1 Introduction
2 EDA
3 Modeling
4 Results
5 Conclusion
6 Appendix

Statistical Learning Project - Predictive Maintenance

Abdurhman Fayad - 1000040486

2023-01-22

1 Introduction

Using the "Predictive maintenance" dataset, which consists of 10,000 observations recorded as rows with eight features in columns, the report's goal is to create a classification model. The classifier must be able to predict the target variable, which has a value of '0' if there is no failure and no maintenance is thus required, and '1' if, on the other hand, some sort of damage has been discovered in the machine and maintenance is required.

The dataset can be found here: https://www.kaggle.com/shivamb/machine-predictive-maintenance-classification (https://www.kaggle.com/shivamb/machine-predictive-maintenance-classification).

The data is split in three parts: - Training set: 60% - Validation set: 20% - Testing set: 20%

The training and validation set will be used during the training, while the testing set will stay untouched until the model is ready for testing again real world data.

In this report,the following three models of classification will be used and compared at the end to find the best model:

- Logistic Regression
- Random Forest
- Neural Network

The following list shows us the details of our feautres:

- **UDI** *int*: Unique identifier ranging from 1 to 10000.
- **Product.ID** *int*: Unique identifier consisting in an alphanumeric string.
- **Type** *chr*: Consisting of a letter L, M, or H for low (50% of all products), medium (30%), and high (20%) as product quality variants and a variant-specific serial number.
- **Ait.temperature** *num*: Generated using a random walk process later normalized to a standard deviation of 2 K around 300 K.
- **Process.temperature** *num*: Generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K.
- **Rotational.speed** *int*: Calculated from power of 2860 W, overlaid with a nor- mally distributed noise.
- **Torque** *num*: Torque values are normally distributed around 40 Nm with an If = 10 Nm and no negative values.
- **Tool.wear** *int*: Torque values are normally distributed around 40 Nm with an If = 10 Nm and no negative values.
- **Target** *int*: The Target variable displays if the machine is failed or not.

1.1 Business Question

The following business questions will guide our analysis and will be our goal to answer:

- Which is the best approach to predict our targer variable ?
- Which variables have the greatest affect in predicting the target variable? and which have the least?
- How accurately can we estimate a possible failure?

Before moving on to model construction, it is necessary to analyze the dataset; in particular, it is important to analyze the distribution of individual predictor variables and the possible relationships present among them. To do this, an Exploratory Data Analysis (EDA) will now be carried out.

2 EDA

Exploratory data analysis (EDA) is used to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. All the variables in the training set are then analysed, first individually and then all together.

2.1 Univariate Anlaysis

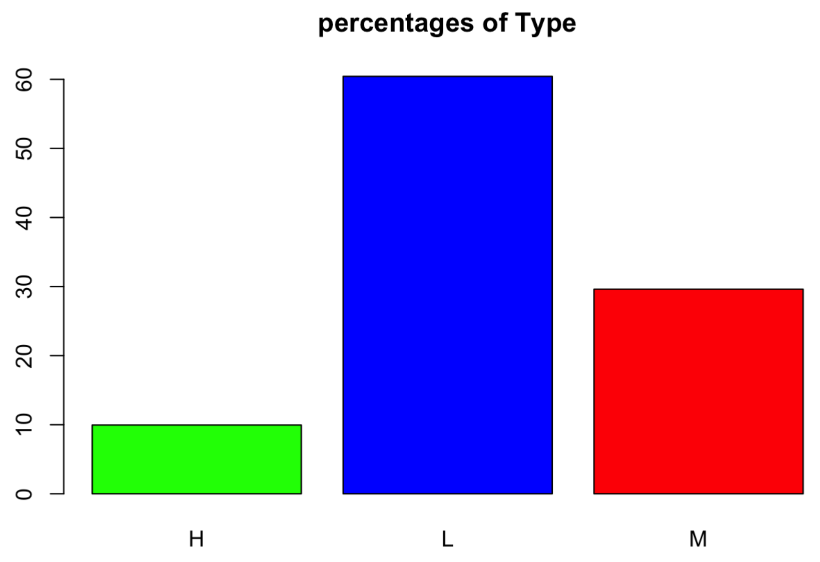
This is simplest form of data analysis, where the data being analyzed consists of just one variable. The main purpose of univariate analysis is to summarize the main features of each variable.

Qualitative Variables

Let us start analyzing the qualitative variables in the training set. Let us begin by saying that the variables UDI and Product.ID have unique values for each observation. It therefore makes little sense to perform a univariate analysis of these.

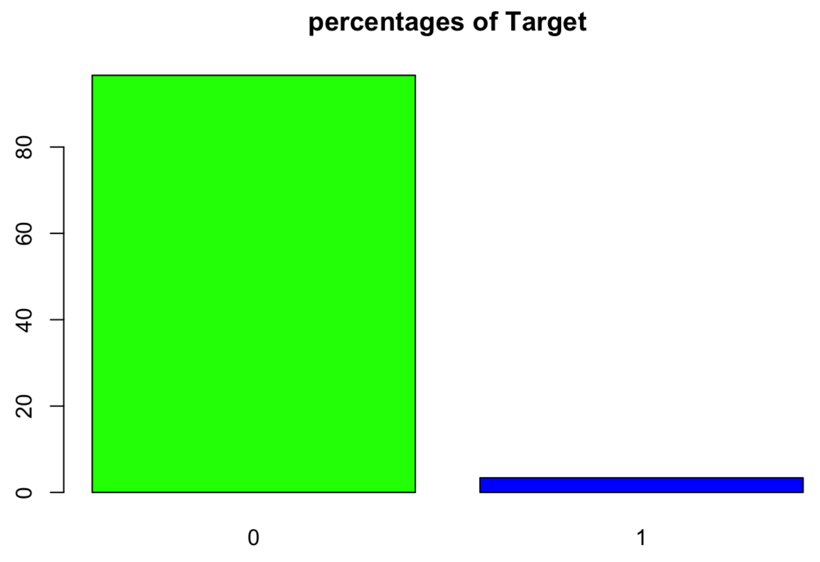
2.1.1 Type

Below we can see the distribution of the qualitative variable Type. We note that most of the observations in the training set are of type 'L'. The remainder (about 40%) is three quarters from 'M' type observations and only one quarter from 'H' type observations.



2.1.2 Target

Below we can see a bar chart of our target variable

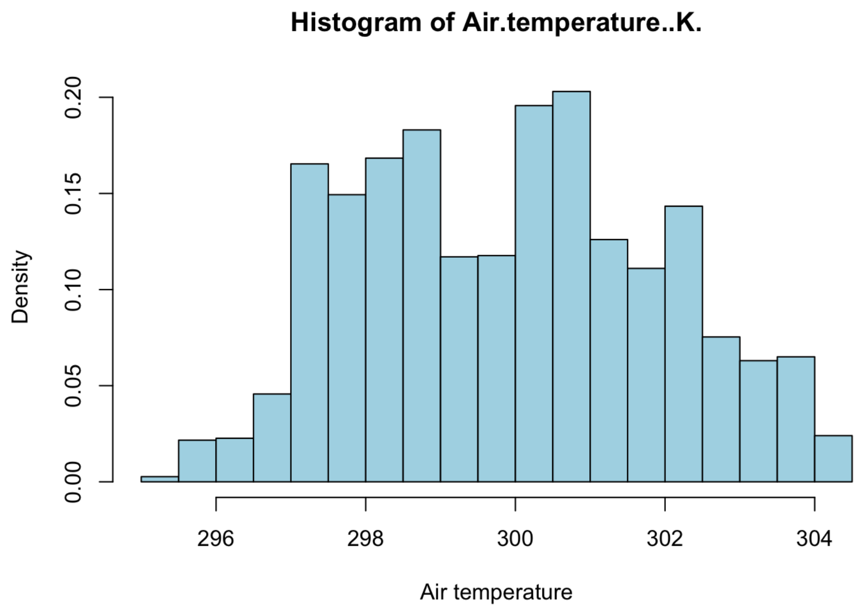


From the bar chat, we see that observations with a Target value of 0 are far greater than those with a Target value of 1 (approximately by a factor of 1:20). This means that our dataset is highly unbalanced.

Quantative Variables

Here we will be analyzing the numeric variables.

2.1.3 Air Temprature



Summary of Air Temperature

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   295.4   298.3   300.1   300.0   301.5   304.5
```

skewness and kurtosis of Air Temperature

```
## [1] "skewness"
```

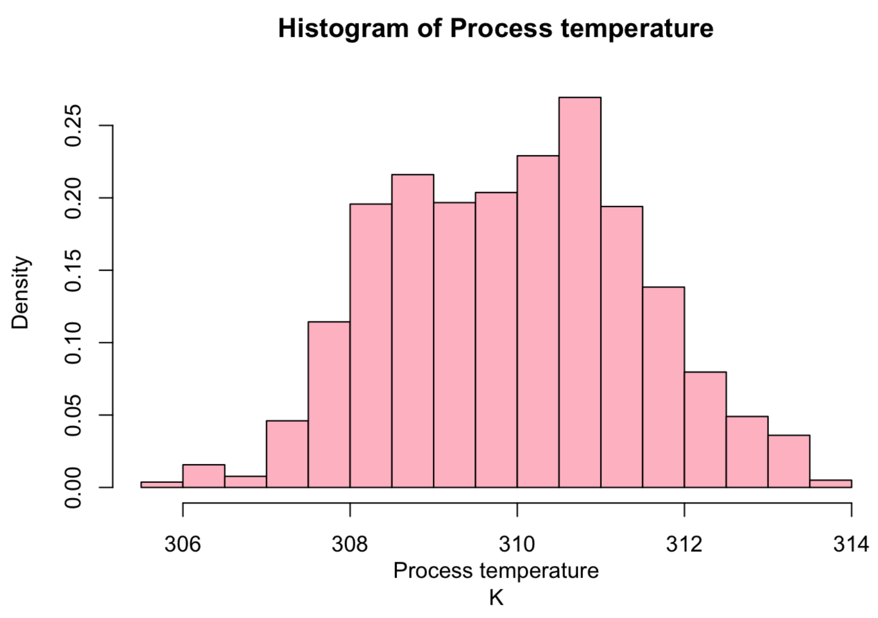
```
## [1] 0.1219379
```

```
## [1] "kurtosis"
```

```
## [1] -0.8442163
```

From the graph we can notice the presence of two peaks, yielding us thinking that there may be two subpopulation that influence the result of whole histogram. From the fact that the skewness value is very close to zero, we can deduce and find that the distribution of the data is almost symmetrical. In contrast, a kurtosis value of less than 3 indicates to us that the distribution is heavy-tailed, and the peak can be flatter.

2.1.4 Process Temprature



Summary of Process Temprature

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   305.8   308.8   310.1   310.0   311.1   313.8
```

skewness and kurtosis of Process Temprature

```
## [1] "skewness"
```

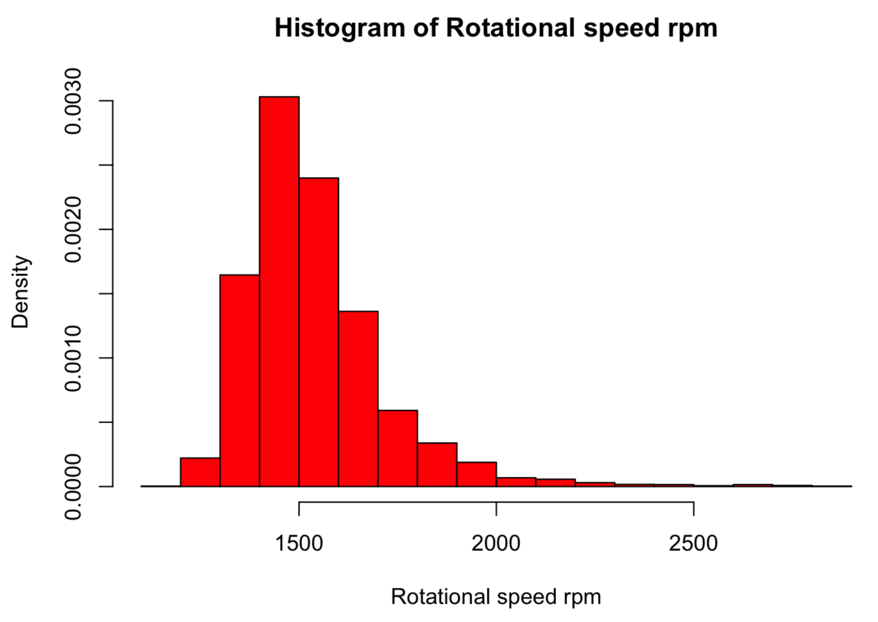
```
## [1] 0.02548641
```

```
## [1] "kurtosis"
```

```
## [1] -0.5098995
```

It can be seen that the distribution of the data is almost symmetrical, which is also confirmed by the low skewness value. The distribution also has a negative kurtosis value, so it has a heavy tail and the peak may be flatter.

2.1.5 Rotational Speed



Summary of Rotational speed rpm

File failed to load: file:///Users/afayad/Library/CloudStorage/OneDrive-Universita%CC%80degliStudidiCatania/Done/Stat%20learning/Fayad%20Project/Statistical%20Learning%20Project%20-%20Pre

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1181    1423    1504    1539    1611    2874
```

skewness and kurtosis of Rotational speed rpm

```
## [1] "skewness"
```

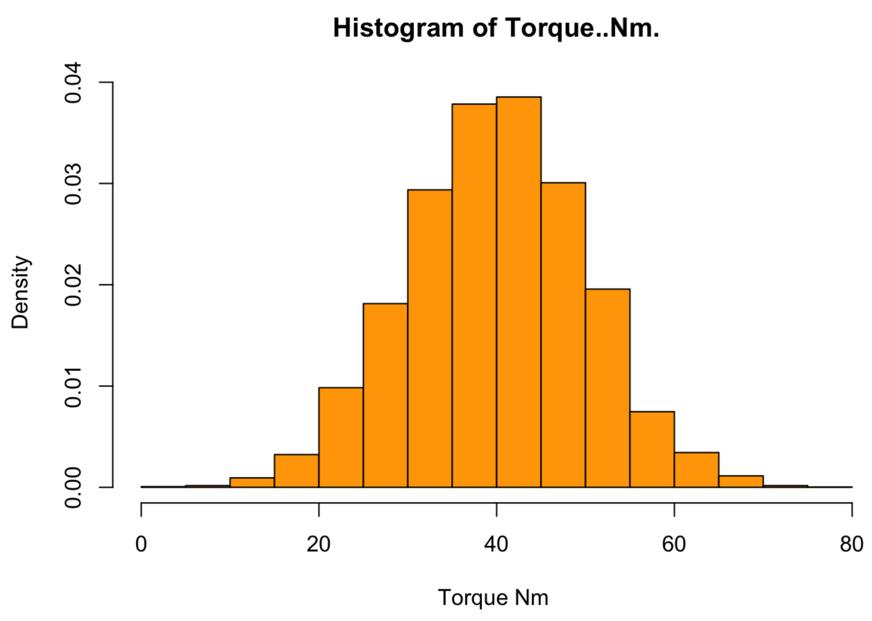
```
## [1] 1.993557
```

```
## [1] "kurtosis"
```

```
## [1] 7.405396
```

Unlike the previous ones, the data distribution of the variable Rotational.speed has a single peak, with a higher skewness value, which can be seen from the fact that a large part of the data lies to the left. The kurtosis value is also very different from the previous cases, in fact it has a very high value and above all is greater than 3. This indicates that the distribution is light-tailed and the top curve steeper, like pulling up the distribution (Leptokurtic).

2.1.6 Torque



Summary of Torque

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.20   33.20   40.10   40.01   46.80   76.20
```

skewness and kurtosis of Torque

```
## [1] "skewness"
```

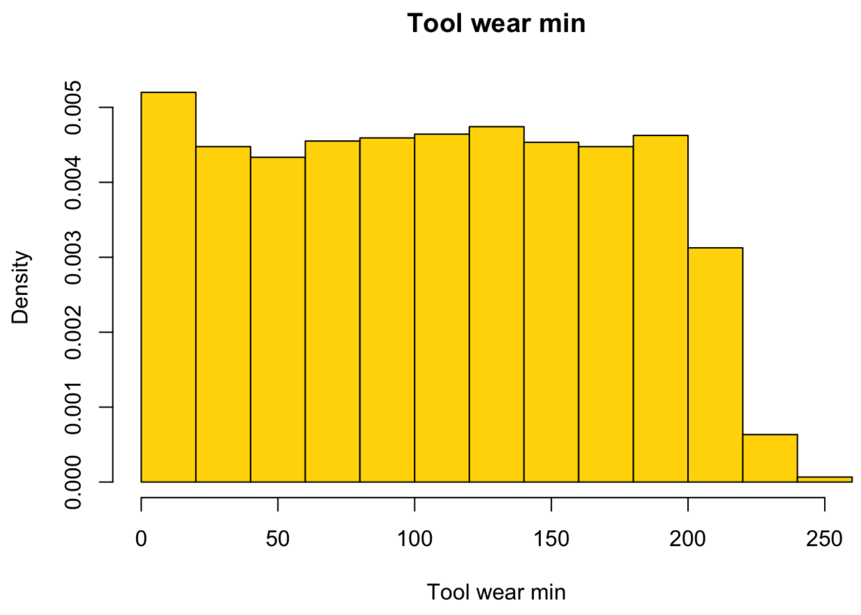
```
## [1] -0.003551766
```

```
## [1] "kurtosis"
```

```
## [1] -0.03281282
```

The distribution of the data for the variable Torque shows a skewness value very close to zero; the symmetry of the data distribution can therefore be seen. Like- wise, the kurtosis value is also very close to zero, but still less than 3. Therefore, this data distribution also has a heavy tail and the peak may be flatter.

2.1.7 Tool Wear



Summary of Tool Wear

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0    53.0   109.0   108.2   162.0   253.0
```

skewness and kurtosis of Tool Wear

```
## [1] "skewness"
```

```
## [1] 0.01914566
```

```
## [1] "kurtosis"
```

```
## [1] -1.152117
```

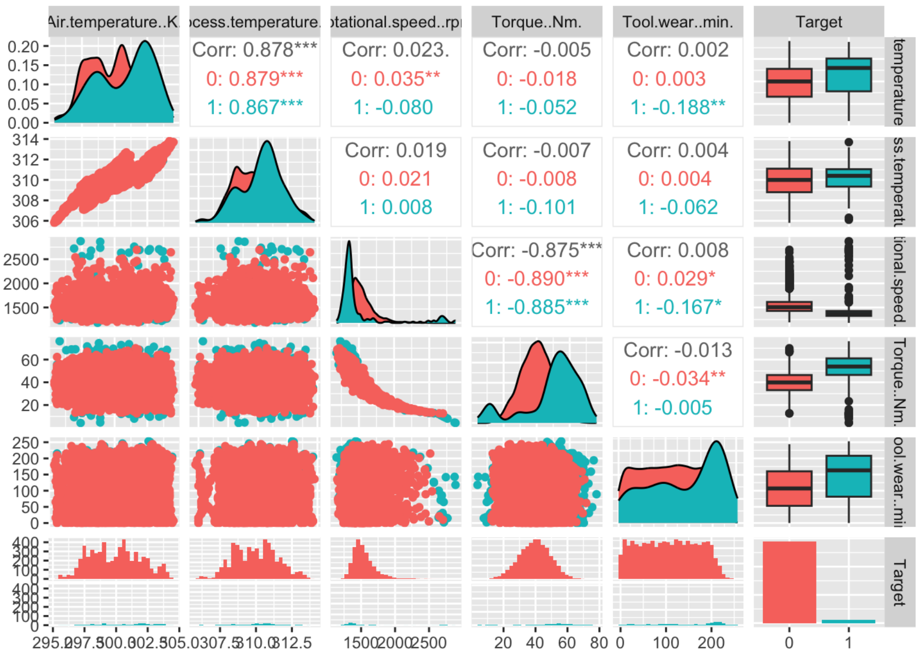
From the graph, we see that the distribution is almost symmetrical, which is also confirmed by the very close to zero skewness value. In addition, a negative kurtosis value also appears; this is due to the presence of a flatter peak and very heavy tail.

2.2 Multivariate Anlaysis

Multivariate EDA techniques generally show the relationship between two or more variables. Let us first take a look at the correlation matrix between the numerical variables.

	Air.temperature..K.	Process.temperature..K.	Rotational.speed..rpm.	Torque..Nm.	Tool.wear..min.
Air.temperature..K.	1.00	0.88	0.02	-0.01	0.00
Process.temperature..K.	0.88	1.00	0.02	-0.01	0.00
Rotational.speed..rpm.	0.02	0.02	1.00	-0.88	0.01
Torque..Nm.	-0.01	-0.01	-0.88	1.00	-0.01
Tool.wear..min.	0.00	0.00	0.01	-0.01	1.00

Looking at the correlation matrix, we can see how the variables are highly correlated in pairs: we can see that the variables Air.temperature and Process.temperature have a high correlation coefficient with each other but not with the other variables, which is null; also, the variables Rotational.speed and Torque have a high correlation coefficient with each other and almost null with the other variables, as for the variable Tool.wear, we note that it has practically zero correlation with all variables.
In the scatterplot matrix below, we can see the conditional distribution between the quantitative variables and the response Target



By looking at the diagonal, we can see the probability density functions of the predictors with respect to our target variable. For instance, we can notice how the more the Air temperature increases, the more machine failure (similar situation also considering the Process temperature). Considering the Rotational speed variable, much more Rotational makes more failure. Looking at the boxplot, we can notice that Rotational speed presents lots of outliers (especially in the case for which target is equal to 1, meaning that failure occur among machines which generally have less Rotational speed, there are some exceptions). There are also outliers for Process temperature, Torque Nm , when target is equal to 1 and Process temperature and Torque Nm when target is equal to 0.

3 Modeling

In the modeling phase, as mentioned above, three approaches will be used: **logistic regression**, **random forest** and **neural network**.
Although the ultimate purpose of the three is the same, the three models are profoundly different.
Logistic regression is an extension of linear regression to the case where the response variable is a class.
Random forest is a ensemble technique in which a classification tree is constructed for each sample, and then, based on the predictor values, the assigned class will be the one "highest rated" by the individual trees.
Neural network is a method in machine learning that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.
The purpose of this section is to build the model from the training set, making all due considerations in the choice of parameters, and to evaluate its accuracy and effectiveness using the validation set. The model found to have the highest accuracy will be chosen to rank the test set data.

3.1 Logistic Regression

Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems. In order to obtain a probability, the model uses a sigmoidal function that provides precisely an output between 0 and 1 that represents the probability of an event occurring taking as input the predictors.
The following results were obtained performing logistic regression using all predictors:

	Estimate	Std..Error	z.value	Pr...z..
(Intercept)	-30.5224381	19.0548086	-1.601823	0.1091947
Type.L	-0.6575340	0.2633904	-2.496423	0.0125453
Type.Q	-0.1279437	0.2019364	-0.633584	0.5263523
Air.temperature..K.	0.7587444	0.0957627	7.923169	0.0000000
Process.temperature..K.	-0.7548886	0.1279235	-5.901094	0.0000000
Rotational.speed..rpm.	0.0120878	0.0006831	17.694644	0.0000000
Torque..Nm.	0.2906290	0.0149737	19.409250	0.0000000
Tool.wear..min.	0.0135706	0.0014789	9.176338	0.0000000

The **AIC** obtained from this model with all variables as predictors is equal **1143.64**, all predictors are statistically significant, even the ordered categorical variable("Type") turns out to be so since it possesses the linear term (Type.L) with a p-value 0.012.

To have a comparison we will perform step wise logistic regression "stepAIC" in order to obtain the variables that provide the lowest value of AIC (provides a means for model selection). From stepwise logistic regression ,we obtained the same model as in the previous case so with the same AIC index and with the same significant variables then we are inclined to say that all variables should be retained.

In addition, we can see that there is a substantial difference between the null and residual deviance respectively 1773.8 and 1127.6, which it means that the model with the variables used fits the data better than the model with only the intercept.
Now to evaluate the model we will create the confusion matrix with the value of the train dataset using 0.5 as the decision threshold.

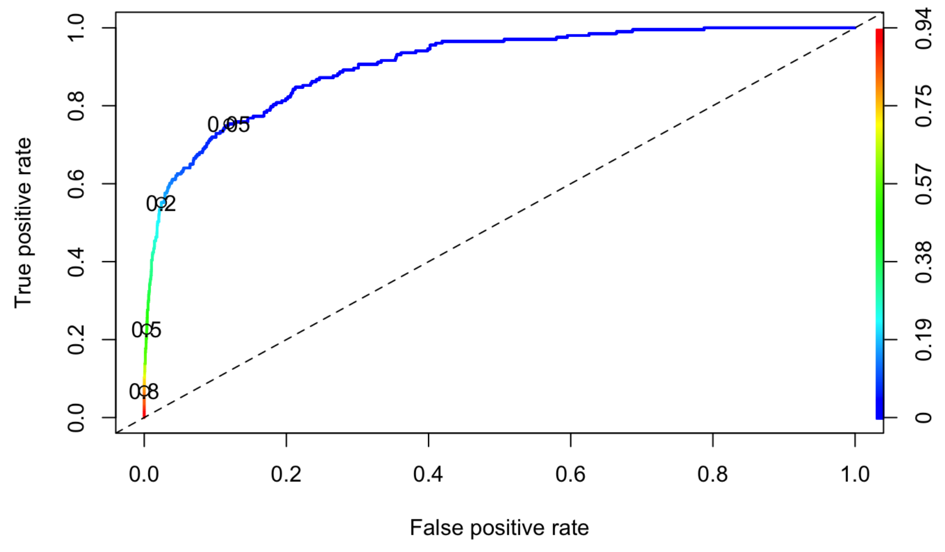
	True Value:0	True Value:1	Sum
Predicted: 0	5776	158	5934
PRedicted: 1	20	45	65
Sum	5796	203	5999

After that we have created the confusion matrix we can compute a series of metrics to evaluate model like:

- Accuracy : 97.04
- Error rate: 2.96
- Sensitivity: 22.16
- Specificity: 99,65

Accuracy can be defined as the percentage of correct predictions made by the classification model. Accuracy is a good metric to use when the classes are balanced, however, it is to be noted that accuracy is not a reliable metric for datasets having class imbalance.
For this reason, although the model has good accuracy, we have a low sensitivity. To increase the sensitivity and consequently reduce the specificity we can compute the ROC curve that is used to assess the accuracy of a continuous measurement for predicting a binary outcome. Each point of the chart has been realized varying the prediction threshold from 0 to 1.

File failed to load: file:///Users/afayad/Library/CloudStorage/OneDrive-Universita%CC%80degliStudidiCatania/Done/Stat%20learning/Fayad%20Project/Statistical%20Learning%20Project%20-%20Pre



From the ROC curve we can see that by decreasing the threshold for example from 0.5 to 0.2 we will get a sensitivity of about 60%, losing only about 5% specificity, moreover to evaluate the best threshold we used a function from the pRoc library that returns a value of 0.05, testing that threshold in the train dataset we obtained the following values:

- Accuracy : 87.64%
- Error rate: 12.36%
- Sensitivity: 75.36%
- Specificity: 88%

As we can see the metrics using 0.05 as decision threshold improved in terms of sensitivity but worsened all others, therefore, the best compromise between accuracy and sensitivity was obtained using 0.1 as decision threshold. The result of this configuration is the following confusion matrix:

	True Value:0	True Value:1	Sum
Predicted: 0	5450	73	5523
PRedicted: 1	346	130	476
Sum	5796	203	5999

From this, the following metrics can be calculated:

- Accuracy : 93%
- Error rate: 7%
- Sensitivity: 64%
- Specificity: 94%

Moreover the overall performance of a classifier, summarized over all possible thresholds, is given by the Area Under the ROC Curve (AUC). An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier.

Validation of the model

In this phase the model is tested with the best configurations (Decision threshold 0.1) on the validation dataset.
We get the below confusion matrix:

	True Value:0	True Value:1	Sum
Predicted: 0	1835	28	1863
PRedicted: 1	98	41	139
Sum	1933	69	2002

and the following metrics were obtained:

- Accuracy: 93,70%
- Error rate: 6.30%
- Sensitivity: 59.40%
- Specificity: 94,90%

The results are very similar to the results on the training set, therefore we conclude that the model is working well and it did not overfit.

3.2 Random Forest

The second approach that will be used to construct the classifier is the "Random Forest." This is an ensemble technique, which consists of applying a basic technique several times to small samples generated from the initial dataset and then averaging the results; this procedure is intended to reduce the variance.

The random forest then consists of applying many classification trees to the various decorrelated samples generated from the dataset. The special feature of the random forest is that, unlike other ensemble techniques such as Bagging (or Bootstrap Aggregation), it uses a smaller number of predictors for each constructed tree. In the case of classification that number is equal to $\lfloor \sqrt{p} \rfloor$ which therefore in our case it is equal to 3.
The random forest in constructing each tree uses two-thirds of the total observations and then evaluates it in the remaining third. It practically builds a validation set automatically for each individual tree. For this reason, the observations of the training set and the validation set are combined so as not to deprive the model of useful observations for a more accurate construction.
Once we constructed the random forest with the above elements, it returned the following confusion matrix:

	True Value:0	True Value:1	class error
Predicted: 0	7706	23	0.0029758
Predicted: 1	83	189	0.3051471

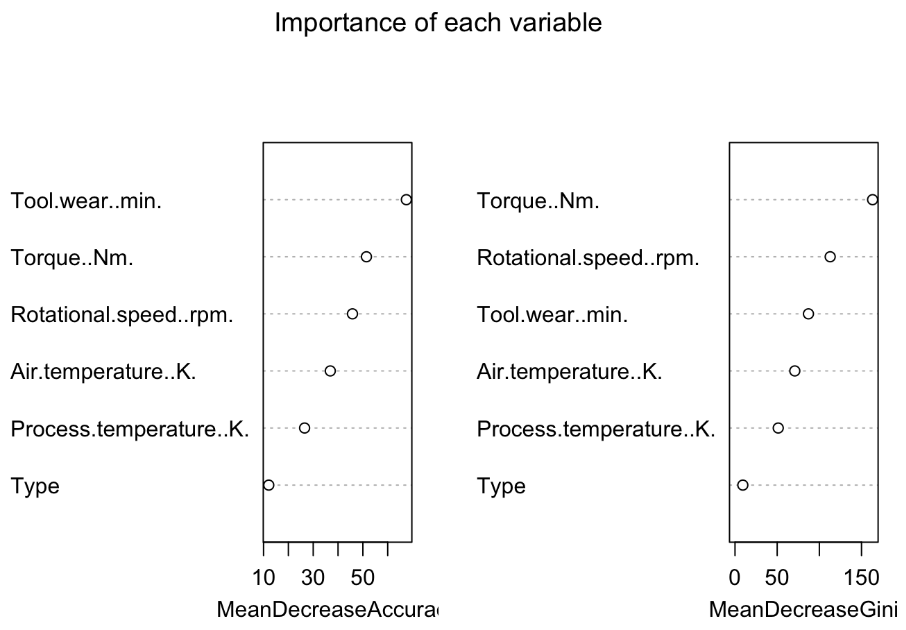
With an OOB estimate of error rate equals to 1.31% and an accuracy of 98.69%. Although the prediction has a very high accuracy, it is predicted to have a classification error for positives of 30%, i.e. the model can only correctly reveal 70% of failed machines. This result is obviously not acceptable, as the prediction of failures is more important than the prediction of non-failures. This error is due to the unbalanced nature of the dataset, in fact the model trains a lot to recognize non-faulty machinery but at the same time very little to recognize faulted machinery.

A sampling method must me used to solve the unbalanced dataset problem,in the next training, the 'stratified sampling' was set as the sampling method for constructing each tree by taking from the new training set (given by the union of the original training and validation) a number of observations with a target value of '0' ten times greater than the observations with a Target value of '1', reducing the proportion from 1:20 to 1:10.
Let us have a look on the new confusion matrix:

	True Value:0	True Value:1	class error
Predicted: 0	7674	55	0.0071161
Predicted: 1	62	210	0.2279412

As we can now see, a good compromise between accuracy and sensitivity has been achieved. In fact, compared with the previous set-up, the accuracy value went from 98.69% to 98.56%, while the specificity increased from (100 – 30)% = 70% to (100 – 23)% = 77%.

Let us no look at the following graph thaty shows the importance that each variable had in the construction of the random forest:



The Mean Decrease Accuracy refers to the mean decrease of accuracy predictions on the OOB (out of bag) samples, when a given variable is excluded by the model; the Mean Decrease in Gini coefficient is a measure of how each variable contributes to the homogeneity of the nodes and leaves in the resulting random forest.
Looking at the left graph, it is possible to see that the variables 'Tool.wear'is the most influential in the model accuracy value; in fact, its value of MeanDecreaseAccuracy differs from that of the other variables by a not insignificant value. Looking instead at the graph on the right, it can be seen that this time the variable 'Torque' is the one that most influences the homogeneity. In fact, its MeanDecreaseGini value deviates from that of the other variables by a not insignificant amount.

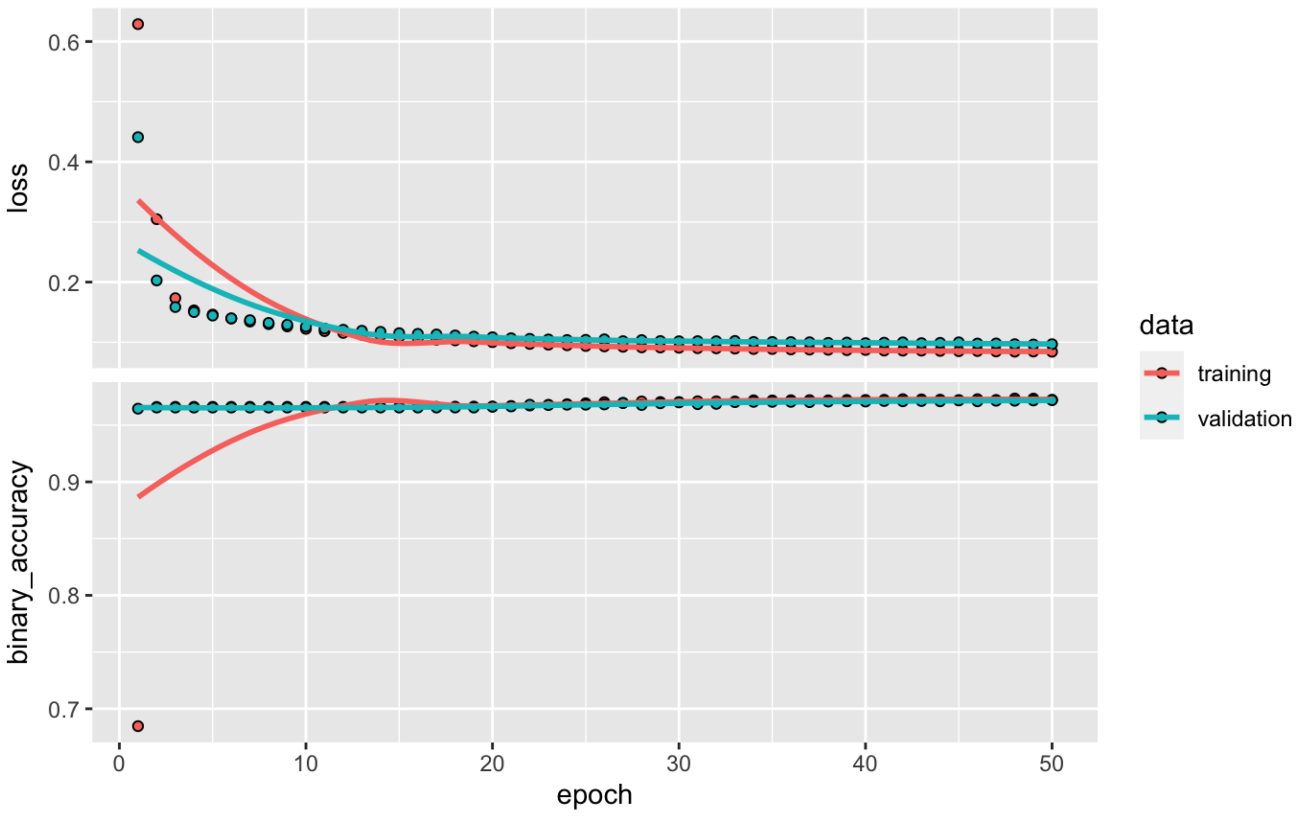
3.3 Neural Network

The last approach that will be used to build our classification model is the Neural Network. Neural networks are computational systems with interconnected nodes that function much like neurons in the human brain. Using algorithms, they can recognize hidden patterns and correlations in unstructured data, group and classify them, and, over time, continuously learn and improve. In feed-forward neural networks each perception in one layer is connected to each perception in the next layer, and information advances in a single direction, without feedback loops.

To build our neural network we made use of the keras library. Several attempts were made, varying the number of hidden layers, the number of neurons in each layer and the learning rate value, eventually a network was constructed with two hidden layers of 10 and 7 neurons respectively with a learning rate value of 0.001. This parameter composition was the one that resulted in the best accuracy and error rate values.

It was decided to fit the newly constructed model to both the training test and the validation set by setting the learning process a number of epochs equal to 50 and a value of batch size equal to 32. Both are integer values and both are hyper-parameters for the learning algorithm, i.e. parameters for the learning process, not internal model parameters found by the learning process. The number of epochs defines the number of times the learning algorithm will work on the entire training data set. The batch size defines the number of samples to be analysed before updating the internal model parameters.

The following graph shows the trend of the accuracy and the value of the loss function as the epochs increase.



The model was able to correctly predict the 97.25% of the target variables in the validation set. As mentioned in the previous approach, only accuracy is not enough to evaluate the goodness of the model in an unbalanced dataset. For this reason, the confusion matrix calculated using the observations of the validation set is presented below, from which we could subsequently derive the sensitivity values.

	True Value: 0	**True Value: 1**
Predicted: 0	1930	54
Predicted: 1	3	15

Looking at the confusion matrix, we can calculate that the sensitivity value is equal to 26.08%. A very low value when compared to that obtained by the random forest.

4 Results

To choose the best model we have to compare the metrics obtained from the 3 models applied to the validation set:

	Accuracy	Sensitivity	Specificity
Logistic regression	93.70%	59.40%	94.90%
Random forest	98.56%	76.83%	99.32%
Neural network	97.25%	26.08%	99.80%

Since we have an unbalanced dataset, we have to look for the best trade-off between accuracy and sensitivity. That's why predicting failures has a higher importance than predicting non-failures without giving up the overall accuracy of the model. Therefore it is possible to observe that the random forest is the best model that is able to predict failure for the machine.

Applying the model to the test set
Here it is shown a preview of the results on the test set:

X	UDI	Product.ID	Type	Air.temperature..K.	Process.temperature..K.	Rotational.speed..rpm.	Torque..Nm.	Tool.wear..min.	id_number	Target
4537	4537	L51716	L	302.4	310.2	1351	45.1	168	1	1
3685	3685	L50864	L	302.0	311.2	1270	65.3	182	2	1
2942	2942	M17801	M	300.7	309.6	1996	19.8	203	3	0
9614	9614	L56793	L	299.0	310.2	1377	62.5	92	4	0
4343	4343	M19202	M	301.7	309.8	1284	68.2	111	5	1
7510	7510	L54689	L	300.6	311.9	1372	60.1	212	6	1

5 Conclusion

After applying many models, we can now answer the busniess quesionts that were asked before.
To answer the first question, the best model, which can best predict the Target variable, is the one with the best trade-off between accuracy and sensitivity. Such a model is the one built using the random forest approach with the sampling method setting as 'stratified sampling'.
With regard to the second question, to see which variables have a greater importance in the construction of the model, we have to look at importance of variables figures, where we find that the variables that most influence the prediction of the Target variable are Torque and Tool.wear.
The last question asks us how accurately failures can be predicted, the answer is not to be found in the accuracy of the random forest, but in the sensitivity, i.e. that index which shows us the percentage of correctly predicted failures. The answer to the question is therefore, the model ia able to predict a failure 76.83% of the time.

6 Appendix

The following blaock has all the code used in the previous analysis.

File failed to load: file:///Users/afayad/Library/CloudStorage/OneDrive-Universita%CC%80degliStudidiCatania/Done/Stat%20learning/Fayad%20Project/Statistical%20Learning%20Project%20-%20Pre

```r
library(gt)
library(gamlss)
library(gamlss.mx)
library(e1071)
library(factoextra)
library(cluster)
library(gridExtra)
library(grid)
library(ggplot2)
library(psych)
library(lattice)
library(hopkins)
library(NbClust)
library(fpc)
library(GGally)
library(lemon) #for pretty printing
library("Hmisc")
library(tidyverse)
library(MASS)
library(vtable)
library(ROCR)
library(pROC)
library(randomForest)
library(tensorflow)
library(keras)
knit_print.data.frame <- lemon_print

barplot(table(train$Type)/length(train$Type)*100, col = c("green","blue",
"red"), main = "percentages of Type")

barplot(table(train$Target)/length(train$Target)*100, col =
c("green","blue"), main = "percentages of Target")
hist(train$Air.temperature..K., freq=FALSE, xlab= "Air temperature", col =
"lightblue", main = "Histogram of Air.temperature..K.")
summary(train$Air.temperature..K.)
print("skewness")
skewness(train$Air.temperature..K.)

print("kurtosis")
kurtosis(train$Air.temperature..K.)
hist(train$Process.temperature..K., freq=FALSE, xlab= "Process temperature
K", col = "#FFC0CB", main = "Histogram of Process temperature")
summary(train$Process.temperature..K.)
print("skewness")
skewness(train$Process.temperature..K.)

print("kurtosis")
kurtosis(train$Process.temperature..K.)
hist(train$Rotational.speed..rpm., freq=FALSE, xlab= "Rotational speed rpm",
col = "red", main = "Histogram of Rotational speed rpm")
summary(train$Rotational.speed..rpm.)
print("skewness")
skewness(train$Rotational.speed..rpm.)

print("kurtosis")
kurtosis(train$Rotational.speed..rpm.)
hist(train$Torque..Nm., freq=FALSE, xlab= "Torque Nm ", col = "orange", main
= "Histogram of Torque..Nm.")
summary(train$Torque..Nm.)
print("skewness")
skewness(train$Torque..Nm.)

print("kurtosis")
kurtosis(train$Torque..Nm.)
hist(train$Tool.wear..min., freq=FALSE, xlab= "Tool wear min", col =
"#FFD700", main = "Tool wear min")
summary(train$Tool.wear..min.)
print("skewness")
skewness(train$Tool.wear..min.)

print("kurtosis")
kurtosis(train$Tool.wear..min.)
train$Type <- factor(train$Type, ordered = TRUE, levels = c("L","M","H"))
train$Target <- as.factor(train$Target)
train$X <- NULL
train$UDI <- NULL
train$Product.ID <- NULL

valid$Type <- factor(valid$Type, ordered = TRUE, levels = c("L","M","H"))
valid$Target <- as.factor(valid$Target)
```

```
valid$X <- NULL
valid$UDI <- NULL
valid$Product.ID <- NULL


#setting categorical variable to null in the dataset used to generate coorelation matrix
cor_train = train
cor_train$Type <- NULL
cor_train$Target <- NULL
res <- cor(as.matrix(cor_train))


dfs <- data.frame(round(res, 2))

knitr::kable(dfs, format = "html")


train_quant_var <- train[, - 1]
Target<- as.factor(train$Target)
ggpairs(train_quant_var, aes(colour=Target))
Log_model = glm(Target~., data=train, family="binomial")
Log_model_df <- data.frame(summary(Log_model)$coefficients)
knitr::kable(Log_model_df, format = "html")

##AIC
aic_lr <- Log_model$aic

##Null Deviance
dn_lr <- Log_model$null.deviance

##Deviance
dev_lr <- Log_model$deviance
predTrainLr = predict(Log_model, type="response")

tb <- addmargins(table(predTrainLr>0.5,train$Target))
rownames(tb) <- c("Predicted: 0", "PRedicted: 1", "Sum")
knitr::kable(tb, format = "html", col.names = c('True Value:0', 'True Value:1', 'Sum') )

predictedtrain <-as.numeric(predTrainLr > 0.5)
error_Lr_0.5 <- mean(predictedtrain != train$Target)
accuracy_Lr_0.5 <- 1 - error_Lr_0.5
# ROC curve
ROCPred <- ROCR::prediction(predTrainLr, train$Target)
ROCPerf <- performance(ROCPred, "tpr", "fpr")
plot(ROCPerf,colorize=TRUE,lwd=2, print.cutoffs.at=c(0.05,0.2,0.5,0.8))
abline(a=0,b=1, lty=2)
# Best threshold value
my_roc <- roc(train$Target, predTrainLr)
coords <- coords(my_roc, "best", ret = "threshold")

# AUC value
ROCauc <-performance(ROCPred, measure ="auc")
AUC <- ROCauc@y.values[[1]]

# testing prediction train set with threshold 0.1

tb <- addmargins(table(predTrainLr>0.1,train$Target))
rownames(tb) <- c("Predicted: 0", "PRedicted: 1", "Sum")
knitr::kable(tb, format = "html", col.names = c('True Value:0', 'True Value:1', 'Sum') )



predTestLr = predict(Log_model, newdata=valid, type="response")

tb <- addmargins(table(predTestLr>0.1,valid$Target))
rownames(tb) <- c("Predicted: 0", "PRedicted: 1", "Sum")
knitr::kable(tb, format = "html", col.names = c('True Value:0', 'True Value:1', 'Sum') )

# The random forest is a ensemble method that use $sqrt(p)$ predictors,therefore in our case we will set mtry = 3.
set.seed(20)
df <- rbind(train, valid)
rf <- randomForest(Target ~ ., data = df, mtry = 3, importance = TRUE)

tb <- (rf$confusion)
rownames(tb) <- c("Predicted: 0", "Predicted: 1")
knitr::kable(tb, format = "html", col.names = c('True Value:0', 'True Value:1', 'class error') )

accuracy.rf <- (rf$confusion[1,1] + rf$confusion[2,2]) / (rf$confusion[1,1] + rf$confusion[1,2]+ rf$confusion[2,1] + rf
$confusion[2,2])

k <- length(df$Target[df$Target == 1])
```

```
stratified.rf = randomForest(Target ~ ., data = df, strata = df$Target, sampsize = c(10*k,k) , mtry = 3, importance = TR
UE)


tb <- (stratified.rf$confusion)
rownames(tb) <- c("Predicted: 0", "Predicted: 1")
knitr::kable(tb, format = "html", col.names = c('True Value:0', 'True Value:1', 'class error') )



#accuracy.stratified.rf <- (stratified.rf$confusion[1,1] + stratified.rf$confusion[2,2]) / (stratified.rf$confusion[1,1]
+ stratified.rf$confusion[1,2] + stratified.rf$confusion[2,1] + stratified.rf$confusion[2,2])

# Now we can visualize the importance of each variable in the built random forest
varImpPlot(stratified.rf, main = "Importance of each variable")
#tensorflow::set_random_seed(20)

#scaled.train <- scale(model.matrix(Target ~. - 1, data = train))
#true.train.targets <- to_categorical(train$Target)
#scaled.valid <- scale(model.matrix(Target ~. - 1, data = valid))
#true.valid.targets <- to_categorical(valid$Target)
#modelnn <-  keras_model_sequential()
#modelnn %>%
 #layer_dense(units = 10, activation = 'relu', input_shape = ncol(scaled.train)) %>% layer_dense(units = 7, activation =
'relu')  %>% layer_dense(units = 2, activation = "sigmoid")


#modelnn %>% compile(
 # loss = loss_binary_crossentropy,
  #optimizer = optimizer_rmsprop(learning_rate = 0.001),
  #metrics = metric_binary_accuracy
#)
#system.time(
 # history <- modelnn %>% fit(
  #  scaled.train,
   # true.train.targets,
    #epochs = 50,
    #batch_size = 32,
    #validation_data = list(scaled.valid, true.valid.targets)
#) )

#plot(history)
#pred<-as.matrix(modelnn %>% predict(scaled.valid) %>% `>`(0.5) %>% k_cast("int32"))


#tb <- table(pred[,2], true.valid.targets[,2])
#rownames(tb) <- c("Predicted: 0", "Predicted: 1")
#knitr::kable(tb, format = "html", col.names = c('True Value:0', 'True Value:1') )
trained.test <- test
trained.test$X <- NULL
trained.test$UDI <- NULL
trained.test$Product.ID <- NULL
trained.test$id_number <- NULL
trained.test$Type <- factor(test$Type, ordered = TRUE, levels = c("L","M","H"))
pred.target = predict(stratified.rf, newdata=trained.test, type="response")
test$Target <- pred.target
knitr::kable(head(test), format = "html" )
```