USE CASE STUDY REPORT

Group No.: Group 6

Student Names: Nazli Rafei and Ana Ruiz

Executive Summary:

This project approached two problems: one using Covid-19 data and the other one using financial information.

The pandemic put the world in a difficult position; as the cases increased, each country followed different strategies to prevent new cases. At the same time, a race to create an effective vaccine started. Now, we have at least four other types of vaccines, but the distribution is unequal, and again countries are taking different approaches to fully vaccinate their population. In this case, we obtained data from Kaggle with the statistics about the number of vaccinated people in each country. We tried to used demographic statistics to predict the total number of vaccinated people per country. The case showed that the data quality was not good enough to get conclusive results using regression algorithms.

Banking is one of the most competitive industries; banks compete with each other and other financial institutions [1]. Retaining customers is one of the critical rules of having a growing industry. They bring money or transfer money and make it possible for a bank to grow and interact with other fund-related industries(non-Banks). In this study, data was obtained from Kaggle[2], and performance of different classification techniques (, in order to make it possible to suggest an efficient model, so we can predict if customers will leave or stay in the studied bank, with regards to all variables we have here (10 demographic and personal attributes) from a data set of 10000 customers of European banks. In this study, we will evaluate the effect of each feature. The study results found that Neural network model is the best amongst all applied models to predict churning in customers.

• Background and Introduction

While banks have central role in the economy, they face a highly competitive environment, due to similar benefits offered by other markets (including other banks, or non-banks), or other factors such as improving technologies and demanding customers, and makes them vulnerable as well[3]. All these factors, leads to an attention to retain and restore customers. A lot of studies have been conducted on evaluating all factors affecting customers retention in a bank. In this study we want to evaluate factors including in Table. 1, on customer churning on our data. Using new methods of machine learning for prediction of churning customers, we intend to propose the most efficient methods amongst studied supervised machine learning classification methods (Naïve Bayes, K-NN, SVM, Neural network, decision tree and random forest), in order to take action before the bank undergoes main problem in future, and takes needed actions ahead of time. We will discuss model performance, feature selection and model selection in this report.

Our second section is about covid vaccines. As countries don't share a common strategy to vaccinate people, the data isn't updated for all countries. Also, the dataset is restricted to daily vaccination, total vaccinations and people vaccinated and different presentations of the same data, so we used data from the United Nations statistics Division (UNSD) of the department of economic and social affairs(DESA) to complement the information for each country and the final dataset is as described in Table 2. In our intent to predict the number of vaccinated people we used regression models(Multivariate linear regression, polynomial regression and kNN regression).

• Data Exploration and Visualization

a. Bank Churning

As we can see in Table.1, there are 13 variables, we will remove Id and surname since they are not affecting our results. Our response variable is "Existed", which later we will convert it to categorical variable and label them as "Stayed", or "Left", so we have a better visualization of results. Fortunately, we did not have any missing values in all data we have, so there is no need to use any method to replace them (As we had a lot for previous project of us – vaccination) Figure 1. First evaluations shows that we have **7963** customers who ***stayed*** with the bank, and **2037** customers who ***left*** the bank.

Table 1.

Variable	Type	Definition	Minimum	Maximum	Mean	Std. Deviation
CustomerId	Nominal	Customer ID				
Surname	Text					
CreditScore	Interval	Customer's credit score	350	850	650.53	96.65
Geography	Nominal	France, Germany, Spain				

Gender	Nominal	Female, Male				
Age	Ratio		18	92	38.92	10.49
Tenure	Interval	Tenure of deposit	0	10	5.01	2.89
Balance	Ratio		0	250898.1	76485.89	62397.41
NumOfProducts	Interval	Number of bank account affiliated products the customer has	1	4	1.53	0.58
HasCrCard	Binary	Does the customer have a credit card through the bank? (Yes=1, No=0)	0	1	0.71	0.46
IsActiveMember	Binary	Is the customer an active member? (Yes=1, No=0)	0	1	0.52	0.50
EstimatedSalary	Ratio	Estimated salary of the customer	11.58	199992.5	100090.2	57510.49
Exited	Binary	Did the customer leave the bank within the last 6 month? (Yes=1, No=0)	0	1	0.2	0.40

As for outliers, we visualized data so we can see outliers, Figure 1.

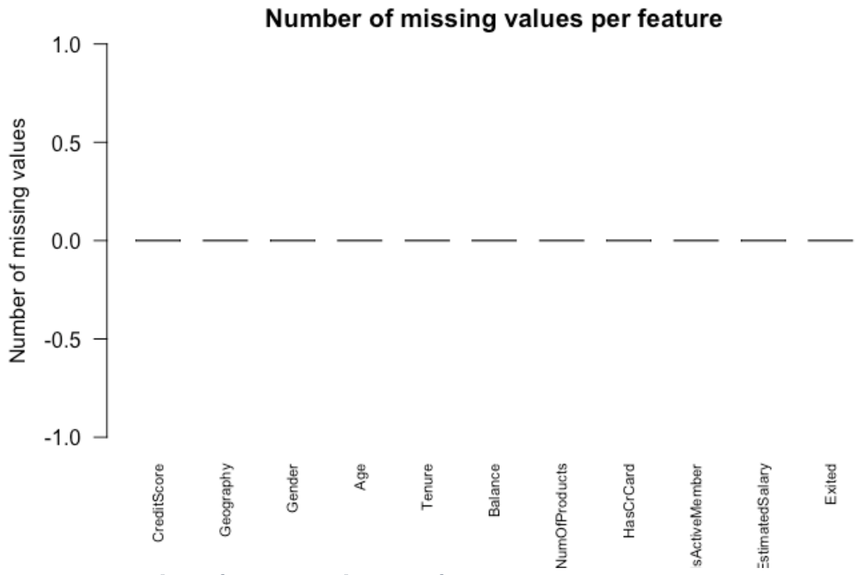


Figure 1. Number of missing values per feature

For numeric data, we did bar plot to see the distribution, and check for the necessity of any data transformation before we start modeling. Figure 1- 5. as we see, majority of customers has score between 600 and 700, and we can say the distribution is fairly normal, and we only have 11 outliers in the class of left customers. And we have outliers both in "Left" and "Stayed" class for "Age" variable for a total of 13 in left class and a total of 486 outliers in both classes in “Age”. And in related histogram we see majority of customers are around

40, and we have light right skewness. 486 outliers are only 0.05% of the data, so we do not have significant number of outliers compared to whole size of the data, so we will continue with data as it is. There are no outliers in "Balance" variable, and we do not have a normal distribution, and majority is 0. For “Estimated Salary” our distribution is not normal, and it is in fact uniform and there is no outlier.

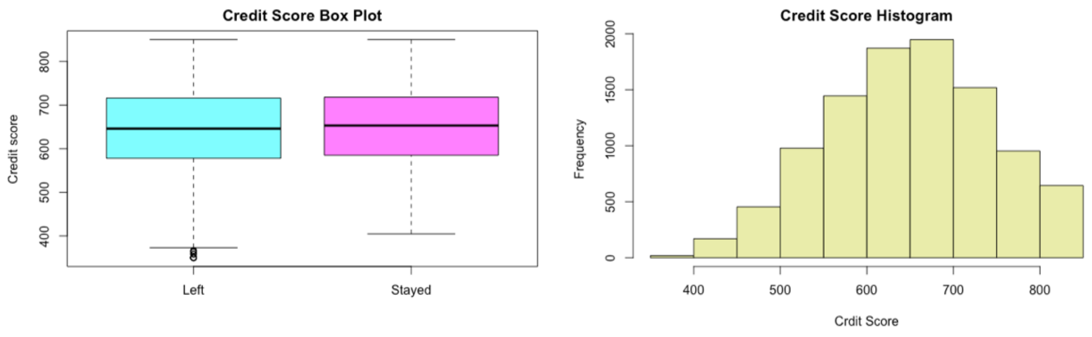


Figure 2 - Credit Score Distribution

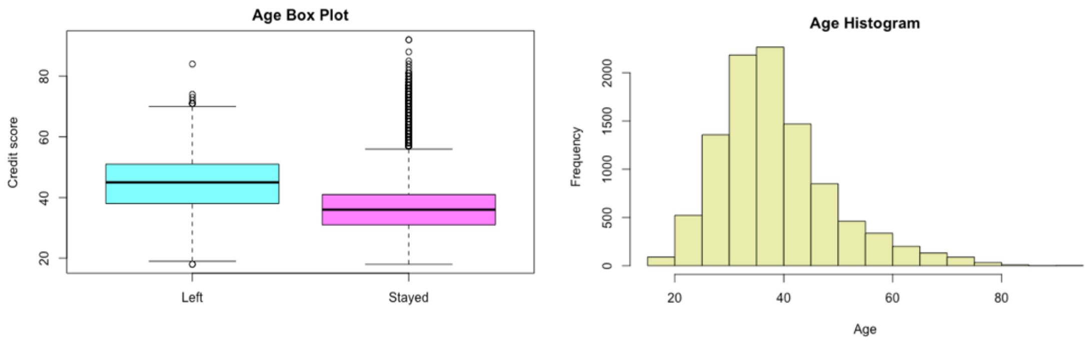


Figure 3. Age distribution

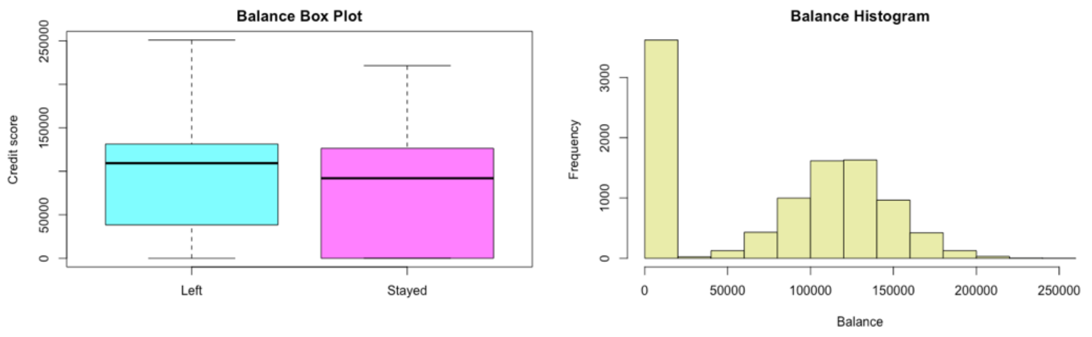


Figure 4. Balance Distribution

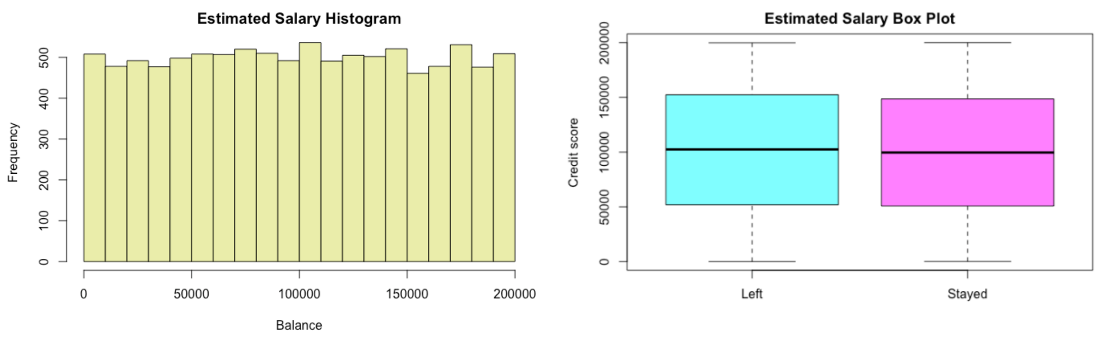
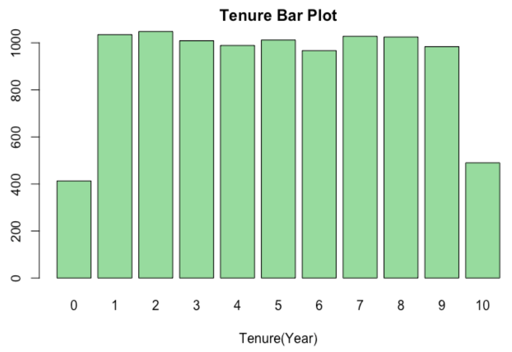


Figure 5. Estimated Salary Distribution

Since, Tenure and Number of Products Bar plot have integers in their data, bar plot has been chose to depict the distribution of data Figure 6 and Figure7. For Tenure we have a symmetric distribution which has the lowest values at both ends, lower end for deposits tenured in less than a year or 10 years and the other for 10 year. For Number of products we do not have normal or uniform distribution and most of customers prefers to have up to two products. Therefore, we need to use transforming data wherever we have to satisfy normality assumptions.



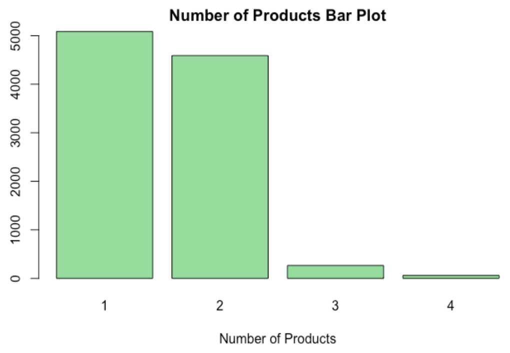
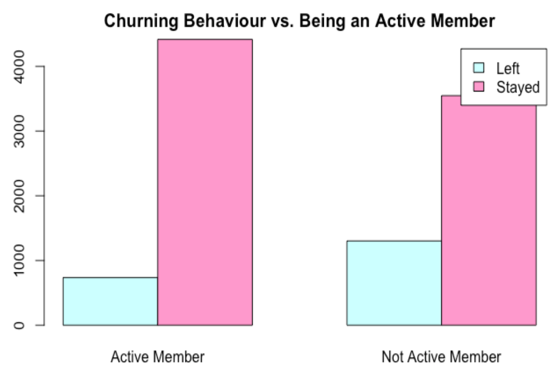


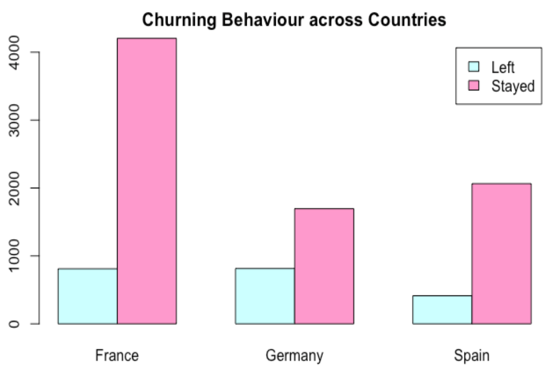
Figure 6. Tenure Distribution

Figure 7. Number of Products Distribution

In order to find out if churning behavior is dependent a=on any of our variables we visualize their distribution also to have a number to get better understanding and verifying the significant of dependency, we used Chi-square test.

Comparing churning behavior in comparison of active members, and non-active members, we see for both groups they mostly will "Stay", but ratio of leaving is more for Non-Active Members, and the P value is significant, and then there is a relation between being or not being an Active member and stay or leave the bank (X-squared = 242.99, df = 1, p-value < 2.2e-16) Figure 8. The plot shows that most customers with one or two products stayed with the bank but customers with more products decided to left, and the p value is significant and we can say that there is a relation between the number of products a customer has and the decision to leave or stay at the bank (X-squared = 1503.6, df = 3, p-value < 2.2e-16) Figure 9. For both men and women, we have the majority deciding to "Stay", and data are balance, but in women group we see ratio of "Stayed" to "Left" is more than men (X-squared = 112.92, df = 1, p-value < 2.2e-16) Figure 10. As for different countries, in all three countries, they preferred to "Stay" with bank. We can say there is a good balance of ratios of stay to leave in Germany and Spain, but in France we have more records, but the behavior is similar to Spain and Germany (X-squared = 301.26, df = 2, p-value < 2.2e-16) Figure 11. Having a credit card in bank In both groups we see majority of customers will stay with bank, and behavior and ratio is similar, and from P value, we can say that, just having a credit card does not say anything about leaving or holding to a bank (X-squared = 0.47134, df = 1, p-value = 0.4924) Figure 12.





Figur

Figure 10 Churning behavior across countries

Figure 8. Churning behavior vs Being Active member

Figure 9. Churning behavior across No. Products

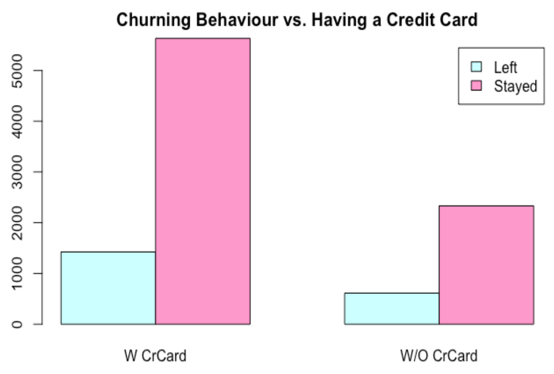


Figure 12. Churning behavior vs Having credit card

b. Covid Vaccination

As we can see in Table 2, our dataset has 15 variables and our target variable is the no_people_vaccinated. As we can see later we are going to remove country_code and country as they don't add any information for generalization purposes. Total_vaccinations and people_fully_vaccinated are also removed, we have many missing values that we are fixing using a regression model.

Table 2.

Variable	Type	Definition	Min	Maximum	Mean
Country_code	Character	ISO country code			
country	Character	Country name			
Total_vaccinations	Numeric	Total vaccinations in the country	107	107060274	2764445
People_fully_vaccinated	Numeric	Total people who completed the required number of vaccinations to be considered inmune	0	37459269	635861
No_people_vaccinated	Numeric	Total number of people who received at least one dose of the vaccine	0	69601005	2128584
surface	character	Surface of the country in squared kilometers			
Pop_density_km	Numeric	Ratio of density of the population per kilometer	0.10	25969.80	105.45
Sex_ratio_p100	Numeric	Ratio of sex in population per hundred of people	-99	301.20	98.20
GDP_capita	Numeric	GDP per capita of each country	-99	169492	10664
Intern_trade	Character	International trade index of each country			
Urban_pop	Numeric	Ratio of urban population per kilometer	8.40	100	66.14
Pop_age	Character	Ratio of population age per country below 14 years and above 65 years			
Health_expend	Numeric	Total expenditure of each country in health	-99	17.100	-4.393
Health_phys	Character	Number of physicians per hundred of population			
Ed_tert	Character	Ratio of people per hundred who has tertiary education			

The percentage of missing values is almost the 70% of the total of records on columns health_phys and ed_tert, the only features with missing values, Figure 1a show the distribution. We have some numeric values as characters (surface, intern_trade, pop_age, health_phys, ed_tert), as some of them are a combination of two statistics, we created new columns from this statistics and found new missing data as we can see in Figure 2a.

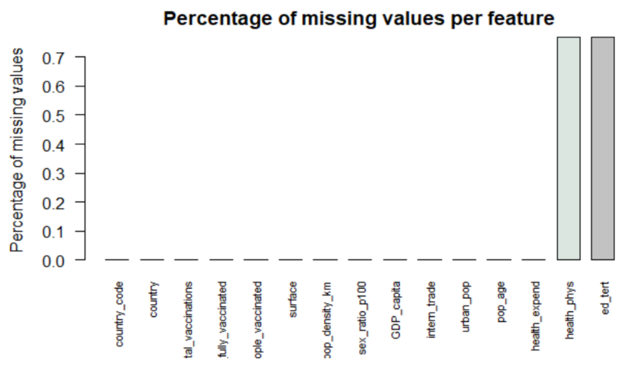


Figure 1a. Percentage of missing values per feature

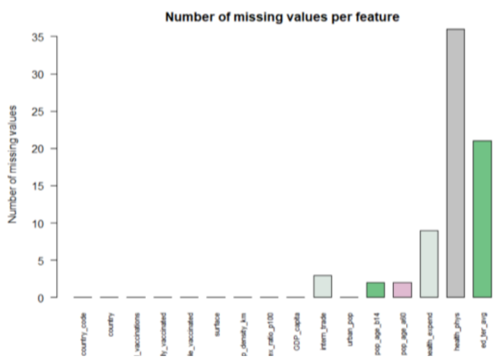


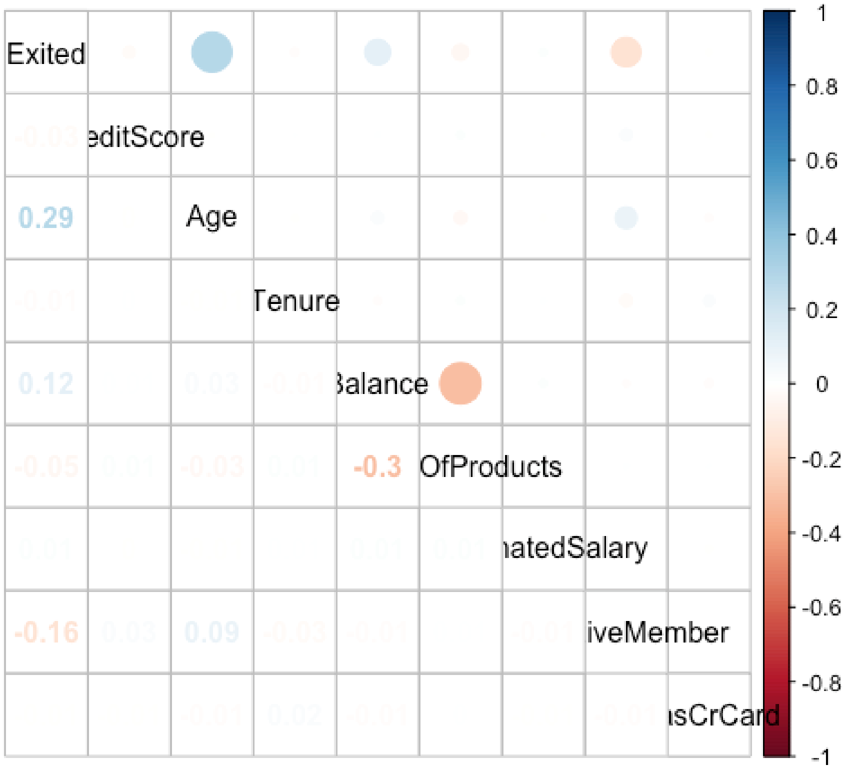
Figure 2a. Percentage of missing values per feature

The new columns are pop_age_b14, pop_age_a60,ed_ter_avg and we are going to use pop_age_a60 as the predictor for the features with the missing values.

- ## Data Preparation and Preprocessing

a. Bank Churning

Making a correlation matrix, we can see there is no correlation between variables, thus there would not be any redundancy in variables, also none of independent variables are not correlated with our dependent variable "Exit".



First we performe some models and make comparison between their performances, at the end we choose the two best and we do more analysis on them.

Naïve Bayes is our first model, in order to perform naïve bayes we need to transfer data, so they follow Gaussian distribution. Doing transformation, we see it improved for Estimated Salary but didn't change anything for Balance values. For other models, including K-NN, ANN, and SVM we need to transfer categorical variables to dummy variables. Another data preprocess is in implementation of SVM, and Neural network (ANN), which a normal scaling and min-max normalization is required. However, in applying decision tree and random forest none of steps above is needed.
In order to choose a set of optimal parameters for our learning algorithms, for each model, we used a grid search with 10- fold cross validation and repetition of 3 times, doing so optimum number of models will be measured Table 3.

Table 3. Optimum number for each model using cross validation

Model	Explanation of number	number
K-NN	optimum number of neighbors	K=9
Decision tree	complexity parameter	Cp = 0.01
Random forest	optimum potential variables to split on	Mtry = 4
Neural network	optimal number of the hidden layer's nodes, and weight decay	Size= 5 weight decay= 0.1

All data split in training and validation data set for ration of (60% over 40%).

b. Covid Vaccination

Figure 3a shows the correlation of the features, we can see that the demographic data is the most highly correlated, especially the predicted where missing data was filled with a regression, but the correlation with the target variable no_people_vaccinated is close to zero. Our dataset is divided in two: 60% for training and 40% for validation.

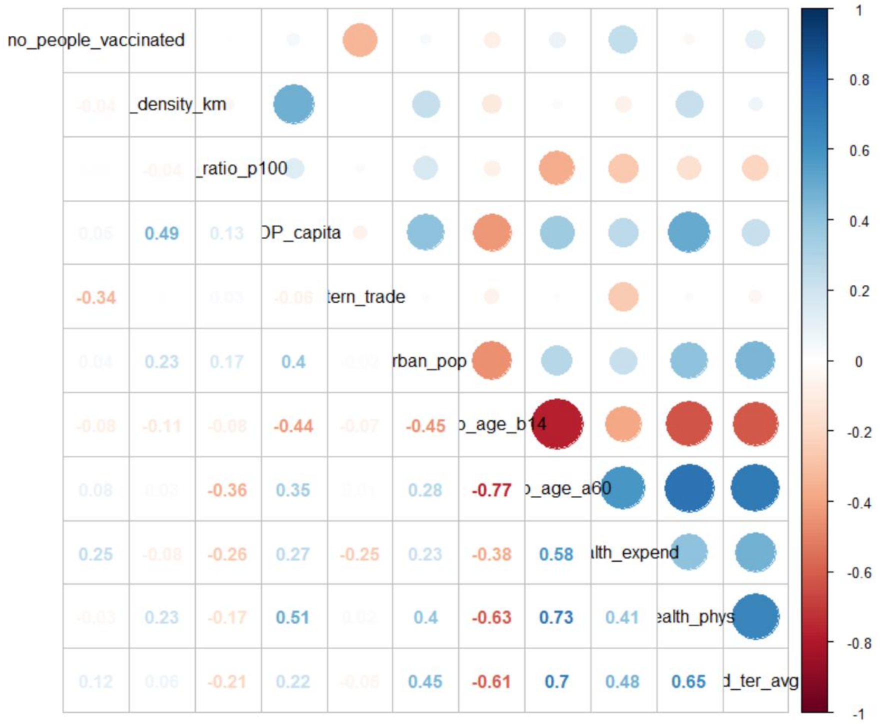


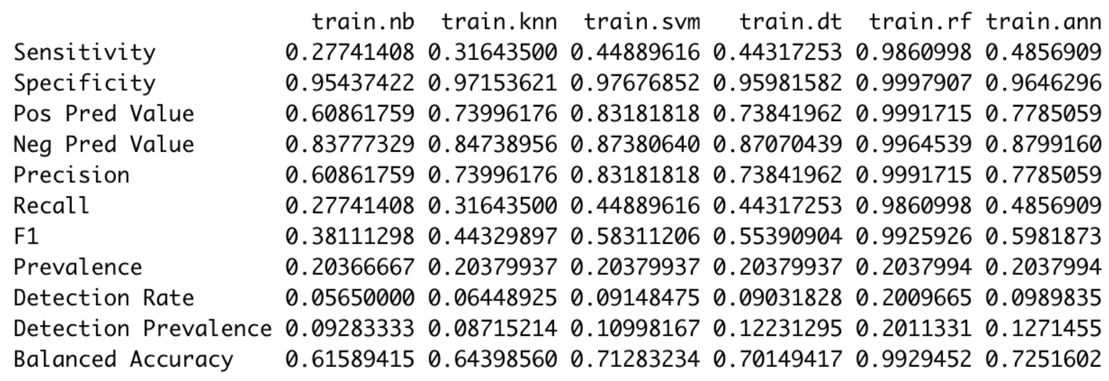
Figure 3a. Correlation of predictors

Our first model was the multivariate linear regression since our features are independent from the outcome. To improve the approximation of the relationship of the predictors with the outcome we used a polynomial linear regression, our options where of second and third degree polynomials. The RMSE values were still too high so our last attempt was to use a kNN regression with an optimal number of neighbors of k = 5.

- ## Performance Evaluation
  ### a. Bank Churning

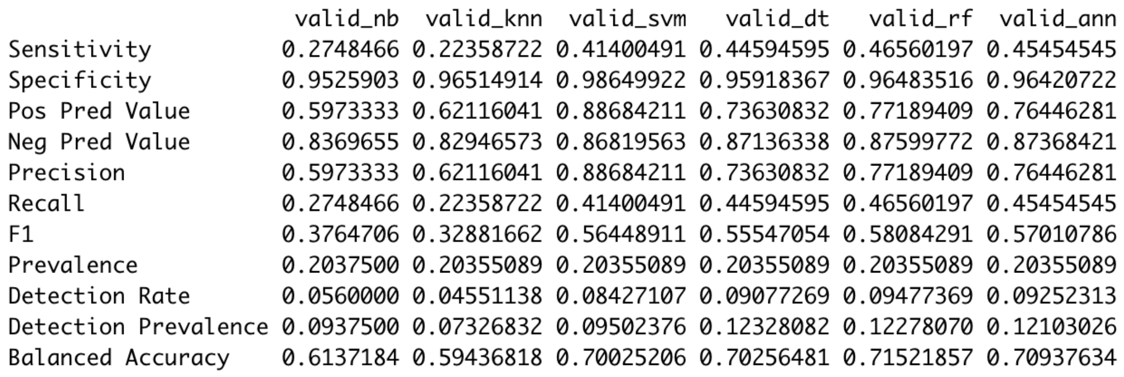
In Table 4, and Table 5, we can see comparison of all methods with regards to different prediction accuracy measures (Kappa, Accuracy, precision, recall FI), for Naïve Bayes, for validation data set, as we can see our Kappa number is 0.2846 which is in range of 0.2-0.4, which we call it as fair. Also, our F: 0.37647 which is close to 0, and as a whole the performance of our model is not that good. For K-NN model, for training data set, the best accuracy (0.8161061) and still a good kappa (0.2663960) was for k = 9, and for training data set , our Kappa number is 0.2478 which is in range of 0.2-0.4, which we call it as fair. Also, our F1: 0.32882 which is close to 0, and as a whole the performance of our model is not that good. As for SVM model, and for validation data set the Kappa number is 0.4997 which is in range of 0.4-0.6, which we call it as Moderate. And our F1: 0.56449 which is close to 0, and as a whole the performance of our model is Moderate. For Decision Tree model, our Kappa number is 0.4748 which is in range of 0.4-0.6, which we call it as Moderate, and F1: 0.55547 which is close to 0, and as a whole the performance of our model is Moderate. So far Decision trees performs better than Naive Bayes and K-NN and SVM performed pretty much the same. Applying Random Forest, we see that, our classification is not as good as expected since the accuracy is 0.863 and kappa = 0.5039. Based on above comparison, random forest and neural network are two potential models to be discussed as solutions for the problem since they have the best F1 value, highest accuracy and the precision also is high. In random forest we have overfitting , on the other hand for Decision tree we see some underfitting.

Table 4. Performance measures (Training data set)



```
                       train.nb  train.knn  train.svm   train.dt  train.rf train.ann
Sensitivity          0.27741408 0.31643500 0.44889616 0.44317253 0.9860998 0.4856909
Specificity          0.95437422 0.97153621 0.97676852 0.95981582 0.9997907 0.9646296
Pos Pred Value       0.60861759 0.73996176 0.83181818 0.73841962 0.9991715 0.7785059
Neg Pred Value       0.83777329 0.84738956 0.87380640 0.87070439 0.9964539 0.8799160
Precision            0.60861759 0.73996176 0.83181818 0.73841962 0.9991715 0.7785059
Recall               0.27741408 0.31643500 0.44889616 0.44317253 0.9860998 0.4856909
F1                   0.38111298 0.44329897 0.58311206 0.55390904 0.9925926 0.5981873
Prevalence           0.20366667 0.20379937 0.20379937 0.20379937 0.2037994 0.2037994
Detection Rate       0.05650000 0.06448925 0.09148475 0.09031828 0.2009665 0.0989835
Detection Prevalence 0.09283333 0.08715214 0.10998167 0.12231295 0.2011331 0.1271455
Balanced Accuracy    0.61589415 0.64398560 0.71283234 0.70149417 0.9929452 0.7251602
```

Table 5. Performance measures (Validation data set)



```
                     valid_nb  valid_knn  valid_svm   valid_dt   valid_rf  valid_ann
Sensitivity          0.2748466 0.22358722 0.41400491 0.44594595 0.46560197 0.45454545
Specificity          0.9525903 0.96514914 0.98649922 0.95918367 0.96483516 0.96420722
Pos Pred Value       0.5973333 0.62116041 0.88684211 0.73630832 0.77189409 0.76446281
Neg Pred Value       0.8369655 0.82946573 0.86819563 0.87136338 0.87599772 0.87368421
Precision            0.5973333 0.62116041 0.88684211 0.73630832 0.77189409 0.76446281
Recall               0.2748466 0.22358722 0.41400491 0.44594595 0.46560197 0.45454545
F1                   0.3764706 0.32881662 0.56448911 0.55547054 0.58084291 0.57010786
Prevalence           0.2037500 0.20355089 0.20355089 0.20355089 0.20355089 0.20355089
Detection Rate       0.0560000 0.04551138 0.08427107 0.09077269 0.09477369 0.09252313
Detection Prevalence 0.0937500 0.07326832 0.09502376 0.12328082 0.12278070 0.12103026
Balanced Accuracy    0.6137184 0.59436818 0.70025206 0.70256481 0.71521857 0.70937634
```

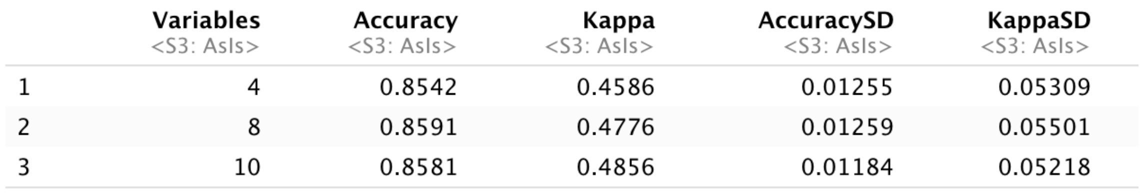
Table 6. AUC of all models

Model	**AUC**
Naïve Bayes	0.6137184
K-NN	0.5943682
SVM	0.7002521
Decision tree	0.7025648
Random Forest	0.7152186
Neural Network	0.7093763

From results in all Tables 4-6, we see that Naïve Bayes and K-NN have the lowest performance measures for both accuracy and AUC, SVM had really slow speed in training step, and best performance are for Random Forest and Neural Network models. So we will continue working on these two models and do further analysis on them.

Applying feature selection method, Recursive feature elimination, we see the model suggests to keep all the variables but to test that affirmation we are going to select just the first 5 features and see how this is going to affect in the model's performance, and then again we did new evaluation of two selected models (ANN and Random Forest).
Table 5, 6, and 7, show performance of models after feature selection.
As for Neural Network, we notice that obtaining top 5 predictors, they would change the results slightly positive and slightly negative for random forest. So, we can a simpler model while having almost the same predictions, and we can see that still we have overfitting in random forest model.

Table 7. Feature selection



	Variables	Accuracy	Kappa	AccuracySD	KappaSD
	<S3: AsIs>	<S3: AsIs>	<S3: AsIs>	<S3: AsIs>	<S3: AsIs>
1	4	0.8542	0.4586	0.01255	0.05309
2	8	0.8591	0.4776	0.01259	0.05501
3	10	0.8581	0.4856	0.01184	0.05218

b. Covid Vaccine

The multivariate linear regression summary shows us none of the variables are statistically significant based on the P-values of Table 8, as we can see the only international trade feature shows some level of significance(0.08%).

Table 8. multivariate linear regression summary

	Estimate Std.	Error	t value	Pr(>|t|)
(Intercept)	1.225e+07	1.936e+07	0.633	0.5291
pop_density_km	-1.565e+02	4.479e+02	-0.349	0.7279
sex_ratio_p100	-2.110e+04	7.881e+04	-0.268	0.7898
GDP_capita	5.931e+01	8.047e+01	0.737	0.4638
intern_trade	-2.051e+01	1.155e+01	-1.775	0.0806 .
urban_pop	-3.469e+04	6.362e+04	-0.545	0.5874
pop_age_b14	-2.982e+05	3.385e+05	-0.881	0.3816
pop_age_a60	-2.756e+05	4.238e+05	-0.650	0.5178
health_expend	1.006e+06	6.297e+05	1.598	0.1150
health_phys	-2.304e+06	1.461e+06	-1.577	0.1197
ed_ter_avg	6.179e+04	6.791e+04	0.910	0.3663

For second degree polynomial regression, using the Q-Q plot of Figure 4a, we can see almost all points fall close the line and are within the confidence envelope, but if we go to analize what is happening with point 37 we find out that it represents Andorra where the actual number of vaccinated people is 3650 but the model predicts 598533.5, again the error is too high and suggest the model is not the correct one.

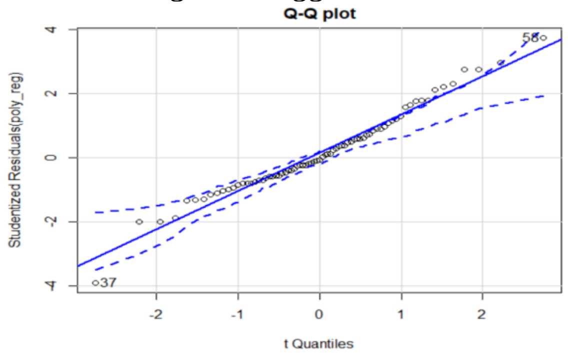


Figure 4a. Q-Q plot for residuals of polynomial regression

The following attempts to use regression are a third degree polynomial regression but its RMSE value is 41527054 and the next attempt of a second degree polynomial regression with an exponent of 0.67 on the outcome variable has a RMSE 91405.29, as we can see those values are very high showing that a regression is not the correct algorithm for the task. A last try to find a better performance was using a kNN regression where we can see the correlation of the predicted values with the valid dataset is of 80% but the RMSE value is still high with 1356367, at this point we can be sure the problem is strictly related with the dataset and not the selection of the algorithm.

V. Discussion and Recommendation

Among all six machine learning models developed in this study, Neural Network had the best overall performance, and there was not underfitting and overfitting concerns regarding this model. Taking everything into consideration, age, number of products, being an active member, balance and location of bank are those attributes a bank can consider in predicting customer behavior in churning a bank. As we see younger customers will stay more with the bank, so if bank has a plan to affect loyalty of customers, best approach is to invest on older age people. We also noticed, having one or wo account is a positive feature in remaining in bank, but having more may result in churning. Balance was not an impactful feature however with regards to old people it is of importance and for people above 61 balance of less than 32K can result in churning.

Suggestion: data were collected from European banks, and other geographical, cultural and etc., can affect these behaviors of churning, thus generalizing this study for all countries is not a good decision. Data can be improved by collecting similar attributes and response value from all over the world and result then can be used in prediction marketing for all banks as general. Future studies can include PCA and logistic regression to have a better understanding of data.

As we can see, the problem with covid vaccine is in the data itself, as the original data included multiples registers with timestamps, the best approach to get the number of vaccinated people may be to consider a forecast problem instead. Also, the data was summarized from thousands of registers for each country to one register per country that reduced a lot the data quality.

VI. Summary

In this study we had the opportunity two work on two datasets, one for prediction and another one for classification. As the data quality for the prediction problem was not good enough for a regression we decided to move on to the classification problem were Random Forest algorithm had the best performance for the task.

Refrences

[1] "5 Ways Banks Can Retain Customers." https://www.xerago.com/blog/5-ways-banks-can-retain-customers/ (accessed Apr. 12, 2021).

[2] "Bank Turnover Dataset | Kaggle." https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling (accessed Apr. 12, 2021).

[3] B. Otieno Ouma *et al.*, "Organizational Customers' Retention Strategies on Customer Satisfaction: Case of Equity Bank Thika Branch, Kenya," Online, 2013. Accessed: Apr. 12, 2021. [Online]. Available: www.iiste.org.

Appendix: R Code for use case study

- **Common packages**

```
library(corrplot)
library(e1071) #naive bayes
library("caTools")
library(dplyr)
library(caret)
library(tidyr)
library("forecast")
library("countrycode")
library("naniar")
library(ggplot2)
library(fable)
library(gains)
library(car)
library(carData)
library(gvlma)
library(class)
library(rpart)
library(rpart.plot)
library(randomForest)
library(nnet)
library('ROCR')
```

- **Bank Churning**

~~~
```{r}
# Importing dataset
data <- read.csv(file = "country_vaccinations.csv")
head(data)
#drop redundant data related to time series, sources, daily reports.
data.df<- data[,c(1,2,4:6,13)]
head(data.df)
~~~

```{r}
## To find number of people vaccinated we had to write a code, since some of them got two shots and some one shot of other kind of vaccine . and we needed to know number of people who were vaciinated.
## because we had the cumulative number we grouped each row by country, we got maximum for all people vaccinated, and we deduct it from people who got their second dose (fully-vaccinated), this way we had number of people were vaccinated in each country.

data.df=data.df %>% mutate_all(funs(replace_na(.,0)))
temp_max= data.df %>% group_by(country) %>% slice_max(total_vaccinations, with_ties = FALSE)
%>% select(country,total_vaccinations)
temp_max2=data.df %>% group_by(country) %>%
slice_max(people_fully_vaccinated, with_ties = FALSE) %>% select(people_fully_vaccinated)

temp_max$people_fully_vaccinated=temp_max2$people_fully_vaccinated
temp_max$no_people_vaccinated=                    temp_max$total_vaccinations                    -
temp_max$people_fully_vaccinated
```

## Merge country statistics data
```{r}
temp_max$country_code<- countrycode(sourcevar=temp_max$country,
                        origin="country.name",
                        destination="iso3c")

statistis.df<-read.csv("country_profile_variables.csv")
statistis.final<-statistis.df[,c(1,3,5,6,9,22,25,29,33,34,38)]
```

~~~
statistis.final$country_code<- countrycode(sourcevar=statistis.final$country,
                              origin="country.name",
                              destination="iso3c")

covid.df<-merge.data.frame(temp_max,statistis.final[,-1],all.x = TRUE)
summary(covid.df)
#Since these countries were already accounted for in UK data we decided to drop them
covid.df<-covid.df[-c(65,133:136),]
```

### The first step is going to be to update the columns names to make it easier to deal with the dataframe.

```{r}
colnames(covid.df)[6]<-"surface"
colnames(covid.df)[7]<-"pop_density_km"
colnames(covid.df)[8]<-"sex_ratio_p100"
colnames(covid.df)[9]<-"GDP_capita"
colnames(covid.df)[10]<-"intern_trade"
colnames(covid.df)[11]<-"urban_pop"
colnames(covid.df)[12]<-"pop_age"
colnames(covid.df)[13]<-"health_expend"
colnames(covid.df)[14]<-"health_phys"
colnames(covid.df)[15]<-"ed_tert"
```

### Add missing values for Cote d'Ivoire record using data extracted from the following sources:
#### https://www.indexmundi.com/cote_d_ivoire/demographics_profile.html
#### https://data.worldbank.org/country/cote-divoire

```{r}
covid.df$surface[covid.df$country_code=='CIV']<-322463
covid.df$pop_density_km[covid.df$country_code=='CIV']<-63.9
covid.df$sex_ratio_p100[covid.df$country_code=='CIV']<-101.7
covid.df$GDP_capita[covid.df$country_code=='CIV']<-2030
covid.df$intern_trade[covid.df$country_code=='CIV']<-1557.18
covid.df$urban_pop[covid.df$country_code=='CIV']<-50.78
covid.df$pop_age[covid.df$country_code=='CIV']<-'42.17/2.87'
covid.df$health_expend[covid.df$country_code=='CIV']<-4.45
```
### Drop territory information for Guernsey record

```{r}
covid.df<-covid.df[-c(45),]
```
## Visualize all missing data:
```{r}
miss_val<- sapply(covid.df, function(x) sum(is.na(x))/length(x)*100)
barplot(miss_val,
    main = "Percentage of missing values per feature",
    ylab = "Percentage of missing values",
    space = 0.5,
    cex.names=.7,
~~~

~~~~
      las=2,
      col = c(rgb(0.6,0.1,0.4,0.3) ,
            rgb(0.3,0.5,0.4,0.2) ,
            rgb(0.4,0.4,0.4,0.4) ,
            rgb(0.3,0.7,0.4,0.8)))
```
```{r}
summary(covid.df)
#some discoveries: missing value is presented as -99 value in numeric columns, and some other
columns with numbers are considered characters,
#Tasks:
#find the number of -99 values per column,
#convert numeric values in characters as numbers again
```
```{r}
covid.df.temp <- covid.df %>% replace_with_na_all(condition = ~.x == -99 )

covid.df.temp <- covid.df.temp %>% separate(pop_age, c("pop_age_b14","pop_age_a60"), sep = "/")
covid.df.temp <- covid.df.temp %>% separate(ed_tert, c("ed_tert_f","ed_tert_m"), sep = "/")
covid.df.temp<-transform(covid.df.temp, ed_tert_f = as.numeric(ed_tert_f),
                  ed_tert_m = as.numeric(ed_tert_m),
                  surface = as.numeric(surface),
                  intern_trade = as.numeric(intern_trade),
                  pop_age_b14 = as.numeric(pop_age_b14),
                  pop_age_a60 = as.numeric(pop_age_a60),
                  health_phys = as.numeric(health_phys))

covid.df.temp$ed_ter_avg<-rowMeans(covid.df.temp[ , c("ed_tert_f","ed_tert_m")], na.rm=TRUE)
#drop unnecessary columns
covid.df.temp$ed_tert_f<- NULL
covid.df.temp$ed_tert_m<- NULL

#Droppping more territories found during exploration
covid.df.temp<-covid.df.temp[-c(41,43,46,57,110),]
summary(covid.df.temp)

```
### Check again for missing values:
```{r}
miss_val_temp<- sapply(covid.df.temp, function(x) sum(is.na(x)))
barplot(miss_val_temp,
      main = "Number of missing values per feature",
      ylab = "Number of missing values",
      space = 0.5,
      cex.names=.7,
      las=2,
      col = c(rgb(0.6,0.1,0.4,0.3) ,
            rgb(0.3,0.5,0.4,0.2) ,
            rgb(0.4,0.4,0.4,0.4) ,
            rgb(0.3,0.7,0.4,0.8)))

```
~~~~

~~~
## Taking care of missing data for population age below 14 ,and population age above 60 by replacing
them by mean of data in correspondence column.
```{r}
summary(covid.df.temp)

covid.df.temp$pop_age_b14 = ifelse(is.na(covid.df.temp$pop_age_b14),
        ave(covid.df.temp$pop_age_b14, FUN = function(x) mean(x , na.rm=TRUE)),
        covid.df.temp$pop_age_b14)

covid.df.temp$pop_age_a60 = ifelse(is.na(covid.df.temp$pop_age_a60),
        ave(covid.df.temp$pop_age_a60, FUN = function(x) mean(x , na.rm=TRUE)),
        covid.df.temp$pop_age_a60)

```
#### Getting correlation of data to see if there is any relationship (correlation) between variables, so
we can predict them based on each other.
```{r}
corrplot.mixed(cor(covid.df.temp[,-c(1,2)],use="complete.obs"),tl.col='black')
```

### Fixing missing values for health_phys through using regression
#### as we see in correlation table there is correlation between health physician numbers and
population age above 60, total health expenditure and population age above 60, and also between
education tertiary gross enrollment and population above 60 age. so we fit a regression model, and
based on their p value and r squared we decided that we can use this model, and predict missing values
in these three variables (health physician number,total health expenditure, and education tertiary
gross enrollment) based on population age above 60.
```{r}
#Subset all records without missing values for column health_phys
cv.df.nan<-subset(covid.df.temp,(!is.na(covid.df.temp[,"health_phys"])))
#subset all records with missing values for column health_phys
cv.df.miss<-subset(covid.df.temp,(is.na(covid.df.temp[,"health_phys"])))

linearMod <- lm(health_phys ~ pop_age_a60, data=cv.df.nan)
summary(linearMod)
new=data.frame(pop_age_a60=cv.df.miss$pop_age_a60)
fitted.df=predict(linearMod, new, interval="prediction")
cv.df.fitted=cv.df.miss
cv.df.fitted$health_phys=as.data.frame(fitted.df)$fit
```

```{r}
cv.df.t<- rbind(cv.df.fitted,cv.df.nan)
dim(cv.df.t)
```

### fix missing values for health_expend
```{r}
#Subset all records without missing values for column health_expend
cv.df.comp<-subset(cv.df.t,(!is.na(cv.df.t[,"health_expend"])))
#subset all records with missing values for column health_expend
cv.df.nanp<-subset(cv.df.t,(is.na(cv.df.t[,"health_expend"])))
~~~

~~~
linearM <- lm(health_expend ~ pop_age_a60, data=cv.df.comp)
summary(linearM)
new=data.frame(pop_age_a60=cv.df.nanp$pop_age_a60)
fitted.df=predict(linearM, new, interval="prediction")
cv.df.fitted=cv.df.nanp
cv.df.fitted$health_expend=as.data.frame(fitted.df)$fit

```

```{r}
cv.df.t<- rbind(cv.df.fitted,cv.df.comp)
dim(cv.df.t)
```

### Fix missing data for international trading by replacing by mean of the column
```{r}

cv.df.t$intern_trade = ifelse(is.na(cv.df.t$intern_trade),
            ave(cv.df.t$intern_trade, FUN = function(x) mean(x , na.rm=TRUE)),
            cv.df.t$intern_trade)
```

### fix missing values for education terciary
```{r}
#Subset all records without missing values for column ed_ter_avg
cv.df.c<-subset(cv.df.t,(!is.na(cv.df.t[,"ed_ter_avg"])))
#subset all records with missing values for column ed_ter_avg
cv.df.nan<-subset(cv.df.t,(is.na(cv.df.t[,"ed_ter_avg"])))

linearM <- lm(ed_ter_avg ~ pop_age_a60, data=cv.df.c)
summary(linearM)
new=data.frame(pop_age_a60=cv.df.nan$pop_age_a60)
fitted.df=predict(linearM, new, interval="prediction")
cv.df.fitted=cv.df.nan
cv.df.fitted$ed_ter_avg=as.data.frame(fitted.df)$fit
```

```{r}
cv.df.t<- rbind(cv.df.fitted,cv.df.c)
dim(cv.df.t)
```

## Encoding cattegorical data
```{r}
#data$x= factor(data$x, levels = c("","") , labels = c())
```

```{r}
set.seed(11)
head(cv.df.t)
cv.df.a = cv.df.t
selected.var <- c(5,7:16)
train.index <- sample(c(1:125), .6*125)
~~~

~~~
train <- cv.df.a[train.index, selected.var]
valid <- cv.df.a[-train.index, selected.var]

```
```{r}
corrplot.mixed(cor(cv.df.a[,selected.var],use="complete.obs"),tl.col='black')
```

## Model 1: Multiple Linear regression
```{r}
mult_reg= lm(formula =no_people_vaccinated~.,
           data = train)
summary(mult_reg)
```
Regress

### As we see none of our variables are not statistically signifacnt in this model(based on their P values),
and there is not linear relationship here,in other words , our P values are not significant meaning that
our dependent variables in this model have low impact on our dpendant variable (assuming tresh hold
of 5% for P value), looking at multiple R squared we can also confirm that this model is not a good
model to fit. For internatioanl trade variable, P value ( 0.08%), shows it might have a certain level of
significance but not that much.

## Model 2: Polynomial Linear regression
```{r}

poly_reg=                                                                          lm(formula
=no_people_vaccinated~(pop_density_km+sex_ratio_p100+GDP_capita+intern_trade+urban_pop+p
op_age_a60+pop_age_b14+health_expend+health_phys+ed_ter_avg)^2,
           data = train)
summary(poly_reg)
```
###  pop_density_km:intern_trade(***,  P_value=7.62e-05),  pop_density_km:health_expend  (**,
P_value     =     0.00575),     pop_density_km:pop_age_a60     (*,     P_value=0.01195)     ,
pop_density_km:health_phys (*, P_value =0.02406 ) , pop_age_b14:ed_ter_avg (*, P_value =0.03103),
these variables and their combination has most significant impact on model and on our prediction
based on their P values.

```{r}
poly_predi= predict(poly_reg, newdata = valid)
poly_reg
some.residuals = valid$no_people_vaccinated -poly_predi
data.frame("Predicted"=poly_predi,                          "Actual"=valid$no_people_vaccinated,
"Residuals"=some.residuals)
```
#### As we see mpoly nomial model is not really a good option

## Visualize the performance of our polynomial model

### Show histogram and boxplot of errors
```{r}
hist(some.residuals)
boxplot(some.residuals)
~~~

~~~~
```
#### as we see we have plenty of outliers.

### Checking accuracy of model
```{r}
accuracy(poly_predi, valid$no_people_vaccinated)
```
#### As we see here numbers are very high, and it is not really a good model.

### Checking correlation of predicted and actual variable
```{r}
cor(poly_predi,valid$no_people_vaccinated)
```
#### As we see here correlation is very low, another proof of not having a good model.

### Lift chart
```{r}
gain= gains(valid$no_people_vaccinated[!is.na(poly_predi)],poly_predi[!is.na(poly_predi)])
cum_no_peop_vaccinated = valid$no_people_vaccinated[!is.na(poly_predi)]
par(pty="s")
plot(c(0,gain$cume.pct.of.total*sum(valid$no_people_vaccinated))~c(0,gain$cume.obs),        xlab="#
Cases",ylab = "Cumulative number of people vaccinated", main="Lift chart", type= "l")
lines(c(0,sum(valid$no_people_vaccinated))~c(0,dim(valid)[50]), col="red", lty=2)

```


### Decile wise lift chart
```{r}

#barplot(gain$mean(valid$no_people_vaccinated),   names.arg=   gain$depth,   xlab=   #"Percentile",
ylab="Mean response" , main="Decile_wise")
```
###
```{r}
cumsum(valid$no_people_vaccinated[order(poly_predi)])
```
#### There are methods we use to evaluate the appropriateness of the regression model. Following
we want to evaluate the model and see if we can uncover the problem with regression model and see
if it is appropriate to use here for this problem.

### Normality

```{r}
qqPlot(poly_reg, labeles= row.names(train), id.method= "identify", simulate= TRUE , main="Q-Q plot")

```
#### With the exception of points (58,and 37), all the points fall close to the line and are within the
confidence envelope, suggesting that we've met the normality assumption fairly well. But for point 37
we should definitely look at that. It has a large negative residual (actual-predicted), indicating that the
model overestimates the number of people vaccinated at this country.
```{r}
train[37,] ## number 4 is Andorra, no_people_vaccinated = 3650
fitted(poly_reg)[37] ## 598533.5
~~~~

```
residuals(poly_reg)[37] ## -594883.5
rstudent(poly_reg)[37] ## -0.5809694
```

#### Here we see that number of people vaccinated is 3650 , but the model predicts 598533.5 number of people vaccinated!!!!

```{r}
residplot <- function(poly_reg, nbreaks=10){
        z <- rstudent(poly_reg)
        hist(z, breaks=nbreaks, freq=FALSE,
        xlab="Studentized Residual",
        main="Distribution of studentized residuals using the residplot function")
        rug(jitter(z), col="brown")
        curve(dnorm(x, mean=mean(z), sd=sd(z)),
            add=TRUE, col="blue", lwd=2)
        lines(density(z)$x, density(z)$y,
        col="red", lwd=2, lty=2)
        legend ( "topright" ,
            legend = c ( "Normal Curve", "Kernel Density Curve"),
            lty=1:2, col=c("blue", "red"), cex=.7)
    }
residplot(poly_reg)
```

#### As we see here, the errors are not following a normal distribution quiet well.

### Linearity:

#### using component plus plots (partial residual plots), we can look for evidence of nonlinearity in relationship between the dependent variable and the independennt variables. we are actually, looking for any systematic departure from the linear model that we specified.Nonlinearity in any of these plots suggests that you may not have adequately modeled the functional form of that predictor in the regression.

```{r}
library(car)

crPlots(mult_reg)
```

### Seeing results of this plot we can see that, for pop_density, international trade, sex ratio per 100 there is clear deviation from linearity and we need to take care of data, either by transforming data or changing to another model.

## Model 2- modified: Based on first poly nomial model variables with high P values are chosen and another polynomial model is modeled

```{r}
head(cv.df.a)
poly2_reg= lm(formula
=no_people_vaccinated~(pop_density_km+intern_trade+pop_age_b14+pop_age_a60+health_expen
d+health_phys+ed_ter_avg)^3,
       data = train)
summary(poly2_reg)
```

```{r}
poly2_predi= predict(poly2_reg, newdata = valid)
```

~~~~
some.residuals = valid$no_people_vaccinated -poly2_predi
data.frame("Predicted"=poly2_predi, "Actual"=valid$no_people_vaccinated,
"Residuals"=some.residuals)
RMSE(poly2_predi,valid$no_people_vaccinated)
```
#### As we see mpoly nomial model is not really a good option.

## Homoscedasticty
#### In order to identify non-constant error variance and check if we met the constant variance
assumption, we perform following tests:
#### for polynomiaal model:
```{r}
ncvTest(poly_reg)
spreadLevelPlot(poly_reg)
```
#### As we see the score test is nonsignificant (p=0.61), suggesting that we have met the constant
variance assumption.However, the line is not horizontal, suggesting we kind of violating this
assumption.suggested transformation power is 0.67.
```{r}
poly_reg03= lm(formula
=no_people_vaccinated^0.67~(pop_density_km+sex_ratio_p100+GDP_capita+intern_trade+urban_p
op+pop_age_a60+pop_age_b14+health_expend+health_phys+ed_ter_avg)^2,
        data = train)
summary(poly_reg03)
```
```{r}
poly_predi03= predict(poly_reg03, newdata = valid)

some.residuals = valid$no_people_vaccinated -poly_predi03
data.frame("Predicted"=poly_predi03, "Actual"=valid$no_people_vaccinated,
"Residuals"=some.residuals)
RMSE(poly_predi03,valid$no_people_vaccinated)
```
#### for polynomiaal model2:
```{r}
ncvTest(poly_reg03)
spreadLevelPlot(poly_reg03)
```

#### for polynomiaal model2:
```{r}
ncvTest(poly2_reg)
spreadLevelPlot(poly2_reg)
```

### Multicoliniarity
```{r}
vif(mult_reg)

sqrt(vif(mult_reg))>2 #Problem?
```
#### As we see we have multicoliniarity for all predictors
~~~~

~~~~
## Visualize the performance of our polynomial model

### Show histogram and boxplot of errors
```{r}
hist(some.residuals)
boxplot(some.residuals)
```

## Unusual observations
##Outliers

```{r}
outlierTest(poly_reg)
```
### Here we identify that ID(122), which is for ???? is an outlier(P=0.01).

```{r}
train[122,] ## number 122 is
fitted(poly_reg)[122] ##
residuals(poly_reg)[122] ##
rstudent(poly_reg)[122] ##
```

## High leverage
```{r}
hat.plot <- function(poly_reg) {
	p <- length(coefficients(poly_reg))
	n <- length(fitted(poly_reg))
	plot(hatvalues(poly_reg), main="Index Plot of Hat Values")
	abline(h=c(2,3)*p/n, col="red", lty=2)
	identify(1:n, hatvalues(poly_reg),
		names(hatvalues(poly_reg) ))
	}
hat.plot(poly_reg)
```
## Influential observations
```{r}
cutoff <- 4/(nrow(train)-length(poly_reg$coefficients)-2)
plot(poly_reg, which=4, cook.levels=cutoff)
abline(h=cutoff, lty=2, col="red")
```
#### The graph identifies 126, 80, and 78 as influential observations. Deleting these states will have a
notable impact on the values of the intercept and slopes in the regression model.

## Added variabe plots
```{r}
influencePlot(poly_reg, main="Influence Plot", sub="Circle size is proportional to Cook's distance")
```
### This plot shows,

### scaling datas (except those factored ones - we don't have )
```{r}
~~~~

~~~
train.scaled <- scale(train[,c()])
valid.scaled <- scale(valid[,c()])
```
### The resulting plot shows that points 122, and 37 are outliers, and points 80,78, and 126 are influential.and all these points are high leverage based on definition for this graph.

# K-NN

```{r}
Covid.knn <- knnreg(no_people_vaccinated~., data = train, k = 11)
Covid.knn.predict = predict(Covid.knn , valid)
accuracy(valid$no_people_vaccinated, Covid.knn.predict)
```
```{r}
cor(valid$no_people_vaccinated, Covid.knn.predict)
```

### This correlation number is not a great number , but it is ok, and to imptove it we can play with k value.
```{r}
Covid.knn2 <- knnreg(no_people_vaccinated~., data = train, k = 5)
Covid.knn.predict2 = predict(Covid.knn2 , valid)
accuracy(valid$no_people_vaccinated, Covid.knn.predict2)
cor(valid$no_people_vaccinated, Covid.knn.predict2)
```
### This improved our correlation number from 69% to 80%.
### ***There is an idea that if we standardize variables we might get even more improved model.

## Standardizing variables
```{r}
class(cv.df.a)
library(dplyr)
cv.df.scaled=cv.df.a %>% mutate_at(c(3:16), funs(c(scale(.))))
set.seed(11)
selected.var <- c(5,7:16)
train.index <- sample(c(1:125), .6*125)
train2 <- cv.df.a[train.index, selected.var]
valid2 <- cv.df.a[-train.index, selected.var]
```
```{r}
Covid.knn3 <- knnreg(no_people_vaccinated~., data = train2, k = 5) # fitting knn to trainin data sets
Covid.knn.predict3 = predict(Covid.knn3 , valid2) #predicting the test set results
accuracy(valid2$no_people_vaccinated, Covid.knn.predict3)
cor(valid2$no_people_vaccinated, Covid.knn.predict3)
RMSE(Covid.knn.predict3,valid2$no_people_vaccinated)
```
~~~

- **Covid vaccine**

~~~
```{r}
Bank.df= read.csv("Churn_Modelling.csv") ## 1000 Observations and 14 VAriables we have
```
~~~

~~~
```{r}
str(Bank.df)
## Response variable is "Exited"
```

### Cleaning data
### As 'RowNumber', 'CustomerId' and 'Surname' are not needed and are extra infromation, we
decided to remove them.
```{r}
Bank.df = Bank.df[,-c(1,2,3)]
head(Bank.df)
```



```{r}
summary(Bank.df)
```

## Visualize all missing data:
```{r}
miss_val_Bank<- sapply(Bank.df, function(x) sum(is.na(x))/length(x)*100)
barplot(miss_val_Bank,
      main = "Number of missing values per feature",
      ylab = "Number of missing values",
      space = 0.5,
      cex.names=.7, las=2,
      col = c(rgb(0.6,0.1,0.4,0.3) , rgb(0.3,0.5,0.4,0.2) , rgb(0.4,0.4,0.4,0.4) , rgb(0.3,0.7,0.4,0.8)))
```
## The diagram shows we don't have any missing values.

## We have "Exited" value as "0" and "1" responses, however, we prefer something more clear , so we
converted "0" to "Stayed" , and "1" to "Left".

```{r}
Bank = Bank.df
Bank$Exited[Bank$Exited==0] = "Stayed"
Bank$Exited[Bank$Exited==1]= "Left"
Bank$Exited<-as.factor(Bank$Exited)
summary(Bank)
```
### As we see we now have :  Left=2037  and Stayed=7963 , here we have those who left are in minority
and this brings bias in prediction against their favor so we need to address this imbalancess data.

## Data Visualization
### credit score
```{r}
hist(Bank$CreditScore,
    main="Credit Score Histogram",  xlab="Crdit Score",  col="#e9ecaa")
```
### As we see , majority of customers has score between 600 and 700,and we can say the distribution
is fairly normal

## Age
~~~

~~~
```{r}
hist(Bank$Age,
    main="Credit Score Histogram",
    xlab="Age", col="#e9ecaa")
```
### Majority of customers are around 40, and we have light right skewness.

## Balance
```{r}
hist(Bank$Balance,
    main="Credit Score Histogram", xlab="Balance", col="#e9ecaa")
```
### We do not have a normal distribution, and majority is 0.

## Estimated Salary
```{r}
hist(Bank$EstimatedSalary,
    main="Credit Score Histogram", xlab="Balance", col="#e9ecaa")
```
### Our distribution here is not normal, and it is infact uniform.

## Detecting outliers
### CreditScore
```{r}

boxplot(Bank$CreditScore~Bank$Exited,
      main="Credit Score Box Plot",
      xlab='', ylab='Credit score', col=cm.colors(2))
```
## For credit score we see a few outliers in the class of left customers, and we can see numbers below:
```{r}
boxplot.stats(Bank$CreditScore[Bank$Exited=='Left'])$out
```

```{r}
summary(Bank$CreditScore)
```

## Age
```{r}

boxplot(Bank$Age~Bank$Exited,
      main="Age Box Plot",
      xlab='',
      ylab='Credit score',
      col=cm.colors(2))
```
### We have outliers both in "Left" and "Stayed" class for "Age" variable.
## Outliers in "Left" class:
```{r}
boxplot.stats(Bank$Age[Bank$Exited=='Left'])$out
```
~~~

~~~
## Outliers in "Stayed" class:
```{r}
boxplot.stats(Bank$Age[Bank$Exited=='Stayed'])$out
```

### We have 486 outliers in "Stayed" class in fo "Age" variable with Min = 63 Mean = 69.27   Max =92
for outliers (Taking a summary).

## Balance
```{r}
boxplot(Bank$Balance~Bank$Exited,
      main="Age Box Plot",
      xlab='',
      ylab='Credit score',
      col=cm.colors(2))
```

### There is no outliers in "Balance" variable.

## Estimated Salary
```{r}
boxplot(Bank$EstimatedSalary~Bank$Exited,
      main="Estimated Salary Box Plot",
      xlab='', ylab='Credit score', col=cm.colors(2))
```

### There is no outlier for "Estimated Salary" as well.

## In all data we do not have significant number of outliers compared to whole size of the data, so we
will continue with data as it is.

### For "Tenure" and "Number of Produxts" variables, as we have a few integer values , we will use bar
plots instead of histogram.

## Tenure
```{r}
barplot(table(Bank$Tenure), main="Tenure Bar Plot", col='#97da9d', xlab='Tenure(Year)')
```

## Here we have a ymmetric distribution which has the lowest values at both ends, which is associated
with deposits tenured in less than a year or 10 years.

## Number of Products
```{r}
barplot(table(Bank$NumOfProducts), main="Number of Products Bar Plot", col='#97da9d',
xlab='Number of Products')
```

### Most of customers prefers to have up to two products.
## For vategorical data and the dependency of response variable to them, we decided to perform Chi-
squared test.

## Gender
~~~

```{r}
barplot(table(Bank$Exited, Bank$Gender), beside = TRUE,
 legend.text = levels(Bank$Exited),main="Churning Behaviour across Males and Females", col=c("#CCFFFF","#FF99CC"))
```

### For both men and women , we have the majority deciding to "Stay", and data are balance, but in women group we see ratio of "Stayed" to "Left" is more than men.

```{r}
chisq.test(table(Bank$Gender, Bank$Exited))
```

### Confirming what we mentioned above, chi-squared ratio shows we have dependency on gender, as for deciding to stay or leave.

## Countries

```{r}
barplot(table(Bank$Exited, Bank$Geography), beside = TRUE,
      legend.text = levels(Bank$Exited), main="Churning Behaviour across Countries", col=c("#CCFFFF","#FF99CC"))
```

### In all three countries, they prefered to "Stay" with bank. We can say there is a good balance of ratios of stay to leave in Germany and Spain, but in France we have more records but the behaviour is similar to Spain and Germany.

```{r}
chisq.test(table(Bank$Geography, Bank$Exited))
```

## Converting integer values to factors for "HasCrCard" and "IsActiveMember" (having a better visual data)

```{r}
Bank2<-Bank
#HasCrCard
Bank2$HasCrCard[Bank2$HasCrCard==0]<-'W/O CrCard'
Bank2$HasCrCard[Bank2$HasCrCard==1]<-'W CrCard'
Bank2$HasCrCard<-as.factor(Bank2$HasCrCard)
#IsActiveMember
Bank2$IsActiveMember[Bank2$IsActiveMember==0]<-'Not Active Member'
Bank2$IsActiveMember[Bank2$IsActiveMember==1]<-'Active Member'
Bank2$IsActiveMember<-as.factor(Bank2$IsActiveMember)
```

## Having a Credit Card

```{r}
barplot(table(Bank2$Exited, Bank2$HasCrCard), beside = TRUE,
      legend.text = levels(Bank2$Exited),
      main="Churning Behaviour vs. Having a Credit Card",
      col=c("#CCFFFF","#FF99CC"))
```

### In both groups we see majority of customers will stay with bank , and behaviour and ratio is similar.

~~~
```{r}
chisq.test(table(Bank$HasCrCard, Bank$Exited))
```

### From P value, we can say that, just having a credit card does not say anything about leaving or holding to bank.

## Being An Active Member
```{r}
barplot(table(Bank2$Exited, Bank2$IsActiveMember), beside = TRUE,
      legend.text = levels(Bank2$Exited),
      main="Churning Behaviour vs. Having a Credit Card", col=c("#CCFFFF","#FF99CC"))
```
### For both groups they mostly will "Stay", but ratio of leabbing is more for Non Active Members.

```{r}
chisq.test(table(Bank$IsActiveMember, Bank$Exited))
```
### The P value is signifacnt, and then there is a relation between being or not being an Active member, and stay or leave the bank.

## Number of products

### See the distribution of outcomes as the numerical values have only 4 possible values.
```{r}
barplot(table(Bank2$Exited, Bank2$NumOfProducts), beside = TRUE,
      legend.text = levels(Bank2$Exited),
      main="Churning Behaviour across Number of Products",
      xlab='Number of Products', col=c("#CCFFFF","#FF99CC"))
```

## The plot show that most customers with one or two products stayed with the bank but customers with more products decided to left.

```{r}
chisq.test(table(as.factor(
  Bank$NumOfProducts), Bank$Exited))
```
## Again, the p value is significant and we can say that there is a relation between the number of products a customer has and the decision to leave of stay at the bank.

##  Correlation between numeric variables
```{r}
Bank3 <- Bank.df[,c('Exited','CreditScore','Age','Tenure','Balance', 'NumOfProducts', 'EstimatedSalary', 'IsActiveMember', 'HasCrCard')]
corrplot.mixed(cor(Bank3,use="complete.obs"),tl.col='black')
corr<-cor(Bank3)
summary(corr[upper.tri(corr)])
summary(Bank3)
```

### Both chart and summary shows our variables have light correlation with each other; the strongest correlation we have are for Age(0.29) and IsActiveMemeber(-0.16)
~~~

~~~
### Naive Bayes

```{r}
#Normalizing data before
prepData <- preProcess(Bank, method="BoxCox")
BankNB <- predict(prepData,Bank)
summary(BankNB)
```

### Showing the difference between columns and how BoxCox affects the normality of the data, for
example it improved for Estimated Salary but didn't change anything for Balance values.

```{r}
hist(BankNB$Balance,
    main="Transformed Balance Histogram",
    xlab="",
    col="#CCFFFF")

## Estimated Salary
hist(BankNB$EstimatedSalary,
    main="Transformed Estimated Salary Histogram",
    xlab="",
    col="#CCFFFF")
```
### Splitting the dataset

```{r}
set.seed(831)
train.index <- sample(c(1:10000), .6*10000)
train <- BankNB[train.index, ]
valid <- BankNB[-train.index, ]
train[,-11]

NB.Model <- naiveBayes(formula=Exited~.,
                 data=train)

## Training Performance
NB.Train <- predict(object=NB.Model,
                newdata=train[,-11],
                type="class")
NB.Train.Performance<-confusionMatrix(data=NB.Train,
                        reference=train$Exited, positive="Left",
                        mode="prec_recall")
NB.Train.Performance
```
## Validation data set Performance
```{r}

NB.Valid<- predict(object=NB.Model,
                newdata=valid[,-11],
                type="class")
NB.Valid.Performance<-confusionMatrix(data=NB.Valid,
~~~

~~~
                         reference=valid$Exited,  positive="Left",
                         mode="prec_recall")
NB.Valid.Performance
```
### As we can se our Kappa number is 0.2846 which is in range of 0.2-0.4, which we call it as fair. ALso our F1 : 0.37647 which is close to 0, and as a whole the performance of our model is not that good.

## Naive Bayes Lift chart
```{r}
predvec <- ifelse(NB.Valid=="Left", 1, 0)
realvec <- ifelse(valid$Exited=="Left", 1, 0)

# And then a lift chart
gain <- gains(realvec, predvec, groups=10)
plot(c(0,gain$cume.pct.of.total*sum(realvec))~
    c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec))~c(0, dim(valid)[1]), lty=2)
```
## ROC plot and AUC value Naive Bayes

```{r}
pred = prediction(predvec, realvec)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
```


# K-Nearest Neighbours

## Converting categorical variables of Gender and Geography to their associated dummies.
```{r}
dum <- dummyVars(~Gender+Geography, data=Bank,
              sep="_", fullRank = TRUE)
knn1 <- predict(dum, Bank)
```
```{r}
knn2 <- data.frame(Bank[,!names(Bank) %in% c("Gender", "Geography")], knn1)
str(knn2)
```
## We want to use min max normalizaion since we might have a better performace of kNN this way.
```{r}
Knn.norm <- preProcess(x=knn2, method="range")
knn.df <- predict(Knn.norm, knn2)
```
## Splitting data set into training and validation sets
```{r}
~~~

~~~~
set.seed(831)
index1 <- createDataPartition(knn.df$Exited, p=0.60, list=FALSE)
knn.train =knn.df[index1, ]
knn.valid = knn.df[-index1, ]
```
## Inorder to get an estimate of k, we used a grid search with a 10-fold cross validation for three times.

```{r}
grids <- expand.grid(k=seq(from=3,to=20,by=2))
grid.control <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3,
                     search = "grid")

set.seed(831)
knn.fit <- train(form = Exited ~ .,
             data = knn.train,  method = "knn",
             trControl = grid.control,
             tuneGrid = grids)
knn.fit
```

## The Best accuracy (0.8161061) and still a good kappa ( 0.2663960)  was for k = 9.

```{r}
plot(knn.fit)
```
## Evaluating performance of the model (Knn) for training data set
```{r}
knn.pred.train <- predict(knn.fit, newdata=knn.train)
knn.perf.train<-confusionMatrix(data=knn.pred.train,
                         reference=knn.train$Exited,
                         positive="Left",
                         mode="prec_recall")
knn.perf.train
```
## Evaluating performance of the model (Knn) for validation data set

```{r}
knn.pred.valid <- predict(knn.fit, newdata=knn.valid)
knn.perf.valid<-confusionMatrix(data=knn.pred.valid,
                         reference=knn.valid$Exited,
                         positive="Left",
                         mode="prec_recall")
knn.perf.valid
```
### As we can se our Kappa number is 0.2478 which is in range of 0.2-0.4, which we call it as fair. ALso
our F1 : 0.32882 which is close to 0, and as a whole the performance of our model is not that good.

## kNN Lift chart
```{r}
predvec.knn <- ifelse(knn.pred.valid=="Left", 1, 0)
realvec.knn <- ifelse(knn.valid$Exited=="Left", 1, 0)
~~~~

~~~
# And then a lift chart
gain <- gains(realvec.knn, predvec.knn, groups=10)
plot(c(0,gain$cume.pct.of.total*sum(realvec.knn))~
    c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec.knn))~c(0, dim(valid)[1]), lty=2)
```

```{r}
pred = prediction(predvec.knn, realvec.knn)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
```

# SVM Model
## Kernel for training data set
```{r}
set.seed(831)
svm.model <- svm(Exited~.,
             data=knn.train,
             method="C-classification",
             kernel="radial",
             scale=TRUE)

## Performance
svm.train <- predict(svm.model,
                  knn.train[,-9],
                  type="class")
svm.perf.train<-confusionMatrix(svm.train,
                          knn.train$Exited,
                          positive="Left",
                          mode="prec_recall")
svm.perf.train
```

## Kernel for validating data set
```{r}
set.seed(831)
svm.model <- svm(Exited~.,
             data=knn.valid,
             method="C-classification",
             kernel="radial",
             scale=TRUE)

## Performance
~~~

~~~~
svm.valid <- predict(svm.model,
                  knn.valid[,-9],
                  type="class")
svm.perf.valid<-confusionMatrix(svm.valid,
                          knn.valid$Exited,
                          positive="Left",
                          mode="prec_recall")
svm.perf.valid
```

### As we can se our Kappa number is 0.4997 which is in range of 0.4-0.6, which we call it as Moderate.
ALso our F1 : 0.56449 which is close to 0, and as a whole the performance of our model is Moderate.

## SVM Lift chart
```{r}
predvec.svm <- ifelse(svm.valid=="Left", 1, 0)

# And then a lift chart
gain <- gains(realvec.knn, predvec.svm, groups=5)
plot(c(0,gain$cume.pct.of.total*sum(realvec.knn))~
    c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec.knn))~c(0, dim(valid)[1]), lty=2)
```

```{r}
pred = prediction(predvec.svm, realvec.knn)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
```

# Decision tree
## Splitting data without converting any categorical variables to dummies
```{r}
set.seed(831)
index2 <- createDataPartition(Bank$Exited, p=0.60, list=FALSE)
dec.train =Bank[index2, ]
dec.valid = Bank[-index2, ]
```

```{r}
grids2 <- expand.grid(cp = seq(from=0,to=.25,by=.01))

grid.control2 <- trainControl(method = "repeatedcv",
                    number = 10, repeats = 3, search = "grid")
~~~~

```
set.seed(831)

dec.fit <- train(form = Exited ~ .,
            data = dec.train,
            method = "rpart",
            trControl = grid.control2,
            tuneGrid = grids2)
dec.fit
```

## The value to use for cp is 0.2.

```{r}
plot(dec.fit)
```

## getting summary of the best fitted tree
```{r}
dec.fit$finalModel
rpart.plot(dec.fit$finalModel)
```

## Evaluating performance of training data set.
```{r}
dec.pred <- predict(dec.fit, newdata=dec.train)
dec.perf.train<-confusionMatrix(data=dec.pred,
                    reference=dec.train$Exited,
                    positive="Left",
                    mode="prec_recall")
dec.perf.train
```
## Evaluating performance of validation data set.
```{r}
dec.pred <- predict(dec.fit, newdata=dec.valid)
dec.perf.valid<-confusionMatrix(data=dec.pred,
                    reference=dec.valid$Exited,
                    positive="Left",
                    mode="prec_recall")
dec.perf.valid
```
### As we can se our Kappa number is 0.4748 which is in range of 0.4-0.6, which we call it as Moderate. ALso our F1 : 0.55547 which is close to 0, and as a whole the performance of our model is Moderate. So far Decision trees performs better than Naive Bayes and K-NN and svm performed pretty much the same.

## Decision Tree Lift chart
```{r}
predvec.dt <- ifelse(dec.pred=="Left", 1, 0)
realvec.dt <- ifelse(dec.valid$Exited=="Left", 1, 0)

# And then a lift chart
gain <- gains(realvec.dt, predvec.dt, groups=5)
plot(c(0,gain$cume.pct.of.total*sum(realvec.dt))~
   c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec.dt))~c(0, dim(valid)[1]), lty=2)
```

```{r}
pred = prediction(predvec.dt, realvec.dt)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
```

#Random Forest

```{r}
grids.rf <- expand.grid(mtry=seq(from = 1, to = 10, by = 1))
grid.controlrf <- trainControl(method = "repeatedcv",
                        number = 5,
                        repeats = 3,
                        search = "grid")
set.seed(831)
rf.fit <- train(Exited~.,
            data=dec.train,
            method="rf",
            trControl=grid.controlrf,
            tuneGrid=grids.rf)
rf.fit

```

# Our results show the best number of trees to grow is 3 since the accuracy is 0.8596902 and the kappa value is 0.49306108

```{r}
plot(rf.fit)
```

#Visualizing the variable importance for the model

```{r}
varImp(rf.fit)
```

```{r}
pred.rf.train <- predict(rf.fit,
                  dec.train[,-11])
perf.rf.train <- confusionMatrix(pred.rf.train,
                        dec.train$Exited, positive="Left",
                        mode="prec_recall")
perf.rf.train
```

~~~
```{r}
pred.rf.valid <- predict(rf.fit,
                         dec.valid[,-11])
perf.rf.valid <- confusionMatrix(pred.rf.valid,
                                 dec.valid$Exited,  positive="Left",
                                 mode="prec_recall")
perf.rf.valid
```
# Our classification is not as good as expected since the accuracy is 0.863 and kappa = 0.5039.

## Random Forest Lift chart
```{r}
predvec.rf <- ifelse(pred.rf.valid=="Left", 1, 0)

# And then a lift chart
gain <- gains(realvec.dt, predvec.rf, groups=5)
plot(c(0,gain$cume.pct.of.total*sum(realvec.dt))~
     c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec.dt))~c(0, dim(valid)[1]), lty=2)
```

```{r}
pred = prediction(predvec.rf, realvec.dt)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
```

# Neural Network

```{r}
ann.df <- preProcess(knn2, method="range")
ann.df1 <- predict(ann.df, knn2)
summary(ann.df1)
```

```{r}
set.seed(831)
index.ann <- createDataPartition(ann.df1$Exited, p=0.60, list=FALSE)
ann.train = ann.df1[index.ann, ]
ann.valid = ann.df1[-index.ann, ]
```

```{r}
grids.ann = expand.grid(size = seq(from = 1, to = 5, by = 1),
~~~

```
                 decay = seq(from = 0.1, to = 0.5, by = 0.1))

ctrl <- trainControl(method="repeatedcv",
                 number = 5,
                 repeats=3,
                 search="grid")

set.seed(831)
ann.model <- train(Exited~ ., data = ann.train,
             method = "nnet",
             maxit=500,
             trControl = ctrl,
             tuneGrid=grids.ann,
             verbose=FALSE)

ann.model
```

```{r}
plot(ann.model)
```

```{r}
pred.ann <- predict(ann.model, newdata = ann.train)
perf.ann.train <- confusionMatrix(pred.ann,
                       ann.train$Exited,
                       positive="Left",
                       mode="prec_recall")
perf.ann.train
```

```{r}
pred.ann.valid <- predict(ann.model, newdata = ann.valid)
perf.ann.valid <- confusionMatrix(pred.ann.valid,
                       ann.valid$Exited, positive="Left",
                       mode="prec_recall")
perf.ann.valid
```

## Neural Network Lift chart

```{r}
predvec.nn <- ifelse(pred.ann.valid=="Left", 1, 0)
realvec.nn <- ifelse(ann.valid$Exited=="Left", 1, 0)

# And then a lift chart
gain <- gains(realvec.nn, predvec.nn, groups=5)
plot(c(0,gain$cume.pct.of.total*sum(realvec.nn))~
    c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec.nn))~c(0, dim(valid)[1]), lty=2)
```

~~~
```{r}
pred = prediction(predvec.nn, realvec.nn)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
~~~

# Comparing performance of each model

```{r}
train.perf<-cbind(train.nb=NB.Train.Performance$byClass,
                train.knn=knn.perf.train$byClass,
                train.svm=svm.perf.train$byClass,
                train.dt=dec.perf.train$byClass,
                train.rf=perf.rf.train$byClass,
                train.ann=perf.ann.train$byClass)
train.perf
```

```{r}
## Valid set
valid.perf<-cbind(test_nb=NB.Valid.Performance$byClass,
              test_knn=knn.perf.valid$byClass,
              test_svm=svm.perf.valid$byClass,
              test_dt=dec.perf.valid$byClass,
              test_rf=perf.rf.valid$byClass,
              test_ann=perf.ann.valid$byClass)
valid.perf
```

### Based on above comparison, random forest and neural network are two potencial models to be discussed as solutions for the problem since they have the best F1 value, highest accuracy and the precision also is high

# Random Forest vs Neural Networks
## Feature selection
```{r}
set.seed(831)
varImp(rf.fit)
```
## Recursive feature elimination

```{r}
set.seed(831)
control.rfe <- rfeControl(functions = rfFuncs,
                  method = "repeatedcv",
                  number = 10,
```

```
                 repeats = 3,
                 verbose = FALSE)
Bank.rfe <- rfe(x = dec.train[,-11],
            y = dec.train$Exited,
            rfeControl = control.rfe)
Bank.rfe
```

# As we can see the model suggests to keep all the variables but to test that affirmation we are going to select just the first 5 features and see how this is going to affect in the model's performance

```{r}
train.fs <- dec.train[, colnames(dec.train) %in% c('Exited',Bank.rfe$optVariables[1:5])]
valid.fs <- dec.valid[,colnames(dec.train) %in% c('Exited',Bank.rfe$optVariables[1:5])]
```
```{r, warning = FALSE}
grids3 = expand.grid(mtry = seq(from = 1, to = 10, by = 1))

grid.ctrl <- trainControl(method = "repeatedcv",
                    number = 5,repeats = 3,search="grid")
set.seed(831)
rf.fs.fit <- train(Exited~.,
             data=train.fs,  method="rf",
             trControl=grid.ctrl,
             tuneGrid=grids3)
rf.fs.fit
```

```{r}
plot(rf.fs.fit)
```

## Check training data set performance
```{r}
rf.fs.train <- predict(rf.fs.fit,
               train.fs[,-6])

rf.fs.perf.train<-confusionMatrix(rf.fs.train,
                       train.fs$Exited,positive="Left",
                       mode="prec_recall")
```

## Check validation data set performance
```{r}
rf.fs.valid <- predict(rf.fs.fit,
               valid.fs[,-6])

rf.fs.perf.valid<-confusionMatrix(rf.fs.valid,
                       valid.fs$Exited, positive="Left",
                       mode="prec_recall")
```

## Random Forest Reduced Features Lift chart
```{r}
```

~~~
predvec.rfs <- ifelse(rf.fs.valid=="Left", 1, 0)
realvec.rfs <- ifelse(valid.fs$Exited=="Left", 1, 0)

# And then a lift chart
gain <- gains(realvec.rfs, predvec.rfs, groups=5)
plot(c(0,gain$cume.pct.of.total*sum(realvec.rfs))~
    c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec.rfs))~c(0, dim(valid)[1]), lty=2)
```

```{r}
pred = prediction(predvec.rfs, realvec.rfs)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
```

# Neural network
## Doing Neural networks for selected features

```{r}
Ann.fs.train <- ann.train[, colnames(ann.train) %in%
                    c('Exited',"Age","NumOfProducts","IsActiveMember","Balance",
                      "Geography_Germany","Geography_Spain")]

Ann.fs.valid <- ann.valid[,colnames(ann.valid) %in%
                    c('Exited',"Age","NumOfProducts","IsActiveMember","Balance",
                         "Geography_Germany","Geography_Spain")]

ann.grids = expand.grid(size = seq(from = 1, to = 5, by = 1),
                   decay = seq(from = 0.1, to = 0.5, by = 0.1))

ann.ctrl <- trainControl(method="repeatedcv",
                 number = 5, repeats=3,
                 search="grid")

set.seed(831)
ann.fs.model <- train(Exited~ ., data = Ann.fs.train,
             method = "nnet",
             maxit=500, trControl = ctrl,  tuneGrid=ann.grids,
             verbose=FALSE)
ann.fs.model
```
```{r}
plot(ann.fs.model)
```
~~~

~~~
## Training data set performance
```{r}
Ann.fs.train.pred <- predict(ann.fs.model, newdata=Ann.fs.train)
Ann.fs.train.perf<-confusionMatrix(Ann.fs.train.pred,
                         Ann.fs.train$Exited,  positive="Left",
                         mode="prec_recall")
```

## Valid data set performance
```{r}
Ann.fs.valid.pred <- predict(ann.fs.model, newdata=Ann.fs.valid)
Ann.fs.valid.perf<-confusionMatrix(Ann.fs.valid.pred,
                         Ann.fs.valid$Exited,
                         positive="Left",
                         mode="prec_recall")
```

```{r}

train.fs.perf<-cbind(train.rf=perf.rf.train$byClass,
                 train.fs.rf=rf.fs.perf.train$byClass,
                 train.ann=perf.ann.train$byClass,
                 train.fs.nn=Ann.fs.train.perf$byClass)
train.fs.perf
```

```{r}

valid.fs.perf<-cbind(valid.rf=perf.rf.valid$byClass,
                valid.fs.rf=rf.fs.perf.valid$byClass,
                valid.ann=perf.ann.valid$byClass,
                valid.fs.nn=Ann.fs.valid.perf$byClass)
valid.fs.perf

```

## We notice that obtaining top 5 predictors, they would change the results slightly negative. So, we
can a simpler model while having almost the same predictions.

## Neural Networks Reduced Features Lift chart
```{r}
predvec.nns <- ifelse(Ann.fs.valid.pred=="Left", 1, 0)
realvec.nns <- ifelse(Ann.fs.valid$Exited=="Left", 1, 0)

# And then a lift chart
gain <- gains(realvec.nns, predvec.nns, groups=5)
plot(c(0,gain$cume.pct.of.total*sum(realvec.nns))~
    c(0,gain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart", type="l",col="red")

lines(c(0,sum(realvec.nns))~c(0, dim(valid)[1]), lty=2)
```
~~~

~~~
## ROC plot and AUC value

```{r}
pred = prediction(predvec.nns, realvec.nns)
roc = performance(pred, "tpr", "fpr")

plot(roc, lwd=2, colorize=TRUE)
lines(x=c(0, 1), y=c(0, 1), col="black", lwd=1)

auc = performance(pred, "auc")
auc = unlist(auc@y.values)
auc
```
~~~