Air Pressure System Failure for Scania Trucks

Predict Failures and Minimize Costs based on Sensor Readings

by

Kelvin Ng Han Yao

Table of Contents

Abstract

This report contains 2 sections. Section 1 is about predictive maintenance for air pressure system failure for Scania trucks to predict failures and minimize costs based on sensor readings while Logistic Regression model and Random Forest model are built to predict type of failure and calculate the failure cost. Source code and data for section 1 can be view and retrieve from https://github.com/kelvinnghy96/Air-Pressure-System-Failure-for-Scania-Trucks.git.

Section 2 is about stroke prediction in healthcare industry where 4 machine learning models approach which are Naïve Bayes, Logistic Regression, Random Forest, and Support Vector Machine (SVM) have been built to compare the accuracy among 4 models. Source code and data for section 2 can be view and retrieve from https://github.com/kelvinnghy96/Stroke-Prediction-with-Data-Science.git.

Section 1

Air Pressure System (APS) Failure for Scania Trucks

Air pressure system failure for Scania trucks is a predictive maintenance use case that predict the type of failure. Positive class failure consists of component failures for a specific component of the APS system while the negative class failure consists of truck with failures for components not related to the APS. Cost-metric of miss-classification is provided in the table below.

		Predicted Class	
		neg	pos
True Class	neg	-	Cost_1 = 10
	pos	Cost_2 = 500	-

Table 1

		Predicted Class	
		neg	pos
True Class	neg	TN	FP
	pos	FN	TP

Table 2

Total Cost Formula = (FP * 10) + (FN * 500)

Figure 1

This use case has been used in an industrial challenge at year 2016 at the 15th International Symposium on Intelligent Data Analysis (IDA) event. In the past usage, top 3 scorer for minimum cost in that event is provided as reference in the table below.

Rank	**Score**	**Number of Type 1 faults (FP)**	**Number of Type 2 faults (FN)**
1	9920	542	9
2	10900	490	12
3	11480	398	15

Table 3

1.1 Dataset Description

The dataset consists of data collected from heavy Scania trucks in everyday usage. The system in focus is the Air Pressure system (APS) which generates pressurized air that are utilized in various functions in a truck, such as braking and gear changes. The data consists of a subset of all available data, selected by experts.

The dataset is split into train and test dataset and can be download from https://archive.ics.uci.edu/ml/machine-learning-databases/00421/. The training dataset contains 60000 examples in total in which 59000 belong to the negative class and 1000 for the positive class while test dataset contains 16000 examples. Both train and test dataset consist of 171 attributes. The attribute names of the data have been anonymized for proprietary reasons. The attributes are as class and other anonymized operational data.

1.2 Python Library

Library used in this assignment is list in the figure below.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import make_scorer
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import scikitplot as skplt

from imblearn.over_sampling import SMOTE
```

Figure 2

1.3 Data Preprocessing

1.3.1 Missing Values

In the dataset, null values or missing values are denoted by "na". Missing values in the dataset is calculated in percentage and visualize in the figure below.

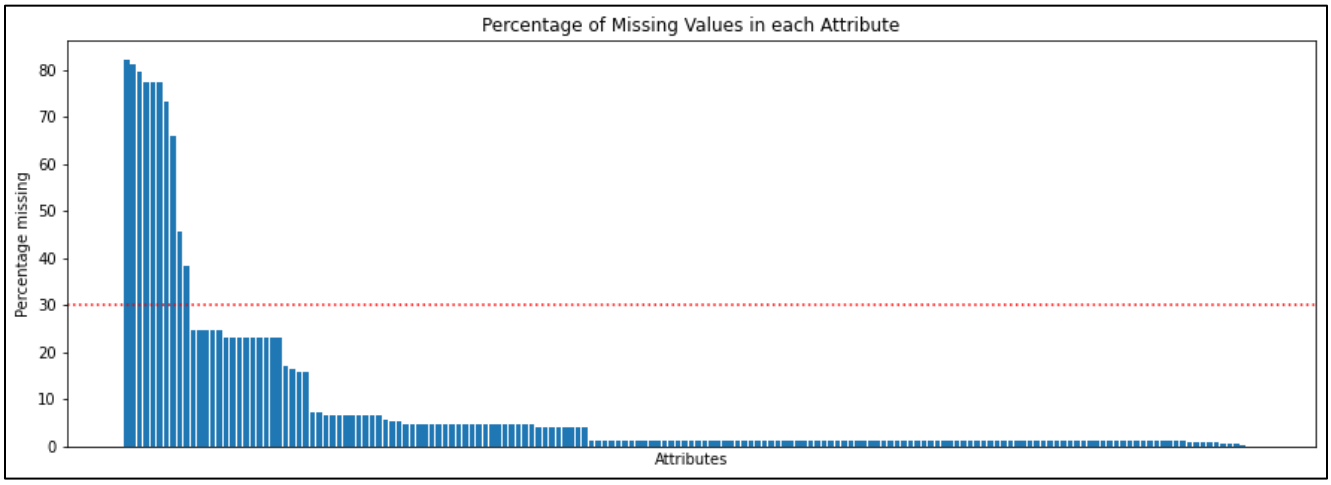

Figure 3

A threshold of 30% is set as the percentage of missing values allowed in each attribute, therefore, attributes that contain more than 30% of missing values is removed from the train and test dataset. There are 10 attributes contain missing values more than 30% and is listed in the table below.

Attributes contain missing values more than 30%				
br_000	bp_000	ab_000	bn_000	bl_000
bq_000	bo_000	cr_000	bm_000	bk_000

Table 4

The leftover 161 attributes with missing values are filled up with their respective attribute's median value. The reason of filling up missing value with median value instead of mean value is because some of the attributes contain outlier value which may contain important information and cant be removed from the dataset, therefore, median value is more suitable than mean value in this situation.

1.3.2 Encoding Categorical Data

As Logistic Regression model is used in this assignment, all categorical data need to be encoded into numeric value. The only categorical attribute in the dataset is the 'class' attribute. Instead of using the LabelEncoder() function from sklearn library, an alternative way is used which direct replace the class label of 'neg' with 0 and 'pos' with 1 by using replace() function which provide the same output as LabelEncoder() function.

1.3.3 Feature Scaling with Standardization

Standardization is use in the feature scaling of this assignment. For every feature, all value gets rescale to mean value of 0 and a standard deviation of 1. The reason of using Standardization instead of Min-max Normalization is because if there's outliers in feature, normalizing the data will scale most of the data to a small interval, which means all features will have the same scale but does not handle outliers well. Standardization is more robust to outliers, and in many cases, it is preferable over Min-max Normalization.

1.3.4 Class Balancing with Synthetic Minority Oversampling Technique (SMOTE)

SMOTE a statistical technique to generate new instances from existing minority cases to increase the number of cases in data in a balanced way. The target variable, 'class' contain imbalance data of 'pos' and 'neg' values which is 1,000 and 59,000 respectively while under SMOTE technique, 'pos' value is oversample to 59,000 rows of data to balance with the 'neg' value as shown in the figure below. Balanced data provide better accuracy and prevent model from penalizing minority samples.

Before SMOTE	
Category	**Row Count**
pos	1,000
neg	59,000

After SMOTE	
Category	**Row Count**
pos	59,000
neg	59,000

Figure 4

1.3.5 Test Data Preprocessing

Apply the same preprocess model from train dataset to test dataset except SMOTE technique. SMOTE technique will only be applied in the train dataset. The data preprocessing steps applied in test dataset is list in below table.

No.	Test Data Preprocessing Steps
1	Drop column that contain more than 30% missing values
2	Replace missing values with median value
3	Encode class label from ‘neg’ to 0 and ‘pos’ to 1
4	Feature Scaling with Standardization

Table 5

1.4 Model Development

1.4.1 Logistic Regression

Logistic Regression model is used to predict type of failure in this assignment. Logistic Regression model is train before and after feature selection to compare the confusion matrix, model accuracy and the total cost.

False positive and false negative number in Logistic Regression model is dropped by 20% and 17.54% respectively after applied with feature selection as shown in the figure below.

Before Feature Selection

		Predicted Class		**Total**
		neg (0)	pos (1)	
True Class	neg (0)	15,265	360	15,625
	pos (1)	57	318	375
Total		15,322	678	16,000

After Feature Selection

		Predicted Class		**Total**
		neg (0)	pos (1)	
True Class	neg (0)	15,325	300	15,625
	pos (1)	47	328	375
Total		15,372	628	16,000

Figure 5

Model accuracy is increase by 0.4375% after applied with feature selection as shown in the figure below.

Model Accuracy without Feature Selection	97.3937%
Model Accuracy with Feature Selection	97.8312%

Figure 6

Precision and recall are increase by 0.05 and 0.02 respectively after applied with feature selection while F1 Score is increase by 0.05 as shown in the figure below.

Before Feature Selection			
Column	**Precision**	**Recall**	**F1 Score**
neg (0)	1	0.98	0.99
pos (1)	0.47	0.85	0.6

After Feature Selection			
Column	**Precision**	**Recall**	**F1 Score**
neg (0)	1	0.98	0.99
pos (1)	0.52	0.87	0.65

Figure 7

Total cost is reduced by 5,600 as shown in the figure below. Although the cost of 26,500 is still larger than the cost reference provided by past use case, but it still clearly shown that total cost is reduced after feature selection is applied.

Model	**Score**	**Number of Type 1 faults (FP)**	**Number of Type 2 faults (FN)**
Logistic Regression without Feature Selection	32,100	360	57
Logistic Regression with Feature Selection	26,500	300	47

Figure 8

1.4.2 Random Forest

Random Forest model is built with feature selection to compare the accuracy and total cost with Logistic Regression with feature selection. Confusion matrix for Random Forest model with feature selection is as below figure.

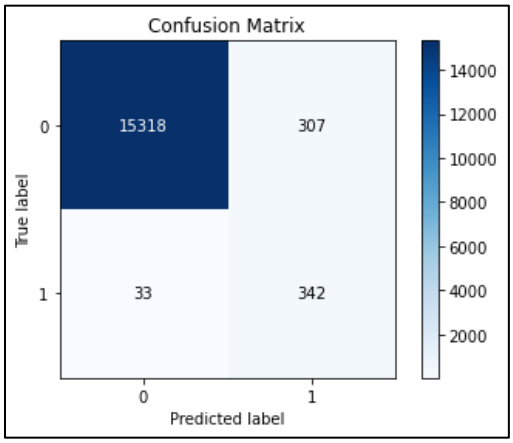

Figure 9

Random Forest confusion matrix is compared with Logistic Regression confusion matrix. Although the FP is higher slightly by 2.33% but FN is lower by 29.79% as shown in figure below.

Logistic Regression with Feature Selection				
		Predicted Class		**Total**
		neg (0)	pos (1)	
True Class	neg (0)	15,325	300	15,625
	pos (1)	47	328	375
Total		15,322	678	16,000

Random Forest with Feature Selection				
		Predicted Class		**Total**
		neg (0)	pos (1)	
True Class	neg (0)	15,318	307	15,625
	pos (1)	33	342	375
Total		15,372	628	16,000

Figure 10

Below figure will compare Precision, Recall and F1 Score between Random Forest and Logistic Regression. Random Forest Precision, Recall and F1 Score is higher by 0.01, 0.04 and 0.02 respectively.

Logistic Regression with Feature Selection			
Column	**Precision**	**Recall**	**F1 Score**
neg (0)	1	0.98	0.99
pos (1)	0.52	0.87	0.65

Random Forest with Feature Selection			
Column	**Precision**	**Recall**	**F1 Score**
neg (0)	1	0.98	0.99
pos (1)	0.53	0.91	0.67

Figure 11

Below figure will compare total cost and model accuracy between Random Forest and Logistic Regression. Random Forest total cost is lower than Logistic Regression total cost by 6,930 while Random Forest model accuracy is higher than Logistic Regression model accuracy by 0.0438%

Model	**Score**	**Number of Type 1 faults (FP)**	**Number of Type 2 faults (FN)**
Logistic Regression with Feature Selection	26,500	300	47
Random Forest with Feature Selection	19,570	307	33

Figure 12

Logistic Regression Model Accuracy with Feature Selection	97.8312%
Random Forest Model Accuracy with Feature Selection	97.8750%

Figure 13

1.4.3 Summary

After perform feature selection the total cost of the model is reduced while the model accuracy increase. Logistic Regression with feature selection have a better model accuracy and total compare to Logistic Regression without feature selection but Random Forest with feature selection is the champion model in this assignment with a total cost of 19,570 and model accuracy of 97.8750%.

Section 2

Stroke Prediction with Data Science

2.1 Dataset Description

This use case used a stroke prediction dataset which contain 5110 observations with 13 attributes. This dataset contains patient id to ensure the uniqueness of each observation and 12 other attributes which will be explained in detail. Gender, categorical data with categories of male and female. Age, continuous data with normal range between 0 to 120. Age categories, categorical data with categories of infants, adults, children, older adults, and teens. Smoking status, categorical data with categories of formerly, never and smokes. Married status, categorical data with categories of yes and no which determine whether an individual is married previously. Employment status with categories of children, government job, never worked, private and self-employed. Region type, categorical data with categories of rural and urban. Body mass index, continuous data with normal range between 20 to 60. Average glucose level, continuous data with normal range between 60 to 240. High blood pressure, categorical data with categories of 0 and 1 which determine whether an individual is having high blood pressure. Heart disease, categorical data with categories of 0 and 1 which determine whether an individual is having a heart disease. Stroke status, target variable in this dataset and categorical data with categories of yes and no which determine whether an individual is having stroke.

2.2 Learning Technique and Library Used

The learning technique used in this assignment is supervised learning technique and the package and library used is listed in figure below.

psych	caTools	lattice	caret
DataExplorer	mltools	grid	randomForest
dplyr	data.table	UBL	tictoc
ggplot2	reshape2	e1071	

Figure 14

The machine learning method use in this assignment are Naïve Bayes, Logistic Regression, Random Forest, and SVM as these methods are good classifier to deal with categorical target variable. The metric used in this assignment to evaluate the performance of the model is accuracy.

2.3 Data Preprocessing

2.3.1 Missing Values, Outlier and Abnormal Values

Before start with dataset preparation, seed is set at 2021 to ensure getting back same result each time the process is rerun. When checking dataset using str(data), wrong data type is detected on body_mass_index which is character type instead of numeric type as figure below.

```
> str(data)
'data.frame':   5110 obs. of  13 variables:
 $ patient_id         : int  292 573 794 1292 1343 1747 1918 1927 2264 2408 ...
 $ gender             : chr  "Male" "Female" "Female" "Female" ...
 $ age                : int  82 75 83 69 55 80 52 64 81 68 ...
 $ age_categories     : chr  "OlderAdults" "OlderAdults" "OlderAdults" "OlderAdults" ...
 $ smoking_status     : chr  "Never" "Never" "Formerly" "Unknown" ...
 $ married_status     : chr  "Yes" "Yes" "No" "Yes" ...
 $ employment_status  : chr  "SelfEmployed" "SelfEmployed" "Private" "Private" ...
 $ region_type        : chr  "Rural" "Urban" "Rural" "Rural" ...
 $ body_mass_index    : chr  "30.4" "25.6" "25.5" "38.3" ...
 $ avg_glucose_level  : num  89.5 73 82 209.1 69.2 ...
 $ high_blood_pressure: int  0 0 1 0 0 1 1 0 1 1 ...
 $ heart_disease      : int  0 0 1 0 0 0 0 0 0 0 ...
 $ stroke_status      : chr  "Yes" "Yes" "Yes" "Yes" ...
```

Figure 15

The body_mass_index data type is character due to “N/A” text value in the column, therefore, as.numeric() function is used to automatically convert “N/A” text value to null value and convert the other data to numeric data type as figures below.

```
data$body_mass_index = as.numeric(data$body_mass_index)
```

Figure 16

```
> class(data$body_mass_index)
[1] "character"
> data$body_mass_index = as.numeric(data$body_mass_index)
Warning message:
NAs introduced by coercion
> class(data$body_mass_index)
[1] "numeric"
```

Figure 17

Patient id is dropped from the dataset and an extra category of “Other” is removed from the Gender column as figure below.

```
# Drop ID column
df <- select(data, -patient_id )


# Remove 'Other' category from gender column
df = df[(df$gender == 'Male' | df$gender == 'Female'),]
table(df$gender)
```

Figure 18

Both missing data and empty string is detected in avg_glucose_level, employment_status, married_status, heart_disease, high_blood_pressure, body_mass_index, smoking_status and removed from the dataset as figure below.

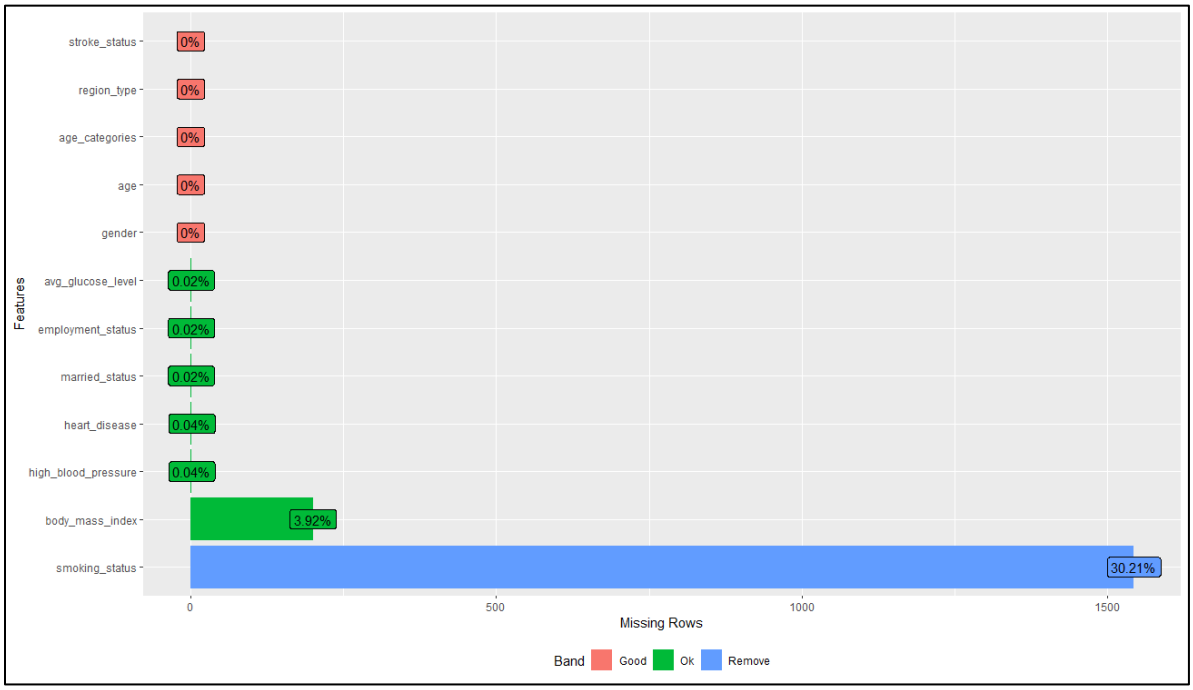

Figure 19

Outlier value and abnormal value is detected and removed from body_mass_index, avg_glucose_level and age as figure below.

```
> describe(df)
                    vars    n   mean     sd median trimmed   mad    min      max    range
gender*                1 5108   1.41   0.49   1.00    1.39  0.00   1.00     2.00     1.00
age                    2 5108  44.18  22.68  46.00   44.57 26.69 -61.00    83.00   144.00
age_categories*        3 5108   1.96   1.41   1.00    1.76  0.00   1.00     5.00     4.00
smoking_status*        4 5108   3.58   1.09   3.00    3.61  1.48   1.00     5.00     4.00
married_status*        5 5108   2.66   0.48   3.00    2.69  0.00   1.00     3.00     2.00
employment_status*     6 5108   4.49   1.28   5.00    4.62  0.00   1.00     6.00     5.00
region_type*           7 5108   1.51   0.50   2.00    1.51  0.00   1.00     2.00     1.00
body_mass_index        8 4908  28.43  39.21  27.10   27.35  6.97 -37.70  2718.00  2755.70
avg_glucose_level      9 5107 109.29 372.42  89.88   95.86 26.06  53.12 26521.76 26468.64
high_blood_pressure   10 5106   0.10   0.30   0.00    0.00  0.00   0.00     1.00     1.00
heart_disease         11 5106   0.05   0.23   0.00    0.00  0.00   0.00     1.00     1.00
stroke_status*        12 5108   1.05   0.21   1.00    1.00  0.00   1.00     2.00     1.00
```

Figure 20

```
# Check for outlier and abnormal data
describe(df)

# Remove body_mass_index with record more than 100 which is out of normal range
df = df[!(df$body_mass_index > 100),]

# Remove avg_glucose_level with record more than 300 which is out of normal range
df = df[!(df$avg_glucose_level > 300),]

# Remove negative value in age and body_mass_index
df = df[!( df$age < 0 | df$body_mass_index < 0),]
```

Figure 21

Children under age 14 in Malaysia are not allowed to be employed, therefore, data are checked for age under 14 and employment_status are not “Children” as figure below.

```
> # Check and remove value for age under 14 but employment status are not 'Children'
> subset(df,df$age < 14 & df$employment_status!="Children",select=c(age,employment_status))
    age employment_status
47    1           Private
165   1      SelfEmployed
```

Figure 22

2.3.2 Correlation Matrix

Correlation matrix heatmap is built to determine the correlation between all numeric attributes as figure below.

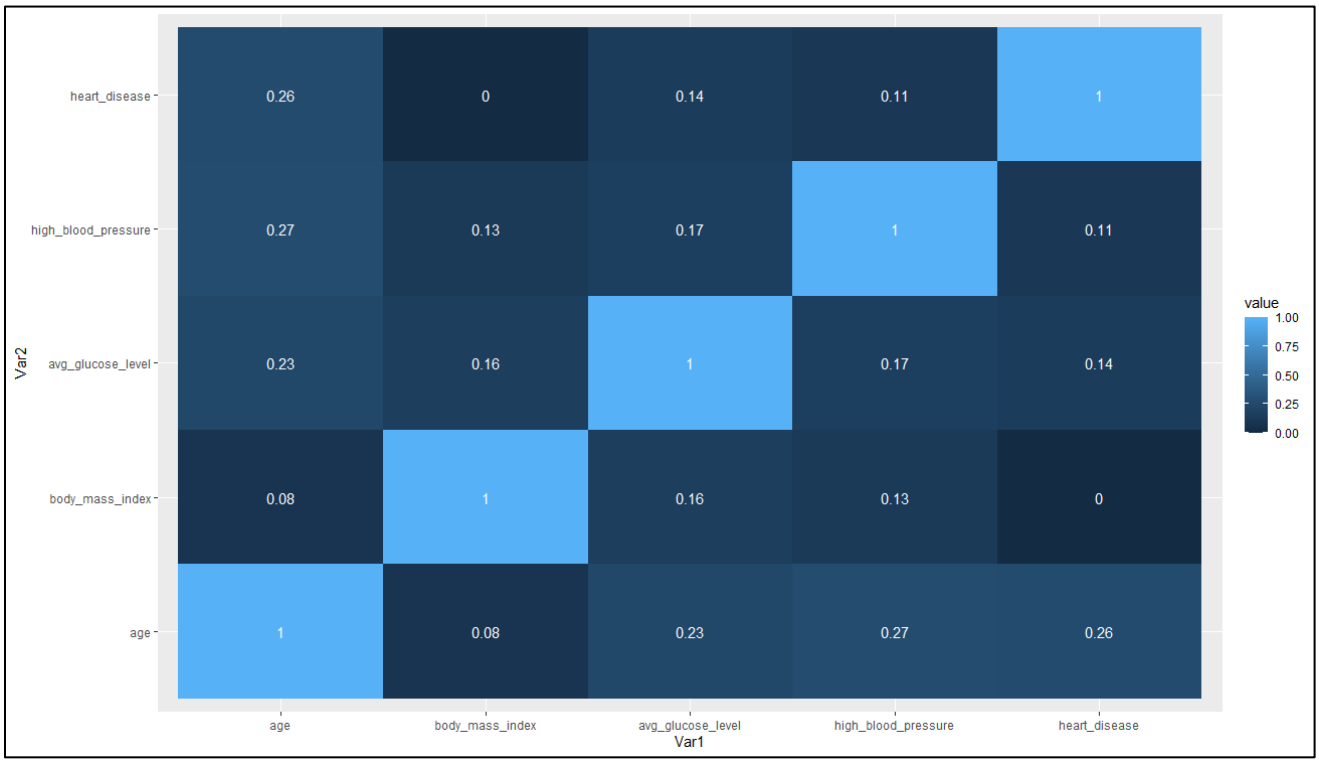

Figure 23

From the figure above, there are no correlation between attribute that is more than 0.8, therefore, no attribute is removed from dataset at this step.

2.3.3 One-hot Encoding

One-hot encoding technique is used to convert categorical variable into binary variable so that the variables can provide more detail information for model training. Categorical column need to be convert to factor type only able to apply with one-hot() function as figure below.

```
# One-hot encoding categorical data with data table
col_names <- c('gender', 'age_categories' ,'smoking_status','married_status',
            'employment_status','region_type')
df[,col_names] <- lapply(df[,col_names] , factor)
df2 <- one_hot(as.data.table(df))
```

Figure 24

Stroke_status which is the target variable is labelled for stroke prediction in later phase as figure below.

```
# Labeling the target variable values
# 'No' to 0 while 'Yes' to 1
df2$stroke_status <- factor(df2$stroke_status, levels=c("No","Yes"),
                            labels=c("0", "1"))
```

Figure 25

Dataset is split into train and test data with a ratio of 0.7 and 0.3 respectively as figure below.

```
> dim(train)
[1] 2400   24
> dim(test)
[1] 1014   24
```

Figure 26

2.3.4 Class Balancing

Once the dataset is split into train and test data, class balancing is performed on the train data. The reason class balancing is performed only on train data is because train data need balance class to train model while test data need to remain as similar as real world data to get a correct accuracy.

```
> table(train$stroke_status)

   0    1
2279  121
```

Figure 27

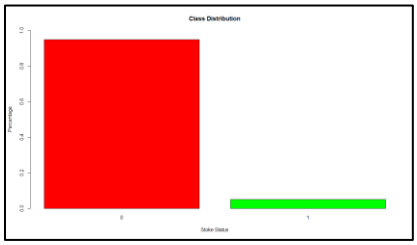

Figure 28

After perform class balancing using SMOTE technique to oversample the dataset, the distribution of target variable is balanced.

```
> table(train_bal$stroke_status)

   0    1
1200 1200
```

Figure 29

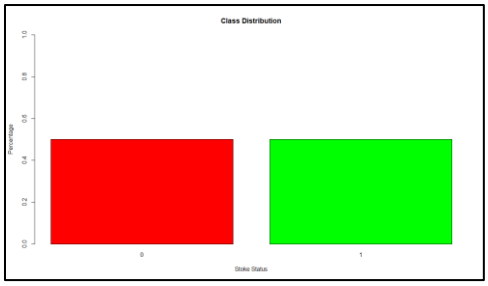

Figure 30

2.3.5 Min-max Normalization

Min-max normalization is performed on both train and test data as preprocess modelling need to be exactly same for train data, test data as well as real world data when the model is in production.

```
# Min- Max Normalization for numerical value
# One-hot encoded column will remain 0 and 1
normalize = function(x) {  return ((x - min(x)) / (max(x) - min(x))) }

# Min- Max Normalization train data
train_bal_norm_tmp <-as.data.frame(lapply(select(train_bal, -stroke_status), normalize))
head(train_bal_norm_tmp)
train_bal_norm <- cbind(train_bal_norm_tmp,stroke_status= train_bal$stroke_status)
head(train_bal_norm)
str(train_bal_norm)

# Min- Max Normalization test data
test_norm_tmp <-as.data.frame(lapply(select(test, -stroke_status), normalize))
head(test_norm_tmp)
test_norm <- cbind(test_norm_tmp,stroke_status= test$stroke_status)
head(test_norm)
str(test_norm)
```

Figure 31

2.4 Model Development

2.4.1 Naïve Bayes

The first model build is Naïve Bayes model. The hyperparameter of Naïve Bayes model is tuned with grid search method as figure below.

```
# Naive Bayes model tuning using grid search method
search_grid = expand.grid(usekernel = c(TRUE, FALSE),fL = 0:5,
                          adjust = seq(0, 5, by = 1))

nb_tune_model = train(xtrain, ytrain, 'nb', metric="Accuracy",tuneGrid = search_grid)
```

Figure 32

```
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

Figure 33

3 hyperparameter in Naïve Bayes were tuned which are fL, usekernel and adjust. fL hyperparameter allow user to include Laplace smoother, usekernel hyperparameter allows user to use a kernel density estimate for continuous variables against a guassian density estimate while adjust hyperparameter is referring to the bandwidth of kernel density. The final value used to build Naïve Bayes model after tuned were fL = 0, usekernel = TRUE and adjust = 1.

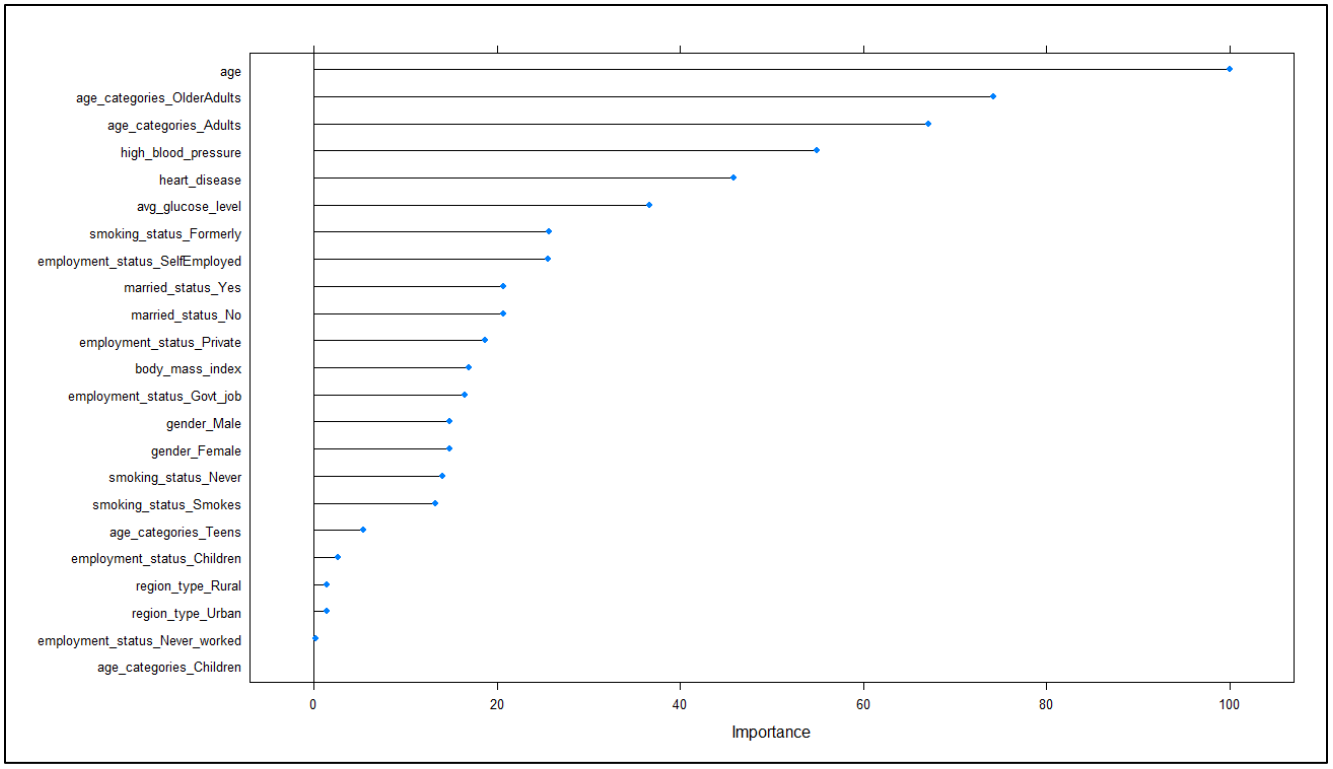

Figure 34

	Importance
age	100.000
age_categories_OlderAdults	74.171
age_categories_Adults	67.088
high_blood_pressure	54.911
heart_disease	45.894
avg_glucose_level	36.619

Figure 35

From the above figure, it's clearly shown that age attribute plays an importance role in model training. The higher the age, the older the age categories, the more importance the role in Naïve Bayes stroke prediction model building. Naïve Bayes stroke prediction model are train again under feature selection with only attributes with importance above 0.3.

```
> nb_cm
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 736  15
         1 228  35
```

Figure 36

The above figure is the confusion matrix of Naïve Bayes model, and the precision of Naïve Bayes model can be calculated which is 35 / (35 + 15) = 0.7 while the recall of Naïve Bayes model is 35 / (35 + 228) = 0.13.

```
> nb_acc
[1] 0.760355
```

Figure 37

The accuracy of Naïve Bayes stroke prediction model is 76.03%.

2.4.2 Logistic Regression

The next model build is Logistic Regression model. The Logistic Regression model is train as figure below.

```
# Logistic Regression model training
logreg_model = glm(stroke_status ~., train_final, family = binomial)
summary(logreg_model)
```

Figure 38

```
Call:
glm(formula = stroke_status ~ ., family = binomial, data = train_final)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.62886  -0.68491   0.08487   0.76979   2.35228

Coefficients: (6 not defined because of singularities)
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                   -1.648e+01  4.531e+02  -0.036  0.97099
gender_Female                 -1.376e-03  1.212e-01  -0.011  0.99094
gender_Male                           NA         NA      NA       NA
age                            8.696e+00  6.065e-01  14.336  < 2e-16 ***
age_categories_Adults          1.058e+01  4.531e+02   0.023  0.98137
age_categories_Children        2.773e-01  1.296e+03   0.000  0.99983
age_categories_OlderAdults     9.534e+00  4.531e+02   0.021  0.98321
age_categories_Teens                  NA         NA      NA       NA
smoking_status_Formerly       -6.655e-01  1.680e-01  -3.961 7.46e-05 ***
smoking_status_Never          -7.975e-01  1.564e-01  -5.099 3.41e-07 ***
smoking_status_Smokes                 NA         NA      NA       NA
married_status_No             -1.992e-01  2.166e-01  -0.920  0.35773
married_status_Yes                    NA         NA      NA       NA
employment_status_Children     2.533e-01  6.894e+02   0.000  0.99971
employment_status_Govt_job     3.655e-01  1.912e-01   1.912  0.05589 .
employment_status_Never_worked -9.684e+00  7.975e+02  -0.012  0.99031
employment_status_Private      2.916e-01  1.443e-01   2.020  0.04333 *
employment_status_SelfEmployed        NA         NA      NA       NA
region_type_Rural             -8.486e-02  1.158e-01  -0.733  0.46373
region_type_Urban                     NA         NA      NA       NA
body_mass_index                5.075e-01  4.786e-01   1.060  0.28897
avg_glucose_level              6.116e-01  2.193e-01   2.789  0.00529 **
high_blood_pressure            1.036e+00  1.461e-01   7.093 1.31e-12 ***
heart_disease                  5.806e-01  1.864e-01   3.115  0.00184 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 39

Feature selection is performed in Logistic Regression model based on significant attributes with 2 * and above based on above figure.

```
Call:
glm(formula = stroke_status ~ ., family = binomial, data = lgxtrain)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.59259  -0.69776   0.03112   0.74190   2.25310

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)              -4.1684     0.2188 -19.055  < 2e-16 ***
age                       6.1232     0.3039  20.146  < 2e-16 ***
smoking_status_Formerly  -0.6485     0.1647  -3.937 8.25e-05 ***
smoking_status_Never     -0.8466     0.1503  -5.633 1.78e-08 ***
high_blood_pressure       0.9539     0.1442   6.613 3.77e-11 ***
heart_disease             0.6131     0.1828   3.355 0.000795 ***
avg_glucose_level         0.7147     0.2045   3.496 0.000473 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3327.1  on 2399  degrees of freedom
Residual deviance: 2246.2  on 2393  degrees of freedom
AIC: 2260.2

Number of Fisher Scoring iterations: 5
```

Figure 40

After feature selection is performed only 7 attributes is retained in Logistic Regression as figure above.

```
> logreg_cm
      logreg_model_predict
ytest   0   1
    0 694 270
    1  11  39
```

Figure 41

The above figure is the confusion matrix of Logistic Regression model, and the precision of Logistic Regression model can be calculated which is 39 / (39 + 270) = 0.13 while the recall of Logistic Regression model is 39 / (11 + 39) = 0.78.

```
> logreg_acc
[1] 0.7228797
```

Figure 42

The accuracy of Logistic Regression stroke prediction model is 72.28%.

2.4.3 Random Forest

The third model build is Random Forest model. The Random Forest model is tuned using grid search method as figure below.

```
# Random Forest model tuning using grid search method

tunegrid <- expand.grid(.mtry=c(1:8))

rf_tune_model <- train(stroke_status~.,data = train_final, method="rf", metric="Accuracy",
tuneGrid=tunegrid)
```

Figure 43

Only one hyperparameter in Random Forest is tuned which is the mtry hyperparameter as tuning on both mtry and ntree parameter will consume a lot of resource of the local machine and the duration took few hours, therefore, in this assignment only mtry hyperparameter is tuned for Random Forest. Mtry hyperparameter will randomly sample variables based on the number assigned as candidate at each split. The final value of mtry hyperparameter tune from grid search method is mtry = 6, therefore, Random Forest model are train with hyperparameter mtry = 6.

```
> rf_cm
                 ytest
rf_model_predict   0   1
               0 945  46
               1  19   4
```

Figure 44

The above figure is the confusion matrix of Random Forest model, and the precision of Random Forest model can be calculated which is 4 / (46 + 4) = 0.08 while the recall of Random Forest model is 4 / (19 + 4) = 0.17.

```
> rf_acc
[1] 0.9358974
```

Figure 45

The accuracy of Random Forest stroke prediction model is 93.59%.

2.4.4 Support Vector Machine (SVM)

The next model build is SVM model. The SVM model is tuned using grid search method as figure below.

```
# SVM model tuning using grid search method
svm_tune_model = tune(svm, stroke_status~., data=train_final,
                      ranges = list(epsilon = seq (0, 1, 0.1), cost = 2^(0:2)))
```

Figure 46

3 hyperparameter in SVM were tuned which are epsilon, cost, and kernel. Epsilon hyperparameter represent the margin that user allow and tolerate that penalty is not given to error, cost hyperparameter also referring to cost of misclassification where user decide how much data that SVM are allowed to misclassify while kernel hyperparameter is referring to the method of mathematical function used to deal with input data and transform into. The final value used to build SVM model after tuned epsilon = 0, cost = 4, kernel = radial.

```
> svm_cm
          Actual
Predicted   0   1
        0 910  43
        1  54   7
```

Figure 47

The above figure is the confusion matrix of SVM model, and the precision of SVM model can be calculated which is 7 / (7 + 43) = 0.14 while the recall of SVM model is 7 / (7 + 54) = 0.11.

```
> svm_acc
[1] 0.9043393
```

Figure 48

The accuracy of SVM stroke prediction model is 90.43%.

2.4.5 Summary

Total of 4 models have been trained in this assignment which are Naïve Bayes, Logistic Regression, Random Forest and SVM. As small dataset gets overfitted easily, therefore, these 4 models is chosen because they can perform better when dealing with small dataset. All four model has been trained and tested with preprocessed data and preprocess model and the main performance metric which is accuracy for the 4 model are recorded as figure below.

```
> nb_acc
[1] 0.760355
> logreg_acc
[1] 0.7228797
> rf_acc
[1] 0.9358974
> svm_acc
[1] 0.9043393
```

Figure 49

Based accuracy of 4 trained model in this assignment, Random Forest model will be the champion model in this assignment with the highest accuracy.

Model	**Precision**	**Recall**	**Accuracy**
Naïve Bayes	70%	13%	76.03%
Logistic Regression	13%	78%	72.28%
Random Forest	8%	17%	93.59%
SVM	14%	11%	90.43%

Table 6

Naïve Bayes model and Logistic Regression model have a high precision and recall respectively compared to Random Forest model and SVM model which mean Naïve Bayes model can get the true positive value and manage to determine patient who are having stroke while Logistic Regression model manage to get a high amount of false negative which mean logistic regression can't determine the patient who having stroke but it somehow predicted those who are not having stroke might have stroke in the future which preventive action can be take in advance.

Although Random Forest model and SVM model don't have high precision rate and recall rate, but their accuracy is higher compared to Naïve Bayes model and Logistic Regression model which are 93.59% and 90.43% respectively.

SVM outperform Logistic Regression because SVM finds the optimum distance between line and support vectors to separate class and this lower risk of classification error, while logistics regression can have different decision line with various weight that close to optimal point.

Random Forest outperformed Logistic Regression because the explanatory variable in this dataset is more while Logistic Regression can only perform better when the explanatory variable is more than noise variable.

SVM outperform Naïve Bayes because SVM deal with the interaction between attribute until certain level, but Naïve Bayes treat all attributes independently which cant interpret a deeper level of relationship between attributes but Naïve Bayes train faster than SVM as only probability of each class needed to be calculated in Naïve Bayes.

Random Forest outperformed Naïve Bayes because it's have a much more complex and large model size compare to Naïve Bayes while Naïve Bayes is simple model and cannot cater complicated data behavior, therefore, Random Forest have better performance than Naïve Bayes with a dataset with complex behavior but advantage of Naïve Bayes is that it can quickly adapt to the changes in any new dataset while Random Forest need to rebuild whenever there's changes in dataset else it will lead to overfitting.

In most of the related work, SVM and Random Forest are used more in stroke prediction model building compared to Naïve Bayes and Logistic Regression as in the real world industry, stroke is a critical disease that can ruin a patient life, therefore, a model with higher accuracy will be prioritize in the real world situation and same case in this assignment where Random Forest stroke prediction model will be recommended as it is the champion model and it has the highest accuracy compare to the other model.

References

AB, S. C. (2016, September). *APS Failure at Scania Trucks Data Set*. Retrieved from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks

Authors, T. (2019). Comparing regression, naive Bayes, and random forest methods in the prediction of individual survival to second lactation in Holstein cattle. *Journal of Dairy Science*, 13.

Benjamin Letham, C. R. (2015). Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *Ann. Appl. Stat.* , 22.

Combi, C. (2020). Artificial Intelligence in Medicine. *Artificial Intelligence in Medicine*, 568.

Dave, P. (2021). *How to create a Stroke Prediction Model?* Retrieved 12 8, 2021, from https://www.analyticsvidhya.com/blog/2021/05/how-to-create-a-stroke-prediction-model/

De, S. (2019, September 14). *APS Failure at Scania Trucks Data Set: Predicting if a Truck needs Servicing*. Retrieved from Medium: https://medium.com/analytics-vidhya/aps-failure-at-scania-trucks-data-set-1eb97b12812

Gondek, C., Hafner, D., & Sampson, O. R. (2016). Prediction of Failures in the Air Pressure. *University of Konstanz, Germany*, 5.

Kuo-Liong Chien, M. P., Ta-Chen Su, M. P., Hsiu-Ching Hsu, P., Wei-Tien Chang, M. P., Pei-Chun Chen, P., Fung-Chang Sung, P., . . . Yuan-Teh Lee, M. P. (2021). *Constructing the Prediction Model for the Risk of Stroke in.* Taiwan: American Heart Association, Inc.

Learning, U. M. (2018). *https://www.kaggle.com/uciml/aps-failure-at-scania-trucks-data-set*. Retrieved from kaggle: https://www.kaggle.com/uciml/aps-failure-at-scania-trucks-data-set

Lemons, K. (2020). A Comparison Between Naïve Bayes and Random Forest to Predict Breast Cancer. *International Journal of Undergraduate Research and Creative Activities*, 6.

Rich Caruana, Y. L. (2015). Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission. *RESEARCH-ARTICLE*, 10.

Risk Factors and Prediction of Stroke in a Population with High Prevalence of Diabetes: The Strong Heart Study. (2017, May). Retrieved from Scientific Research: https://www.scirp.org/journal/paperinformation.aspx?paperid=76573

Sailasya, G., & Kumari, G. L. (2021). Analyzing the Performance of Stroke Prediction using. *International Journal of Advanced Computer Science and Applications*, 7.

sawant, m. (2019, June 29). *APS Failure at Scania Trucks*. Retrieved from Medium: https://medium.com/swlh/aps-failure-at-scania-trucks-203975cdc2dd

Shetty, R. (2021, January 2). *Predicting a Failure in Scania's Air Pressure System*. Retrieved from towards data science: https://towardsdatascience.com/predicting-a-failure-in-scanias-air-pressure-system-aps-c260bcc4d038

Sida Wang, C. D. (n.d.). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. 5.

Stroke Prediction using Data Analytics and Machine Learning. (2021, June 22). Retrieved from TechTarget: https://www.datasciencecentral.com/profiles/blogs/stroke-prediction-using-data-analytics-and-machine-learning

Sundaresan, B. (2021, July 3). *Stroke Prediction*. Retrieved from RPubs: https://rpubs.com/bharath2925/strokeprediction

Teoh, D. (2018). Towards stroke prediction using. *Teoh BMC Medical Informatics and Decision Making*, 11.

Veena Potdar, L. S. (2021). A Survey on Stroke Disease Classification and Prediction using Machine Learning Algorithms. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT)*.

Vida, Avula, V., Chaudhary, D., Shahjouei, S., Khan, A., Griessenauer, C. J., . . . Ramin. (2021). Prediction of Long-Term Stroke Recurrence Using Machine Learning Models. *JCM*.

Zdrodowska, M. (2019). ATTRIBUTE SELECTION FOR STROKE PREDICTION. 4.

Zhang, R. (2019). Interpretable Machine Learning Methods for Stroke Prediction. 75.