Capstone Project II

Tarasov Oleksiy

6/2/2019

Introduction

This report is part of the HarvardX Certification of the edX HarvardX: PH125.9x Data Science: Capstone Online Training The data was taken in acccordance with the proposal of the course from Kaggle curated list of datasets link https://www.kaggle.com/annavictoria/ml-friendly-public-datasets?utm_medium=email&utm_source=intercom&utm_campaign=data+projects+onboarding (https://www.kaggle.com/annavictoria/ml-friendly-public-datasets?utm_medium=email&utm_source=intercom&utm_campaign=data+projects+onboarding) The following dataset were chosen for the analysis

(https://www.kaggle.com/uciml/adult-census-income) https://www.kaggle.com/uciml/adult-census-income (https://www.kaggle.com/uciml/adult-census-income) in case that connection to the Kaggle will not work I have copied the file to my github directory https://github.com/alexej-tarasov/CapstoneII (https://github.com/alexej-tarasov/CapstoneII) file adult.csv

Overview

The target cell is the column which predicts whether the person has incomes more or less then 50K USD annually. In this paper the data first will be shared to test and validation data, second the features will be evaluated based on its importance, third different models will be used to predict the values and in the end the best model will be selected based on accuaracy. The package "caret" of the language R is used for making prediction.

Methods/analysis

This section explains the process and techniques used, such as data cleaning, data exploration and visualization, any insights gained, and your modeling approach; First libraries for analysis should be activated

```
library(dplyr)
library(caret)
library(tidyr)
library(plotly)
library(tidyverse)
library(lubridate)
library(broom)
library(ggplot2)
library(RCurl)
library(kableExtra)
library(e1071)
library(parallel)
library(doParallel)
library(rpart)
library(caTools)
library(Rborist)
library(randomForest)
library(gbm)
```

In this section data will be read If you have any problem with the download of this dataset please download it from my github https://github.com/alexej-tarasov/CapstoneII (https://github.com/alexej-tarasov/CapstoneII)

```
#data<-read.csv("~/CapstoneII/CapstoneII/adult.csv")
data <- read.csv("https://raw.githubusercontent.com/alexej-tarasov/CapstoneII/master/adult.csv")# please see comm
ents above if you have any problem with download
```

The data has following attributes

Target income: >50K, <=50K

age: continuous

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked

fnlwgt: continuous

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

education-num: continuous

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse

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

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black

sex: Female, Male

capital-gain: continuous

capital-loss: continuous

hours-per-week: continuous

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

Below the example of the top10 rows is shown

```
as_tibble(head(data,n=10))%>%kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.we
90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	
82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	
66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	
54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	
41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	
34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	3770	
38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0	3770	
74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	0	3683	
68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	0	3683	
41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	0	3004	

The data is separated to the test and validation set using the caret package. Validation set will be 10 % percent of the total records in adult.csv

```
set.seed(1)
test_index <- createDataPartition(y = data$income, times = 1, p = 0.1, list = FALSE)
trainset <- data[-test_index,]
temp <- data[test_index,]
validation <- temp %>%
     semi_join(trainset, by = "workclass") %>%
     semi_join(trainset, by = "education") %>%
     semi_join(trainset, by = "marital.status") %>%
     semi_join(trainset, by = "occupation") %>%
     semi_join(trainset, by = "relationship")%>%
     semi_join(trainset, by = "native.country")
accuracy_results<-data.frame()
```

Below is shown numbers of records in training and validation set

```
print(paste("Trainset has ",count(trainset)," records"))
```

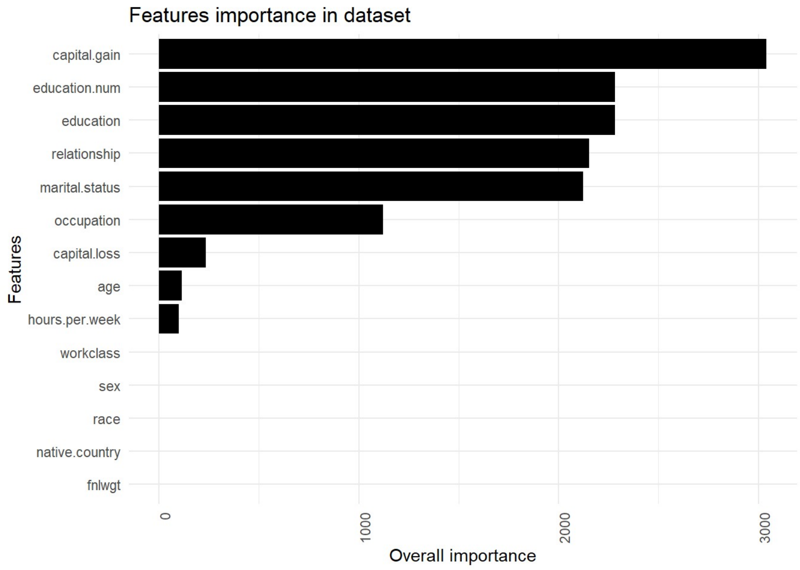
```
## [1] "Trainset has  29304  records"
```

```
print(paste("Validation set has ",count(validation)," records"))
```

```
## [1] "Validation set has  3257  records"
```

Check the importance of the variables is shown below in the chart. Bigger values means more importance

```
model1<-rpart(income~.,data = trainset)
var1<-varImp(model1) # Calculation of iimportance
var1<-rownames_to_column(var1) # Transfer of the row names to the separate column in data frame
varImpotance<-var1[order(-var1$Overall),] # Reorder of the importances by value
ggplot(varImpotance,aes(x = reorder(rowname, Overall),y=Overall),fill = variable)+ #chart creation
  geom_bar(stat = "identity",fill="black")+
  xlab("Features")+ylab("Overall importance")+ggtitle("Features importance in dataset")+
  theme_minimal()+ theme(axis.text.x = element_text(angle = 90, hjust = 1))+coord_flip()
```



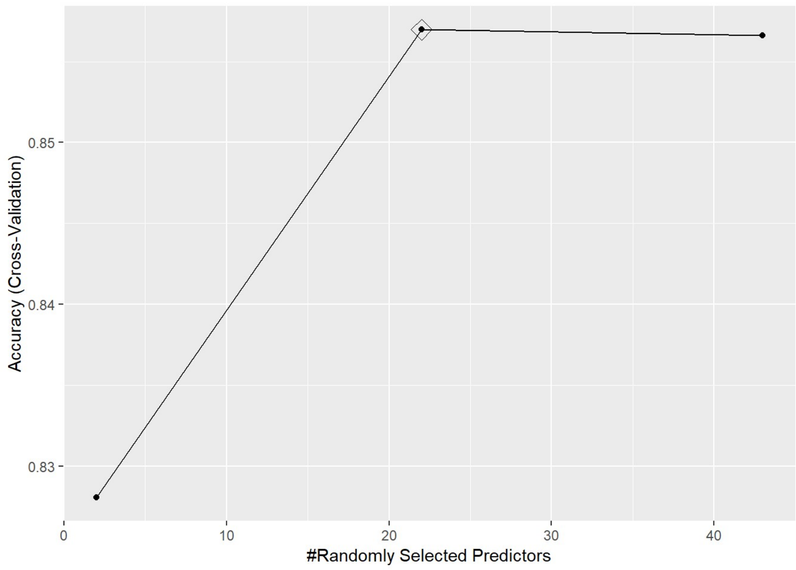
Features workclass,sex, race, native.country,hours.per.week,age and fnlwgt has no significant importance to the prediction in and these features will be excluded from the analysis and save new feature optimised dataset as “trainset_opt”

```
trainset_opt<-trainset%>%select(-workclass,-sex, -race, -native.country,-fnlwgt,-hours.per.week,-age)
as_tibble(trainset_opt)%>%head(n=10)%>%kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

education	education.num	marital.status	occupation	relationship	capital.gain	capital.loss	income
HS-grad	9	Widowed	?	Not-in-family	0	4356	<=50K
HS-grad	9	Widowed	Exec-managerial	Not-in-family	0	4356	<=50K
Some-college	10	Widowed	?	Unmarried	0	4356	<=50K
7th-8th	4	Divorced	Machine-op-inspct	Unmarried	0	3900	<=50K
Some-college	10	Separated	Prof-specialty	Own-child	0	3900	<=50K
HS-grad	9	Divorced	Other-service	Unmarried	0	3770	<=50K
10th	6	Separated	Adm-clerical	Unmarried	0	3770	<=50K
Doctorate	16	Never-married	Prof-specialty	Other-relative	0	3683	>50K
HS-grad	9	Divorced	Prof-specialty	Not-in-family	0	3683	<=50K
Some-college	10	Never-married	Craft-repair	Unmarried	0	3004	>50K

For all training model was used the same method of cross validation with 5 folds with parallel processing on 7 cores of desktop Below first try is to use random forest algoriithm with parallel processing with parameter selection Duration of theh process with 7 parallel cores is ca 10 minutes

```
tsCVstart<-Sys.time()
#print(paste("Process started",tsCVstart))
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)  # registration of the clusters for parallel processing
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE) # adjusting control for allowing parrallel processing
set.seed(1)
model_rf<-train(income~., method="rf",data = trainset_opt,trControl = fitControl) # creation of the model for al
l attributes, take a lot of time
#model_rf # display of the model
ggplot(model_rf,highlight = T) # result of optimisation
```




```
prediction_rf<-predict(model_rf,validation)
#confusionMatrix(data=prediction_rf,reference=validation$income)
stopCluster(cluster)
registerDoSEQ()
accuracy_rf<-confusionMatrix(data=prediction_rf,reference=validation$income)$overall[["Accuracy"]]

accuracy_results <- bind_rows(accuracy_results,
                          data_frame(method="rf ",
                                     Accuracy = accuracy_rf))
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
print(accuracy_results)
```

```
##   method  Accuracy
## 1    rf  0.8652134
```

```
tsCVfinish<-Sys.time()
#print(paste("Process finished ",tsCVfinish))
print(paste("Process duration ",trunc(unclass(tsCVfinish)-unclass(tsCVstart)),"sec"))
```

```
## [1] "Process duration  483 sec"
```

Second model for the training was selected xgbTree method from caret package. Method xgbTree is one of the methods of xgBoosting and is supposed for classification

```
getModelInfo()$xgbTree$type
```

```
## [1] "Regression"     "Classification"
```

Calculation will take approximately 2 minutes with 7 cores

```
tsCVstart<-Sys.time()
#print(paste("Process started",tsCVstart))
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)  # registration of the clusters for parallel processing
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE) # adjusting control for allowing parrallel processing
set.seed(1)
model_xgbTree<-train(income~., method="xgbTree",data = trainset_opt,trControl = fitControl) # creation of the mod
el for all attributes, take a lot of time
#model_xgbTree # display of the model
#ggplot(model_xgbTree,highlight = T)
prediction_xgbTree<-predict(model_xgbTree,validation)
#confusionMatrix(data=prediction_xgbTree,reference=validation$income)
stopCluster(cluster)
registerDoSEQ()
accuracy_xgbTree<-confusionMatrix(data=prediction_xgbTree,reference=validation$income)$overall[["Accuracy"]]

accuracy_results <- bind_rows(accuracy_results,
                          data_frame(method="xgbTree ",
                                     Accuracy = accuracy_xgbTree))
print(accuracy_results)
```

```
##     method  Accuracy
## 1      rf  0.8652134
## 2 xgbTree  0.8698189
```

```
tsCVfinish<-Sys.time()
#print(paste("Process finished ",tsCVfinish))
print(paste("Process duration ",trunc(unclass(tsCVfinish)-unclass(tsCVstart)),"sec"))
```

```
## [1] "Process duration  110 sec"
```

Third methon was taken very popular method of xgbDart Method xgbDART is XGBoost method and is supposed for classification

```
getModelInfo()$xgbDART$type
```

```
## [1] "Regression"     "Classification"
```

Calculation will take approximately 16 minutes with 7 cores

```
tsCVstart<-Sys.time()
#print(paste("Process started",tsCVstart))
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)  # registration of the clusters for parallel processing
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE) # adjusting control for allowing parrallel processing
set.seed(1)
model_xgbDART<-train(income~., method="xgbDART",data = trainset_opt,trControl = fitControl) # creation of the mod
el for all attributes, take a lot of time
#model_xgbDART # display of the model
#ggplot(model_xgbDART,highlight = T)
prediction_xgbDART<-predict(model_xgbDART,validation)
#confusionMatrix(data=prediction_xgbDART,reference=validation$income)
stopCluster(cluster)
registerDoSEQ()
accuracy_xgbDART<-confusionMatrix(data=prediction_xgbDART,reference=validation$income)$overall[["Accuracy"]]

accuracy_results <- bind_rows(accuracy_results,
                          data_frame(method="xgbDART ",
                                     Accuracy = accuracy_xgbDART))
print(accuracy_results)
```

```
##     method  Accuracy
## 1      rf  0.8652134
## 2 xgbTree  0.8698189
## 3 xgbDART  0.8725821
```

```
tsCVfinish<-Sys.time()
#print(paste("Process finished ",tsCVfinish))
print(paste("Process duration ",trunc(unclass(tsCVfinish)-unclass(tsCVstart)),"sec"))
```

```
## [1] "Process duration  825 sec"
```

Training model with logistic regression (glm method in cated package) Method glm is logistic regression model and is supposed for classification

```
getModelInfo()$glm$type
```

```
## [1] "Regression"     "Classification"
```

Calculation will take approximately 0.5 minutes with 7 cores

```
tsCVstart<-Sys.time()
#print(paste("Process started",tsCVstart))
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)  # registration of the clusters for parallel processing
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE) # adjusting control for allowing parrallel processing
set.seed(1)
model_glm<-train(income~., method="glm",data = trainset_opt,trControl = fitControl) # creation of the model for a
ll attributes, take a lot of time
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#model_glm # display of the model
#ggplot(model_glm,highlight = T)
prediction_glm<-predict(model_glm,validation)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading
```

```
#confusionMatrix(data=prediction_glm,reference=validation$income)
stopCluster(cluster)
registerDoSEQ()
accuracy_glm<-confusionMatrix(data=prediction_glm,reference=validation$income)$overall[["Accuracy"]]

accuracy_results <- bind_rows(accuracy_results,
                          data_frame(method="glm",
                                     Accuracy = accuracy_glm))
print(accuracy_results)
```

```
##     method  Accuracy
## 1       rf  0.8652134
## 2 xgbTree  0.8698189
## 3 xgbDART  0.8725821
## 4      glm 0.8507829
```

```
tsCVfinish<-Sys.time()
#print(paste("Process finished ",tsCVfinish))
print(paste("Process duration ",trunc(unclass(tsCVfinish)-unclass(tsCVstart)),"sec"))
```

```
## [1] "Process duration  25 sec"
```

Training model with Generalized Boosted Regression Modeling Method gbm is Generalized Boosted Regression Modeling (GBM) and is supposed for classification

```
getModelInfo()$gbm$type
```

```
## [1] "Regression"     "Classification"
```

Calculation will take approximately 1 minutes with 7 cores

```
tsCVstart<-Sys.time()
#print(paste("Process started",tsCVstart))
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)  # registration of the clusters for parallel processing
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE) # adjusting control for allowing parrallel processing
set.seed(1)
model_gbm<-train(income~., method="gbm",data = trainset_opt,trControl = fitControl) # creation of the model for a
ll attributes, take a lot of time
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0433             nan     0.1000    0.0303
##      2        0.9959             nan     0.1000    0.0234
##      3        0.9568             nan     0.1000    0.0189
##      4        0.9248             nan     0.1000    0.0162
##      5        0.8974             nan     0.1000    0.0135
##      6        0.8759             nan     0.1000    0.0105
##      7        0.8534             nan     0.1000    0.0114
##      8        0.8341             nan     0.1000    0.0099
##      9        0.8193             nan     0.1000    0.0073
##     10        0.8090             nan     0.1000    0.0050
##     20        0.7266             nan     0.1000    0.0029
##     40        0.6708             nan     0.1000    0.0008
##     60        0.6490             nan     0.1000    0.0004
##     80        0.6383             nan     0.1000    0.0002
##    100        0.6305             nan     0.1000    0.0000
##    120        0.6256             nan     0.1000   -0.0000
##    140        0.6208             nan     0.1000   -0.0000
##    150        0.6192             nan     0.1000   -0.0000
```

```
#model_gbm # display of the model
#ggplot(model_gbm,highlight = T)
prediction_gbm<-predict(model_gbm,validation)
#confusionMatrix(data=prediction_gbm,reference=validation$income)
stopCluster(cluster)
registerDoSEQ()
accuracy_gbm<-confusionMatrix(data=prediction_rf,reference=validation$income)$overall[["Accuracy"]]

accuracy_results <- bind_rows(accuracy_results,
                          data_frame(method="GBM ",
                                     Accuracy = accuracy_gbm))
print(accuracy_results)
```

```
##     method  Accuracy
## 1       rf  0.8652134
## 2 xgbTree   0.8698189
## 3 xgbDART   0.8725821
## 4      glm  0.8507829
## 5      GBM  0.8652134
```

```
tsCVfinish<-Sys.time()
#print(paste("Process finished ",tsCVfinish))
print(paste("Process duration ",trunc(unclass(tsCVfinish)-unclass(tsCVstart)),"sec"))
```

```
## [1] "Process duration  47 sec"
```

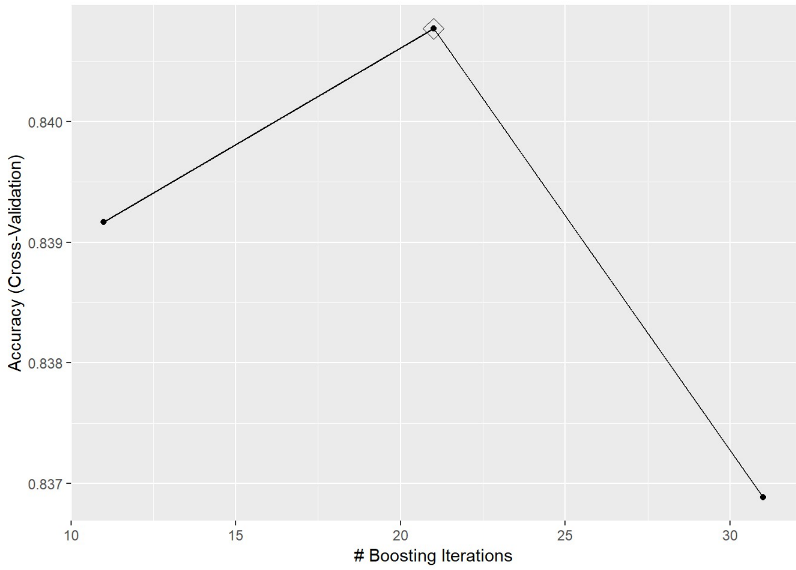
Training model with LogitBoost LogitBoost is Boostet Logistic regression models and is supposed for classification

```
getModelInfo()$LogitBoost$type
```

```
## [1] "Classification"
```

LogitBoost

```
tsCVstart<-Sys.time()
#print(paste("Process started",tsCVstart))
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)  # registration of the clusters for parallel processing
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE) # adjusting control for allowing parralel processing
set.seed(1)
model_LogitBoost<-train(income~., method="LogitBoost",data = trainset_opt,trControl = fitControl) # creation of t
he model for all attributes, take a lot of time
#model_LogitBoost # display of the model
ggplot(model_LogitBoost,highlight = T)
```




```
prediction_LogitBoost<-predict(model_LogitBoost,validation)
#confusionMatrix(data=prediction_LogitBoost,reference=validation$income)
stopCluster(cluster)
registerDoSEQ()
accuracy_LogitBoost<-confusionMatrix(data=prediction_rf,reference=validation$income)$overall[["Accuracy"]]

accuracy_results <- bind_rows(accuracy_results,
                          data_frame(method="LogitBoost ",
                                     Accuracy = accuracy_LogitBoost))

tsCVfinish<-Sys.time()
#print(paste("Process finished ",tsCVfinish))
print(paste("Process duration ",trunc(unclass(tsCVfinish)-unclass(tsCVstart)),"sec"))
```

```
## [1] "Process duration  26 sec"
```

Results section

After evaluation of the methods rf, xgbTree, xgbDART,glm,LogitBoost and GBM the best model was selected. The best model is xgbDart with the accuracy of 0.873. xgbDART is extreme Gradient boosting algorithm.alo All other used algorithm delivered also high accuracy which is comparable to xgbDart and lies in the range of 0.85-0.86

```
print(accuracy_results)%>%kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

```
##        method  Accuracy
## 1          rf  0.8652134
## 2     xgbTree  0.8698189
## 3     xgbDART  0.8725821
## 4         glm 0.8507829
## 5         GBM  0.8652134
## 6 LogitBoost  0.8652134
```

method	Accuracy
rf	0.8652134
xgbTree	0.8698189
xgbDART	0.8725821
glm	0.8507829
GBM	0.8652134
LogitBoost	0.8652134

Conclusion section.

We have used publically available dataset *Adult Census Income* to create the best prediction model. Data was separated into train and validation set. Features(Dimenstions) were analysed and seven of them were deleted due to the slow significance. Six models of machine learning were used for the analysis and after all analysis the best algorithm xgbDART was selected with the best accuracy of 0.8725821