Project Report: Analysis of Stroke Data

Elias Joseph

June 21, 2021

Problem Statement

I will be analyzing data on strokes to see if I can use the features provided to predict which patients are at risk of getting a stroke. These features are gender, age, hypertension, heart disease, married, work type, residence type, glucose level, bmi, and smoking status.

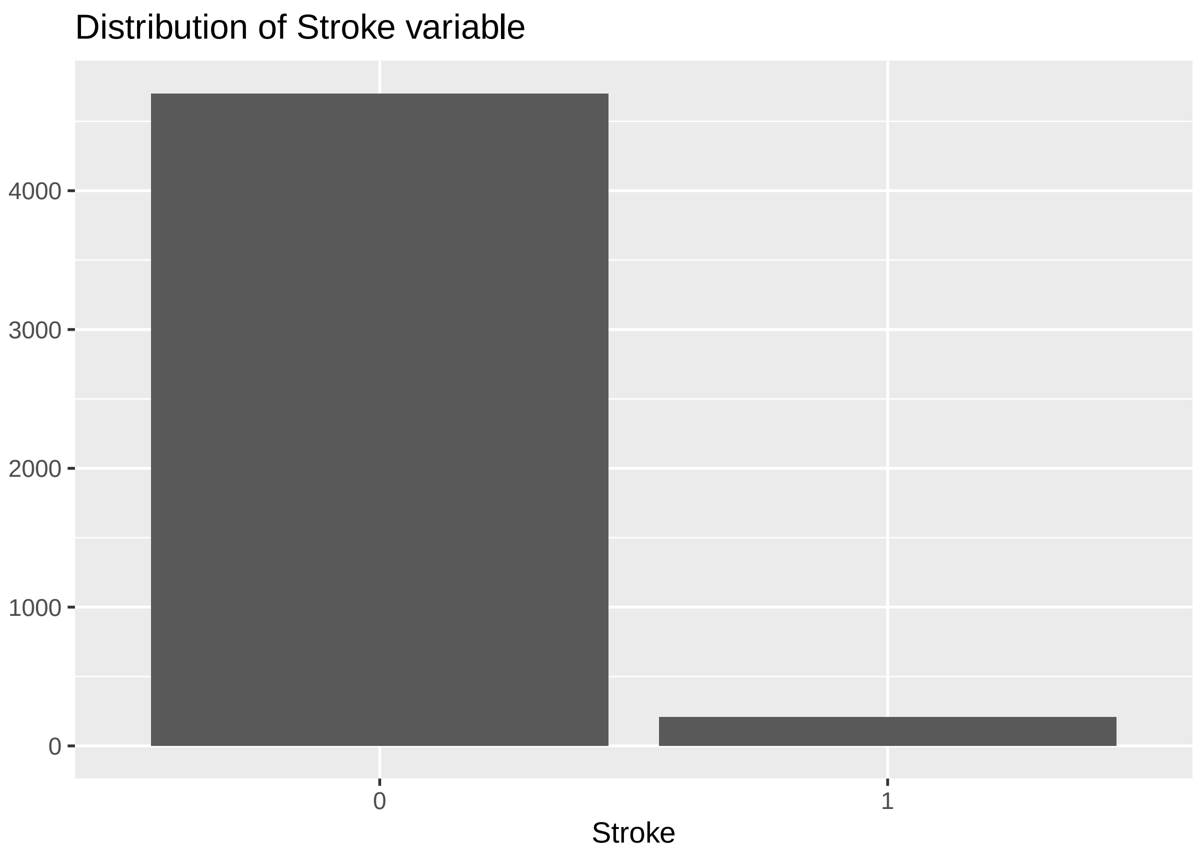
Some of the code has been included, however all the code will be accessable in the form of an attatched R markdown file and jupyter notebook file.

Data Summary

- ID: a unique identifier. This doesn't provide any useful information when it comes to looking at trends, so it won't be included in further analysis.
- Gender: he gender of the patient: Male, Female, or other.
- Age: The patient's age.
- Hypertension: A boolean value indicating if the patient has hypertension.
- Heart Disease: A boolean value indicating if the patient has heart disease.
- Ever Married: A boolean value indicating if the patient was ever married.
- Work Type: The patients employment type. Factors are 'children', 'Govt Job', 'Never Worked', 'Private', and 'Self Employed'.
- Residence Type: A 2 level factor indicating if the patient lives in an urban or rural environment.
- Average Glucose Level: The average glucose level in the patients blood.
- BMI: The patient's body mass index.
- Smoking Status: Details if the patient ever smoked. factors include 'formerly smoked', 'never smoked', 'smokes', and 'unknown'.
- Stroke: A boolean value indicating whether the patient suffered a stroke.

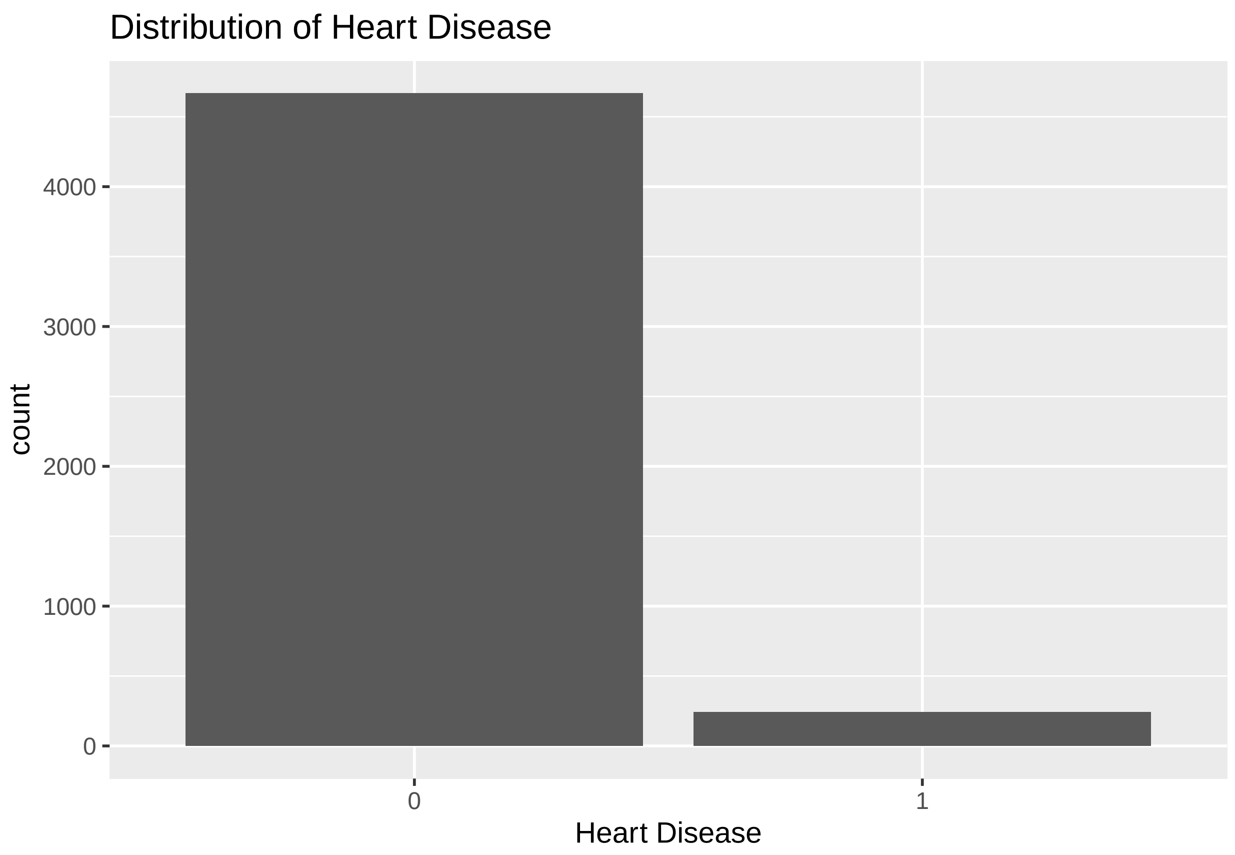
EDA

```r
ggplot(stroke.df, aes(x = stroke)) +
  geom_bar() +
  labs(title = 'Distribution of Stroke variable', x = 'Stroke', y = '')
```

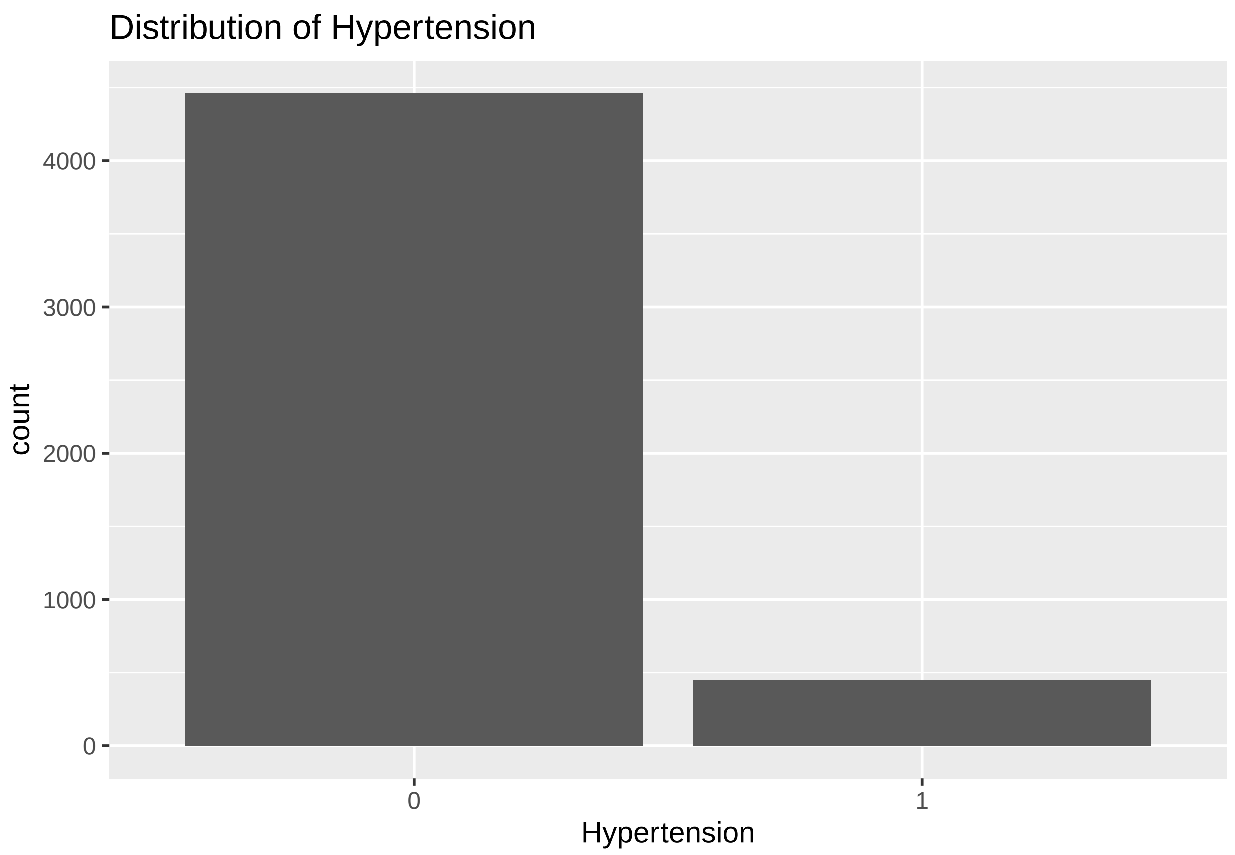


The most difficult thing about this data set is the fact that about 95% of the cases are negative. Therefore, if accuracy is used as a metric to find a model, the models will most likely classify everything as negative, as that would have a 95% accuracy.

```
ggplot(stroke.df, aes(x = heart_disease)) +
  geom_bar() +
  labs(title = 'Distribution of Heart Disease', x = 'Heart Disease')
```

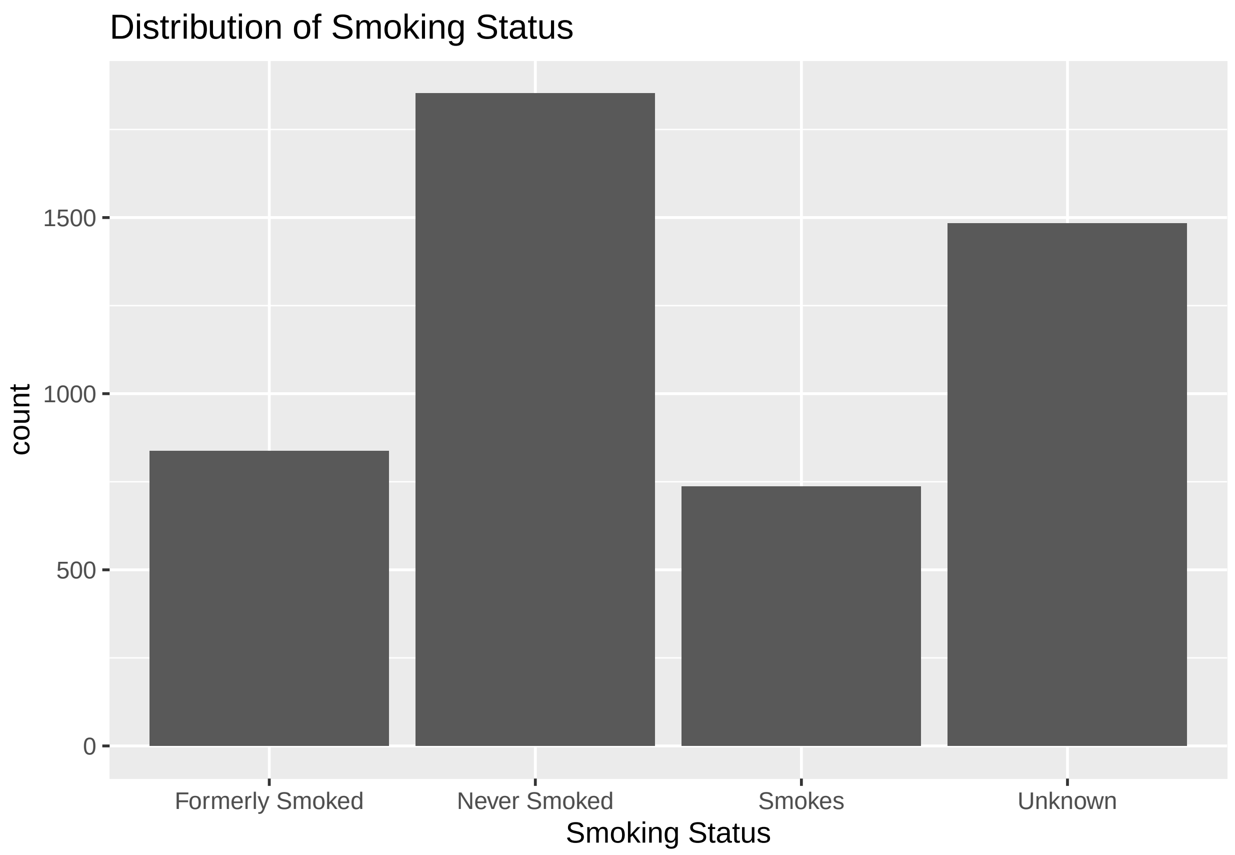


```
ggplot(stroke.df, aes(x = hypertension)) +
  geom_bar() +
  labs(title = 'Distribution of Hypertension', x = 'Hypertension')
```

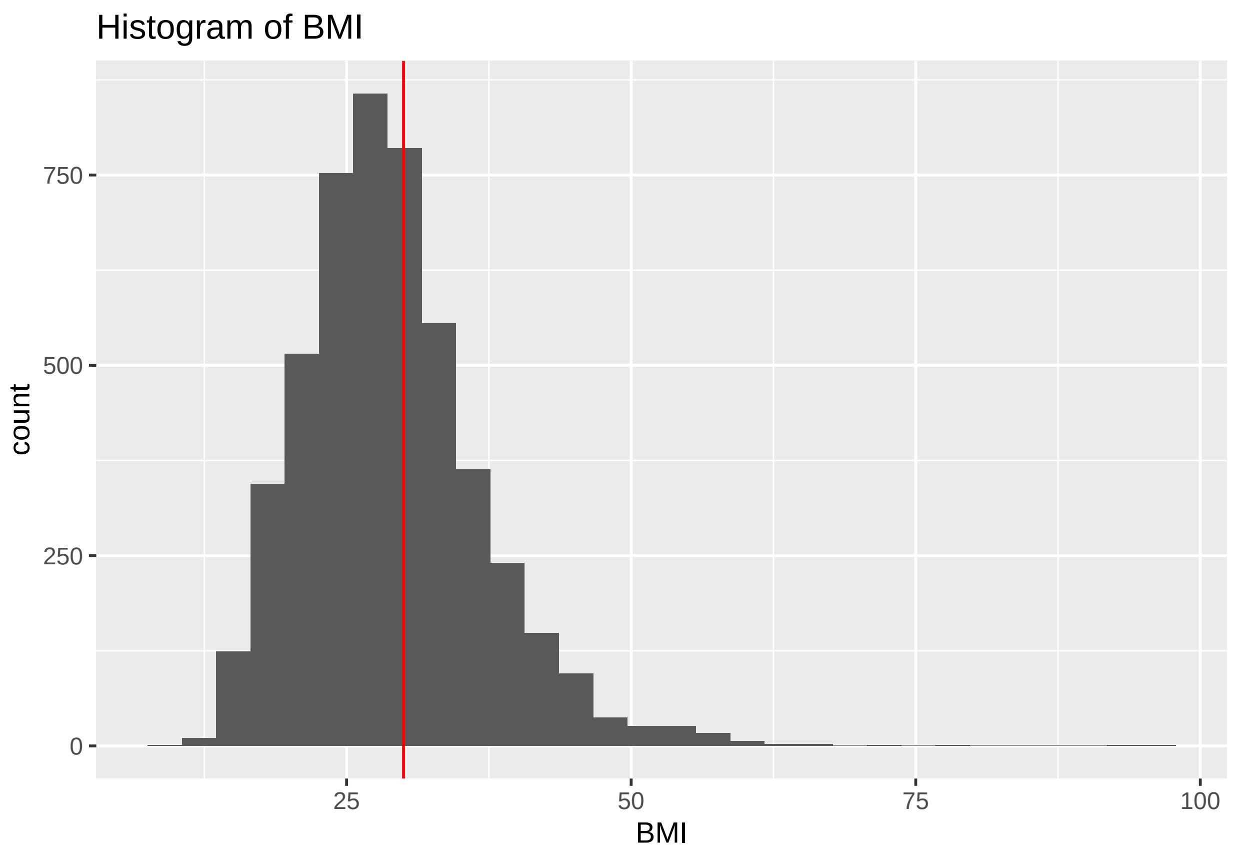


```r
smoking_labels =  c('Formerly Smoked', 'Never Smoked', 'Smokes', 'Unknown')

ggplot(stroke.df, aes(x = factor(smoking_status, levels = c(1,2,3,4), labels = smoking_labels))) +
  geom_bar() +
  labs(title = 'Distribution of Smoking Status', x = 'Smoking Status')
```



```
ggplot(stroke.df, aes(x = bmi)) +
  geom_histogram() +
  labs(title = 'Histogram of BMI', x = 'BMI') +
  geom_vline(xintercept = 30, color = 'red')
```



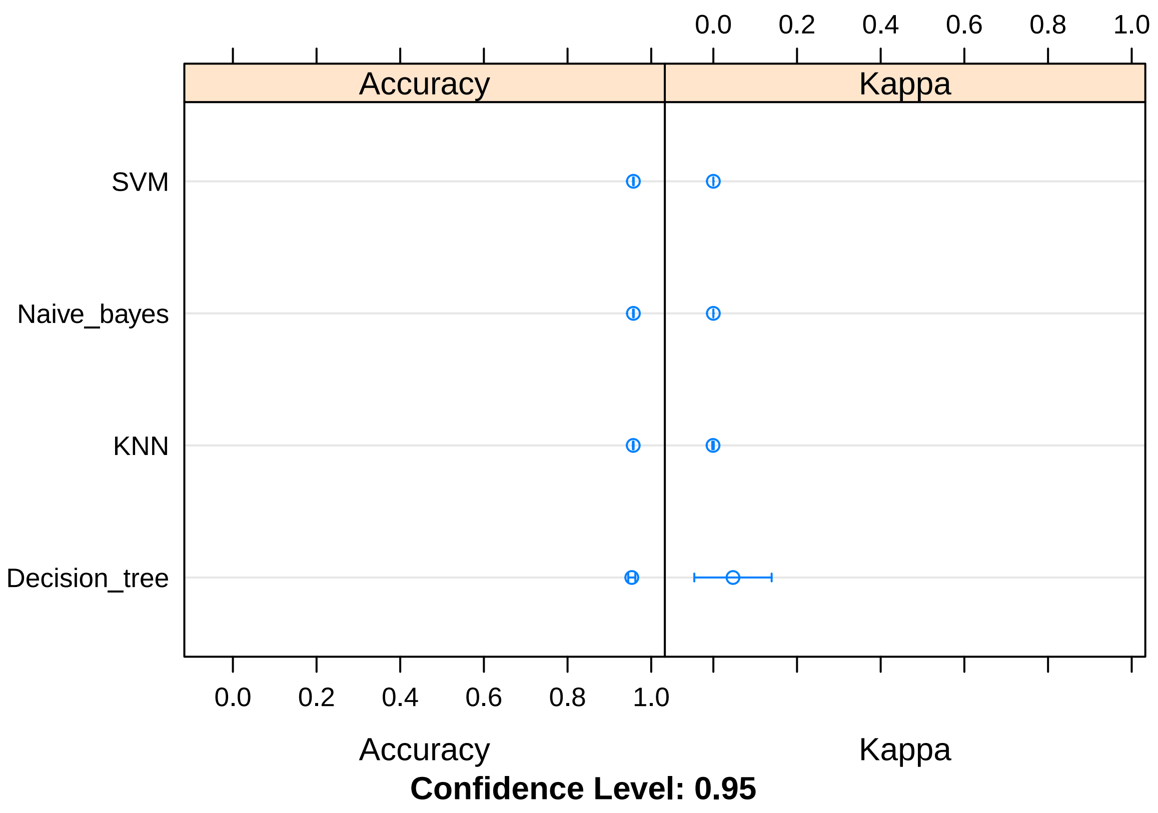
A lot of the other variables that represent health problems have similar distribution to stroke data. This is not as much of a problem when they are explanatory variables. it looks like half the patients either smoke or were smokers, although it is hard to tell because there is such a large unknown category.

Based on the distribution of BMI, it looks like most of the patients were overweight with about half being obese, as represented by the red line. Overall, while serious health problems are fairly rare, most of the patients appear the be unhealthy. This makes sense as most people who get strokes tend to have some preexisting health issues. # Testing models on raw data

```
tree.model <- train(stroke~., data = stroke.df, method = 'rpart', metric = 'Accuracy', trControl = cont

nb.model <- train(stroke~., data = stroke.df, method = 'nb', metric = 'Accuracy', laplace = 1, trContro
```

```
knn.model = train(stroke~., data = stroke.df, method = 'knn', metric = 'Accuracy', trControl = control,
svm.model = train(stroke~., data = stroke.df, method = 'svmRadial', metric = 'Accuracy', trControl = co
```



Model Evaluation

As expected, when working with the raw data, all the models hit 95% accuracy, which is the accuracy you would get by classifying everything as negative.

Decision Tree

```
##      truth
## pred    0    1
##    0 4700  209
##    1    0    0
```

Naive Bayes

```
##      truth
## pred    0    1
##    0 4700  209
##    1    0    0
```

KNN

```
##      truth
## pred    0    1
##    0 4699  209
##    1    1    0
```

SVM

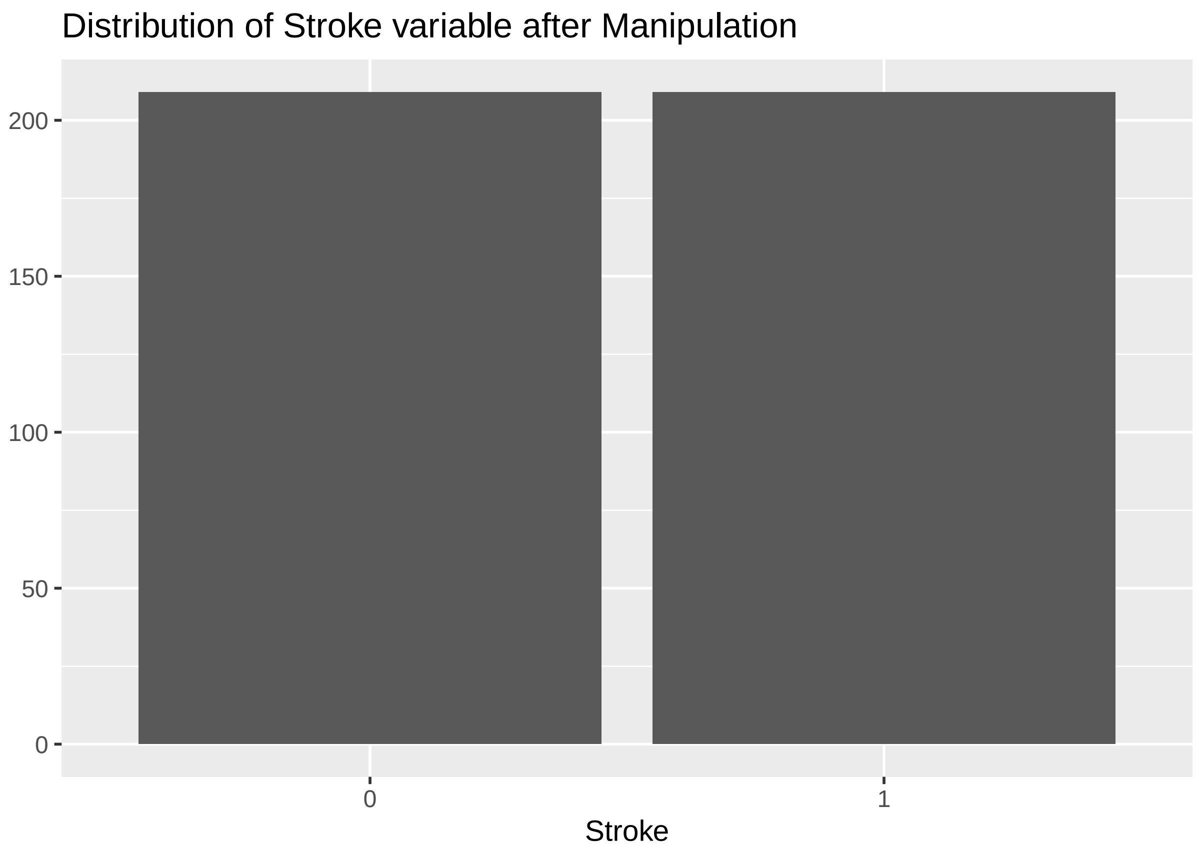
```
##      truth
## pred    0    1
##    0 4700  209
##    1    0    0
```

Of the initial models tried the best performing was technically the SVM, as it was able to detect one true positive. However, these confusion matrices confirm that the models are learning to classify everything as 'no stroke' and are not actually useful.

Trimming down the data

To attempt to fix this issue, most of the negative cases will be removed to make them comparable with the amount of positive cases.

```
## [1] 0.5
```



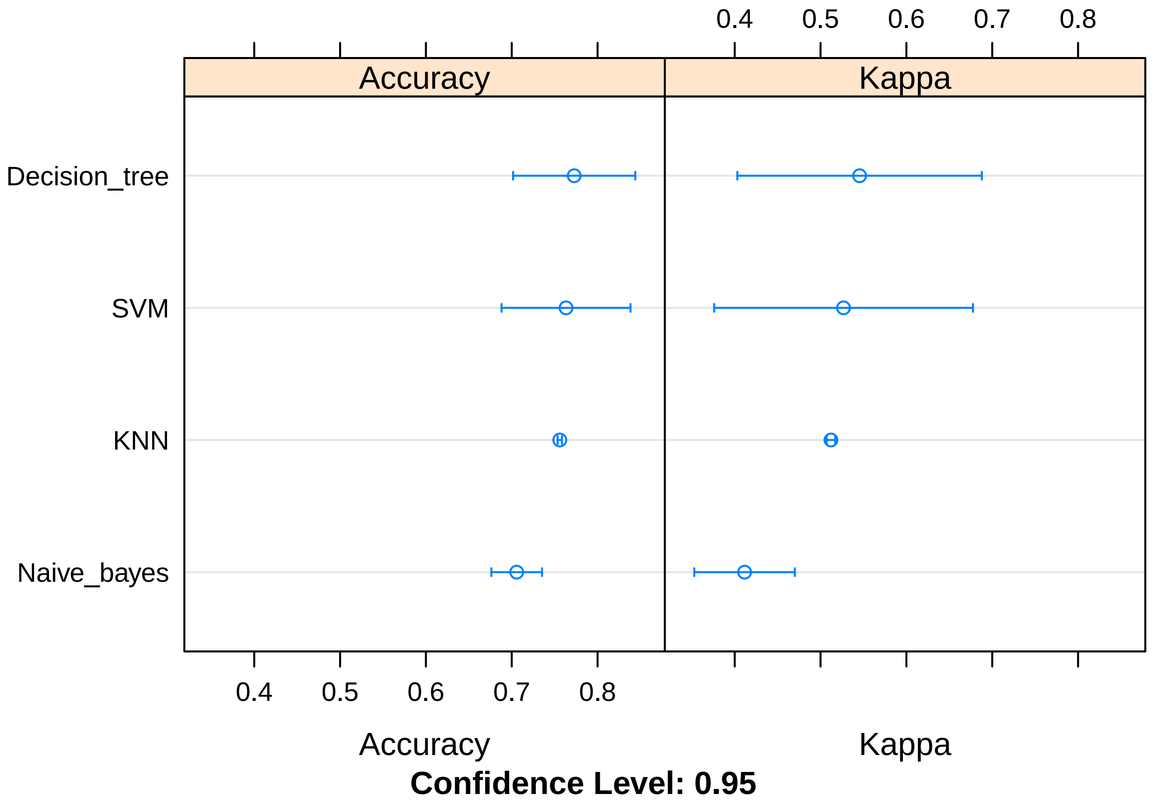
There will be a lot less data to work with, but there are still over 400 data points, so trends should still emerge.

```
tree.model.b <- train(stroke~., data = stroke.df.balanced, method = 'rpart', metric = 'Accuracy', trCon

nb.model.b <- train(stroke~., data = stroke.df.balanced, method = 'nb', metric = 'Accuracy', laplace = 

knn.model.b = train(stroke~., data = stroke.df.balanced, method = 'knn', metric = 'Accuracy', trControl
svm.model.b = train(stroke~., data = stroke.df.balanced, method = 'svmRadial', metric = 'Accuracy', trC
```

Model Evaluation



The accuracy of these models is less than the previous models, but if they are actually ably to detect true positives, it will be an improvement.

Decision Tree

```
##       truth
## pred     0    1
##    0 3252   27
##    1 1448  182
```

Naive Bayes

```
##       truth
## pred     0    1
##    0 3752   65
##    1  948  144
```

KNN

```
##       truth
## pred     0    1
##    0 3160   31
##    1 1540  178
```

SVM

```
##      truth
## pred    0    1
##    0 3078   29
##    1 1622  180
```

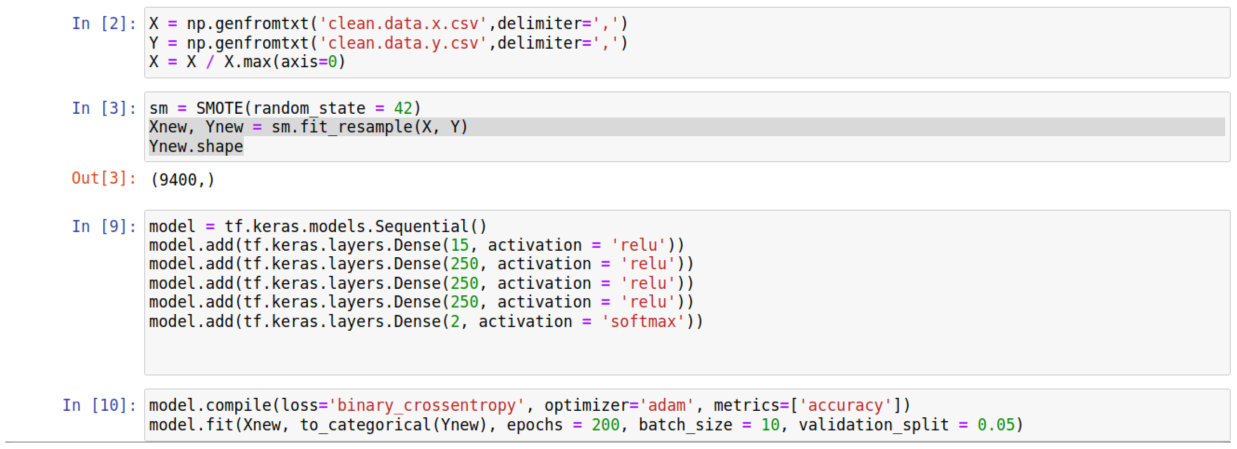
These models are able to correctly identify positives, although they do have a much higher false positive rate. This is somewhat acceptable, as there are so few true positives, that even a small false positive rate percentage wise will cause the precision to tank. Also in cases like this it is better to be over cautious, and give attention to those who are not actually at risk, than ignore those who are at risk. That being said, the false positive rate is incredibly high, meaning these models are still not very good.

Neural nets

Neural networks offer a more advanced type of model, that can often provide more accurate results than simpler models. However, they require a lot more data, so trimming down the data to make it balanced will no longer work.

SMOTE

To increase the number of data points in the train set, I used synthetic minority oversampling techniques (SMOTE). This method finds data points in the minority (positive stroke cases), and finds the K nearest neighbors of the same class, then generates new data points on the line between them. This creates a very large, balanced data set, that will help the neural network train more effectively. ## Archietcture



```
In [2]: X = np.genfromtxt('clean.data.x.csv',delimiter=',')
        Y = np.genfromtxt('clean.data.y.csv',delimiter=',')
        X = X / X.max(axis=0)

In [3]: sm = SMOTE(random_state = 42)
        Xnew, Ynew = sm.fit_resample(X, Y)
        Ynew.shape

Out[3]: (9400,)

In [9]: model = tf.keras.models.Sequential()
        model.add(tf.keras.layers.Dense(15, activation = 'relu'))
        model.add(tf.keras.layers.Dense(250, activation = 'relu'))
        model.add(tf.keras.layers.Dense(250, activation = 'relu'))
        model.add(tf.keras.layers.Dense(250, activation = 'relu'))
        model.add(tf.keras.layers.Dense(2, activation = 'softmax'))

In [10]: model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
         model.fit(Xnew, to_categorical(Ynew), epochs = 200, batch_size = 10, validation_split = 0.05)
```

Figure 1: Code for building a neural network using TensorFlow

The architecture for the final neural network consisted of 3 layers with 250 neurons each, using the Relu activation function (figure 1). ## Model Evaluation

The neural network greatly outperformed all previous models. When ran on the original data set, it correctly identified 206 out of 209 stroke cases, and only had 118 false positives from a data set of almost 5000 data points (figure 2). While the neural net performed well, it also had an advantage of a larger data set due to the SMOTE methodology. Unfortunately, I could not get SMOTE to work in R. Since other types of models are less notorious for needing huge data sets, it shouldn't impact performance too much, and much of the improvements can be attributed to the model

```
         Epoch 200/200
         893/893 [==============================] - 1s 2ms/step - loss: 0.0564 -
         accuracy: 0.9796 - val_loss: 0.0588 - val_accuracy: 0.9851

   [10]: <tensorflow.python.keras.callbacks.History at 0x7f69a874af90>

   [11]: sklearn.metrics.precision_score(Y, np.argmax(model.predict(X), axis = 1))

   [11]: 0.6358024691358025

   [12]: sklearn.metrics.recall_score(Y, np.argmax(model.predict(X), axis = 1))

   [12]: 0.9856459330143541

   [13]: sklearn.metrics.accuracy_score(Y, np.argmax(model.predict(X), axis = 1))

   [13]: 0.9753513953962111

   [14]: sklearn.metrics.confusion_matrix(Y, np.argmax(model.predict(X), axis = 1))

   [14]: array([[4582,  118],
                [   3,  206]])
```

Figure 2: Metrics from the Neural Network

Conclusion

Overall, the neural network provided by far the best results of any of the models, providing both an incredibly high accuracy rate, as well as very good detection of true positives. Other models were very difficult to balance between a very high false negative rate, and a very high false positive rate. manipulation of the data was very necessary, as without it, all the models (even the neural net) had horrible false negative rates, but only the neural net seemed to not be negatively effected in the opposite direction when the data was balanced.

Thus, using neural networks, it is very possible to predict with a high degree of accuracy, whether a patient is at risk of having a stroke given the variables provided. Due to the nature of neural networks, it is difficult to predict which variables are the most helpful, but it is definatley possible to get very accurate predictions.