Predictive Modeling of COVID-19 Outcomes:

Identifying important features affecting patient Severity and Mortality

Presented by:

Shubham Kamboj

and

Supervised by:

Dr. Sudheesh K Kattumannil

Indian Statistical Institute

Chennai

01-06-2023

Objective

The COVID-19 pandemic has had a significant impact on public health worldwide, with millions of cases and thousands of deaths reported globally.

Identifying the factors that affect COVID-19 patient severity and mortality is critical to improving patient outcomes and reducing the burden on healthcare systems.

This project aims to develop a predictive model using machine learning algorithms after identify the key features affecting COVID-19 patient's severity and mortality.

The developed model can also aid in identifying patients at high risk of severe outcomes, which can help in allocating resources and planning interventions to reduce patient's morbidity and mortality.

Data used

Dimension : 305×111

The variables we have are clinical features of patient that were recorded at the time of admission in the hospital.

There are total 111 features. Approx. 40% are continuous and remaining are categorical features.

There are lot's of missing values in the data set.

Some variables...

We have information about Name, Age, Gender, Height, Weight, BMI, Fever, Cough, Nausea, Asthma, Spo2, BP, Heart Rate, and some other medical history like if Patient is Diabetic or not.

Result of some diagnostic test is also part of data set. For example...

Anemia, Platelet counts, Blood Urea nitrogen level, Liver function test abnormality, Acute kidney injury, HbA1c, Patches present in chest X-ray, Side of the lung involved with patch and Lobe involved with patch.

Information about death, length of hospital stay and if the patient was categorised as severe or not is also available.

In total 111 such features were present in the data set.

Data Cleaning

As the data set we have received is very raw. It needs to be processed and transformed before it can be analyzed and used to make any model.

This process involves data cleaning, which includes imputing missing values and identifying and dealing with outliers.

Outliers were present in almost every variable. As some patient were whose test result were at extreme.

Dealing with missing values

For some patients information about more than 30 attributes were missing.

We have removed patient data from the study if 15 or more parameter's observation are missing for them. As imputing them may lead to biasness in the data.

As they were only 36 out of 305 so removing them will not lead to loss of so much information.

After this, significant number of missing values were gone. Only less than 5-7% missing values were left for all variables. That can be imputed with central measures.

Why Feature Selection?

Feature selection help us in selecting the relevant features which is playing significant role in classifying the subject.

One easy option is to start building the model considering all the variables. But biggest drawback of this is the loss of interpretability of the model.

Variable selection is an important aspect of model building. It helps in building predictive models free from correlated variables, biases and unwanted noise.

Feature selection using Boruta

What is Boruta and how to perform feature selection with this?

Boruta is a feature selection algorithm. Precisely, it works as a wrapper algorithm around Random Forest. This package derives its name from a demon in Slavic mythology who dwelled in pine forests.

We know that feature selection is a crucial step in predictive modeling. This technique achieves supreme importance when a data set comprised of several variables, as in our data set, is given for model building.

Boruta is the algorithm of choice to deal with such data sets. Particularly when one is interested in understanding the mechanisms related to the variable of interest, rather than just building a black box predictive model with good prediction accuracy.

How does Boruta work?

Feature Selection when coronaseve is response variable.

```r
# Feature Selection
set.seed(111)
boruta_seve <- Boruta(coronaseve ~ ., data = data_seve, maxRuns = 500)

# The above command applies the Boruta feature selection algorithm to
# the dataset data_seve . The formula death ~ . specifies that
# the variable death is the outcome variable, and . indicates that all other
# variables in the dataset are considered as potential predictors. The doTrace
# parameter controls the level of verbosity during the Boruta analysis, with
# a value of 2 indicating more detailed output. The maxRuns parameter
# specifies the maximum number of iterations to run the algorithm. The result
# of the Boruta analysis is stored in the boruta_seve object.

print(boruta_seve)

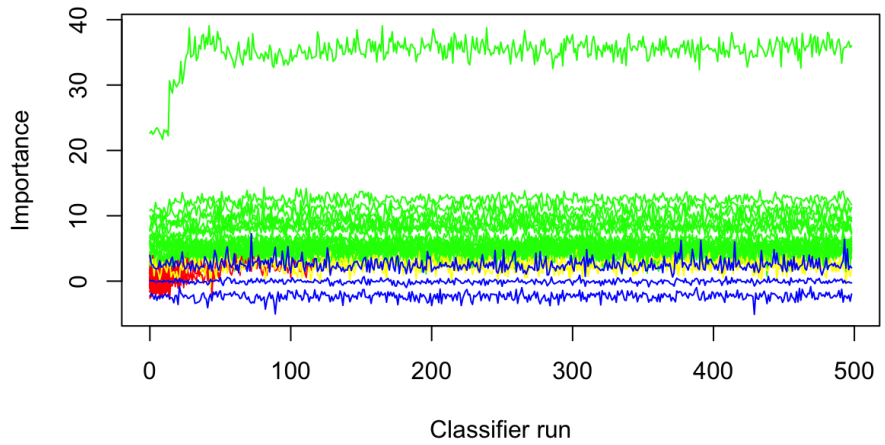
#The above command prints the summary or information about the boruta object.
# It displays the variables considered in the Boruta analysis, their
# importance scores, and the final decision on whether they are selected
# as important predictors or not. This information helps assess the relevance
# of variables in predicting the outcome variable (death) based on the
# Boruta analysis.

plot(boruta_seve, las = 2, cex.axis = 0.7)
# Above command is used to create a plot of the Boruta analysis results.
# The boruta_seve object is passed as the argument, and additional parameters
#  las and cex.axis are used to modify the appearance of the plot. las controls
# the orientation of the axis labels, and cex.axis controls the size of the
# axis labels. By customizing these parameters, you can enhance the
# readability of the plot and visualize the importance of variables determined
# by Boruta.

plotImpHistory(boruta_seve)


bor <- TentativeRoughFix(boruta_seve)
print(bor)
att_seve<-attStats(boruta_seve)
```

```
Boruta performed 499 iterations in 12.95194 secs.
 21 attributes confirmed important: abgph, abgspo2, age, bun, cxrpneu
and 16 more;
 69 attributes confirmed unimportant: abgacidos, af0, afl, aki, anemia
and 64 more;
 2 tentative attributes left: arrythmia, dyspnoea;
Boruta performed 499 iterations in 12.95194 secs.
Tentatives roughfixed over the last 499 iterations.
 23 attributes confirmed important: abgph, abgspo2, age, arrythmia, bun
and 18 more;
 69 attributes confirmed unimportant: abgacidos, af0, afl, aki, anemia
and 64 more;
```

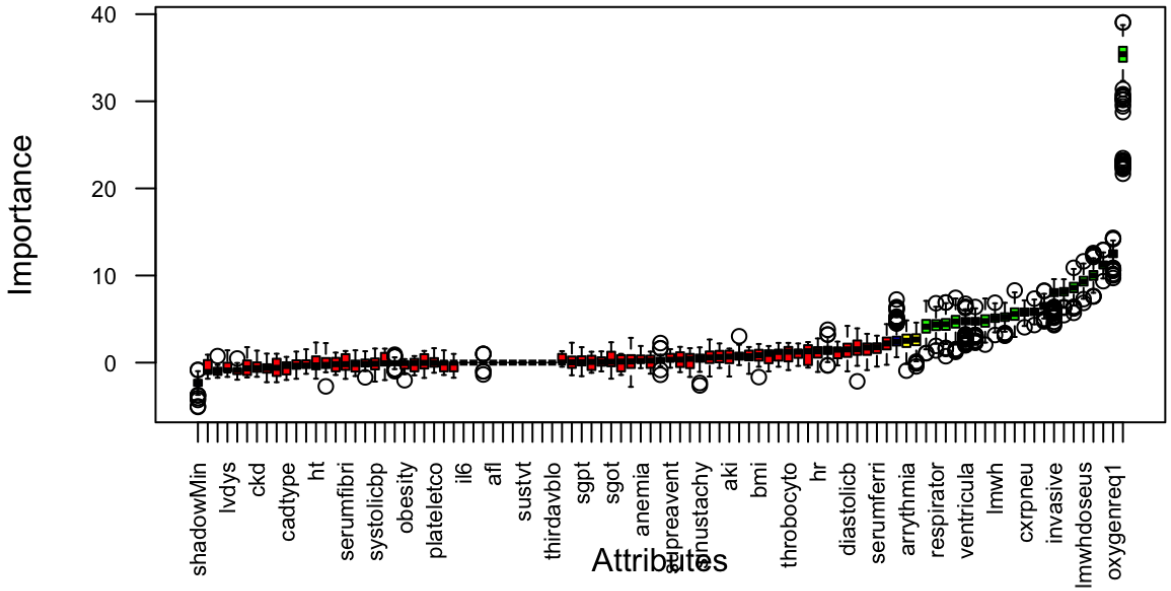


Above is a plot of the variable importance history during the Boruta analysis. The boruta object is passed as the argument, and the function generates a plot that shows the change in variable importance over iterations.

This plot helps understand how the importance of variables evolves during the Boruta feature selection process and can assist in determining the optimal set of variables to include in the final model.

Plot

This is a plot of the Boruta analysis results. The boruta_seve object is passed as the argument, and additional parameters las and cexaxis are used to modify the appearance of the plot. las controls the orientation of the axis labels, and cex.axis controls the size of the axis labels.
By customizing these parameters, you can enhance the readability of the plot and visualize the importance of variables determined by Boruta.



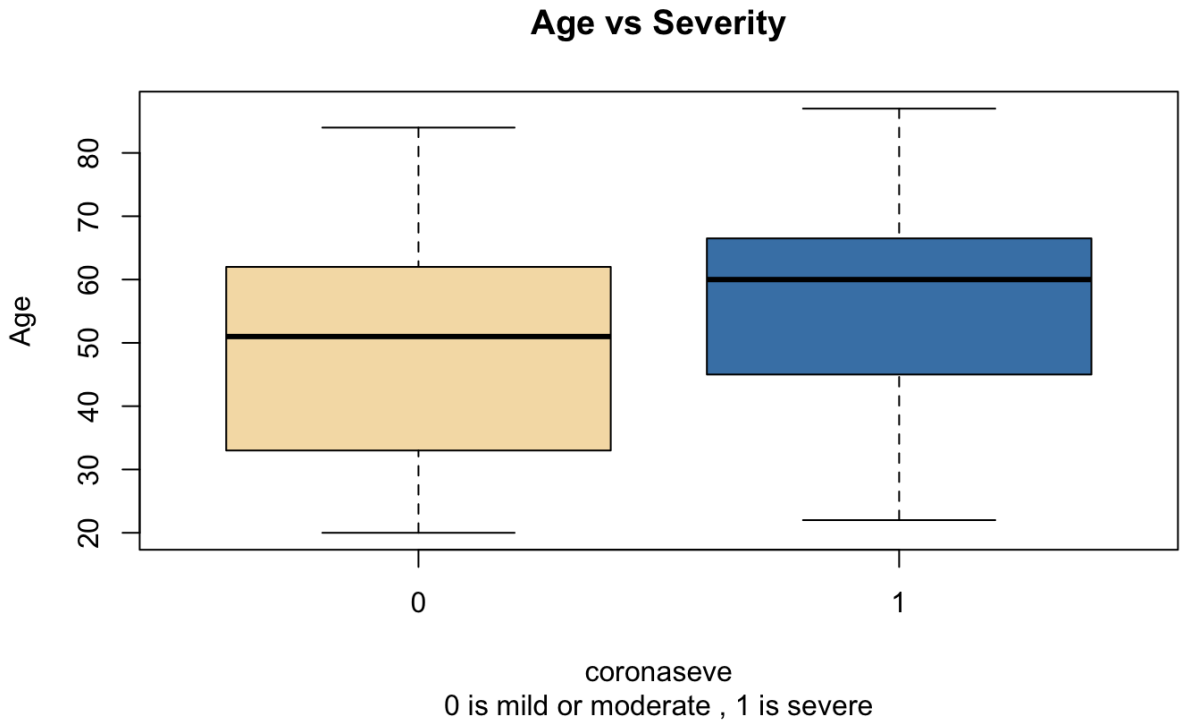
Important features for coronaseve

getConfirmedFormula() gives us the formula that should be used while building model with confirmed variables i.e. variables that was not rejected by *Boruta()*

```
formula_confirmed_seve <- getConfirmedFormula(boruta_seve)
formula_confirmed_seve
```

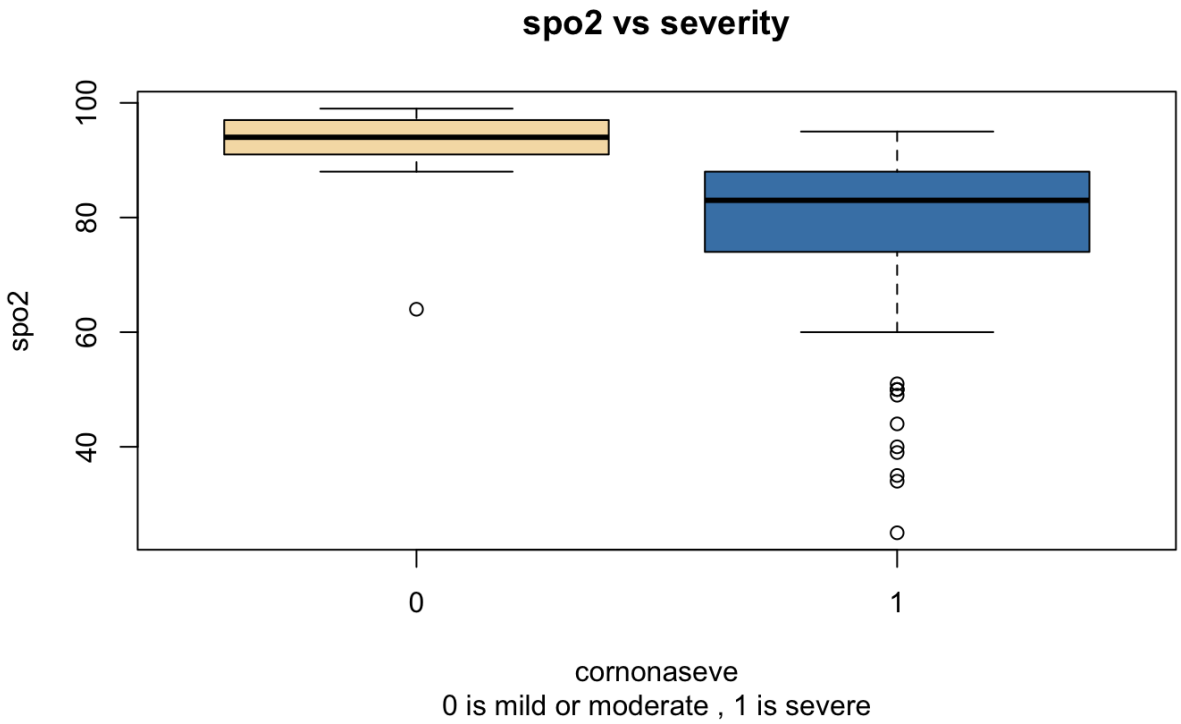
```
## coronaseve ~ age + respirator + spo2 + resrate + bun + abgph +
##     abgspo2 + ventricula + vpc + oxygenrequ + cxrpneu + patchprsen +
##     patchside + patchlobel + oxygenreq1 + noninvasie + invasive +
##     lmwh + lmwhdoseus + ramdesevir + steroid
## <environment: 0x1301f4698>
```

Age vs Corona severity



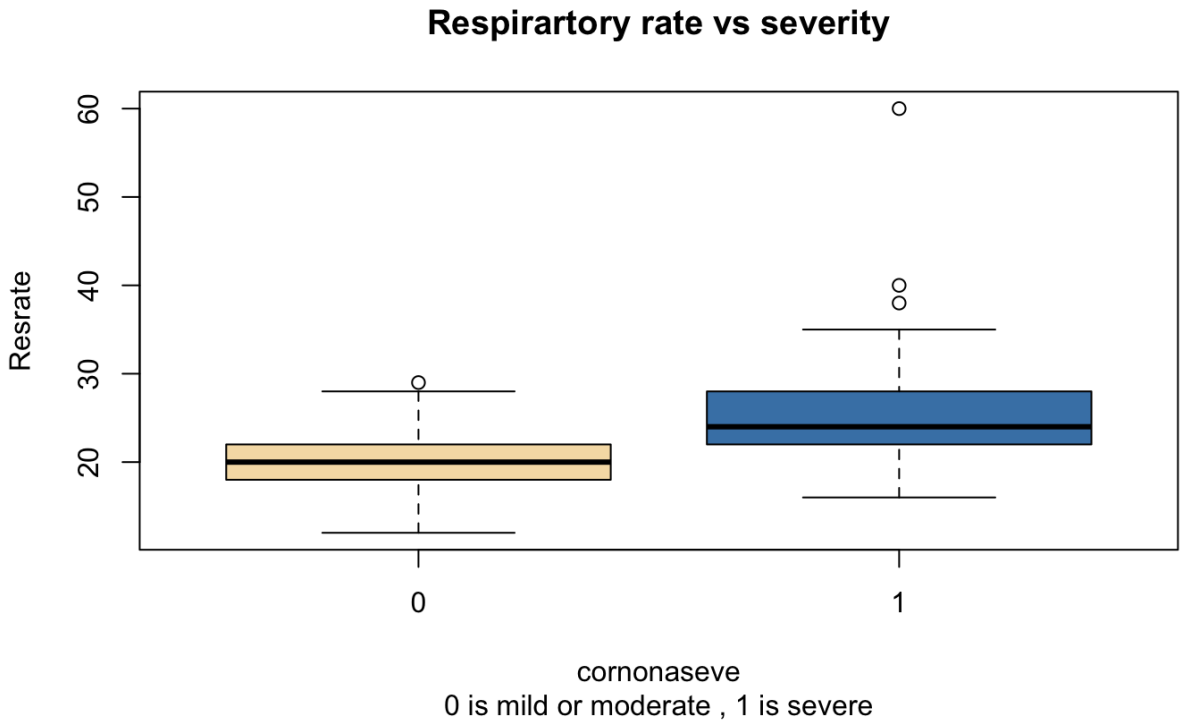
Age	Severe	Mild or Not severe
Mean	56.78	49.53
Median	60	51
sd	14.63	17.23

Spo2 vs Corona severity



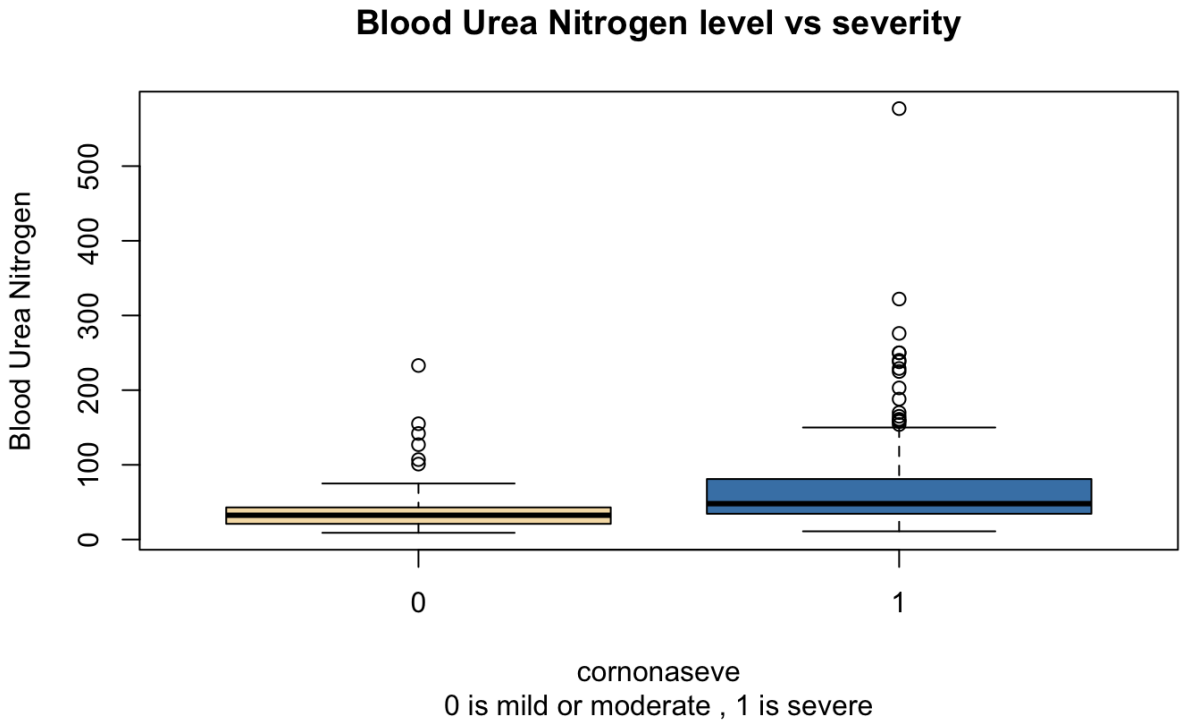
Age	Severe	Mild or Not severe
Mean	78.78	93.96
Median	83	94
sd	12.45	4.40

Respiratory rate vs Corona severity



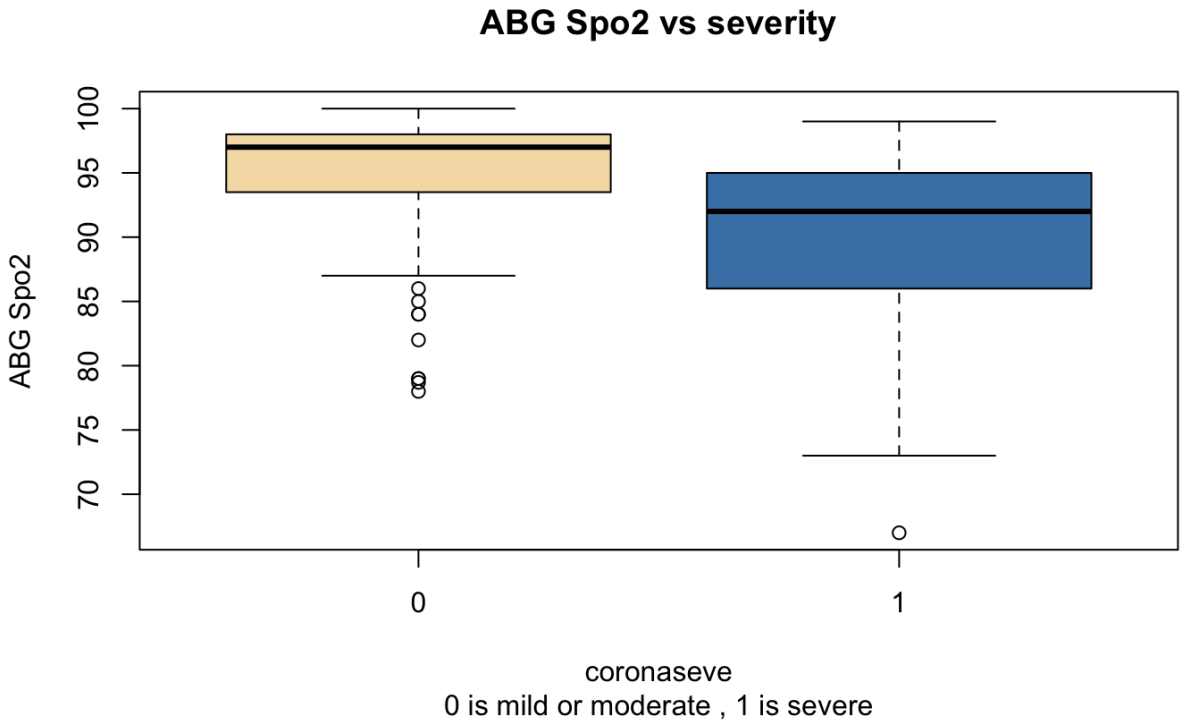
Age	Severe	Mild or Not severe
Mean	24.83	19.67
Median	24	20
sd	5.45	3.18

Blood urea nitrogen vs Corona severity



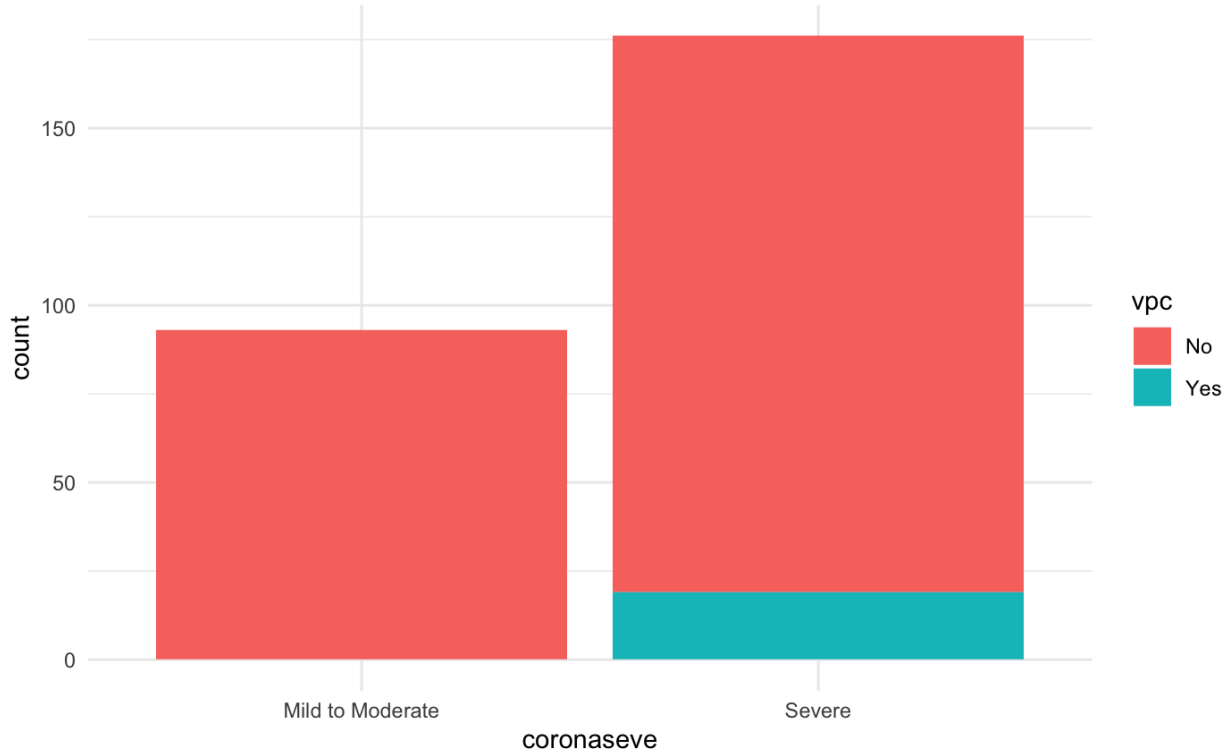
Age	Severe	Mild or Not severe
Mean	71.95	39.86
Median	48	32.50
sd	67.88	32.87

Arterial blood gas SpO2 vs Corona severity

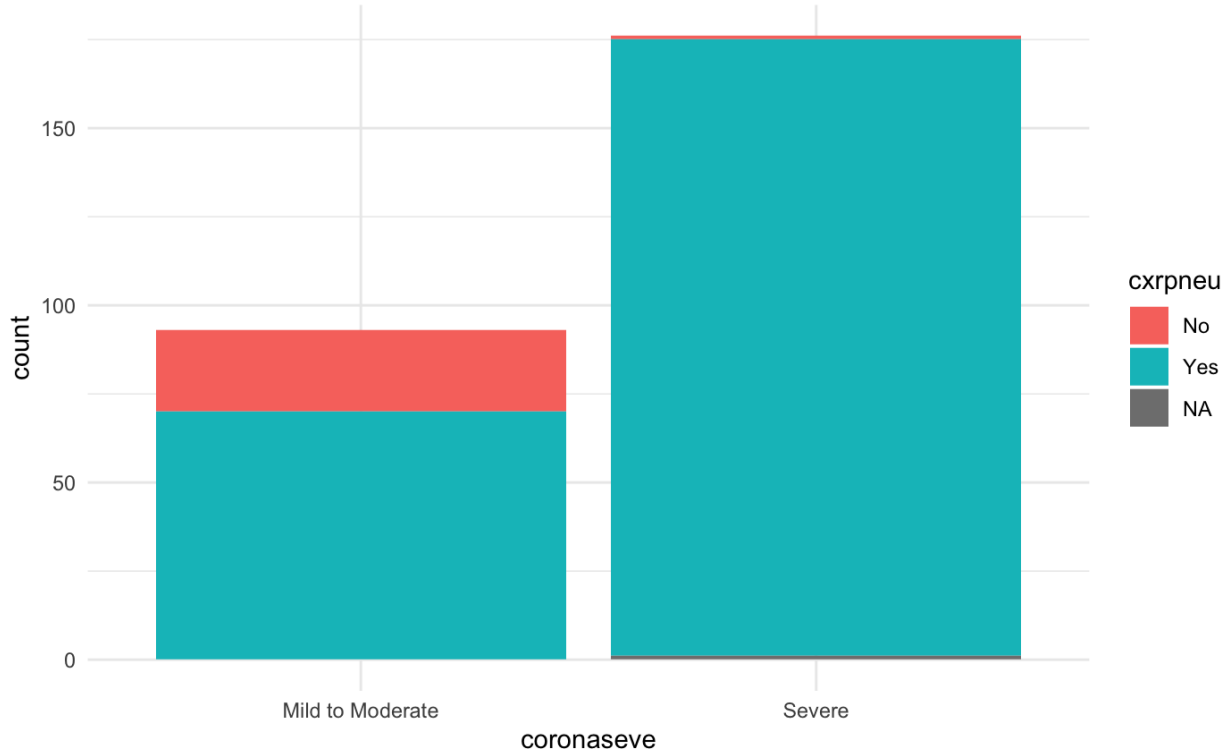


Age	Severe	Mild or Not severe
Mean	90.28	94.36
Median	92	97
sd	6.42	5.49

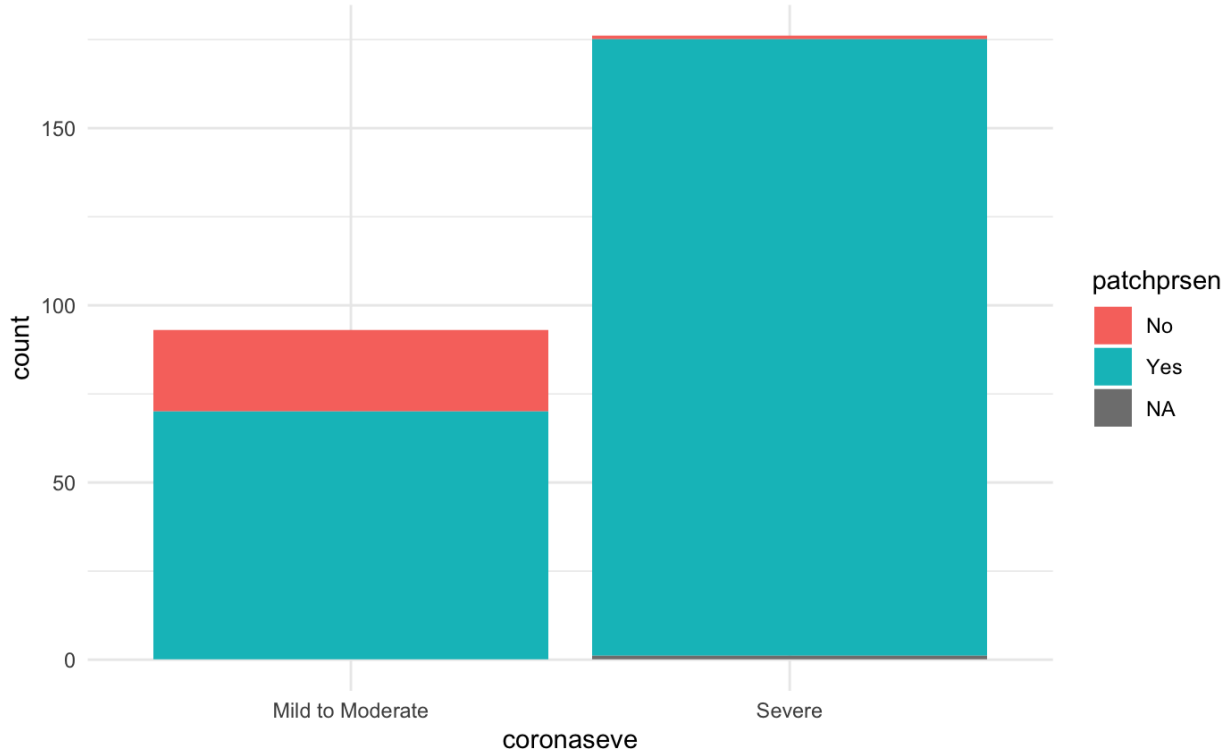
Ventricular premature Complex vs severity



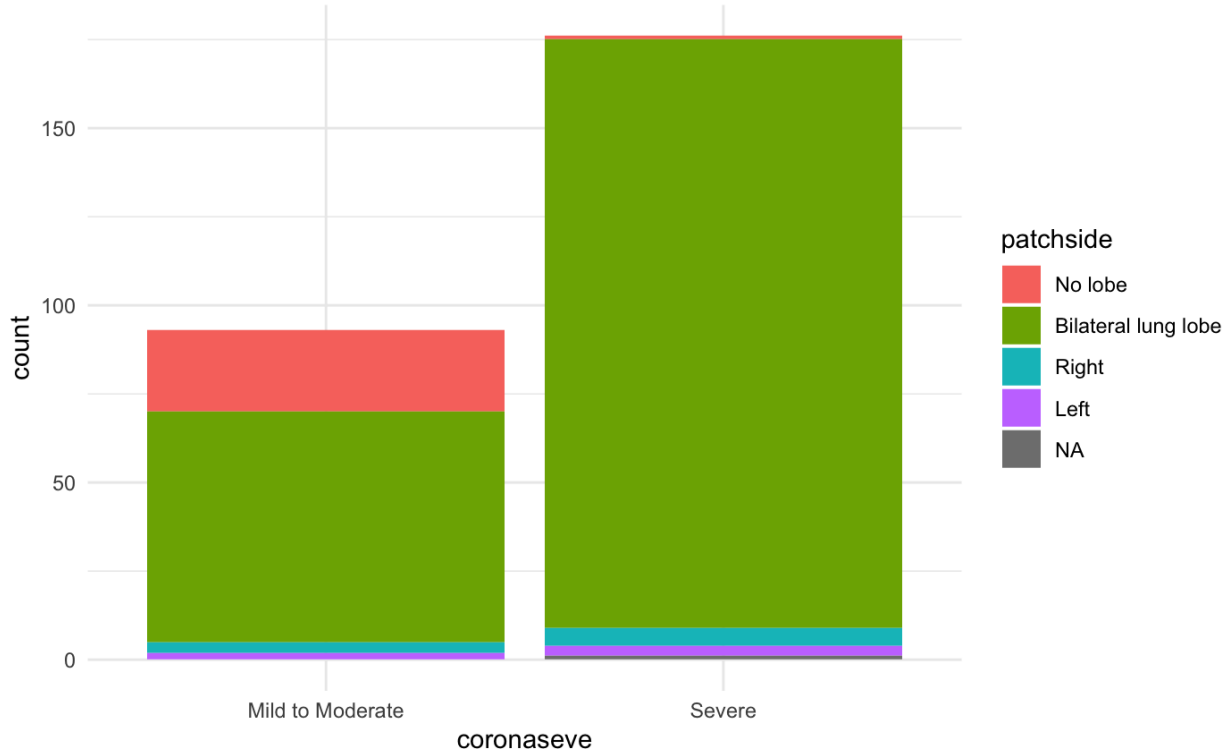
Pneumonia in Chest X-ray vs severity



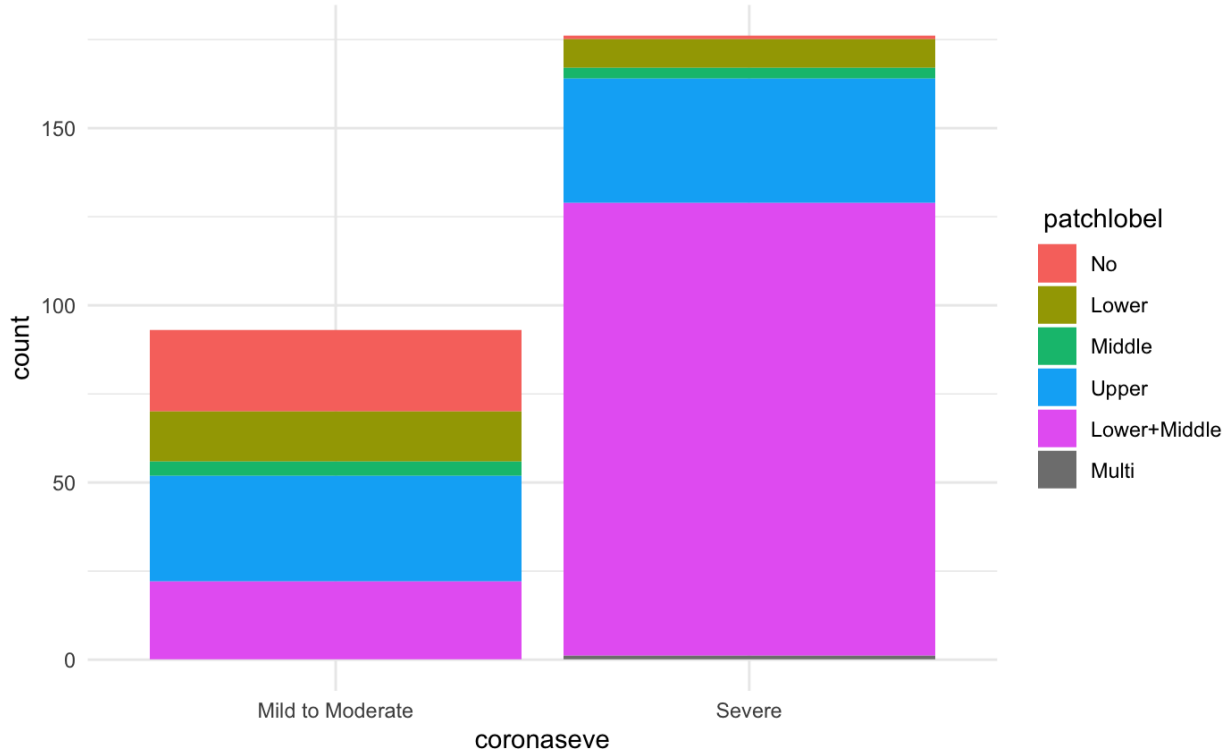
Patch present vs severity



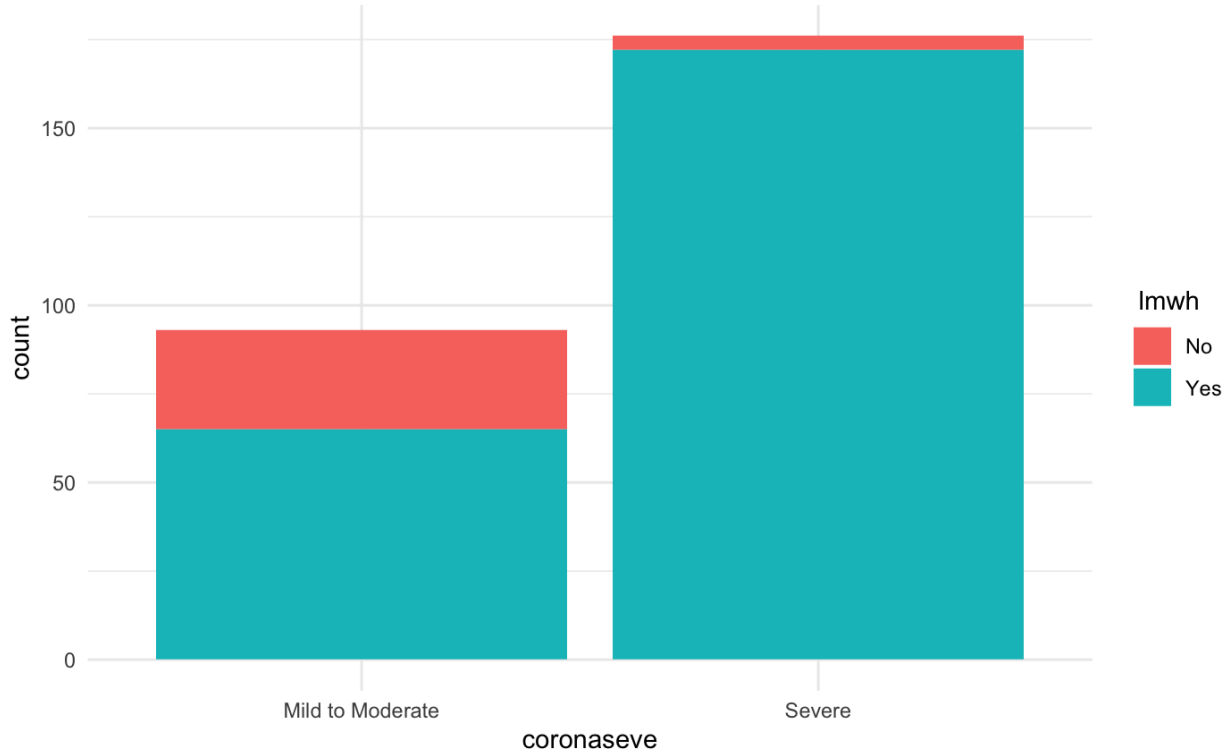
Patch-side vs severity



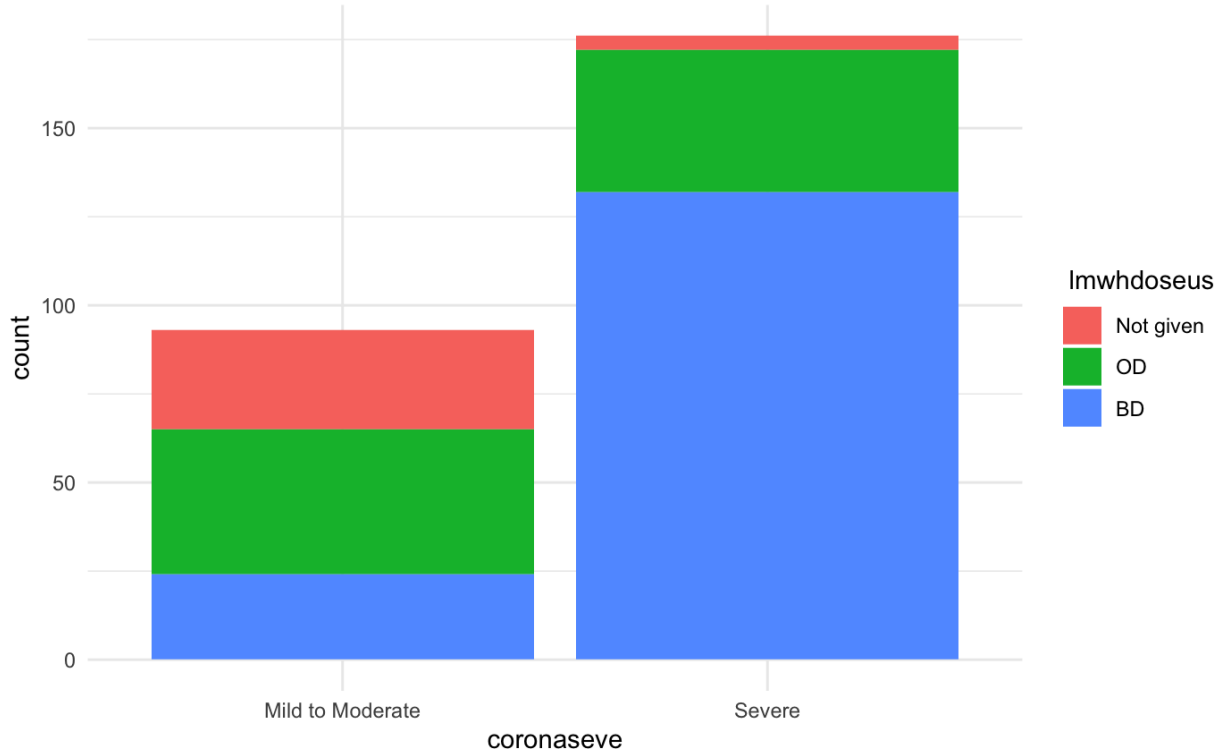
Patchlobel vs severity



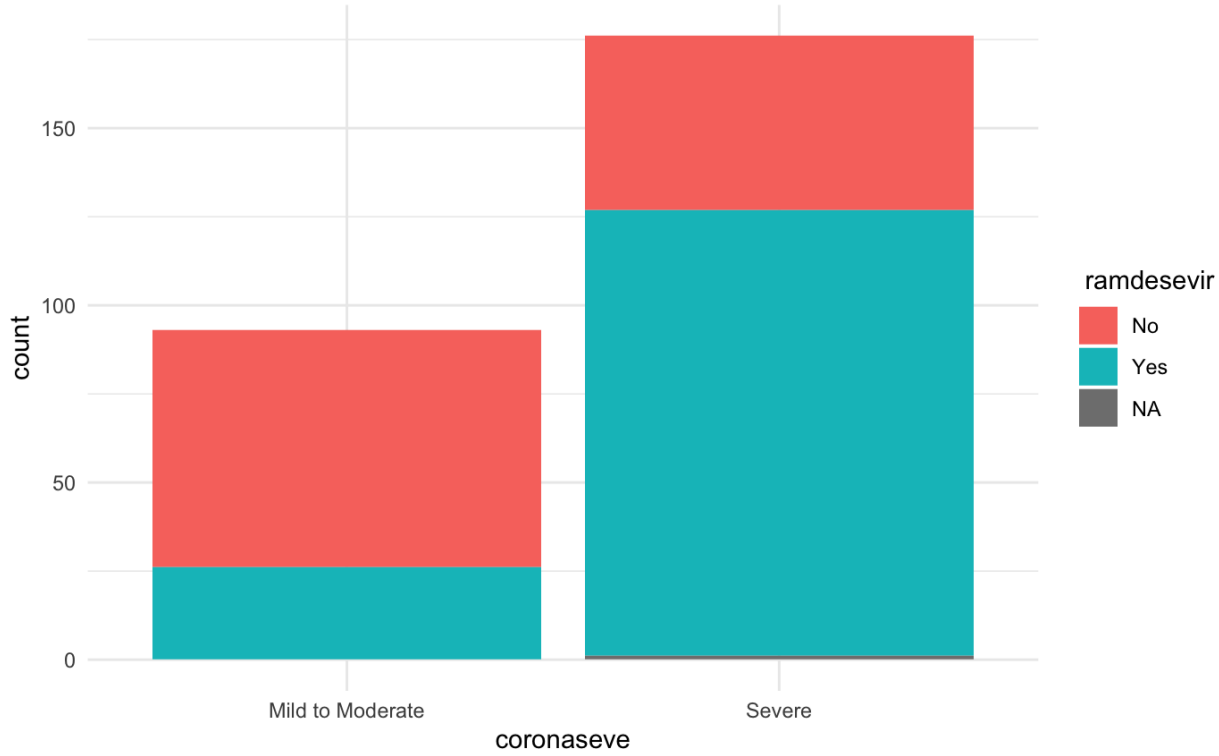
Low molecular weight heparin vs severity



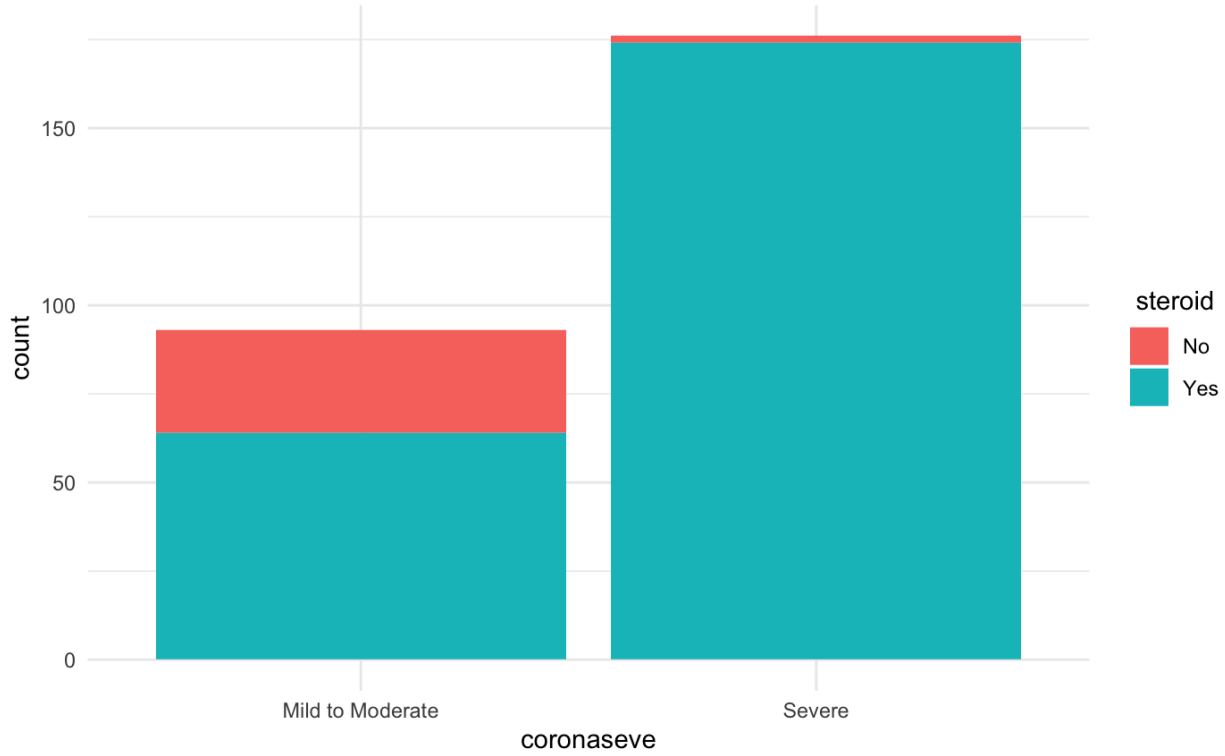
Dose of low molecular vs severity



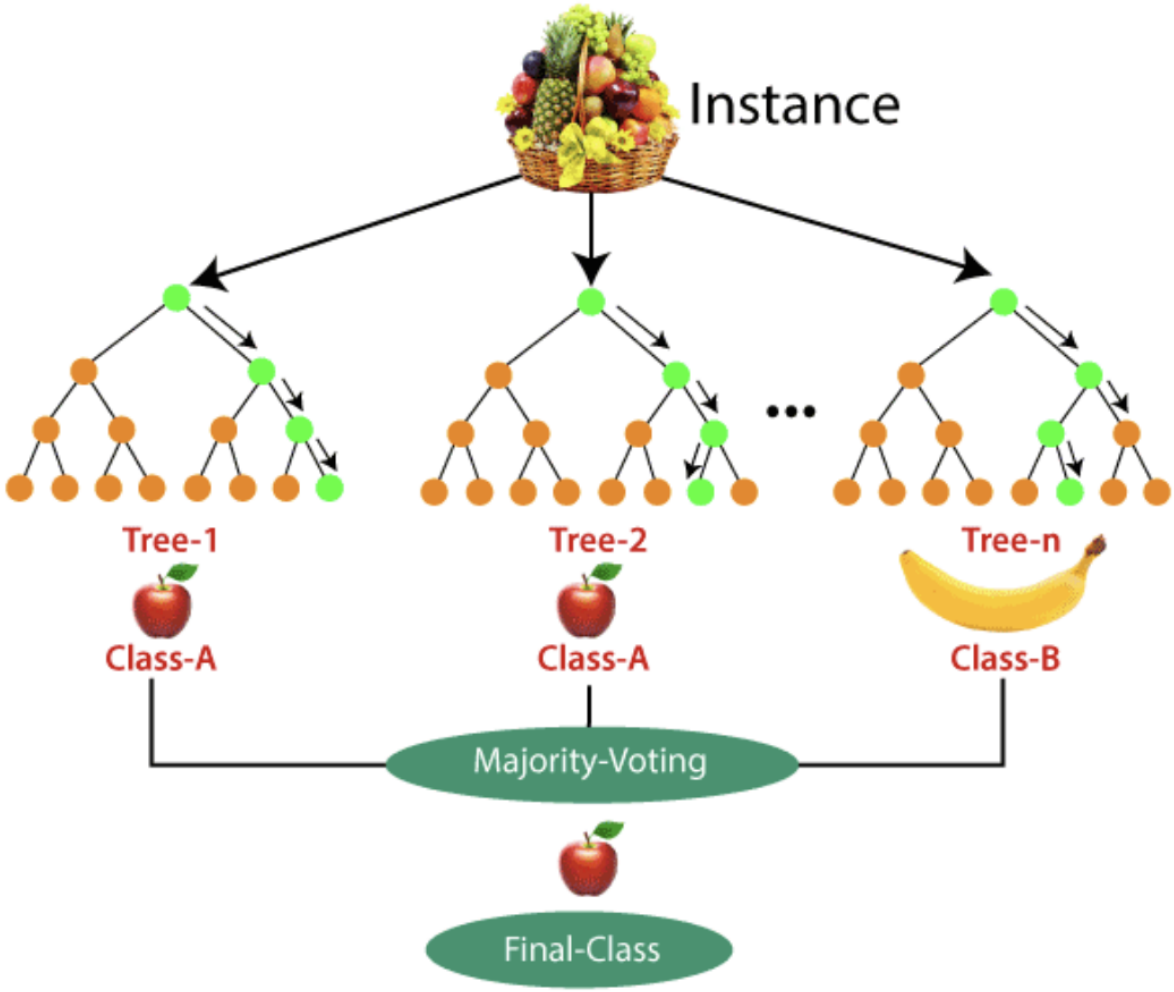
remdeseivier vs severity



Steroid vs severity



What is Random Forest?



Why we choose Random Forest?

Highly accurate predictions

It averages the predictions of multiple decision trees. Each decision tree is trained on a random subset of the data and considers only a subset of features, reducing the risk of overfitting and improving generalization.

Robust to overfitting

The randomness in feature selection and data sampling helps to decorrelate the trees and prevents them from memorizing noise or outliers in the training data.

Robust to missing data

It leverages the available data during training and can still make predictions on instances with missing values.

Less prone to bias

It reduces the risk of overfitting by aggregating predictions, resulting in more reliable and less biased models.

Efficient for large datasets

Random Forest can efficiently handle large datasets with many features and instances.

More Importantly, it worked in our case

Random Forest model with only Confirmed attributes for Severity Prediction

```
# model with 21 confirmed important predictors

```{r}
rf_seve21<- randomForest(formula_confirmed_seve, data=train_seve)

rf_seve21
# Prediction & Confusion Matrix - Test with 21 important variables only
p <- predict(rf_seve21, test_seve)
confusionMatrix(p, test_seve$coronaseve)

CM_rf_seve21 <- as.matrix(confusionMatrix(p, test_seve$coronaseve))
err_metric(CM_rf_seve21)
score_vec_rf_seve21 <- err_metric(CM_rf_seve21)
score_vec_rf_seve21


```
```

```
 randomForest(formula = formula_confirmed_seve, data = train_seve)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 3.09%
Confusion matrix:
   0   1 class.error
0 54   2  0.03571429
1  3 103  0.02830189
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 37  2
         1  0 68

               Accuracy : 0.9813
                 95% CI : (0.9341, 0.9977)
    No Information Rate : 0.6542
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9592

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.9714
         Pos Pred Value : 0.9487
         Neg Pred Value : 1.0000
             Prevalence : 0.3458
         Detection Rate : 0.3458
   Detection Prevalence : 0.3645
      Balanced Accuracy : 0.9857

       'Positive' Class : 0

[1] "Precision value of the model:  1"
[1] "Accuracy of the model:  0.98"
[1] "Recall value of the model:  0"
[1] "False Positive rate of the model:  0"
[1] "False Negative rate of the model:  0.05"
[1] "f1 score of the model:  0"
```

SVM model with Confirmed attributes only for Severity Prediction

````
**SVM for classification of Severity**

```{r}
# library(e1071) # library required for svm

svmfit_seve <- svm(formula_confirmed_seve , data = train_seve,
                   kernal = "linear" )
summary(svmfit_seve)
set.seed(9973)
# tune() performs cross validation
tune.out <- tune(svm , formula_confirmed_seve , data = train_seve,
                 kernal = "linear",
                 ranges = list(
                   cost = c(  1.8, 2, 3 ,3.5)

                 ))
summary(tune.out)
p<- predict(tune.out$best.model, newdata = test_seve)
confusionMatrix(p , test_seve$coronaseve)

CM_svmfit_seve <- as.matrix(confusionMatrix(p , test_seve$coronaseve))
err_metric((CM_svmfit_seve))
score_vec_svmfit_seve <- err_metric((CM_svmfit_seve))
score_vec_svmfit_seve
```
````

```
            Sensitivity : 0.7568
            Specificity : 0.9857
         Pos Pred Value : 0.9655
         Neg Pred Value : 0.8846
             Prevalence : 0.3458
         Detection Rate : 0.2617
   Detection Prevalence : 0.2710
      Balanced Accuracy : 0.8712

       'Positive' Class : 0

[1] "Precision value of the model:  0.76"
[1] "Accuracy of the model:  0.91"
[1] "Recall value of the model:  0.12"
[1] "False Positive rate of the model:  0.12"
[1] "False Negative rate of the model:  0.03"
[1] "f1 score of the model:  0.2"
[1] 0.91 0.76 0.97 0.88
[1] "Precision value of the model:  0.76"
[1] "Accuracy of the model:  0.91"
[1] "Recall value of the model:  0.12"
[1] "False Positive rate of the model:  0.12"
[1] "False Negative rate of the model:  0.03"
[1] "f1 score of the model:  0.2"
```

Comparision between models

```
**Comaprision of models**
```{r}
df <- rbind(score_vec_rf_seve92 , score_vec_rf_seve23 , score_vec_rf_seve21,
            score_vec_svmfit_seve , score_vec_rf_death95, score_vec_rf_death17,
            score_vec_rf_death15 , score_vec_svmfit_death)
colnames(df) <- c("Accuracy" , "Precision" , "Sensitivity" , "Specificity")
df <- as.data.frame(df)
df


```
```

Description: df [8 × 4]

	Accuracy <dbl>	Precision <dbl>	Sensitivity <dbl>	Specificity <dbl>
score_vec_rf_seve92	0.95	0.92	0.94	0.96
score_vec_rf_seve23	0.98	1.00	0.95	1.00
score_vec_rf_seve21	0.95	0.97	0.90	0.99
score_vec_svmfit_seve	0.91	0.78	0.94	0.89
score_vec_rf_death95	0.78	0.88	0.84	0.57
score_vec_rf_death17	0.78	0.84	0.86	0.55
score_vec_rf_death15	0.80	0.88	0.86	0.60
score_vec_svmfit_death	0.80	0.88	0.86	0.60

Thank You!