STATISTICAL METHODS FOR DS

DS502 Final Project Report

Make Them Stay

Team Member:
Han Yin
Yuchen Shen
Guocheng Yao
Yao-Chun Hsieh
Guohui Huang

Supervisor:
Prof.Randy
Paffenroth

April 19, 2018

Abstract

The purpose of Human resource management is to maximize the productivity of an organization by optimizing the effectiveness of its employees. Thus maintenance of work force is a significant job for HRD. They should make the biggest efforts to keep experienced employees stay.Companies may try implementing reward system or providing long-term professional development for employees.

In this project,we used different statistic methods over R language to efficiently make prediction about the attrition of staffs and retrieve some attributions which may have significant influence on deciding the consistency of employment.

Keywords: Human Resource, Attrition, Prediction, Statistics, R

1 Overview

In this project, we implemented four prevalent statistic methods: Logistic Regression, SVM, Decision Tree and Artificial neural networks (ANNs) using R to predict the attrition of employees and gained some factors who contribute significantly to the reassignment.

2 Background Material

2.1 Data Set We Use

Our dataset comes from Kaggle community, calls "IBM HR Analytics Employee Attrition and Performance". Details of data sets show as the following:

- The original data file consists of 1470 rows and 35 columns (26 numeric and 9 categorical).
- It is a fictional data set created by IBM data scientists.

Description about specific attribute and preprocessing(We turned the categorical values into numeric for future analyzing.):

Age: Age of employees, integer;
Attrition: "Yes"=1 or "No"=0;

BusinessTravel:"Non-Travel"=0, "Travel-Rarely"=1, "Travel-Frequently"=2;

DailyRate: The amount of money paid per day, integer;

Department: "Human Resources"=0,"Research & Development"=1,"Sales"=2;

DistanceFromHome: Miles, integer;

Education:"Below College"=1, "College"=2, "Bachelor"=3, "Master"=4, "Doctor"=5;

EducationField: "Human Resources"=0,"Life Sciences"=1,"Marketing"=2, "Medical"=3,"Technical Degree"=4,"Other"=5;

EmployeeCount: 1;

EmployeeNumber: Number of employees, integer;

EnvironmentSatisfaction:"Low"=1, "Medium"=2, "High"=3, "Very High"=4;

Gender: "Female"=0,"Male"=1;

HourlyRate: the amount of money paid per hour, integer;

JobInvolvement: "Low"=1, "Medium"=2, "High"=3, "Very High"=4;

JobLevel: "Low"=1, "Medium"=2, "High"=3, "Very High"=4;

JobRole: "Sales Executive"=0,"Research Scientist"=1,"Laboratory Technician"=2,"Manufacturing Director"=3,"Healthcare Representative"=4,"Manager"=5, "Sales Representative"=6,"Research Director"=7, "Human Resources"=8;

JobSatisfaction: "Low"=1, "Medium"=2, "High"=3, "Very High"=4;

MaritalStatus: "Divorced"=0, "Married"=1, "Single"=2;

MonthlyIncome: the amount of money paid per month, integer;

MonthlyRate: the amount of money paid per month, integer;

NumCompaniesWorked: numbers of companies employees had worked, integer;

Over18: "No"=0,"Yes"=1;

OverTime: "No"=0,"Yes"=1;

PercentSalaryHike: The amount of incrementation of salary by percentage, integer;

PerformanceRating: "Low"=1, "Good"=2, "Excellent"=3, "Outstanding"=4;

RelationshipSatisfaction: "Low"=1, "Medium"=2, "High"=3, "Very High"=4;

StandardHours: 80hr, integer;

StockOptionLevel: Number of company's stock owns, integer;

TotalWorkingYears: Total work year of employees, integer;

WorkLifeBalance: "Bad"=1, "Good"=2, "Better"=3, "Best"=4;

YearsAtCompany: Years at company, integer;

YearsInCurrentRole: Years in current role, integer;
YearsSinceLastPromotion: Years since last promotion, integer;
YearsWithCurrManager: Years with current manager, integer;
TrainingTimesLastYear: The amount time of training last year, integer;

2.2 Data Preprocessing and Splitting

There are 1470 records in our data set, we split them into training set with 1200 samples and final testing set with 270 samples. Since we are trying to use different methods and parameters, we split our training data into validation set with 200 samples and 1000 final training set. In the validation set, we can use cross validation to find the best parameter for best model. Considering the training set, understanding the distribution of our data is important. In our target attribute 'Attrition' only 159 samples have category '1' which means the employee will leave the company. Obviously, the training data is unbalanced. Without further processing, we apply our model to the data set and none of them can recognize category '1' well. Even in ANN model, all the data is classified into category '0', since the accuracy will be high up to 84.1% when classified into only one category.

In order to get a better result, we use further data splitting method to get several balanced training set. In our 1000 samples training set, there are 159 samples with category '1' and 841 samples with category '0', we split those 841 samples into 5 set which have 168, 168, 168, 168 and 169 samples. Then, we combine 159 samples in category '1' with each set in category '0'. Finally, we now have 5 balanced training set with 327, 327, 327, 327 and 328 samples. We train the model in each of these five balanced training sets, and use the voting result as our final classification result.

2.3 Tools Referred To

R, RStudio, Excel

3 Our Statistic Approaches

3.1 Logistic Regression

Since our target "Attrition" categorical attribute, we try to apply logistic regression first.

$$\ln\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x$$

Figure 1: LR function

In linear logistics regression method, we need to do data cleaning before apply the model to our data. We removed 3 attributes that has only one value which means these attributes will not influence our target. We do the linear regression using three different methods. First, with all of our remaining attributes; Second, apply PCA and Lasso to our original. PCA is a dimension reduction method using orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. It will generate new variables which are the combination of the old ones. On the other hand, Lasso is a feature selection method which select important variables from old ones. After reducing the number of features, since our problem is a classification problem, we will apply logistic regression method to our new data and get the possibility of the binary class. Since our data is unbalanced, in our test set, "Attrition" equals to 1 is minority which means the employee will leave the company, we are taking more attention to these results. Confusion matrix can give us a clear vision of our classification.

For normal linear logistic regression:

Pred\True	0	1
0	164	10
1	56	40

Figure 2: LR Result

For PCA + linear logistic regression:

Pred\True	0	1
0	149	10
1	71	40

Figure 3: LR Result

For Lasso + linear logistic regression:

Pred\True	0	1
0	171	8
1	49	42

Figure 4: LR Result

Based on these, we generate the result of all three methods: We can find

Method\Evaluation	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.7555556	0.4166667	0.8	0.5479452
PCA + Logistic Regression	0.7	0.3603604	0.8	0.4968944
Lasso + Logistic Regression	0.7814815	0.4505495	0.82	0.5815603

Figure 5: LR Result

Logistic regression with Lasso gives the best result in both accuracy and f1-score.

3.2 Tree Models

Tree models are often used for classification. In this section, we train decision tree model and random forest for further prediction. The training dataset has been separated into 5 subset, aiming to balance the number of data for each class. We totally train 5 models with these subsets, and apply voting mechanism to decide whether a test data should be categorized as [Attrition: Yes] or [Attrition: No]. The detail of our training process for each model is introduced as follows:

Decision Tree:

For each subset, we apply cross-validation to choose the best tree node number for building the tree model. Also we employ pruning method to reduce the miss classification on training set. The five model are as follows:

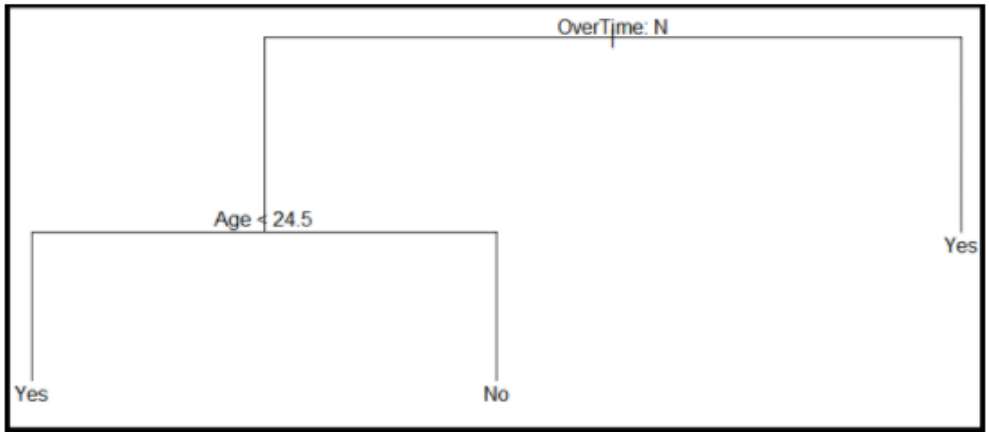

Figure 6: MODEL 1

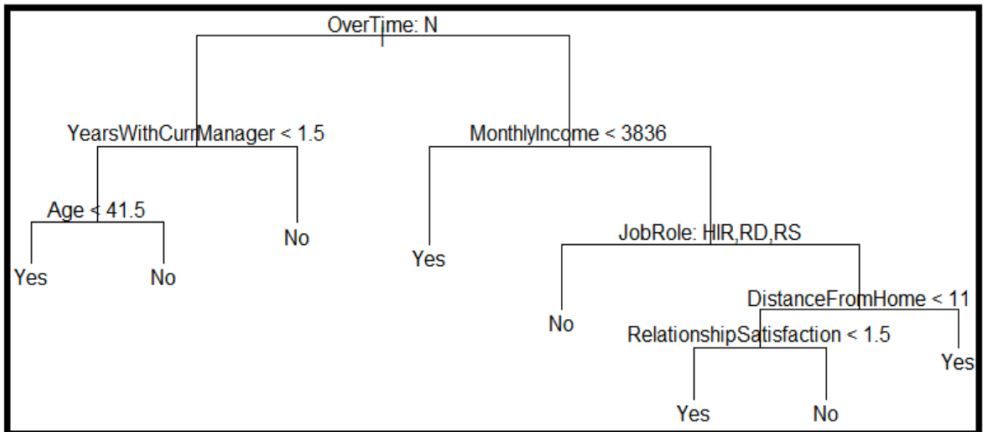

Figure 7: MODEL 2

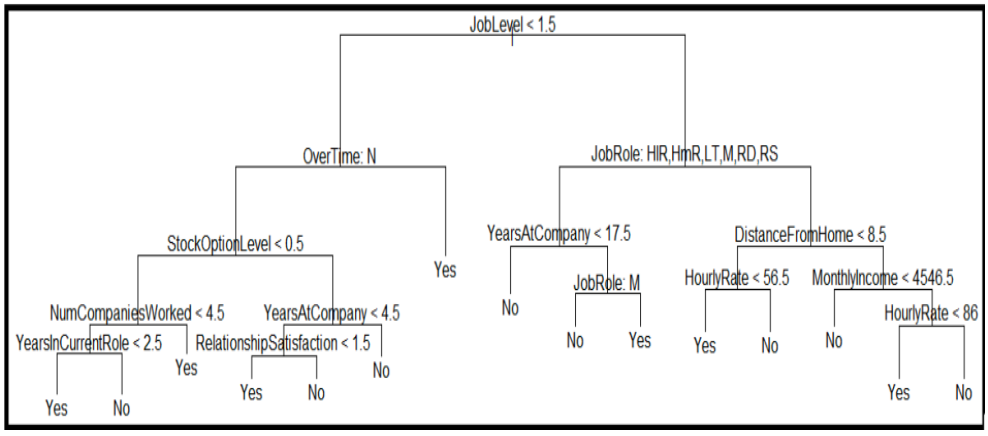

Figure 8: MODEL 3

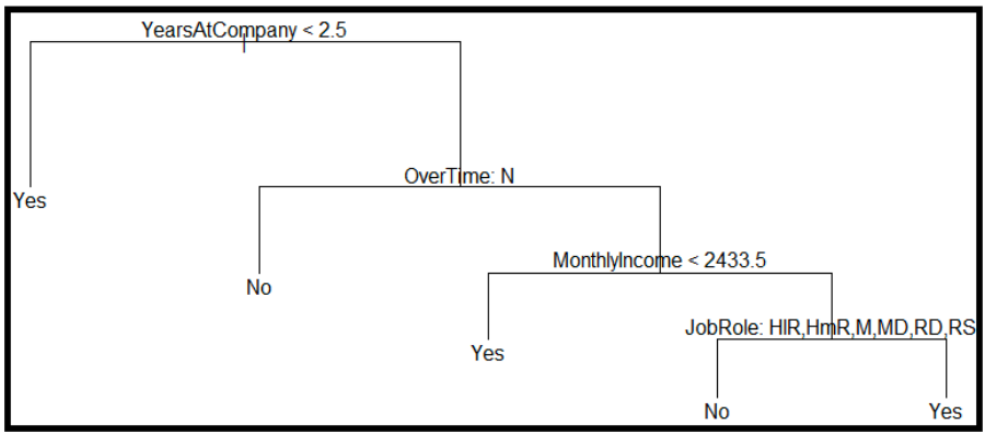

Figure 9: MODEL 4

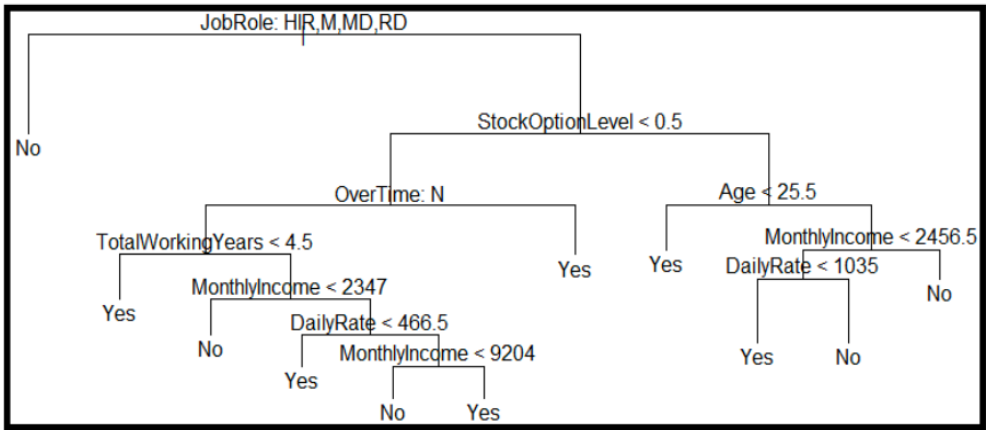

Figure 10: MODEL 5

We obtain the testing accuracy and related evaluations with these 5 models. For each test data, if the result from 3 of these model is [Attrition: No], then the data is assigned to [Attrition: No] class, vise versa. The final result shows in figure 11:

pred\True	0	1
0	179	15
1	41	35

Accuracy	0.7925
Precision	0.4605
Recall	0.7
f-Measure	0.5556

Figure 11: Tree Model Result

Random Forest
We build a random forest for each training subset using the same procedure as building a single decision tree. Also, voting mechanism is applied among these forests model. The test result is as follows:

pred\True	0	1
0	180	13
1	40	37

Accuracy	0.8037
Precision	0.4805
Recall	0.7400
f-Measure	0.5827

Figure 12: Random Forest Result

These models do not perform really well, however, we can still get information from these model. Attribute [OverTime] seems to be a significant respectively, since it appears in all decision trees, and all in early separation stage.

3.3 Support Vector Machine

In addition to the result we get from Logistic regression, Decision tree and Random forest, we now try another method called Support Vector Machine(SVM). SVM is a supervised learning model with associated learning algorithms that analyze data used for classification. It separates the classes with a decision surface that maximizes the margin between the classes. The surface is often called the optimal hyperplane and the data points closest to the hyperplane are called support vectors.

Follow the same training dataset setting above, the training dataset has been separated into 5 subsets, aiming to balance the number of data for each class. We totally train 5 models with these subsets, and apply voting mechanism to decide whether a test data should be categorized as [Attrition: Yes] or [Attrition: No].

For this classification problem, we do Support Vector Machine using three different Kernel functions: linear kernel, polynomial kernel and radial kernel. Then we start to train our three different SVM model by the training dataset. And use tuning dataset to do cross validation to find the optimal accuracy of the models.

For SVM with linear kernel, we get the optimal accuracy of 0.875 when the cost is 0.001.

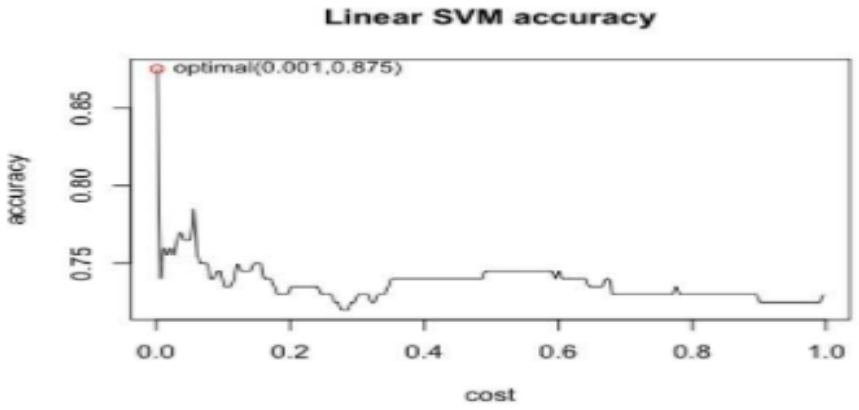

Figure 13: Linear SVM Accuracy

And the optimal confusion matrix of linear SVM is:

Pred\True	0	1
0	167	20
1	5	8

Figure 14: Confusion Matrix of Linear SVM

For SVM with polynomial kernel, we get the optimal accuracy of 0.875 anywhere:

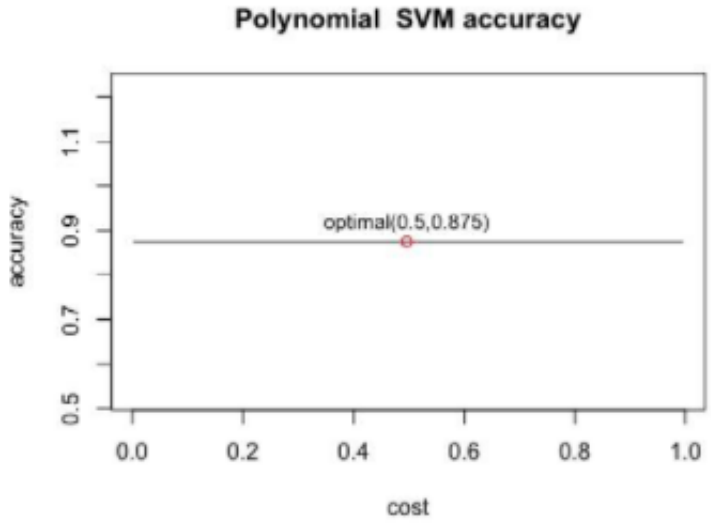

Figure 15: Polynomial SVM Accuracy

And the optimal confusion matrix of Polynomial SVM is:

Pred\True	0	1
0	137	10
1	35	18

Figure 16: Confusion Matrix of Polynomial SVM

For SVM with radial kernel, we get the optimal accuracy of 0.87 at cost of 0.09.

And the optimal confusion matrix of Radial SVM is:

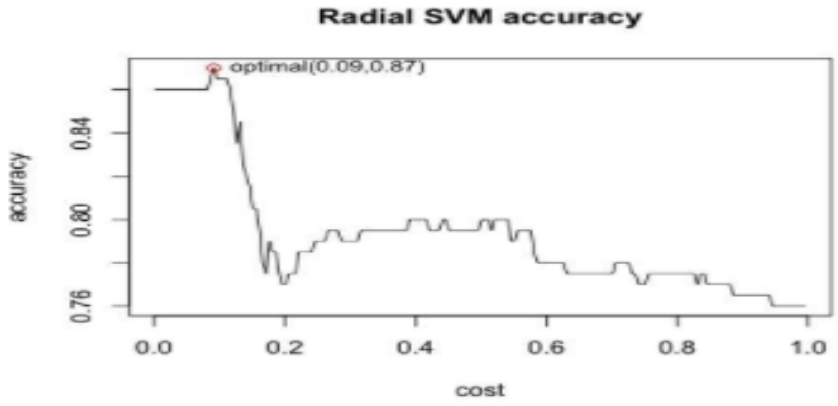

Figure 17: Radial SVM Accuracy

Pred\True	0	1
0	171	25
1	1	3

Figure 18: Confusion Matrix of Radial SVM

Finally, we use testing dataset to compare the performance of these SVM functions with different kernels. And we get the result as follow:

Method (SVM)	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Linear	0.8592593	0.9285714	0.26	0.40625
Polynomial	0.7925926	0.4594595	0.68	0.5483871
Radial	0.8296296	0.8333333	0.1	0.1785714

Figure 19: Testing Linear SVM Accuracy

Testing confusion matrix of linear SVM:

Pred\True	0	1
0	219	37
1	1	13

Figure 20: Testing Confusion Matrix of Linear SVM

Testing confusion matrix of Polynomial SVM:

Pred\True	0	1
0	180	16
1	40	34

Figure 21: Testing Radial SVM Accuracy

Testing confusion matrix of Radial SVM:

Pred\True	0	1
0	219	45
1	1	5

Figure 22: Testing Confusion Matrix of Radial SVM

Above all, we find that Support Vector Machine with linear kernel get a better prediction of the classification of 'Attrition'.

3.4 Artificial Neural Network

ANN is based on a collection of connected units or nodes called artificial neurons. Each connection between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it. ANN can solve some really interesting problems once they are trained. They are very good at pattern recognition problems and with enough neurons(elements) can classify any data with arbitrary accuracy. They are particularly well suited for complex decision boundary problems over many variables.

ANNS are applied in many situations. In this project,we use neuralnet function in R to train neutral network. It is built to train multi-layer perceptrons in the context of regression analyses, i.e. to approximate functional relationships between covariates and response variables.

Here we used the sum of the cross-entropy as error function E to measure the difference between predicted and observed output:

$$E = -\sum_{l=1}^{L}\sum_{h=1}^{H}(y_{lh}\log(o_{lh}) + (1-y_{lh})\log(1-o_{lh}))$$

Figure 23: Error Function:Sum of the Cross-Entropy

where l = 1,. . ., L indexes the observations, i.e. given input-output pairs, and h = 1,. . ., H the output nodes.

Then confusion matrix of ANN is:

Pred\True	0	1
0	214	6
1	43	7

Figure 24: Confusion Matrix of ANN

Based on confusion matrix, we generate the result of all other measurements.

Method\Evaluation	Accuracy	Recall	Precision	F1-Score
ANN(neuralnet)	0.8185	0.5385	0.14	0.22

Figure 25: ANN Result

The error is 0.18 and F1 score is 0.22.Compared with other method we have implemented,this result is shockingly poor because of the limitation of dataset,which contains only 1470 samples.

3.5 Additional Method:XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

Then confusion matrix of XGBoost is:

Pred\True	0	1
0	200	20
1	19	31

Figure 26: Confusion Matrix of XGBoost

The result of all other measurements:

Method\Evaluation	Accuracy	Precision	Recall	F1-Score
XGBoost	0.8556	0.6078	0.62	0.6139

Figure 27: XGBoost Result

We get a much better result than ANN.

4 Conclusion

In this case, we want to pay more attention on the group of leaving staffs. Meanwhile, in the consideration of unbalance problem, we choose F1 score rather than accuracy here. We implemented 5 different models above and we get best result over XGBoost(F1 score is 0.6139).

Also, according to results observed from decision trees, attribute: [Overtime] plays an important role during prediction. Therefore, if companies want to reduce the attrition rate, paying more attention to those who work overtime might be a right direction.

The performance is still a little bit low than what we expected. We think it makes sense in this case, for two reason: The first one is sample size. Our training data set is almost one thousand, and due to unbalance problem, we need to divide it into several smaller subsets. These subsets are not enough for us to train a good model. The second reason is the choose of threshold. In logistic and XGboost, the response is a probability. We need to find a threshold, combine the response with this threshold, and then decide which

class the observation should be assigned. Different thresholds will lead to a big difference on F1 score, however, because of data snooping, we can't try different thresholds on test set.

5 Reference

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: With applications in R. New York: Springer.