Analysis and Classification of Stroke Data

Erick Castillo
Math 445
Dr. Adriano Zambom
4/20/2021

Abstract

This study investigates the relationship between 12 variables and the occurrence of a cerebrovascular accident—colloquially known as a stroke, using data available for download on Kaggle. Construction of a successful classification model is pursued to determine whether an individual with specific characteristics will have a stroke. This analysis reveals a strong relationship between stroke and factors such as increasing age, high average glucose levels, pre-existing heart disease, and pre-existing hypertension. The success of the models was mixed—however, quadratic discriminant analysis and logistic regression showed the best results in terms of models with potent predictive power.

Introduction

This section briefly summarizes

- The nature of the unmodified data,
- Methods for cleaning the data,
- Methods of imputing missing data,
- Objectives.

The Raw Data

The dataset can be found on Kaggle[1]—it is simply titled 'Stroke Prediction Dataset'. The data was cleaned utilizing Python and its Pandas library on Google Colaboratory.

The raw dataset has 5110 rows and 12 columns: 8 of which are categorical, the rest being continuous. The columns in the raw dataset include:

- ID: continuous, discrete.
- Gender: categorical, binary.
- Age: continuous, discrete.
- Hypertension: categorical, binary.
- Heart disease: categorical, binary.
- Ever married: categorical, binary.
- Work type: categorical, five categories.
- Residence: categorical, binary.
- Average glucose levels: continuous.
- BMI: continuous.
- Smoking status: categorical, four categories.
- Stroke: categorical, binary.

Cleaning the Data

I immediately dropped the ID column as it won't be necessary for the analysis. Because this cleaning is being done using Python software, the predictor columns with strings were converted to zeros and ones; this includes hypertension, heart disease, ever married, and residence. In this case, one indicates a success—that is the patient has hypertension or heart disease, the patient was/is married, or the patient lives in an Urban region.

[1] fedesoriano, "Stroke Prediction Dataset," Kaggle (Kaggle, January 26, 2021), https://www.kaggle.com/fedesoriano/stroke-prediction-dataset.

Gender also received the same treatment, where one indicates the patient is a male. The single "Other" variable present in the column was converted to a zero as females were the most present in that set.

One-hot encoding of the data was necessary to perform any regression imputation of missing values. This was done using Pandas on only one of the categorical columns, work type. For the other categorical column, smoking status, values titled "Unknown" were replaced with NA's. A preprocessing method known as Ordinal Encoding was used to convert the remaining non-null values into integers.

The information provided by the columns is mostly explained by their titles. After carefully exploring the data, we found 201 missing values in the BMI column and 1544 missing values for smoking status. These are the only values missing in the raw dataset.

The other columns have values which fall within a reasonable range and are complete.

Imputing the Missing Data

To preserve the entirety of the dataset, we imputed values for the missing data for the BMI and smoking status columns.

For the BMI column we used a method referred to as Stochastic Regression Imputation (SRI)[2]. We first split the data frame in two—where one part consists only of individuals who had a stroke, while the other part doesn't. The average BMI for each is 30.47 and 28.82 respectively. This indicates that individuals who had a stroke had slightly higher BMIs than their counterparts. To preserve this discrepancy, I then performed SRI on each of these separate datasets. This involved creating an Ordinary Least Squares (OLS) model with BMI as a response variable, and every other column—except smoking status because of its missing values, as predictor variables. Once the model is created, numbers from a normal distribution are generated using the average of the predicted values (μ), and the residual standard deviation (σ).

For the smoking status column I used a method referred to as KNN Imputation[3]. This method uses the KNN algorithm to classify missing values based on the complete information available in the rest of the data frame. Note, because this column had 1544 missing values—about 30% of the total rows, the imputed values may be misleading. This issue will be further explored in the Data Analysis section of this paper.

---

[2] Adriano Zambom, "Math 445," *Math 445* (March 29, 2021).

[3] Kyaw Saw Htoon, "A Guide To KNN Imputation," Medium (Medium, July 3, 2020), https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e.

Once the missing values were imputed, encoded values were reverted into their string form for further analysis in R.

The Objectives

With the cleaned data, there are three main objectives we seek to achieve:

1. Create a "best" logistic model.
2. Use a variety of classification models to predict the occurrence of a stroke.
3. Explore the data graphically to verify conclusions about the nature of strokes.

To create the "best" logistic model, we will use traditional methods such as measuring deviance analysis of deviance to eliminate unnecessary predictors from the model, then the model with the smallest AIC will be the best model.

Classification models used include a probit-link logistic model QDA, LDA, and random forest.

Graphs including density plots, bar charts, and pie charts will be used to validate the final best logistic model and predictors used in the classification model.

Methodology

Logistic Regression

One of the many types of models classified under the umbrella term generalized linear models (GLM). This regression type is used to model binary response variables. For this particular data, the "Probit" link produced the smallest AIC when compared to other links such as "ClogLog" and "Logit". Probit uses the function $g(p_i) = \boldsymbol{\Phi}^{-1}(p)$ as a link to connect to the recognizable regression function $g(p_i) = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 X_{i1}$— where $\boldsymbol{\Phi}$ is the CDF of the normal distribution[4]. The goal in selecting the best logistic regression model is finding the model with the smallest possible AIC while simultaneously exploring the analysis of deviance of non-significant predictors.

Classifying with Logistic Regression

For classification of this type, and subsequent forms which will be explained, the data must be split into training and test sets. In this case a "best" logistic model is built with the training set, and response predictions can be generated. Response predictions in this case are best described as probabilities[5]. A function can be created to sort through these probabilities. For example,

[4] "Probit Regression." IDRE Stats. UCLA. Accessed April 22, 2021. https://stats.idre.ucla.edu/stata/dae/probit-regression/.

[5] Soner Yildirim, "How Is Logistic Regression Used as A Classification Algorithm?," Medium (Towards Data Science, May 3, 2020), https://towardsdatascience.com/how-is-logistic-regression-used-as-a-classification-algorithm-51eaf0d01a78.

"If the response prediction is ≥ 0.5, classify the prediction as a success (1), otherwise it's a failure (0)."

In general, after predictions are generated for any classification model, its validity is tested by checking if the predictions fall in line with the actual values.

Linear Discriminant Analysis

A classification model that does not assume if the information in the columns of the dataframe are normal; however, it does assume that the population covariances are equal[6]. That is, for populations $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$, $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. To classify the data into either $\boldsymbol{\pi}_1$ or $\boldsymbol{\pi}_2$, the expected cost of misclassification (ECM) must be minimized. The ECM is given by

$$ECM = c(2|1)P(2|1)p_1 + c(1|2)P(1|2)p_2$$

Where

- $c(2|1)$ and $c(1|2)$ is the cost of misclassifying the data
- $P(2|1)$ and $P(1|2)$ is the probability of misclassifying the data.
- p_1 and p_2 are the proportion of the data that fall into $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ respectively.

The minimum ECM can be described by the following inequalities:

$$R_1: \frac{f_1(x)}{f_2(x)} \geq \frac{c(1|2)}{c(2|1)}\left(\frac{p_2}{p_1}\right)$$
$$R_2: \frac{f_1(x)}{f_2(x)} < \frac{c(1|2)}{c(2|1)}\left(\frac{p_2}{p_1}\right)$$

It is important to note LDA does not assume the distributions are normal—when using Fisher's approach[7].

Quadratic Discriminant Analysis

Another classification model used for this dataset. Unlike LDA, the populations $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$are described by multivariate normal densities that do not assume equal covariances[8]. Let $f_1(x) \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $f_2(x) \sim N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ describe the densities of the populations—then $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$. The minimum ECM is the same as LDA's, but because the population densities are assumed to be

[6] Adriano Zambom, "Classification," *Classification* (March 2021).
[7] Ibid.
[8] Ibid.

normal, definitive regions can be created. The minimum ECM in this case is described by the following inequalities:

$$R_1: -\frac{1}{2}\mathbf{x}'(\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1})\mathbf{x} + (\boldsymbol{\mu}_1'\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_2'\boldsymbol{\Sigma}_2^{-1})\mathbf{x} - k \geq ln[\frac{c(1|2)}{c(2|1)}\frac{p_2}{P_1}]$$

$$R_2: -\frac{1}{2}\mathbf{x}'(\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1})\mathbf{x} + (\boldsymbol{\mu}_1'\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_2'\boldsymbol{\Sigma}_2^{-1})\mathbf{x} - k < ln[\frac{c(1|2)}{c(2|1)}\frac{p_2}{P_1}]$$

Random Forests

The final classification method utilized in this analysis. In essence, a random forest is a non-parametric classification model pertaining to the Bagging category. The way the algorithm works is it randomly samples the training set with replacement a set amount of times (N). One of these sets of samples from the training set is called a tree. At this point, predictions can be made—this is done by taking these trees individually and passing the observation for which we want to make a prediction through. A prediction can be extracted from every tree, then an overall aggregated prediction can be obtained. These predictions can then be aggregated using either the mean or the mode[9].

There are methods to test the efficiency of the classification methods described above. These will be discussed in the Data Analysis section of the report.

Data Analysis

The Best Logistic Model

See Figure 1. The model was obtained through an iterative process where non-significant predictors were removed. The full model would then be tested along with the reduced model using analysis of deviance (AOD) techniques. This eliminated the following four predictors:

- Gender
- Residence
- Ever married
- BMI

Remaining predictors that were not significant were categorical. When analysis of deviance was performed on them, there was strong evidence to suggest they provided reliable explanatory power to the model. AOD results for work type and smoking status can be seen in Figures 2 and 3 respectively. The final model in Figure 1 had the lowest attainable AIC.

---

[9] Tony Yiu, "Understanding Random Forest," Medium (Towards Data Science, August 14, 2019), https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

The resulting logistic model with Probit link can be interpreted as follows:

- For a one unit increase in age, the Z-score increases by 0.0322.
- A patient with hypertension has a Z-score increase by a factor of 0.188.
- A patient with heart disease has a Z-score increase by a factor 0.179.
- A one unit increase in a patient's average glucose level increases their Z-score by a factor of 0.0022.
- For smoking status:
    - A patient classified as a nonsmoker has a decrease in Z-score by a factor of 0.211.
    - A patient classified as a regular smoker has a decrease in Z-score by a factor of 0.161.
- For work type:
    - A patient working a government job has an increase in Z-score of 1.242.
    - A patient who never worked before has a Z-score decrease of -2.50.
    - A patient working for a private organization has a Z-score increase of 1.254.
    - A self-employed patient has a Z-score increase of 1.434.

Note that any individual interpretation provided above explains a change when holding all other variables constant. An increase in a Z-score means that the event of a stroke is more likely.

Success of Classification Models

A train-test split of 90%—10% was performed on the dataset using seed 505. A split of this nature was necessary as less than 6% of the patients had a stroke.

To begin, a total of three different logistic models were created using the training set:

1. All predictors present.
2. All predictors except smoking status are present.
3. All predictors except gender, BMI, residence, and ever married are present.

All three models performed similarly. Each of their respective matrices can be viewed in Figures 4, 5, and 6 respectively. The accuracy of each of these models stays at about the same amount, in the lower 90% range.

The next model was LDA. The predictors included were smoking status, age, average glucose level, and heart disease. These specific predictors were selected because they were the most significant variables in the final logistic model that did not cause R to execute a warning. The confusion matrix for this model can be seen in Figure 7. The accuracy of this model increased in comparison to the different logistic models; however, this model had a more difficult time correctly identifying patients who had a stroke. It had a tendency to mostly classify individuals as not having strokes.

The QDA model used the same predictors as the LDA model for the same reasons. The results of its confusion matrix can be seen in Figure 8. The accuracy of this model is somewhat smaller than that seen in the LDA model; however, unlike the LDA model, QDA was not afraid to make more stroke predictions. It scored more correct stroke predictions when compared to LDA, but less when compared to the Logistic Models.

The final model was the Random Forest, and like QDA and LDA models, it used the same predictors due to warnings R would generate otherwise. This confusion matrix can be seen in Figure 9. The accuracy of this model is very much comparable with that seen in LDA's. LDA was able to at least correctly predict one of the patients who did have a stroke, while Random Forest had a hard time getting any of the actual stroke predictions right. LDA and Random Forest may be equal in accuracy, but that one correct stroke prediction puts LDA miles above Random Forest in terms of predictive power.

The Data Speaking for Itself

Given all the results above, we thought a validation of the logistic/classification models would be necessary through graphical interpretation.

Figure 10 provides reasonably strong graphical evidence that as age increases, the possibility of stroke increases with it. Notice that the sampled individuals that did not have a stroke were mostly below the age of 50.

Figure 11 depicts stroke patients were more likely to have higher than average glucose levels, while their counterparts were on the lower end of glucose levels.

Both Figures 15 and 16 indicate that stroke patients were more likely to have pre-existing hypertension and heart disease.

These 4 figures fall in line with the best logistic model, and indicate significance in their presence.

Notice that the figures for BMI (Figure 12), work type (Figure 13), and smoking status (Figure 14) all appear to not have a significant relationship with the patient having a stroke or not. Recall that smoking status was imputed utilizing a KNN imputer, meaning that the data may not be reliable. Due to the method that BMI was imputed, the differences between stroke and non-stroke patients have been accentuated; because of this, it is safe to conclude that BMI is not significant in determining the probability of a patient having a stroke.

Figure 17 displays a 2×2 contingency table that groups patients by the occurence of stroke and gender. The calculated odds ratio confidence interval indicates there is weak evidence to suggest gender plays a role in a patient having a stroke.

Conclusion

There are predictors in this dataset which are essential in predicting whether a patient will have a stroke. These predictors include age, average glucose level, pre-existing hypertension, and pre-existing heart disease. The utility of the other predictors is dubious.

Models that performed the best included QDA and Logistic Models. These should be considered for predicting the occurrence of a stroke for additional patients.

The graphs produced agree with the predictors found in the best logistic model and the variables present in the constructed predictive models.

Overall, improvements could be implemented to the Logistic Model and the Classification Models with the addition of variables such as cholesterol levels, blood pressure levels, average alcohol consumption in a week, drug use, amount of exercise a week, etc.[10]

[10] "Stroke Risk Factors and Prevention," Better Health Channel (Better Health Channel), accessed April 22, 2021, https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/stroke-risk-factors-and-prevention.

Appendix

Bibliography

Fedesoriano, fedesoriano. “Stroke Prediction Dataset.” Kaggle. Kaggle, January 26, 2021. https://www.kaggle.com/fedesoriano/stroke-prediction-dataset.

Htoon, Kyaw Saw. “A Guide To KNN Imputation.” Medium. Medium, July 3, 2020. https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e.

“Probit Regression.” IDRE Stats. UCLA. Accessed April 22, 2021. https://stats.idre.ucla.edu/stata/dae/probit-regression/.

“Stroke Risk Factors and Prevention.” Better Health Channel. Better Health Channel. Accessed April 22, 2021. https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/stroke-risk-factors-and-prevention.

Yildirim, Soner. “How Is Logistic Regression Used as A Classification Algorithm?” Medium. Towards Data Science, May 3, 2020. https://towardsdatascience.com/how-is-logistic-regression-used-as-a-classification-algorithm-51eaf0d01a78.

Yiu, Tony. “Understanding Random Forest.” Medium. Towards Data Science, August 14, 2019. https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Zambom, Adriano. “Missing Values.” *Math 445*. Lecture presented at the Math 445, March 29, 2021.

———. “Classification.” *Math 445*. Lecture presented at the Classification, March 2021.

Images

```
> summary(bestmod1)

Call:
glm(formula = stroke ~ age + hypertension + heart_disease + avg_glucose_level +
    smoking_status + work_type, family = binomial(link = "probit"),
    data = df1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1553  -0.3251  -0.1457  -0.0474   4.1854

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                -4.833368   0.326463 -14.805  < 2e-16 ***
age                         0.032245   0.002455  13.132  < 2e-16 ***
hypertension                0.188482   0.088250   2.136 0.032697 *
heart_disease               0.178869   0.105428   1.697 0.089774 .
avg_glucose_level           0.002204   0.000625   3.527 0.000421 ***
smoking_statusnever smoked -0.210805   0.083128  -2.536 0.011216 *
smoking_statussmokes       -0.161312   0.107555  -1.500 0.133665
work_typeGovt_job           1.242236   0.282466   4.398 1.09e-05 ***
work_typeNever_worked      -2.502890  64.320297  -0.039 0.968960
work_typePrivate            1.254264   0.269196   4.659 3.17e-06 ***
work_typeSelf-employed      1.434037   0.275491   5.205 1.94e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1990.4  on 5109  degrees of freedom
Residual deviance: 1523.9  on 5099  degrees of freedom
AIC: 1545.9

Number of Fisher Scoring iterations: 14
```

Figure 1: The best logistic model given the data.

```
> anova(testmod1, bestmod1, test = 'LRT')
Analysis of Deviance Table

Model 1: stroke ~ age + hypertension + heart_disease + avg_glucose_level +
    smoking_status
Model 2: stroke ~ age + hypertension + heart_disease + avg_glucose_level +
    smoking_status + work_type
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1      5103     1578.7
2      5099     1523.9  4   54.814 3.554e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 2: Analysis of Deviance of work type

```
> anova(testmod2, bestmod1, test = 'LRT')
Analysis of Deviance Table

Model 1: stroke ~ age + hypertension + heart_disease + avg_glucose_level +
    work_type
Model 2: stroke ~ age + hypertension + heart_disease + avg_glucose_level +
    smoking_status + work_type
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      5101     1530.2
2      5099     1523.9  2     6.36  0.04158 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 3: Analysis of Deviance of smoking status

```
> confusionMatrix(factor(binary.pred), factor(test$stroke))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 453  14
         1  37   7

               Accuracy : 0.9002
                 95% CI : (0.8709, 0.9248)
    No Information Rate : 0.9589
    P-Value [Acc > NIR] : 1.000000

                  Kappa : 0.1692

 Mcnemar's Test P-Value : 0.002066

            Sensitivity : 0.9245
            Specificity : 0.3333
         Pos Pred Value : 0.9700
         Neg Pred Value : 0.1591
             Prevalence : 0.9589
         Detection Rate : 0.8865
   Detection Prevalence : 0.9139
      Balanced Accuracy : 0.6289

       'Positive' Class : 0
```

Figure 4: Confusion Matrix for Logistic Model with all predictors.

```
> confusionMatrix(factor(binary.pred), factor(test$stroke))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 459  15
         1  31   6

               Accuracy : 0.91
                 95% CI : (0.8818, 0.9333)
    No Information Rate : 0.9589
    P-Value [Acc > NIR] : 1.00000

                  Kappa : 0.163

 Mcnemar's Test P-Value : 0.02699

            Sensitivity : 0.9367
            Specificity : 0.2857
         Pos Pred Value : 0.9684
         Neg Pred Value : 0.1622
             Prevalence : 0.9589
         Detection Rate : 0.8982
   Detection Prevalence : 0.9276
      Balanced Accuracy : 0.6112

       'Positive' Class : 0
```

Figure 5: Confusion Matrix for Logistic Model without smoking status.

```
> confusionMatrix(factor(binary.pred), factor(test$stroke))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 456  15
         1  34   6

               Accuracy : 0.9041
                 95% CI : (0.8752, 0.9282)
    No Information Rate : 0.9589
    P-Value [Acc > NIR] : 1.00000

                  Kappa : 0.151

 Mcnemar's Test P-Value : 0.01013

            Sensitivity : 0.9306
            Specificity : 0.2857
         Pos Pred Value : 0.9682
         Neg Pred Value : 0.1500
             Prevalence : 0.9589
         Detection Rate : 0.8924
   Detection Prevalence : 0.9217
      Balanced Accuracy : 0.6082

       'Positive' Class : 0
```

Figure 6: Confusion Matrix for Logistic Model without BMI, residence, ever married, and gender.

```
> confusionMatrix(as.factor(lda.pred$class), as.factor(test$stroke))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 488  20
         1   2   1

               Accuracy : 0.9569
                 95% CI : (0.9355, 0.9728)
    No Information Rate : 0.9589
    P-Value [Acc > NIR] : 0.6422233

                  Kappa : 0.0738

 Mcnemar's Test P-Value : 0.0002896

            Sensitivity : 0.99592
            Specificity : 0.04762
         Pos Pred Value : 0.96063
         Neg Pred Value : 0.33333
             Prevalence : 0.95890
         Detection Rate : 0.95499
   Detection Prevalence : 0.99413
      Balanced Accuracy : 0.52177

       'Positive' Class : 0
```

Figure 7: LDA Confusion Matrix.

```
> confusionMatrix(as.factor(qda.pred$class), as.factor(test$stroke))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 466  17
         1  24   4

               Accuracy : 0.9198
                 95% CI : (0.8927, 0.9418)
    No Information Rate : 0.9589
    P-Value [Acc > NIR] : 1.0000

                  Kappa : 0.122

 Mcnemar's Test P-Value : 0.3487

            Sensitivity : 0.9510
            Specificity : 0.1905
         Pos Pred Value : 0.9648
         Neg Pred Value : 0.1429
             Prevalence : 0.9589
         Detection Rate : 0.9119
   Detection Prevalence : 0.9452
      Balanced Accuracy : 0.5707

       'Positive' Class : 0
```

Figure 8: QDA Confusion Matrix.

```
> confusionMatrix(as.factor(prediction_for_table), as.factor(rfset_test$stroke))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 489  21
         1   1   0

               Accuracy : 0.9569
                 95% CI : (0.9355, 0.9728)
    No Information Rate : 0.9589
    P-Value [Acc > NIR] : 0.6422

                  Kappa : -0.0037

 Mcnemar's Test P-Value : 5.104e-05

            Sensitivity : 0.9980
            Specificity : 0.0000
         Pos Pred Value : 0.9588
         Neg Pred Value : 0.0000
             Prevalence : 0.9589
         Detection Rate : 0.9569
   Detection Prevalence : 0.9980
      Balanced Accuracy : 0.4990

       'Positive' Class : 0
```

Figure 9: Random Forest Confusion Matrix.

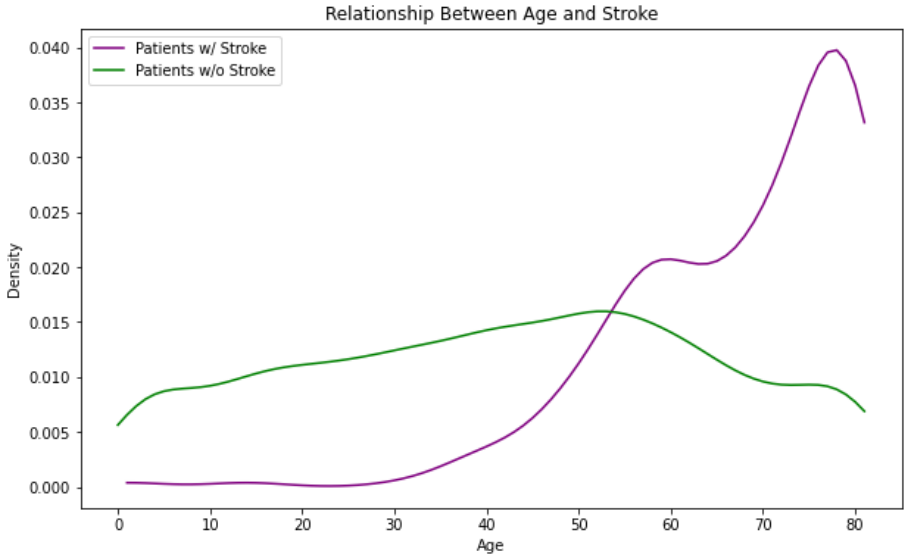

Figure 10: Density plot of age against stroke.

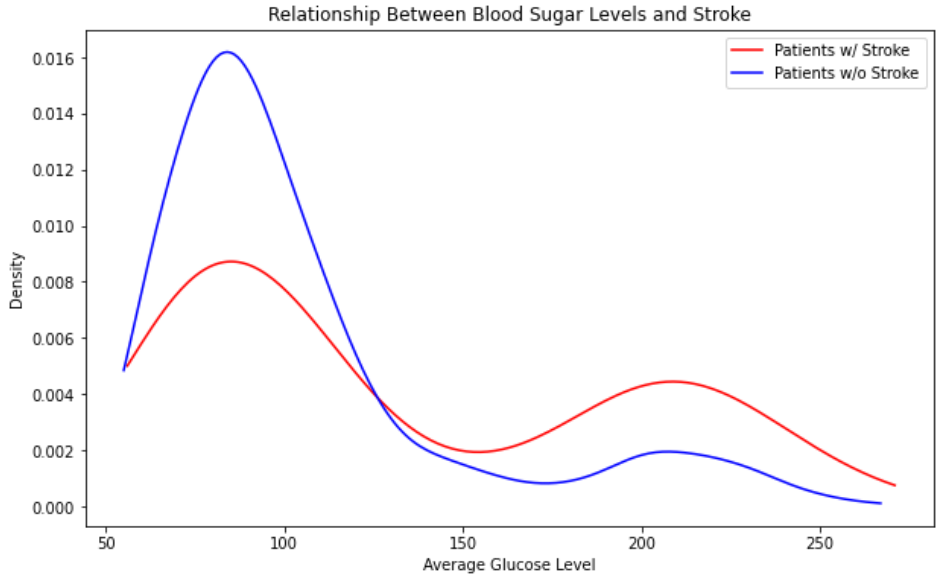

Figure 11: Density plot of average glucose levels against stroke.

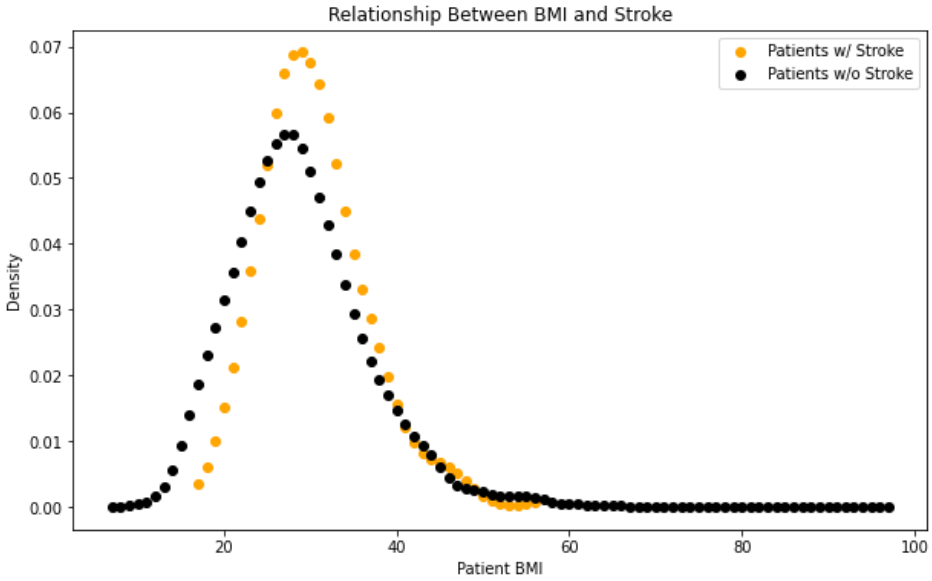

Figure 12: Density plot of BMI against stroke.

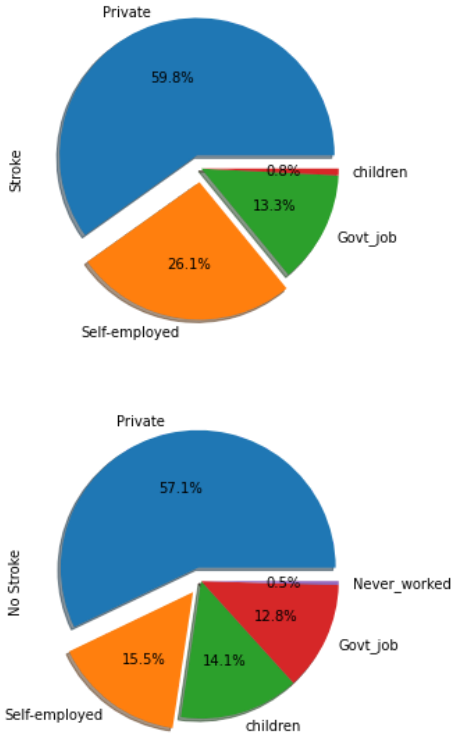

Figure 13: Work type against stroke

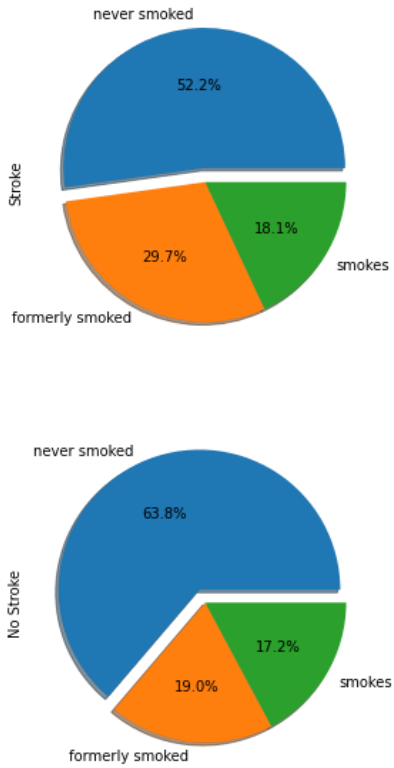

Figure 14: Smoking status and stroke.

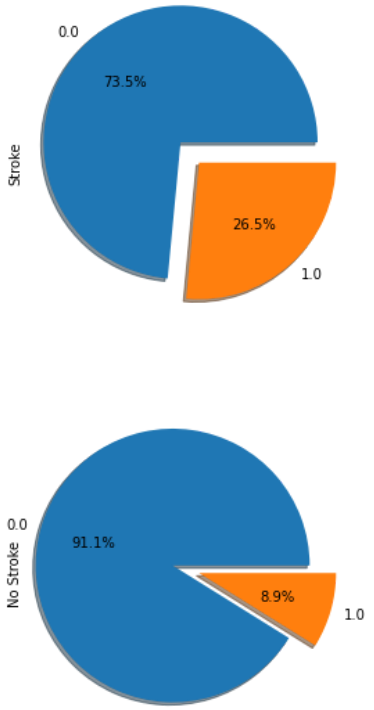

Figure 15: Hypertension and stroke.

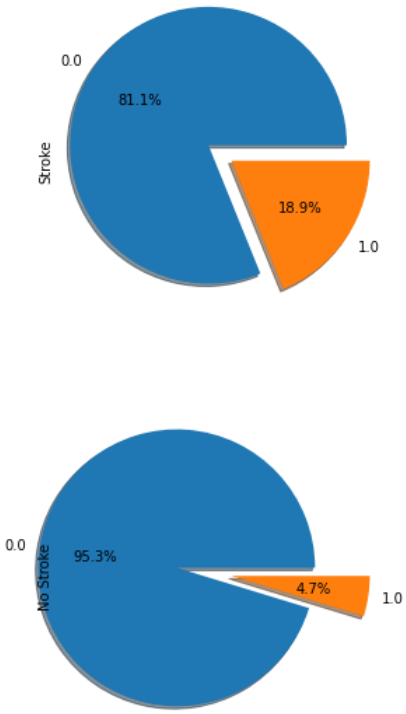

Figure 16: Heart disease and stroke.

```
tab1 = pd.crosstab(final_data.stroke, final_data.gender)
tab1
```

gender	0.0	1.0
stroke		
0.0	2854	2007
1.0	141	108

```
OR, pval = stats.fisher_exact(tab1)
print('The odds ratio is',OR,'. While the p-value is',pval)
```

```
The odds ratio is 1.0892090449384602 . While the p-value is 0.5104574776018146
```

```
seOR = np.sqrt(1/2854+1/141+1/2007+1/108)
confint = 1.96
print('(',np.exp(np.log(OR)-seOR*confint),',', np.exp(np.log(OR)+seOR*confint),')')
```

```
( 0.8423155717205929 , 1.4084701546622826 )
```

Figure 17: Contingency Tables.

Python Code

https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

"""

```
from google.colab import files
files.upload()

#important stuff. always add
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

#download the dataset
!kaggle datasets download -d fedesoriano/stroke-prediction-dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statistics as stats
from scipy.stats import trim_mean as tm
import plotly.express as px
from sklearn import linear_model
import random
from fancyimpute import KNN
from sklearn.preprocessing import OrdinalEncoder
from scipy import stats

stroke = pd.read_csv('/content/stroke-prediction-dataset.zip')
print(stroke.shape)
stroke[:3]

stroke = stroke.drop(columns = 'id')
stroke.index = range(len(stroke))

#cleaning the gender column
stroke.gender = [1 if i == 'Male' else 0 for i in stroke.gender] #male is 1, female 0

#change age from float to int type
stroke.age = [int(round(i)) for i in stroke.age]

#cleaning residence
stroke.Residence_type = [1 if i == 'Urban' else 0 for i in stroke.Residence_type]

#cleaning martial status
stroke.ever_married = [1 if i == 'Yes' else 0 for i in stroke.ever_married]
```

```
one_smoke = pd.get_dummies(stroke.smoking_status, drop_first=True)
one_work = pd.get_dummies(stroke.work_type, drop_first=True)
stroke = stroke.join(one_smoke)
stroke = stroke.join(one_work)

stroke = stroke.drop(columns = ['work_type', 'smoking_status'])

stroke.head(3)

stroke.isna().sum() #bmi still missing 201 values.

had_stroke = stroke[stroke['stroke'] == 1]
no_stroke = stroke[stroke['stroke'] == 0]

print('mean bmi of patients who had stroke',round(had_stroke.bmi.mean(),2))
print('mean bmi of patients who did not have stroke',round(no_stroke.bmi.mean(),2))

avg_ages, str_stat = [round(had_stroke.age.mean()),round(no_stroke.age.mean())], ['stroke', 'no
stroke']
plt.bar(str_stat, avg_ages)

"""The following imputes missing values for bmi. Adds variability to the missing predictor val with
the use of a normal density. Does not use the values from the smoking status to predict."""

obs_stroke = had_stroke[(had_stroke.bmi.notna())] #complete values for individuals who had a
stroke.
miss_stroke = had_stroke[had_stroke.bmi.isna()] #missing values for people who had stroke.
obs_nostroke = no_stroke[no_stroke.bmi.notna()] #complete values for individuals who did not
have stroke.
miss_nostroke = no_stroke[no_stroke.bmi.isna()] #missing values for people who did not have a
stroke.

param = list(set(obs_stroke.columns) - set(['bmi', 'formerly smoked', 'smokes', 'never smoked',
'stroke']))
model1 = linear_model.LinearRegression()
model1.fit(X = obs_stroke[param], y = obs_stroke['bmi'])

model2 = linear_model.LinearRegression()
model2.fit(X = obs_nostroke[param], y = obs_nostroke['bmi'])

pred1 = model1.predict(obs_stroke[param])
pred2 = model2.predict(obs_nostroke[param])

std_error1 = (pred1 - obs_stroke.bmi).std()
```

```
std_error2 = (pred2 - obs_nostroke.bmi).std()

random.seed(505)
stoch_val1 = np.round(np.random.normal(size = miss_stroke.shape[0], loc = pred1.mean(),
scale = std_error1),1)
random.seed(505)
stoch_val2 = np.round(np.random.normal(size = miss_nostroke.shape[0], loc = pred2.mean(),
scale = std_error2),1)

miss_stroke.bmi = stoch_val1
miss_nostroke.bmi = stoch_val2

df_nomiss = pd.concat([obs_stroke, miss_stroke, obs_nostroke, miss_nostroke],
ignore_index=True)

"""I noticed that there are individuals in this dataframe that have BMI's that are just below 100. In history, the heaviest person ever recorded was 1000 lbs and had a height of 6'1". His BMI was 184.7. These recorded values don't seem too ridiculous LOL."""

stroke = pd.read_csv('/content/stroke-prediction-dataset.zip')
stroke[:3]

df_nomiss['smoking_status'] = stroke.smoking_status
df_nomiss = df_nomiss.drop(columns = ['formerly smoked', 'never smoked', 'smokes'])
df_nomiss[:3]

df_nomiss.smoking_status = [np.nan if i == 'Unknown' else i for i in df_nomiss.smoking_status]

df_nomiss.isna().sum()
df_nomiss.smoking_status.value_counts()

encoder = OrdinalEncoder()
imputer = KNN()
#only contains values which are not null
nonull = np.array(df_nomiss['smoking_status'].dropna())
#reshapes the data for encoding
impute_reshape = nonull.reshape(-1,1)
#encode data
impute_ordinal = encoder.fit_transform(impute_reshape)
#Assign back encoded values to non-null values
df_nomiss['smoking_status'].loc[df_nomiss['smoking_status'].notnull()] =
np.squeeze(impute_ordinal)

df_nomiss.smoking_status.value_counts()
```

```
#0 is formerly smoked
#1 is never smoked
#2 is smokes

final_data = pd.DataFrame(np.round(imputer.fit_transform(df_nomiss)),columns =
df_nomiss.columns)

final_data.isna().sum()

final_data[:6]

final_data = final_data.drop(columns = ['Never_worked', 'Private', 'Self-employed', 'children'])
final_data['work_type'] = stroke.work_type
final_data['age'] = [round(i) for i in final_data['age']]

final_data[:3]

smoke_strings = []
for i in final_data.smoking_status:
  if i == 0:
    smoke_strings.append('formerly smoked')
  if i == 1:
    smoke_strings.append('never smoked')
  if i == 2:
    smoke_strings.append('smokes')
final_data['smoking_status'] = smoke_strings

final_data[:6]

fin_stroke = final_data[final_data['stroke'] == 1]
fin_nostroke = final_data[final_data['stroke'] == 0]

stroke_age_density = stats.kde.gaussian_kde(fin_stroke.age)
x_int1 = np.arange(min(fin_stroke.age), max(fin_stroke.age), 1)
nostroke_age_density = stats.kde.gaussian_kde(fin_nostroke.age)
x_int2 = np.arange(min(fin_nostroke.age), max(fin_nostroke.age), 1)

plt.figure(figsize=(10,6))
plt.plot(x_int1, stroke_age_density(x_int1), c = 'purple', label = 'Patients w/ Stroke')
plt.plot(x_int2, nostroke_age_density(x_int2), c = 'green', label = 'Patients w/o Stroke')
plt.title('Relationship Between Age and Stroke')
plt.xlabel('Age')
plt.ylabel('Density')
plt.legend()
```

```
plt.show();

stroke_glucose_density = stats.kde.gaussian_kde(fin_stroke.avg_glucose_level)
x_int3 = np.arange(min(fin_stroke.avg_glucose_level), max(fin_stroke.avg_glucose_level), 1)
nostroke_glucose_density = stats.kde.gaussian_kde(fin_nostroke.avg_glucose_level)
x_int4 = np.arange(min(fin_nostroke.avg_glucose_level), max(fin_nostroke.avg_glucose_level),
1)

plt.figure(figsize=(10,6))
plt.plot(x_int3, stroke_glucose_density(x_int3), c = 'red', label = 'Patients w/ Stroke')
plt.plot(x_int4, nostroke_glucose_density(x_int4), c = 'blue', label = 'Patients w/o Stroke')
plt.title('Relationship Between Blood Sugar Levels and Stroke')
plt.xlabel('Average Glucose Level')
plt.ylabel('Density')
plt.legend()
plt.show();

stroke_bmi_density = stats.kde.gaussian_kde(fin_stroke.bmi)
x_int5 = np.arange(min(fin_stroke.bmi), max(fin_stroke.bmi), 1)
nostroke_bmi_density = stats.kde.gaussian_kde(fin_nostroke.bmi)
x_int6 = np.arange(min(fin_nostroke.bmi), max(fin_nostroke.bmi), 1)

plt.figure(figsize=(10,6))
plt.scatter(x_int5, stroke_bmi_density(x_int5), c = 'orange', label = 'Patients w/ Stroke')
plt.scatter(x_int6, nostroke_bmi_density(x_int6), c = 'black', label = 'Patients w/o Stroke')
plt.title('Relationship Between BMI and Stroke')
plt.xlabel('Patient BMI')
plt.ylabel('Density')
plt.legend()
plt.show();

fig, axs = plt.subplots(2, 1, figsize = (14,10))
fin_stroke.work_type.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[0], explode
= (0.1,0.1,0,0), shadow = True, label = 'Stroke')
fin_nostroke.work_type.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[1],
explode = (0.1,0.1,0,0,0), shadow =  True, label = 'No Stroke');

fig, axs = plt.subplots(2, 1, figsize = (14,10))
fin_stroke.smoking_status.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[0],
explode = (0.1,0,0), shadow = True, label = 'Stroke')
fin_nostroke.smoking_status.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[1],
explode = (0.1,0,0), shadow = True, label = 'No Stroke');

fig, axs = plt.subplots(2, 1, figsize = (14,10))
```

```
fin_stroke.hypertension.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[0],
shadow = True, label = 'Stroke', explode = (0.2,0))
fin_nostroke.hypertension.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[1],
shadow = True, label = 'No Stroke', explode = (0.2,0));

fig, axs = plt.subplots(2, 1, figsize = (14,10))
fin_stroke.heart_disease.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[0],
shadow = True, label = 'Stroke', explode = (0.2, 0))
fin_nostroke.heart_disease.value_counts().plot(kind = 'pie', autopct = '%1.1f%%', ax = axs[1],
shadow = True, label = 'No Stroke', explode = (0.4, 0));

tab1 = pd.crosstab(final_data.stroke, final_data.gender)
tab1

OR, pval = stats.fisher_exact(tab1)
print('The odds ratio is',OR,'. While the p-value is',pval)

seOR = np.sqrt(1/2854+1/141+1/2007+1/108)
confint = 1.96
print('(',np.exp(np.log(OR)-seOR*confint),',', np.exp(np.log(OR)+seOR*confint),')')

"""There is little evidence to suggest that gender and the event of having a stroke are categories
that are dependent on one another."""

final_data.to_csv(r'/content/imputed_stroke.csv', index = False)
```

R Code

```
#import both data sets.
df1 <- read.csv("~/M445_HW/project/imputed_stroke.csv") #values for BMI and smoking status
were imputed
df2 <- read.csv("~/M445_HW/project/stroke_miss.csv") #nas are present in the data
df2 = subset(df2, select = -c(id))

library(MASS)
library(mice)
library(caret)
library(tidyverse)
library(data.table)
library(randomForest)

df2$ss_ordinal <- as.factor(df2$ss_ordinal)
set.seed(505)
imp1 = mice(df2, m = 11, method = c('logreg','','','','','','','pmm','','lda'), maxit=75)
```

```
fitm1 <- with(imp1, glm(stroke~gender+age+hypertension+heart_disease+
                        ever_married+work_type+Residence_type+avg_glucose_level+bmi+
                        factor(ss_ordinal), family = binomial()))

#in this case, 0 is never smoked; 1 is used to smoke; 2 is smokes.
summary(pool(fitm1))

mod1 = glm(stroke~., data = df1, family = binomial(link='probit'))
summary(mod1)

# 90% of the sample size
smp_size <- floor(0.9 * nrow(df1))

# set the seed to make your partition reproducible
set.seed(505)
train_ind <- sample(seq_len(nrow(df1)), size = smp_size)

train <- df1[train_ind, ]
test <- df1[-train_ind, ]

#using ALL predictors to create a model
mod2 = glm(stroke~., data = train, family = binomial(link = 'probit'))
summary(mod2)

predvals = predict(mod2, newdata = test, type = 'response')
plot(test$stroke,predvals)

bb = lm(predvals~test$stroke)
abline(bb)

binary.pred = ifelse(predvals > 0.16, 1, 0)
confusionMatrix(factor(binary.pred), factor(test$stroke))

#getting rid of the smoking status to make predictions.
mod3 = update(mod2, .~. -smoking_status)
summary(mod3)

predvals = predict(mod3, newdata = test, type = 'response')
plot(test$stroke,predvals)

bb = lm(predvals~test$stroke)
abline(bb)

binary.pred = ifelse(predvals > 0.17, 1, 0)
```

```
confusionMatrix(factor(binary.pred), factor(test$stroke))

#dropping gender, bmi, residence, and marital status to make predictions.
mod4 = update(mod2, .~. -Residence_type -gender -bmi -ever_married)
summary(mod4)

predvals = predict(mod4, newdata = test, type = 'response')
plot(test$stroke,predvals)

bb = lm(predvals~test$stroke)
abline(bb)

binary.pred = ifelse(predvals > 0.17, 1, 0)
confusionMatrix(factor(binary.pred), factor(test$stroke))
#note that this has predictions that are just as good as the one with all the predictors.


#time to use other methods of classification.

#random forest
rfset_train = subset(train, select = -c(gender, work_type, ever_married, Residence_type, bmi))
rfset_test = subset(test, select = -c(gender, work_type, ever_married, Residence_type, bmi))
rf_classifier = randomForest(factor(stroke) ~ ., data=rfset_train, ntree=200, mtry=2,
importance=TRUE)
test_set_predictors = subset(rfset_test, select = -c(stroke))
prediction_for_table <- predict(rf_classifier, test_set_predictors)
confusionMatrix(as.factor(prediction_for_table), as.factor(rfset_test$stroke))

#qda
qda.mod = qda(stroke~age+avg_glucose_level+heart_disease+smoking_status, data = train)
qda.pred <- predict(qda.mod, test)
confusionMatrix(as.factor(qda.pred$class), as.factor(test$stroke))

#lda
lda.mod = lda((stroke~age+avg_glucose_level+heart_disease+smoking_status), data = train)
lda.pred <- lda.mod %>% predict(test)
confusionMatrix(as.factor(lda.pred$class), as.factor(test$stroke))

#create the best logistic model, starting from all the data.
summary(mod1)
bestmod1 = update(mod1, .~. -gender -Residence_type -ever_married -bmi)
summary(bestmod1)

testmod1 = update(bestmod1, .~. -work_type)
```

```
summary(testmod1)
anova(testmod1, bestmod1, test = 'LRT')

testmod2 = update(bestmod1, .~. -smoking_status)
summary(testmod2)
anova(testmod2, bestmod1, test = 'LRT')

#here is the overall best model!
summary(bestmod1)
```