Lung Cancer UCI - Capstone Project

Mitja Prah, MD - HarvardX Data Science Professional Program

16/12/2020

Contents

1 Overview

Lung cancer is the leading cause of death from cancer worldwide (18.4% of all cancer deaths) and causes more deaths than breast, colorectal, and cervical cancers combined. Only 15% of patients with lung cancer are still alive 5 years after diagnosis, because approximately 70% of patients have advanced disease at the time of diagnosis. Lung cancer and other tobacco related diseases are expected to remain important health problems worldwide for decades.[1]

The relationship between smoking, lung cancer and airflow obstruction is recognised. A reduction in the percent of predicted forced expiratory volume in 1 second (*FEV1%pred*) is a significant predictor of increased lung cancer risk.[2]

The treatments for lung cancer mainly include surgical resection and chemotherapy. In general, patients with early-stage lung cancer usually undergo surgery, while patients with advanced or metastatic disease are treated with chemotherapy. Several factors, such as age, sex, lung function, clinical and pathological stage, body constitution, comorbidity, and optimal treatment, influence the survival of lung cancer patients.[3]

Patients with stage I tumours have a high chance and those with stage II tumours a reasonable chance of being cured by surgery alone. Patients with stage I (cT1N0 and cT2N0) and stage II (cT1N1, cT2N1 and cT3N0) tumours should be considered operable. Although evidence is contradictory, the majority of studies have shown that at 2 years survival after surgery for lung cancer is similar in elderly patients to that in other age groups. Beyond stage II, survival is very poor.[4]

Since thoracic surgery is not without its own inherent risks, it could be beneficial to patients and healthcare providers to have some insight into expected post-surgical risk/benefit scenarios as they may relate to a patient's health history and baseline characteristics at the time of diagnosis and surgery. If there is a pattern to be recognized in various patient and disease characteristics, this would help physicians to make a more educated decision on whether an individual patient is a suitable candidate for surgical resection. In case of a recognized high risk of a patient's death in 1 year after the surgery, alternative treatments or paliative care may be preferred.

1.1 Project Objective

Using retrospective clinical study data, we will construct and evaluate several supervised machine learning models to predict patient survival past 1 year after the surgical resection of lung cancer. Data will be transformed as required to reveal patterns in the dataset and create more robust analyses. Finally, the optimal model will be selected based on several evaluation metrics.

Similar research projects, using the same dataset, have previously been conducted by different authors. In most cases the prediction accuracy of machine learning algorithms was up to 87% [5, 6]. With the use of data balancing and feature engineering, testing accuracy of more than 97% can be achieved [7].

1.1.1 Evaluation metric definitions

Sensitivity is defined as the ability of an algorithm to predict a positive outcome when the actual outcome is positive: $\hat{Y} = 1$ when $Y = 1$. Because an algorithm that calls everything positive ($\hat{Y} = 1$ no matter what) has perfect sensitivity, this metric on its own is not enough to judge an algorithm. For this reason, we also examine specificity. Sensitivity, also referred to as the *true positive rate (TPR) or recall*, is typically quantified by $TP/(TP + FN)$, where TP are false positives and FN are false negatives.

Specificity is generally defined as the ability of an algorithm to not predict a positive $\hat{Y} = 0$ when the actual outcome is not a positive $Y = 0$. Specificity is typically quantified as $TN/(TN + FP)$, where TN are true negatives and FP are false positives. It can sometimes also be quantified as $TP/(TP + FP)$ or the proportion of outcomes called positives that are actually positives.

We can summarize both in the following way:

- High sensitivity: $Y = 1 \implies \hat{Y} = 1$
- High specificity: $Y = 0 \implies \hat{Y} = 0$; or also
- High specificity: $\hat{Y} = 1 \implies Y = 1$

The multiple names can be confusing, and are presented in the following table.

Table 1: Sensitivity and specificity definitions

Measure of	**Name 1**	**Name 2**	**Definition**	**Probability representation**
sensitivity	TPR	Recall	$\frac{\text{TP}}{\text{TP+FN}}$	$\Pr(\hat{Y} = 1 \mid Y = 1)$
specificity	TNR	1-FPR	$\frac{\text{TN}}{\text{TN+FP}}$	$\Pr(\hat{Y} = 0 \mid Y = 0)$
specificity	PPV	Precision	$\frac{\text{TP}}{\text{TP+FP}}$	$\Pr(Y = 1 \mid \hat{Y} = 1)$

TPR is True Positive Rate, TNR is True Negative Rate, FPR is False Positive Rate, and PPV is Positive Predictive Value.

Accuracy is the percentage of observations that were correctly predicted and is defined by the following equation:

$$Accuracy = \frac{TP + TN}{P + N}$$

P stands for all (predicted and actual) positive outcomes, N stands for all negative outcomes. Accuracy is not a reliable metric for the real performance of a machine learning technique, because it will yield misleading results if the data set is imbalanced.

Balanced Accuracy is a useful one-number summary metric, defined as the average of specificity and sensitivity. For imbalanced data, it is a more reliable method than overall accuracy.

$$BalancedAccuracy = \frac{TPR + TNR}{2}$$

The F_1 score is another one-number summary metric, defined as the harmonic average of *precision* and *recall*:

$$F_1 = \frac{1}{\frac{1}{2}\left(\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}\right)}$$

Reciever Operating Characteristic (ROC) curve is a graphical plot (where TPR is plotted against the FPR) that illustrates the diagnostic or predictive ability of a binary classifier system. It is used for the evaluation of various methods/models by comparing them graphically.

Precision-Recall Curve (PRC) is a graphical plot (where Precision is plotted against the Recall). It has similar use as the ROC curve, but also takes the prevalence into account. It is particularly powerful for imbalanced datasets [7], where it can be more informative than a ROC curve.

For more details on evaluation metrics, please reference https://rafalab.github.io/dsbook/.[8]

1.2 Dataset and Data Collection

The dataset used in this research project is available online at https://archive.ics.uci.edu/ml/datasets/Thoracic+Surgery+Data#. The data was collected retrospectively at Wroclaw Thoracic Surgery Centre for over *1200 consecutive patients* who underwent major lung resections for primary lung cancer in the years

2007-2011. The Centre is associated with the Department of Thoracic Surgery of the Medical University of Wroclaw and Lower-Silesian Centre for Pulmonary Diseases, Poland, while the research database constitutes a part of the National Lung Cancer Registry, administered by the Institute of Tuberculosis and Pulmonary Diseases in Warsaw, Poland. The main dataset included 139 predictors, of which 36 from pre-operative, 37 from peri-operative, and 46 (including 17 pathology-related) from post-operative periods.[9]

The freely available dataset, used in our current project, is composed of 470 observations, 16 pre-operative features and 1 outcome (*1-year survival*). We will refer to it as the **lc dataset**. The variables are presented in the following table.

Table 2: Variables in the original lc dataset

Variable	**Description**	**Characteristics**
DGN	Diagnosis - specific combination of ICD-10 codes for primary and secondary as well multiple tumours if any	Nominal, factor with 7 levels
PRE4	Forced vital capacity - FVC	Numeric
PRE5	Volume that has been exhaled at the end of the first second of forced expiration - FEV1	Numeric
PRE6	Performance status - Zubrod scale	Ordinal, factor with 3 levels (PRZ0, PRZ1, PRZ2)
PRE7	Pain before surgery	Binary - True/False
PRE8	Haemoptysis before surgery	Binary - True/False
PRE9	Dyspnoea before surgery	Binary - True/False
PRE10	Cough before surgery	Binary - True/False
PRE11	Weakness before surgery	Binary - True/False
PRE14	T in clinical TNM - size of the original tumour, from OC11 (smallest) to OC14 (largest)	Ordinal, factor with 4 levels
PRE17	Type 2 Diabetes mellitus	Binary - True/False
PRE19	Myocardial infarction (MI) up to 6 months	Binary - True/False
PRE25	Peripheral arterial disease (PAD)	Binary - True/False
PRE30	Smoking	Binary - True/False
PRE32	Asthma	Binary - True/False
AGE	Age at surgery	Numeric
Risk1Yr	1 year survival period - True if died, False if survived	Binary - True/False

2 Methods

2.1 Data Pre-Processing

When observing the outcomes and other binary variables, it is clear that the dataset is highly imbalanced. This is shown in the following table.

Table 3: Distribution of binary variables

	Risk1Yr	PRE7	PRE8	PRE9	PRE10	PRE11	PRE17	PRE19	PRE25	PRE30	PRE32
T	70	31	68	31	323	78	35	2	8	386	2
F	400	439	402	439	147	392	435	468	462	84	468

Only 70 out of 470 observations, i.e. 14.9% of the data, are associated with the outcome of death within one year. Similar imbalance is seen for all observed variables. Such imbalance can have a negative impact on machine learning model performance. This issue can be resolved with a bootstrapping algorithm SMOTE (Synthetic Minority Over-sampling Technique), or a ROS (Random Over Sampler) method.

I replaced the dataset column names with more intuitive names to facilitate further procedures. Binary variables appearing as "T" or "F" were converted into integers, where each "F" was replaced with a 0 and each "T" with a 1. This conversion is required, since many machine learning algorithms cannot operate on label/categorical data directly, but require all input variables and output variables to be numeric.

There are two categorical ordinal variables in the original dataset: PRE6 (renamed into Zubrod) and PRE14 (renamed into Tumor_size).

The Zubrod score, also called the WHO score, is a widely used scoring system for assessing patient **performance status**. The performance status is an attempt to quantify cancer patients' general well-being and activities of daily life. The Zubrod score is a simple measure that runs from 0 to 5, with 0 denoting perfect health and 5 denoting death. In the lc dataset, there are only observations with Zubrod score from 0 to 2, which seems reasonable, since patients with higher Zubrod score would likely be unfit for undergoing a major surgery. The text portions from the original variable, presented above, were redundant, so they were stripped, and the remaining numerical portions of the variable were converted into integers 0, 1, or 2, respectively, to optimize further analyses and machine learning. The integer values have the following meaning:

- 0 - asymptomatic patient
- 1 - symptomatic, but completely ambulatory patient
- 2 - symptomatic patient, <50% time in bed during the day

The size of a primary tumor, denoted by the PRE14 variable, is part of the **TNM classification** of malignant tumors, which is a globally recognised standard for classifying the extent of spread of cancer. The value of T in the TNM classification system describes the size of the primary tumor and whether it has invaded nearby tissue. In the lc dataset, there are observations with T values from 1 (smallest tumor) to 4 (largest tumor). Most characters from the original variable values, presented above, were redundant, so only the last digits of the values were kept and converted into integers 1, 2, 3, or 4, respectively, to optimize further analyses and machine learning.

There is one categorical nominal variable in the original dataset: DGN. It has seven possible values (DGN1, DGN2, DGN3, DGN4, DGN5, DGN6, and DGN8) with no ordinal relationship. For such variables, the integer encoding that was carried out with the Zubrod score and Tumor size is inadequate, as allowing the machine learning model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories). Hence, the **one-hot encoding** was used to convert the nominal variable with multiple categories (factors) into multiple variables (Dg1, Dg2, Dg3, Dg4, Dg5, Dg6, and Dg6, each represented by an integer value 0 or 1).

Table 4: lc dataset after pre-processing (first 6 rows)

FVC	FEV1	Zubrod	Pain	Hemoptysis	Dyspnea	Cough	Weakness	Tumor_size	T2DM	MI	PAD	Smoking	Asthma	Age	Death	Dg3	Dg2	Dg4	Dg6	Dg5	Dg8	Dg1
2.88	2.16	1	0	0	0	1	1	4	0	0	0	1	0	60	0	0	1	0	0	0	0	0
3.40	1.88	0	0	0	0	0	0	2	0	0	0	1	0	51	0	1	0	0	0	0	0	0
2.76	2.08	1	0	0	0	1	0	1	0	0	0	1	0	59	0	1	0	0	0	0	0	0
3.68	3.04	0	0	0	0	0	0	1	0	0	0	0	0	54	0	1	0	0	0	0	0	0
2.44	0.96	2	0	1	0	1	1	1	0	0	0	1	0	73	1	1	0	0	0	0	0	0
2.48	1.88	1	0	0	0	1	0	1	0	0	0	0	0	51	0	1	0	0	0	0	0	0

2.2 Data Exploration

2.2.1 Missing values

The dataset contains no missing values. Observations with missing values were removed by the authors of the original dataset [9].

2.2.2 Outliers

Only metric (non-categorical) variables are checked for outliers. These variables are *FVC*, *FEV1* and *Age*.

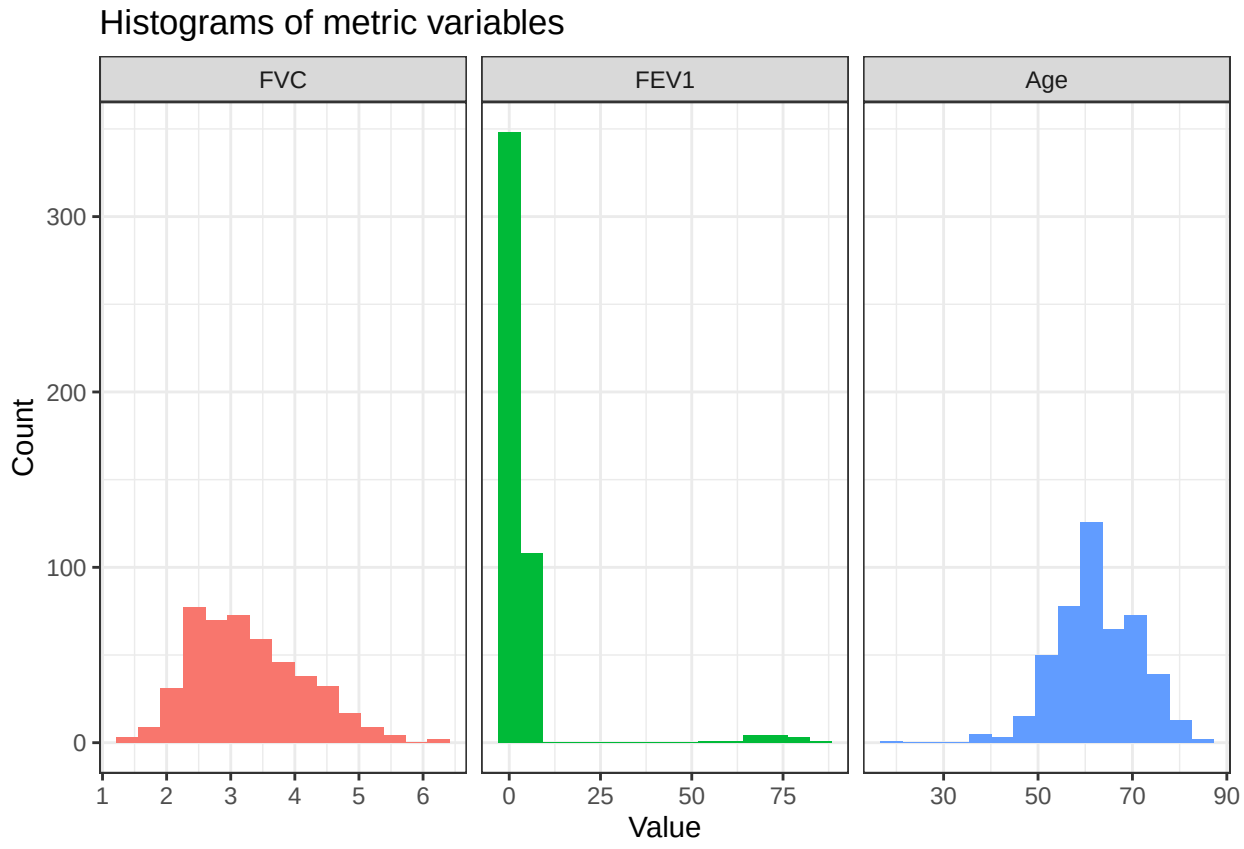

Figure 1: Histograms of metric variables

Figure 1 reveals several FEV1 outliers. We can explore this further with the Tukey's method which confirms that there are **15 FEV1 outliers, 1 Age outlier, and 0 FVC outliers**.

Table 5 and Table 6 present all outliers, confirmed by the Tukey's method.

All identified outliers can be confirmed, since such values are not plausible in real-life setting.

Table 5: FEV1 Outliers

	FVC	FEV1	Age	Death
26	4.56	72.80	57	0
90	2.83	66.40	75	0
99	2.63	67.30	54	0
113	3.68	64.10	60	0
133	2.50	71.10	64	1
216	2.66	8.56	61	0
256	3.72	78.30	44	0
320	2.10	69.10	62	0
326	5.03	79.30	38	0
331	2.94	76.00	61	0
350	1.82	86.30	67	0
353	2.94	73.30	60	0
354	3.24	52.30	55	0
439	3.67	76.80	61	0
445	2.56	60.90	50	0

Table 6: Age Outlier

	FVC	FEV1	Age	Death
397	2.76	2.08	21	0

Normal predicted FEV1 value for a healthy 195 cm tall male, 18-25 years of age, is up to 5.17 L (reference https://vitalograph.com/resources/ers-normal-values). FEV1 decreases with increasing age and decreasing body height. As all FEV1 outliers highly exceed the normal FEV1 value, it seems reasonable to assume that these values are errors.

Similarly, the Age outlier is only 21 years old. Since lung cancer is very rare in patients so young, this is a likely error as well.

Outliers can have strongly negative impact on performance of machine learning models, therefore I decided to remove them from the dataset before proceeding.

There are **454 observations remaining in the cleaned dataset**, which should still be sufficient for the model development.

2.2.3 Distribution of variables

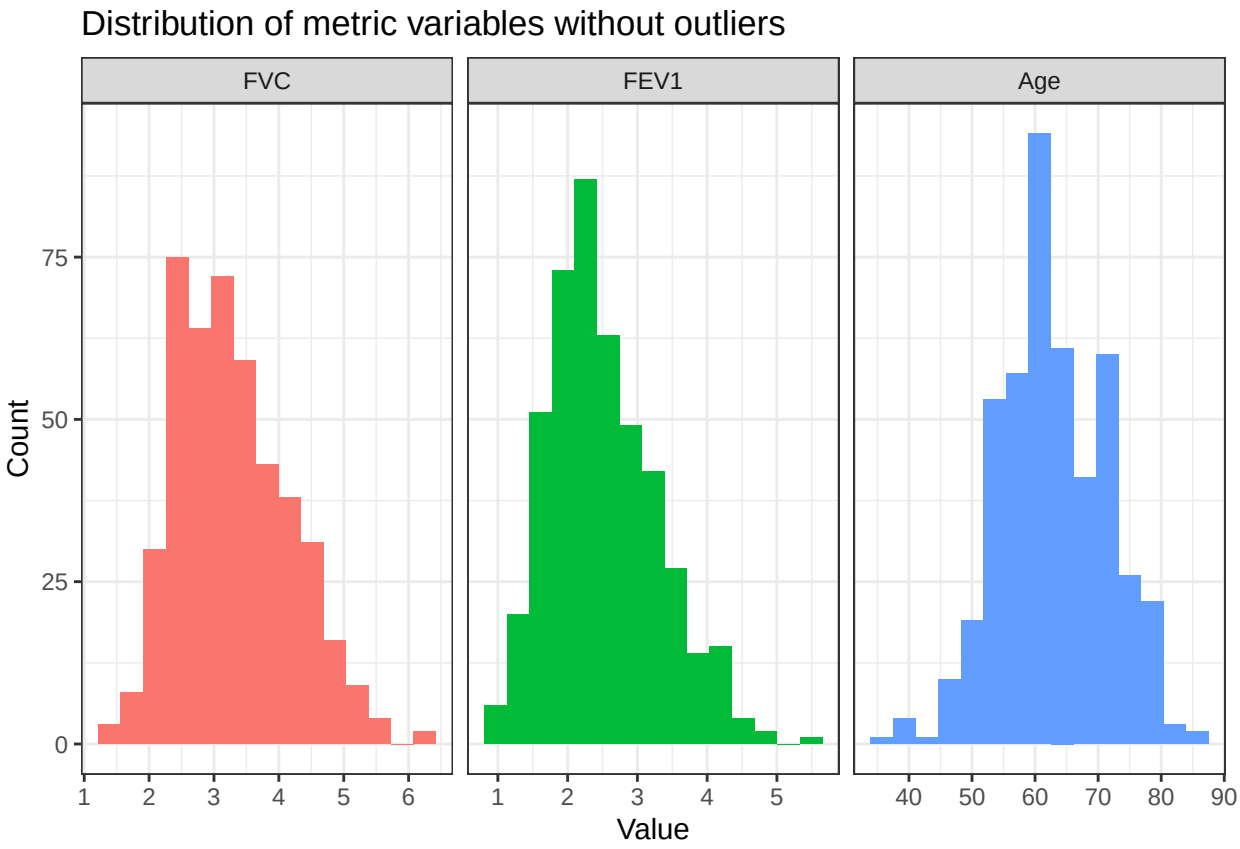

Figure 2: Distribution of metric variables without outliers

After the removal of outliers, distributions of all metric variables are slightly skewed, but close to the normal distribution.

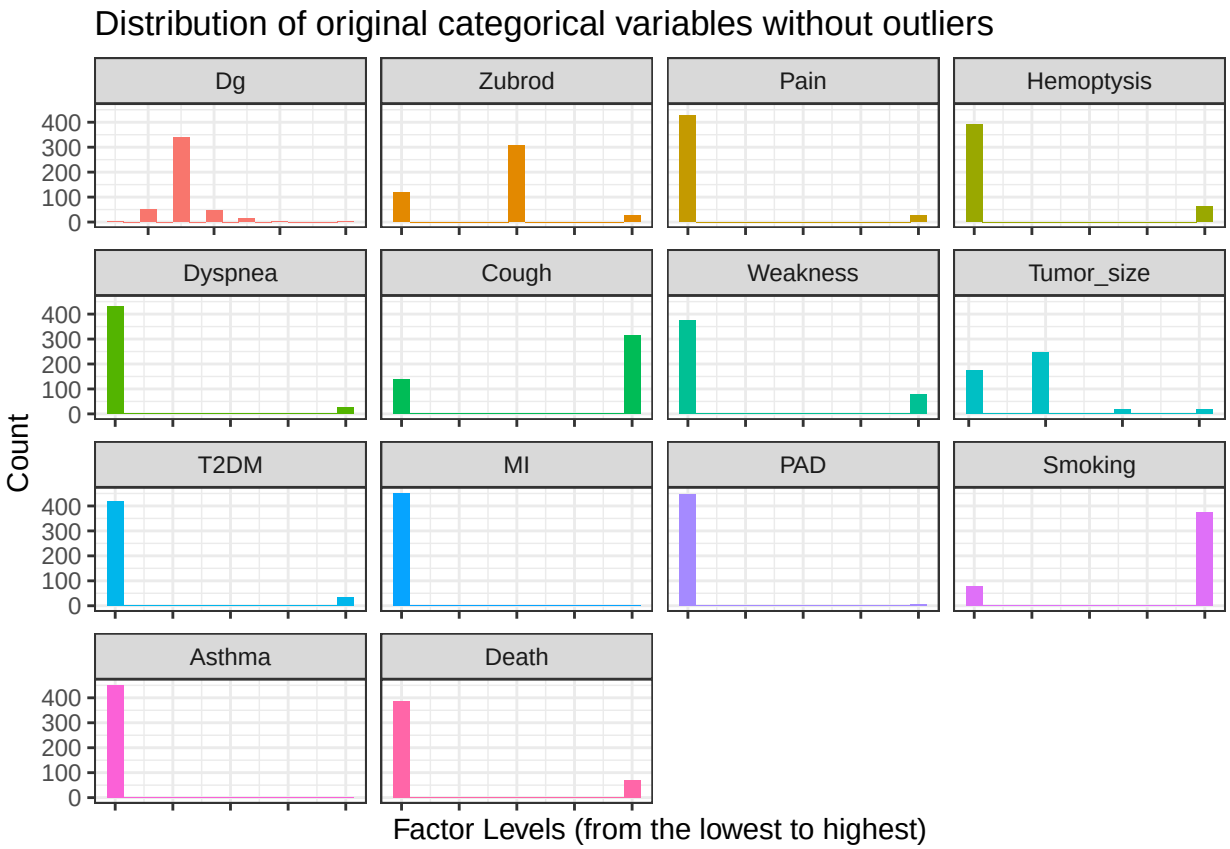

Figure 3: Distribution of original categorical variables without outliers

Please note that the values on the x axis are not shown to ease readability, which would otherwise be compromised due to differing subplot scales. Distributions of all binary variables are **highly imbalanced**, as discussed in previous sections. A strong imbalance is also seen for all three non-binary categorical variables. A large majority of observations have a *DGN3* diagnosis combination, a Zubrod score *1* and smaller tumor sizes of either *1* or *2*.

2.2.4 Correlation of variables

Correlations of variables need to be assessed, as machine learning algorithms assume that features are independent from each other.

The correlations, assessed by Pearson's correlation coefficients, are presented in the following plot. Given that most variables in the dataset are categorical, the Pearson's correlation coefficient may not be the best statistics for measuring their relationships. Nevertheless, it can give a quick view of relationships.

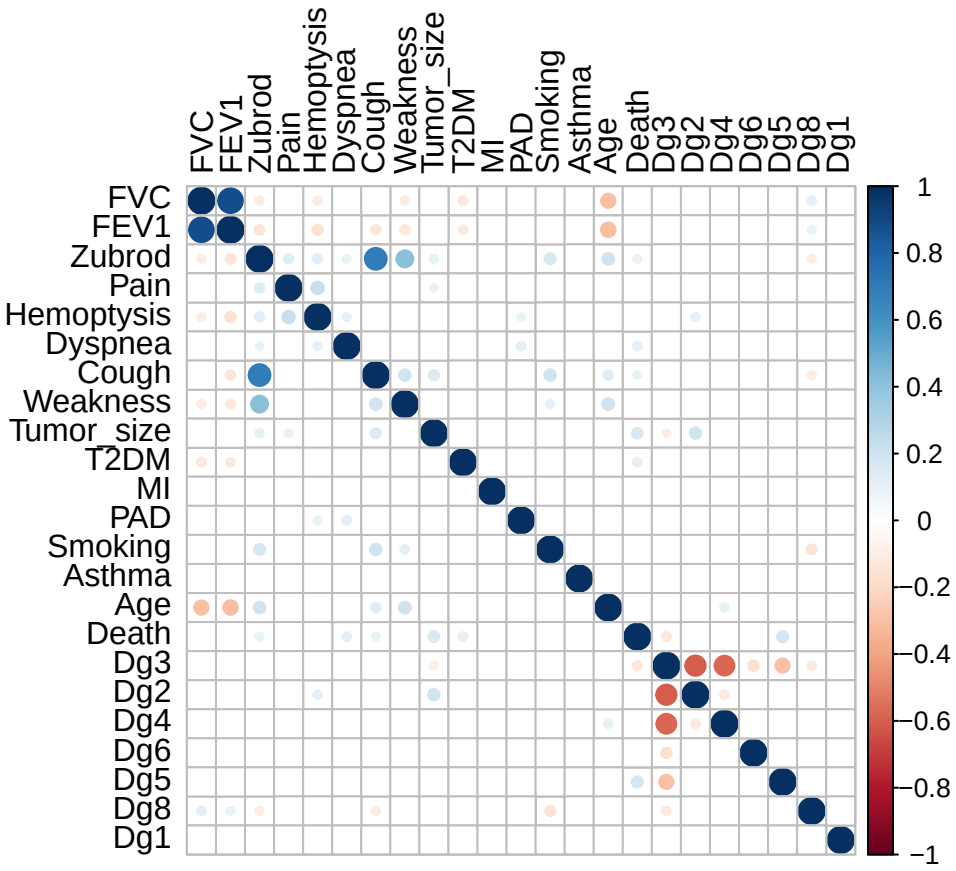

Figure 4: Correlation Plot

Please note that the circles are shown only when the correlation coefficients are statisticaly significant. Blank boxes indicate statistical insignificance. The size and the color of each circle correspond to the value of the correlation coefficient.

In general, observed correlations between variables seem to be low. There are only 4 variable pairs with strong correlation: FVC and FEV1, Cough and Zubrod, Dg2 and Dg3, Dg4 and Dg3. The correlation coefficients between them are:

- FVC and FEV1 (positive relationship)

```
## [1] 0.89
```

- Cough and Zubrod (positive relationship)

```
## [1] 0.69
```

- Dg2 and Dg3 (negative relationship)

```
## [1] -0.6
```

- Dg4 and Dg3 (negative relationship)

```
## [1] -0.57
```

A strong correlation between FVC and FEV1 is expected, as both measure a volume of exhaled air during a forced breath.

A strong correlation between Cough and Zubrod score is also not surprising. As Zubrod score is a metric of a patient's performance status, cough would be expected to appear with a worse Zubrod score.

Correlations between the different diagnoses cannot be interpreted, since the original dataset does not provide detailed information about the constituents of each combination of diagnoses.

The correlation plot also shows that none of the variables show a high degree correlation with the target variable (Death).

The *caret R package* includes the *findCorrelation* function, which searches through a correlation matrix and returns columns to be removed in order **to reduce pair-wise correlations**. For the presented lc dataset correlation matrix, it suggests the removal of the following variables:

```
## [1] "Zubrod" "Dg3"    "FEV1"
```

I explored how the removal of the suggested variables influenced the model performance *after data balancing and feature engineering*, as described in the following sections.

Relationship between FVC, FEV1 and Death

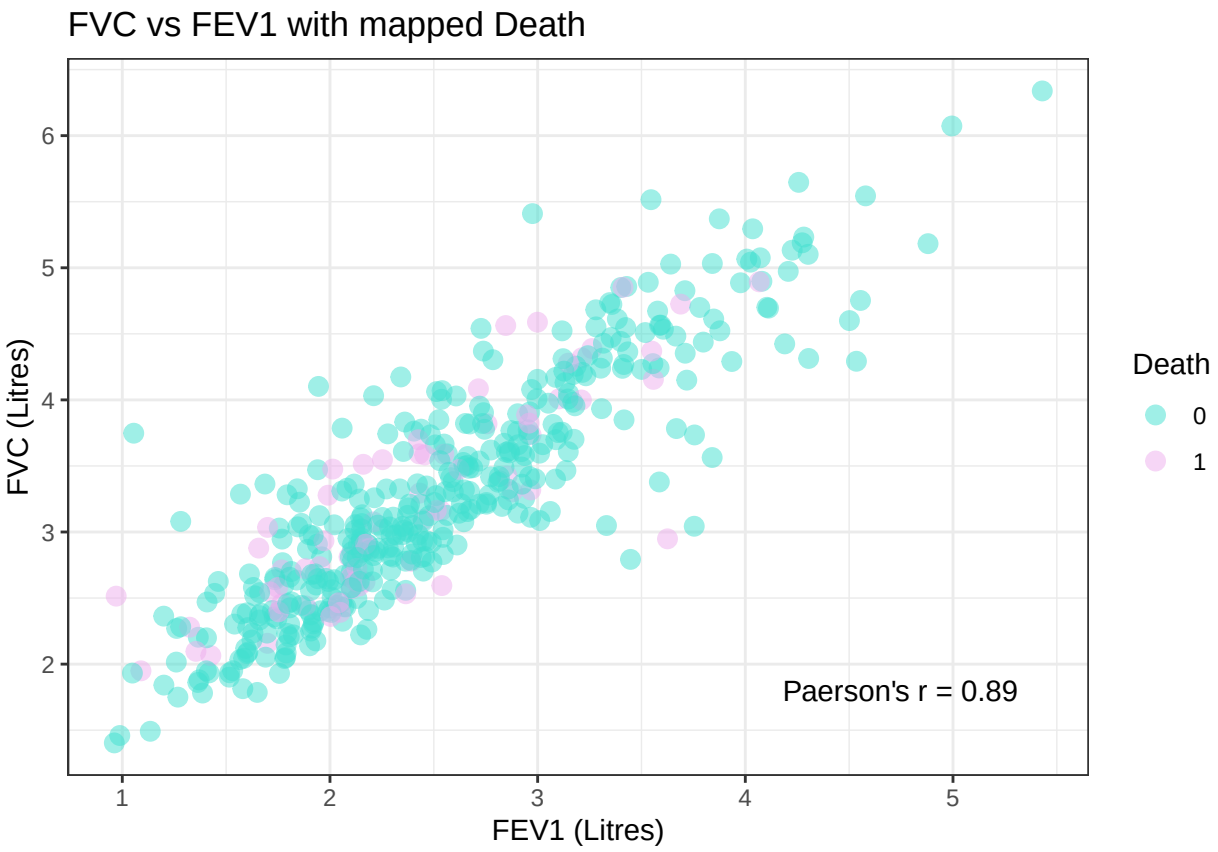

Figure 5: FVC vs FEV1 with mapped Death

A strong positive correlation between FVC and FEV1 is visible. There is no clear relationship between FVC or FEV1 and Death after the surgery. As already noted, the data is highly imbalanced and death (purple points) is much less prevalent than survival (green points).

Relationship between FVC, Age and Death

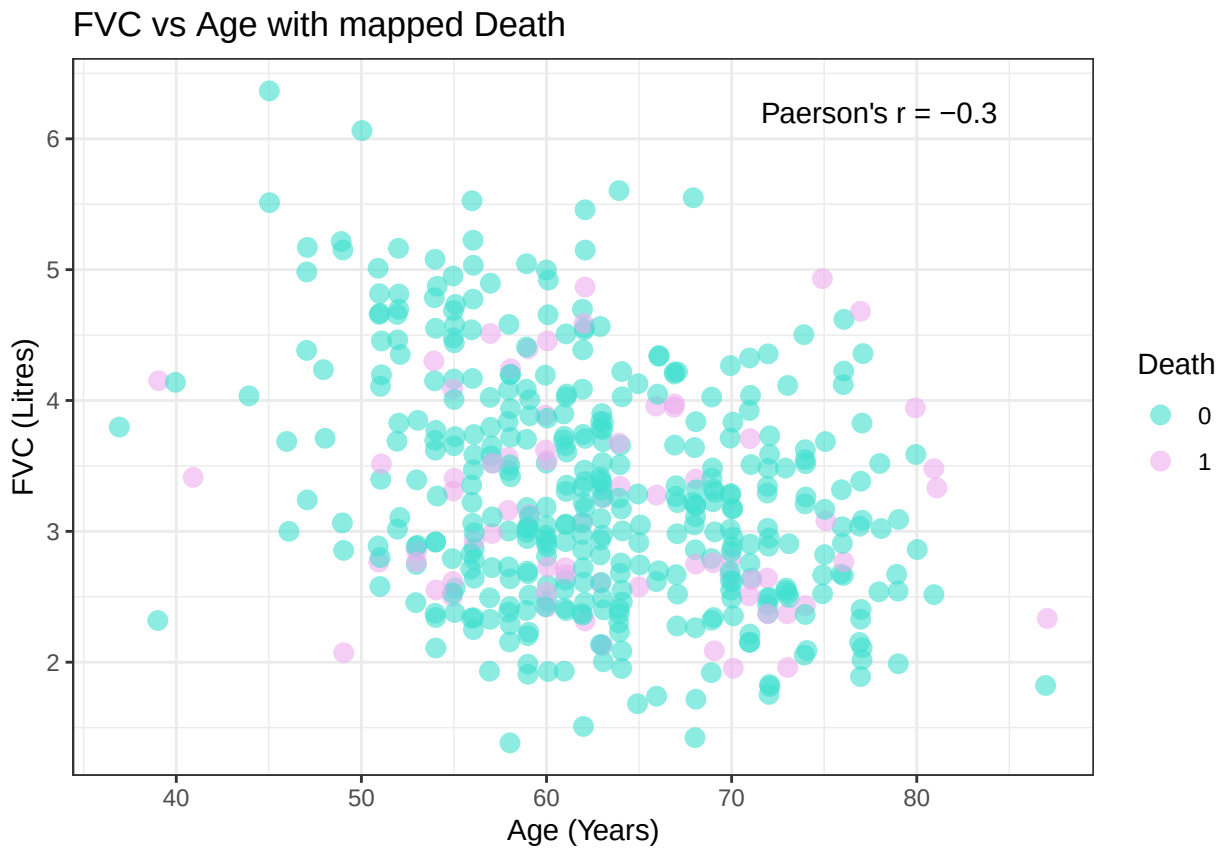

Figure 6: FVC vs Age with mapped Death

A weak negative correlation between FVC and Age is seen. This is expected, as the lung capacity decreases with ageing. There is no clear relationship between FVC or Age and Death after the surgery.

Relationship between Zubrod score, Tumor Size and Death

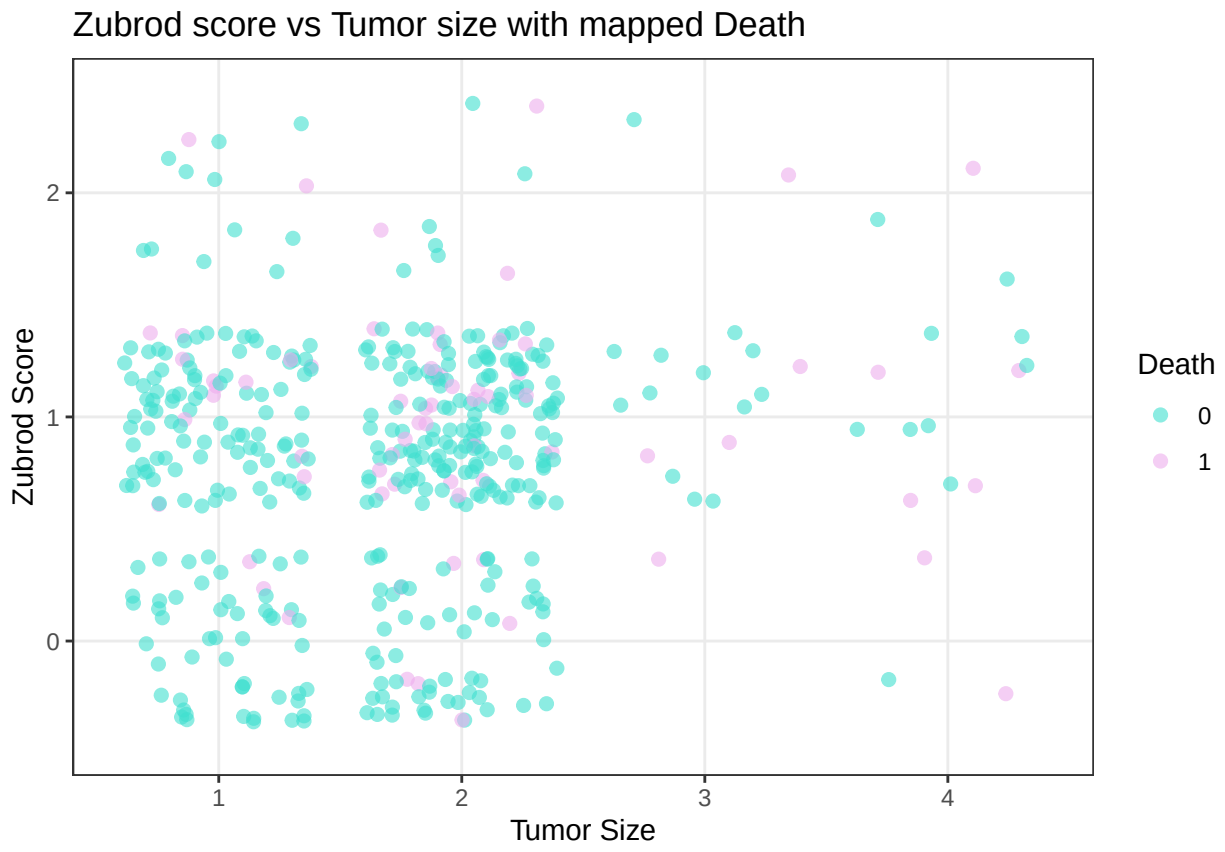

Figure 7: Zubrod score vs Tumor size with mapped Death

As already noted, the data is highly imbalanced. Most of the observations have a Zubrod score *1* and tumor sizes of either *1* or *2*. There is a very weak correlation between the two variables, where:

```
## [1] "Paerson's r = 0.1"
```

With such high data imbalance it is not possible to reveal any potential relationship between the outcome of death and either of the ordinal variables.

Relationship between FEV1/FVC Ratio, Age and Death

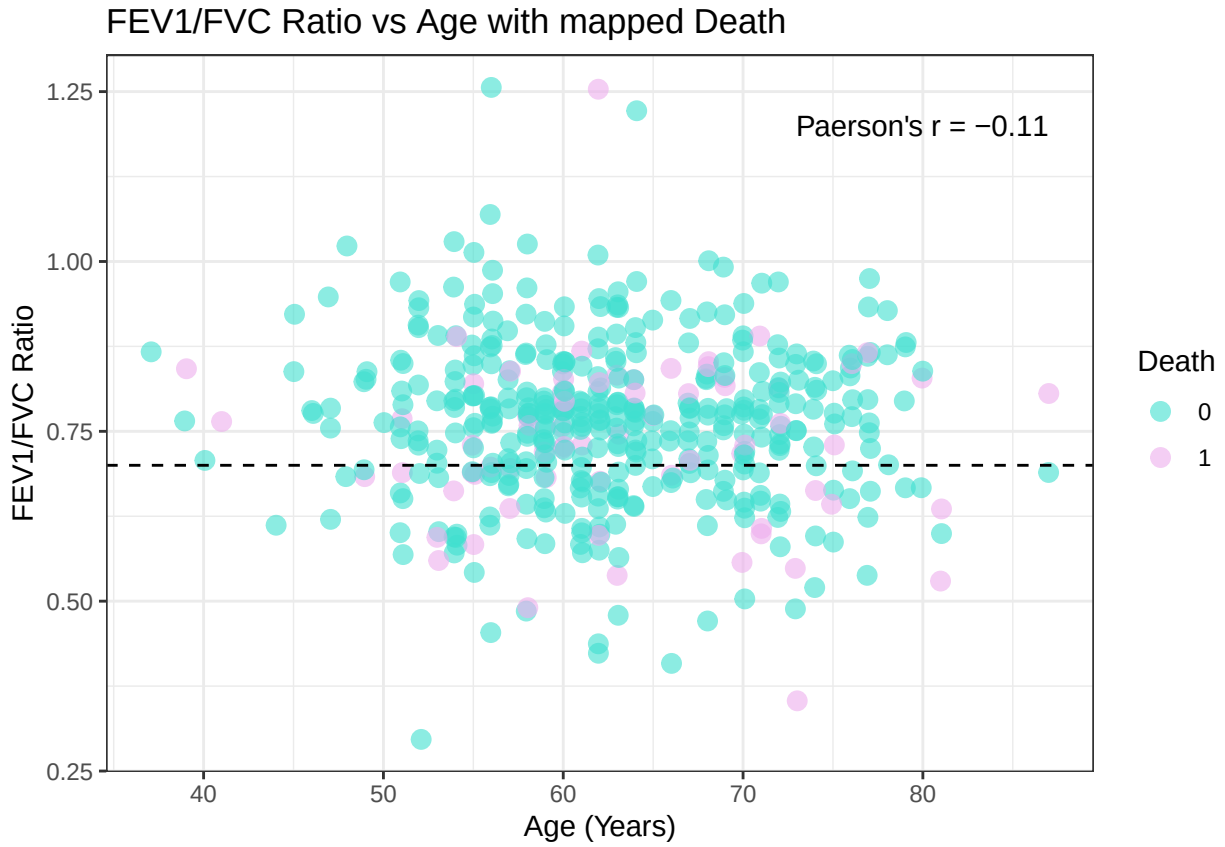

Figure 8: FEV1/FVC Ratio vs Age with mapped Death

FEV1/FVC ratio, also known as Tiffeneau-Pinelli index, is a calculated ratio used in the diagnosis of obstructive and restrictive lung disease. The diagnosis of COPD (Chronic Obstructive Pulmonary Disease) is made when the FEV1/FVC ratio is less than 0.7. COPD is an independent risk factor for lung carcinoma and has a marked effect on a patient's quality of life. The plot above shows its very weak negative correlation with Age. Please note that **the dotted line on the plot marks the FEV1/FVC ratio of 0.7**, hence the points below this line represent values in COPD patients. There is no clear relationship of FEV1/FVC ratio to death, where:

```
## [1] "Paerson's r = -0.08"
```

This could potentially change if the data would be more balanced, so this observation needs to be interpreted with caution.

Relationship between COPD, Smoking and Death

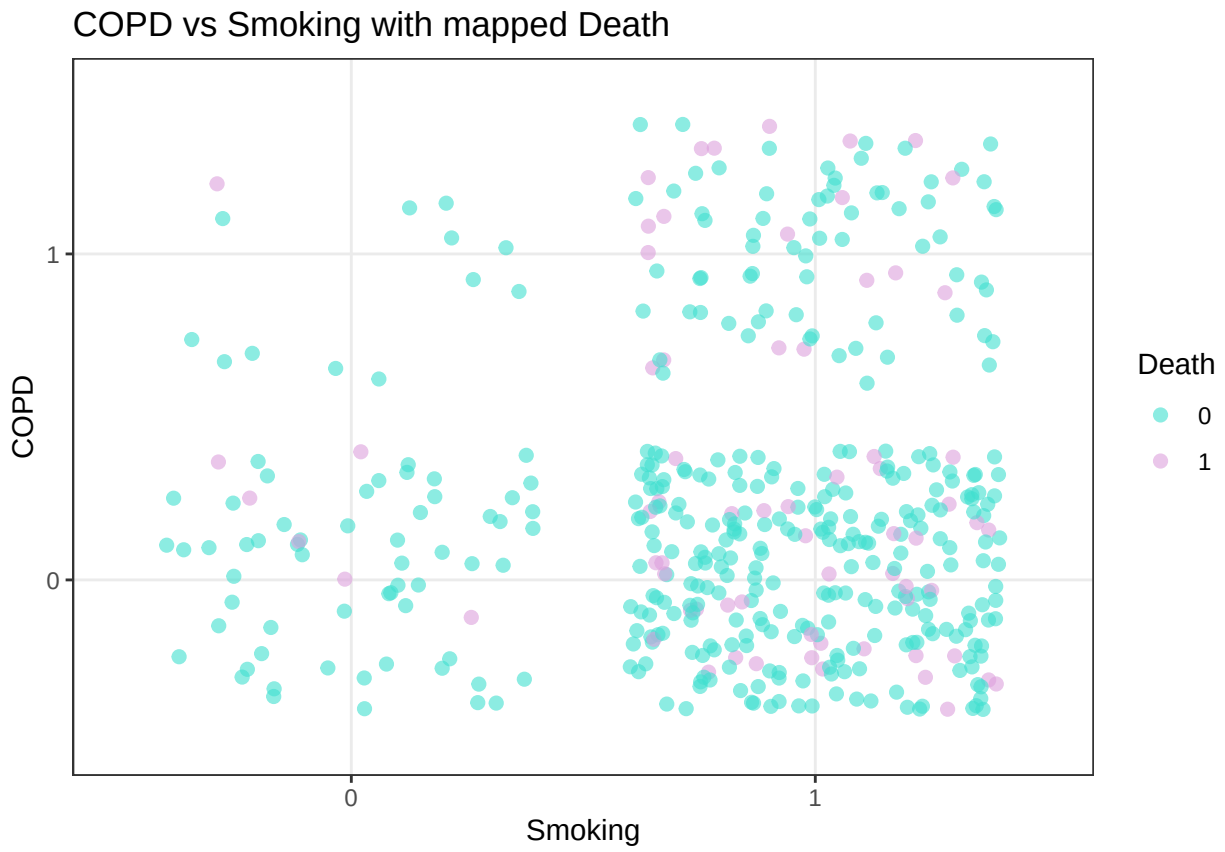

Figure 9: COPD vs Smoking with mapped Death

Imbalanced distribution of all variables is prominent once again. No clear relationship can be seen between COPD and smoking, although smoking is a well known risk factor for the development of the COPD. This indicates that it is not safe to make any conclusions with such a high degree of data imbalance. The calculated correlation between COPD and Death is very weak:

```
## [1] "Paerson's r = 0.06"
```

2.3 Data Analysis

I built and evaluated several supervised machine learning classification models. Prior to any further processing, data were split into training and testing subsets (80% of data for training, 20% for testing). The *lc_train* dataset contains 363 observations, the *lc_test* dataset contains 91 observations. All models were tuned by **10-fold cross-validation**, using full training dataset, with the goal of optimizing the ROC. The *set.seed* R function was used in all instances, to enable reproducibility of the results.

2.3.1 Model Building And Evaluation On The Cleaned Original Dataset

Three models were evaluated on the cleaned original lc dataset: Naive Bayes, Random Forest, and eXtreme Gradient Boosting (XGBoost). Their basic metrics (comparing model predictions with the test dataset) are presented in the following table.

Table 7: Basic Metric Scores per Model - Cleaned Original Dataset

Metric	Naive Bayes	Random Forest	XGBoost
Accuracy	0.8131868	0.8131868	0.7472527
Sensitivity	1.0000000	1.0000000	0.9054054
Specificity	0.0000000	0.0000000	0.0588235
Precision	0.8131868	0.8131868	0.8072289
Recall	1.0000000	1.0000000	0.9054054
F1	0.8969697	0.8969697	0.8535032
Prevalence	0.8131868	0.8131868	0.8131868
Balanced_accuracy	0.5000000	0.5000000	0.4821145

Table 8: Mean AUROC per Model - Cleaned Original Dataset

Model	Mean AUROC
NaiveBayes	0.6375806
RandomForest	0.6709409
XGBoost	0.6279704

The evaluated models have basically no discriminatory power. Due to highly imbalanced data, the models simply predict the most prevalent outcome (patient being alive at 1 year after the surgery) in almost all cases. Hence, the Naive Bayes and Random Forest have zero specificity. XGBoost performs only slightly better, but it is capable of predicting the non-prevalent outcome of death only in rare instances.

The training AUROC values (derived from the model, without comparing the predictions with the test dataset), as calculated by the caret R package, are also very low, as demonstrated in the Table 8.

The MLeval R package was used to plot the training ROC curves and PRCs, and calculate their Areas Under Curves (AUCs). *Please note that the estimates, which these calculations are based on are slightly different from those derived by the caret R package and presented in the previous table.* A likely reason for this is that the packages use different methods of calculating the performance metrics across resamples. Therefore, the results presented in the plots (derived by MLeval) are *not comparable* to AUROCs derived by caret. Nevertheless, these plots can give us an estimation on the model performance with the use of different datasets.

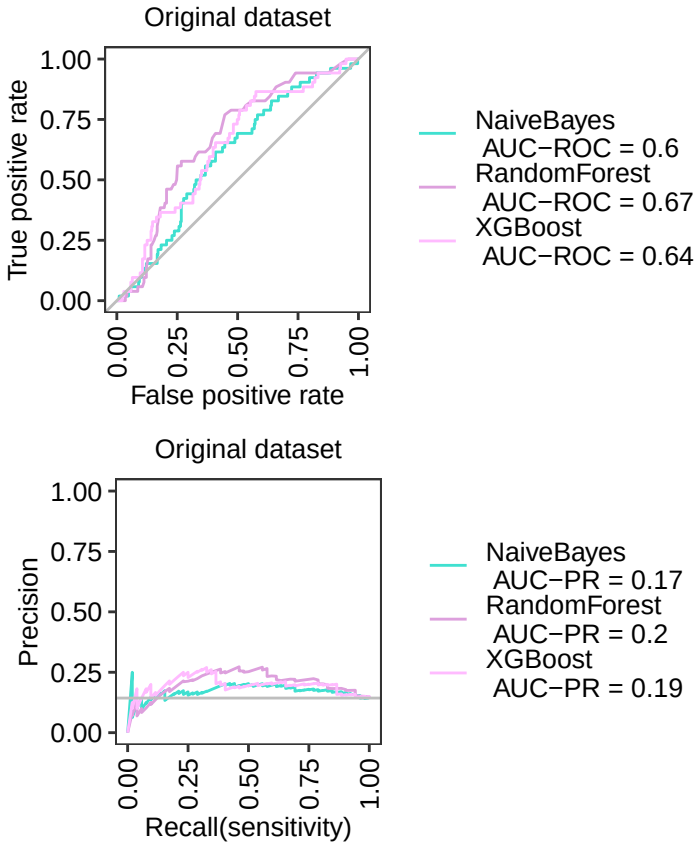

Figure 10: ROC Curves, PRCs, AUCs - Cleaned Original Dataset

Both, AUROC and AUPRC values, are very low. This is a clear indicator that we need to balance our dataset to improve the performance, and prior to evaluating additional machine learning models.

2.3.2 Data Balancing

Data balancing was performed on the *lc_train* dataset with the SMOTE (Synthetic Minority Over-sampling Technique) method. The SMOTE method generates new samples through interpolation of nearby samples [7].

By using the SMOTE, the target variable is much more balanced:

```
##
## Alive  Dead
##   208   156
```

However, the imbalance of other binary variables is not corrected.

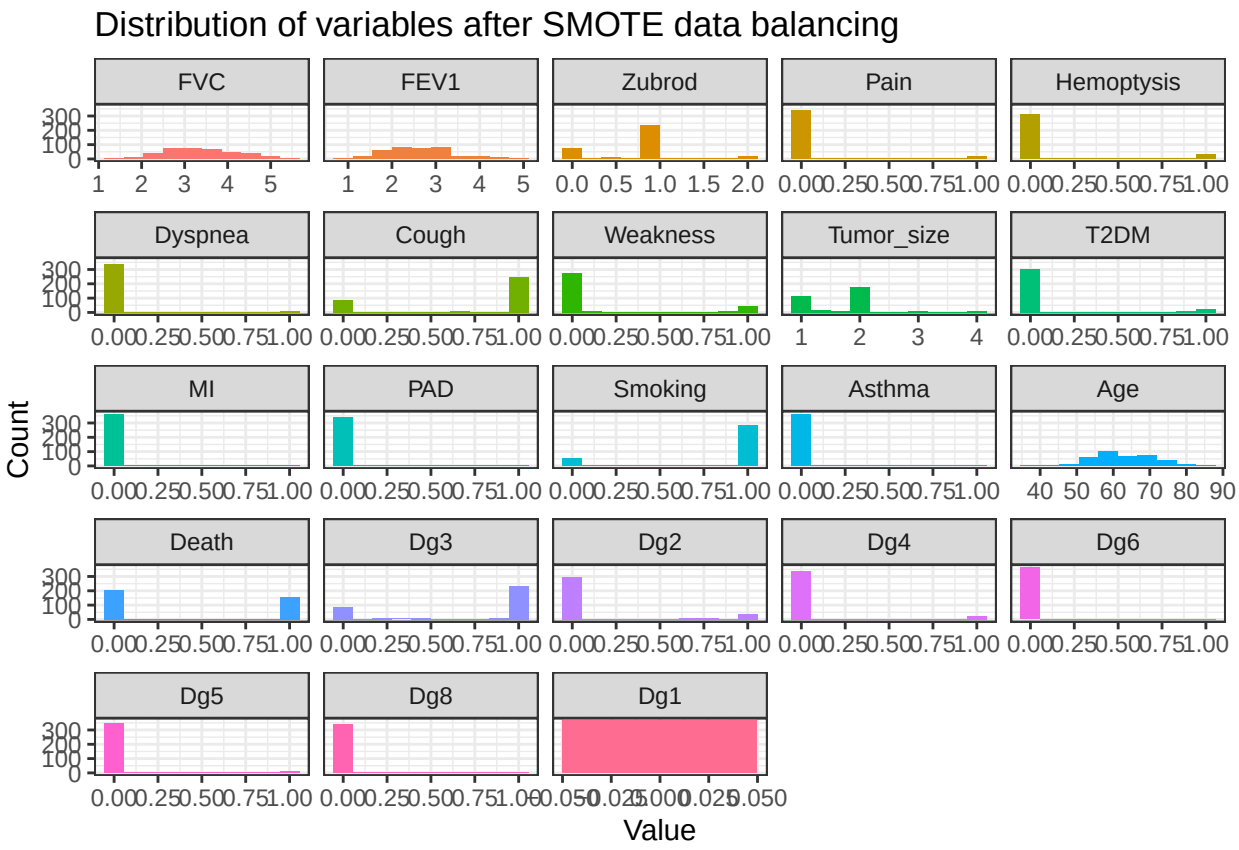

Figure 11: Distribution of variables after data balancing (training subset)

Table 9 shows the estimates of the basic metrics for the Random Forest and XGBoost model after data balancing with SMOTE. There is a noticeable improvement in the majority of metrics. The most prominent improvements for both models are seen with Specificity/Precision.

Table 10 shows a substantial increase in AUROC (as calculated by the caret R package) for both evaluated models after the SMOTE data balancing.

Table 9: Basic Metric Scores per Model - After SMOTE Data Balancing

Metric	Random Forest - Balanced	XGBoost - Balanced
Accuracy	0.6923077	0.7142857
Sensitivity	0.7837838	0.8243243
Specificity	0.2941176	0.2352941
Precision	0.8285714	0.8243243
Recall	0.7837838	0.8243243
F1	0.8055556	0.8243243
Prevalence	0.8131868	0.8131868
Balanced_accuracy	0.5389507	0.5298092

Table 10: Mean AUROC per Model - Smote Balanced Dataset

Model	Mean AUROC
RandomForest_balanced	0.9212937
XGBoost_balanced	0.9222857

Similarly, the presented plots in Figure 12 show a substantial increase in AUROC and AUPRC (by MLeval package) after the SMOTE data balancing. This implies that both models have gained discriminatory power. *As previously noted, the results presented in the plots (derived by MLeval) are not comparable to those in the table with basic metric scores (derived by caret) due to differences in the calculation of estimates.* The MLeval estimated specificity of both models is much higher than the caret estimation.

To improve the model performance even further, I implemented the feature engineering technique, as described in the following section.

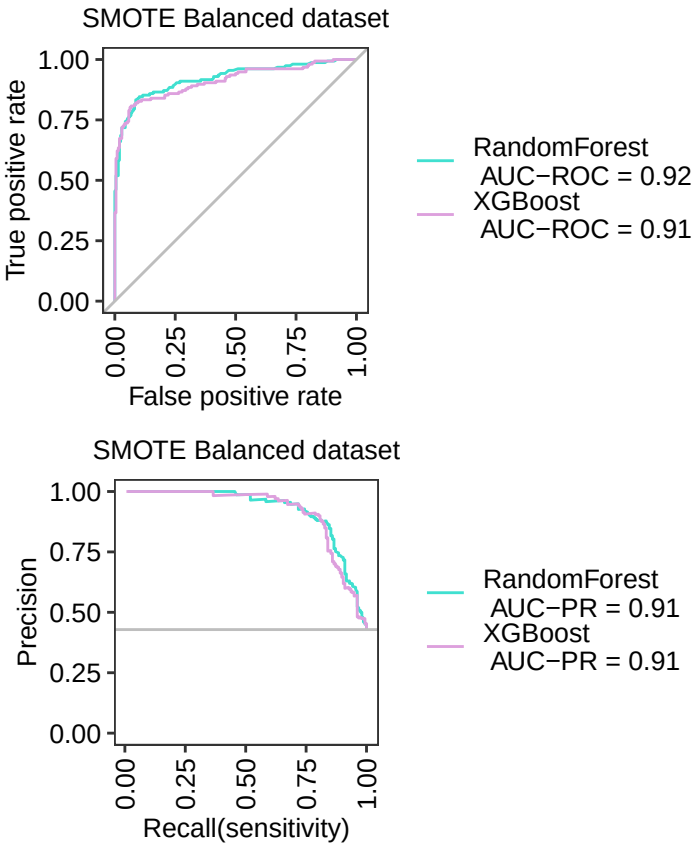

Figure 12: ROC Curves, PRCs, AUCs - After SMOTE Data Balancing

2.3.3 Feature Engineering

Besides the highly imbalanced data, another challenge lies in the number of features available for in-depth analysis of the data. It was previously shown that there was much difficulty in drawing correlations between the various features. In previous research [7], it was shown that with the creation of new features it is possible to obtain a better representation of the underlying relationships among various features.

To derive important features that could be used for creating new features, I built the Random Forest and XGBoost as feature ranking models on the cleaned original dataset (prior to data balancing with SMOTE).

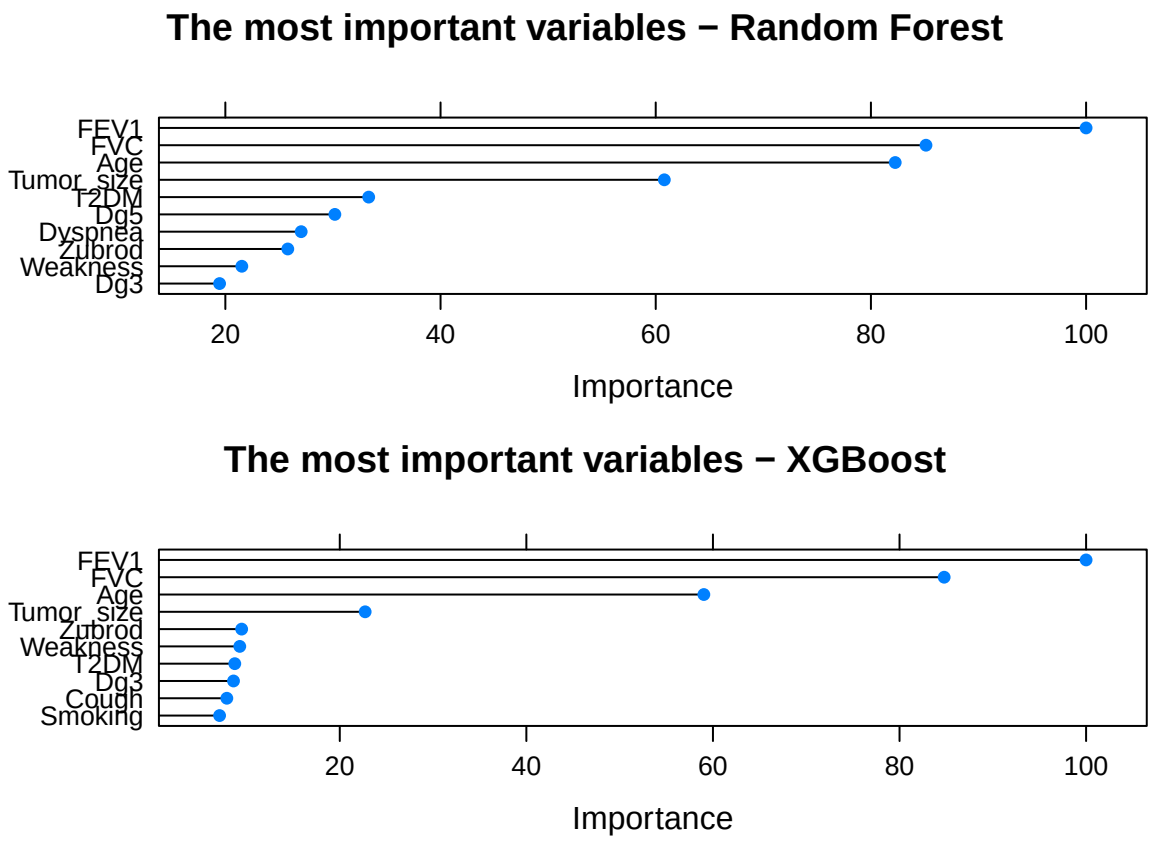

Figure 13: Feature Importance - Original Dataset

The first 4 most important features are the same in both evaluated ranking models. All other features have much lower impact on the model's performance. I used FEV1, FVC, Age, and Tumor_size to create 9 new features in total, by performing various basic mathematical calculations. The 9 engineered new features are:

- FEV1/FVC (Tiffeneau-Pinelli index)
- FVC x FEV1
- FVC x FEV1^2
- FVC^2 x FEV1
- FVC^2
- Age x FVC
- Age x FEV1
- Age / Tumor_size
- FVC x Tumor_size

After the feature engineering, feature importance was reevaluated with the best performing model so far, the XGBoost.

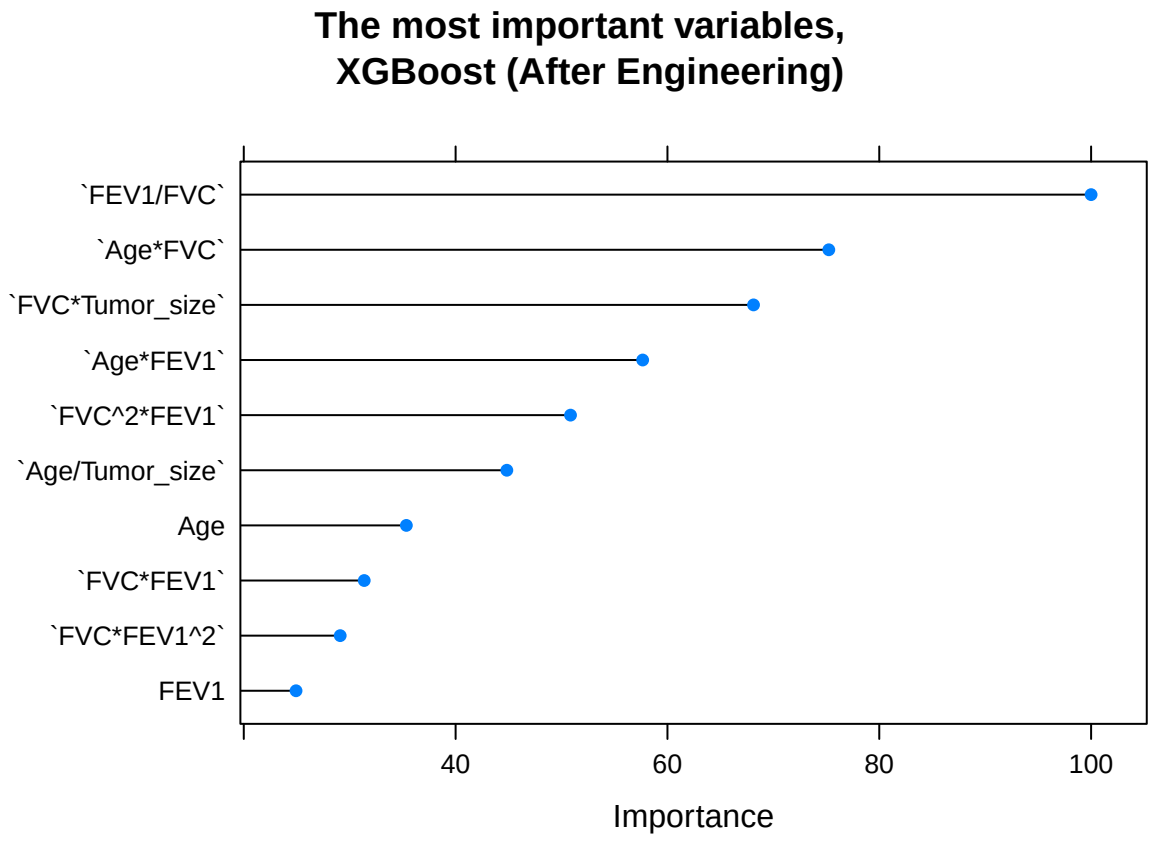

Figure 14: Feature Importance - Original Dataset After Feature Engineering

8 out of 10 most important features, as ranked by the XGBoost model on the original imbalanced dataset, were manually created. Several of them are strongly correlated. The *findCorrelation* function suggests the removal of the following variables (the correlation cutoff was set to 0.5):

```
##  [1] "FVC*FEV1"       "FEV1"           "FVC*FEV1^2"     "FVC^2*FEV1"
##  [5] "FVC^2"          "FVC"            "Age*FEV1"       "Age*FVC"
##  [9] "FVC*Tumor_size" "Zubrod"         "Age/Tumor_size" "Dg3"
```

The impact of the removal of the suggested variables on the model performance is assessed *after data balancing and feature engineering.*

2.3.4 Model Building And Evaluation On The Final Balanced Dataset With Added Features

2.3.4.1 XGBoost Model

Table 11: Basic Metric Scores - XGB, Original and Modified Datasets

Metric	XGBoost	XGBoost - Balanced	XGBoost - Final	XGBoost - Final (No_Corr)
Accuracy	0.7472527	0.7142857	0.7142857	0.7472527
Sensitivity	0.9054054	0.8243243	0.8243243	0.8513514
Specificity	0.0588235	0.2352941	0.2352941	0.2941176
Precision	0.8072289	0.8243243	0.8243243	0.8400000
Recall	0.9054054	0.8243243	0.8243243	0.8513514
F1	0.8535032	0.8243243	0.8243243	0.8456376
Prevalence	0.8131868	0.8131868	0.8131868	0.8131868
Balanced_accuracy	0.4821145	0.5298092	0.5298092	0.5727345

Table 12: Mean AUROC for XGBoost per each dataset

Model	Mean AUROC
XGB_orig	0.6279704
XGB_balanced	0.9222857
XGB_final	0.9323095
XGB_final_No_Cor	0.9051468

Table 11 and Table 12 demonstrate that the highest balanced accuracy was reached, when the XGBoost model was built on the final dataset with removed variables with strong correlation. The highest mean AUROC was reached on the final model without the removal of any variables. It is interesting that the addition of newly engineered features has shown no influence on the balanced accuracy, however, it caused a slight increase in the mean AUROC.

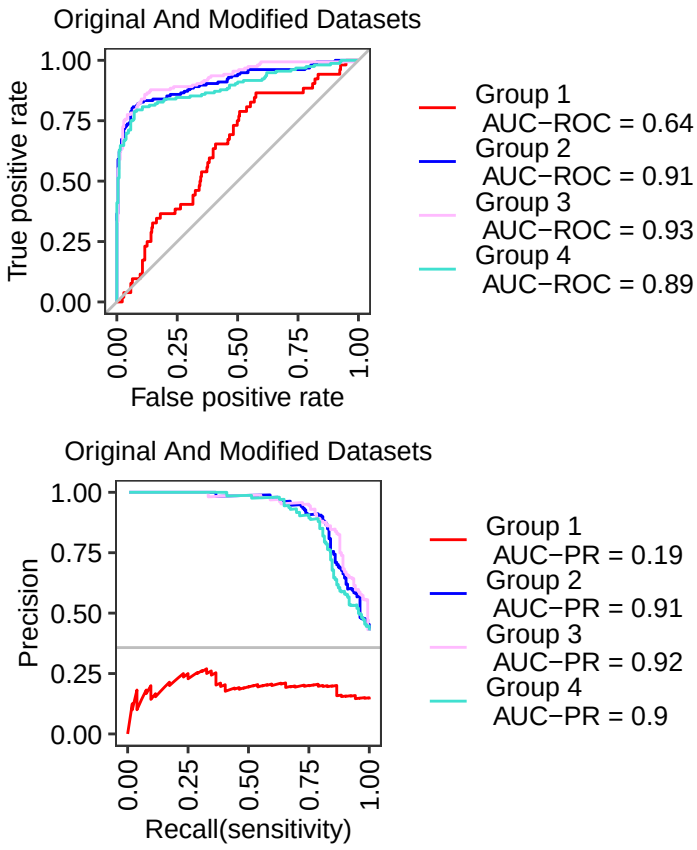

Figure 15: ROC Curves, PRCs, AUCs - XGB on different datasets

Figure 15 includes the following groups:

- Group 1 = XGB_orig,
- Group 2 = XGB_balanced,
- Group 3 = XGB_final,
- Group 4 = XGB_final_NoCor

The plots confirms the described findings. A strong positive influence of data balancing on the model's performance is evident.

However, a look at the respective confusion matrices reveals that the model is still rather unreliable, despite the noticeable improvement in its performance.

Confusion matrix of the XGBoost model on the cleaned original dataset:

```
##            Reference
## Prediction Alive Dead
##      Alive    67   16
##      Dead      7    1
```

Confusion matrix of the XGBoost model on the balanced dataset:

```
##            Reference
## Prediction Alive Dead
```

```
##      Alive    61   13
##      Dead     13    4
```

Confusion matrix of the XGBoost model on the final dataset, after the removal of highly correlated variables:

```
##           Reference
## Prediction Alive Dead
##      Alive    63   12
##      Dead     11    5
```

2.3.4.2 Random Forest Model

Table 13 shows improved Random Forest model performance after data balancing with SMOTE, most prominent with the increase of specificity and balanced accuracy. The addition of the new engineered variables has not caused a convincing further improvement (there is an increase in sensitivity and accuracy, but a drop in specificity and balanced accuracy).

Table 13: Basic Metric Scores - RF, Original and Modified Datasets

Metric	Random Forest	Random Forest - Balanced	RF - Final
Accuracy	0.8131868	0.6923077	0.7582418
Sensitivity	1.0000000	0.7837838	0.9054054
Specificity	0.0000000	0.2941176	0.1176471
Precision	0.8131868	0.8285714	0.8170732
Recall	1.0000000	0.7837838	0.9054054
F1	0.8969697	0.8055556	0.8589744
Prevalence	0.8131868	0.8131868	0.8131868
Balanced_accuracy	0.5000000	0.5389507	0.5115262

This improvement is evident also from the confusion matrices, despite that the model remains unreliable. Confusion matrix of the Random Forest model on the cleaned original dataset:

```
##            Reference
## Prediction Alive Dead
##      Alive    74   17
##      Dead      0    0
```

Confusion matrix of the Random Forest model on the balanced dataset:

```
##            Reference
## Prediction Alive Dead
##      Alive    58   12
##      Dead     16    5
```

Confusion matrix of the Random Forest model on the final dataset (without the removal of highly correlated variables):

```
##            Reference
## Prediction Alive Dead
##      Alive    67   15
##      Dead      7    2
```

2.3.4.3 Support Vector Machine (SVM) Model

Confusion matrix and basic metrics of the SVM model:

```
##           Reference
## Prediction Alive Dead
##      Alive    55   13
##      Dead     19    4
```

```
## # A tibble: 8 x 2
##   Metric            `SVM - Final`
##   <chr>                     <dbl>
## 1 accuracy                  0.648
## 2 sensitivity               0.743
## 3 specificity               0.235
## 4 precision                 0.809
## 5 recall                    0.743
## 6 f1                        0.775
## 7 prevalence                0.813
## 8 balanced_accuracy         0.489
```

2.3.4.4 Model Averaged Neural Network Model

Overall, this is the best performing model of all the evaluated models. A particulary great improvement is achieved in specificity and balanced accuracy. This model performs better, when strongly correlated variables are not removed.

Table 14: Basic Metric Scores - NeuralNetwork

Metric	AvNNet - Final	AvNNet - Final_NoCor
Accuracy	0.7472527	0.6923077
Sensitivity	0.8108108	0.7567568
Specificity	0.4705882	0.4117647
Precision	0.8695652	0.8484848
Recall	0.8108108	0.7567568
F1	0.8391608	0.8000000
Prevalence	0.8131868	0.8131868
Balanced_accuracy	0.6406995	0.5842607

Confusion matrix of the Neural Network Model on the final dataset (without the removal of highly correlated variables):

```
##            Reference
## Prediction Alive Dead
##      Alive    60    9
##      Dead     14    8
```

Confusion matrix of the Neural Network Model on the final dataset, after the removal of highly correlated variables:

```
##            Reference
## Prediction Alive Dead
##      Alive    56   10
##      Dead     18    7
```

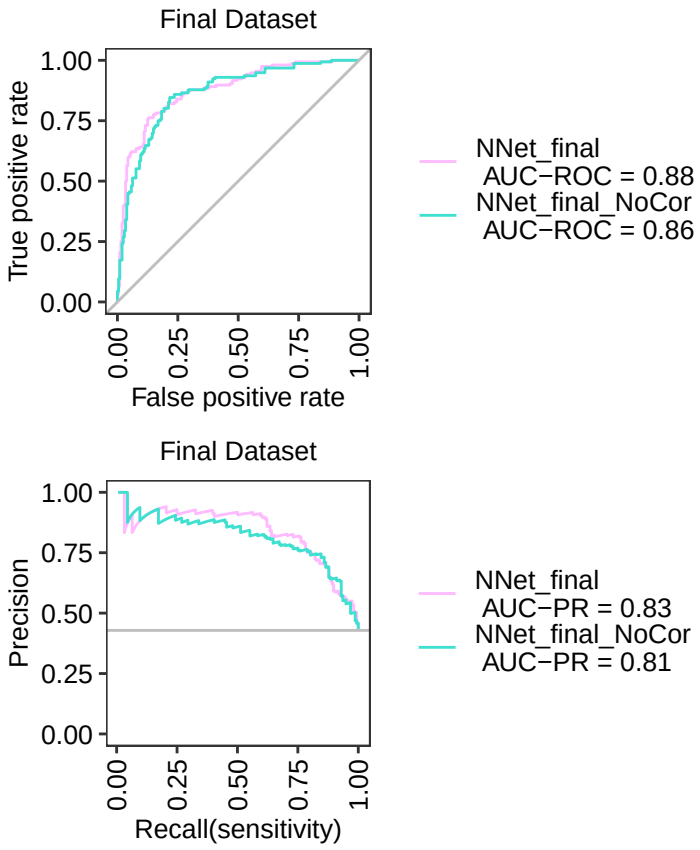

Figure 16: ROC Curves, PRCs, AUCs - Neural Network on The Final Dataset

Both, AUROC and AUPRC values, are higher when strongly correlated variables are kept. In comparison with the best XGBoost model, the AUROC and AUPRC values are lower.

2.3.4.5 The Impact Of One Hot Encoding On The Model Performance

To evaluate the impact of one-hot encoding, I explored the cleaned original dataset where **Dg variable** was not manipulated by the one-hot encoding. All machine learning procedures were performed in the same way as previously described; i.e. checking the correlations between variables, splitting the dataset into the training and testing subset, data balancing, and feature engineering. **Due to separate splitting of the original dataset, it needs to be emphasized that this section provides only a rough estimation on the impact of One-Hot Encoding, and the results are not directly comparable.**

Only two highly correlated variables were identifed, namely the FEV1 and Zubrod. Since these were the same as in the one-hot encoded dataset, the final model was not re-evaluated for the impact of the removal of highly correlated variables.

Table 15: Basic Metric Scores - Impact of One-Hot Encoding

Metric	XGBoost - DGN original	XGBoost	XGBoost - DGN Final	XGBoost - Final
Accuracy	0.8131868	0.7472527	0.7362637	0.7142857
Sensitivity	1.0000000	0.9054054	0.8243243	0.8243243
Specificity	0.0000000	0.0588235	0.3529412	0.2352941
Precision	0.8131868	0.8072289	0.8472222	0.8243243
Recall	1.0000000	0.9054054	0.8243243	0.8243243
F1	0.8969697	0.8535032	0.8356164	0.8243243
Prevalence	0.8131868	0.8131868	0.8131868	0.8131868
Balanced_accuracy	0.5000000	0.4821145	0.5886328	0.5298092

Table 16: Mean AUROC - Impact of One-Hot Encoding

Model	Mean AUROC
XGB_dgn_orig	0.6296102
XGB_orig	0.6279704
XGB_dgn_final	0.9300238
XGB_final	0.9323095

Table 15 and Table 16 show that One-Hot Encoding has not caused a convincing improvement of the model's performance. On the contrary, the highest specificity and balanced accuracy were reached on the final dataset without the One-Hot Encoding (i.e. *DGN final* dataset). However, the AUROC values on both final datasets were comparable.

The observed strong positive influence of data balancing on the model's performance is consistent in all settings.

Table 17: Basic Metric Scores - All Top Models

Metric	XGBoost - Final	XGBoost - Final (No_Corr)	RF - Final	SVM - Final	AvNNet - Final	AvNNet - Final_NoCor
Accuracy	0.7142857	0.7472527	0.7582418	0.6483516	0.7472527	0.6923077
Sensitivity	0.8243243	0.8513514	0.9054054	0.7432432	0.8108108	0.7567568
Specificity	0.2352941	0.2941176	0.1176471	0.2352941	0.4705882	0.4117647
Precision	0.8243243	0.8400000	0.8170732	0.8088235	0.8695652	0.8484848
Recall	0.8243243	0.8513514	0.9054054	0.7432432	0.8108108	0.7567568
F1	0.8243243	0.8456376	0.8589744	0.7746479	0.8391608	0.8000000
Prevalence	0.8131868	0.8131868	0.8131868	0.8131868	0.8131868	0.8131868
Balanced_accuracy	0.5298092	0.5727345	0.5115262	0.4892687	0.6406995	0.5842607

3 Results

3.1 Basic Metrics

Combined basic metric results of all top performing models are presented. Overall, the best results were obtained with the neural network model with all available variables. This model was able to perform with sensitivity of 81%, specificity of 47%, precision of 87% and balanced accuracy of 64%.

The XGBoost model, recognized as the most optimal model in some of previous published research, achieved the best perfomance after the removal of highly correlated variables. It was able to perform with sensitivity of 85%, specificity of 29%, precision of 84% and balanced accuracy of 57%.

3.2 AUROC

Table 18: Mean AUROC - All Top Models

Model	Mean AUROC
XGBoost_final	0.9323095
XGBoost_final_NoCor	0.9051468
RandomForest_final	0.9312520
SVM_final	0.7602619
NNet_final	0.8771071
NNet_final_NoCor	0.8715357

With respect to the mean AUROC value, the best result is achieved by using the XGBoost model with all available variables. The neural network model, which was recognized as the optimal model with respect to the basic metrics, achieved considerably lower AUROC. The XGBoost model with all available variables also has the lowest variability of the mean AUROC value (as calculated by the *caret* R package). This is demonstrated in the Figure 17.

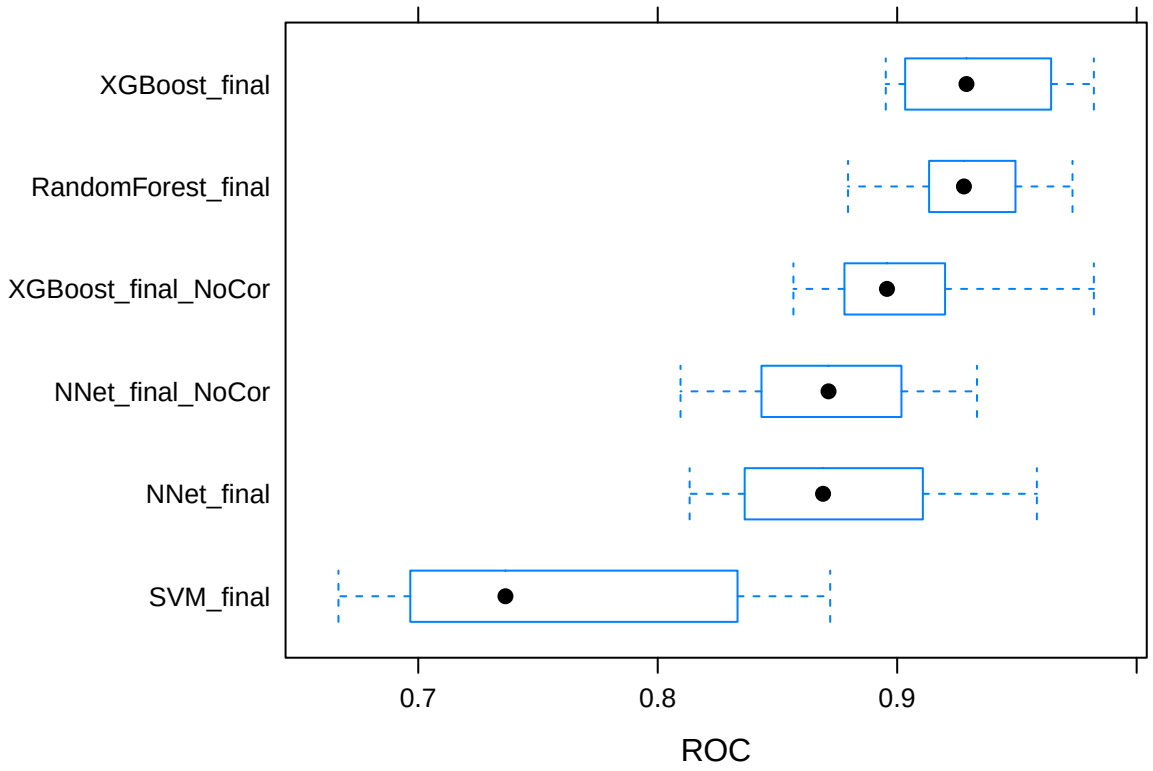

Figure 17: AUROC Variability - Top Models

3.3 AUPRC

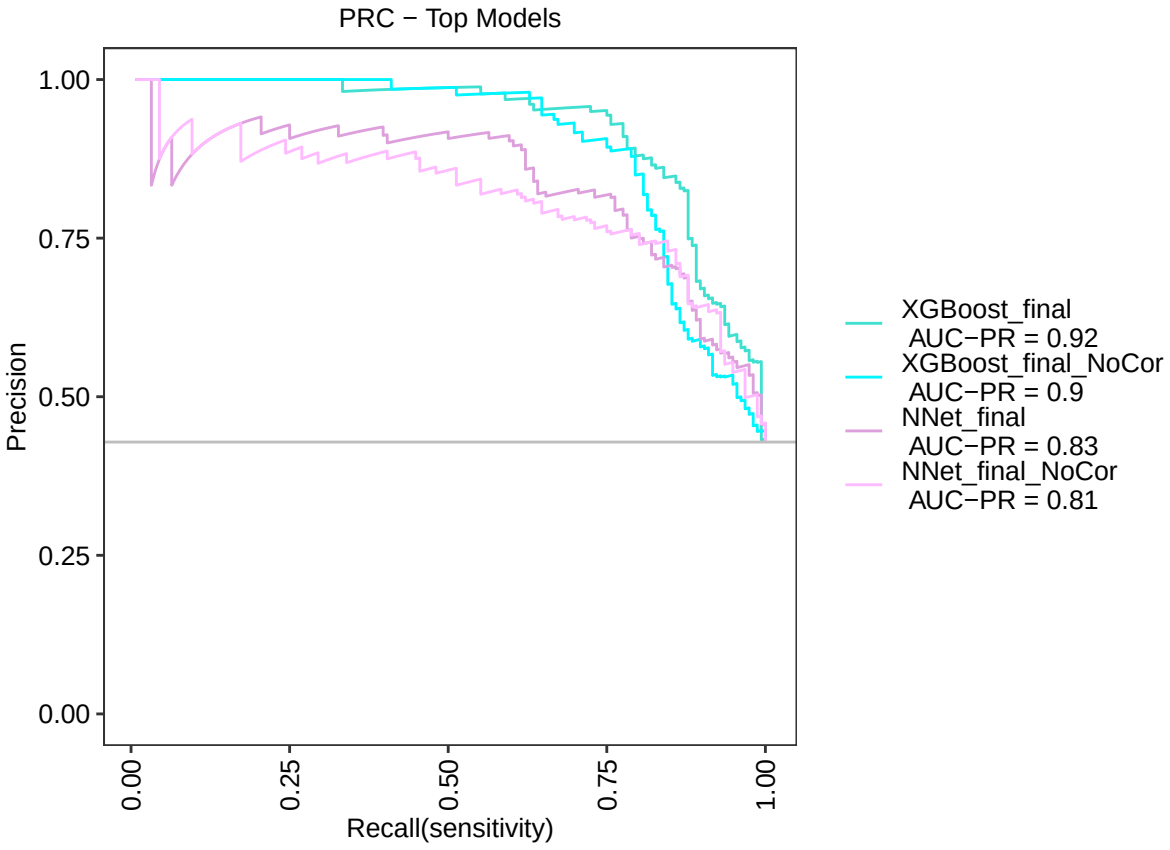

Figure 18: Precision-Recall Curve - Top Models

As in the case of mean AUROC, with respect to the mean AUPRC value, the best result is achieved by using the XGBoost model with all available variables.

4 Discussion And Conclusion

The main challenge of this research project was the highly imbalanced data. All evaluated machine learning algorhitms struggled with predicting the low-prevalent negative outcome of death within 1 year after the lung cancer resection.

A strong positive impact on models' performance was observed after data balancing with the SMOTE method. This impact was consistent for all models, and in all tested settings.

Another challenge lies in the number of features available for in-depth analysis of the data. It was shown in previous studies that there was much difficulty in drawing correlations between the various features. This issue was addressed by feature engineering, that had a much less prominent impact on models' performance than data balancing. In the case of the XGBoost model, feature engineering helped to achieve a slight increase in AUROC and AUPRC, but had no influence on the other metrics.

Similarly, the removal of recognized strongly correlated variables had a minor and varying impact. The neural network model performed worse after the removal, while the XGBoost model performed better, but this improvement was limited to basic metrics. A possible reason for such effect is a low number of available variables. Namely, after the removal of strongly correlated variables, only two numeric variables remained, i.e. Age and FEV1/FVC ratio. And since all binary variables were highly imbalanced, the models struggled due to zero variability of some features.

The original dataset included one categorical nominal variable, Dg, which was treated with One-Hot Encoding. With such approach, I intended to avoid that models would assume a natural ordering between categories, which may result in poor performance. However, the explorative analysis has demonstrated that One-Hot Encoding has not caused a convincing improvement of the performance. On the contrary, the highest specificity and balanced accuracy were reached on the final dataset without the One-Hot Encoding. This implies that caution is needed when deciding for One-Hot Encoding, as it may not always be optimal the optimal approach.

Previous research projects, involving the same dataset and evaluating similar models, reported prediction accuracy of machine learning algorithms as high as 87% [5, 6], and even over 97% with the use of data balancing and feature engineering [7]. The results are hardly comparable, as the exact methods of calculating the reported accuracies and other metrics are not always clear. However, in this research project, evaluated models were unable to reach such high values of performance metrics. Overall, in terms of specificity (47%), precision (87%) and balanced accuracy (64%), the best results were obtained with the neural network model, without the removal of strongly correlated variables. In terms of AUROC (0.93) and AUPRC (0.92), the best perfomance was obtained with the XGBoost model, without the removal of strongly correlated variables.

All evaluated models yielded relatively high rate of false positive and false negative predictions. Therefore, the models in such form have limited applicability in a clinical setting. However, the performance could be much improved in the future with further model enhancements (e.g. applying advanced attribute ranking and selection methods, other data balancing techniques, fine tuning of model parameters), the collection of higher number of observations to balance the dataset, and with the collection of additional features relating to patient general state, pulmonary function and other comorbidities beyond the few considered here. An enhanced model would provide a useful tool to healthcare providers in informing the selection of suitable patients and in predicting patient outcomes after lung cancer resection.

5 References

1. de Koning HJ, van der Aalst CM, de Jong PA, Scholten ET, Nackaerts K, Heuvelmans MA, et al. Reduced Lung-Cancer Mortality with Volume CT Screening in a Randomized Trial. N Engl J Med. 2020;382(6):503-13.

2. Calabro E, Randi G, La Vecchia C, Sverzellati N, Marchiano A, Villani M, et al. Lung function predicts lung cancer risk in smokers: a tool for targeting screening programmes. Eur Respir J. 2010;35(1):146-51.

3. Lu T, Yang X, Huang Y, et al. Trends in the incidence, treatment, and survival of patients with lung cancer in the last four decades. Cancer Manag Res. 2019;11:943-953. Published 2019 Jan 21. doi:10.2147/CMAR.S187317

4. British Thoracic Society; Society of Cardiothoracic Surgeons of Great Britain and Ireland Working Party. BTS guidelines: guidelines on the selection of patients with lung cancer for surgery. Thorax. 2001;56(2):89-108. doi:10.1136/thorax.56.2.89

5. Desuky, Abeer S., and Lamiaa M. El Bakrawy. Improved prediction of post-operative life expectancy after Thoracic Surgery. Advances in Systems Science and Applications 16.2 (2016): 70-80.

6. Abdulhamid, A. (2014). Life Expectancy Post Thoracic Surgery. Stanford University: CS 229.

7. Xu, S. (2019). Machine Learning-Assisted Prediction of Surgical Mortality of Lung Cancer Patients. In ICDM (Posters) (pp. 46-51).

8. Irizarry RA. Introduction to Data Science: Data Analysis and Prediction Algorithms with R: CRC Press; 2019.

9. Zięba, M., Tomczak, J. M., Lubicz, M., & Świątek, J. (2014). Boosted SVM for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients. Applied soft computing, 14, 99-108.

6 Appendices

6.1 Code Generated in This Research Project

```
##########################
#DATA COLLECTION AND PROCESSING
##########################

#Install the required R packages, if applicable
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")
if(!require(farff)) install.packages("farff", repos = "http://cran.us.r-project.org")
if(!require(mltools)) install.packages("mltools", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(funModeling)) install.packages("funModeling", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(DMwR)) install.packages("DMwR", repos = "http://cran.us.r-project.org")
if(!require(broom)) install.packages("broom", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(MLeval)) install.packages("MLeval", repos = "http://cran.us.r-project.org")
if(!require(ggpubr)) install.packages("ggpubr", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(kableExtra)
library(farff) #needed to convert the original .arff file into a dataframe
library(mltools) #used for one-hot encoding
library(data.table)
library(funModeling) #needed for some functions, e.g. plot_num(); NOTE: masks dplyr::summarize!
library(corrplot) #for creating correlation plot
library(DMwR) #for data balancing (SMOTE method)
library(broom)
library(randomForest)
library(MLeval) #for ROC and other cuves and metrics
library(ggpubr) #for combining plots into one

#Download the dataset from the internet and save into temporary file:
dl <- tempfile()
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00277/ThoracicSurgery.arff",
              dl)
#Read the downloaded .arff file
lc <- readARFF(dl) %>% data.frame()

#Remove the temporary file
file.remove(dl)

#Observe the dataset
str(lc)

#Determine if the outcome and other binary variables are balanced
z <- sapply(lc, summary)
as.data.frame(z[c("Risk1Yr", "PRE7", "PRE8", "PRE9", "PRE10", "PRE11",
                  "PRE17", "PRE19", "PRE25", "PRE30", "PRE32")]) %>%
```

```
  kable() %>%
  kable_styling(latex_options = "scale_down",
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Rename the columns with more intuitive names
colnames(lc) <- c("Dg", "FVC", "FEV1", "Zubrod", "Pain", "Hemoptysis",
                "Dyspnea", "Cough", "Weakness", "Tumor_size", "T2DM", "MI", "PAD",
                "Smoking", "Asthma", "Age", "Death")

#Save the original dataset for further exploration
lc_orig <- lc

#Convert Boolean variables into integers; F into 0, T into 1
lc <- lc %>% mutate(Pain = as.factor(str_replace_all(Pain, "F", "0"))) %>%
  mutate(Pain = as.factor(str_replace_all(Pain, "T", "1"))) %>%
  mutate(Hemoptysis = as.factor(str_replace_all(Hemoptysis, "F", "0"))) %>%
  mutate(Hemoptysis = as.factor(str_replace_all(Hemoptysis, "T", "1"))) %>%
  mutate(Dyspnea = as.factor(str_replace_all(Dyspnea, "F", "0"))) %>%
  mutate(Dyspnea = as.factor(str_replace_all(Dyspnea, "T", "1"))) %>%
  mutate(Cough = as.factor(str_replace_all(Cough, "F", "0"))) %>%
  mutate(Cough = as.factor(str_replace_all(Cough, "T", "1"))) %>%
  mutate(Weakness = as.factor(str_replace_all(Weakness, "F", "0"))) %>%
  mutate(Weakness = as.factor(str_replace_all(Weakness, "T", "1"))) %>%
  mutate(T2DM = as.factor(str_replace_all(T2DM, "F", "0"))) %>%
  mutate(T2DM = as.factor(str_replace_all(T2DM, "T", "1"))) %>%
  mutate(MI = as.factor(str_replace_all(MI, "F", "0"))) %>%
  mutate(MI = as.factor(str_replace_all(MI, "T", "1"))) %>%
  mutate(PAD = as.factor(str_replace_all(PAD, "F", "0"))) %>%
  mutate(PAD = as.factor(str_replace_all(PAD, "T", "1"))) %>%
  mutate(Smoking = as.factor(str_replace_all(Smoking, "F", "0"))) %>%
  mutate(Smoking = as.factor(str_replace_all(Smoking, "T", "1"))) %>%
  mutate(Asthma = as.factor(str_replace_all(Asthma, "F", "0"))) %>%
  mutate(Asthma = as.factor(str_replace_all(Asthma, "T", "1"))) %>%
  mutate(Death = as.factor(str_replace_all(Death, "F", "0"))) %>%
  mutate(Death = as.factor(str_replace_all(Death, "T", "1")))
lc <- lc %>% mutate(Pain = as.numeric(levels(Pain))[Pain],
                    Hemoptysis = as.numeric(levels(Hemoptysis))[Hemoptysis],
                    Dyspnea = as.numeric(levels(Dyspnea))[Dyspnea],
                    Cough = as.numeric(levels(Cough))[Cough],
                    Weakness = as.numeric(levels(Weakness))[Weakness],
                    T2DM = as.numeric(levels(T2DM))[T2DM],
                    MI = as.numeric(levels(MI))[MI],
                    PAD = as.numeric(levels(PAD))[PAD],
                    Smoking = as.numeric(levels(Smoking))[Smoking],
                    Asthma = as.numeric(levels(Asthma))[Asthma],
                   Death = as.numeric(levels(Death))[Death])
lc <- lc %>% mutate(Pain = as.integer(Pain),
                      Hemoptysis = as.integer(Hemoptysis),
                      Dyspnea = as.integer(Dyspnea),
                      Cough = as.integer(Cough),
                      Weakness = as.integer(Weakness),
```

```
                        T2DM = as.integer(T2DM),
                        MI = as.integer(MI),
                        PAD = as.integer(PAD),
                        Smoking = as.integer(Smoking),
                        Asthma = as.integer(Asthma),
                    Death = as.integer(Death))

#OPTIONAL: Double-check if the sums of True values remained unchanged
#lc %>% select(-FVC, -FEV1, -Age) %>% summarize_if(is.numeric, sum, na.rm=TRUE)

#Convert categorical ordinal variables into integers, by removing the repeating strings
lc <- lc %>% mutate(Zubrod = str_replace(Zubrod, "PRZ", "")) %>%
  mutate(Tumor_size = str_replace(Tumor_size, "OC1", ""))
lc <- lc %>% mutate(Zubrod = as.integer(Zubrod),
              Tumor_size = as.integer(Tumor_size))

#Convert nominal variable with multiple categories (factors)
#into multiple variables (integers with values 0 or 1) by one-hot encoding
data <- data.table(
  ID = seq(1,nrow(lc),by=1),
  Variable = lc$Dg)
newdata <- one_hot(data)
colnames(newdata) <- c("ID", "DGN3", "DGN2", "DGN4", "DGN6",
                       "DGN5", "DGN8", "DGN1")

#Join the dataframe with one-hot encoding variables and lc
lc <- lc %>% mutate(ID = row_number()) %>%
  left_join(newdata, by="ID") %>%
  mutate(Dg3 = DGN3, Dg2 = DGN2, Dg4 = DGN4, Dg6 = DGN6,
         Dg5 = DGN5, Dg8 = DGN8, Dg1 = DGN1) %>%
  select(-Dg, -ID, -DGN3, -DGN2, -DGN4, -DGN6, -DGN5, -DGN8, -DGN1)

#Show the first 6 rows of the pre-processed dataset
head(lc) %>%
  kable(caption = "lc dataset after pre-processing (first 6 rows)") %>%
  kable_styling(latex_options = "scale_down",
                bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Check the number of missing values
sum(is.na(lc))

#Plot the histograms of FVC, FEV1 and Age to check for outliers
lc_numerics <- lc %>% select(FVC, FEV1, Age)
plot_outliers <- plot_num(lc_numerics, bins=15)
#Format the plot (Code separated due to markdown issues)
plot_outliers +
  labs(title = "Histograms of metric variables",
       x = "Value",
       y = "Count")+
  theme_bw()
```

```r
#Count the numbers of outliers for FEV1 with the Tukey's test
sum(is.na(prep_outliers(
  lc$FEV1,
  type = "set_na",
  method = "tukey")))

#Confirm there are no outliers for FVC with the Tukey's test
sum(is.na(prep_outliers(
  lc$FVC,
  type = "set_na",
  method = "tukey")))

#Count the numbers of outliers for Age with the Tukey's test
sum(is.na(prep_outliers(lc$Age,
                        type = "set_na",
                        method = "tukey")))

#Extract the rows with the outliers for FEV1
FEV1_outliers <- which(is.na(prep_outliers(lc$FEV1,
                          type = "set_na",
                          method = "tukey")))
lc[FEV1_outliers,] %>%
  select(FVC, FEV1, Age, Death) %>%
  kable(caption = "FEV1 Outliers") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Extract the row with the outlier for Age
Age_outlier <- which(is.na(prep_outliers(lc$Age,
                        type = "set_na",
                        method = "tukey")))
lc[Age_outlier,] %>%
  select(FVC, FEV1, Age, Death) %>%
  kable(caption = "Age Outlier") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Remove outliers from the lc dataset
lc <- lc[-c(FEV1_outliers, Age_outlier),]
#Check the structure of the cleaned lc dataset
str(lc)

#Join the cleaned lc dataset with "Dg" from the original lc dataset,
#then remove all added columns/variables
lc_orig_cleaned <- lc_orig %>% mutate(Dg = as.factor(str_replace_all(Dg, "DGN1", "1"))) %>%
  mutate(Dg = as.factor(str_replace_all(Dg, "DGN2", "2"))) %>%
  mutate(Dg = as.factor(str_replace_all(Dg, "DGN3", "3"))) %>%
  mutate(Dg = as.factor(str_replace_all(Dg, "DGN4", "4"))) %>%
  mutate(Dg = as.factor(str_replace_all(Dg, "DGN5", "5"))) %>%
  mutate(Dg = as.factor(str_replace_all(Dg, "DGN6", "6"))) %>%
```

```
  mutate(Dg = as.factor(str_replace_all(Dg, "DGN8", "8"))) %>%
  mutate(Dg = as.numeric(levels(Dg))[Dg]) %>%
  mutate(Dg = as.integer(Dg)) %>%
  mutate(ID = row_number())
lc_orig_to_join <- lc_orig_cleaned %>%
  filter(!((row_number() %in% FEV1_outliers) | (row_number() %in% Age_outlier))) %>%
  mutate(ID = row_number()) %>%
  select(ID, Dg)
lc_wID <- lc %>%
  mutate(ID = row_number())
lc_orig_cleaned_final <- left_join(lc_wID, lc_orig_to_join, by="ID") %>%
  select(FVC, FEV1, Age, Dg, Zubrod, Pain, Hemoptysis,
         Dyspnea, Cough, Weakness, Tumor_size,
         T2DM, MI, PAD, Smoking, Asthma, Death)

#Plot the distribution of the metric variables after removing the outliers
plot_lc_metric_cleaned <-
  plot_num(lc_orig_cleaned_final[,c("FVC","FEV1","Age")], bins = 15)

#Format the plot (Code separated due to markdown issues)
plot_lc_metric_cleaned +
  labs(title = "Distribution of metric variables without outliers",
       x = "Value",
       y = "Count")+
  theme_bw()

#Plot the distribution of the original categorical variables after removing the outliers
plot_lc_categorical_cleaned <-
  plot_num(lc_orig_cleaned_final[,c("Dg","Zubrod","Pain",
                                    "Hemoptysis", "Dyspnea", "Cough", "Weakness",
                                    "Tumor_size", "T2DM", "MI", "PAD", "Smoking",
                                    "Asthma", "Death")], bins = 15)

#Format the plot (Code separated due to markdown issues)
plot_lc_categorical_cleaned +
  labs(title = "Distribution of original categorical variables without outliers",
       x = "Factor Levels (from the lowest to highest)",
       y = "Count")+
  theme_bw()+
  theme(axis.text.x=element_blank())

#Create correlation matrix
correlationMatrix <- cor(lc)

#Create correlation plot
res1 <- cor.mtest(lc, conf.level = .95) #used for p.mat argument to determine stat. significance
corrplot(correlationMatrix, addrect = 6, tl.col = "black",
         p.mat = res1$p, insig = "blank") #squares with insignif. coefficients are left blank

#Calculate strong correlations
cm <- as.data.frame(as.table(correlationMatrix))
cm %>% filter(abs(Freq) > 0.5 & abs(Freq) < 1)

#Print the correlation coefficients with absolute values between 0.5 and 1.0
```

```r
round(cor(lc$FVC, lc$FEV1), digits = 2)
round(cor(lc$Cough, lc$Zubrod), digits = 2)
round(cor(lc$Dg2, lc$Dg3), digits = 2)
round(cor(lc$Dg4, lc$Dg3), digits = 2)

#Determine which strongly correlated variables to be removed
strong_cor <- findCorrelation(correlationMatrix, cutoff=0.5)
colnames(lc[,strong_cor])

#Plot the FVC vs FEV1 and mapped Death
lc %>% group_by(Death) %>%
  ggplot(aes(FEV1, FVC, color=factor(Death), fill=factor(Death)))+
  geom_point(size=3, shape=21, alpha=0.5, position=position_jitter(h=0.1, w=0.1))+
  scale_fill_manual(values=c("turquoise", "plum2"))+ #change default fill color scheme
  scale_color_manual(values=c("turquoise", "plum2"))+
  theme_bw()+
    labs(fill = "Death", color = "Death",
         title = "FVC vs FEV1 with mapped Death",
         x = "FEV1 (Litres)",
         y = "FVC (Litres)")+
  annotate(geom="text", x=4.75, y=1.8,
           label=paste("Paerson's r =",round(cor(lc$FEV1, lc$FVC),2)), color="black")

#Plot the FVC vs Age and mapped Death
lc %>% group_by(Death) %>%
  ggplot(aes(Age, FVC, color=factor(Death), fill=factor(Death)))+
  geom_point(size=3, shape=21, alpha=0.6, position=position_jitter(h=0.1, w=0.1))+
  scale_fill_manual(values=c("turquoise", "plum2"))+ #change default fill color scheme
  scale_color_manual(values=c("turquoise", "plum2"))+
  theme_bw()+
  labs(fill = "Death", color = "Death",
       title = "FVC vs Age with mapped Death",
       x = "Age (Years)",
       y = "FVC (Litres)")+
  annotate(geom="text", x=78, y=6.2,
           label=paste("Paerson's r =",round(cor(lc$Age, lc$FVC),2)), color="black")

#Plot the Zubrod vs Tumor Size and mapped Death
lc %>% ggplot(aes(factor(Tumor_size), factor(Zubrod),
                  color=factor(Death), fill=factor(Death)))+
  geom_jitter(size=2, shape=21, alpha=0.6)+
  scale_fill_manual(values=c("turquoise", "plum2"))+ #change default fill color scheme
  scale_color_manual(values=c("turquoise", "plum2"))+
  theme_bw()+
  labs(fill = "Death", color = "Death",
       title = "Zubrod score vs Tumor size with mapped Death",
       x="Tumor Size",
       y="Zubrod Score")

#Calculate Paerson's r for Tumor_size and Zubrod
paste("Paerson's r =", round(cor(lc$Tumor_size, lc$Zubrod),2))


#Plot the FEV1/FVC ratio vs Age and mapped Death
```

```
lc_wCOPD <- lc %>% mutate(Tiff = FEV1/FVC, COPD=as.numeric(FEV1/FVC <=0.7))
lc_wCOPD %>%
  group_by(Death) %>%
  ggplot(aes(Age, Tiff, color=factor(Death), fill=factor(Death)))+
  geom_point(size=3, shape=21, alpha=0.6, position=position_jitter(h=0.1, w=0.1))+
  scale_fill_manual(values=c("turquoise", "plum2"))+ #change default fill color scheme
  scale_color_manual(values=c("turquoise", "plum2"))+
  theme_bw()+
  labs(fill = "Death", color = "Death",
       title = "FEV1/FVC Ratio vs Age with mapped Death",
       x="Age (Years)",
       y="FEV1/FVC Ratio")+
  geom_hline(yintercept=0.7, col = "black", lty=2)+
  annotate(geom="text", x=80, y=1.2,
           label=paste("Paerson's r =",round(cor(lc_wCOPD$Tiff, lc_wCOPD$Age),2)),
           color="black")

#Calculate Paerson's r for FEV1/FVC Ratio and Death
paste("Paerson's r =", round(cor(lc_wCOPD$Tiff, lc$Death),2))


#Plot the COPD vs Smoking and mapped Death
lc_wCOPD %>%
  ggplot(aes(factor(Smoking), factor(COPD),
             color=factor(Death), fill=factor(Death)))+
  geom_jitter(size=2, shape=21, alpha=0.6)+
  scale_fill_manual(values=c("turquoise", "plum"))+ #change default fill color scheme
  scale_color_manual(values=c("turquoise", "plum"))+
  theme_bw()+
  labs(fill = "Death", color = "Death",
       title = "COPD vs Smoking with mapped Death",
       x="Smoking",
       y="COPD")

#Calculate Paerson's r for COPD and Smoking
paste("Paerson's r =", round(cor(lc_wCOPD$COPD, lc$Smoking),2))

############################
#DATA ANALYSIS
############################

# Split the dataset to train and test set; test set will have 20% of the lc data
set.seed(16)
test_index <- createDataPartition(y = lc$Death, times = 1, p = 0.2, list = FALSE)
lc_train <- lc[-test_index,]
lc_test <- lc[test_index,]

#Observe the structure of both subsets
str(lc_train)
str(lc_test)

#prepare the training scheme with 10-fold cross validation on the whole train dataset
control <- trainControl(method="cv",
                        number=10,
```

```
                        classProbs = TRUE,
                        summaryFunction = twoClassSummary,
                        savePredictions = TRUE)

#Extract outcomes y and features x from the lc_train and lc_test
x_train <- lc_train %>% dplyr::select(-Death)
y_train <- factor(lc_train$Death, labels = c("Alive", "Dead"))
x_test <- lc_test %>% dplyr::select(-Death)
y_test <- factor(lc_test$Death, labels = c("Alive", "Dead"))

#Build a Naive Bayes model
if(!require(klaR)) install.packages("klaR", repos = "http://cran.us.r-project.org")
library(klaR) #for Naive Bayes model; open here to avoid masking some dplyr functions (select)
set.seed(16)
model_nbayes <- train(x_train, y_train,
                      method = "nb",
                      metric="ROC",
                      tuneGrid = data.frame(fL = 0, usekernel = TRUE, adjust = 1),
                      trControl = control)
detach("package:klaR", unload = TRUE)
#use the model to predict the observations from the lc_test features
prediction_nbayes <- predict(model_nbayes, x_test)

#Create the model's confusion matrix
cm_nbayes <- confusionMatrix(prediction_nbayes, y_test)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_nbayes <- tidy(cm_nbayes) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "Naive Bayes"=estimate)


#Build a Random Forest model
set.seed(16)
model_rf <- train(x_train, y_train,
                  method = "rf",
                  metric="ROC",
                  trControl = control)
#use the model to predict the observations from the lc_test features
prediction_rf <- predict(model_rf, x_test)

#Create the model's confusion matrix
cm_rf <- confusionMatrix(prediction_rf, y_test)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_rf <- tidy(cm_rf) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "Random Forest"=estimate)

#Build a XGBoost model
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
library(xgboost) #open here to avoid masking some dplyr functions (slice)
set.seed(16)
model_xgb <- train(x_train, y_train,
                  method = "xgbTree",
                  metric = "ROC",
                  trControl = control)
detach("package:xgboost", unload = TRUE)
```

```r
#use the model to predict the observations from the lc_test features
prediction_xgb <- predict(model_xgb, x_test)

#Create the model's confusion matrix
cm_xgb <- confusionMatrix(prediction_xgb, y_test)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_xgb <- tidy(cm_xgb) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, XGBoost=estimate)

#Basic metrics of the models built on the original dataset
metrics_orig <- left_join(tidy_cm_nbayes, tidy_cm_rf, by="Metric") %>%
  left_join(.,tidy_cm_xgb, by="Metric")
metrics_orig %>% mutate(Metric = str_to_title(Metric)) %>%
  kable(caption = "Basic Metric Scores per Model - Cleaned Original Dataset") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Calculate and compare the model AUROCs
models_list_orig <- list(NaiveBayes = model_nbayes, RandomForest = model_rf, XGBoost = model_xgb)
models_results_orig <- resamples(models_list_orig)
AUROCs_orig <- summary(models_results_orig)
AUROCs_orig_df <- data.frame(AUROCs_orig$statistics$ROC[,4])
AUROCs_orig_df <- cbind(rownames(AUROCs_orig_df), AUROCs_orig_df)
rownames(AUROCs_orig_df) <- NULL
colnames(AUROCs_orig_df) <- c("Model", "Mean AUROC")
AUROCs_orig_df %>%
  kable(caption = "Mean AUROC per Model - Cleaned Original Dataset") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)



#Plot the ROCs with AUROCs
auroc_orig <- evalm(list(model_nbayes,model_rf,model_xgb),
             gnames=c('NaiveBayes','RandomForest','XGBoost'),
             cols=c("turquoise", "plum", "plum1"),
             fsize=10,
             plots=c("r"),
             rlinethick = 0.5,
             dlinethick = 0.5,
             title="Original dataset")
#Plot the PRCs with AUPRCs
auprc_orig <- evalm(list(model_nbayes,model_rf,model_xgb),
                    gnames=c('NaiveBayes','RandomForest','XGBoost'),
                    cols=c("turquoise", "plum", "plum1"),
                    fsize=10,
                    plots=c("pr"),
                    rlinethick = 0.5,
                    dlinethick = 0.5,
                    title="Original dataset")
```

```r
#Combine ROCs and PRCs into one plot
ggarrange(auroc_orig$roc, auprc_orig$proc,
          ncol = 1, nrow = 2)


###DATA BALANCING
#Apply the SMOTE method to the lc_train dataset
lc_train_fact <- lc_train %>% mutate(Death = factor(Death, labels = c("Alive", "Dead")))
set.seed(16)
lc_train_smote <- SMOTE(Death ~ ., data = lc_train_fact, k=5)
#Check the balance of the target variable
table(lc_train_smote$Death)

#Plot the variable distributions after the SMOTE
lc_train_smote_num <- lc_train_smote %>%
  mutate(Death = as.factor(str_replace_all(Death, "Alive", "0"))) %>%
  mutate(Death = as.factor(str_replace_all(Death, "Dead", "1"))) %>%
  mutate(Death = as.numeric(levels(Death))[Death])
plot_lc_train_smote_num <- plot_num(lc_train_smote_num)
plot_lc_train_smote_num +
  labs(title = "Distribution of variables after SMOTE data balancing",
       x = "Value",
       y = "Count")+
  theme_bw()

#Build a Random Forest model on the balanced SMOTE dataset
set.seed(16)
model_rf_smote <- train(Death ~ ., data = lc_train_smote,
                        method = "rf",
                        metric="ROC",
                        trControl = control)
#use the model to predict the observations from the lc_test features
prediction_rf_smote <- predict(model_rf_smote, x_test)
#Create the model's confusion matrix
cm_rf_smote <- confusionMatrix(prediction_rf_smote, y_test)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_rf_smote <- tidy(cm_rf_smote) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "Random Forest - Balanced"=estimate)


#Build a XGBoost model on the SMOTE balanced dataset
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
library(xgboost) #open here to avoid masking some dplyr functions (slice)
set.seed(16)
model_xgb_smote <- train(Death ~ ., data = lc_train_smote,
                         method = "xgbTree",
                         metric = "ROC",
                         trControl = control)
detach("package:xgboost", unload = TRUE)
#use the model to predict the observations from the lc_test features
prediction_xgb_smote <- predict(model_xgb_smote, x_test)
#Create the model's confusion matrix
cm_xgb_smote <- confusionMatrix(prediction_xgb_smote, y_test)
```

```r
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_xgb_smote <- tidy(cm_xgb_smote) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "XGBoost - Balanced"=estimate)

#Basic metrics of the models after data balancing with SMOTE
metrics_smote <- left_join(tidy_cm_rf_smote, tidy_cm_xgb_smote, by="Metric")
metrics_smote  %>%
  mutate(Metric = str_to_title(Metric)) %>%
  kable(caption = "Basic Metric Scores per Model - After SMOTE Data Balancing") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Calculate and compare the model AUROCs - after SMOTE
models_list_smote <- list(RandomForest_balanced = model_rf_smote, XGBoost_balanced = model_xgb_smote)
models_results_smote <- resamples(models_list_smote)
AUROCs_smote <- summary(models_results_smote)
AUROCs_smote_df <- data.frame(AUROCs_smote$statistics$ROC[,4])
AUROCs_smote_df <- cbind(rownames(AUROCs_smote_df), AUROCs_smote_df)
rownames(AUROCs_smote_df) <- NULL
colnames(AUROCs_smote_df) <- c("Model", "Mean AUROC")
AUROCs_smote_df %>%
  kable(caption = "Mean AUROC per Model - Smote Balanced Dataset") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Plot the ROCs with AUROCs - after SMOTE
auroc_smote <- evalm(list(model_rf_smote,model_xgb_smote),
                     gnames=c('RandomForest','XGBoost'),
                     cols=c("turquoise", "plum"),
                     fsize=10,
                     plots=c("r"),
                     rlinethick = 0.5,
                     dlinethick = 0.5,
                     title="SMOTE Balanced dataset")
#Plot the PRCs with AUPRCs  - after SMOTE
auprc_smote <- evalm(list(model_rf_smote,model_xgb_smote),
                     gnames=c('RandomForest','XGBoost'),
                     cols=c("turquoise", "plum"),
                     fsize=10,
                     plots=c("pr"),
                     rlinethick = 0.5,
                     dlinethick = 0.5,
                     title="SMOTE Balanced dataset")
#Combine ROCs and PRCs into one plot - after SMOTE
ggarrange(auroc_smote$roc, auprc_smote$proc,
          ncol = 1, nrow = 2)

#FEATURE IMPORTANCE
#Plot the RF model's 10 most important variables
plot_fi_rf <- plot(varImp(model_rf), top=10, main="The most important variables - Random Forest")
```

```r
#Plot the XGB model's 10 most important variables
plot_fi_xgb <- plot(varImp(model_xgb), top=10, main="The most important variables - XGBoost")
ggarrange(plot_fi_rf, plot_fi_xgb,
          ncol = 1, nrow = 2)


#Add new computated features to lc_train and lc_test
lc_train_eng <- lc_train %>% mutate(Death = factor(Death, labels = c("Alive", "Dead")),
                                    "FEV1/FVC" = FEV1/FVC,
                                    "FVC*FEV1" = FVC*FEV1,
                                    "FVC*FEV1^2" = FVC*(FEV1)^2,
                                    "FVC^2*FEV1" = (FVC^2)*FEV1,
                                    "FVC^2" = FVC^2,
                                    "Age*FVC" = Age*FVC,
                                    "Age*FEV1" = Age*FEV1,
                                    "Age/Tumor_size" = Age/Tumor_size,
                                    "FVC*Tumor_size" = FVC*Tumor_size)
lc_test_eng <- lc_test %>% mutate(Death = factor(Death, labels = c("Alive", "Dead")),
                                  "FEV1/FVC" = FEV1/FVC,
                                  "FVC*FEV1" = FVC*FEV1,
                                  "FVC*FEV1^2" = FVC*(FEV1)^2,
                                  "FVC^2*FEV1" = (FVC^2)*FEV1,
                                  "FVC^2" = FVC^2,
                                  "Age*FVC" = Age*FVC,
                                  "Age*FEV1" = Age*FEV1,
                                  "Age/Tumor_size" = Age/Tumor_size,
                                  "FVC*Tumor_size" = FVC*Tumor_size)
x_lc_test_eng <- lc_test_eng %>% dplyr::select(-Death)

#Evaluate the importance with the new features on XGB model
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
library(xgboost) #open here to avoid masking some dplyr functions (slice)
set.seed(16)
model_xgb_eng <- train(Death ~ ., data = lc_train_eng,
                       method = "xgbTree",
                       metric = "ROC",
                       trControl = control)
detach("package:xgboost", unload = TRUE)
#use the model to predict the observations from the lc_test features
prediction_xgb_eng <- predict(model_xgb_eng, x_lc_test_eng)
#Create the model's confusion matrix
cm_xgb_eng <- confusionMatrix(prediction_xgb_eng, lc_test_eng$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_xgb_eng <- tidy(cm_xgb_eng) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "XGBoost - New Features"=estimate)
#Plot the XGB model's 10 most important variables After Engineering
plot_fi_xgb_eng <- plot(varImp(model_xgb_eng), top=10,
                        main="The most important variables, \n XGBoost (After Engineering)")
plot_fi_xgb_eng

#FIND STRONG CORRELATIONS after the feature engineering
lc_train_eng_numeric <- lc_train %>% mutate("FEV1/FVC" = FEV1/FVC,
                                            "FVC*FEV1" = FVC*FEV1,
                                            "FVC*FEV1^2" = FVC*(FEV1)^2,
```

```
                                          "FVC^2*FEV1" = (FVC^2)*FEV1,
                                          "FVC^2" = FVC^2,
                                          "Age*FVC" = Age*FVC,
                                          "Age*FEV1" = Age*FEV1,
                                          "Age/Tumor_size" = Age/Tumor_size,
                                          "FVC*Tumor_size" = FVC*Tumor_size)
#Create correlation matrix
correlationMatrix_eng <- cor(lc_train_eng_numeric)
#Determine which strongly correlated variables to be removed
strong_cor_eng <- findCorrelation(correlationMatrix_eng, names=TRUE, cutoff=0.5)
strong_cor_eng

#############
#CONSTRUCT THE FINAL DATASET (Balanced train subset + Added New Features)
lc_train_final <- lc_train_smote %>% mutate("FEV1/FVC" = FEV1/FVC,
                            "FVC*FEV1" = FVC*FEV1,
                            "FVC*FEV1^2" = FVC*(FEV1)^2,
                            "FVC^2*FEV1" = (FVC^2)*FEV1,
                            "FVC^2" = FVC^2,
                            "Age*FVC" = Age*FVC,
                            "Age*FEV1" = Age*FEV1,
                            "Age/Tumor_size" = Age/Tumor_size,
                            "FVC*Tumor_size" = FVC*Tumor_size)
lc_test_final <- lc_test_eng
x_lc_test_final <- x_lc_test_eng

#Build a XGBoost model on the FINAL dataset (balanced + added features)
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
library(xgboost) #open here to avoid masking some dplyr functions (slice)
set.seed(16)
model_xgb_final <- train(Death ~ ., data = lc_train_final,
                         method = "xgbTree",
                         metric = "ROC",
                         trControl = control)
detach("package:xgboost", unload = TRUE)
#use the model to predict the observations from the lc_test features
prediction_xgb_final <- predict(model_xgb_final, x_lc_test_final)
#Create the model's confusion matrix
cm_xgb_final <- confusionMatrix(prediction_xgb_final, lc_test_final$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_xgb_final <- tidy(cm_xgb_final) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "XGBoost - Final"=estimate)

#Costruct the Final Dataset With Removed highly correlated variables
lc_train_final_num <- lc_train_final %>%
  mutate(Death = as.factor(str_replace_all(Death, "Alive", "0"))) %>%
  mutate(Death = as.factor(str_replace_all(Death, "Dead", "1"))) %>%
  mutate(Death = as.numeric(levels(Death))[Death]) %>%
  dplyr::select(-Dg1) #removed because it has zero SD, and prevents calculation of correlations
lc_train_final_nocor <- lc_train_final %>% dplyr::select(-strong_cor_eng)
lc_test_final_nocor <- lc_test_final %>% dplyr::select(-strong_cor_eng)
x_lc_test_final_nocor <- lc_test_final_nocor %>% dplyr::select(-Death)

#Build a XGBoost model on the FINAL dataset without highly correlated variables
```

```
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
library(xgboost) #open here to avoid masking some dplyr functions (slice)
set.seed(16)
model_xgb_final_nocor <- train(Death ~ ., data = lc_train_final_nocor,
                          method = "xgbTree",
                          metric = "ROC",
                          trControl = control)
detach("package:xgboost", unload = TRUE)
#use the model to predict the observations from the lc_test features
prediction_xgb_final_nocor <- predict(model_xgb_final_nocor, x_lc_test_final_nocor)
#Create the model's confusion matrix
cm_xgb_final_nocor <- confusionMatrix(prediction_xgb_final_nocor, lc_test_final_nocor$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_xgb_final_nocor <- tidy(cm_xgb_final_nocor) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "XGBoost - Final (No_Corr)"=estimate)

#Basic metrics of the XGB models
metrics_xgbs <- left_join(tidy_cm_xgb, tidy_cm_xgb_smote, by="Metric") %>%
  left_join(.,tidy_cm_xgb_final, by="Metric") %>%
  left_join(.,tidy_cm_xgb_final_nocor, by="Metric")
metrics_xgbs %>% mutate(Metric = str_to_title(Metric)) %>%
  kable(format = "pandoc",
        caption = "Basic Metric Scores - XGB, Original and Modified Datasets") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Calculate and compare the AUROCs - XGBs
models_list_xgbs <- list(XGB_orig = model_xgb, XGB_balanced = model_xgb_smote,
                          XGB_final = model_xgb_final,
                          XGB_final_No_Cor = model_xgb_final_nocor)
models_results_xgbs<- resamples(models_list_xgbs)
AUROCs_xgbs <- summary(models_results_xgbs)
AUROCs_xgbs_df <- data.frame(AUROCs_xgbs$statistics$ROC[,4])
AUROCs_xgbs_df <- cbind(rownames(AUROCs_xgbs_df), AUROCs_xgbs_df)
rownames(AUROCs_xgbs_df) <- NULL
colnames(AUROCs_xgbs_df) <- c("Model", "Mean AUROC")
AUROCs_xgbs_df %>%
  kable(caption = "Mean AUROC for XGBoost per each dataset") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Plot the ROCs with AUROCs - all XGBs
auroc_xgbs <- evalm(list(model_xgb, model_xgb_smote, model_xgb_final, model_xgb_final_nocor),
                    cols = c("red", "blue", "plum1", "turquoise"),
                    fsize=10,
                    plots=c("r"),
                    rlinethick = 0.5,
                    dlinethick = 0.5,
                    title="Original And Modified Datasets")
auprc_xgbs <- evalm(list(model_xgb, model_xgb_smote, model_xgb_final, model_xgb_final_nocor),
```

```
                    cols = c("red", "blue", "plum1", "turquoise"),
                    fsize=10,
                    plots=c("pr"),
                    rlinethick = 0.5,
                    dlinethick = 0.5,
                    title="Original And Modified Datasets")
#Combine ROCs and PRCs into one plot - all XGBs
ggarrange(auroc_xgbs$roc, auprc_xgbs$proc,
          ncol = 1, nrow = 2)

#Show the confusion matrix of the xgbs
cm_xgb$table
cm_xgb_smote$table
cm_xgb_final_nocor$table


#Build a RF model on the FINAL dataset (balanced + added features)
set.seed(16)
model_rf_final <- train(Death ~ ., data = lc_train_final,
                        method = "rf",
                        metric = "ROC",
                        trControl = control)
#use the model to predict the observations from the lc_test_final features
prediction_rf_final <- predict(model_rf_final, x_lc_test_final)
#Create the model's confusion matrix
cm_rf_final <- confusionMatrix(prediction_rf_final, lc_test_final$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_rf_final <- tidy(cm_rf_final) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "RF - Final"=estimate)

#Compare the performance with RF original and balanced
#Basic metrics of the RF models
metrics_rfs <- left_join(tidy_cm_rf, tidy_cm_rf_smote, by="Metric") %>%
  left_join(.,tidy_cm_rf_final, by="Metric")
metrics_rfs %>% mutate(Metric = str_to_title(Metric)) %>%
  kable(format = "pandoc",
        caption = "Basic Metric Scores - RF, Original and Modified Datasets") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Show the CM
cm_rf$table
cm_rf_smote$table
cm_rf_final$table

#Build a SVM model on the FINAL dataset (balanced + added features)
if(!require(kernlab)) install.packages("kernlab", repos = "http://cran.us.r-project.org")
library(kernlab)
set.seed(16)
model_svm_final <- train(Death ~ ., data = lc_train_final,
                         method = "svmRadial",
                         metric="ROC",
```

```
                            trControl = control)
detach("package:kernlab", unload = TRUE)
#use the model to predict the observations from the lc_test_final features
prediction_svm_final <- predict(model_svm_final, x_lc_test_final)
#Create the model's confusion matrix
cm_svm_final <- confusionMatrix(prediction_svm_final, lc_test_final$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_svm_final <- tidy(cm_svm_final) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "SVM - Final"=estimate)
#Show the CM and basic metrics
cm_svm_final$table
tidy_cm_svm_final

#Build a Model Averaged Neural Network model on the FINAL dataset (balanced + added features)
if(!require(nnet)) install.packages("nnet", repos = "http://cran.us.r-project.org")
library(nnet)
set.seed(16)
model_avnnet_final <- train(Death ~ ., data = lc_train_final,
                            method = "avNNet",
                            metric="ROC",
                            trControl = control)
detach("package:nnet", unload = TRUE)
#use the model to predict the observations from the lc_test_final features
prediction_avnnet_final <- predict(model_avnnet_final, x_lc_test_final)
#Create the model's confusion matrix
cm_avnnet_final <- confusionMatrix(prediction_avnnet_final, lc_test_final$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_avnnet_final <- tidy(cm_avnnet_final) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "AvNNet - Final"=estimate)

#Build a Model Averaged NN model on the FINAL dataset without highly correlated variables
if(!require(nnet)) install.packages("nnet", repos = "http://cran.us.r-project.org")
library(nnet)
set.seed(16)
model_avnnet_final_nocor <- train(Death ~ ., data = lc_train_final_nocor,
                            method = "avNNet",
                            metric="ROC",
                            trControl = control)
detach("package:nnet", unload = TRUE)
#use the model to predict the observations from the lc_test_final features
prediction_avnnet_final_nocor <- predict(model_avnnet_final_nocor, x_lc_test_final_nocor)
#Create the model's confusion matrix
cm_avnnet_final_nocor <- confusionMatrix(prediction_avnnet_final_nocor, lc_test_final_nocor$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_avnnet_final_nocor <- tidy(cm_avnnet_final_nocor) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "AvNNet - Final_NoCor"=estimate)

#Basic metrics of the NNet models
metrics_nn <- left_join(tidy_cm_avnnet_final, tidy_cm_avnnet_final_nocor, by="Metric")
metrics_nn %>% mutate(Metric = str_to_title(Metric)) %>%
  kable(format = "pandoc", caption = "Basic Metric Scores - NeuralNetwork") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
```

```
                full_width = FALSE)

#Show the CMs of the NNet models
cm_avnnet_final$table
cm_avnnet_final_nocor$table


#Plot the ROCs with AUROCs - both NNs
auroc_nn <- evalm(list(model_avnnet_final, model_avnnet_final_nocor),
                    cols = c("plum1", "turquoise"),
                  gnames = c("NNet_final", "NNet_final_NoCor"),
                    fsize=10,
                    plots=c("r"),
                    rlinethick = 0.5,
                    dlinethick = 0.5,
                    title="Final Dataset")
auprc_nn <- evalm(list(model_avnnet_final, model_avnnet_final_nocor),
                  cols = c("plum1", "turquoise"),
                  gnames = c("NNet_final", "NNet_final_NoCor"),
                  fsize=10,
                  plots=c("pr"),
                  rlinethick = 0.5,
                  dlinethick = 0.5,
                  title="Final Dataset")
#Combine ROCs and PRCs into one plot - all XGBs
ggarrange(auroc_nn$roc, auprc_nn$proc,
          ncol = 1, nrow = 2)


####EXPLORE THE IMPACT OF ONE HOT ENCODING

#Cleaned original dataset without One Hot Encoding (=DGN dataset)
lc_orig_cleaned_final

#Plot the correlation plot of this dataset
correlationMatrix_dgn <- cor(lc_orig_cleaned_final)
res_dgn <- cor.mtest(lc_orig_cleaned_final, conf.level = .95) #used for p.mat argument to determine sta
corrplot(correlationMatrix_dgn, addrect = 6, tl.col = "black",
         p.mat = res_dgn$p, insig = "blank")

#Determine which strongly correlated variables to be removed
strong_cor_dgn <- findCorrelation(correlationMatrix_dgn, cutoff=0.5)
colnames(lc_orig_cleaned_final[,strong_cor_dgn])

#Split the DGN dataset to train and test set; test set will have 20% of the lc data
set.seed(16)
test_index_dgn <- createDataPartition(y = lc_orig_cleaned_final$Death, times = 1, p = 0.2, list = FALSE)
lc_train_dgn <- lc_orig_cleaned_final[-test_index_dgn,] %>%
  mutate(Death = factor(Death, labels = c("Alive", "Dead")))
lc_test_dgn <- lc_orig_cleaned_final[test_index_dgn,] %>%
  mutate(Death = factor(Death, labels = c("Alive", "Dead")))
x_lc_test_dgn <- lc_test_dgn %>% dplyr::select(-Death)
```

```r
#Build a Random Forest model - DGN
set.seed(16)
model_rf_dgn <- train(Death ~ ., data = lc_train_dgn,
                      method = "rf",
                      metric="ROC",
                      trControl = control)
#use the model to predict the observations from the lc_test features
prediction_rf_dgn <- predict(model_rf_dgn, x_lc_test_dgn)
#Create the model's confusion matrix
cm_rf_dgn <- confusionMatrix(prediction_rf_dgn, lc_test_dgn$Death)


#Build a XGB model - DGN
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
library(xgboost) #open here to avoid masking some dplyr functions (slice)
set.seed(16)
model_xgb_dgn <- train(Death ~ ., data = lc_train_dgn,
                       method = "xgbTree",
                       metric="ROC",
                       trControl = control)
detach("package:xgboost", unload = TRUE)
#use the model to predict the observations from the lc_test features
prediction_xgb_dgn <- predict(model_xgb_dgn, x_lc_test_dgn)
#Create the model's confusion matrix
cm_xgb_dgn <- confusionMatrix(prediction_xgb_dgn, lc_test_dgn$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_xgb_dgn <- tidy(cm_xgb_dgn) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "XGBoost - DGN original"=estimate)


###DATA BALANCING
#Apply the SMOTE method to the lc_train_dgn dataset
set.seed(16)
lc_train_dgn_smote <- SMOTE(Death ~ ., data = lc_train_dgn, k=5)
#Check the balance of the target variable
table(lc_train_dgn_smote$Death)

###Add new engineered variables
lc_train_dgn_final <- lc_train_dgn_smote %>%
  mutate("FEV1/FVC" = FEV1/FVC,
         "FVC*FEV1" = FVC*FEV1,
         "FVC*FEV1^2" = FVC*(FEV1)^2,
         "FVC^2*FEV1" = (FVC^2)*FEV1,
         "FVC^2" = FVC^2,
         "Age*FVC" = Age*FVC,
         "Age*FEV1" = Age*FEV1,
         "Age/Tumor_size" = Age/Tumor_size,
         "FVC*Tumor_size" = FVC*Tumor_size)
lc_test_dgn_final <- lc_test_dgn %>%
  mutate("FEV1/FVC" = FEV1/FVC,
         "FVC*FEV1" = FVC*FEV1,
         "FVC*FEV1^2" = FVC*(FEV1)^2,
         "FVC^2*FEV1" = (FVC^2)*FEV1,
         "FVC^2" = FVC^2,
```

```
          "Age*FVC" = Age*FVC,
          "Age*FEV1" = Age*FEV1,
          "Age/Tumor_size" = Age/Tumor_size,
          "FVC*Tumor_size" = FVC*Tumor_size)
x_lc_test_dgn_final <- lc_test_dgn_final %>% dplyr::select(-Death)


#Build a XGB model - DGN final
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
library(xgboost) #open here to avoid masking some dplyr functions (slice)
set.seed(16)
model_xgb_dgn_final <- train(Death ~ ., data = lc_train_dgn_final,
                             method = "xgbTree",
                             metric="ROC",
                             trControl = control)
detach("package:xgboost", unload = TRUE)
#use the model to predict the observations from the lc_test features
prediction_xgb_dgn_final <- predict(model_xgb_dgn_final, x_lc_test_dgn_final)
#Create the model's confusion matrix
cm_xgb_dgn_final <- confusionMatrix(prediction_xgb_dgn_final, lc_test_dgn_final$Death)
#Transform the CM into the tidy format and extract key evaluation metrics
tidy_cm_xgb_dgn_final <- tidy(cm_xgb_dgn_final) %>% slice(1,4,5,8,9,10,11,14) %>%
  dplyr::select(term, estimate) %>% rename(Metric=term, "XGBoost - DGN Final"=estimate)

#Basic metrics of the XGB models with and without one hot encoding
metrics_xgbs_dgn <- left_join(tidy_cm_xgb_dgn, tidy_cm_xgb, by="Metric") %>%
  left_join(.,tidy_cm_xgb_dgn_final, by="Metric") %>%
  left_join(.,tidy_cm_xgb_final, by="Metric")
metrics_xgbs_dgn %>% mutate(Metric = str_to_title(Metric)) %>%
  kable(format = "pandoc",
        caption = "Basic Metric Scores - Impact of One-Hot Encoding") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#AUROC with caret
models_list_dgn <- list(XGB_dgn_orig = model_xgb_dgn, XGB_orig = model_xgb,
                        XGB_dgn_final = model_xgb_dgn_final, XGB_final = model_xgb_final)
models_results_dgn <- resamples(models_list_dgn)
AUROCs_dgn <- summary(models_results_dgn)
AUROCs_dgn_df <- data.frame(AUROCs_dgn$statistics$ROC[,4])
AUROCs_dgn_df <- cbind(rownames(AUROCs_dgn_df), AUROCs_dgn_df)
rownames(AUROCs_dgn_df) <- NULL
colnames(AUROCs_dgn_df) <- c("Model", "Mean AUROC")
AUROCs_dgn_df %>%
  kable(format = "pandoc", caption = "Mean AUROC - Impact of One-Hot Encoding") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

######################
#RESULTS COMPARISON
```

```
######################

#Basic metrics of the best individual models
metrics_all_top <- left_join(tidy_cm_xgb_final, tidy_cm_xgb_final_nocor, by="Metric") %>%
  left_join(.,tidy_cm_rf_final, by="Metric") %>%
  left_join(.,tidy_cm_svm_final, by="Metric") %>%
  left_join(.,tidy_cm_avnnet_final, by="Metric") %>%
  left_join(.,tidy_cm_avnnet_final_nocor, by="Metric")
metrics_all_top %>% mutate(Metric = str_to_title(Metric)) %>%
  kable(format = "pandoc",
        caption = "Basic Metric Scores - All Top Models") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Mean AUROC all top performing models (CARET)
models_list_all_top <- list(XGBoost_final = model_xgb_final,
                            XGBoost_final_NoCor = model_xgb_final_nocor,
                            RandomForest_final = model_rf_final,
                            SVM_final = model_svm_final,
                            NNet_final = model_avnnet_final,
                            NNet_final_NoCor = model_avnnet_final_nocor)
models_results_all_top <- resamples(models_list_all_top)
AUROCs_all_top <- summary(models_results_all_top)
AUROCs_all_top_df <- data.frame(AUROCs_all_top$statistics$ROC[,4])
AUROCs_all_top_df <- cbind(rownames(AUROCs_all_top_df), AUROCs_all_top_df)
rownames(AUROCs_all_top_df) <- NULL
colnames(AUROCs_all_top_df) <- c("Model", "Mean AUROC")
AUROCs_all_top_df %>%
  kable(format = "pandoc", caption = "Mean AUROC - All Top Models") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)

#Plot the AUROCs and their variability for the top models
bwplot(models_results_all_top, metric="ROC")

#AUPRC all top performing models
auprc_all_top <- evalm(list(model_xgb_final, model_xgb_final_nocor,
                            model_avnnet_final, model_avnnet_final_nocor),
                       gnames = c("XGBoost_final", "XGBoost_final_NoCor",
                            "NNet_final", "NNet_final_NoCor"),
                       cols = c("turquoise", "turquoise1", "plum", "plum1"),
                   fsize=10,
                   plots=c("pr"),
                   rlinethick = 0.5,
                   dlinethick = 0.5,
                   title="PRC - Top Models")
auprc_all_top$proc
```

6.2 Software Environment

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          0.3
## year           2020
## month          10
## day            10
## svn rev        79318
## language       R
## version.string R version 4.0.3 (2020-10-10)
## nickname       Bunny-Wunnies Freak Out
```