Report

Gabrielle Korkmaz

2022-10-09

Contents

Introduction

As the science and medical abilities increase from a year to another, it becomes important to have a better understanding of diseases. In the United States, one in three people will develop a cancer in their lifetime which is a significant proportion of the population so it should be a major forcus to know how to deal with it.

I decided to work on a set that contains many cancer indicators and the issue that I am going to try to predict : Did the patient survived ?

Preparing the environment

Load and divide dataset

The first step of this project is to oad the libraries and the data then to split it into a main and validation set.

I downloaded the data from the following link on Kaggle : https://www.kaggle.com/datasets/reihanenamdari/breast-cancer

As the data does not contain a lot of observations (4024), I decided to split it to have a validation set representing 10% of the total dataset. This allow us to keep a good number of observations to train and choose the models that will be used to predict the status of a patient.

To do so, I used the **createDataPartition** of the caret package :

```
trainIndex <- createDataPartition(df$Status, p = 0.9, list = FALSE, times = 1)
validation  <- df[-trainIndex,]
df <- df[ trainIndex,]
```

So now df reprensents the main dataset that will be used to study it and build our model.

Clean and transform

As there were no null values in this dataset, I did not have to delete or replace some values.

However there are many non numerical values so I checked the number of unique values per column.

Table 1: Number of unique values per columns

	Unique values
Age	40
Race	3
Marital.Status	5
T.Stage	4
N.Stage	3
X6th.Stage	5
differentiate	4
Grade	4
A.Stage	2
Tumor.Size	108
Estrogen.Status	2
Progesterone.Status	2
Regional.Node.Examined	54
Reginol.Node.Positive	37
Survival.Months	107
Status	2

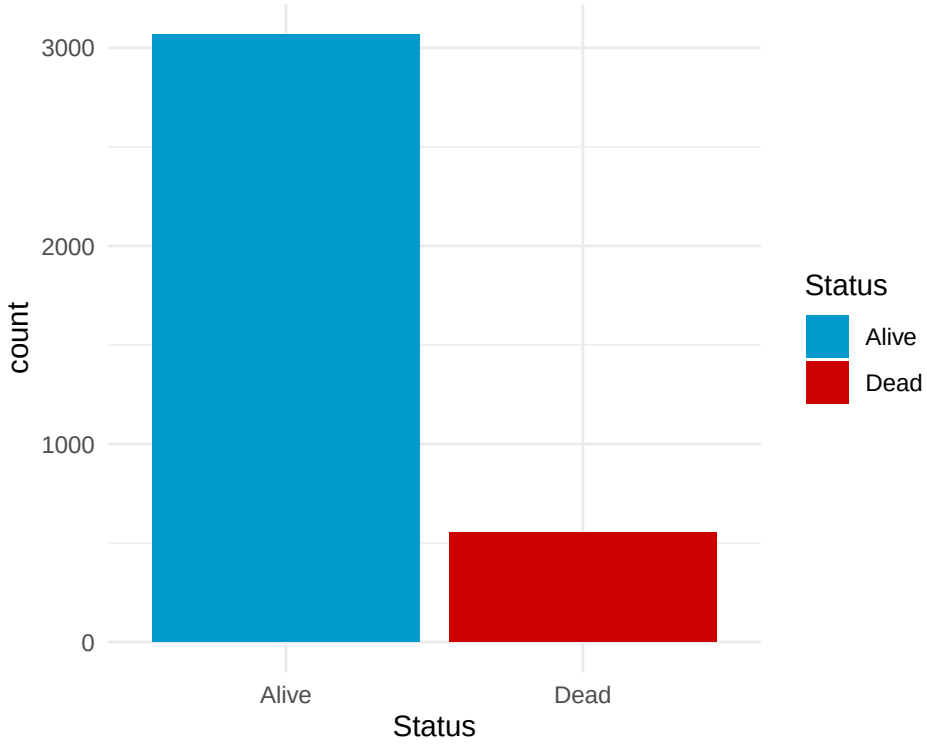
We can observe that most of the non numerical columns have only a few unique values so I decided to keep the original version of df (***df_original***) and transform df into all numerical predictors by transforming them into factors then numerical.

```
df_original <- df
df <- as.data.frame(apply(df, 2, function(x) ifelse(x == "numeric", x, as.numeric(as.factor(x)))))
```

Exploratory Data Analysis

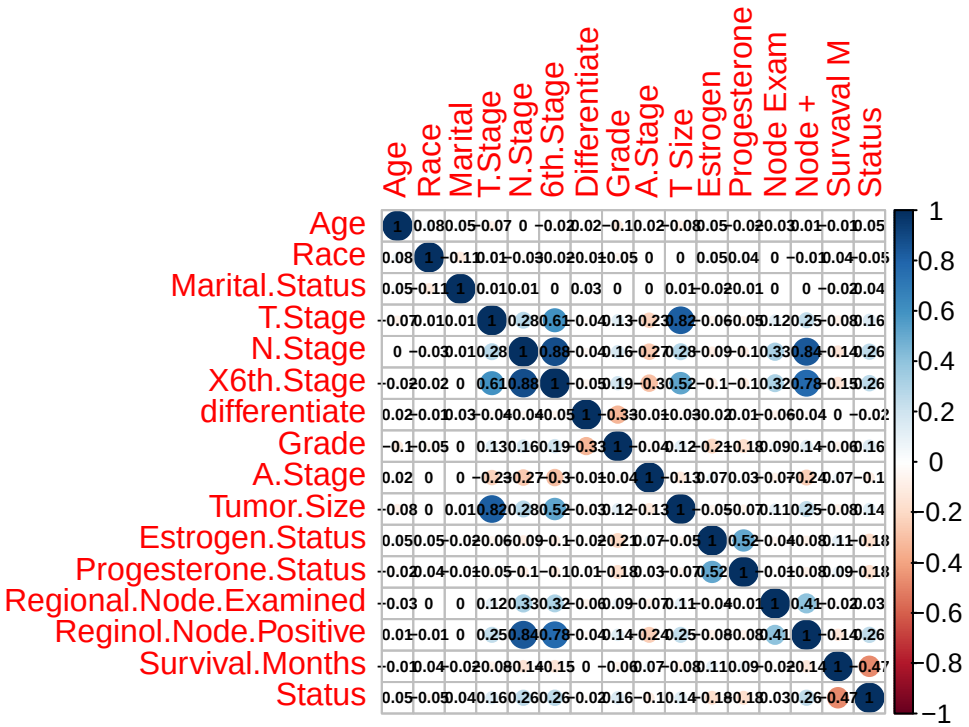
Proportion based on status

I first started to check how many people survived the cancer by calculating the proportion of person alive which is **0.847** and by doing the following simple plot :



Correlation matrix

To get which predictors are the main ones and which variable has the most influence of the status and the chances of surviving of someone, I realized a correlation plot.

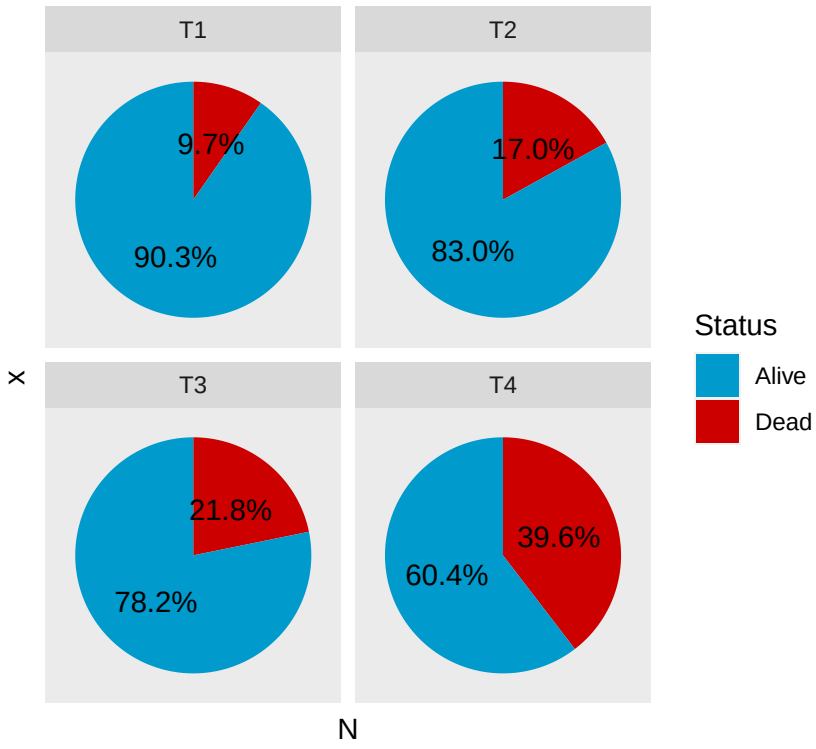


As we can observe, some variables like the marital status, the race, the age or the differentiation have almost no impact on the status so we'll see later that we can remove them from our data frame to obtain the best accuracy.

Impact of T.Stage

One of the parameter that is often used to describe a cancer and understand its gravity is the stage. There are four stages : - **1 : Early Stage** - The tumour is small - Localized - **2 : Localized** - Larger than stage 1 - **3 : Regional Spread** - Surrounding tissues are affected - Lymph nodes - Tumor has grown - **4 : Distant Spread** - Spreaded more than original location - Matastases

As it is one of the first thing that is found and told to the patient, I thought that is was important to verify how much the impact is important.

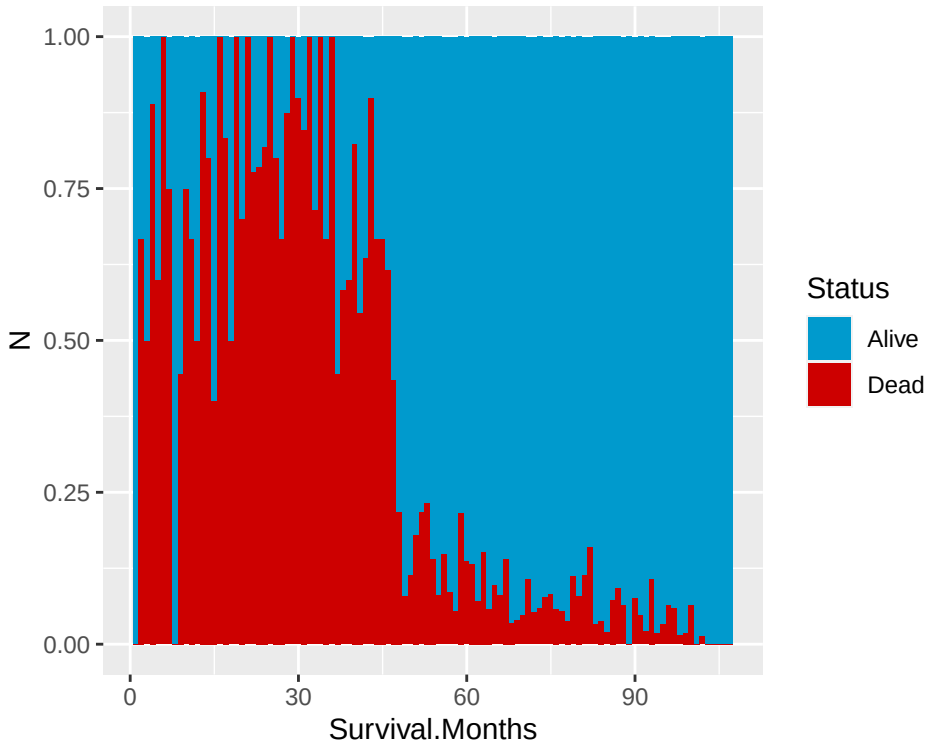


We can see that the percentage of people that did not survive increases a lot as the stage is advanced which is not surprising.

Survival Months

On the correlation matrix we saw that the predictor that has the more influence is the survival months predictor. We can tell from the following plot that if the survival months increase, the chances of not dying from cancer increases too. This mostly depends on when does the person was taken in consideration in the study that is at the origin of this dataset.

```
## Adding missing grouping variables: `Survival.Months`
```



Predictions

Now, I am going to split my main dataset ***df*** into a test and train model to try many models and at the end choose which one is the best. As we're dealing with medical and life/death situation, and taking into account that the dataset is quite small, my goal is to reach an accuracy over 85% with those train and test set and around 80% with validation set.

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
colnames(df) <- make.names(colnames(df))
trainIndex <- createDataPartition(df$Status, p = 0.85, list = FALSE, times = 1)
train_set <- df[ trainIndex,]
test_set  <- df[-trainIndex,]
```

For the models, I mostly used the **caret** package.

For every model I tried to predict the outcome by using all the predictors and by using only the ones with the most impact to then compare which method is the best.

KNN (K-Nearest Neighbors)

This supervised machine learning algorithm useful in classification predictions as we have here. KNN is mostly used for classification as it classifies the data point on how its neighbor is classified. Which means that the new data points are classified based on the similarity measure of the earlier stored data points.

I chose k by taking the closest odd number after the square root of the number of variables.

```
fit_knn <- train(as.factor(Status) ~ .,  method = "knn",
                 tuneGrid = data.frame(k = 5),
```

```
                 data = train_set)
predict_knn_all <- predict(fit_knn, newdata = test_set)
acc_all <- sum(predict_knn_all==test_set$Status)/nrow(test_set)

fit_knn <- train(as.factor(Status) ~ Survival.Months + Reginol.Node.Positive + T.Stage + N.Stage + Estr
                 tuneGrid = data.frame(k = 3),
                 data = train_set)
predict_knn <- predict(fit_knn, newdata = test_set)
acc_few <- sum(predict_knn==test_set$Status)/nrow(test_set)

accuracy_results <- tibble(method = "KNN Accuracy", Accuracy_All = acc_all, Accuracy_Part = acc_few)
```

```
## KNN Accuracy (with 7 predictors) :  0.8802947
```

```
## KNN Accuracy (with all predictors) :  0.8821363
```

Random forest

To apply this algorithm, I used caret package with the *rf* method and also with the *Rborist* method which do the same process but faster by applying it differently on the system side.

At the end off each method, I add the results to the result table named ***accuracy_results***.

```
## RBorist Accuracy (with 7 predictors) :  0.8802947
```

```
## RBorist Accuracy (with all predictors) :  0.8563536
```

```
# Random forest
#With all the predictors
fit_rf <- train(as.factor(Status) ~  .,
                data=train_set,
                method = "rf", trControl=trainControl(method="cv", number=5))
predict_rf_all <- predict(fit_rf, newdata = test_set)
acc_all_rf <- sum(predict_rf_all==test_set$Status)/nrow(test_set)

# With only a few predictors
fit_rf <- train(as.factor(Status) ~  Survival.Months + Reginol.Node.Positive + T.Stage + N.Stage + Estr
                    data=train_set,
                    method = "rf", trControl=trainControl(method="cv", number=5))
predict_rf <- predict(fit_rf, newdata = test_set)
acc_few_rf <- sum(predict_rf==test_set$Status)/nrow(test_set)

accuracy_results <- bind_rows(accuracy_results,
                              tibble(method="Random Forest",
                                     Accuracy_All = acc_all_rf, Accuracy_Part = acc_few_rf))
```

```
## Rf Accuracy (with 7 predictors) :  0.8876611
```

```
## Rf Accuracy (with all predictors) :  0.9023941
```

Generalized Linear Model

We know that linear regression is not the most powerful algorithm for classification models so I decided to use the generalized linear model that is based on a function that will give a discrete output.

By using a code similar to what was done before, I obtained the following results :

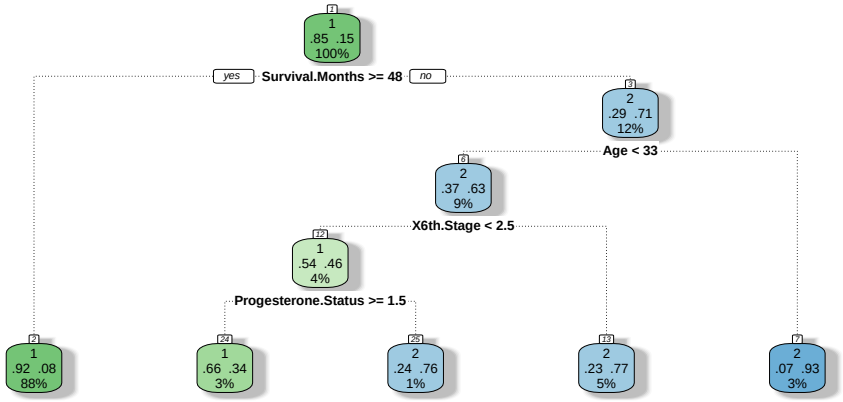
```
## Rf Accuracy (with 7 predictors) :  0.8802947
```

```
## Rf Accuracy (with all predictors) :  0.878453
```

RPart

The *Rpart* method stands for Recursive Partitioning and Regression Trees.

By using all the predictors, I obtained this classification tree :



Rattle 2022−Nov−07 23:19:25 gabriellekorkmaz

Those are the obtained accuracies :

```
## Rf Accuracy (with 7 predictors) :  0.8950276
```

```
## Rf Accuracy (with all predictors) :  0.8968692
```

Combining methods

Finally, I decided to try to combine the methods so that it will classify the output depending on what is predicted by the majority of the previous algorithms.

```
# Creating a dataframe with all the previous predictions
pfinal <- tibble(predict_knn = as.numeric(predict_knn_all),
                 predict_rborist = as.numeric(predict_rborist_all),
                 predict_rf = as.numeric(predict_rf_all),
                 predict_glm = as.numeric(predict_glm_all),
                 predict_rpart = as.numeric(predict_rpart_all))

# Choose the predicted outcome depending on the majority
prediction_combined <- ifelse(rowSums(pfinal)>7, 2, 1)

# Compute accuracy
acc_all_combined <- sum(prediction_combined==test_set$Status)/nrow(test_set)
```

Results

After testing all the different algorithms, we have this final results table :

Table 2: Accuracy of every tested machine learning algorithm/ method

method	Accuracy_All	Accuracy_Part
KNN Accuracy	0.8821363	0.8802947
RBorist	0.8563536	0.8802947
Random Forest	0.9023941	0.8876611
GLM	0.8784530	0.8802947
RPart	0.8968692	0.8950276
Combined	0.8913444	0.8987109

We can observe that the best accuracy is obtained with the Random forest method so I chose to apply it with the validation set.

First we transform the validation set so that it becomes only numerical like our *df* and then I applied the random forest method :

```
# Validation set transformation
validation <- as.data.frame(apply(validation, 2, function(x) ifelse(x == "numeric", x, as.numeric(as.fa

# Random Forest model
fit_final <- train(as.factor(Status) ~  .,
                data=df,
                method = "rf", trControl=trainControl(method="cv", number=5))
predict_final <- predict(fit_final, newdata = validation)
acc_final <- sum(predict_final==validation$Status)/nrow(validation)
```

The accuracy on the validation set is :

```
## [1] 0.7955112
```

Conclusion

During this final project, I observed that all the classification models gave a similar accuracy th less than 0.3 maximum difference. I also think that the accuracy could be improved with a bigger dataset and that we could probably reach a 90% or more accuracy which will be very useful in medical procedures to analyze the cancer and its level of danger for the life of the patient.

I wanted to thank HarvardX and edX for this course that improved my skills in R by taking some real life examples and applications to make everyone realize the importance of data and data science in our world.