BIG DATA ANALYTICS

FINAL PROJECT (UCI-INCOME)

Saksham Dixit (sxd165530)
Shivam Tiwari (sxt167530)
Gaurav Khatavkar (gvk150030)
Wanjou Liu(wxl161430)

05/03/2017

INTRODUCTION

We're living in the era of big data. The rapid advancement in technology mostly fueled the organizations across the globe to produce data that kept accumulating at a large scale. While maintaining the data is a challenging task for the enterprises, there arises the need to explore possibilities to use historical data intelligence.

Data in general is structured data and unstructured data. The understanding of the nature of these data types is crucial to understand the world of Big Data. Thus, the process of big data analytics includes exploring the data, cleaning the data, splitting the data, training the data, and testing the data. By those steps, we find out which model fit the best.

R has become the main stream with statisticians and data miners who use it to develop statistical software, and is widely used for advanced data analysis. It provides a wide variety of statistical and graphical techniques such as linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, etc. Hence, we choose R on the big data analytics project. In addition, we use Pig Latin on data splitting step.

INDEX

DATA SPLITTING

After extracting the data from the compressed file, the complete training data was moved to HDFS, using Pig the training data was loaded, with each column being read as chararray. It was found that the data had few NULL values represented as question mark ('?'). To remove the NULLs, we first replaced question marks with NULLs and then removed those NULL values.

```
train = LOAD '/user/root/project/train_data.data' using PigStorage(',') AS
        (age:chararray,
         workclass:chararray,
         fnlwgt:chararray,
         education:chararray,
         education_num:chararray,
         marital_status:chararray,
         occupation:chararray,
         relationship:chararray,
         race:chararray,
         sex:chararray,
         capital_gain:chararray,
         capital_loss:chararray,
         hpw:chararray,
         native_cont:chararray,
         salary:chararray);

 -- Converting '?' to NULL

train1 = FOREACH train GENERATE
        (int)REPLACE(age,'\\u003F',null) as age,
         REPLACE(workclass,'\\u003F',null) as workclass,
         REPLACE(fnlwgt,'\\u003F',null) as fnlwgt,
         REPLACE(education,'\\u003F',null) as education,
         REPLACE(education_num,'\\u003F',null) as education_num,
         REPLACE(marital_status,'\\u003F',null) as marital_status,
         REPLACE(occupation,'\\u003F',null) as occupation,
         REPLACE(relationship,'\\u003F',null) as relationship,
         REPLACE(race,'\\u003F',null) as race,
         REPLACE(sex,'\\u003F',null) as sex,
         (int)REPLACE(capital_gain,'\\u003F',null) as capital_gain,
         (int)REPLACE(capital_loss,'\\u003F',null) as capital_loss,
         (int)REPLACE(hpw,'\\u003F',null) as hpw,
         REPLACE(native_cont,'\\u003F',null) as native_cont,
```

```
    REPLACE(salary,'\\u003F',null) as salary;

--  removing NULL Values
trainData = filter train2 by (age is not null) AND
    (workclass is not null) AND
    (fnlwgt is not null) AND
    (education is not null) AND
    (education_num  is not null) AND
    (marital_status is not null) AND
    (occupation is not null) AND
    (relationship is not null) AND
    (race is not null) AND
    (sex is not null) AND
    (capital_gain is not null) AND
    (capital_loss is not null) AND
    (hpw is not null) AND
    (native_cont is not null) AND
    (salary_greater_50k is not null);
----
```

The target variable of salary was having two types of values representing the above 50k income category and the below 50k category. We decided to make it a binary variable with '1' for above 50k and '0' for the other.

```
----
train2 = FOREACH train1 GENERATE age..native_cont, (salary == ' >50K'? '1':'0') as salary_greater_50k;
----
```

Then we sampled out 25% of the training data from the complete data set for data exploration and feature selection

```
----
smpTrain = SAMPLE train2 0.25;
countsmp = FOREACH (GROUP smpTrain ALL) GENERATE COUNT(smpTrain);
STORE smpTrain into '/user/root/project/smpTrain' using PigStorage(',');
----
```

After exploration we decided to remove the predictor 'education num' as it was redundant

```
finalTrainData = FOREACH trainData GENERATE .. education, marital_status ..;
```

As per the assumption that the data is very big to be handled at once, we split the complete training data into three parts. For splitting we defined a function in which random variable column is assigned with the function RANDOM() and then data is split with 0.33 and 0.66 as separators. The number of records in each set is then checked to see equal division.

```
----
DEFINE split_into_3(inputData)
RETURNS split1, split2, split3
{
   data = foreach $inputData generate RANDOM() as random_assignment, *;
   split data into split1 if random_assignment <= 0.33,
                                   split2 if random_assignment > 0.33 and random_assignment <= 0.66,
                                   split3 otherwise;
   $split1 = foreach split1 generate $1..;
   $split2 = foreach split2 generate $1..;
   $split3 = foreach split2 generate $1..;
};
train1, train2, train3 = split_into_3(finalTrainData);

--count the number of records in each training set
count1 = FOREACH (GROUP train1 ALL) GENERATE COUNT(train1);
count2 = FOREACH (GROUP train2 ALL) GENERATE COUNT(train2);
count3 = FOREACH (GROUP train3 ALL) GENERATE COUNT(train3);

-- store the three training sets
STORE train1 into '/user/root/project/train1' using PigStorage(',');
STORE train2 into '/user/root/project/train2' using PigStorage(',');
STORE train3 into '/user/root/project/train3' using PigStorage(',');
----
```

These training data sets were applied with various models. There was no need to create dummy variables for categorical predictors as R implicitly creates them if predictor is identified as a factor with more than one levels. Hence, these data sets were directly used for modelling.

The similar steps were followed to load the test data remove the 'education_no' column and the Null values.

--Load test data

```
test= LOAD '/user/root/project/adult_test.test' using PigStorage(',') AS
        ( age:chararray,
          workclass:chararray,
          fnlwgt:chararray,
          education:chararray,
          education_num:chararray,
          marital_status:chararray,
          occupation:chararray,
          relationship:chararray,
          race:chararray,
          sex:chararray,
          capital_gain:chararray,
          capital_loss:chararray,
          hpw:chararray,
          native_cont:chararray,
         salary:chararray );
----
```

--Remove the colum 'education_num'

```
test1 = FOREACH test GENERATE
         (int)REPLACE(age,'\\u003F',null) as age,
          REPLACE(workclass,'\\u003F',null) as workclass,
          REPLACE(fnlwgt,'\\u003F',null) as fnlwgt,
          REPLACE(education,'\\u003F',null) as education,
          REPLACE(marital_status,'\\u003F',null) as marital_status,
          REPLACE(occupation,'\\u003F',null) as occupation,
          REPLACE(relationship,'\\u003F',null) as relationship,
          REPLACE(race,'\\u003F',null) as race,
          REPLACE(sex,'\\u003F',null) as sex,
          (int)REPLACE(capital_gain,'\\u003F',null) as capital_gain,
          (int)REPLACE(capital_loss,'\\u003F',null) as capital_loss,
          (int)REPLACE(hpw,'\\u003F',null) as hpw,
          REPLACE(native_cont,'\\u003F',null) as native_cont,
          REPLACE(salary,'\\u003F',null) as salary;
```

After this test data was transformed with no Null values as well as binary target variable was generated and stored.

```
test2 = filter test1 by (age is not null) AND
          (workclass is not null) AND
          (fnlwgt is not null) AND
          (education is not null) AND
          (marital_status is not null) AND
          (occupation is not null) AND
          (relationship is not null) AND
          (race is not null) AND
          (sex is not null) AND
          (capital_gain is not null) AND
          (capital_loss is not null) AND
          (hpw is not null) AND
          (native_cont is not null) AND
          (salary is not null);

testData = FOREACH test2 GENERATE age..native_cont,
         (salary == ' >50K.'? '1':'0') as salary_greater_50k;

STORE testData INTO '/user/root/project/testData' using PigStorage(',');
----
```

DATA EXPLORATION (PART 1)

The smpTrain.csv file contains 25% sample, 7620 records of data that we took from the training data using PIG for exploration and feature selection.

In the file, names of column and details are **1)** age – continuous **2)** Workclass - Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked **3)** fnlwgt – continuous **4)** education - Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool **5)** education_num – continuous **6)** marital_status - Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse **7)** occupation - Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces **8)** relationship - Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried **9)** race - Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried **10)** sex - Female, Male **11)** capital_gain – continuous **12)** capital_loss – continuous **13)** hpw – hours-per-week **14)** native_country – United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands **15)** salarymorethan50k

```
```{r}
smpTrain = read.csv(file ='C:\\Users\\Angeles\\Desktop\\Big Data\\final project\\UCI-Adult\\smpTrain.csv',header=
FALSE,stringsAsFactors = TRUE)

#assigning predictor names
colnames(smpTrain) <- c('age','workclass','fnlwgt','education','education_num',
                        'marital_status', 'occupation', 'relationship', 'race',
                        'sex', 'capital_gain', 'capital_loss', 'hpw',
                        'native_country','salarymorethan50k')

smpTrain$age <- as.numeric(smpTrain$age)
smpTrain$fnlwgt <- as.numeric(smpTrain$fnlwgt)
smpTrain$capital_gain <- as.numeric(smpTrain$capital_gain)
smpTrain$capital_loss <- as.numeric(smpTrain$capital_loss)
smpTrain$hpw <- as.numeric(smpTrain$hpw)
```

DATA CLEANING

Because education number is redundant, we would like to remove the column.

```
smpTrain <- smpTrain[-5]
```
```

DATA EXPLORATION (PART 2)

After cleaning the data, now let's see a summary:

```
summary(smpTrain)
      age                       workclass         fnlwgt              education
 Min.    :17.00    Federal-gov      : 243    Min.   :  19302          age                         workclass          fnlwgt
education
 Min.    :17.00    Federal-gov      : 243    Min.   :  19302    HS-grad      :2448
 1st Qu.:28.00     Local-gov        : 500    1st Qu.: 117844    Some-college:1691
 Median :37.00     Private          :5625    Median : 177285    Bachelors    :1290
 Mean   :38.44     Self-emp-inc     : 269    Mean   : 189511    Masters      : 396
 3rd Qu.:47.00     Self-emp-not-inc: 655     3rd Qu.: 236714    Assoc-voc    : 318
 Max.   :90.00     State-gov        : 323    Max.   :1484705    11th         : 281
                   Without-pay      :   5                       (Other)      :1196
                 marital_status            occupation          relationship                race
  Divorced              :1109     Craft-repair    :1027    Husband        :3105    Amer-Indian-Eskimo:  78     HS-grad
:2448
 1st Qu.:28.00     Local-gov        : 500    1st Qu.: 117844    Some-college:1691
 Median :37.00     Private          :5625    Median : 177285    Bachelors    :1290
 Mean   :38.44     Self-emp-inc     : 269    Mean   : 189511    Masters      : 396
 3rd Qu.:47.00     Self-emp-not-inc: 655     3rd Qu.: 236714    Assoc-voc    : 318
 Max.   :90.00     State-gov        : 323    Max.   :1484705    11th         : 281
                   Without-pay      :   5                       (Other)      :1196
                 marital_status            occupation          relationship                race
  Divorced              :1109     Craft-repair    :1027    Husband        :3105    Amer-Indian-Eskimo:  78
  Married-AF-spouse     :   5     Exec-managerial:1018     Not-in-family :2009     Asian-Pac-Islander: 235
  Married-civ-spouse    :3480     Prof-specialty :1018     Other-relative: 236     Black             : 707
  Married-spouse-absent:  86      Adm-clerical   : 954     Own-child     :1123     Other             :  55
  Never-married         :2489     Sales          : 871     Unmarried     : 826     White             :6545
  Separated             : 238     Other-service  : 817     Wife          : 321
  Widowed               : 213    (Other)         :1915
  Married-AF-spouse     :   5     Exec-managerial:1018     Not-in-family :2009     Asian-Pac-Islander: 235
  Married-civ-spouse    :3480     Prof-specialty :1018     Other-relative: 236     Black             : 707
  Married-spouse-absent:  86      Adm-clerical   : 954     Own-child     :1123     Other             :  55
  Never-married         :2489     Sales          : 871     Unmarried     : 826     White             :6545
  Separated             : 238     Other-service  : 817     Wife          : 321
  Widowed               : 213    (Other)         :1915
      sex          capital_gain      capital_loss          hpw                native_country  salarymorethan50k
  Female:2459    Min.   :    0    Min.   :   0.00    Min.   : 1.00    United-States:6959    Min.   :0.0000
  Male  :5161    1st Qu.:    0    1st Qu.:   0.00    1st Qu.:40.00    Mexico       : 143    1st Qu.:0.0000
                 Median :    0    Median :   0.00    Median :40.00    Philippines  :  44    Median :0.0000
                 Mean   : 1118    Mean   :  85.93    Mean   :40.87    Germany      :  31    Mean   :0.2467
```

Then, we find the frequency of salary less than 50k or more than 50k. In the smpTrain.csv file, if the value is less than 50k, showing 0. If the value is more than 50k, showing 1. The value of 1, 1880/7620, is around 25%.

```
print(c('frequency of the target',table(smpTrain$salarymorethan50k)),quote= FALSE)
                                          0                        1
frequency of the target                5740                     1880
```

The skewness values of the numerical predictors are acceptable.

```
predictors <- smpTrain[c(1,3,10,11,12)]
skewValues <- apply(predictors,2,skewness)
print(skewValues)
      age       fnlwgt capital_gain capital_loss          hpw
0.4845819    1.5256418   11.6819329    4.5396290    0.3553208
```

TRAINING DATA

The training data was split into three parts using PIG Latin. The three sets of training data were saved in CSV files and were downloaded to a local system. We load the three files containing training data into R and continue with our data modelling process

Loading the first training set into R. We assign the data types to the variables and give the predictors appropriate names. We also make sure that the categorical predictors 'occupation' and 'native-country' have all the levels.

```
train1 <- read.csv(file = 'C:\\Users\\Angeles\\Desktop\\Big Data\\final project\\UCI-Adult\\train1.csv',header =
FALSE, stringsAsFactors = TRUE)

#assigning predictor names
colnames(train1) <- c('age','workclass','fnlwgt','education','marital_status',
                      'occupation', 'relationship','race', 'sex','capital_gain',
                      'capital_loss','hpw','native_country','salarymorethan50k')

train1$age  <- as.numeric(train1$age)
train1$fnlwgt  <- as.numeric(train1$fnlwgt)
train1$capital_gain  <- as.numeric(train1$capital_gain)
train1$capital_loss  <- as.numeric(train1$capital_loss)
train1$hpw  <- as.numeric(train1$hpw)
train1$salarymorethan50k <- as.factor(train1$salarymorethan50k)

levels(train1$occupation) <- c('Tech-support', 'Craft-repair', 'Other-service', 'Sales', 'Exec-managerial',
'Prof-specialty', 'Handlers-cleaners', 'Machine-op-inspct', 'Adm-clerical', 'Farming-fishing', 'Transport-moving',
'Priv-house-serv', 'Protective-serv', 'Armed-Forces')

levels(train1$native_country) <- c('United-States', 'Cambodia', 'England', 'Puerto-Rico', 'Canada', 'Germany',
'Outlying-US(Guam-USVI-etc)', 'India', 'Japan', 'Greece', 'South', 'China', 'Cuba', 'Iran', 'Honduras',
'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico', 'Portugal', 'Ireland', 'France',
'Dominican-Republic', 'Laos', 'Ecuador', 'Taiwan', 'Haiti', 'Columbia', 'Hungary', 'Guatemala', 'Nicaragua',
'Scotland', 'Thailand', 'Yugoslavia', 'El-Salvador', 'Trinadad&Tobago', 'Peru', 'Hong', 'Holand-Netherlands')
```

Training set 2:

Loading the second training set into R

```
train2 <- read.csv(file = 'C:\\Users\\Angeles\\Desktop\\Big Data\\final project\\UCI-Adult\\train2.csv',header =
FALSE, stringsAsFactors = TRUE)

#assigning predictor names
colnames(train2) <- c('age','workclass','fnlwgt','education','marital_status',
                      'occupation', 'relationship','race', 'sex','capital_gain',
                      'capital_loss','hpw','native_country','salarymorethan50k')

train2$age  <- as.numeric(train2$age)
train2$fnlwgt  <- as.numeric(train2$fnlwgt)
train2$capital_gain  <- as.numeric(train2$capital_gain)
train2$capital_loss  <- as.numeric(train2$capital_loss)
train2$hpw  <- as.numeric(train2$hpw)
train2$salarymorethan50k <- as.factor(train2$salarymorethan50k)


levels(train2$occupation) <- c('Tech-support', 'Craft-repair', 'Other-service', 'Sales', 'Exec-managerial',
'Prof-specialty', 'Handlers-cleaners', 'Machine-op-inspct', 'Adm-clerical', 'Farming-fishing', 'Transport-moving',
'Priv-house-serv', 'Protective-serv', 'Armed-Forces')

levels(train2$native_country) <- c('United-States', 'Cambodia', 'England', 'Puerto-Rico', 'Canada', 'Germany',
'Outlying-US(Guam-USVI-etc)', 'India', 'Japan', 'Greece', 'South', 'China', 'Cuba', 'Iran', 'Honduras',
'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico', 'Portugal', 'Ireland', 'France',
'Dominican-Republic', 'Laos', 'Ecuador', 'Taiwan', 'Haiti', 'Columbia', 'Hungary', 'Guatemala', 'Nicaragua',
'Scotland', 'Thailand', 'Yugoslavia', 'El-Salvador', 'Trinadad&Tobago', 'Peru', 'Hong', 'Holand-Netherlands')
```

Training set 3:

Loading the third training set into R

```
train3 <- read.csv(file = 'C:\\Users\\Angeles\\Desktop\\Big Data\\final project\\UCI-Adult\\train3.csv',header =
FALSE, stringsAsFactors = TRUE)

#assigning predictor names
colnames(train3) <- c('age','workclass','fnlwgt','education','marital_status',
                      'occupation', 'relationship','race', 'sex','capital_gain',
                      'capital_loss','hpw','native_country','salarymorethan50k')

train3$age  <- as.numeric(train3$age)
train3$fnlwgt  <- as.numeric(train3$fnlwgt)
train3$capital_gain  <- as.numeric(train3$capital_gain)
train3$capital_loss  <- as.numeric(train3$capital_loss)
train3$hpw  <- as.numeric(train3$hpw)
train3$salarymorethan50k <- as.factor(train3$salarymorethan50k)


levels(train3$occupation) <- c('Tech-support', 'Craft-repair', 'Other-service', 'Sales', 'Exec-managerial',
'Prof-specialty', 'Handlers-cleaners', 'Machine-op-inspct', 'Adm-clerical', 'Farming-fishing', 'Transport-moving',
'Priv-house-serv', 'Protective-serv', 'Armed-Forces')

levels(train3$native_country) <- c('United-States', 'Cambodia', 'England', 'Puerto-Rico', 'Canada', 'Germany',
'Outlying-US(Guam-USVI-etc)', 'India', 'Japan', 'Greece', 'South', 'China', 'Cuba', 'Iran', 'Honduras',
'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico', 'Portugal', 'Ireland', 'France',
'Dominican-Republic', 'Laos', 'Ecuador', 'Taiwan', 'Haiti', 'Columbia', 'Hungary', 'Guatemala', 'Nicaragua',
'Scotland', 'Thailand', 'Yugoslavia', 'El-Salvador', 'Trinadad&Tobago', 'Peru', 'Hong', 'Holand-Netherlands')
```

Note : The modelling function used in R handle categorical variables(factors) well and do implicit dummy variable creation. Therefore we do not need to create dummy variables.

TEST DATA

The test data was edited in PIG and saved as CSV file. It was downloaded to the local system. Here we load it into R assign the predictor names.

```
#load test data
test <- read.csv(file = 'testData.csv',header = FALSE, stringsAsFactors = TRUE)

#assigning predictor names
colnames(test) <- c('age','workclass','fnlwgt','education','marital_status',
                    'occupation', 'relationship','race', 'sex','capital_gain',
                    'capital_loss','hpw','native_country','salarymorethan50k')

test$age  <- as.numeric(test$age)
test$fnlwgt  <- as.numeric(test$fnlwgt)
test$capital_gain  <- as.numeric(test$capital_gain)
test$capital_loss  <- as.numeric(test$capital_loss)
test$hpw  <- as.numeric(test$hpw)
test$salarymorethan50k <- as.factor(test$salarymorethan50k)
```

BALANCE OF TARGET VARIABLE CLASSES

We want to see the distribution of classes in the target variable in the test and the training sets. From the result, the classes '0' and '1' is unbalanced.

Input	Output
`table(train1$salarymorethan50k)` First training set	`0 1` `7536 2463`
`table(train2$salarymorethan50k)` Second training set	`0 1` `7520 2525`
`table(train3$salarymorethan50k)` Third training set	`0 1` `7456 2474`
`table(test$salarymorethan50k)` Fourth training set	`0 1` `11360 3700`

As there is class imbalance in our dataset, it would affect our model accuracy. To deal with this, we smote the data, i.e. we over sample the class which is minority. (Smote: synthetic minority over-sampling technique)

HANDLING CLASS IMBALANCE

Here using SMOTE package of our we under samples the majority class '0' and oversample the majority class '1' to balance the classes. SMOTE functions oversamples by creating synthetic data using KNN. The number of neighbors used by default = 5.

```
set.seed(111)
smoteTrain1 <- SMOTE(salarymorethan50k ~ .,train1,perc.over = 100, perc.under = 200)
table(smoteTrain1$salarymorethan50k)


levels(smoteTrain1$occupation) <- c('Tech-support', 'Craft-repair', 'Other-service', 'Sales', 'Exec-managerial',
'Prof-specialty', 'Handlers-cleaners', 'Machine-op-inspct', 'Adm-clerical', 'Farming-fishing', 'Transport-moving',
'Priv-house-serv', 'Protective-serv', 'Armed-Forces')

levels(smoteTrain1$native_country) <- c('United-States', 'Cambodia', 'England', 'Puerto-Rico', 'Canada',
'Germany', 'Outlying-US(Guam-USVI-etc)', 'India', 'Japan', 'Greece', 'South', 'China', 'Cuba', 'Iran', 'Honduras',
'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico', 'Portugal', 'Ireland', 'France',
'Dominican-Republic', 'Laos', 'Ecuador', 'Taiwan', 'Haiti', 'Columbia', 'Hungary', 'Guatemala', 'Nicaragua',
'Scotland', 'Thailand', 'Yugoslavia', 'El-Salvador', 'Trinadad&Tobago', 'Peru', 'Hong', 'Holand-Netherlands')
```

```
set.seed(111)
smoteTrain2 <- SMOTE(salarymorethan50k ~ .,train2,perc.over = 100, perc.under = 200)
table(smoteTrain2$salarymorethan50k)


levels(smoteTrain2$occupation) <- c('Tech-support', 'Craft-repair', 'Other-service', 'Sales', 'Exec-managerial',
'Prof-specialty', 'Handlers-cleaners', 'Machine-op-inspct', 'Adm-clerical', 'Farming-fishing', 'Transport-moving',
'Priv-house-serv', 'Protective-serv', 'Armed-Forces')

levels(smoteTrain2$native_country) <- c('United-States', 'Cambodia', 'England', 'Puerto-Rico', 'Canada',
'Germany', 'Outlying-US(Guam-USVI-etc)', 'India', 'Japan', 'Greece', 'South', 'China', 'Cuba', 'Iran', 'Honduras',
'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico', 'Portugal', 'Ireland', 'France',
'Dominican-Republic', 'Laos', 'Ecuador', 'Taiwan', 'Haiti', 'Columbia', 'Hungary', 'Guatemala', 'Nicaragua',
'Scotland', 'Thailand', 'Yugoslavia', 'El-Salvador', 'Trinadad&Tobago', 'Peru', 'Hong', 'Holand-Netherlands')
```

```
set.seed(111)
smoteTrain3 <- SMOTE(salarymorethan50k ~ .,train3,perc.over = 100, perc.under = 200)
table(smoteTrain3$salarymorethan50k)


levels(smoteTrain3$occupation) <- c('Tech-support', 'Craft-repair', 'Other-service', 'Sales', 'Exec-managerial',
'Prof-specialty', 'Handlers-cleaners', 'Machine-op-inspct', 'Adm-clerical', 'Farming-fishing', 'Transport-moving',
'Priv-house-serv', 'Protective-serv', 'Armed-Forces')

levels(smoteTrain3$native_country) <- c('United-States', 'Cambodia', 'England', 'Puerto-Rico', 'Canada',
'Germany', 'Outlying-US(Guam-USVI-etc)', 'India', 'Japan', 'Greece', 'South', 'China', 'Cuba', 'Iran', 'Honduras',
'Philippines', 'Italy', 'Poland', 'Jamaica', 'Vietnam', 'Mexico', 'Portugal', 'Ireland', 'France',
'Dominican-Republic', 'Laos', 'Ecuador', 'Taiwan', 'Haiti', 'Columbia', 'Hungary', 'Guatemala', 'Nicaragua',
'Scotland', 'Thailand', 'Yugoslavia', 'El-Salvador', 'Trinadad&Tobago', 'Peru', 'Hong', 'Holand-Netherlands')
```

After using smote on our three training sets we see the distribution of classes in the target variable. From the result we can see that the classes '0' and '1' are balanced

Input	Output
`table(smoteTrain1$salarymorethan50k)` First training set after SMOTE	0 1 4926 4926
`table(smoteTrain2$salarymorethan50k)` Second training set after SMOTE	0 1 5050 5050
`table(smoteTrain3$salarymorethan50k)` Third training set after SMOTE	0 1 4948 4948

We are going to proceed with these balanced training sets.

MAJORITY VOTE FUNCTION

Now, we create a function that takes the majority vote. If two more models predict the test record as ‘1’, we can say the individual’s income is greater than 50k, and vice versa.

```
judge <- function(row){
        row <- as.numeric(row)
        if (sum(row) >= 2){
                pred <- 1
        }else{
                pred <- 0
        }
        return(pred)
}
```

MODEL 1: DECISION TREE

Overfitting is a significant practical difficulty for decision tree models and many other predictive models. It happens when the learning algorithm continues to develop hypotheses that reduce training set error at the cost of an increased test set error. So, we prune back the tree to avoid overfitting the data. We select a tree size that minimize the cross-validated error.

Function to create and prune a decision tree

```
createTreeModel <- function(trainData){
   treeModel <- rpart(formula = salarymorethan50k ~ .,data = trainData,
                      method = 'class',control = rpart.control(maxdepth = 30))

  ptreeModel <- prune(treeModel,
           cp =treeModel$cptable[which.min(treeModel$cptable[,"xerror"]),"CP"])


     return(ptreeModel)
}
```

Creating decision tree for the first training set

```
#for first set of training Data
tree1 <- createTreeModel(smoteTrain1)
predictTree1 <- predict(tree1,test[,-14],type = 'class')
tree1Cnm <- confusionMatrix(data = predictTree1,
                            reference = test$salarymorethan50k)
print(tree1Cnm$overall[1])
```

For the first set of training data, the accuracy is 0.784595.

Creating decision tree for the second training set

```
#for second set of training Data
tree2 <- createTreeModel(smoteTrain2)
predictTree2 <- predict(tree2,test[,-14], type ='class')
tree2Cnm <- confusionMatrix(data = predictTree2,
                            reference = test$salarymorethan50k)
print(tree2Cnm$overall[1])
```

For the second set of training data, the accuracy is 0.8077025.

Creating decision tree for the third training set

```
#for third set of training Data
tree3 <- createTreeModel(smoteTrain3)
predictTree3 <- predict(tree3,test[,-14], type ='class')
tree3Cnm <- confusionMatrix(data = predictTree3,
                            reference = test$salarymorethan50k)
print(tree3Cnm$overall[1])
```

For the third set of training data, the accuracy is 0.8028552.

Combined all the predictions into one data frame:

```
#combine all the predictions into one
treePredictions <- data.frame(predictTree1,predictTree2,predictTree3)
```

Call the 'judge' function created to take the majority vote:

```
#take the majority vote
finalTreePred <- as.factor(apply(treePredictions,1,judge))
```

Then build the confusion matrix of the majority vote and calculate the final accuracy.

```
#take the majority vote
finalTreePred <- as.factor(apply(treePredictions,1,judge))

#build the confusin matrix
finalTreeCnm <- confusionMatrix(data = finalTreePred,
                                reference = test$salarymorethan50k)
#final accuracy
print(finalTreeCnm$table)
print(finalTreeCnm$overall[1])

```

The confusion matrix is as follows:

```
            Reference
Prediction     0    1
          0 9471 1068
          1 1889 2632
```

The accuracy is 0.8036521, which is calculated by (9471+2632) / (9471+1068+1889+2632).

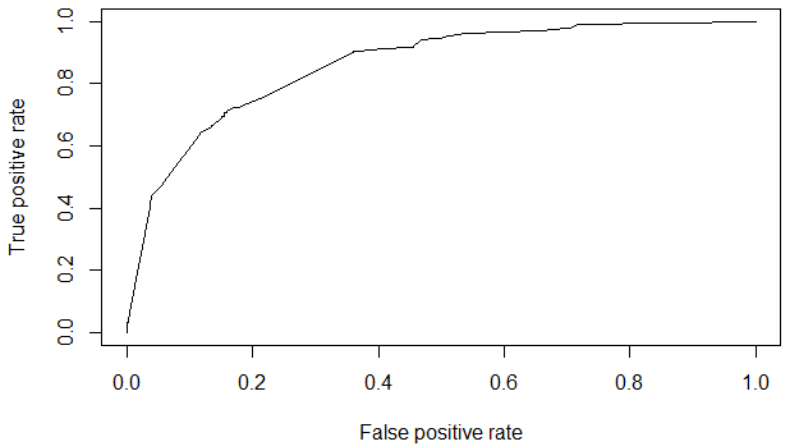
Lastly, we plot the ROC, which is a plot of the true positive rate against the false positive rate for the different possible cutpoints. Since the curve follows the left-hand border and then the top border of the ROC space, we can say it is an accurate test.

```
#plot ROC
predictTree1Prob <- predict(tree1,test[,-14],type = 'prob')
predictTree2Prob <- predict(tree2,test[,-14],type = 'prob')
predictTree3Prob <- predict(tree3,test[,-14],type = 'prob')

treePredictionsProb <- data.frame(predictTree1Prob[,2],predictTree2Prob[,2],
                                  predictTree3Prob[,2])

finalTreePredProb <- apply(treePredictionsProb,1,prod)

prefTree<-performance(prediction(finalTreePredProb,test$salarymorethan50k),
                  measure = "tpr",x.measure = "fpr")
plot(prefTree)
```




```r
aucTree<-performance(prediction(finalTreePredProb,test$salarymorethan50k),
                  measure = "auc")
print(c("Area Under the Curve: ", aucTree@y.values[[1]]),quote = FALSE)
```

AUC (Area Under the Curve) is literally the pertentage of the box that is under the curve. The value lies between 0.5 to 1. A very poor classifier has an AUC of around 0.5, while an excellent classifier has an AUC of 1. In our result, we have an AUC of 0.859601553578226.

MODEL 2: NAIVE BAYES

Secondly, we choose Naïve Bayes model. It is based on Bayes' theorem with an assumption of independence among predictors.

Function to create Naïve Bayes Model

```r
```{r}

createNbModel <- function(trainData){

        nbModel <- naiveBayes(formula = salarymorethan50k ~ .,data = trainData)

        return(nbModel)
}
```
```

We create models for three dataset, get the predictions for the individual models.

```
#create modles for the three data sets
nbModel1 <- createNbModel(smoteTrain1)
nbModel2 <- createNbModel(smoteTrain2)
nbModel3 <- createNbModel(smoteTrain3)

#get the predictions of the models
predNbModel1 <- predict(nbModel1,test, type = 'class')
predNbModel2 <- predict(nbModel2,test, type = 'class')
predNbModel3 <- predict(nbModel3,test, type = 'class')
```

We get the accuracy of each individual. Accuracy of the first model is 0.8146082, the second model is 0.8108898, and the third model is 0.8010624.

```
#create modles for the three data sets
nbModel1 <- createNbModel(smoteTrain1)
nbModel2 <- createNbModel(smoteTrain2)
nbModel3 <- createNbModel(smoteTrain3)

#get the predictions of the models
predNbModel1 <- predict(nbModel1,test, type = 'class')
predNbModel2 <- predict(nbModel2,test, type = 'class')
predNbModel3 <- predict(nbModel3,test, type = 'class')

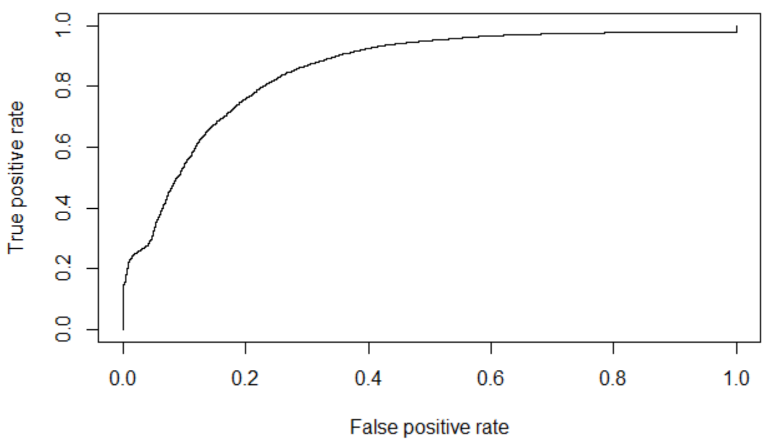
#get the accuracy of each individual model
nbCnm1 <- confusionMatrix(data = predNbModel1,
                          reference = test$salarymorethan50k)
print(nbCnm1$overall[1])

nbCnm2 <- confusionMatrix(data = predNbModel2,
                          reference = test$salarymorethan50k)
print(nbCnm2$overall[1])
nbCnm3 <- confusionMatrix(data = predNbModel3,
                          reference = test$salarymorethan50k)
print(nbCnm3$overall[1])

```

Getting the majority vote on all the three predictions using the 'judge' function we created and then calculating the final predictions, accuracy and confusion matrix.

```
#collect all the predictions
predictionsNb <- data.frame(predNbModel1,predNbModel2,
                            predNbModel3)

finalNbPred <- as.factor(apply(predictionsNb,1,judge))
finalNbCnm <- confusionMatrix(data = finalNbPred,
                              reference = test$salarymorethan50k)

#print the confusion matrix and the accuracy
print(finalNbCnm$table)
print(finalNbCnm$overall[1])

```

Again, we print the overall confusion matrix and the final accuracy. The overall accuracy is 0.8112882.

```
 Accuracy
0.8112882
```

```
          Reference
Prediction     0     1
         0 10425  1907
         1   935  1793
```

Here are how the ROC looks like and the value of AUC is 0.852711065378735.



MODEL 3: ARTIFICIAL NEURAL NETWORK

Determining the size of a neural network is important and difficult. A network with too few hidden units gives poor predictions for new data, because it has too little flexibility (it has a large bias). Increasing the size of a neutral network may lead to better fits on training data, but may result in overfitting and poor predictions. However, too many hidden units will give us poor generalization because it fits too much to the noise on training data (it has a large variance).

Although there are many ways, the existing comparison procedures fall into two main categories: analytical approaches, and resampling. Here, we use resampling based methods that involve much more computation. They remove the risk of making faulty statements due to unsatisfied assumptions and create optimal tuning parameter by using function of 5-fold cross validation for all the training set. Then, we use those tuned hyper parameters to create models

Function to create ANN

```
```{r,message = FALSE,warning = FALSE}
createANN <- function(trainData,testData,nodes,tune = FALSE){

        ANNModel <- nnet(salarymorethan50k ~., data = trainData, size = nodes,
                           maxit = 2000,trace = FALSE, MaxNWts = 20000)

        predANN <- predict(ANNModel,testData[,-14], type = 'class')
        ANNCnm <- confusionMatrix(data = predANN,
                                  reference = testData$salarymorethan50k)
        if(tune == TRUE){
                return(ANNCnm$overall[1])
        }else if (tune == FALSE){
                return(ANNModel)
        }
}
```

Function tunes takes the data uses 5- Fold Cross validation to create ANN to get the best number of nodes to be used in the hidden layer

```
tuneANN <- function(trainData){

        #Using 5-fold CrossValiadation on 1st training set
        set.seed(111)
        cv_train_index <- createFolds(seq_len(nrow(trainData)), k = 5, list = TRUE,
                                                    returnTrain = TRUE)
        acc <- NA
        avgacc <- NA
        k = 1
        for(j in seq(from = 1,to = 10 ,by = 1)){
                acc <- NA
                for(i in 1:5){
                        acc[i] <- createANN(trainData[cv_train_index[[i]],],
                                   trainData[-cv_train_index[[i]],], j,tune = TRUE)
                }
        avgacc[k] <- mean(acc)
        k <- k+1
        }

ledger_mat <- data.frame(seq(from = 1, to = 10, by = 1),avgacc)

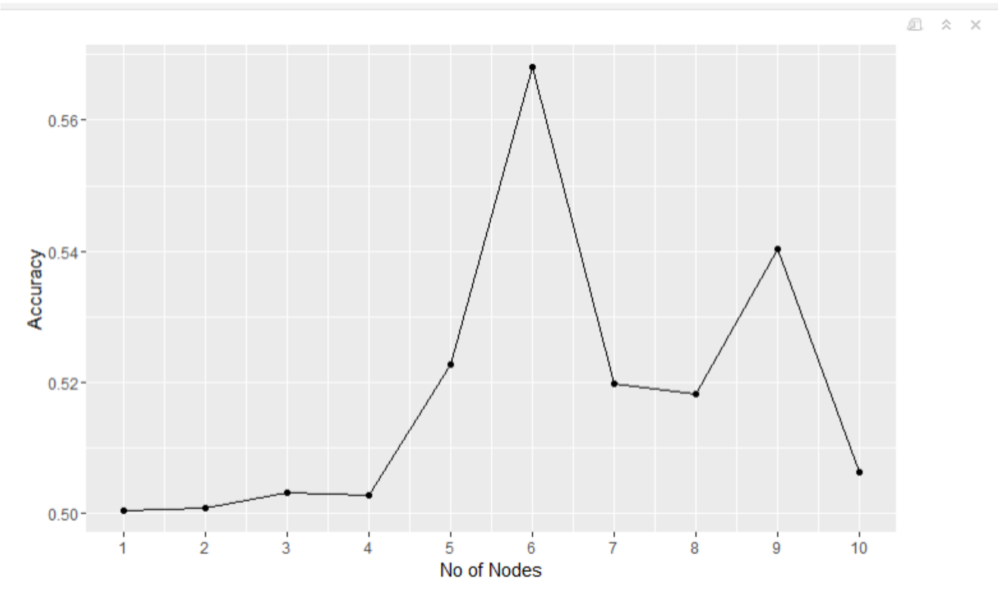
tunePlot <- ggplot(ledger_mat, aes(x = ledger_mat[,1], y = ledger_mat[,2]))+
        geom_point()+ geom_line()+xlab('No of Nodes')+ylab('Accuracy')+
        scale_x_continuous(breaks = round(seq(1,10,1),1))

print(tunePlot)
return(ledger_mat[which.max(ledger_mat[,2]),1])
}
```

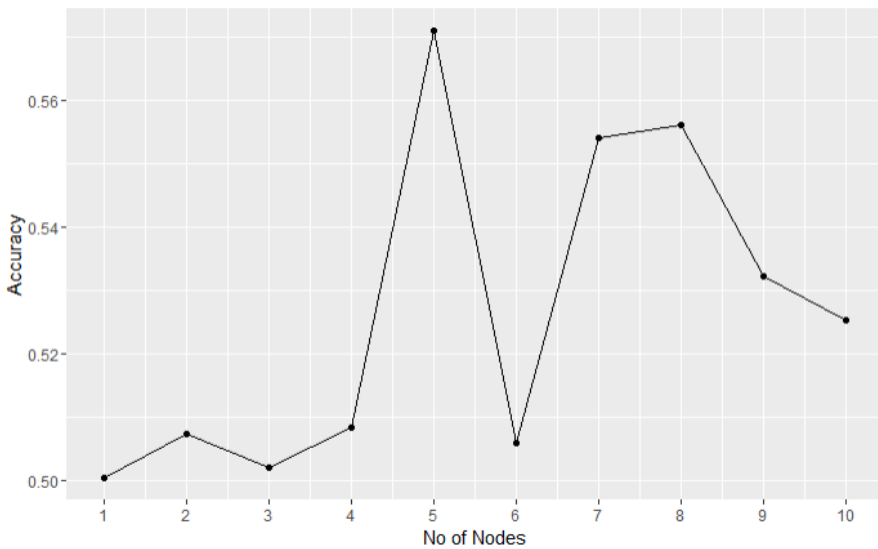
We use the tuneANN function created above to get the best number of nodes for each of the training data.
```

```
#tune first training set using 5 fold cross validation
size1 <- tuneANN(smoteTrain1)
print(c('No of nodes in the hidden layer',size1), quote = FALSE)
#tune second training set using 5 fold cross validation
size2 <- tuneANN(smoteTrain2)
print(c('No of nodes in the hidden layer',size2), quote = FALSE)
#tune third training set using 5 fold cross validation
size3 <- tuneANN(smoteTrain3)
print(c('No of nodes in the hidden layer',size3), quote = FALSE)
```

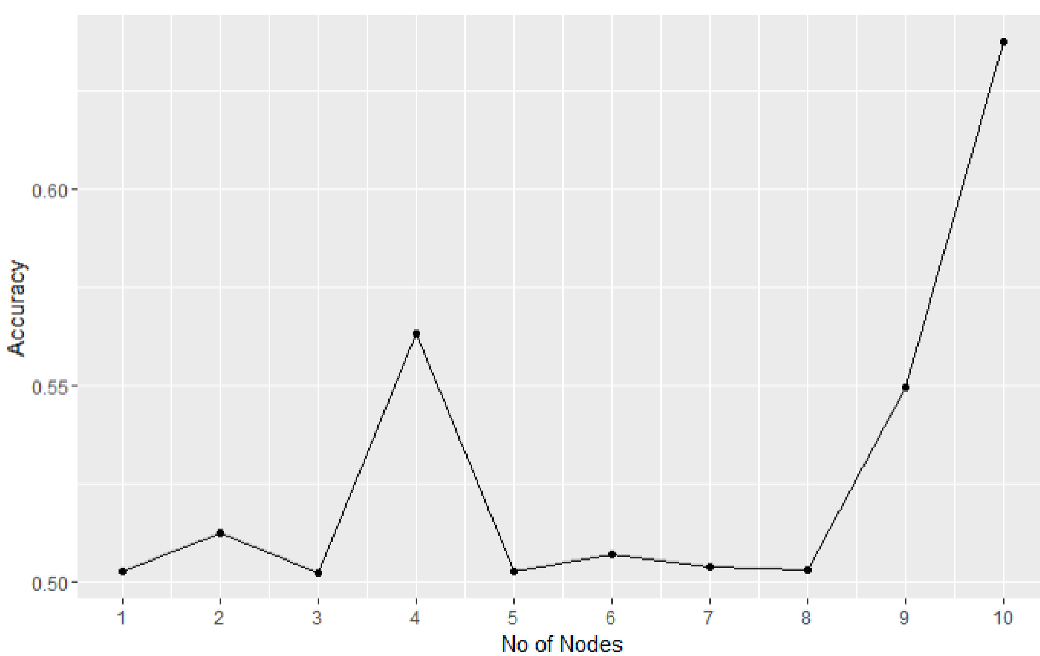
The figure below show the tuning profile for the first data set. Number of nodes in the hidden layer which produces best cross validated accuracy = 6. Will use 6 nodes in the hidden layer to make ANN for the first training set



The figure below show the tuning profile for the second data set. Number of nodes in the hidden layer which produces best cross validated accuracy = 5. Will use 5 nodes in the hidden layer to make ANN for the second training set



The figure below show the tuning profile for the third data set. Number of nodes in the hidden layer which produces best cross validated accuracy = 10. Will use 10 nodes in the hidden layer to make ANN for the third training set



We then use these number of nodes to create three ANN's for the training sets

```
#creating the three models using the tuned hyper parameter
ANN1 <- createANN(smoteTrain1,test,nodes = size1)
ANN2 <- createANN(smoteTrain2,test,nodes = size2)
ANN3 <- createANN(smoteTrain3,test,nodes = size3)

```

Getting predictions and accuracies for individual models.

```
predANN1 <- predict(ANN1,test, type = 'class')
ANNCnm1 <- confusionMatrix(data = predANN1,
                           reference = test$salarymorethan50k)
print(ANNCnm1$table)
print(ANNCnm1$overall[1])

predANN2 <- predict(ANN2,test, type = 'class')
ANNCnm2 <- confusionMatrix(data = predANN2,
                           reference = test$salarymorethan50k)
print(ANNCnm2$table)
print(ANNCnm2$overall[1])

predANN3 <- predict(ANN3,test, type = 'class')
ANNCnm3 <- confusionMatrix(data = predANN3,
                           reference = test$salarymorethan50k)
print(ANNCnm3$table)
print(ANNCnm3$overall[1])

```

The accuracy for the first ANN model is:

```
 Accuracy
0.7568393
```

The confusion matrix of the first ANN model is as below:

```
          Reference
Prediction     0     1
         0 11360  3662
         1     0    38
```

The accuracy for the second ANN model is:

```
Accuracy
0.7575033
```

The confusion matrix of the second ANN model is as below:

```
          Reference
Prediction     0     1
         0 11360  3652
         1     0    48
```

The accuracy for the third ANN model is

```
Accuracy
0.7573041
```

The confusion matrix of the third ANN model is as below:

```
          Reference
Prediction     0     1
         0 11360  3655
         1     0    45
```

Getting the majority vote on all the three predictions and then calculating the final predictions, accuracy and confusion matrix

```
#collection all the 3 sets of predicitions
predictionANN <- data.frame(predANN1,predANN2,predANN3)

#taking a majority vote for each record of test data
finalANNpredict <- as.factor(apply(predictionANN,1,judge))

finalANNCnm <- confusionMatrix(data = finalANNpredict,
                               reference = test$salarymorethan50k)
print(finalANNCnm$overall[1])
print(finalANNCnm$table)
```

The overall accuracy is 0.7861222,

```
 Accuracy
0.7861222
```

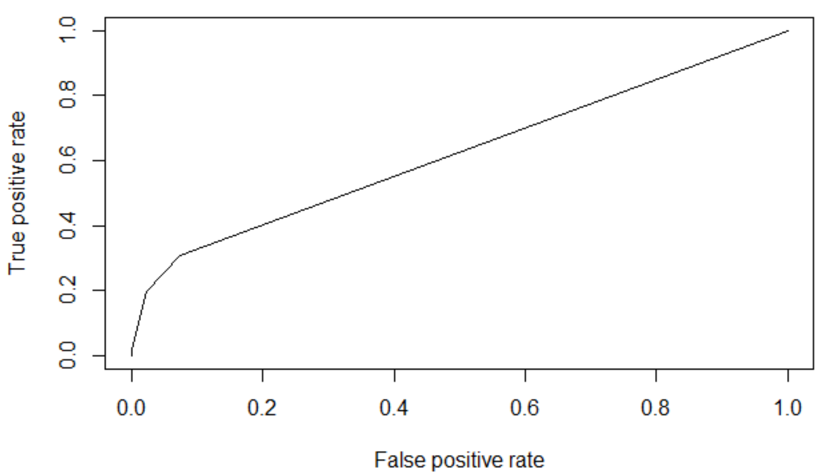
The final Confusion Matrix

```
          Reference
Prediction     0     1
         0 11105  2966
         1   255   734
```

Plot the ROC and Calculate the area under the curve.

```r
#Pot ROC
predANN1Prob <- predict(ANN1,test, type = 'raw')
predANN2Prob <- predict(ANN2,test, type = 'raw')
predANN3Prob <- predict(ANN3,test, type = 'raw')

ANNPredictionsProb <- data.frame(predANN1Prob,predANN2Prob,predANN3Prob)

finalANNPredProb <- apply(ANNPredictionsProb,1,prod)

prefANN<-performance(prediction(finalANNPredProb,test$salarymorethan50k),
                  measure = "tpr",x.measure = "fpr")
plot(prefANN)

#auc
aucANN <- performance(prediction(finalANNPredProb,test$salarymorethan50k),measure = "auc")
print(c("Area Under the Curve: ", aucANN@y.values[[1]]),quote = FALSE)

```

AUC is 0.62151739151123 and the ROC looks like this.



MODEL 4: SUPPORT VECTOR MACHINE

Support Vector Machine, known as SVM, is a supervised learning technique from the field of machine learning applicable to both classification and regression. It is a prediction tool to maximize predictive accuracy while automatically avoiding over-fit to the data.

The steps are the same as above: we create models for three datasets, get the predictions individually,then we take a majority vote, get the confusion matrix, calculate the accuracy, plot the ROC, and get the value of AUC.`

Using 5-Fold Cross Validation we determined that the best kenel function to use was 'Radial Basis kernel "Gaussian"' : rbfdot and the cost of constraints violation 'C' should be 1.25

Function to create SVM

```
#Function to create SVM model and return predictions
createSVM <- function(trainData){

        SVM <- ksvm( salarymorethan50k~.,data=trainData, type="C-bsvc",
                     kernel= 'rbfdot', C= 1.25, prob.model=TRUE)

          return(SVM)
}
```

We used the createSVM function to create models for the three training sets.

```
#Create models for the three data sets
SVM1 <- createSVM(smoteTrain1)
SVM2 <- createSVM(smoteTrain2)
SVM3 <- createSVM(smoteTrain3)
```

Overview of the SVM Model on the first training set :

```
Support Vector Machine object of class "ksvm"

SV type: C-bsvc  (classification)
 parameter : cost C = 1.25

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0803626053477514

Number of Support Vectors : 4208

Objective Function Value : -4224.462
Training error : 0.125051
Probability model included.
```

Got the predictions for the individual models on the test data and got the confusion matrix and the individual model accuracies.

```
#Get the acurracy of each individual model
SVMCnm1 <- confusionMatrix(data = predictSVM1,
                           reference = test$salarymorethan50k)
print(SVMCnm1$overall[1])
SVMCnm2 <- confusionMatrix(data = predictSVM2,
                           reference = test$salarymorethan50k)
print(SVMCnm2$overall[1])
SVMCnm3 <- confusionMatrix(data = predictSVM3,
                           reference = test$salarymorethan50k)
print(SVMCnm3$overall[1])
```

The accuracy of the first model was 0.8195219, of the second model was 0.7994024, and of the third model was 0.82251.

Then we get the majority vote using the ‘judge’ function, created the final confusion matrix and calculated the accuracy.

```
#majority vote and final accuracy calculation
predictionSVM <- data.frame(predictSVM1,predictSVM2,predictSVM3)
finalSVMpredict <- as.factor(apply(predictionSVM,1,judge))

finalSVMCnm <- confusionMatrix(data = finalSVMpredict,
                               reference = test$salarymorethan50k)
print(finalSVMCnm$overall[1])
print(finalSVMCnm$table)
```

The final confusion matrxi was :

```
          Reference
Prediction    0    1
         0 9509  891
         1 1851 2809
```

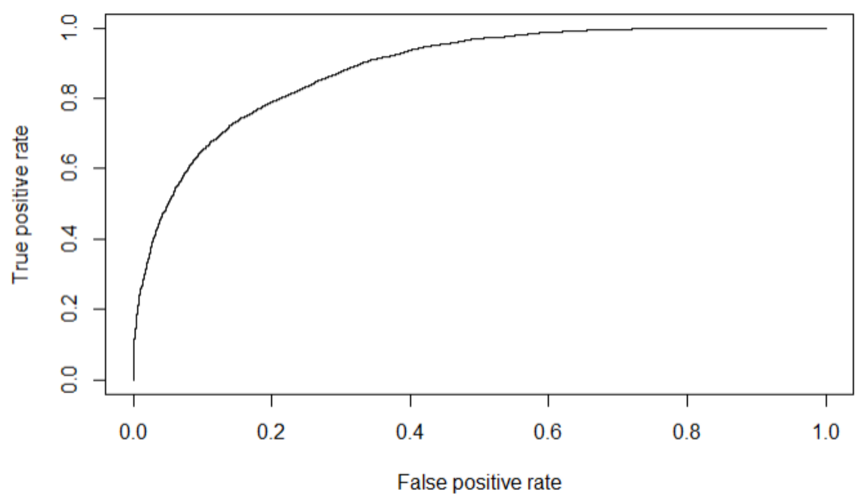
The final accuracy using SVM was :

```
 Accuracy
0.8179283
```

Plotting the ROC and calculating the AUC

```r
#Plot ROC
predictSVM1Prob <- predict(SVM1, test, type = 'probabilities')
predictSVM2Prob <- predict(SVM1, test, type = 'probabilities')
predictSVM3Prob <- predict(SVM1, test, type = 'probabilities')

SVMPredictionsProb <- data.frame(predictSVM1Prob[,2],predictSVM2Prob[,2],
                                 predictSVM3Prob[,2])

finalSVMPredProb <- apply(SVMPredictionsProb,1,prod)
finalSVMPredProb <- finalSVMPredProb

prefSVM <-performance(prediction(finalSVMPredProb,test$salarymorethan50k),
                  measure = "tpr",x.measure = "fpr")
plot(prefSVM)

#auc
aucSVM <- performance(prediction(finalSVMPredProb,test$salarymorethan50k),measure = "auc")
print(c("Area Under the Curve: ", aucSVM@y.values[[1]]),quote = FALSE)
```

The AUC was calculated to be 0.886122347259239, and the below graph is ROC.



CONCLUSION

Through the project, the handling and analysis of big data was executed under big data environment/tools and the data was classified using various algorithm on the sliced and diced data to come up with appropriate model for complete big data set.

Based on above analysis:

Decision tree has accuracy of ROC 0.80 and AUC 0.86.

Naïve Baye has accuracy of ROC 0.81 and AUC 0.85.

Artificial Neural Network has accuracy of ROC 0.79, and AUC 0.62.

Support Vector Machine has accuracy of ROC 0.82 and AUC 0.89.

Since SVM has the highest values, we can conclude that SVM is the best model.