**Topic**: Stroke Classification

**Source**: Kaggle

(<https://www.kaggle.com/fedesoriano/stroke-prediction-dataset>)

**Summary**:

For this project in IST 687, students were instructed to find a dataset of interest and follow the data science project lifecycle in creating a project from data cleaning to Model driven insights. From this direction, I had particular interest in human health, which led me to pick a dataset about people who may or may not have had a stroke and different information/demographics about those individuals. The dataset came from the popular data science website Kaggle and has 11 columns (not including ID column) and 5110 observations. I decided on this dataset because I believed it had good information and indicators for why someone may or may not experience a stroke. The end goal of this project was to be able to classify whether a given person(observation) experienced a stoke at some time in their life.

**Data Preparation:**

* First, I read the dataset into R.
  + #read in data set

strokeData <- read.csv('C:/Users/ticorder/OneDrive - Syracuse University/Semester1/ist\_687/Project/healthcare-dataset-stroke-data.csv')

* The first step in knowing how to clean the data is understanding what the data contains. Therefore, my first step was to use the View command and see the data in a tabular format.
  + View(strokeData)
* This gave a good oversight and pointed out N/A’s that needed to be handled in the dataset. The only column that contained N/A’s was the BMI field which was a numeric attribute. At first, I thought the best plan of action was to remove all records that contained N/A’s by using the N/A omit function within R. However, after more gaining experience in R, I switched my method of handling N/A’s by replacing the missing values with the mean of the BMI attribute. This way we could keep all the records and not exclude any data when it was not necessary.
  + strokeData$bmi[is.na(strokeData$bmi)]<-mean(strokeData$bmi,na.rm=TRUE)
* After removing N/A’s, another data cleaning step that needed to be taken was removing the ID attribute. The ID attribute was not going to have any use in providing insights or driving analysis and could therefore be removed from the dataset without harm.
  + strokeData <- strokeData[, -1]
* The dataset was collected in a fairly “clean” state, so there was not a whole lot of work to be done in this step of the data science life cycle. My final task before moving on to Exploratory data analysis was to look at broad overviews of the data and look for any anomalies or any other opportunities for cleaning. All was well and I was ready to move on.
  + #high level outline of dataset

#structure

str(strokeData)

#summary

summary(strokeData)

# first 6

head(strokeData)

# last 6

tail(strokeData)

# tabular view

View(strokeData)

**Exploratory Data Analysis:**

* In the exploratory analysis phase, I started to get into some descriptive statistics about the data. I decided to create a function that prints out common descriptive statistics for a numeric vector to allow for some code reuse.
  + desc\_stats <- function(vec){

cat("Mean: ", mean(vec), "\n")

cat("Min: ", min(vec), "\n")

cat("Max: ", max(vec), "\n")

cat("Standard Dev: ",sd(strokeData$avg\_glucose\_level), "\n")

cat("Median: ", median(strokeData$avg\_glucose\_level), "\n")

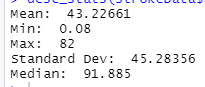
}

* Using this function, I called all numeric attributes in the stroke dataset to generate descriptive statistics.
  + # age attribute

strokeData$age <- as.numeric(strokeData$age)

desc\_stats(strokeData$age)

Result:



# gulcose attribute

strokeData$avg\_glucose\_level <- as.numeric(strokeData$avg\_glucose\_level)

desc\_stats(strokeData$avg\_glucose\_level)

Result:

Text

Description automatically generated

# bmi attribute

strokeData$bmi <- as.numeric(strokeData$bmi)

desc\_stats(strokeData$bmi)

Result:

Letter

Description automatically generated with low confidence

* As for the non-numeric attributes, I found unique values for the attribute and the found how many of each value were present in the dataset. This allowed me to see if certain attributes were heavily distributed to one value.
  + #exploring non numeric attributes

# how many of each for attributes

#gender

length(strokeData$gender[strokeData$gender == 'Male'])

length(strokeData$gender[strokeData$gender == 'Female'])

# fairly even amount of each:

Graphical user interface

Description automatically generated with low confidence

#hypertension

length(strokeData$hypertension[strokeData$hypertension == 0])

length(strokeData$hypertension[strokeData$hypertension == 1])

# many more people without hypertensions comapred to positive hypertension indicator:

Graphical user interface, text

Description automatically generated with medium confidence

#married

length(strokeData$ever\_married[strokeData$ever\_married == 'Yes'])

length(strokeData$ever\_married[strokeData$ever\_married == "No"])

# many more people have been married comapred to those who have not

Text

Description automatically generated with low confidence

#work type

unique(strokeData$work\_type)

length(strokeData$work\_type[strokeData$work\_type == 'Private'])

length(strokeData$work\_type[strokeData$work\_type == "Self-

employed"])

length(strokeData$work\_type[strokeData$work\_type == 'Govt\_job'])

length(strokeData$work\_type[strokeData$work\_type == "children"])

length(strokeData$work\_type[strokeData$work\_type ==

"Never\_worked"])

# majority of people work in the private sector

Text

Description automatically generated

#smoker

unique(strokeData$smoking\_status)

length(strokeData$smoking\_status[strokeData$smoking\_status ==

'formerly smoked'])

length(strokeData$smoking\_status[strokeData$smoking\_status == "never

smoked"])

length(strokeData$smoking\_status[strokeData$smoking\_status ==

'smokes'])

length(strokeData$smoking\_status[strokeData$smoking\_status ==

"Unknown"])

# results are mixed

Text

Description automatically generated

#Residence

unique(strokeData$Residence\_type)

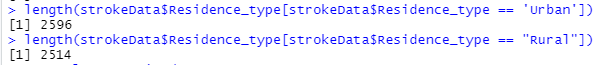
length(strokeData$Residence\_type[strokeData$Residence\_type ==

'Urban'])

length(strokeData$Residence\_type[strokeData$Residence\_type ==

"Rural"])

# results are mixed



#prediction attribute: stroke indicator

length(strokeData$stroke[strokeData$stroke == 0])

length(strokeData$stroke[strokeData$stroke == 1])

# majority of people in dataset did NOT have a stroke

Graphical user interface

Description automatically generated with low confidence

* After these initial descriptive statistics, I did some sampling on the numerical attributes and checked if their distributions matched a bell curve, which all mostly did.
  + # creating function for code reuse

sampling <- function(vec){

sampleMeans <- replicate(1000,mean(sample(vec, 50, replace=TRUE)),

simplify = TRUE)

cat("Mean: ", mean(sampleMeans))

hist(sampleMeans)

}

#age

sampling(strokeData$age)

# has a fairly normal disribution with 50 samples taken 1000 times

Chart, histogram

Description automatically generated

#bmi

sampling(strokeData$bmi)

# has a fairly normal disribution with 50 samples taken 1000 times

Chart, histogram

Description automatically generated

#glucose

sampling(strokeData$avg\_glucose\_level)

# has a fairly normal disribution with 50 samples taken 1000 timesChart, histogram

Description automatically generated

* My last step in the exploratory data analysis phase was to create some visualizations to describe, summarize, and display the data.
  + The first visualization I created was a box plot that displayed ages of people with consideration of their respective genders:

# box plot to compare ages across genders

ggplot(strokeData, aes(x=gender, y=age)) + geom\_boxplot(aes(fill=factor(gender)))

# majority of ages between genders is between 25 and 60

Chart, box and whisker chart

Description automatically generated

* + The next visualization is a scatterplot to compare a persons average glucose level to their age, while also taking into account if they have hypertension.

# scatter plot to compare glucose level to age, as well as considering hypertension

ggplot(strokeData, aes(x=avg\_glucose\_level, y=age))+ geom\_point(aes(color=hypertension))

# the older someone is the more likely they are to have high glucose

# also the older someone is the more likely they can have hypertensionChart, scatter chart

Description automatically generated

* + This visualization is a bar chart to display smoking status frequencies as well as using colors to display stroke distributions.

# bar chart to see smoking status' along with stroke indicator

ggplot(strokeData, aes(x=smoking\_status)) + geom\_bar(aes(fill=factor(stroke)))

# majority of dataset did not have stroke, however distributions of strokes across smoking status is fairly even

Chart, bar chart

Description automatically generated

* + Finally, the last visualization is a bar chart as well to see married status frequencies as well as using colors to show stroke distributions.

# bar chart to see marital status along with stroke indicator

ggplot(strokeData, aes(x=ever\_married)) + geom\_bar(aes(fill=factor(stroke)))

# even though much of data set was married before, there is still strong indication that those who have been married are more likely to have a stroke

Chart, bar chart

Description automatically generated

* The visualizations gave some great insight on the data and the correlations between attributes. However, they have not answered all my questions, which leads me to the next step of the data science phase which is model development.

**Model Development:**

* The next step in the data science project lifecycle is building models to make predictions about your data. This is done by splitting the data into training and test data so the predictions are not overfitting to the data, and we can test the model on a new population of records. There are many different algorithms out there to build a model such as linear modeling, associative rules mining, support vector machines, naïve bayes, and many others. For my project, I wanted to classify a label as the end result, which is why I decided to use Support Vector Machines(SVM/KSVM) and Naïve Bayes as the main algorithms to build the models. SVM’s map the data into multi-dimensional space to see where the data point lay relative to one another and group the data points and make predictions by seeing which group the observation falls into. Naïve Bayes, on the other hand, assumes all attributes are independent of each other, which is why it is called Naive. Naïve Bayes uses probabilities to calculate and classify a given observation.
* The first model uses the KSVM algorithm, predicts the stroke indicator attribute, and uses all other available attributes to make the prediction.
  + Before building the models I had to change some modes of the attributes, as well as build the training and test sets.
    - # set non-numerics to factors so KSVM will work

strokeData$gender <- as.factor(strokeData$gender)

strokeData$ever\_married <- as.factor(strokeData$ever\_married)

strokeData$work\_type <- as.factor(strokeData$work\_type)

strokeData$Residence\_type <- as.factor(strokeData$Residence\_type)

strokeData$smoking\_status <- as.factor(strokeData$smoking\_status)

strokeData$stroke <- as.factor(strokeData$stroke)

str(strokeData)

* + - # 1st Model: using Support Vecotor Machines (ksvm)

nrows <- nrow(strokeData)

#Q: What should our cutpoint be for 5110 observations (2/3)?

#A : Estimaed: Between ~ 3,372-3,423 should be cutpoint

cutpoint <- floor(nrows/3\*2)

cutpoint

rand <- sample(1:nrows)

head(rand)

# creating training and test data

stroke.train <- strokeData[rand[1:cutpoint],]

stroke.test <- strokeData[rand[(cutpoint+1):nrows],]

str(stroke.train)

str(stroke.test)

* + After putting the data in the necessary modes and creating the training and test data sets, I was ready to move on to the model. However, after digging into the results of the model, there was a problem. The model was always predicting no stroke. Since so many of the observations were people that did not have a stroke, the algorithm still was able to do very well by always predicting no stroke. This does not give me a very accurate model and does not tell me the indications of why someone might have a stroke.
    - # create model and train with training data

model <- ksvm(stroke~.,data=stroke.train)

model

#use model to predict and analyze accuracy

pred <- predict(model, stroke.test)

results <- table(pred,stroke.test$stroke)

#finding percentage classified correctly from test data

totalCorrect <- results[1,1] + results[2,2]

totalInTest <- nrow(stroke.test)

totalCorrect/totalInTest

# ~95%

A picture containing text

Description automatically generated

# Not applicable, always predicting No stroke

* + After looking at the results from the previous model, I decided to build another model with KSVM predicting the stroke indicator with only one predicting attribute. I did this in hopes the model would predict both stroke and no stroke labels unlike the previous model with all attributes as predictors. The attribute I used for predicting was BMI. However, the model produced the same results.
    - # Using same model structure with single column as predictor

model <- ksvm(stroke~bmi,data=stroke.train)

model

#use model to predict and analyze accuracy

pred <- predict(model, stroke.test)

results <- table(pred,stroke.test$stroke)

#finding percentage classified correctly from test data

totalCorrect <- results[1,1] + results[2,2]

totalInTest <- nrow(stroke.test)

totalCorrect/totalInTest

#~95%

# Not applicable, always predicting No stroke

A picture containing text

Description automatically generated

* + #KSVM insights

# - Whether I use all columns to predict or a single column, ksvm will always predict no stroke. This is because a majority of the dataset has not had a stroke before so the algorithm can achieve a high accuracy rate with always predicting no stroke. However, there is a key difference in the number of support vector machines as the model with a single predictor only uses ~334 SVM's, whereas the model with all predictors uses ~602 SVM's. To get more accurate results, I am going to use a dataset of 500 records, 249 who have had a stoke, and 251 who have not. This will give us a more accurate model for predicting those who have had a stroke.

limited\_strokeData <- strokeData[0:500, ]

nrows <- nrow(limited\_strokeData)

cutpoint <- floor(nrows/3\*2)

cutpoint

rand <- sample(1:nrows)

head(rand)

# creating training and test data

limited\_stroke.train <- limited\_strokeData[rand[1:cutpoint],]

limited\_stroke.test <- limited\_strokeData[rand[(cutpoint+1):nrows],]

str(limited\_stroke.train)

str(limited\_stroke.test)

# create model and train with training data

model <- ksvm(stroke~.,data=limited\_stroke.train)

model

#use model to predict and analyze accuracy

pred <- predict(model, limited\_stroke.test)

results <- table(pred,limited\_stroke.test$stroke)

#finding percentage classified correctly from test data

totalCorrect <- results[1,1] + results[2,2]

totalInTest <- nrow(limited\_stroke.test)

L\_KSVM <- totalCorrect/totalInTest

L\_KSVM

# ~75%

# predictions are now mixed and we have a more accurate model because KSVM is not always predicting no stroke due to majority of original dataset not having a stroke.

* + The next model I created used the limited dataset again, however this time I am using one attribute, the age attribute, to make predictions to try and get a more parsimonious model.

#try model with single predictor: AGE

model <- ksvm(stroke~age,data=limited\_stroke.train)

model

#use model to predict and analyze accuracy

pred <- predict(model, limited\_stroke.test)

results <- table(pred,limited\_stroke.test$stroke)

#finding percentage classified correctly from test data

totalCorrect <- results[1,1] + results[2,2]

totalInTest <- nrow(limited\_stroke.test)

L\_age\_KSVM <- totalCorrect/totalInTest

L\_age\_KSVM

# ~73%

# We can get about the same accuracy with only using age as a predictor compared to using all columns as a predictor. The SVM's are slightly lower with only age as a predictor as well.

* The second set of models uses the Naïve Bayes algorithm which was explained earlier in the document. The first model uses all attributes as predictors.
  + #compute model in Naive Bayes using e1071 package

mod\_NB <- naiveBayes(stroke ~ ., data = stroke.train)

mod\_NB\_pred <- predict(mod\_NB, stroke.test)

mod\_NB\_results <- table(mod\_NB\_pred, stroke.test$stroke)

# calculate correct and plot for Naive Bayes

totalCorrectNB <- mod\_NB\_results[1,1] + mod\_NB\_results[2,2]

totalInTestNB <- nrow(stroke.test)

NB <- totalCorrectNB/totalInTestNB

NB

# ~85%

* The last model I created used the Naïve Bayes algorithm again but with only the average glucose level as a predictor for if someone has had a stroke.
  + #compute model in Naive Bayes using single predictor

mod\_NB <- naiveBayes(stroke ~ avg\_glucose\_level, data = stroke.train)

mod\_NB\_pred <- predict(mod\_NB, stroke.test)

mod\_NB\_results <- table(mod\_NB\_pred, stroke.test$stroke)

# calculate correct and plot for Naive Bayes.

totalCorrectNB <- mod\_NB\_results[1,1] + mod\_NB\_results[2,2]

totalInTestNB <- nrow(stroke.test)

gluc\_NB <- totalCorrectNB/totalInTestNB

gluc\_NB

# ~93%

# Naive Bayes insights

# - Naive Bayes is not always predicting no stroke with the full dataset.

# However, for the model with a single predictor, the majority attributes resulted in a model that always predicts no stroke. The only columns that did not generate model which always predict no stroke were heart\_disease and avg\_glucose\_level, and avg\_glucose\_level was 3% more accurate.

**Interpretation of Results:**

* To look at the model’s side by side and compare the accuracies, I created a visualization to compare the 4 models (excluding first two models which always predicted no stroke).
  + vec <- c(L\_KSVM, L\_age\_KSVM, NB, gluc\_NB)

name <- c("Limited KSVML", "Limited age KSVM", "Naive Bayes", "AVG Glucose Naive Bayes")

accuracies <- data.frame(name, vec)

g <- ggplot(accuracies) + geom\_col(aes(x=name, y=vec\*100, fill=factor(name)))

g + theme(axis.text.x = element\_text(angle = 45, hjust=1))

Chart, bar chart

Description automatically generated

* **Conclusion:**
  + The most thorough, parsimonious, and most accurate model is the Naive Bayes model with only the average glucose column as the predictor. With this model we can limit the number of predictors as well as achieve a high accuracy rate in predicting. Also, we can use the entire dataset because Naive Bayes did not limit the predictions to no stroke only even though the majority of the dataset did not a have stroke. KSVM did do this with the full dataset which is why I had to create a limited dataset to even the distribution of stroke to non-stroke. Not only were the KSVM accuracies lower, but the models were less accurate and less trustworthy.
  + I also believed I accomplished the original goals I set out to achieve in this project. A person is more likely to have a stroke if they are either older in age or if they have other medical conditions that may lead to a stroke such as high glucose levels.
  + Original Business Questions:
    - Did this person have a stroke based on the details provided? (Prediction)
      * Was able to classify using classification machine learning algorithms.
    - What details were used to predict whether a person had a stroke?
      * Age, high glucose, BMI, hypertension, and others.
    - What are common factors among persons who have had a stroke?
      * High glucose and older in age.
    - What are common factors among persons who have NOT had a stroke?
      * Younger and no other influencing medical conditions.
    - What is the average age across those who have had a stroke?
      * mean(strokeData$age[strokeData$stroke==1])
      * 