Predicting Stroke

Charlie Oliver Ontoria Gomez

March 29, 2021

Introduction

This project aims to determine the impact of predictors in the data set on the risk of having a stroke. To do this, classification algorithms k-nearest neighbors, random forest, and logistic regression were used individually and together as an ensemble. A combination of accuracy and f1 score were used to determine the effectiveness of the algorithms.

The following code will show more information regarding the data set

```
dim(dataset)
```

```
## [1] 5110   12
```

```
names(dataset)
```

```
##  [1] "id"                "gender"            "age"
##  [4] "hypertension"      "heart_disease"     "ever_married"
##  [7] "work_type"         "Residence_type"    "avg_glucose_level"
## [10] "bmi"               "smoking_status"    "stroke"
```

```
head(dataset)
```

```
##      id gender age hypertension heart_disease ever_married     work_type
## 1  9046   Male  67            0             1          Yes       Private
## 2 51676 Female  61            0             0          Yes Self-employed
## 3 31112   Male  80            0             1          Yes       Private
## 4 60182 Female  49            0             0          Yes       Private
## 5  1665 Female  79            1             0          Yes Self-employed
## 6 56669   Male  81            0             0          Yes       Private
##   Residence_type avg_glucose_level  bmi  smoking_status stroke
## 1          Urban            228.69 36.6 formerly smoked      1
## 2          Rural            202.21  N/A    never smoked      1
## 3          Rural            105.92 32.5    never smoked      1
## 4          Urban            171.23 34.4          smokes      1
## 5          Rural            174.12   24    never smoked      1
## 6          Urban            186.21   29 formerly smoked      1
```

This dataset was downloaded from https://www.kaggle.com/fedesoriano/stroke-prediction-dataset. The stroke column is what we want to predict. Gender, age, a history of hypertension, a history of heart disease, being married, work type, residence type, average glucose level, and body mass index (bmi) are our predictors.

Method/Analysis

Before performing the algorithms, I made sure that the data set is clean and that the columns are of the right class.

Checking for NAs and Correcting Column Class

```
sum(is.na(dataset))
```

```
## [1] 0
```

It says there are no NAs but visually inspecting the data set, the column bmi has NAs. You can easily see the one from row 2. This means that the column's class is character. Let's check

```
class(dataset$bmi)
```

```
## [1] "character"
```

Changing bmi column into numeric and replacing NAs with the mean of the column

```
dataset <- dataset %>% mutate(bmi = as.numeric(bmi))
dataset$bmi[is.na(dataset$bmi)] <- mean(dataset$bmi, na.rm = TRUE)
```

Since we are doing classification, we need the stroke column to be a factor

```
class(dataset$stroke)
```

```
## [1] "integer"
```

This is the code to change that character column into a factor column

```
dataset <- dataset %>% mutate(stroke = as.factor(stroke))
```

I also removed a row because the gender is "Other". I could not make sense of it so I simply removed it knowing that it would not affect the results.

```
dataset <- dataset[-which(dataset$gender == "Other"),]
```

Exploratory Data Analysis

```
mean(dataset$gender == "Male")
```

```
## [1] 0.4139753
```

```
median(dataset$age)
```

```
## [1] 45
```

```
min(dataset$age)
```

```
## [1] 0.08
```

```
sum(dataset$age <1)
```

```
## [1] 43
```

```
max(dataset$age)
```

```
## [1] 82
```

```
mean(dataset$Residence_type == "Urban")
```

```
## [1] 0.5081229
```

From the above analysis we can see that there's more women (59%) than men (41%). We have 43 infants in the data set. I did not see this as an anomaly since babies also get stroke. The residence_type distribution is well balanced.

Checking the prevalence in the stroke column was also important

```
mean(dataset$stroke == 1)
```

```
## [1] 0.04873752
```

We can see that there is a very low prevalence of people with stroke. This affected the algorithm results.

The data set was first divided into a training set and a test set. Then I ran the algorithms.

Running the Algorithms

K-Nearest Neighbors

```
control <- trainControl(method = "cv", number = 10) #classic 10-fold cross validation
fit_knn <- train(stroke ~ gender + age + hypertension + heart_disease + ever_married + work_type + Resi
prediction_knn <- predict(fit_knn, newdata = test_set)

confusionMatrix(prediction_knn, test_set$stroke)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 2422  125
##          1    8    0
##
##                Accuracy : 0.9479
##                  95% CI : (0.9386, 0.9562)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : 0.7838
##
##                   Kappa : -0.0059
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9967
##             Specificity : 0.0000
##          Pos Pred Value : 0.9509
##          Neg Pred Value : 0.0000
##              Prevalence : 0.9511
##          Detection Rate : 0.9479
##    Detection Prevalence : 0.9969
##       Balanced Accuracy : 0.4984
##
##        'Positive' Class : 0
##
```

As we can see we have an accuracy of 94.79% which is great. But unfortunately, the algorithm was not able to predict any true positives. Accuracy alone is not a good measurement of effectiveness of our model. We will use f1 score along with accuracy. This is due to the low prevalence of stroke in the data. We can get a high accuracy simply by predicting 0.

Random Forest

Let's see if random forest trees are better. We'll use rborist for faster computation time

```
control <- trainControl(method = "cv", number = 5, p = 0.8) #5-fold for faster computation
fit_rborist <- train(stroke ~ gender + age + hypertension + heart_disease + ever_married + work_type + I
prediction_rborist <- predict(fit_rborist, newdata = test_set)

confusionMatrix(prediction_knn, test_set$stroke)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2422  125
##          1    8    0
##
##                Accuracy : 0.9479
##                  95% CI : (0.9386, 0.9562)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : 0.7838
##
##                   Kappa : -0.0059
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9967
##             Specificity : 0.0000
##          Pos Pred Value : 0.9509
##          Neg Pred Value : 0.0000
##              Prevalence : 0.9511
##          Detection Rate : 0.9479
##    Detection Prevalence : 0.9969
##       Balanced Accuracy : 0.4984
##
##        'Positive' Class : 0
##
```

The results were the same. It was not able to prevail over the prevalence.

Modifying the Data

There are several ways to go about this but the easiest way seems to be balancing the data set. A small sample of the no-stroke data equivalent to the number of stroke-data was used.

```
dataset_stroke <- dataset %>% filter(stroke == 1)
dataset_no_stroke <- dataset %>% filter(stroke == 0)

set.seed(1, sample.kind = "Rounding") #set.seed(1) for R versions 3.5 or older
index <- sample(1:nrow(dataset_no_stroke), size = nrow(dataset_stroke), replace = FALSE)
dataset_no_stroke <- dataset_no_stroke[index,]

dataset_balanced <- full_join(dataset_stroke, dataset_no_stroke)
```

Redoing the Exploratory Data Analysis

```
mean(dataset_balanced$gender == "Male")
```

```
## [1] 0.437751
```

```
median(dataset_balanced$age)
```

```
## [1] 59
```

```
min(dataset_balanced$age)
```

```
## [1] 0.24
```

```
sum(dataset_balanced$age <1)
```

```
## [1] 2
```

```
max(dataset_balanced$age)
```

```
## [1] 82
```

```
mean(dataset_balanced$Residence_type == "Urban")
```

```
## [1] 0.5200803
```

The male distribution changed from 41% to 44% which is better. The median age changed from 43 to 59 years old. The number of infants are down from 43 to 2 which is expected. There's a bigger percentage of urban dwellers at 52% up from 50%. This looks ok.

The dataset was again split into a training set and a test set. Since there is only a few rows , a 50/50 distribution was used.

Running the Algorithms with the Balanced Data Set

K-Nearest Neighbors

```
control <- trainControl(method = "cv", number = 10) #classic 10-fold cross validation
fit_knn_balanced <- train(stroke ~ gender + age + hypertension + heart_disease + ever_married + work_typ
prediction_knn_balanced <- predict(fit_knn_balanced, newdata = test_set_balanced)

confusionMatrix(prediction_knn_balanced, test_set_balanced$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  80  19
##          1  45 106
##
##                Accuracy : 0.744
##                  95% CI : (0.6852, 0.7969)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 2.777e-15
##
##                   Kappa : 0.488
##
##  Mcnemar's Test P-Value : 0.001778
##
```

```
##             Sensitivity : 0.8480
##             Specificity : 0.6400
##          Pos Pred Value : 0.7020
##          Neg Pred Value : 0.8081
##              Prevalence : 0.5000
##          Detection Rate : 0.4240
##    Detection Prevalence : 0.6040
##       Balanced Accuracy : 0.7440
##
##        'Positive' Class : 1
##
```

```
#F1 Score
F_meas(prediction_knn_balanced, test_set_balanced$stroke)
```

```
## [1] 0.7142857
```

This is much better. We are able to predict 106 positives on the people with stroke. We have a higher sensitivity vs specificity which is what we want but not too high. The f1 score looks ok.

Random Forest

```
control <- trainControl(method = "cv", number = 10) #classic 10-fold cross validation
fit_rf_balanced <- train(stroke ~ gender + age + hypertension + heart_disease + ever_married + work_typ
prediction_rf_balanced <- predict(fit_rf_balanced, newdata = test_set_balanced)

confusionMatrix(prediction_rf_balanced, test_set_balanced$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  80  21
##          1  45 104
##
##                Accuracy : 0.736
##                  95% CI : (0.6768, 0.7895)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 2.276e-14
##
##                   Kappa : 0.472
##
##  Mcnemar's Test P-Value : 0.004639
##
##             Sensitivity : 0.8320
##             Specificity : 0.6400
##          Pos Pred Value : 0.6980
##          Neg Pred Value : 0.7921
##              Prevalence : 0.5000
##          Detection Rate : 0.4160
##    Detection Prevalence : 0.5960
##       Balanced Accuracy : 0.7360
##
##        'Positive' Class : 1
##
```

```
#F1 Score
F_meas(prediction_rf_balanced, test_set_balanced$stroke)
```

```
## [1] 0.7079646
```

It looks like knn is slightly better. In here there are 104 true positives compared to knn's 106 true positives. There's an additional 2 false negatives that were true positives in knn. The accuracy of knn is 0.744 compared to random forest's 0.736. Because of that the f1 score is slight lower as well. Of course we cannot use this statistic to judge which algorithm is better because that includes the test_set. Let's check the minimum accuracy estimates

```
fit_rf_balanced$results$Accuracy[fit_rf_balanced$bestTune[1,]]
```

```
## [1] 0.7817051
```

```
fit_knn_balanced$results$Accuracy[2]
```

```
## [1] 0.7718077
```

Here we can see that random forest is slightly better than knn with our train_set alone. All in all, it's almost the same. Random forest has better accuracy according to minimum accuracy estimates but knn performed slightly better on the test set with both accuracy and f1 score.

Checking for Variable Importance of the Predictors

```
varImp(fit_rf_balanced)
```

```
## rf variable importance
##
##                              Overall
## age                          100.000
## avg_glucose_level             47.562
## bmi                           33.141
## ever_marriedYes               14.339
## hypertension                  12.597
## smoking_statusnever smoked     9.591
## heart_disease                  9.003
## work_typeSelf-employed         7.404
## work_typePrivate               5.974
## genderMale                     5.782
## Residence_typeUrban            5.665
## smoking_statussmokes           4.526
## work_typeGovt_job              4.361
## smoking_statusUnknown          4.242
## work_typeNever_worked          0.000
```

From this we can see that age is the biggest factor in strokes. It is followed by avg_glucose_level and bmi. These are well known factors. genderMale and Residence_typeUrban have minor effects. This is something we can consider. There might be underlying reasons concerning genetics and environment. I am wondering about the ever_marriedYes, hypertension, and heart_disease. Hypertension and heart disease are known to contribute to the risk of having a stroke and I expected them to have more impact. Intuitively, marriage does not seem like it should have a big impact. This could be due to our data.

```
#Checking for Married Distribution
mean(dataset_balanced$ever_married == "Yes")
```

```
## [1] 0.7670683
```

```
mean(dataset$ever_married == "Yes")
```

```
## [1] 0.6562928
```

There is a prevalence of married people in our dataset. Therefore, we cannot conclusively say marriage has an impact on increasing the risk of stroke.

```
#Checking for Heart Disease Distribution
mean(dataset_balanced$heart_disease)
```

```
## [1] 0.1144578
```

```
mean(dataset$heart_disease)
```

```
## [1] 0.05402231
```

```
#Checking for Hypertension Distribution
mean(dataset_balanced$hypertension)
```

```
## [1] 0.184739
```

```
mean(dataset$hypertension)
```

```
## [1] 0.09747504
```

Only a minority of the people in our data set have heart_disease or hypertension. This could be the reason why it has less impact than it should. As for smoking, it seems that non-smokers have a higher risk of having a stroke. Let's check

```
#Checking for Smoking Status Distribution
data.frame(dataset$smoking_status) %>% distinct()
```

```
##   dataset.smoking_status
## 1        formerly smoked
## 2           never smoked
## 3                 smokes
## 4                Unknown
```

```
mean(dataset$smoking_status == "never smoked")
```

```
## [1] 0.3703269
```

```
mean(dataset$smoking_status == "formerly smoked")
```

```
## [1] 0.173028
```

```
mean(dataset$smoking_status == "Unknown")
```

```
## [1] 0.3022118
```

```
mean(dataset$smoking_status == "smokes")
```

```
## [1] 0.1544334
```

```
data.frame(dataset_balanced$smoking_status) %>% distinct()
```

```
##   dataset_balanced.smoking_status
## 1                 formerly smoked
## 2                    never smoked
## 3                          smokes
## 4                         Unknown
```

```
mean(dataset_balanced$smoking_status == "never smoked")
```

```
## [1] 0.3995984
```

```
mean(dataset_balanced$smoking_status == "formerly smoked")
```

```
## [1] 0.2148594
```

```
mean(dataset_balanced$smoking_status == "Unknown")
```

```
## [1] 0.2208835
```

```
mean(dataset_balanced$smoking_status == "smokes")
```

```
## [1] 0.1646586
```

A third of the data is unknown. Only 15%/16% actually smokes. This assessment by the algorithm is unreliable.

The work_type variable importance does not make much sense either. We will attribute whatever value associated with it as random chance.

There are faults in our data that could be attributed to data collection.

Logistic Regression

We will now test logistic regression (glm). Let's check how it does individually

```
control <- trainControl(method = "cv", number = 10) #classic 10-fold cross validation
fit_glm_balanced <- train(stroke ~ gender + age + hypertension + heart_disease + ever_married + work_ty
prediction_glm_balanced <- predict(fit_glm_balanced, newdata = test_set_balanced)

confusionMatrix(prediction_glm_balanced, test_set_balanced$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  81  22
##          1  44 103
##
##                Accuracy : 0.736
##                  95% CI : (0.6768, 0.7895)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 2.276e-14
##
##                   Kappa : 0.472
##
##  Mcnemar's Test P-Value : 0.00974
##
##             Sensitivity : 0.8240
##             Specificity : 0.6480
##          Pos Pred Value : 0.7007
##          Neg Pred Value : 0.7864
##              Prevalence : 0.5000
##          Detection Rate : 0.4120
##    Detection Prevalence : 0.5880
##       Balanced Accuracy : 0.7360
```

```
##
##        'Positive' Class : 1
##
```

```
#F1 Score
F_meas(prediction_glm_balanced, test_set_balanced$stroke)
```

```
## [1] 0.7105263
```

Looks ok. It almost has the same statistics as knn and rf, only slightly worse.

Running the Algorithms with Chosen Predictors

We will redo our models with our chosen predictors which are gender, age, hypertension, heart_disease, residence type, and bmi and see how it fares with the old model with all the predictors

K-Nearest Neighbors Model with Chosen Predictors

```
control <- trainControl(method = "cv", number = 10) #classic 10-fold cross validation
fit_knn_balanced_2 <- train(stroke ~ gender + age + hypertension + heart_disease + Residence_type + avg
prediction_knn_balanced_2 <- predict(fit_knn_balanced, newdata = test_set_balanced)

confusionMatrix(prediction_knn_balanced_2, test_set_balanced$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  80  19
##          1  45 106
##
##                Accuracy : 0.744
##                  95% CI : (0.6852, 0.7969)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 2.777e-15
##
##                   Kappa : 0.488
##
##  Mcnemar's Test P-Value : 0.001778
##
##             Sensitivity : 0.8480
##             Specificity : 0.6400
##          Pos Pred Value : 0.7020
##          Neg Pred Value : 0.8081
##              Prevalence : 0.5000
##          Detection Rate : 0.4240
##    Detection Prevalence : 0.6040
##       Balanced Accuracy : 0.7440
##
##        'Positive' Class : 1
##
```

```
#F1 Score
F_meas(prediction_knn_balanced_2, test_set_balanced$stroke)
```

```
## [1] 0.7142857
```

Same results as with the k-nearest neighbors model using all the predictors.

Random Forest Model with Chosen Predictors

```
control <- trainControl(method = "cv", number = 10) #classic 10-fold cross validation
fit_rf_balanced_2 <- train(stroke ~ gender + age + hypertension + heart_disease + Residence_type + avg_g
prediction_rf_balanced_2 <- predict(fit_knn_balanced, newdata = test_set_balanced)

confusionMatrix(prediction_rf_balanced_2, test_set_balanced$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  80  19
##          1  45 106
##
##                Accuracy : 0.744
##                  95% CI : (0.6852, 0.7969)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 2.777e-15
##
##                   Kappa : 0.488
##
##  Mcnemar's Test P-Value : 0.001778
##
##             Sensitivity : 0.8480
##             Specificity : 0.6400
##          Pos Pred Value : 0.7020
##          Neg Pred Value : 0.8081
##              Prevalence : 0.5000
##          Detection Rate : 0.4240
##    Detection Prevalence : 0.6040
##       Balanced Accuracy : 0.7440
##
##        'Positive' Class : 1
##
```

```
#F1 Score
F_meas(prediction_rf_balanced_2, test_set_balanced$stroke)
```

```
## [1] 0.7142857
```

Not surprisingly, our random forest model improved a little bit. It has now an equal score on both accuracy and f1 as with k-nearest neighbors. It was able to predict 2 additional true positives.

Let's check the new variable importance

```
varImp(fit_rf_balanced_2)
```

```
## rf variable importance
##
##                    Overall
## age               100.0000
## avg_glucose_level  45.5989
## bmi                35.3379
## hypertension        5.0635
## heart_disease         2.3305
## Residence_typeUrban   0.5722
## genderMale            0.0000
```

In our new model, gender and residence type has almost no effect. Most of the calculations are made on age, avg_glucose_level, and bmi, with a minor heart_disease and hypertension effect.

Logistic Regression Model with Chosen Predictors

```
control <- trainControl(method = "cv", number = 10) #classic 10-fold cross validation
fit_glm_balanced_2 <- train(stroke ~ gender + age + hypertension + heart_disease + Residence_type + avg
prediction_glm_balanced_2 <- predict(fit_glm_balanced_2, newdata = test_set_balanced)

confusionMatrix(prediction_glm_balanced_2, test_set_balanced$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  84  21
##          1  41 104
##
##                Accuracy : 0.752
##                  95% CI : (0.6937, 0.8043)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 3.119e-16
##
##                   Kappa : 0.504
##
##  Mcnemar's Test P-Value : 0.01582
##
##             Sensitivity : 0.8320
##             Specificity : 0.6720
##          Pos Pred Value : 0.7172
##          Neg Pred Value : 0.8000
##              Prevalence : 0.5000
##          Detection Rate : 0.4160
##    Detection Prevalence : 0.5800
##       Balanced Accuracy : 0.7520
##
##        'Positive' Class : 1
##
```

```
#F1 Score
F_meas(prediction_glm_balanced_2, test_set_balanced$stroke)
```

```
## [1] 0.7304348
```

Our logistic regression model also performed slightly better, predicting an additional true positive which improved accuracy and f1 score. It has the highest f1 score and accuracy among the all the models tested so far.

Ensemble Model

Here we will check if an ensemble of random forest, k-nearest neighbors, and logistic regression is better than the individual algorithms. An All Predictors Model and Chosen Predictors Model was used.

```
models <- c("knn", "rf", "glm")

set.seed(1, sample.kind = "Rounding") #set.seed(1) for R versions 3.5 or older
fits <- lapply(models, function(x){
  print(x)
  train(stroke ~ gender + age + hypertension + heart_disease + ever_married + work_type + Residence_typ
})
```

Ensemble All Predictors Model

```
## [1] "knn"
## [1] "rf"
## [1] "glm"
```

```
names(fits) <- models

predictions <- sapply(fits, function(x)
  predict(x, newdata = test_set_balanced))

votes <- rowMeans(predictions == "1")
y_hat <- ifelse(votes > 0.5, "1", "0")
confusionMatrix(factor(y_hat), test_set_balanced$stroke)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  81  19
##          1  44 106
##
##                Accuracy : 0.748
##                  95% CI : (0.6894, 0.8006)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 9.405e-16
##
##                   Kappa : 0.496
##
##  Mcnemar's Test P-Value : 0.002497
##
##             Sensitivity : 0.6480
##             Specificity : 0.8480
##          Pos Pred Value : 0.8100
##          Neg Pred Value : 0.7067
##              Prevalence : 0.5000
##          Detection Rate : 0.3240
##    Detection Prevalence : 0.4000
##       Balanced Accuracy : 0.7480
##
##        'Positive' Class : 0
##
```

```
#F1 Score
F_meas(factor(y_hat), test_set_balanced$stroke)
```

```
## [1] 0.72
```

```r
models <- c("knn", "rf", "glm")

set.seed(1, sample.kind = "Rounding") #set.seed(1) for R versions 3.5 or older
fits_2 <- lapply(models, function(x){
  print(x)
  train(stroke ~ gender + age + hypertension + heart_disease + Residence_type + avg_glucose_level + bmi
})
```

Ensemble Chosen Predictors Model

```
## [1] "knn"
## [1] "rf"
## [1] "glm"
```

```r
names(fits_2) <- models

predictions_2 <- sapply(fits_2, function(x)
  predict(x, newdata = test_set_balanced))

votes_2 <- rowMeans(predictions_2 == "1")
y_hat_2 <- ifelse(votes_2 > 0.5, "1", "0")
confusionMatrix(factor(y_hat_2), test_set_balanced$stroke)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  82  18
##          1  43 107
##
##                Accuracy : 0.756
##                  95% CI : (0.6979, 0.8079)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.512
##
##  Mcnemar's Test P-Value : 0.00212
##
##             Sensitivity : 0.6560
##             Specificity : 0.8560
##          Pos Pred Value : 0.8200
##          Neg Pred Value : 0.7133
##              Prevalence : 0.5000
##          Detection Rate : 0.3280
##    Detection Prevalence : 0.4000
##       Balanced Accuracy : 0.7560
##
##        'Positive' Class : 0
##
```

```r
#F1 Score
F_meas(factor(y_hat_2), test_set_balanced$stroke)
```

```
## [1] 0.7288889
```

Results

The ensemble model is better than the individual algorithms based on accuracy. Logistic Regression Chosen Predictors model was able to get a slightly better f1 score. Using our chosen predictors improve the accuracy and f1 score slightly as shown here.

```
# Accuracy and F1 Scores

#Ensemble All Predictors Model
mean(y_hat == test_set_balanced$stroke)         #Accuracy
```

```
## [1] 0.748
```

```
F_meas(factor(y_hat), test_set_balanced$stroke) #F1 Score
```

```
## [1] 0.72
```

```
#Ensemble Chosen Predictors Model
mean(y_hat_2 == test_set_balanced$stroke)         #Accuracy
```

```
## [1] 0.756
```

```
F_meas(factor(y_hat_2), test_set_balanced$stroke) #F1 Score
```

```
## [1] 0.7288889
```

These are the best results of the individual algorithms which are on the chosen predictors model.

```
#K-nearest Neighbors Chosen Predictors Model
mean(prediction_knn_balanced_2 == test_set_balanced$stroke) #Accuracy
```

```
## [1] 0.744
```

```
F_meas(prediction_knn_balanced_2, test_set_balanced$stroke) #F1 Score
```

```
## [1] 0.7142857
```

```
#Random Forest Chosen Predictors Model
mean(prediction_rf_balanced_2 == test_set_balanced$stroke) #Accuracy
```

```
## [1] 0.744
```

```
F_meas(prediction_rf_balanced_2, test_set_balanced$stroke) #F1 Score
```

```
## [1] 0.7142857
```

```
#Logistic Regression Chosen Predictors Model
mean(prediction_glm_balanced_2 == test_set_balanced$stroke) #Accuracy
```

```
## [1] 0.752
```

```
F_meas(prediction_glm_balanced_2, test_set_balanced$stroke) #F1 Score
```

```
## [1] 0.7304348
```

We can clearly see that the Ensemble Chosen Predictors Model is the best model out of the ones we tested based on accuracy. The model with the highest f1 score is the Logistic Regression Chose Predictors Model but only slightly higher than the Ensemble Chose Predictors Model.

Conclusion

```
## rf variable importance
##
##                     Overall
## age                100.0000
## avg_glucose_level   45.5989
## bmi                 35.3379
## hypertension         5.0635
## heart_disease        2.3305
## Residence_typeUrban  0.5722
## genderMale           0.0000
```

Based on this analysis, we can conclusively say that age, diet (average glucose level), and weight (bmi) contribute to having a stroke. A history of hypertension and heart disease has a minor effect on stroke risk. Being male and living in an urban setting was first thought to have a minor influence in increasing the risk of stroke but with the new random forest model, their value in the calculations is zero or close to it. We cannot say with certainty that the other factors have an effect on increasing the risk of stroke due to the nature of our data. An ensemble of random forest, logistic regression and k-nearest neighbors is a better algorithm than any of the individual algorithms. Using only the chosen unbiased predictors gives a slightly better result. The Ensemble Chosen Predictors Model is the best model out of all the models tested. The results of this report was limited by the bias in the excluded predictors in the data and the low prevalence of stroke in the original data set.