HarvardX PH125.9x Capstone CYO Project

Daniel Cejudo-Bruno

12/10/2020

Introduction

This report was produced for the HarvardX PH125.9x Data Science course. The Capstone "Choose Your Own project" is the second and last of the two projects that have to be submitted to pass the course. In this project, we can chose our own dataset to analyze and implement machine learning to predict an outocme.

Dataset

We chose a diabetes dataset taken from the hospital Frankfurt, Germany. The dataset was obtained from https://www.kaggle.com/c/diabetes/overview. The data is structurd as follows:

Column Name	Data Type	Description
Id	Integer	Unique Pation Identifier
Pregnancies	Integer	Number of Pregnancies
Glucose	Integer	Glucouse
BloodPressure	Integer	Blood Preasure
SkinThickness	Integer	Skin Thinkness
Insulin	Integer	Insulin
BMI	Decimal	BMI
DiabetesPedigreeFunction	Decimal	Diabetes Pedigree Function
Age	Integer	Age
Outcome	Integer	Outcome

The **Outcome** field is what we want to predict. We will use the rest of the fields, except for the column **Id**, as predictors.

Objective

The objective of this project is to learn which machine learning algorithm can achieve the best predictive accuracy. The machine learning algorithms that we will consider are: Classification and Regression Trees, Random Forest, Gradient Boosting Machine, k-Nearest Neighbors, Generalized Linear Model, Support Vector Machine with Radial Basis Function Kernel, and eXtreme Gradient Boosting. We set a target goal of **95% accuracy**.

Train & Test dataset breakdown

The dataset is given in two different files with identical structure. The first file, named **train.csv**, contains the data that will be used to train the models. The second file, named **test.csv**, will be used to score the models and obtain accuracy metrics.

Load Libraries

We automate the installation of the necessary libraries for convinience and resuability.

```
list.of.packages <- c("DBI", "dplyr", "tidyverse","Hmisc","odbc","reshape2","gridExtra","ggplot2","plotl

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]

if(length(new.packages)) install.packages(new.packages)

library(DBI)
library(dplyr)
library(tidyverse)
library(Hmisc)
library(reshape2)
library(gridExtra)
library(ggplot2)
library(plotly)
library(ggcorrplot)
library(GGally)
library(forecast)
library(fpp3)
library(lessR)
library(furrr)
library(feasts)
library(tinytex)
library(knitr)
library(caret)
library(Rborist)
library(glmnet)
library(kableExtra)
library(caretEnsemble)
library(PerformanceAnalytics)
library(mlbench)
library (nnet)
library(gbm)
```

Methods & Analysis

We conduct exploratory data analysis. We seek to visualize the data to understand patterns, and distribution. The insights we gain from this analysis will inform our decisions to create a machine learning model that can be effective at predicting ratings for new reviews.

Exploratory Data Analysis

Read the dataset

We read the dataset which are in **csv** format. The files are located in the dataset subfolder.

```
data_train <- read.csv("dataset/train.csv")
data_test <- read.csv("dataset/test.csv")
```

Dataset Schema

We display the schema of the training data to ensure it matches our expectations.

```
str(data_train)
```

```
## 'data.frame':    1405 obs. of  10 variables:
##  $ Pregnancies             : int  2 0 0 4 8 2 2 4 3 6 ...
##  $ Glucose                 : int  138 135 173 99 194 83 89 99 80 166 ...
##  $ BloodPressure           : int  62 68 78 72 80 65 90 68 0 74 ...
##  $ SkinThickness           : int  35 42 32 17 0 28 30 38 0 0 ...
##  $ Insulin                 : int  0 250 265 0 0 66 0 0 0 0 ...
##  $ BMI                     : num  33.6 42.3 46.5 25.6 26.1 36.8 33.5 32.8 0 26.6 ...
##  $ DiabetesPedigreeFunction: num  0.127 0.365 1.159 0.294 0.551 ...
##  $ Age                     : int  47 24 58 28 67 24 42 33 22 66 ...
##  $ Outcome                 : int  1 1 0 0 0 0 0 0 0 0 ...
##  $ Id                      : int  0 3 5 6 7 8 9 10 12 13 ...
```

Data Quality Checks

We need to understand if there are any missing values in the dataset. We find that there are no missing values.

```
sapply(data_train, function(x) sum(is.na(x))) %>%
  melt() %>%
  kable() %>%
  kable_classic(full_width = F, html_font = "Calibri")
```

	value
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0
Id	0

Summary Statistics

We begin to explore the dataset to understand the distribution of the data, patterns and trends.

```
summary(data_train)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.668   Mean   :121.2   Mean   : 69.59   Mean   :20.73
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:33.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.00   Min.   : 0.00   Min.   :0.0780           Min.   :21.0
##  1st Qu.:  0.00   1st Qu.:27.40   1st Qu.:0.2450           1st Qu.:24.0
##  Median : 29.00   Median :32.30   Median :0.3760           Median :29.0
##  Mean   : 79.93   Mean   :32.18   Mean   :0.4777           Mean   :33.3
##  3rd Qu.:130.00   3rd Qu.:36.60   3rd Qu.:0.6310           3rd Qu.:41.0
##  Max.   :744.00   Max.   :80.60   Max.   :2.4200           Max.   :81.0
##     Outcome             Id
##  Min.   :0.0000   Min.   :   0.0
##  1st Qu.:0.0000   1st Qu.: 496.0
##  Median :0.0000   Median :1001.0
##  Mean   :0.3466   Mean   : 997.7
##  3rd Qu.:1.0000   3rd Qu.:1499.0
##  Max.   :1.0000   Max.   :1999.0
```

Row Counts

Next we want to understand the distinct counts for each column. We find that there are **1405** observations. The Id field is indeed unique to each observation. There are only two outcomes 1 and 0 correspoting to **postive** diagnosis and **negative** diagnosis. We learn that categorical machine learning models would be appropriate for this type of dataset. We understand that the **Outcome** column will need to be converted to **factor** to make it compatible with the machine learning models.

```
data_train %>% summarise(n_patients = n_distinct(Id),
                         n_pregnancies = n_distinct(Pregnancies),
                         n_blood_preassure = n_distinct(BloodPressure),
                         n_skin_thicknes = n_distinct(SkinThickness),
                         n_insulin = n_distinct(Insulin),
                         n_bmi = n_distinct(BMI),
                         n_diabetes_pedigree_function = n_distinct(DiabetesPedigreeFunction),
                         n_age = n_distinct(Age),
                         n_outcome = n_distinct(Outcome)) %>%
  melt() %>%
  kable() %>%
  kable_classic(full_width = F, html_font = "Calibri")
```

variable	value
n_patients	1405
n_pregnancies	17
n_blood_preassure	47
n_skin_thicknes	51
n_insulin	175
n_bmi	239
n_diabetes_pedigree_function	486
n_age	52
n_outcome	2

Next we seek to expand our understanding by breaking down distinct value counts by the two possible outcomes. We find that in general the means of the values are higher when the patient is diagnosed with diabetes.

```
data_train %>% group_by(Outcome) %>%
               summarise(n_patients = n_distinct(Id),
                         n_pregnancies = mean(Pregnancies),
                         n_bp = mean(BloodPressure),
                         n_skin_thicknes = mean(SkinThickness),
                         n_insulin = mean(Insulin),
                         n_bmi = mean(BMI),
                         n_diabetes_pedigree = mean(DiabetesPedigreeFunction),
                         n_age = mean(Age)) %>%
  kable()  %>%
  kable_classic(full_width = F, html_font = "Calibri")
```

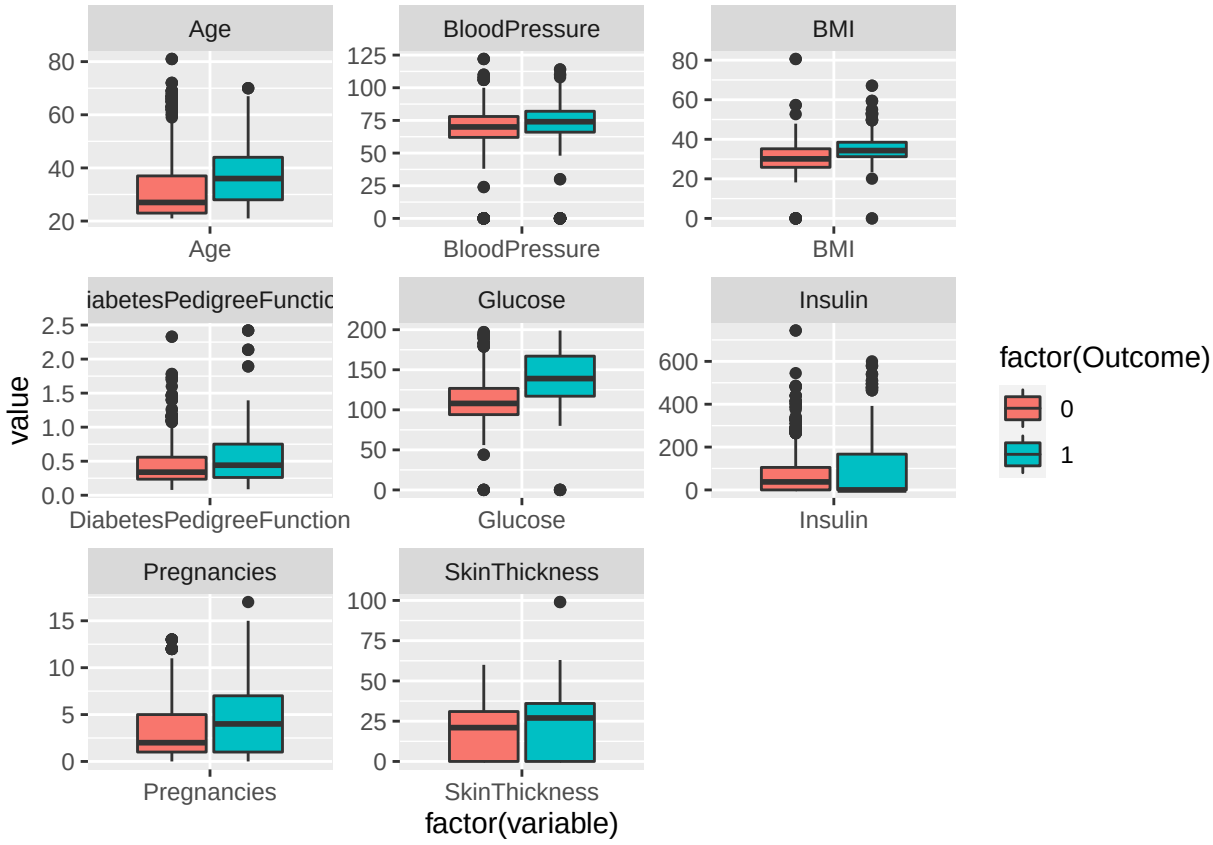
Outcome	n_patients	n_pregnancies	n_bp	n_skin_thicknes	n_insulin	n_bmi	n_diabetes_pedigree	
0	918	3.127451	68.59913	19.64924	69.53813	30.48965	0.4331307	3
1	487	4.685832	71.44559	22.75565	99.51540	35.37413	0.5618049	3

Boxplots

We seek to better understand the distribution of the data for each outcome. For this we need to knwo more than the means. We find that the medians are generally higher for all predictors when the patient is diagnozed with diabetes.

```
data_train %>% select(Pregnancies,
                      Glucose,
                      BloodPressure,
                      SkinThickness,
                      Insulin,
                      BMI,
                      DiabetesPedigreeFunction,
                      Age,
                      Outcome) %>%
  pivot_longer(., cols = c(Pregnancies,
                           Glucose,
                           BloodPressure,
                           SkinThickness,
                           Insulin,
```

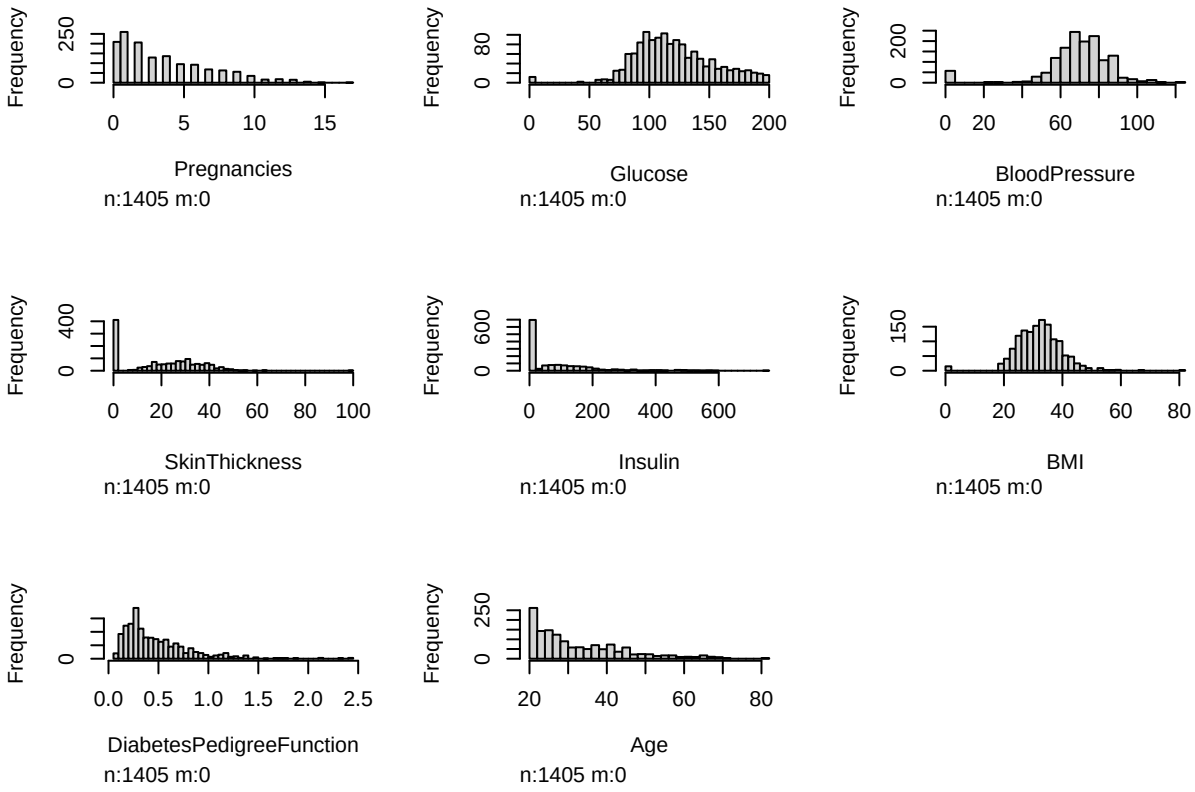
```
                          BMI,
                          DiabetesPedigreeFunction,
                          Age),
               names_to = "variable", values_to = "value") %>%
  ggplot(aes(x = factor(variable), y = value, fill = factor(Outcome))) +
  geom_boxplot() +
  facet_wrap(~variable, scale="free")
```



Historgrams

We want to understand the distribution of the predictor values. We find **Blood Pressure** and **BMI** are not significantly different from a normal distribution, although they don't match a normal distribution perfectly. The rest of the predictors are significantly different from a normal distribution.

```
data_train %>% select(-Id, -Outcome) %>%
               hist.data.frame()
```



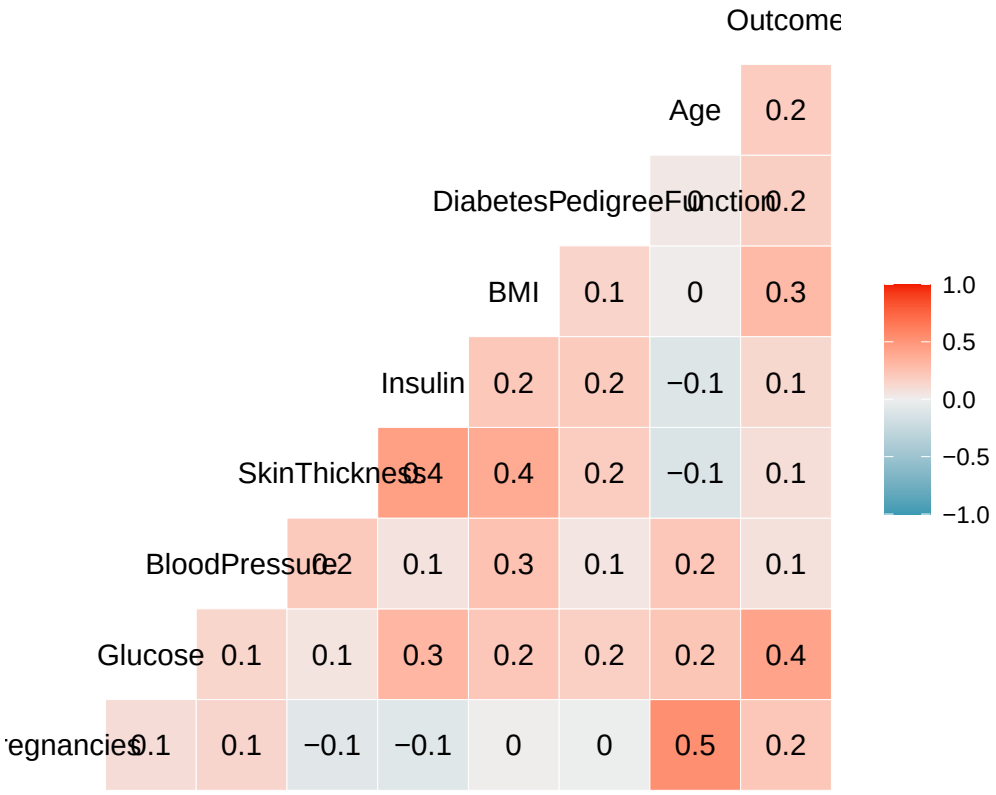
Correlation Analysis

We want to understand the predictive power of each field in the dataset. We also seek to understand if there is any correlation between the predictors themselves. The linear correlation between variables x and y is given by:

$$r = \frac{\sum(x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum(x_t - \bar{x})^2}\sqrt{\sum(y_t - \bar{y})^2}}.$$

We find no strong correlations between any one predictors and the **Outcome** field. Hence, linear models may not perform great compared to classification trees or random trees models. Although we find some correlation between some of the predictors, in no case the correlation is above 0.5. Therefore, it may be safe to use all predictors.

```
data_train %>% select(-Id) %>% ggcorr(label = TRUE)
```



Preprocessing

We remove the Id column since it is a unique identifier for each observation. We also transform the **Outcome** column to **factor** as this is required by the Caret package when using classification algorithms. We inspect the first 10 records of the transofmred trained dataset for quality assurance.

```
data_train_transformed <- data_train %>% select(-Id)
data_train_transformed$Outcome <- as.factor(ifelse(data_train$Outcome == 1,"Yes","No"))

# Isolate Outcome column into its own object for later use in measuring performance
test_outcome <- as.factor(ifelse(data_test$Outcome == 1,"Yes","No"))

# Remove Id and Outcome from test dataset that will be used to score the trained models.
data_test_transformed <- data_test %>% select(-Id, -Outcome, -split)

data_train_transformed %>% head() %>%
  knitr::kable(caption = "Preprocessing Output Sample")  %>%
  kable_classic(full_width = F, html_font = "Calibri")
```

Modeling

In this section we use the insights gain in our exploratory data analysis to create models trained by the training dataset. We will then use the test dataset to meassure the accuracy of our models.

Table 2: Preprocessing Output Sample

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
2	138	62	35	0	33.6	0.127	47	Yes
0	135	68	42	250	42.3	0.365	24	Yes
0	173	78	32	265	46.5	1.159	58	No
4	99	72	17	0	25.6	0.294	28	No
8	194	80	0	0	26.1	0.551	67	No
2	83	65	28	66	36.8	0.629	24	No

Cross Validation

We implement cross validation to reduce mitigate the issue of the model performing well due to chance and to avoid overfilling. Set set **K** to **10** as this is standard approach. We set seeds before we train each model to ensure evaluations are deterministic. We also set the seed values which we will use before we train each model to ensure that our results will be deterministic.

```
# Cross Validation using 10-fold cross validation
control <- trainControl(method = "cv",
                        number = 10,
                        savePredictions = "final",
                        allowParallel = TRUE,
                        classProbs = TRUE,
                        summaryFunction = twoClassSummary)

metric <- "Accuracy"

seed <- 7
```

Train Models

We proceed to build different classification models to understand which ones can be more effective based on our dataset. We also will conduct hyperparameter tuning. We identify tuning parameters for each model and we will perform a grid search to find which values optimize our models best. The following table describes the models we will build along with their tuning parameters. Lastly, we resample the performance of these models with training data and plot it to compare them to each other. We find that the model that performs best with training data is "Random Forest."

Classification Trees (rpart)

The rpart programs build classification models of a very general structure using a two stage procedure; the resulting models can be represented as binary trees.

Random Forest

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier.

Gradient Boosting Machine (GBM)

GBM was is introduced by Friedman in 2001. It is also known as MART (Multiple Additive Regression Trees) and GBRT (Gradient Boosted Regression Trees). GBM constructs a forward stage-wise additive model by implementing gradient descent in function space.

k-Nearest Neighbors (KNN)

KNN finds the distances between a target point and all the examples in the data, selects the specified number examples (K) closest to the target point, then votes for the most frequent label.

Support Vector Machine with Radial Basis Function Kernel

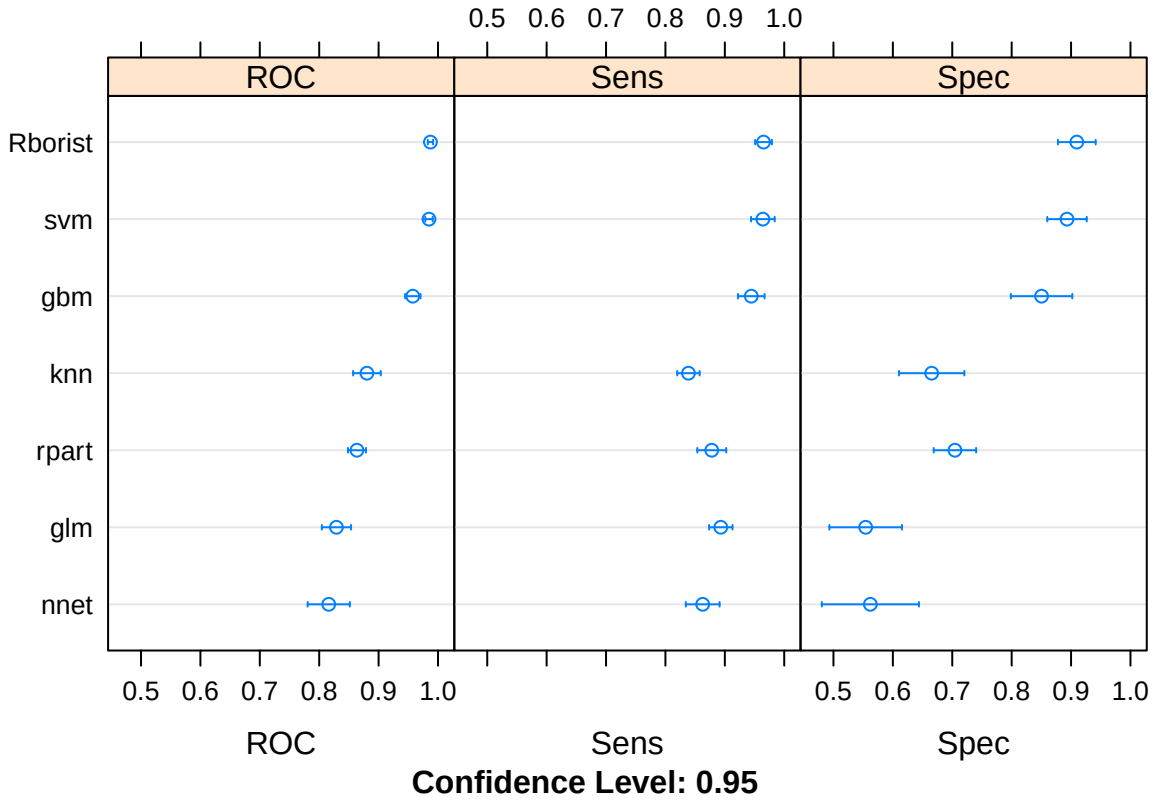
The algorithm creates a line or a hyperplane which separates the data into classes. Radial kernel support vector machine can be used when the data is not linearly separable. It generates non-linear decision boundaries by transforming the features into a higher dimensional space.

Neural Networks (nnet)

Algorithm based on feed-forward neural networks with a single hidden layer, and for multinomial log-linear models.

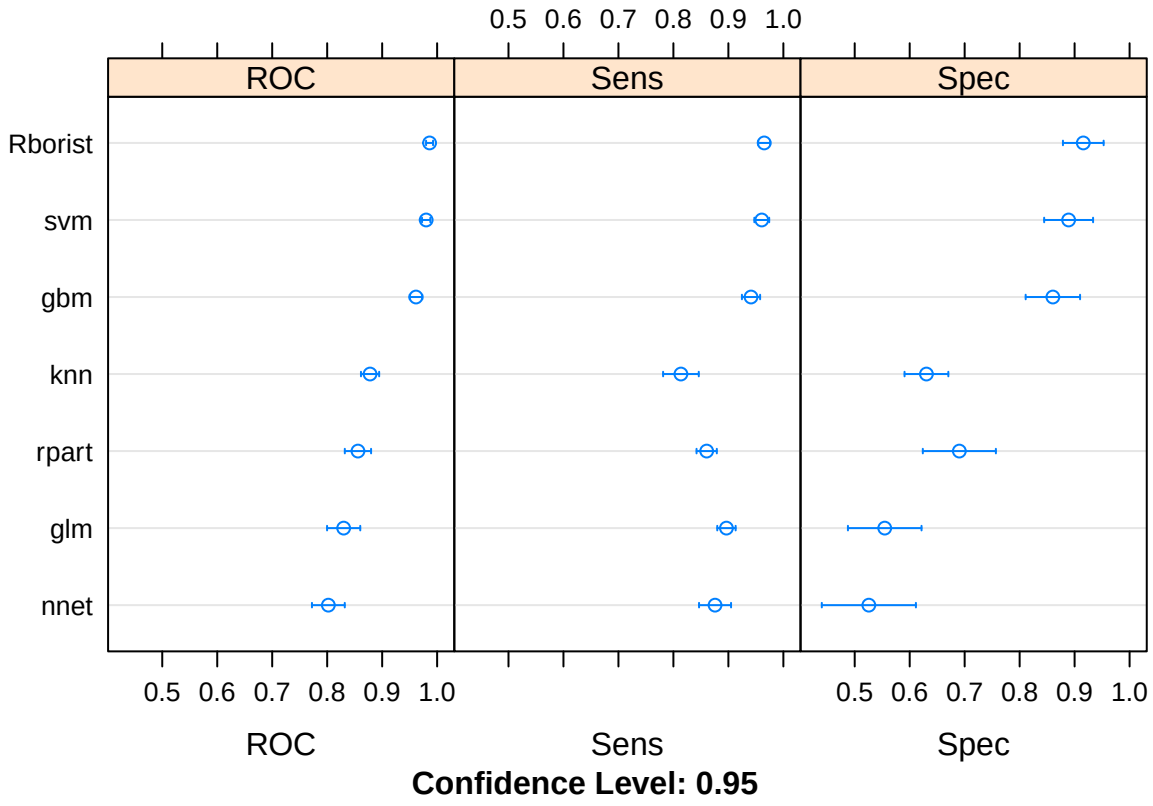
Models Summary:

Model	Method Value	Tuning Parameters
Classification and Regression Trees	rpart	cp
Random Forest	Rborist	predFixed, minNode
Gradient Boosting Machine	gbm	interaction.depth, n.trees, shrinkage, n.minobsinnode
k-Nearest Neighbors	nnet	K
Generalized Linear Model	glmnet	Matrix, alpha, lambda
Support Vector Machine with Radial Basis Function Kernel	svmRadial	sigma, C
Neural Networks	nnet	size, decay



Ensamble Models

We want to understand if we can improve the accuracy by combining the previous models into an ensemble of models which can perform better than an individual model. **caretEnsemble** is a package for making ensembles of caret models. In the **Caret** package, the function caretList is the preferred way to construct list of caret models, because it will ensure the resampling indexes are identical across all models. Lastly, we resample the performance of this model list with training data and plot it.



Results

We use the test data to score our trained models and obtained accuracy metrics.

```
predict_glm <- predict(model_glm, data_test_transformed)
accuracy_glm <- confusionMatrix(predict_glm, test_outcome)$overall["Accuracy"]

predict_rpart <- predict(model_rpart, data_test_transformed)
accuracy_rpart <- confusionMatrix(predict_rpart, test_outcome)$overall["Accuracy"]

predict_gbm <- predict(model_gbm, data_test_transformed)
accuracy_gbm <- confusionMatrix(predict_gbm, test_outcome)$overall["Accuracy"]

predict_nnet <- predict(model_nnet, data_test_transformed)
accuracy_nnet <- confusionMatrix(predict_nnet, test_outcome)$overall["Accuracy"]
```

Table 4: Results Table

Model	Accuracy
NNET	0.7378151
K-Nearest Neighbor	0.7932773
Generalized Linear Model	0.7949580
Classification Trees	0.8470588
Gradient Boosting Machine	0.9176471
Support Vector Machine	0.9596639
Random Forest	0.9647059
Models Ensemble	0.9680672

```
predict_svm <- predict(model_svm, data_test_transformed)
accuracy_svm <- confusionMatrix(predict_svm, test_outcome)$overall["Accuracy"]

predict_knn <- predict(model_knn, data_test_transformed)
accuracy_knn <- confusionMatrix(predict_knn, test_outcome)$overall["Accuracy"]

predict_rf <- predict(model_rf, data_test_transformed)
accuracy_rf <- confusionMatrix(predict_rf, test_outcome)$overall["Accuracy"]

predict_ensemble <- predict(model_ensemble, data_test_transformed)
accuracy_ensemble <- confusionMatrix(predict_ensemble, test_outcome)$overall["Accuracy"]

results <- data_frame(Model = "NNET", Accuracy = accuracy_nnet)
results <- rbind(results, data_frame(Model = "K-Nearest Neighbor", Accuracy = accuracy_knn))
results <- rbind(results, data_frame(Model = "Generalized Linear Model", Accuracy = accuracy_glm))
results <- rbind(results, data_frame(Model = "Classification Trees", Accuracy = accuracy_rpart))
results <- rbind(results, data_frame(Model = "Gradient Boosting Machine", Accuracy = accuracy_gbm))
results <- rbind(results, data_frame(Model = "Support Vector Machine", Accuracy = accuracy_svm))
results <- rbind(results, data_frame(Model = "Random Forest", Accuracy = accuracy_rf))
results <- rbind(results, data_frame(Model = "Models Ensemble", Accuracy = accuracy_ensemble))

results %>% knitr::kable(caption = "Results Table")  %>%
kable_classic(full_width = F, html_font = "Calibri")
```

Conclussion

We have analyzed the diabetes dataset and applied different machine learning algorithms to predict the **Outcome** variable. We used the machine learning algorithms covered in the Edex course plus we introduced other algorithms that are popular for classification scenarios. We used a training set to build the model and separate test dataset to measure the accuracy of the models. We use the metric **Accuracy** to measure the model performance.

We showed that while all the machine learning algorithms showed some predictive power, some proved better than others. The models Support Vector Machine with Radial Basis Function Kernel, Random Forest and our ensemble model scored an accuracy higher than **0.9500** which was our original goal. We found that our ensemble model archived the highest accuracy of all our models at **0.9680672%**.

A next step may be to run an **A/B test** where **Random Forest** and the **Ensemble model** are compared based on production data that is randomized.