ISyE 6416 – Basic Statistical Methods - Spring 2016
Bonus Project: "Big" Data Analytics
Final Report

Team Member Names: Xi Yang, Yi Wen, Xue Zhang

Project Title: Improve Room Utilization

Introduction

Problem Statement

As the development of information technology, the operation and management efficiency of room reservation has been improved significantly. People can reserve a certain room for a specific time slot in advance which reduces scheduling conflicts and waiting time. In such way, the room utilization seems to be improved, however, it also comes with couple of big drawbacks for the operation. For instance, some room reservation was made in advance, and the length of reservation is also determined at that time, which is not flexible for any changes. Some reservation can be made instantly as for needed if there are available rooms. Most of the room reservation systems was designed into time slot with certain length (e.g 1hr), and the room can only be reserved based on number of time slots. However, the length of a meeting is hardly fitted in to certain amount of time, and has a big variation and effected by many factors, for example, number of participants, contents and progress of the meeting. In order to avoid lack of time, people may easily reserve a time slot that is much longer than they actually need. (e.g The meeting only takes 1hr and 10min to finish, however, for room reservation people may need to reserve the room for 2hrs, and the in the rest 50mins, the room will not be used, and either available for reservation). This causes a lower room utilization and operation efficiency. In order solve this problem, we would like to collect real time information of room occupancy. With this data, the information of room occupancy can be provided in real time, and rooms that are not in used can be available for reservation immediately even though it is reserved under previous reservation.

In this project, we used the data that can be a factor to determine room occupancy, and using several classification methods to develop a effective model for room occupancy determination.

Data Source and pre-processing

- **Data Source**
  UCI Machine Learning Repository: Occupancy Detection Data set
  http://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+
  Data attribute:
  - date time year-month-day hour:minute:second
  - Temperature, in Celsius
  - Relative Humidity, %
  - Light, in Lux
  - CO2, in ppm
  - Humidity Ratio, Derived quantity from temperature and relative humidity, in kgwater-vapor/kg-air
  - Occupancy, 0 or 1, 0 for not occupied, 1 for occupied status

- **Data Pre-processing**

  - There are total 6 variables that may be an important factor to determine room occupancy, and binary response variable the indicates weather the room is occupied or not. ("0" for not being occupied, and "1" for being occupied)
  - We assume the variable date time year-month-day hour:minute:second is only a index of the data, and does not have any dependent relationship with the final result. We did not select it as a factor to build the model
  - After study the data, we found out that the variable Relative Humidity and Humidity Ratio are fully linear dependent, so we only select Relative Humidity as a factor to build the model

Strategy & Methodology

- **Strategy**

  We used the data to build different models to determine the room occupancy, and used following procedure to evaluate the model

  AUC – An Indicator of Model Performance

  In statistics, a receiver operating characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, or recall in machine learning. The false-positive rate is also known as the fall-out and can be calculated as (1 - specificity).

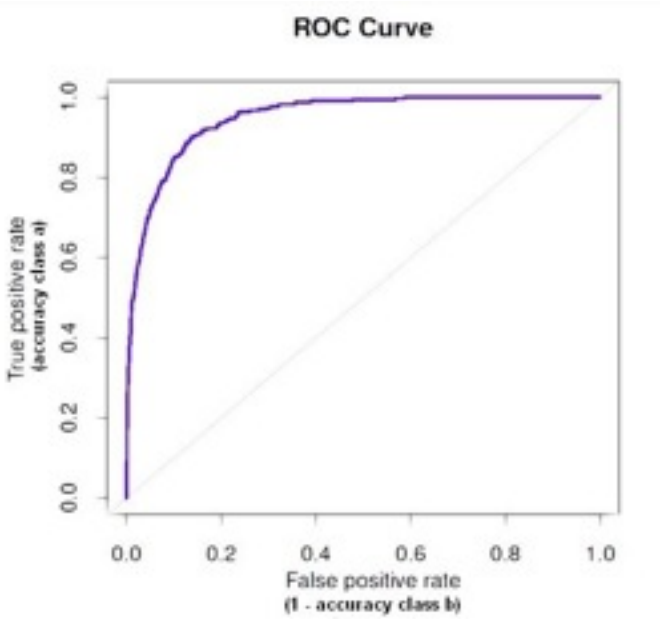

Figure. 1 – A Typical ROC Curve

The area under the curve (often referred to as simply the AUC, or AUROC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming "positive" ranks higher than "negative"). The implicit goal of AUC is to deal with situations where you have a skewed sample distribution, and don't want to over-fit to a single class.

In this project, AUC is used as the indicator of model performance for model comparison.

Model Comparison Mechanism

We design the following process for model comparison. The process is a combination of Monte Carlo cross-validation and t test comparison

Monte Carlo cross-validation can also be called as repeated random sub-sampling validation. This method randomly splits the dataset into training and validation data. For each such split, the model is fit to the training data, and predictive accuracy is assessed using the validation data. The results are then averaged over the splits. The advantage of this method (over k-fold cross validation) is that the proportion of the training/validation split is not dependent on the number of iterations (folds). The disadvantage of this method is that some observations may never be selected in the validation subsample, whereas others may be selected more than once. In other words, validation subsets may overlap. This method also exhibits Monte Carlo

variation, meaning that the results will vary if the analysis is repeated with different random splits.

Specifically, in our project, for each given model,

a) Perform 100 iterations of Monte Carlo cross-validation. In each iteration, randomly select 1/10 data as the testing data, the left as the training data.

b) Fit the training data to a specific model, then make prediction on the testing data. (Note that the predictions are in probability form)

c) Use the predictions and the true responses (occupancy status) to calculate the AUCs (note that we have 100 AUC numbers from the 100 iteration for each model).

d) At last we conduct pair-wise T-tests to compare the AUCs obtained from different models.

- **Methodology**

In this project, we have conducted 4 models using the following methods: Logistic Regression, LDA/QDA Discriminant Analysis, and Random Forest

1. Logistic Regression

Logistic regression is a flexible method for situations that variables are not in normal distribution. In this case, we have a binary outcome Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

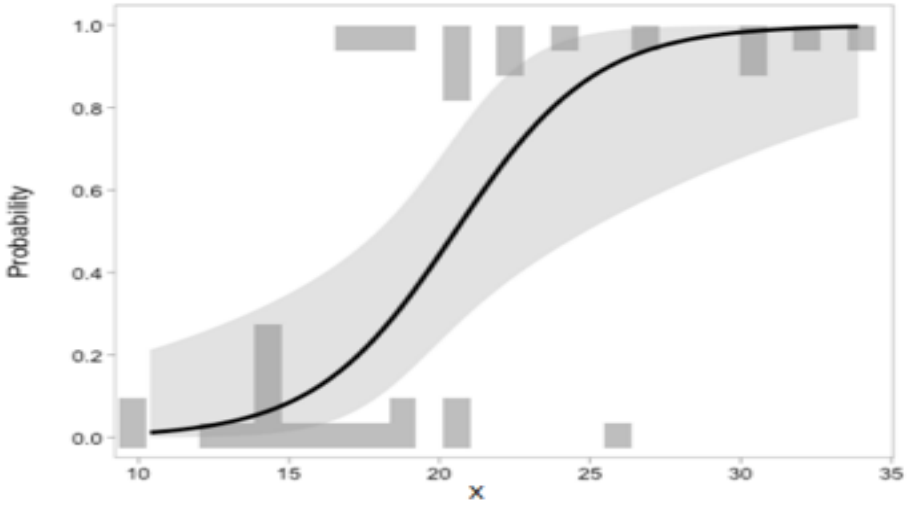

Figure 2: Logistic Regression Example

Our data set has 4 variables so variable selection and shrinkage methods are not necessary needed. At first we perform 100 iterations Monte Carlo cross-validation and make prediction. Then we compare the prediction of testing data with true response to calculate average AUC. The result shows that Logistic regression has AUC larger than 99%.

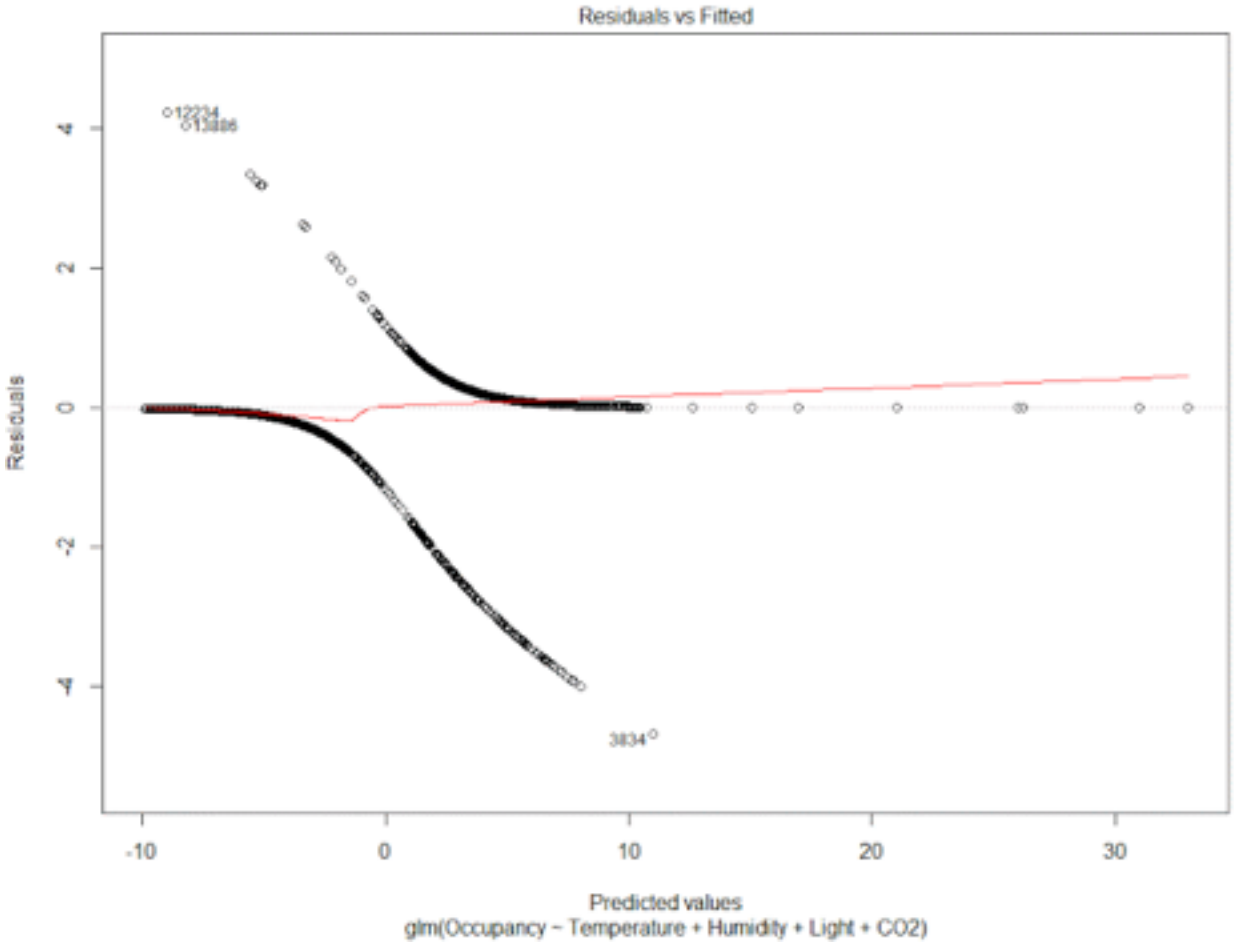

Figure 3: Residual Plots of Logistic Regression

Call: glm(formula = Occupancy ~ Temperature + Humidity + Light + CO2, family = binomial(link = "logit"), data = train)

Coefficients:

(Intercept)	Temperature	Humidity	Light	CO2
6.643337	-0.933915	0.087574	0.024596	0.003726

$$P(Y_i = 1) = \pi_i, \quad P(Y_i = 0) = 1 - \pi_i$$

$$\pi_i = \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_{p-1} x_{i,p-1}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_{p-1} x_{i,p-1}}}$$

(Source: ISyE 2016 spring 7406 lecture notes by Dr. Mei, Yajun)

In our case, temperature is most important factor which is negatively related to occupacy. According to equation above, with increasing temperature, the probability of a room being occupied decreased. In the opposite, humidity, light and CO_2 have positive corelation with occupacy.

2. LDA/QDA Discriminant Analysis

The other method we used is LDA (Linear Discriminant Analysis) and QDA(Quadratic Discriminant Analysis). This is a very common used and powerful method to classify or characterize two or more class or events. As our problem is basically a classification problem("0" for the room not being occupied, and "1" for being occupied), we think this should be an appropriate method to be used. LDA/ QDA assumes that all independent variables are normally distributed which we assume is true in our case.

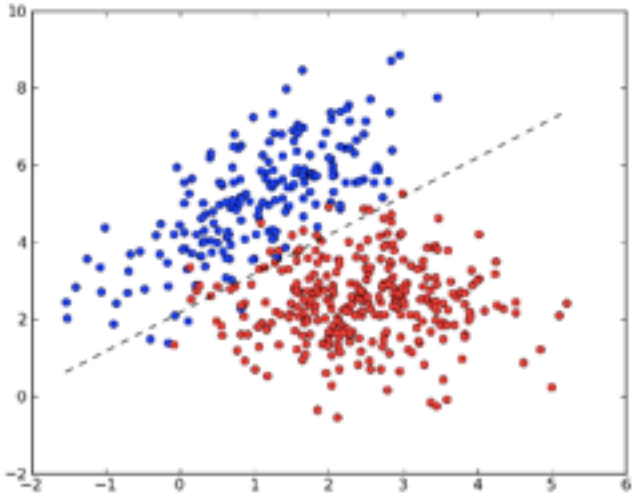

Figure 4: LDA classification example

We applied Monte Carlo Cross Validation to the model in order to evaluate its performance. The result of both LDA model and QDA model is desirable, as the testing error is under 2%, and the AUC value is above 99%

3. Random Forest

1) Method introduction

Random forest is based on tree based method. The tree based method will start with the entire space and recursively divide it into smaller regions based on the value of input variables (as illustrated in Figure 1). At the end, it gives out a bunch of small regions with assigned class label, here in our case, the lable is either 0, which mean the room is not occupied or the label is 1, which means the room is occupied.

Random forest will construct a multitude of this kind of decision trees and aggregating the result from all the trees and output the final classification information. By considering the result from different trees, random forest method can correct the decision trees' habit of overfitting to the training data.

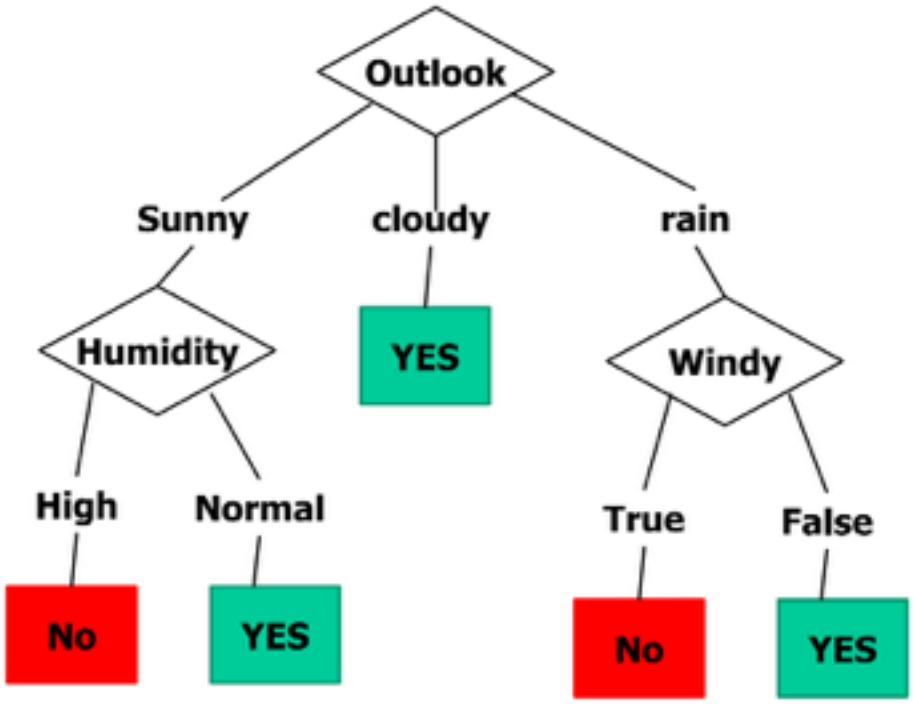

Figure 5 (Source: ISyE 2016 spring 7406 lecture notes by Dr. Mei, Yajun)

2) Parameter tuning

An important parameter for random forest is the number of trees to grow for the model. If we grow too less number of tress, the result is of high randomness, which does not represent the model's true predictability very well. If we grow to many tress, it will cost a lot of time and computational resource. So we repeat the application with different number of trees, starting from 100 to 1500 to find out the appropriate number of trees. As show in below plot, the average AUC converged to 0.995, we believe this reflect this model's true predictability, therefore 1500 is the desired level of tree numbers to grow.

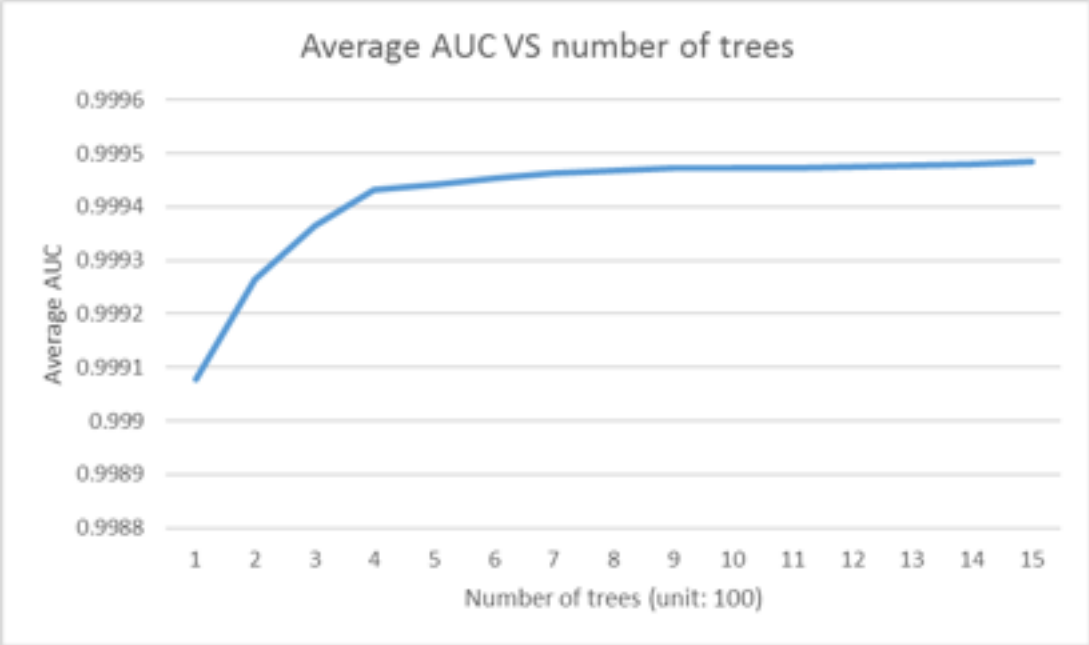

Figure 6: Average AUC VS number of trees

3) Result and summary

The final average AUC for this method is 99.95% with 1500 trees. We believe this is a pretty good model to do the prediction.

Result

Method	*Logistic*	*LDA*	*QDA*	*Random Forest*
Average AUC	***99.450%***	***99.360%***	***99.355%***	***99.95%***

T-test	log vs LDA	log vs QDA	log vs Random Forest	LDA vs QDA	LDA vs Random Forest	QDA vs Random Forest
P-value	8.976E-05	3.157E-05	2.2E-16	0.8193	2.2E-16	2.2E-16

As the result shown above, all four model returned a desired result. The average AUC are all above 99%. Random Forest model got the best result which the average AUC is almost 100%. As the p-value obtained from the t-test indicates, Random Forest is the best model to be chosen.

Conclusion and Future Development

In this project we conducted LDA, Logistic Regression and Random Forrest methods. Each of them has a good performance with AUC larger than 99% which means the 4 variables can explain most of the incidence in our data set.

The Random Forest method has the best performance. Although it is time consuming in tree selection step, after our model has built, the prediction for 2065 data points only takes less than 1 second. Using our model we can update the status of 2065 rooms within one second.

We evaluated our project, and concluded that this solution can be easily applied to current room reservation system as the data can be collected by existing equipment (temperature control system, AC…)

We also noticed that the time horizon of the data collected is only one week. The seasonality might also be a factor that affects the results. For future development, we also need to take this effect into consideration, and collect data from a longer horizon.

Appendix:

Logistic Regression

```
# ISyE 6416 project
# Logistic regression
library(nnet)
library(pROC)
setwd("C:/Users/ywen49/Documents/R")
data = read.csv('occupancy_data.csv')
data = data[, -c(1, 2)]
n = dim(data)[1]
n1 = round(n/10)
set.seed(20160424)

B = 100;
AUClog = NULL;
for (b in 1:B){
  set.seed(20160415+b)
  flag = sort(sample(1:n, n1))
  train = data[-flag,]
  test = data[flag,]
  fitlog <- glm(Occupancy ~ Temperature+Humidity+Light+CO2, family =
 binomial(link="logit"), data=train)
  predlog <- predict(fitlog, test[,1:4])
  AUC = auc(test[,6], predlog)
  AUClog = rbind( AUClog, AUC)
  }
AUCmean = mean(AUClog)
```

LDA/QDA

```
n = 20560
n1 = round(n/10)
predlda1 = matrix(nrow = 2056,ncol = 100)
predlda2 = matrix(nrow = 2056,ncol = 100)
predqda1 = matrix(nrow = 2056,ncol = 100)
predqda2 = matrix(nrow = 2056,ncol = 100)
auclda = NULL
aucqda = NULL
telda = NULL
teqda = NULL
```

```
for (b in 1:100){
  set.seed(20160415+b)
  flag <- sort(sample(1:n,n1))
  datatrain = occupancy_data[-flag,];
  datatest = occupancy_data[flag,];
  ##LDA
  fit1 <- lda(Occupancy ~ Temperature+Humidity+Light+CO2, data = datatrain)
  predlda1[,b] <- predict(fit1,datatest[,3:6])$class
  predlda1[,b] <- predlda1[,b] -1
  predlda <- predict(fit1,datatest[,3:6])$posterior
  predlda2[,b] <- predlda[,2]
  telda[b] <- mean(predlda1[,b] != datatest[,8])
  auclda[b] <- auc(datatest[,8],predlda2[,b])
  ##QDA
  fit2 <- qda(Occupancy ~ Temperature+Humidity+Light+CO2, data = datatrain)
  predqda1[,b] <- predict(fit2,datatest[,3:6])$class
  predqda1[,b] <- predqda1[,b] -1
  predqda <- predict(fit2,datatest[,3:6])$posterior
  predqda2[,b] <- predqda[,2]
  teqda[b] <- mean(predqda1[,b] != datatest[,8])
  aucqda[b] <- auc(datatest[,8],predqda2[,b]);}
```

Random Forest

```
library(randomForest)
library(pROC)
attach(occupancy_data)
occupancy_data$Occupancy=as.factor(occupancy_data$Occupancy)
AUC<-matrix(nrow=15,ncol=100)
for (k in 1:15)
 {
       for (i in 1:100)
       {
         set.seed(20160415+i)
         flag<-sort(sample(1:(length(Occupancy)),0.1*length(Occupancy)))
         train1<- occupancy_data[-flag,]
        test1<- occupancy_data[flag,]
         rf1 <- randomForest(Occupancy ~Temperature+Humidity+Light
              +CO2+HumidityRatio,
        data=train1, ntree=100*k)
        rf1.pred <- predict(rf1, test1[,3:7], "prob")
       AUC[k,i]<- auc(test1[,8],rf1.pred[,2])
```

```
print(k)
}}
write.csv(AUC,file="C:/Stef/GT/ISyE_6416_Computational Statistics/
Porject/AUC.csv")
```