PSTAT 131 Final Project

Jiacong Wu

2022-11-22

Contents

Abstract

In this report, I will be using R and a stroke data set from https://www.kaggle.com/datasets/fede soriano/stroke-prediction-dataset (Fedesoriano, 2021). I did some data analysis first. And then, I fitted the data with different popular machine-learning models to see which works best. I used models including Logistic Regression, Linear Discriminant Analysis (LDA), Decision Tree, Random Forest, and Support Vector Machine(SVM). Finally, I selected the best-performing model by comparing which model generates the highest roc_auc. Random Forest was the best performing model, and it has roc_auc = 0.807 for the testing set.

Introduction

Health care has become one of the most important topic nowadays, especially after the pandemic. The stroke sounds severe, but it is actually not far from our life. "Stroke is a common disease, with one in four people affected over their lifetime, and is the second leading cause of death and third leading cause of disability in adults worldwide"(Campbell, Khatri 2020). Furthermore, stroke usually happens very in a short time. "The key clinical feature of stroke is the sudden onset of a focal neurological deficit"(Campbell, Khatri 2020). It is a fatal and sudden brain injury so the consequence can be severe. The best method to solve a sudden disaster is to nip it in the bud. How is it possible if strokes are always unexpected? Machine Learning can help us prevent it by predicting who is more likely to catch a stroke. This paper is to illustrate how a statistical machine learning model is created to predict whether a person is likely to have a stroke based on their basic health information.

"Stroke is defined as a neurological deficit at- tributed to an acute focal injury of the CNS by a vascular cause. Most strokes are ischaemic due to reduced blood flow, generally resulting from arterial occlusion"(Campbell, Khatri 2020). There are two kinds of strokes, brain ischemia, and intracranial hemorrhagic stroke. Brian ischemia is caused by blocked brain vessels, while intracranial hemorrhagic stroke is caused by blood leaking in the brain. Both will cause a lack of blood in the brain, so the brain cannot work properly. Therefore, some symptoms include "sudden unilateral weakness, numbness, or visual loss"(Hankey, 2017). Usually, in hospitals, the diagnostic includes brain CT and MRI.

Using machine learning models can save the trouble of medical examinations and can predict stroke before you have it. However, machine learning is not always correct. Even for the best model, the accuracy is still very low compared to medical examinations. Therefore, machine learning can only be used as a side tool to prepare people for the possibility of the bad news. It will never replace medical diagnostics. Predicting stroke is one of the small benefits we can take from using machine learning models.

Preparation

Loading packages

Loading Data

Data Cleaning

Since many of the predictor variables are character type, but they are actually levels. Therefore, I mutated all the categorical variable to be factors.

I also checked for NA values. I found that there are not many NA values, so I decided to get rid of those rows with NA values.

Writing the processed data into the processed_data folder.

Data Splitting

I split the data into training and testing sets with 80% of training and 20% of testing.

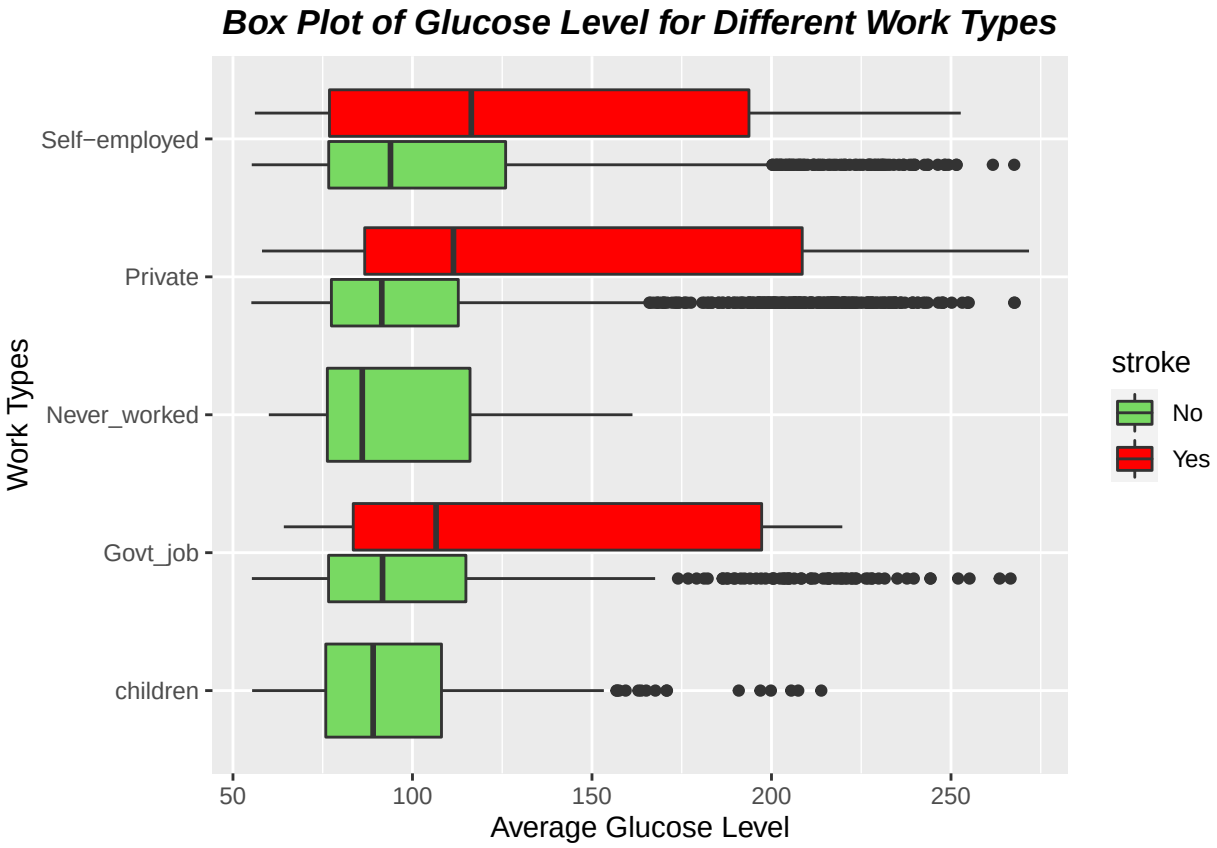
Checking the split:

```
## [1] 3926   11
```

```
## [1] 982  11
```

Exportary Data Analysis

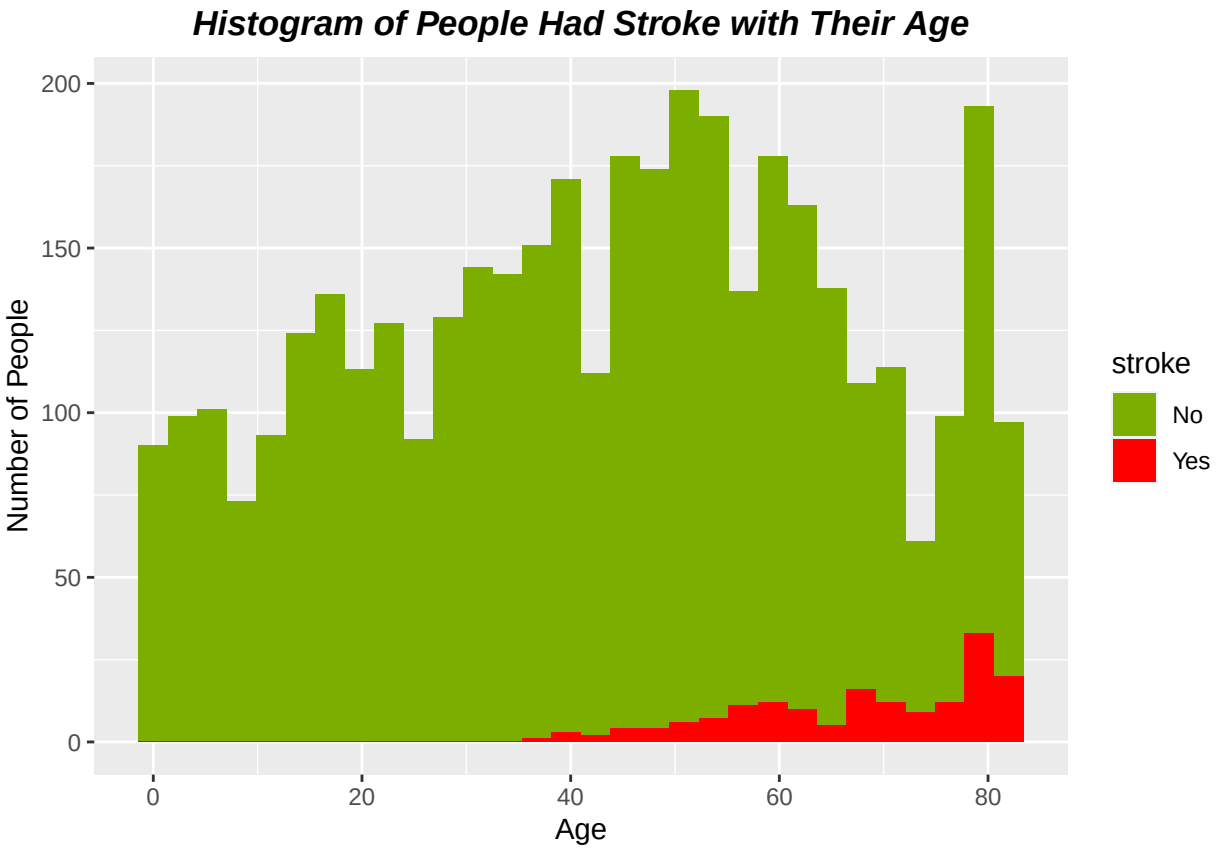
This part is about getting some insights from our data using data visualization. After we learn more about the relationships among columns, we can not only gain insight into the topic but also make some adjustments to the model to improve its performance.



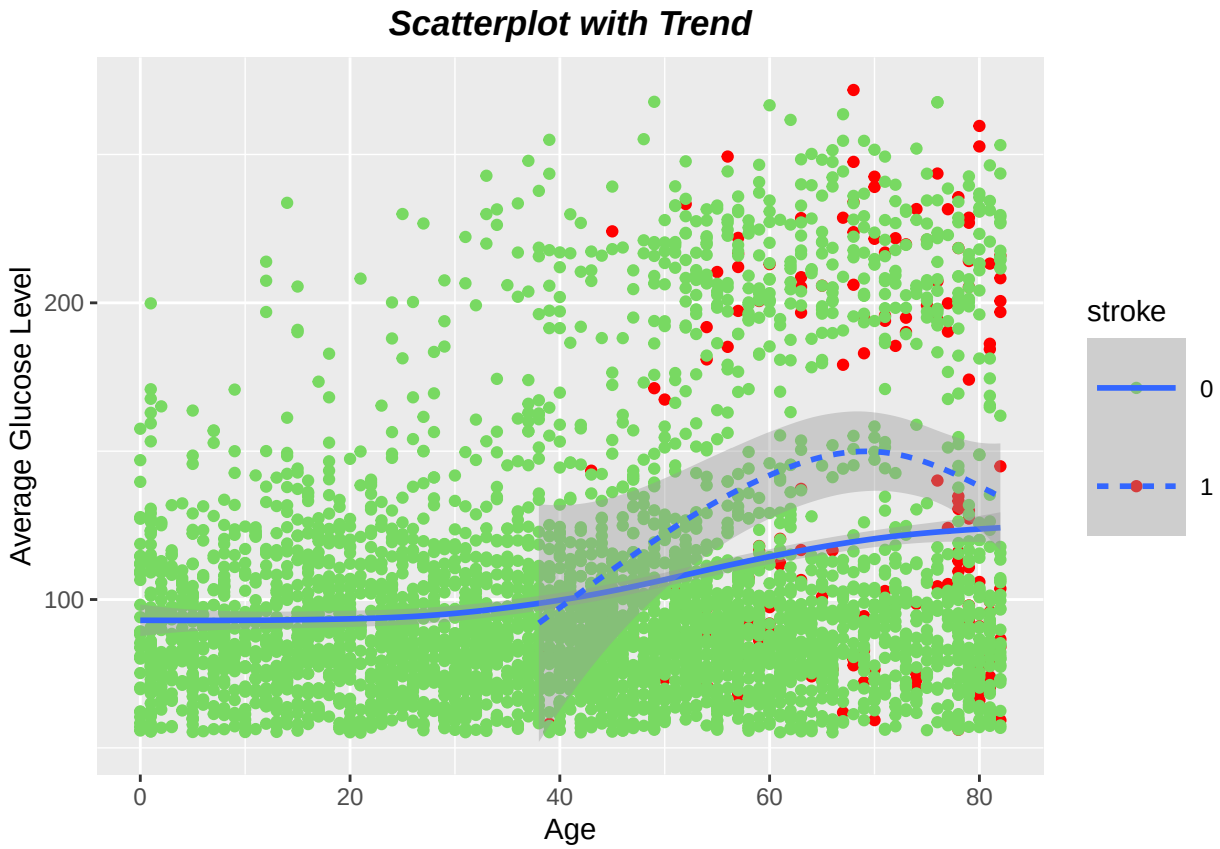
This is a box plot of the average glucose level for different work types. The box contains 50% of the data(from 25% to 75%), so it can show approximately how the majority of the average glucose levels are distributed for different work types. The lines in the boxes are the median glucose levels for each work type, showing where the middle value is. Median values are not affected by extreme values, so it is a good measure of the center. The black dots are outliers, extreme data far from the majority of data in the same category. The different color here shows whether the candidate had a stroke or not; red means they had a stroke and green means they did not. We can conclude that adult patients with relatively high average glucose levels are more likely to have a stroke.

There are two groups that are interesting in this graph. No one who is "Never worked" or "children" ever had a stroke. It is reasonable that children are not likely to have a stroke. However, does "not working" really prevents people from having a stroke? Curious about the "Never worked" group, I generated the following graph about the relationship between their work type and age.

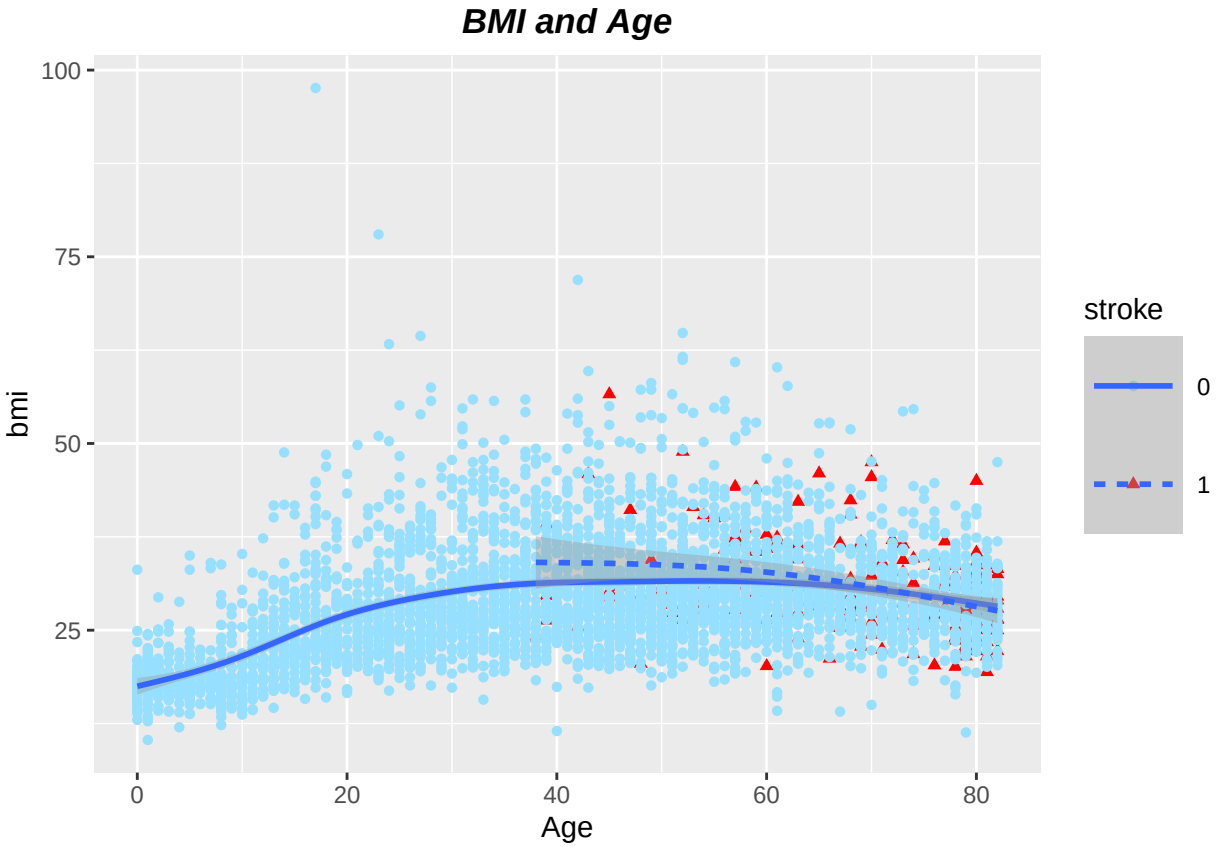
It is a violin plot. The area of each "violin" shows the distribution of each type. The bigger the area, the more people are in that group. The graph shows that people in the "never worked" group are all under 25. We cannot completely rule out the possibility that not working might help prevent stroke, but it is more likely that no one in the "never worked" group had a stroke because they are all young, and young people are not likely to have a stroke.



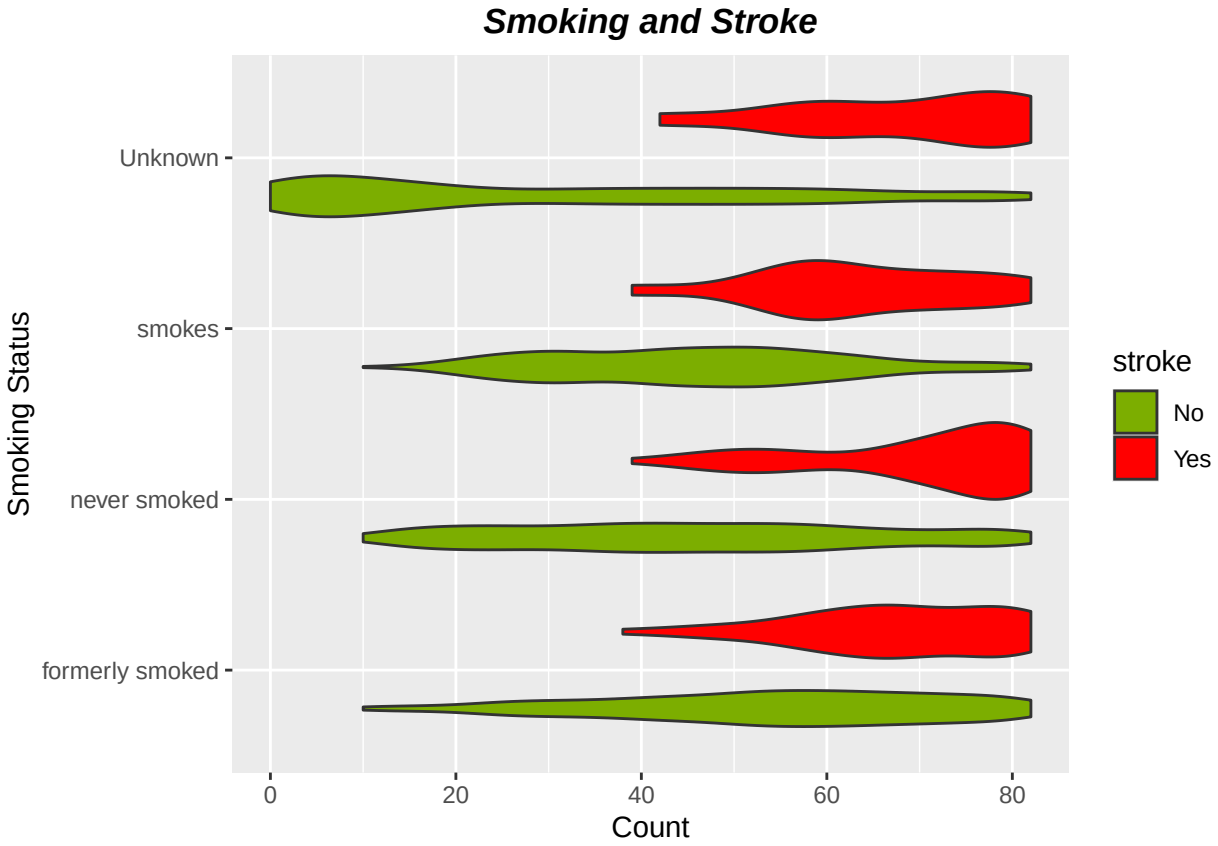
This histogram shows the age distribution, and how many of them had a stroke. The green bars show people who did not have a stroke, and the red bars represent people who had a stroke. We can notice that the green bars are reasonably normally distributed. However, the red bars are heavily left-skewed. It illustrates that elderly people are more likely to have a stroke than young people.



As the legend shows, red dots with solid lines show people who did not have a stroke, and blue dots and the dotted line represent people who had a stroke. Since the dotted line is estimated from the data, there are errors in estimations. The grey area is the error area, which means that every possible line within the grey area is an acceptable estimated line for the trend of people who have had a stroke. For people younger than 38, no one has a stroke, so the dotted line starts from the middle. After analyzing this graph, we can conclude that older adults and people with high glucose levels are the higher-risk groups.



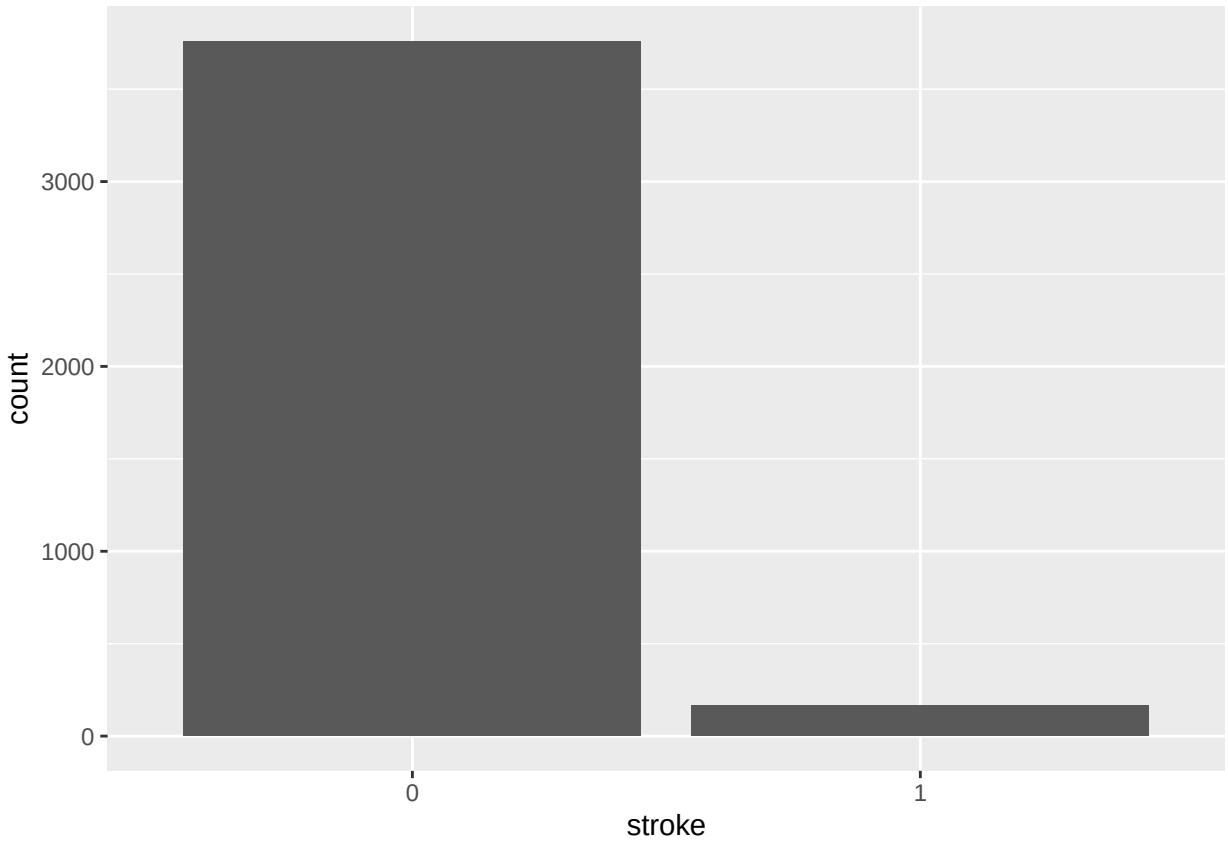
BMI, a basic body parameter, is also one of the predictor variables here. BMI stands for Body Mass Index, a measure of obesity. Unlike the last graph, people with and without strokes have completely different distributions, the majority of red triangles are hidden in the blue dots, and the regression lines are close to each other. It indicates that BMI is not a major contributing factor to the risk of stroke. However, this does not mean that BMI and the risk of stroke are unrelated. A healthy BMI is between 18.5 to 24.9. Most stroke patients are above the healthy range. Also, based on the slightly higher regression dotted line, stroke patients tend to have higher BMI. This result happened to be consistent with the research that the group of Liu did. "Both overweight and obesity increase the risk of stroke with a J-shaped dose-response relation, and the nadir of the curve was observed at BMI 23-24 kg/m2"(Liu et al., 2018).



Here is another violin graph showing the distribution of people with and without stroke history and their smoking status. In this graph, the three green "violins" and three red "violins" are relatively the same. Therefore, smoking might be bad for health but does not contribute to stroke risk. The data suggest that no matter whether the person smokes, the chance of getting a stroke is about the same. However, the result is contradictory to the conclusion from Pan's team. They think "the relationship between stroke of any type and smoking status was also statistically significant"(Pan et al., 2019). I suspect the data I used did not rule out passive when classifying people who never smoked. In Pan's peer-reviewed article, they also mentioned that "passive smoking increased the overall risk of stroke by 45%"(Pan et al., 2019).

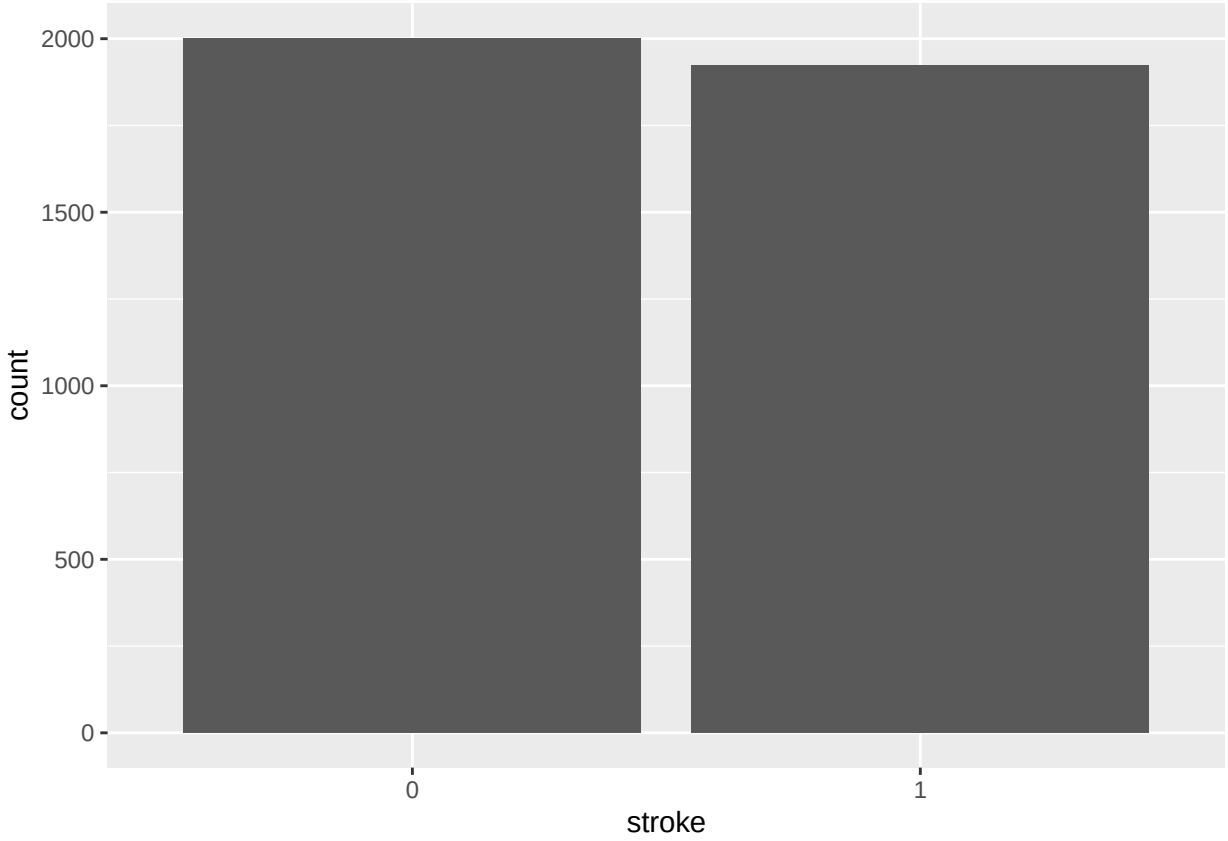
Dealing with Imbalanced Data

Since I notice the imbalanced distribution of the outcome variable stroke from the graphs plotted above, I plotted a bar chart of stroke, and it looks like this:



As the graph shows the distribution of the patients having stroke are extremely skewed. Only a very small portion has stroke. This will result in a problem that if the machine predict everything to be 0, then it can still have a very good performance and high accuracy.

Therefore, I researched online about upsampling and downsampling trying ro solve the problem. I found out a method call ROSE(Random oversampling examples) in the ROSE package, and it will apply both up and down sampling. So I chose to use it to balance the outcome inbalance.



I ploted the bar chart again, and the distribution looks fairly even now.

Cross Validation

I performed a 5-folds cross validation to the training set.

```
## #  5-fold cross-validation using stratification
## # A tibble: 5 x 2
##   splits             id
##   <list>             <chr>
## 1 <split [3140/786]> Fold1
## 2 <split [3140/786]> Fold2
## 3 <split [3141/785]> Fold3
## 4 <split [3141/785]> Fold4
## 5 <split [3142/784]> Fold5
```

Creating Recipe

Then I created a recipe. Since a lot of my predict variables are categorical data, I used step_dummy() function to all the categorical variables, and I set all the predictors to the same scale and centered at zero.

Model Selection

Logistic Regression

I first created a workflow and a logistic regression model, using the glm engine.

I then fitted the cross validation folds to the logistic regression model.

Here is the performance of the logistic regression model:

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.768     5 0.00535 Preprocessor1_Model1
## 2 roc_auc  binary     0.848     5 0.00310 Preprocessor1_Model1
```

Logistic Regression generates a roc_auc of 0.8476221 and an accuracy of 0.7684652.

LDA

Then I created a Linear Discriminant Analysis(LDA) model using MASS engine.

I fitted the model to the resamples and here is how LDA performed:

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.768     5 0.00495 Preprocessor1_Model1
## 2 roc_auc  binary     0.848     5 0.00279 Preprocessor1_Model1
```

LDA generates a roc_auc of 0.8476126 and an accuracy of 0.7679547.

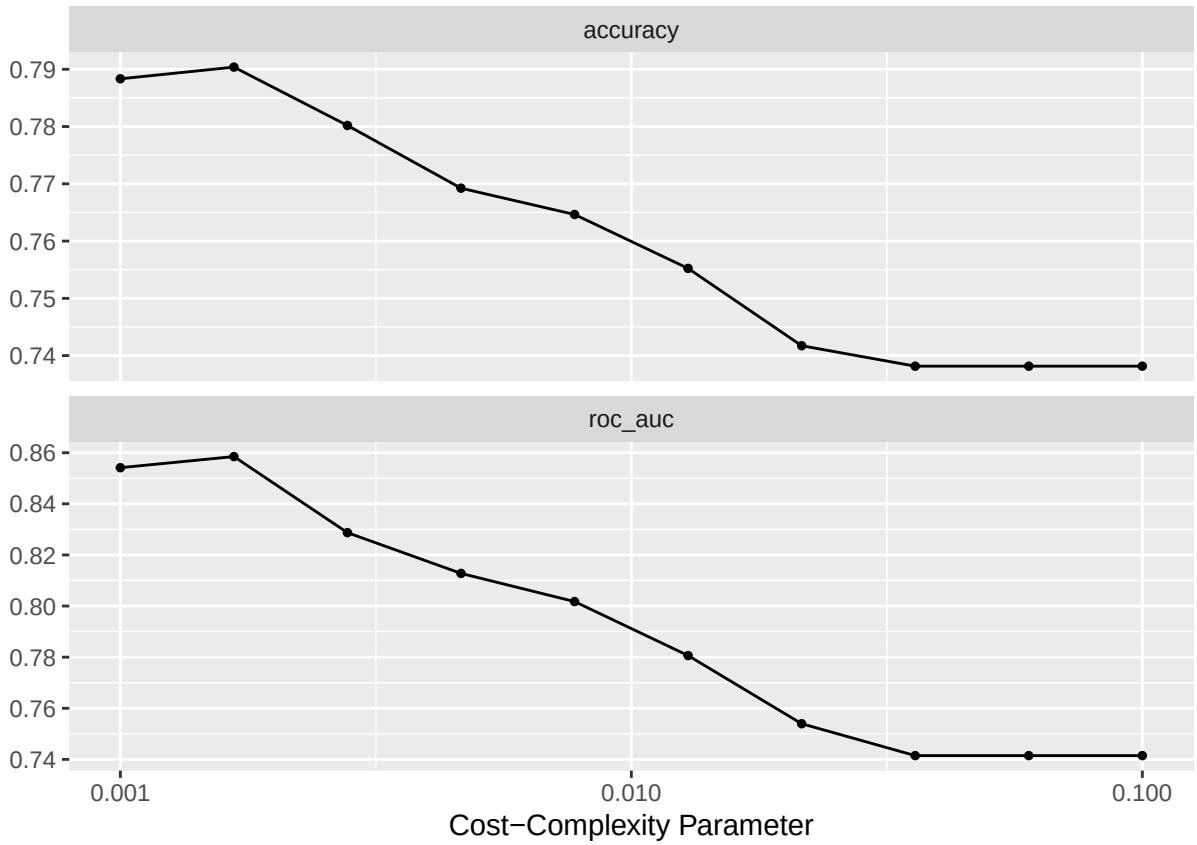
Decision Tree

Next, I tried to fit the data on a decision tree model using rpart engine.

I set the cost_complexity to be the tuning parameter, and I set a parameter grid using grid_regular() with cost_complexity to range from -3 to -1 and 10 levels .

I pruned the decision tree with the by tuning the cost_complexity using tune_grid(). Then, I save the result of the model in the rds folder.

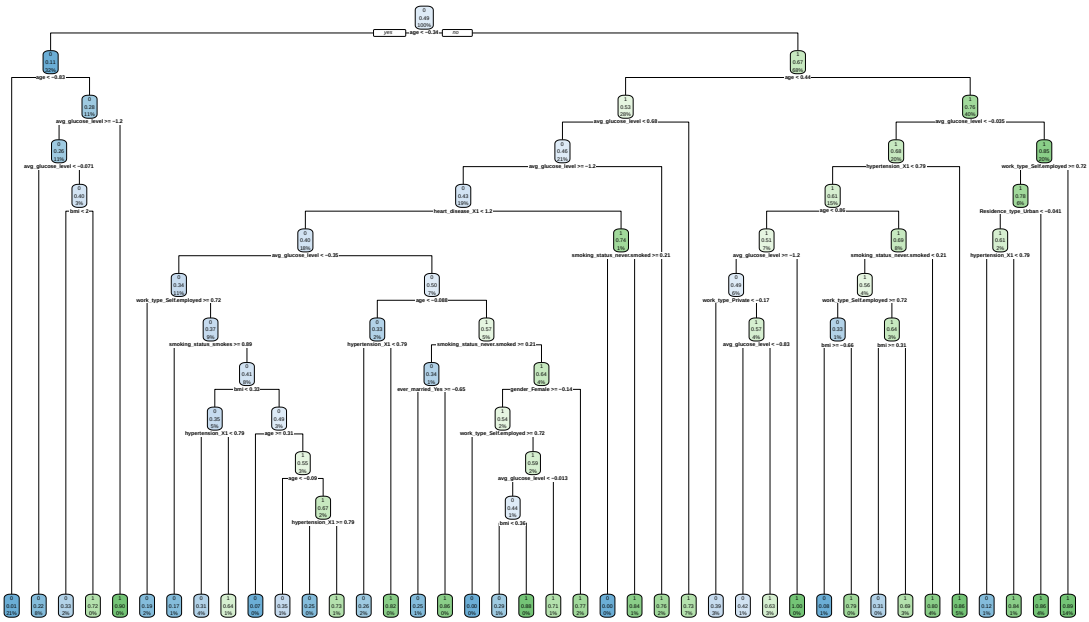
Loading the model result saved before and ploted the change of roc_auc with corresponding cost complext=ity.




```
## # A tibble: 1 x 7
##   cost_complexity .metric .estimator  mean     n std_err .config
##             <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1         0.00167 roc_auc binary     0.858     5 0.00383 Preprocessor1_Model02
```

I found out that when cost_complexity is equal to 0.001668101, the roc_auc reaches the highest point, equal to 0.8584949.

This is what the final decision tree looks like:



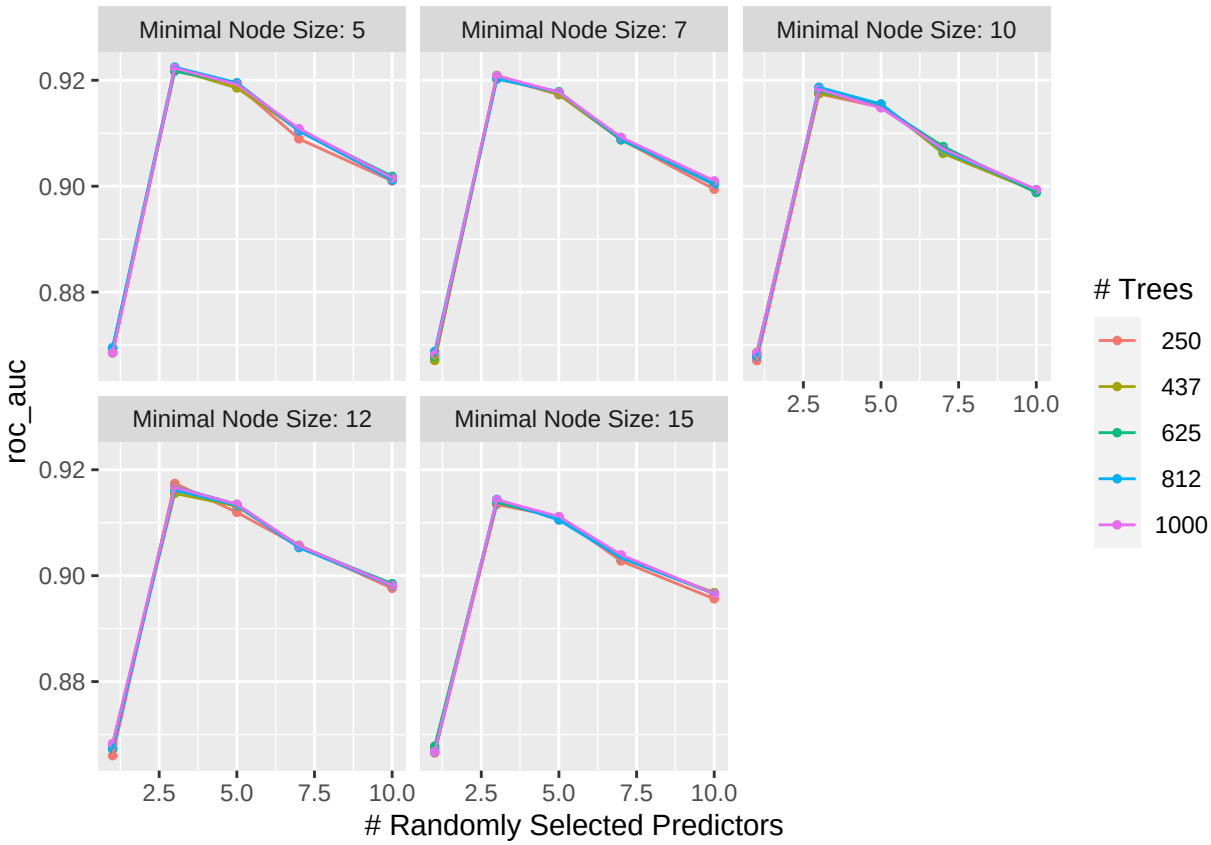
Random Forest

I also created a random forest forest model with ranger engine. Mtry, number of trees are the tuning parameter here.

I set the tune grid here. The number of mtry should be within 1 to 10 as I only have 10 parameters. It is not possible to select from more than 10 variables. I do not want there to be too few or too many trees in the random forest, so I set the number of trees to be between 250 and 1000. I set the levels to be 5 to avoid the model running for a very long time.

I execute the model by tuning and fitting. Since random forest will take very long time to run, I saved the result of the tuning and fitting in a rds file to avoid running it again.

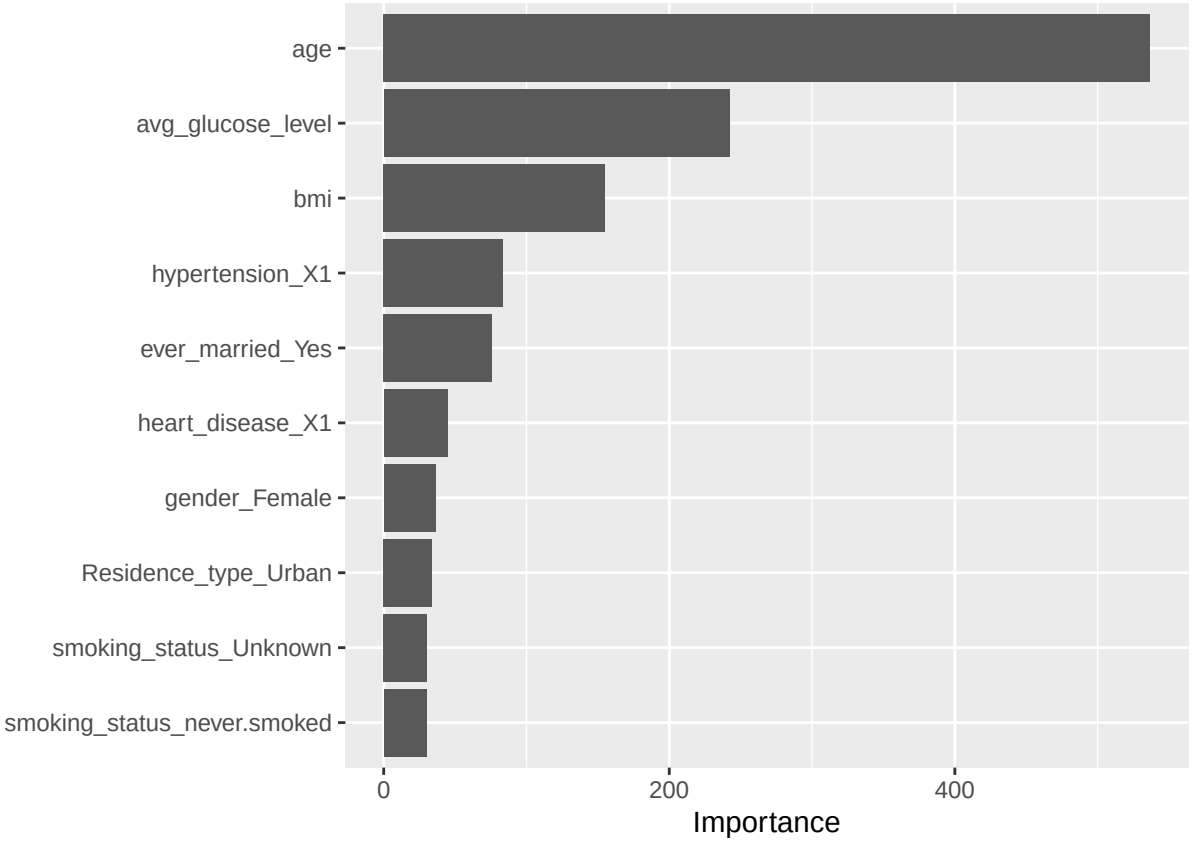
Here are the plot of roc_auc values for different minimal node sizes.



From the graphs, it does not seem like there is a big difference when using different number of trees, but using diffferent number of predictors will affect the result. It seems that when using around three precdictors, the model has the highest roc_auc. After three predictors, the roc_auc is likely to dicrease.

```
## # A tibble: 1 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     3   812     5 roc_auc binary     0.922     5 0.00368 Preprocessor1_Model0~
```

The roc_auc of the best performing model has the roc_auc of 0.9224811. It has 812 trees, and used three randomly selected variables.



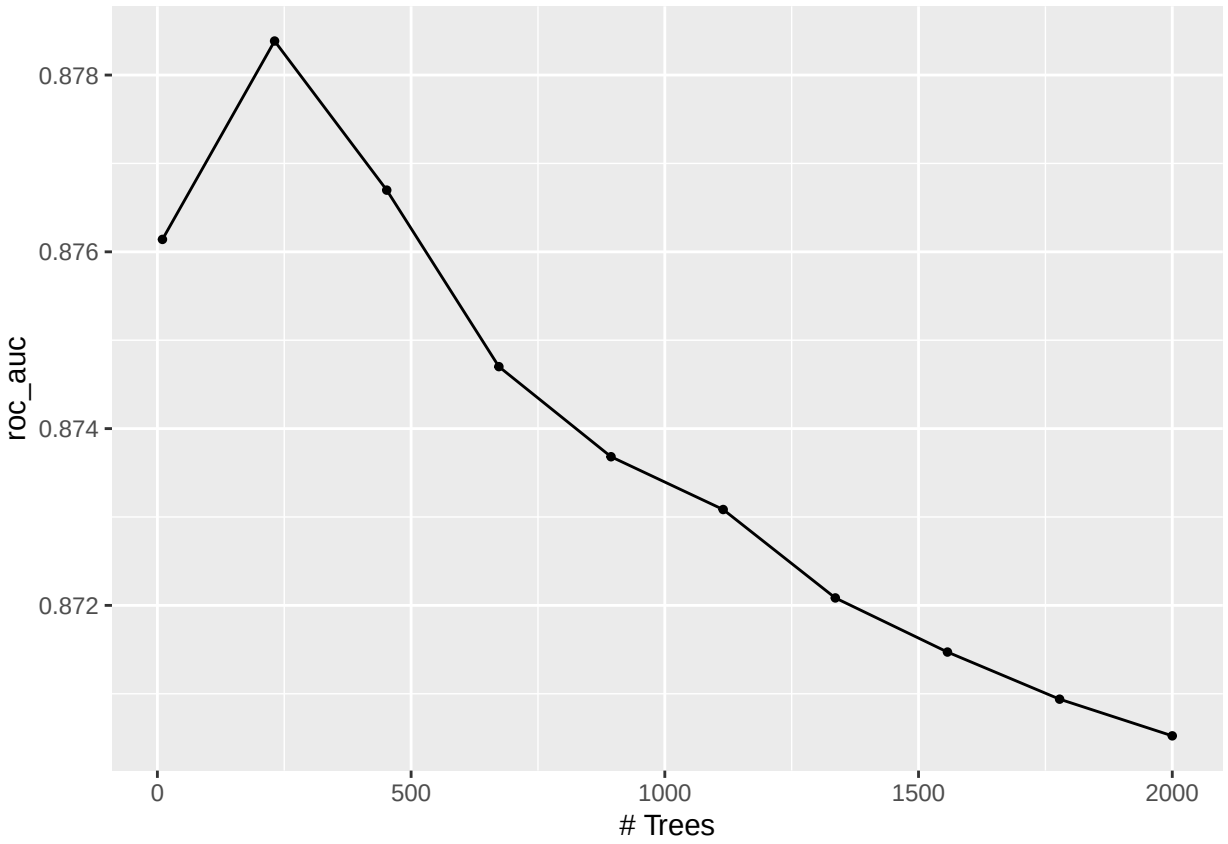
This is a vip plot, and the most important variables are age and glucose level.

Boosted Tree

I also fitted the boosted tree model using xgboost engine. I set the grid to be using number of trees from 10 to 2000 with 10 levels.

I executed the boosted tree model by tuning and fitting. I set the metric to be roc_auc because that is the metric I want to use to dicided which model is the best.

The result is saved above and loaded below to save the time repeating the time-taking tuning process.



As the graph shows, the roc_auc rises before around 250 trees. After the peak, the roc_auc decreases.

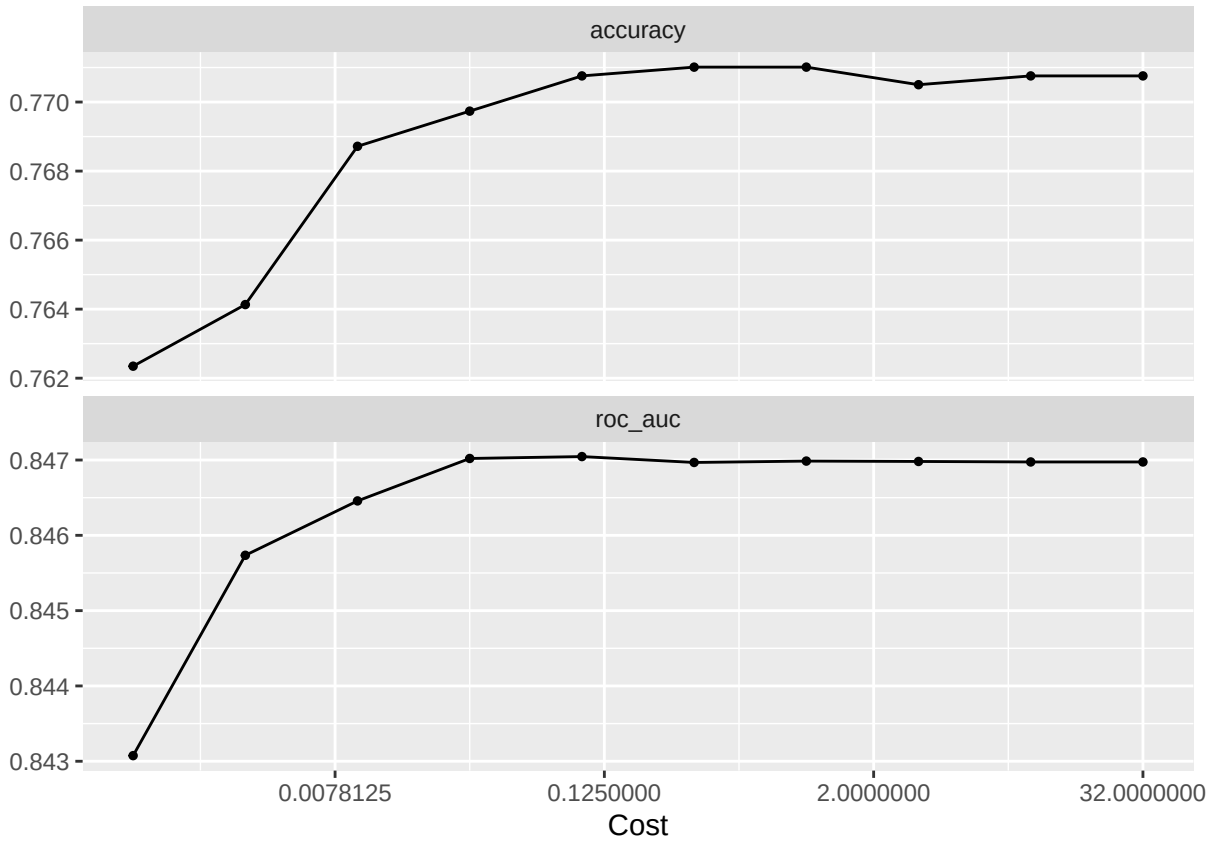
```
## # A tibble: 1 x 7
##   trees .metric .estimator  mean     n std_err .config
##   <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1   231 roc_auc binary     0.878     5 0.00496 Preprocessor1_Model02
```

The best boosted tree generates the model with roc_auc = 0.878385 with 231 trees.

Support Vector Machine

Finally, I used support vector machine to build a model using kernlab enginae and fitted to the data.

I saved the tuned result in the rds folder and loaded for plotting below.



The accuracy and the roc_auc both rises when the cost increase. After the cost passes 0.125, it tends to be the same.

```
## # A tibble: 1 x 7
##     cost .metric .estimator  mean     n std_err .config
##    <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 0.0992 roc_auc binary     0.847     5 0.00262 Preprocessor1_Model05
```

The best svm is when the cost is equal to 0.99, and it produces roc_auc = 0.8470457.

Best Model Fitting and Testing

This is a table containing all the roc_auc of the models I fitted.

```
##      Models                    Roc_auc
## [1,] "Logistic Regression"     "0.8476221"
## [2,] "LDA"                     "0.8476126"
## [3,] "Decision Tree"           "0.8584949"
## [4,] "Random Forest"           "0.9224811"
## [5,] "Boosted Tree"            "0.878385"
## [6,] "Support Vector Machine" "0.8470457"
```

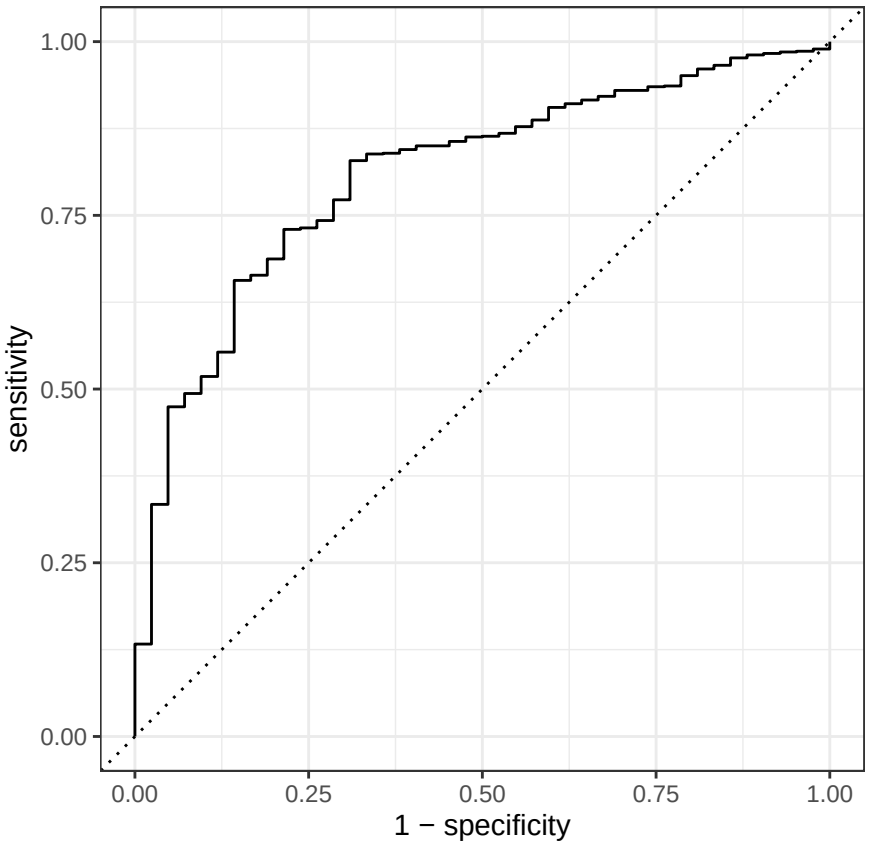
Based on the result of all the model fitting before, Random Forest has the best performance with roc_auc = 0.9224811. So I dicided to fit the Random Forest to the testing data set to see how it reacts.

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.808
```

Random Forest gives me a roc_auc of 0.8076241 for the test data set. Comparing to the roc_auc of the training set, it is 0.12 lower. Therefore, overfitting exists here, but it is not too bad. It is normal to have the training set behave better than the testing set, as the data was trained on the training set.

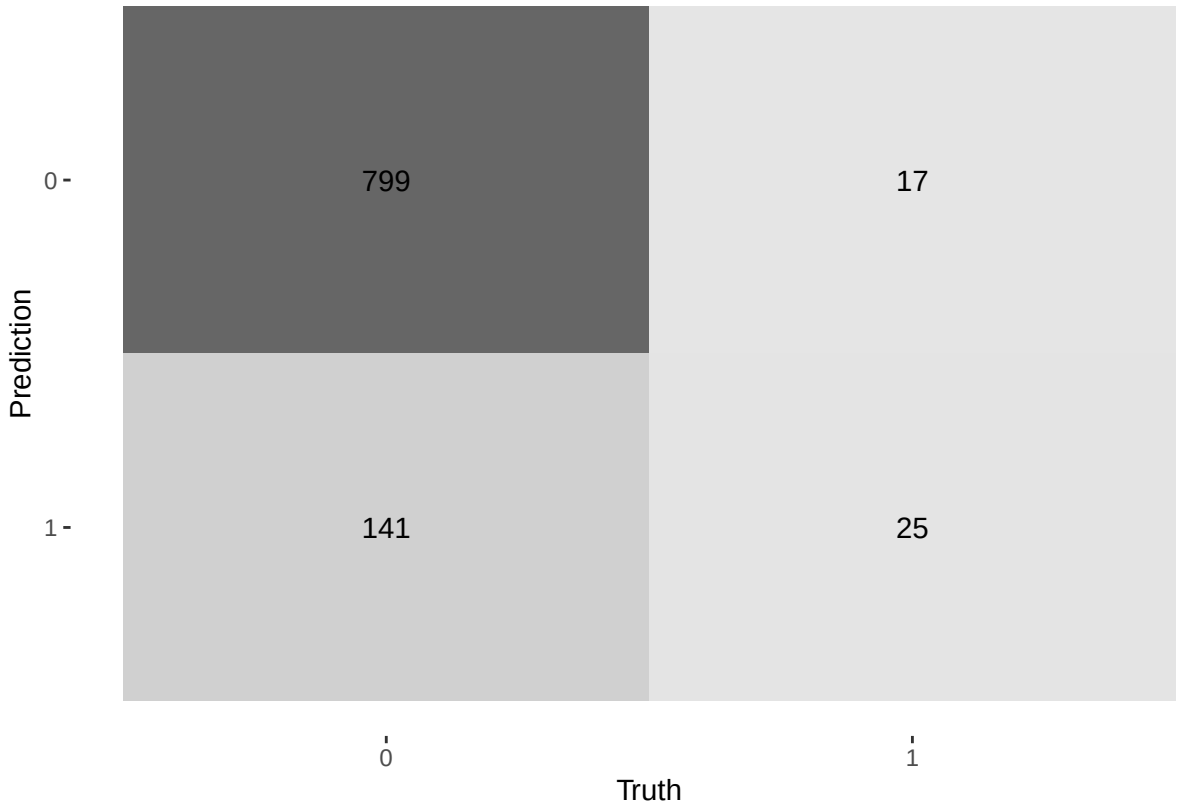
I think there is another reason might be responsible for the decrease in the roc_auc that is the up and down sampling I did. The distribution of the outcomes in the training set is different from the testing set.

Here is a ROC curve plotted:



From the ROC graph, the model is not perfect, but it is fairly good model. The ROC curve is high above the dotted line, so it predicts stroke a lot better than a coin flap.

Here is a confusion matrix heat map:



As we can see from the confusion matrix heat map, majority of the 0 are predicted correctly. However, the false positive rate is very high; a lot of negative are predicted as positive. At the same time false nagative rate is not too high. I think this is the result of the up and down sampling I did when I notice the imbalanced distribution of the outcome variable. As this is about a medical concept, stroke, I think it is fine even if the false positive rate is high. It is better to be misinterpret as positive when you do not have stroke than having a high likelihood of get a stroke without knowing it.

Conclusion

Since stroke is a common fatal disease that usually happens very fast, it is essential to predict and prevent stroke before a person really gets it. Machine Learning is one of the best modern ways to predict something. So this paper used machine learning knowledge to try to build models to predict the likelihood of a person getting a stroke. First, I cleaned the data and made it in the best form to work with. After that, I found out the imbalanced distribution of the outcome variable and applied upsampling and downsampling by using ROSE. Then, I tried to fit models, including Logistic Regression, Linear Discriminant Analysis (LDA), Decision Tree, Random Forest, and Support Vector Machine(SVM), to the training set separated from the original data set, and Random Forest worked the best with 0.92 roc_auc, so I fitted the testing data set with Random Forest model, and it generated roc_auc of 0.81. I disscussed what are the possible reasons that caused the roc_auc to decrease. I hope that one-day machine learning can really meet 95% or even 100% accuracy. In that case, a human would be afraid of stroke as we can predict and prevent it.

Reference

Lecture slides, labs, and homeworks

Campbell, B. C., Khatri, P. (2020). Stroke. The Lancet, 396(10244), 129–142. https://doi.org/10.1016/s0140-6736(20)31179-x

Fedesoriano. (2021, January 26). Stroke prediction dataset. Kaggle. Retrieved October 31, 2022, from https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

Hankey, G. J. (2017). Stroke. The Lancet, 389(10069), 641–654. https://doi.org/10.1016/s0140-6736(16)30962-x

Liu, X., Zhang, D., Liu, Y., Sun, X., Hou, Y., Wang, B., Ren, Y., Zhao, Y., Han, C., Cheng, C., Liu, F., Shi, Y., Chen, X., Liu, L., Chen, G., Hong, S., Zhang, M., & Hu, D. (2018). A J-shaped relation of BMI and stroke: Systematic review and dose–response meta-analysis of 4.43 million participants. Nutrition, Metabolism and Cardiovascular Diseases, 28(11), 1092–1099. https://doi.org/10.1016/j.numecd.2018.07.004

Pan, B., Jin, X., Jun, L., Qiu, S., Zheng, Q., & Pan, M. (2019). The relationship between smoking and stroke. Medicine, 98(12). https://doi.org/10.1097/md.0000000000014872

Appendix

```
# Loading packages
library(janitor)
library(tidymodels)
library(ISLR)
library(ISLR2)
library(tidyverse)
library(glmnet)
library(rpart.plot)
library(discrim)
library(ranger)
library(vip)
library(xgboost)
library(ggthemes)
library(neuralnet)
library(caret)
library(keras)
library(tensorflow)
library(kernlab)
library(yardstick)
tidymodels_prefer()

stroke_original = read.csv("data/unprocessed/healthcare-dataset-stroke-data.csv")

# Data cleaning
stroke_original = stroke_original%>%
  select(-id) %>%
  mutate(gender = factor(gender, levels = c("Male", "Female")),
         heart_disease = factor(heart_disease),
         work_type = factor(work_type),
         ever_married = factor(ever_married),
         smoking_status = factor(smoking_status),
         hypertension = factor(hypertension),
         Residence_type = factor(Residence_type),
         bmi = as.numeric(bmi),
         stroke = factor(stroke),
         age = as.integer(age))
stroke = na.omit(stroke_original)
#head(stroke)

#data split

write_csv(stroke, file = "data/processed/processed_data.csv")

set.seed(1000)
stroke_split = initial_split(stroke, prop = 0.8, strata = stroke)
train = training(stroke_split)
test = testing(stroke_split)

print(dim(train))
print(dim(test))

#EDA
```

```r
train %>%
  ggplot() +
  ggtitle("Box Plot of Glucose Level for Different Work Types")+
  geom_boxplot(aes(x = work_type, y = avg_glucose_level, fill = stroke)) +
  labs(x = "Work Types", y = "Average Glucose Level") +
  scale_fill_manual(values=c("#78D962", "#FF0000"), labels = c("No", "Yes"))+
  coord_flip()+
  theme(plot.title = element_text(face = "bold.italic", hjust=0.5))

train %>%
  ggplot() +
  ggtitle("Age Distribution of Different Work Types")+
  geom_violin(aes(x = work_type, y = age, fill = stroke)) +
  labs(x = "Work Types", y = "Age") +
  scale_fill_manual(values=c("#78D962", "#FF0000"), labels = c("No", "Yes"))+
  coord_flip()+
  theme(plot.title = element_text(face = "bold.italic", hjust=0.5))

train %>%
  ggplot(aes(x = age, fill = stroke)) +
  ggtitle("Histogram of People Had Stroke with Their Age")+
  geom_histogram()+
  labs(x = "Age", y = "Number of People")+
  theme(plot.title = element_text(face = "bold.italic", hjust=0.5))+
  scale_fill_manual(values=c("#7CAE00", "#FF0000"), labels = c("No", "Yes"))

train %>%
  ggplot(mapping = aes(x = age, y = avg_glucose_level)) +
  ggtitle("Scatterplot with Trend")+
  geom_point(aes(color = stroke)) +
  geom_smooth(aes(linetype = stroke), se = TRUE)+
  labs(x = "Age", y = "Average Glucose Level")+
  theme(legend.key.size = unit(1.3, 'cm'),
        plot.title = element_text(face = "bold.italic", hjust=0.5))+
  scale_color_manual(values = c("0" = "#78D962", "1" = "red"))

train %>%
  ggplot(mapping = aes(x = age, y = bmi)) +
  ggtitle("BMI and Age")+
  geom_point(aes(color = stroke, shape = stroke)) +
  scale_color_manual(values=c('#97DFFE','#FF0000'))+
  geom_smooth(aes(linetype = stroke), se = TRUE)+
  labs(x = "Age", y = "bmi")+
  theme(legend.key.size = unit(1.3, 'cm'),
        plot.title = element_text(face = "bold.italic", hjust=0.5))+
  scale_fill_discrete(labels= c("No", "Yes"))


ggplot(train, aes()) +
  geom_violin(aes(x = smoking_status, y = age, fill = stroke))+
  labs(
    title = "Smoking and Stroke",
    x = "Smoking Status",
```

```
    y = "Count"
  ) +
  coord_flip()+
  theme(plot.title = element_text(face = "bold.italic", hjust=0.5))+
  scale_fill_manual(values=c("#7CAE00", "#FF0000"), labels = c("No", "Yes"))

train %>% ggplot()+geom_bar(aes(x = stroke))


#Dealing with imblanced outcome

library(ROSE)
train<-ROSE(stroke~.,data=train)$data

train %>% ggplot()+geom_bar(aes(x = stroke))

stroke_folds <- vfold_cv(train, v = 5, strata = stroke)
stroke_folds

stroke_recipe = train %>%
  recipe(stroke ~ gender+
           age+
           hypertension+
           heart_disease+
           ever_married+
           work_type+
           Residence_type+
           avg_glucose_level+
           bmi+
           smoking_status) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

#Linear models

control <- control_resamples(save_pred = TRUE)
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(stroke_recipe)

log_fit <- fit_resamples(log_wkflow, stroke_folds)

collect_metrics(log_fit)

lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")
lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
```

```
add_recipe(stroke_recipe)

lda_tune_res <- fit_resamples(resamples = stroke_folds, lda_wkflow, control = control)

collect_metrics(lda_tune_res)

#Tree models

tree_spec <- decision_tree() %>%
  set_engine("rpart")
stroke_class_tree_spec <- tree_spec %>%
  set_mode("classification")

stroke_class_tree_wf <- workflow() %>%
  add_model(stroke_class_tree_spec %>%
              set_args(cost_complexity = tune())) %>%
  add_recipe(stroke_recipe)
param_grid_1 <- grid_regular(cost_complexity(range = c(-3, -1)), levels = 10)


tune_res <- tune_grid(
  stroke_class_tree_wf,
  resamples = stroke_folds,
  grid = param_grid_1,
  metrics = metric_set(roc_auc)
)
write_rds(tune_res, file = "rds/decision_tree_res.rds")

decision_tree = read_rds("rds/decision_tree_res.rds")
autoplot(decision_tree)

best = decision_tree%>%
  collect_metrics() %>%
  arrange(desc(mean)) %>%
  slice(1)
best

tree_final <- finalize_workflow(stroke_class_tree_wf, best)
tree_fit <- fit(tree_final,train)
tree_fit %>%
  extract_fit_engine() %>%
  rpart.plot()

forest_spec = rand_forest() %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification") %>%
  set_args(mtry = tune(),
           trees = tune(),
           min_n = tune())
forest_wf = workflow() %>%
  add_model(forest_spec) %>%
  add_recipe(stroke_recipe)
```

```
param_grid = grid_regular(mtry(range = c(1,10)),
                          min_n(range = c(5,15)),
                          trees(range = c(250,1000)),
                          levels = 5)

rf_tune_res <- tune_grid(
  forest_wf,
  resamples = stroke_folds,
  grid = param_grid,
  metrics = metric_set(roc_auc)
)
write_rds(rf_tune_res, file = "rds/rf_res_2.rds")

set.seed(1000)
rf = read_rds("rds/rf_res_2.rds")
autoplot(rf)

set.seed(1000)
best_rf = rf%>%
  collect_metrics() %>%
  arrange(desc(mean)) %>%
  slice(1)
best_rf

set.seed(1000)
rf_final = finalize_workflow(forest_wf,best_rf)
final_fit = fit(rf_final, data = train)
final_fit %>%
  extract_fit_engine() %>%
  vip()

boost = boost_tree(trees = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")
boost_wf = workflow() %>%
  add_model(boost) %>%
  add_recipe(stroke_recipe)

boost_grid = grid_regular(trees(c(10,2000)), levels = 10)

boost_tune_res = tune_grid(
  boost_wf,
  resamples = stroke_folds,
  grid = boost_grid,
  metrics = metric_set(roc_auc)
)
write_rds(boost_tune_res, file = "rds/boost_res.rds")

bst = read_rds("rds/boost_res.rds")
autoplot(bst)

best_boosting = bst%>%
  collect_metrics() %>%
```

```
  arrange(desc(mean)) %>%
  slice(1)
best_boosting

#SVM

svm_linear_spec <- svm_poly(degree = 1) %>%
  set_mode("classification") %>%
  set_engine("kernlab")
svm_linear_wf <- workflow() %>%
  add_model(svm_linear_spec %>% set_args(cost = tune())) %>%
  add_formula(stroke ~ .)

param_grid <- grid_regular(cost(), levels = 10)
svm_tune_res <- tune_grid(
  svm_linear_wf,
  resamples = stroke_folds,
  grid = param_grid
)
write_rds(svm_tune_res, file = "rds/svm_res.rds")

svm = read_rds("rds/svm_res.rds")
autoplot(svm)

svm %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  arrange(desc(mean)) %>%
  slice(1)

#Finding the best model

Models = c("Logistic Regression", "LDA", "Decision Tree", "Random Forest", "Boosted Tree", "Support Vec
Roc_auc = c(0.8476221, 0.8476126, 0.8584949, 0.9224811, 0.878385, 0.8470457 )
cbind(Models, Roc_auc)

#Fitting Random Forest to test set

set.seed(1000)
rf_res = augment(final_fit, new_data = test)
rf_roc = rf_res%>%
  roc_auc(truth = stroke, estimate = .pred_0)
rf_roc

#testing model Analysis

rf_res %>%
  roc_curve(stroke,.pred_0) %>%
  autoplot()

rf_res %>%
  conf_mat(stroke, .pred_class) %>%
  autoplot("heatmap")
```