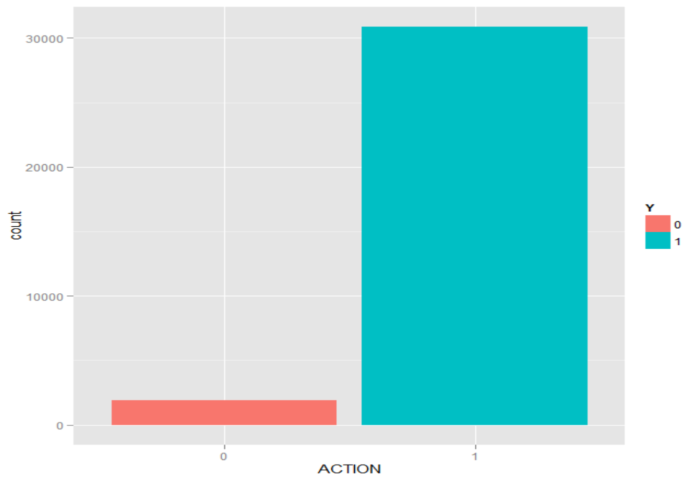
| Variable Name | No of Unique instances | Median |
|---|---|---|
| ACTION | 2 | 30872 |
| RESOURCE | 7518 | 42924 |
| MGR ID | 4243 | 25989 |
| ROLE 1 | 128 | 116953 |
| ROLE 2 | 177 | 118302 |
| ROLE DEPARTMENT | 449 | 118913 |
| ROLE TITLE | 343 | 125916 |
| ROLE FAMILY | 2358 | 170178 |
| ROLE CODE | 343 | 119789 |
| ROLE FAMILY DESC | 67 | 183703 |

All variables has numeric values and assumes certain unique values, mostly consists of IDs.

Shown below is marginal distribution of all variables with respect to target variable ACTION.



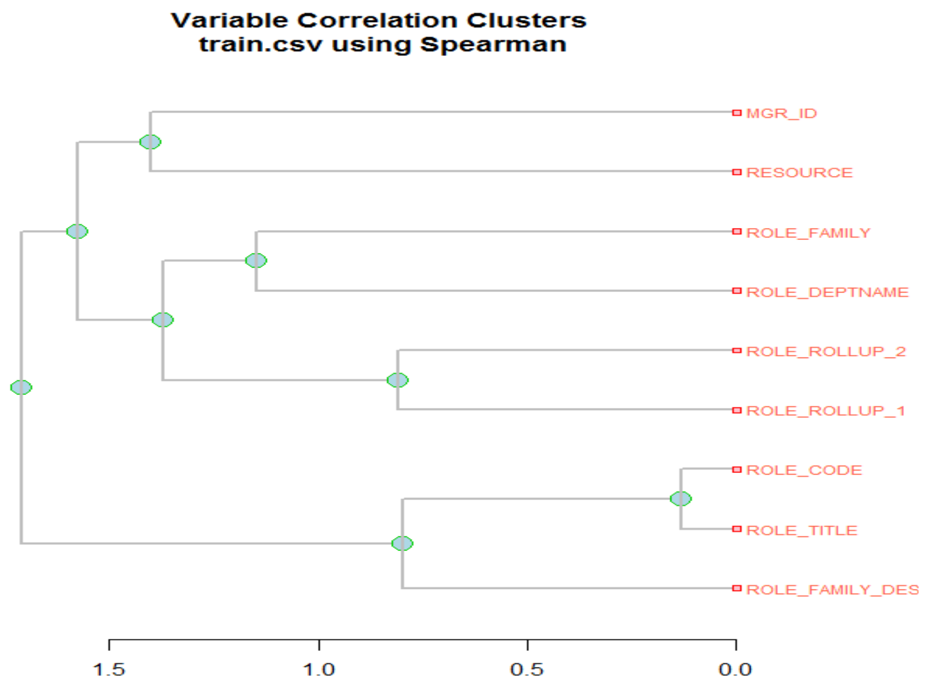
Number of times resource not granted was much higher for ROLL_ROLLUP_1 (between ID 11000 to 12000) compared to all other attributes where number of times resources was granted is much higher.



The number of instances where resource was granted is much higher, this is why marginal distribution shows higher number of granted ACTION per attributes. There are 1897 instances in training set where ACTION equals 0 compared to 30872 instances where ACTION equals 1, ratio is 17:1 which suggests we have class imbalance problem where 0 is minority class and 1 is majority. This will create problems with classification task.

Apart from class imbalance problem there are no apparent problems in data, there are no missing instances in training and testing set and no data inconsistency problems could be detected, this is largely because no information was given by amazon about attributes and valid values of variables, thus checking for inconsistency was left to amazon.

Graph below shows hierarchical relationship between different variables:



As shown above, role code and title are correlated together with role family description. The other pairs which are correlated are role rollup 1& 2, role department and family and manager id and resource.

# Class Imbalance Problem

## Review

In training data set ratio of minority to majority class is 1:17, this makes standard classifier more error prone. This is because lot of classification algorithms have a bias towards classes with greater number of instances, since rules which accurately predicts minority classes are ignores (treating them as noise) because more general rules are preferred.

There are 4 ways to address this problem:

**1) Data Level Approach:** This method is independent of underlying algorithms. General approach is to over sample or under sample minority or majority class to re-balance class distribution. There is standard algorithm called "synthetic minority oversampling technique" (SMOTE). Main idea is to create new minority instances by interpolating several minority class that lie together.

**2) Algorithm Level Approach:** This requires modification of existing classification algorithm by introducing bias towards learning minority class. But this requires data domain knowledge and see why and where algorithm fails.

**3) Cost Sensitive Learning Framework:** This falls between data and algorithm level approaches. It requires both data and algorithm modification i.e. adding cost to each instances and modifying algorithm to accept and minimize the cost of mis-classifying minority class.

**4) Ensemble Methods:** This is classification boosting based approach, where algorithm starts with giving equal weights to all instances and on every new iteration it readjusts the weights of misclassified (or hard to classified) instance and improves on overall accuracy.

Out of these 4 I have tried 1st and 4th approach for now since solution is readily available in R.

## Performance Evaluation in Imbalanced Domains

To evaluate the performance of classifier, I used ROC graphic and AUC score which tends to combine TP, TN, FP and FN rate and produces evaluation criterion. AUC formula is as follows

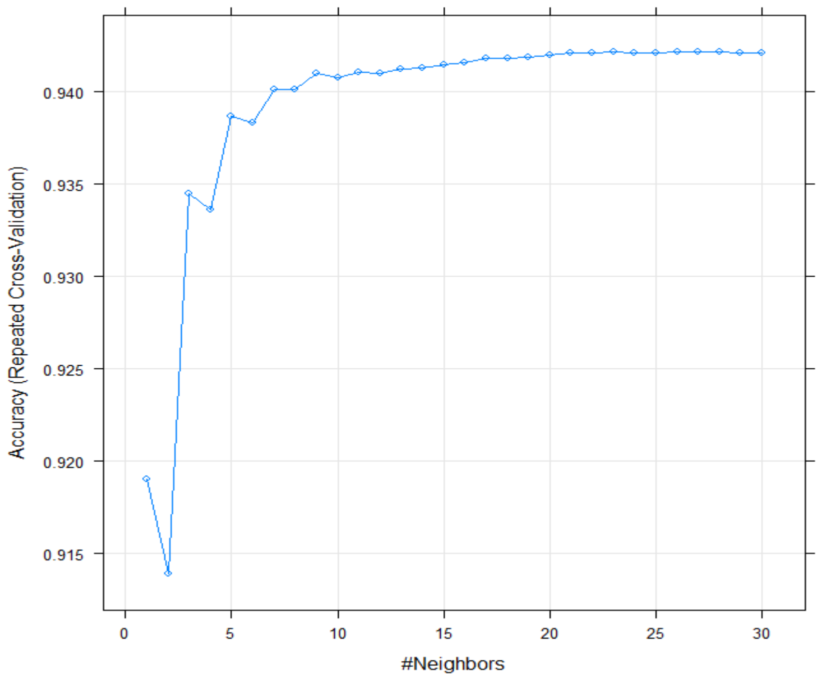
$$\text{AUC} = \frac{1 + TP_{\text{rate}} - FP_{\text{rate}}}{2}.$$

Wherever possible this criteria is used to to select best possible model which maximizes the AUC score.

# Model Analysis

## KNN

First model that I considered was KNN with 10 fold cross validation and 10 repetition on training set. K equals 28 gave best AUC score 0.89 on training set. But on testing set this algorithm performed badly, AUC score was less than 0.71.



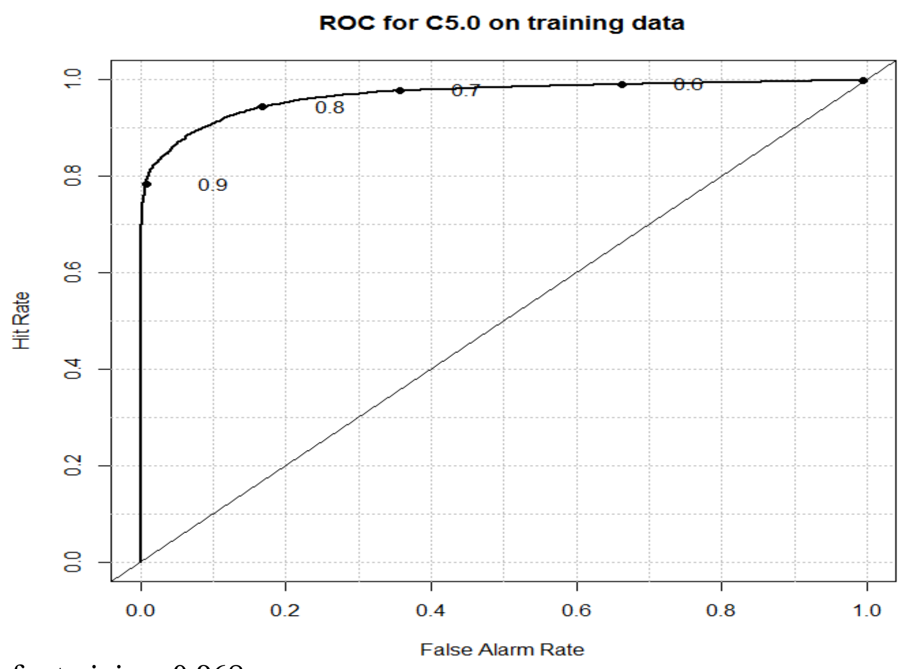
KNN was used without boosting but with ROC evaluation metric to select optimal model from cross validation. Since this was just initial model boosting was not attempted.

# Model Analysis

## C5.0

There are several model of C5.0 was fitted to evaluate the value of "confidence factor" (CF) and number of boosting iterations required to get best possible AUC score.

After fitting several models and using CV, estimated CF value of 0.95 and number of boosting iterations of 100 gave best AUC score with training set.



AUC score for training: 0.968

Confusion Matrix:

| **Actual / Predicted (==>)** | **0** | **1** |
|---|---|---|
| **0** | 683 | 1214 |
| **1** | 33 | 30839 |

Variable Importance Estimated by C5.0:

| **Attribute** | **Overall** |
|---|---|
| MGR_ID | 27.499871 |
| ROLE_DEPTNAME | 18.364285 |
| ROLE_FAMILY_DESC | 15.669132 |
| ROLE_TITLE | 11.566913 |
| ROLE_ROLLUP_2 | 11.546221 |
| ROLE_ROLLUP_1 | 8.349284 |
| ROLE_FAMILY | 7.004294 |
| RESOURCE | 0.000000 |

This was quite strange, The RESOURCE which we trying to predict (whether it'll be granted or not) is least importance.

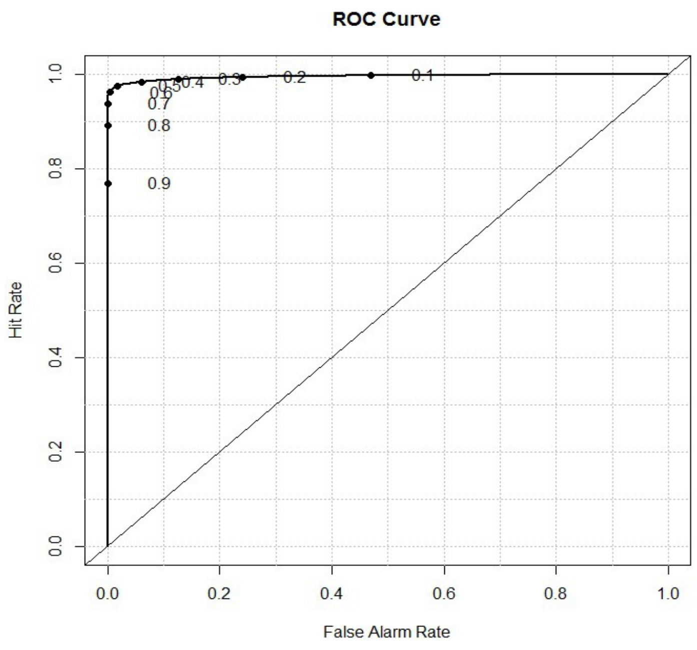
Finally AUC score on testing set was: 0.857

## Random Forest

Random forest model was built using under sampling technique instead of boosting. Several sample size was tried along with other model parameters like number of trees to grow and number of variables randomly sampled at each split.

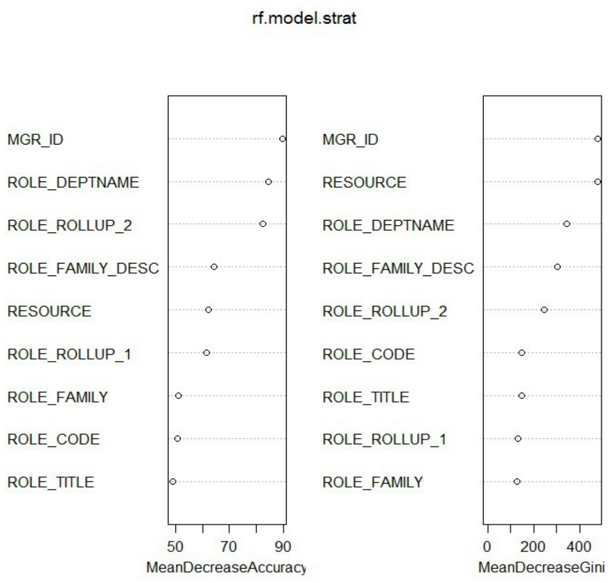
Final model was built using stratified random sampling without replacement based on attribute ACTION and sample size was chosen to be 1700 and 8000 within each strata.

ROC curve:



AUC score on training set was: 0.998

The variable importance computed by random forest:



This is very different than C5.0 but could be because data used on algorithms are quite different.

Confusion Matrix:

| **Actual / Predicted (==>)** | **0** | **1** |
|---|---|---|
| **0** | 739 | 1158 |
| **1** | 372 | 30500 |

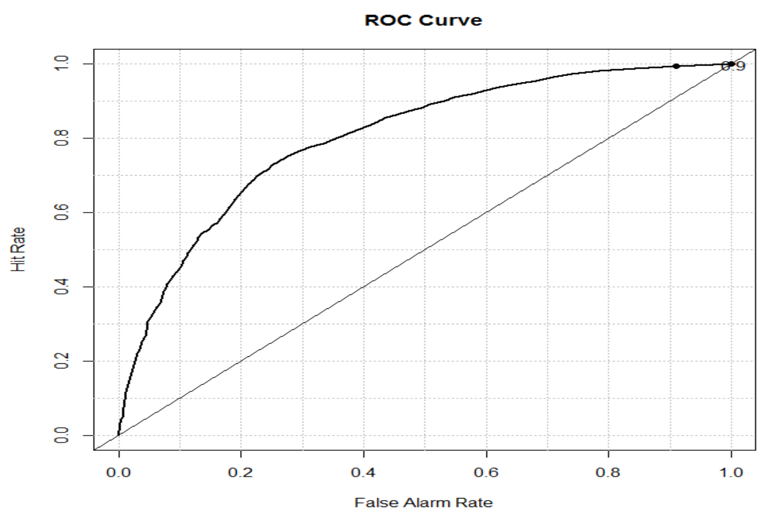
For ACTION 1 overall class error was higher than C5.0 (used with boosting) but still on training set AUC score was better.

The AUC score on test set was: 0.870 which is higher than C5.0

## ADABOOST

ADABOOST works by readjusting the internal weights it gives to each misclassified instance to improve on overall accuracy rate with each iterations. ADABOOST was used with 500 iterations with real and gentle logistic loss functions and AUC score was evaluated.

The AUC score on training was: 0.8



The AUC score on test was 0.67

## SVM and Generalized boosted Logistic regression

Next Tried SVM using default parameters and GBM models which didn't performed as good as C5.0 and Random Forest. For SVM have to estimate best parameters for model fitting rather than using default parameters but GBM even with best parameters model's prediction accuracy was not as good as C5.0 or random forest.

## Summary of Models and its AUC score on testing and training set

| **Models** | **AUC on training** | **AUC on testing** |
|---|---|---|
| KNN | 0.89 | 0.71 |
| C5.0 | 0.968 | 0.857 |
| Random Forest | 0.998 | 0.87 |
| ADABOOST | 0.80 | 0.67 |
| SVM (with defaults) | 0.51 | 0.47 |
| GBM | 0.95 | 0.846 |

## Conclusion

So far random forest with stratified random sampling and C5.0 with boosting gave good performance on test set. SVM performance was worst but maybe after doing parameter tuning it'll give better results. In order to improve on error rate started using SMOTE algorithm for class re-balancing and other technique like clustering and semi-supervised learning.

## References

**GGPLOT2**
http://docs.ggplot2.org/current/geom_bar.html

**Class Imbalanced Problem**
http://sci2s.ugr.es/keel/pdf/algorithm/articulo/2011-IEEE%20TSMC%20partC-%20GalarFdezBarrenecheaBustinceHerrera.pdf

**Kaggle:**
https://www.kaggle.com/c/amazon-employee-access-challenge