Code ▾

Random Forest Classification using Machine Learning for Stroke Prediction

Building a step wise step Machine Learning Mode, Here we are using the Stroke Prediction Dataset from Kaggle to make predictions. Our task is to examine existing patient records in the training set and use that knowledge to predict whether a patient in the evaluation set is likely to have a stroke or not.

Hide

```
library(tidyverse)
library(caret)
library(randomForest)
library(ggplot2)
```

Read and Import the data

Hide

```
df_stroke <- read.csv("~/Downloads/healthcare-dataset-stroke-data.csv")
```

Undertanding the data and observing our column and rows to understand the structure better

Hide

```
glimpse(df_stroke)
```

```
Rows: 5,110
Columns: 12
$ id                <int> 9046, 51676, 31112, 60182, 1665, 56669, 53882, 10434, 2741
9, 60491, 12109, 12095, 12175, 8213, 5317, 58202, 56112, 3412…
$ gender            <chr> "Male", "Female", "Male", "Female", "Female", "Male", "Mal
e", "Female", "Female", "Female", "Female", "Female", "Female…
$ age               <dbl> 67, 61, 80, 49, 79, 81, 74, 69, 59, 78, 81, 61, 54, 78, 79,
50, 64, 75, 60, 57, 71, 52, 79, 82, 71, 80, 65, 58, 69, 59,…
$ hypertension      <int> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,…
$ heart_disease     <int> 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,…
$ ever_married      <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", "Ye
s", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "…
$ work_type         <chr> "Private", "Self-employed", "Private", "Private", "Self-emp
loyed", "Private", "Private", "Private", "Private", "Private…
$ Residence_type    <chr> "Urban", "Rural", "Rural", "Urban", "Rural", "Urban", "Rura
l", "Urban", "Rural", "Urban", "Rural", "Rural", "Urban", "U…
$ avg_glucose_level <dbl> 228.69, 202.21, 105.92, 171.23, 174.12, 186.21, 70.09, 94.3
9, 76.15, 58.57, 80.43, 120.46, 104.51, 219.84, 214.09, 167.…
$ bmi               <chr> "36.6", "N/A", "32.5", "34.4", "24", "29", "27.4", "22.8",
"N/A", "24.2", "29.7", "36.8", "27.3", "N/A", "28.2", "30.9"…
$ smoking_status    <chr> "formerly smoked", "never smoked", "never smoked", "smoke
s", "never smoked", "formerly smoked", "never smoked", "never …
$ stroke            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
```

Hide

```
summary (df_stroke)
```

```
       id           gender               age         hypertension     heart_disease    
ever_married        work_type         Residence_type    
 Min.   :   67   Length:5110        Min.   : 0.08   Min.   :0.00000   Min.   :0.00000  
Length:5110        Length:5110        Length:5110       
 1st Qu.:17741   Class :character   1st Qu.:25.00   1st Qu.:0.00000   1st Qu.:0.00000  
Class :character   Class :character   Class :character  
 Median :36932   Mode  :character   Median :45.00   Median :0.00000   Median :0.00000  
Mode  :character   Mode  :character   Mode  :character  
 Mean   :36518                      Mean   :43.23   Mean   :0.09746   Mean   :0.05401  
 3rd Qu.:54682                      3rd Qu.:61.00   3rd Qu.:0.00000   3rd Qu.:0.00000  
 Max.   :72940                      Max.   :82.00   Max.   :1.00000   Max.   :1.00000  
 avg_glucose_level     bmi            smoking_status         stroke       
 Min.   : 55.12    Length:5110        Length:5110        Min.   :0.00000  
 1st Qu.: 77.25    Class :character   Class :character   1st Qu.:0.00000  
 Median : 91.89    Mode  :character   Mode  :character   Median :0.00000  
 Mean   :106.15                                          Mean   :0.04873  
 3rd Qu.:114.09                                          3rd Qu.:0.00000  
 Max.   :271.74                                          Max.   :1.00000  
```

There is a row in gender column who do not regnise themselves as male or female so we will kick them out.

Hide

```
count(df_stroke, gender == 'Other')
```

gender == "Other"	**n**
<lgl>	<int>
FALSE	5109
TRUE	1

2 rows

Hide

```
df_stroke<- df_stroke[df_stroke$gender != 'Other',]
```

Now we have new dataframe containing only the male and female dataset, we will check the NA in the table and drop those rows so that there is no error encountered later creating the algorithm.

Hide

```
sum(df_stroke$bmi == 'N/A')
```

```
[1] 201
```

Hide

```
summary(df_stroke)
```

```
       id           gender                age         hypertension     heart_disease 
ever_married        work_type         Residence_type    
 Min.   :   67   Length:5109        Min.   : 0.08   Min.   :0.00000   Min.   :0.00000  
Length:5109        Length:5109        Length:5109       
 1st Qu.:17740   Class :character   1st Qu.:25.00   1st Qu.:0.00000   1st Qu.:0.00000  
Class :character   Class :character   Class :character  
 Median :36922   Mode  :character   Median :45.00   Median :0.00000   Median :0.00000  
Mode  :character   Mode  :character   Mode  :character  
 Mean   :36514                      Mean   :43.23   Mean   :0.09748   Mean   :0.05402  
 3rd Qu.:54643                      3rd Qu.:61.00   3rd Qu.:0.00000   3rd Qu.:0.00000  
 Max.   :72940                      Max.   :82.00   Max.   :1.00000   Max.   :1.00000  
 avg_glucose_level     bmi              smoking_status         stroke       
 Min.   : 55.12    Length:5109        Length:5109        Min.   :0.00000  
 1st Qu.: 77.24    Class :character   Class :character   1st Qu.:0.00000  
 Median : 91.88    Mode  :character   Mode  :character   Median :0.00000  
 Mean   :106.14                                          Mean   :0.04874  
 3rd Qu.:114.09                                          3rd Qu.:0.00000  
 Max.   :271.74                                          Max.   :1.00000  
```

We can see threre are 201 enteries in bmi column which is empty, this 201 entries consisit of 5% of the total data so I would replace it with mean of the bmi column instead of just dropping the column

Hide

```
df_stroke$bmi <- as.numeric(df_stroke$bmi)# converting the datatype to numeric as it
was in character before

df_stroke$bmi[is.na(df_stroke$bmi)] <- mean(df_stroke$bmi, na.rm = TRUE)
```

Hide

```
glimpse(df_stroke)
```

```
Rows: 5,109
Columns: 12
$ id                <int> 9046, 51676, 31112, 60182, 1665, 56669, 53882, 10434, 2741
9, 60491, 12109, 12095, 12175, 8213, 5317, 58202, 56112, 3412…
$ gender            <chr> "Male", "Female", "Male", "Female", "Female", "Male", "Mal
e", "Female", "Female", "Female", "Female", "Female", "Female…
$ age               <dbl> 67, 61, 80, 49, 79, 81, 74, 69, 59, 78, 81, 61, 54, 78, 79,
50, 64, 75, 60, 57, 71, 52, 79, 82, 71, 80, 65, 58, 69, 59,…
$ hypertension      <int> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,…
$ heart_disease     <int> 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,…
$ ever_married      <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", "Ye
s", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "…
$ work_type         <chr> "Private", "Self-employed", "Private", "Private", "Self-emp
loyed", "Private", "Private", "Private", "Private", "Private…
$ Residence_type    <chr> "Urban", "Rural", "Rural", "Urban", "Rural", "Urban", "Rura
l", "Urban", "Rural", "Urban", "Rural", "Rural", "Urban", "U…
$ avg_glucose_level <dbl> 228.69, 202.21, 105.92, 171.23, 174.12, 186.21, 70.09, 94.3
9, 76.15, 58.57, 80.43, 120.46, 104.51, 219.84, 214.09, 167.…
$ bmi               <dbl> 36.60000, 28.89456, 32.50000, 34.40000, 24.00000, 29.00000,
27.40000, 22.80000, 28.89456, 24.20000, 29.70000, 36.80000,…
$ smoking_status    <chr> "formerly smoked", "never smoked", "never smoked", "smoke
s", "never smoked", "formerly smoked", "never smoked", "never …
$ stroke            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
```

Now we have to convert each charatcer column to factor to implement machine learning algorithm in future.

Hide

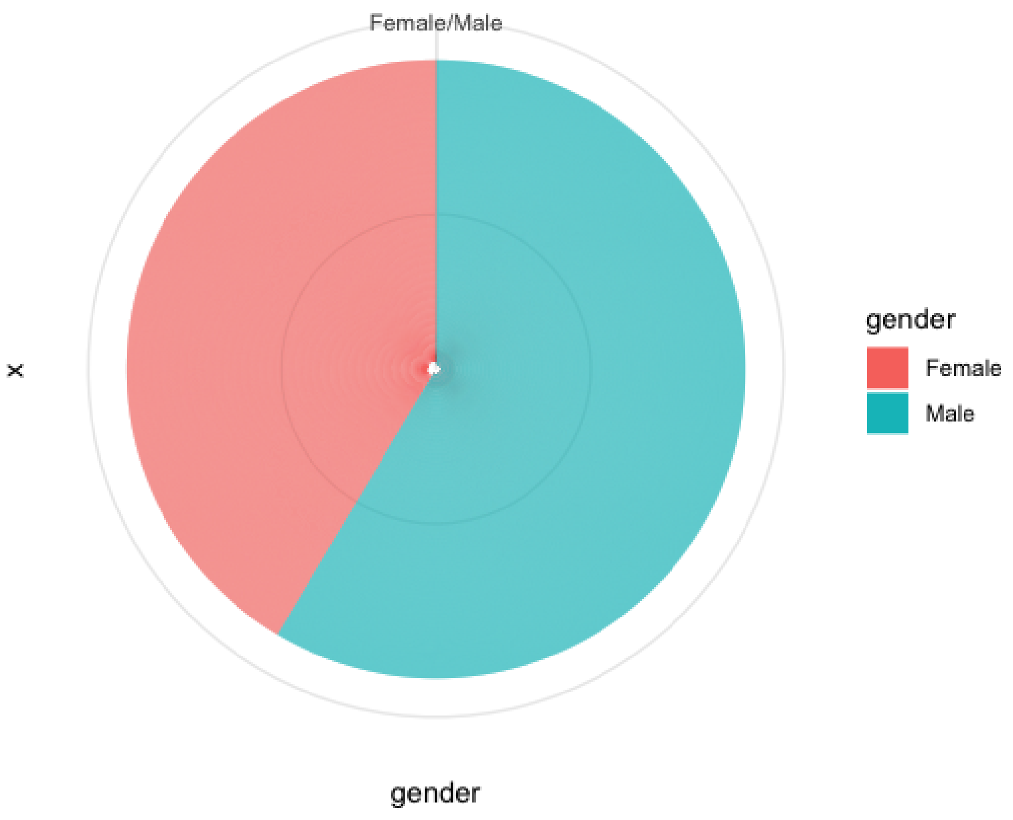
```
library(dplyr)

df_stroke <- df_stroke %>%
  mutate_if(is.character, as.factor)

str(df_stroke)
```

```
'data.frame':	5109 obs. of  12 variables:
 $ id               : int  9046 51676 31112 60182 1665 56669 53882 10434 27419 60491
...
 $ gender           : Factor w/ 2 levels "Female","Male": 2 1 2 1 1 2 2 1 1 1 ...
 $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
 $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
 $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
 $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
 $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 5 4 4 5 4 4 4 4
4 ...
 $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...
 $ avg_glucose_level: num  229 202 106 171 174 ...
 $ bmi              : num  36.6 28.9 32.5 34.4 24 ...
 $ smoking_status   : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4
...
 $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
```

Now we will generate some graphs:

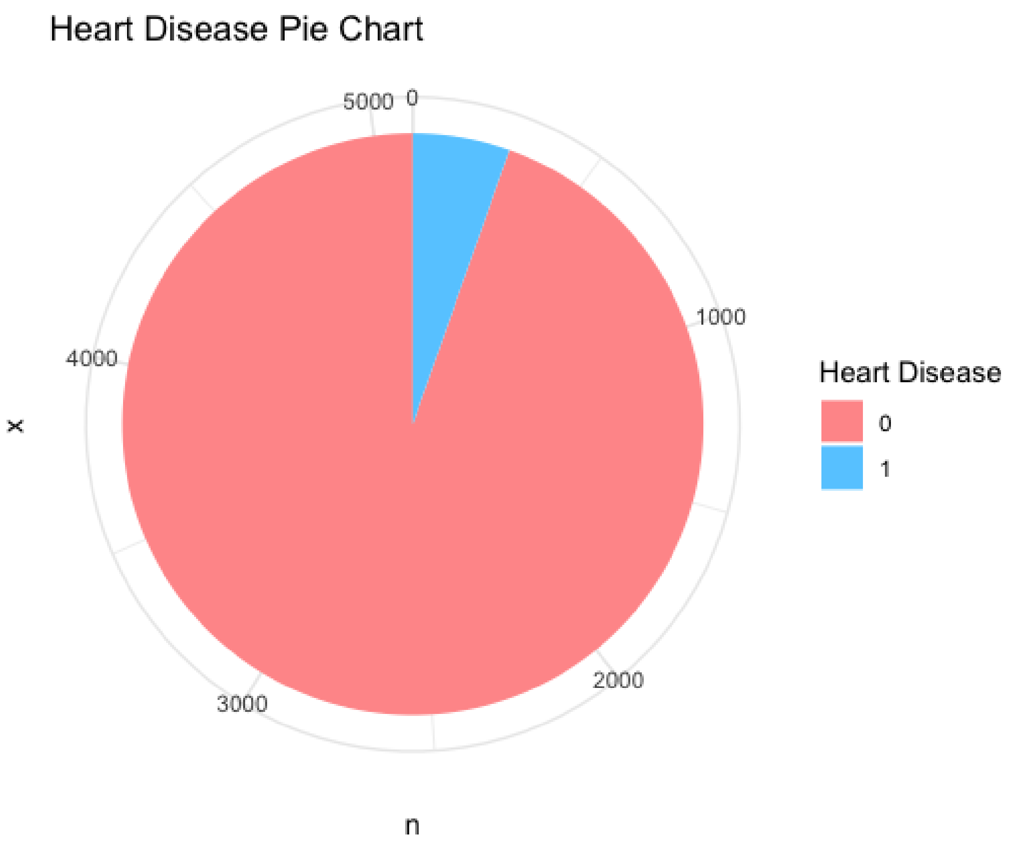
Hide

```
ggplot(df_stroke, aes(x= "", y = gender, fill = gender)) +geom_bar(stat = "identity",
width = 1)+
  coord_polar("y", start = 0)+theme_minimal()
```



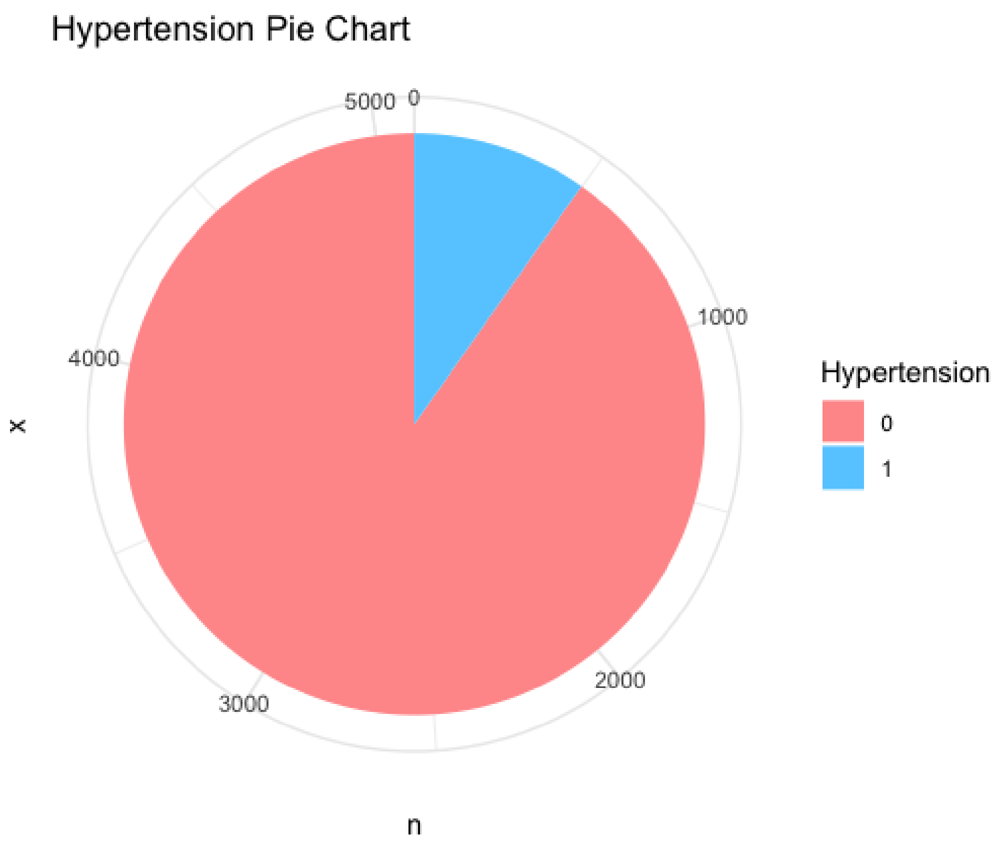
Hide

```
# calculate frequency of heart disease values
heart_disease <- df_stroke %>%
  count(heart_disease)

# create pie chart
ggplot(heart_disease, aes(x = "", y = n, fill = factor(heart_disease)))+
  geom_bar(stat = "identity", width = 1)+
  coord_polar("y", start = 0)+
  theme_minimal()+
  labs(fill = "Heart Disease")+
  scale_fill_manual(values = c("#FF9999", "#66CCFF"))+
  ggtitle("Heart Disease Pie Chart")
```



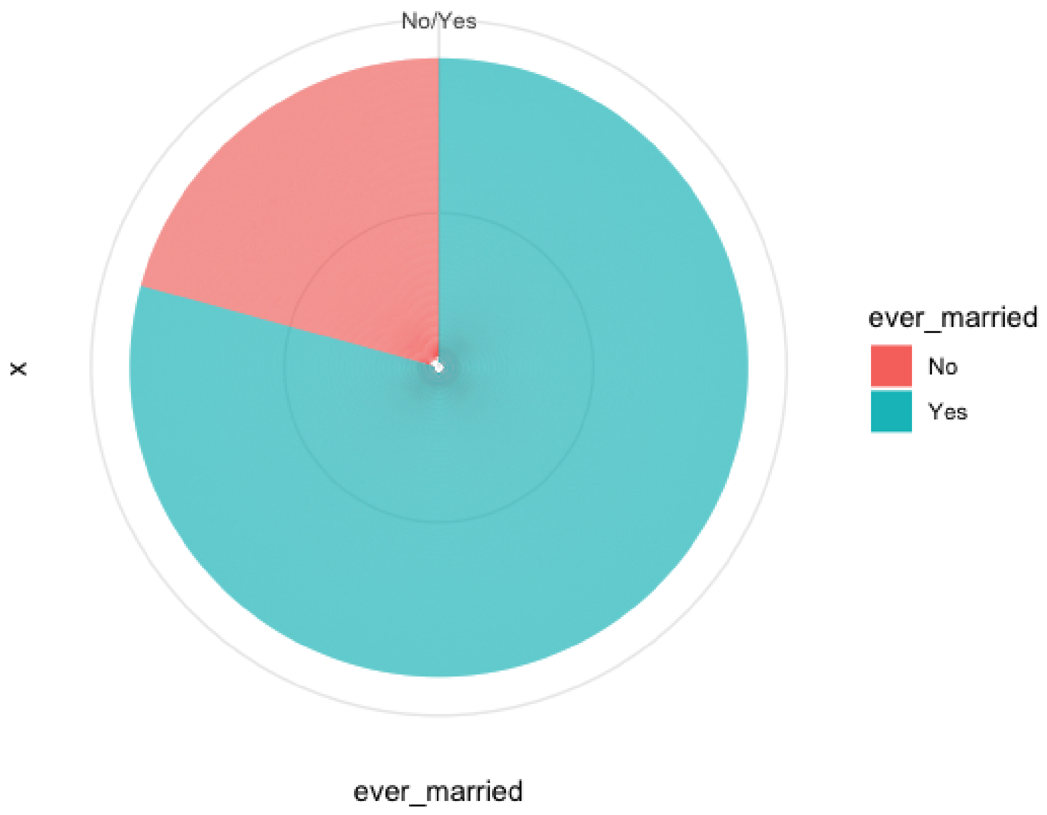
Hide

```
hypertension <- df_stroke %>%
  count(hypertension)

ggplot(hypertension, aes(x = "", y = n, fill = factor(hypertension)))+
  geom_bar(stat = "identity", width = 1)+
  coord_polar("y", start = 0)+
  theme_minimal()+
  labs(fill = "Hypertension")+
  scale_fill_manual(values = c("#FF9999", "#66CCFF"))+
  ggtitle("Hypertension Pie Chart")
```



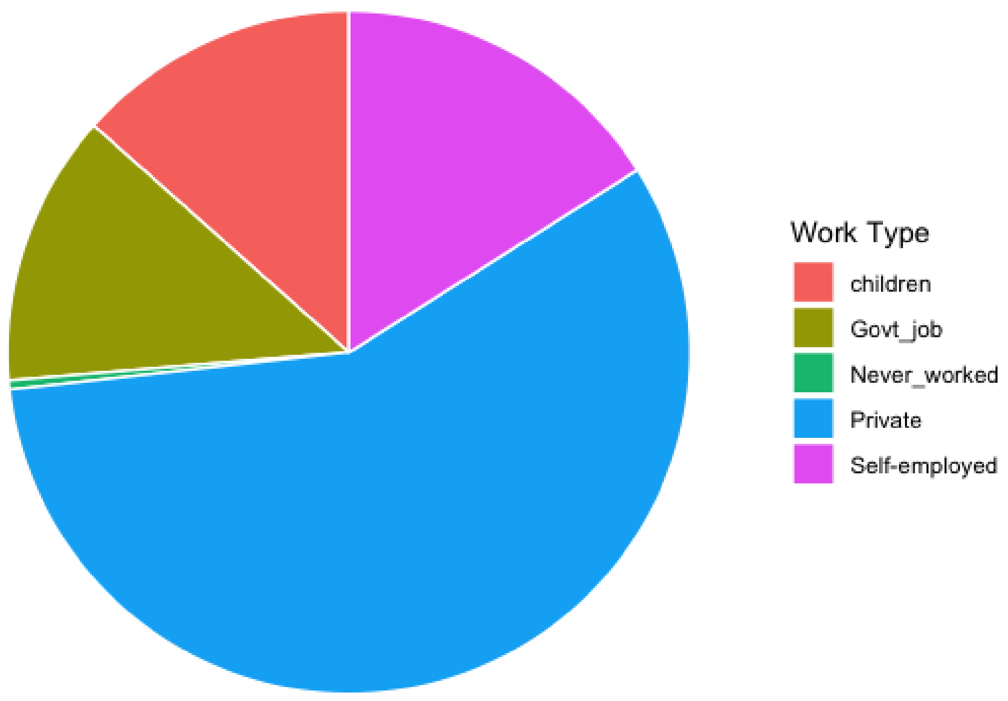
Hide

```
ggplot(df_stroke, aes(x = "", y = ever_married, fill = ever_married)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  theme_minimal()
```



Hide

```
ggplot(df_stroke, aes(x = "", fill = work_type)) +
  geom_bar(width = 1, color = "white") +
  coord_polar(theta = "y") +
  theme_void() +
  labs(fill = "Work Type")
```



Model Building and Prediction. Let’s split the final data set into training and test data set.

Hide

```
n_obs<- nrow(df_stroke)
split<- round(n_obs * 0.7)
train<- df_stroke [1:split,]
```

Variable called n_obs and assigns it the value of the number of rows in the df_stroke dataset. ‘split’ assigns it the value of 70% of the total number of observations rounded to the nearest whole number. Last line creates a subset of the df_stroke dataset called train by selecting the rows from 1 to split.

Now we will create test datatset

Hide

```
test<- df_stroke[(split +1): nrow(df_stroke),]
```

Hide

```
dim(train)
```

```
[1] 3576   12
```

Hide

```
dim(test)
```

```
[1] 1533   12
```

Above lines print the dimensions (number of rows and columns) of the train and test datasets.

Hide

```
train$stroke <- as.factor(train$stroke)
test$stroke <- as.factor(test$stroke)
```

Above lines convert the stroke variable in both the train and test datasets to a categorical factor variable.

Modeling

We use Random Forest algorithm for this problem as it is normally used in supervised learning since our problem has only two possible outcomes.

Hide

```
rf_model<-randomForest(formula= stroke~.,data = train)
rf_model
```

```


Call:
 randomForest(formula = stroke ~ ., data = train) 
              Type of random forest: classification
                    Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 7.21%
Confusion matrix:
     0 1 class.error
0 3318 9  0.00270514
1  249 0  1.00000000
```

Out-of-Bag (OOB) estimate of error rate (7.13%), the number of trees (500), the variables at each split (3), and the function used to build the classifier (randomForest). We must evaluate the model's performance on similar data once trained on the training set. We will make use of the test dataset for this. Let us print the confusion matrix to see how our classification model performed on the test data –Check levels of stroke in train and test datasets

Hide

```
levels(train$stroke)
```

```
[1] "0" "1"
```

Hide

```
levels(test$stroke)
```

```
[1] "0"
```

Ensure that both datasets have the same levels for stroke factor variable.

Hide

```
test$stroke <- factor(test$stroke, levels = levels(train$stroke))
```

Our Final Code to see how model is performing on the test dataset.

Hide

```
confusionMatrix(predict(rf_model, test), test$stroke)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1532    0
         1    1    0

               Accuracy : 0.9993
                 95% CI : (0.9964, 1)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9993
            Specificity :     NA
         Pos Pred Value :     NA
         Neg Pred Value :     NA
             Prevalence : 1.0000
         Detection Rate : 0.9993
   Detection Prevalence : 0.9993
      Balanced Accuracy :     NA

       'Positive' Class : 0

```

Conclusions:

We can see that the accuracy is nearly 100% with a validation dataset, suggesting that the model was trained well on the training data.

The confusion matrix shows the performance of the random forest model on the test dataset. The rows correspond to the predicted classes (0 and 1) and the columns correspond to the actual classes.

The confusion matrix shows that out of 1533 instances of class 0, the model correctly predicted all of them as class 0. However, out of 2 instances of class 1, the model incorrectly predicted them as class 0.

The accuracy of the model is calculated as (number of correct predictions)/(total number of predictions), which in this case is (1531+0)/(1531+0+2+0) = 0.9987, or 99.87%. This means that the model is very accurate at predicting the absence of stroke (class 0), but not very good at predicting the presence of stroke (class 1).

The other statistics in the confusion matrix such as Sensitivity, Specificity, Pos Pred Value, and Neg Pred Value are not calculated because there are no true positives, true negatives, false positives, or false negatives for class 1.