CS-C3240 - Machine Learning D

Predicting Cases of Stroke using Logistic Regression and Multilayer Perceptron

1. Introduction

In a world where overindulging and unhealthy living habits are becoming more normal than ever, we can see an increase in obesity and diabetes. Both of these health conditions increase your risk of high blood pressure which combined with the earlier mentioned health concerns are some of the largest contributors to people having strokes [1]. We have in recent years seen an increase in especially young people having strokes. Just in the past decade can we see a 44% increase in young people being hospitalized due to strokes [2].

This report will attempt to predict if a certain type of person will have a stroke, based on the leading causes for strokes. We will train the models using logistic regression and multilayer perceptron, which is a type of artificial neural network. In section 2 we will go over the problem formulation and define the labels and features we will use. The processing of our data will be explained in section 3 alongside with the machine learning methods we will be using for our problem, their loss functions and how we have decided to split our data. Section 4 presents the training and validation errors we have gotten from the different models and the test error for the final chosen method. Our report will conclude with a summary of our findings and discussion on areas that could be improved in section 5.

2. Problem formulation

2.1 Problem

The aim of this machine learning project is to know if a person based on will have a stroke or not. There are a number of factors that contribute to a person having a stroke. We will take into account age, if the person is diagnosed with hypertension or heart disease, BMI and if the person is a smoker or not. We have collected our data from Kaggle, where it was published by one of its users by the name fedesoriano [3]. The data contains information needed to predict if a person will have a stroke or not. The data provides us with a lot of attributes, but

we have decided to use the ones mentioned above because we think they are the most relevant.

2.2 Dataset

The dataset consists of 5110 data points, and one data point consists of 12 different attributes. As mentioned earlier we have decided to only use 5 of the 12 attributes as features in our machine learning task. The features we will use are: age, if the person is diagnosed with hypertension or heart disease, BMI and if the person is a smoker or not. This machine learning task is supervised and we will be using the attribute for if a person has had a stroke or not as our label.

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

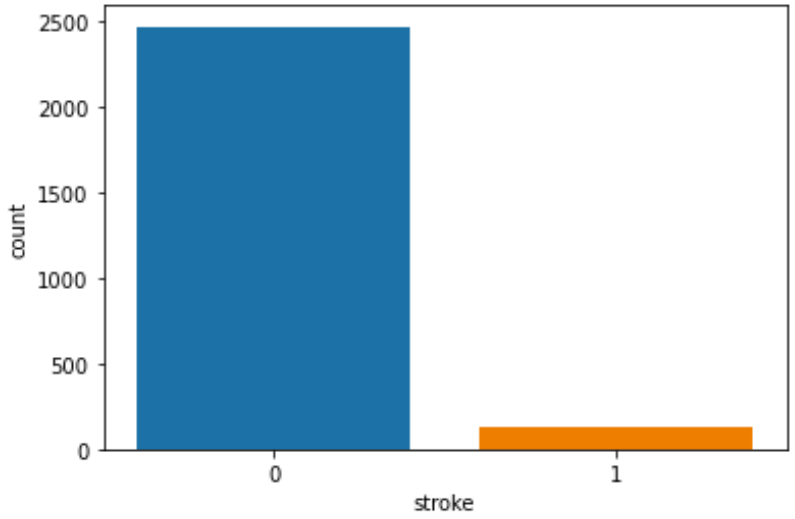
Data before clean up.

3. Methods

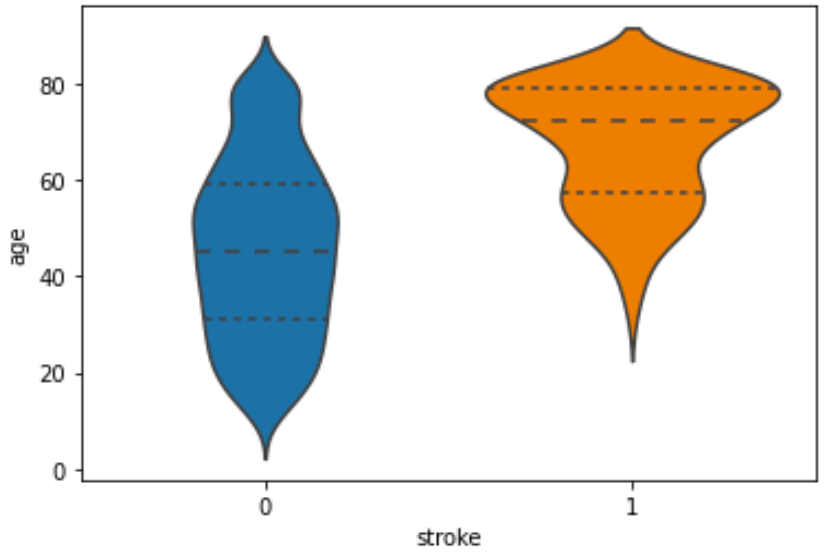
3.1 Preprocessing and feature selection

The preprocessing we have done to our data is the following: we started off by removing the attributes we weren't going to use as features, we removed the rows from the remaining data that were missing values or the data was not disclosed. We did this by using pandas. When it came to the attribute regarding if the person was a smoker or not we had to do a bit more work. The data was given in three different categories, 'smoker', 'never smoked' or 'formerly smoked'. We decided to rephrase this so that the data was in binary form. Smokers were turned into 1, people who never smoked to 0 and we decided to remove the values 'former smokers' all together. The reason we did this is because we thought the information given by the category wasn't clear enough. The effects of smoking on your risk of getting a stroke after quitting varies a lot depending on how long ago you quit and how long you were smoking for.

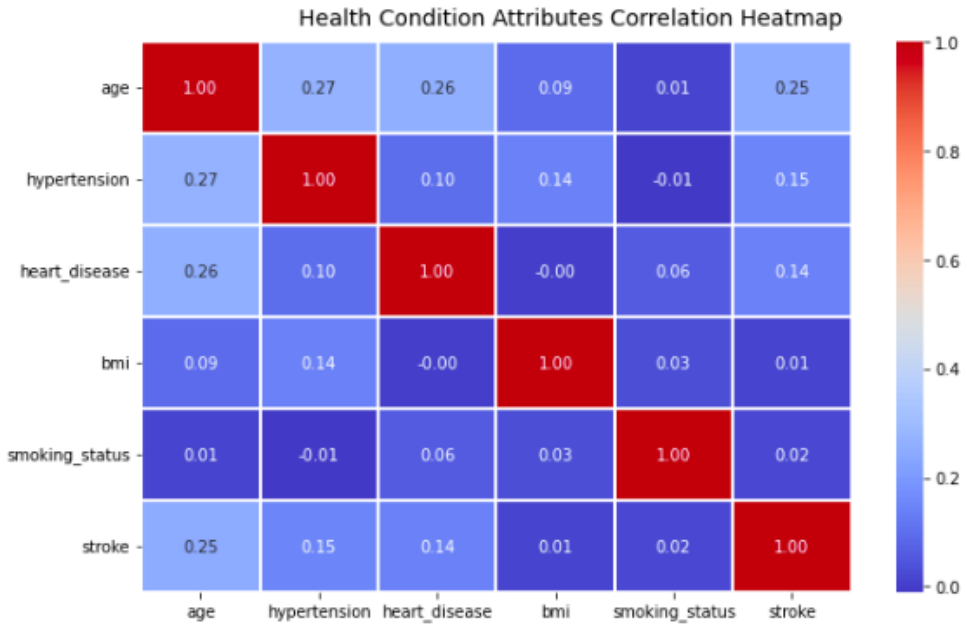
After the clean up we are left with 2589 data points and 6 columns (5 features and 1 label). The ratio of people who have had a stroke to those who have not is 5:100 in our data. This means that our data is imbalanced, which might lead to misleading results in the training phase. All our data is also now in numeric form, some of it binary and two of the features (age and BMI) are continuous data.



The number of data points where people have had a stroke (1), and people who have not (0).



Distribution of people that have had a stroke based on age.



Heatmap of health condition attributes

The reason we decided to choose these features is because we redeemed them the most relevant. In our opinion your marital status and id for example have little to no correlation to whether you are going to have a stroke or not. The only attribute that we decided to discard that might have been useful was the participants average glucose level. This does have some impact on whether or not you are going to have a stroke, but it would have been difficult for us to process the data into a form that we could easily work with.

	age	hypertension	heart_disease	bmi	smoking_status	stroke
2	80.0	0	1	32.5	0.0	1
3	49.0	0	0	34.4	1.0	1
4	79.0	1	0	24.0	0.0	1
6	74.0	1	1	27.4	0.0	1
7	69.0	0	0	22.8	0.0	1
...	...	...	...	...	...	...
5096	57.0	0	0	28.2	0.0	0
5100	82.0	1	0	28.3	0.0	0
5102	57.0	0	0	21.7	0.0	0
5106	81.0	0	0	40.0	0.0	0
5107	35.0	0	0	30.6	0.0	0

Data after clean up.

3.2 Machine learning methods and loss functions

The first machine learning method we have decided to use is logistic regression. The reason we have decided to use this method is because it is used for classification, which we are doing in our machine learning problem. The method also works very well with binary

classification. Logistic regression also clearly explains how several independent factors affect the outcome of the dependent value. The result we get from logistic regression is a binary output based on the label we have given, 1 for going to have a stroke and 0 for not going to have a stroke.

Because we are using logistic regression we will also be using logistic loss. This allows us to use ready made libraries for logistic regression and it is also very compatible with binary classification. Logistic loss measures how useful the linear hypothesis is. When the result predicted is in binary form the punishment for a false positive prediction needs to be harsher and this is why we use logistic loss over for example least square error [4].

The second machine learning method we are going to use is a feedforward Artificial neural network (ANN) called Multilayer perceptron (MLP). The reason we have decided to use this method is because it is very different from logistic regression, it uses deep learning and can be applied to classification. The structure of artificial neural networks can be divided into three categories, an input layer that takes in the input, hidden layers that perform calculations and an output layer that presents the output. There can be several hidden layers and in these ANN computes a weighted sum from the inputs using a bias function. The weighted total is passed to an activation function that produces the output. Because we want our results in binary form we use the Sigmoid function as our activation function [5].

The loss function we decided to use for our MLP regressor is the Mean squared error (MSE), because it is the loss function already implemented in the scikit-learn library and the one we used in a similar assignment on this course. The model calculates the average square difference between the predicted and observed values. The smaller the value is the more accurate the method used has been. Mean squared error also penalizes larger errors more than smaller ones, which is good when you want your errors to be as small as possible [6].

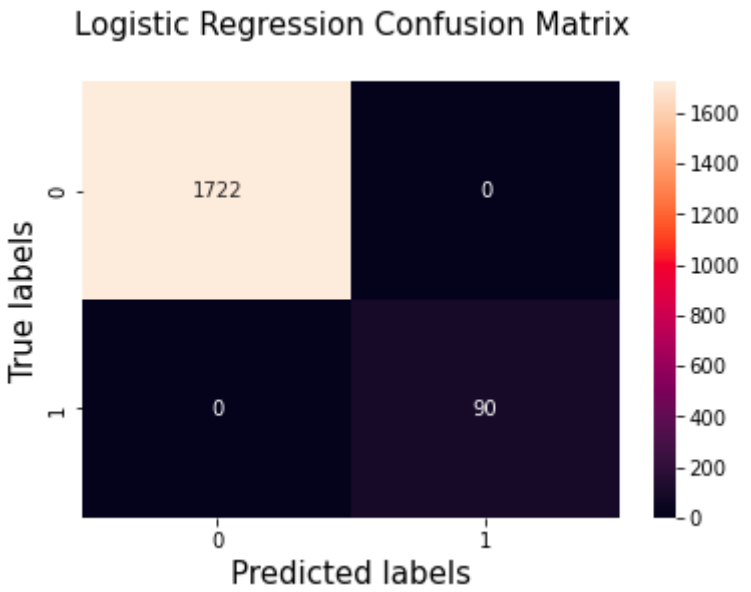
3.3 Training set, validation set and test set

We will be sectioning our data according to the hold-out method. With this method the data is split into a training set, validation set, and a test set with the ratio 70:15:15. The reason we use this split is because it is the most common split for the size of our data. We use the train_test_split-method in the sklearn-library to make sure that the entire dataset is

represented in the different sets. The training set is used to train the different machine learning methods, the validation set is used to validate the training set and finally the testing set is used to test the final chosen model [7].

4. Results

Both the Logistic Regressor and the MLP regressor gave very small training and validation errors. After plotting the confusion matrix for the logistic regressor we found that there were no cases of false negatives or false positives. Therefore the accuracy score for the logistic regressor was 1 and the logistic loss for both the training- and validation was $9.9 * 10^{-16}$, which is extremely small. The reasons for the "good" result might be that our models were overfitted. When a model is overfitted it means that the learned model consists of more parameters than what can be justified by the data and will not make an accurate prediction on unseen data.



Confusion matrix of the logistic regression model.

The MLP regressor also gave similar results with small errors, 0,00027 for the training set and 0,00024 the validation set. When comparing the training error and validation error for the models we see that the logistic regressor's errors are much smaller than the errors for the MLP regressor. Under normal circumstances we would have chosen the logistic regressor, but since we suspect that both our models are overfitted, the MPL regressor seems like the better

choice of the two. This is why we have decided to choose the MPL regressor as the model for our machine learning problem. The test error we got for the MPL regressor was 0,00058.

5. Conclusion

5.1 Summary

In this report we have researched if a person will have a stroke or not by using two different machine learning methods. The methods in question are linear regression and MLP. We have used five of the leading causes of strokes as features and a binary category of if a person has had a stroke or not as our label. We have processed our data to get it to our desired form and then split it into three different sets. Each set is used for either training, validation or testing. We have used linear loss and mean squared error as our loss functions. The results from our training and validation are assessed and our conclusion yields that MLP is the more suitable method for our problem.

5.2 Limitations and Improvements

Since we came to the conclusion that both our models are to some degree overfitted we thought of ways we could have prevented that from happening. We could have removed data points that could be classified as noise in our feature selection process and inserted our own data points to make the positive stroke class larger [8]. There are also other techniques we could have used, such as model comparison, cross-validation, regularization, early stopping, and pruning [9], but these techniques seemed beyond the scope of this project. We could also have looked at other machine learning methods for our problem such as, Support-Vector Machine and Decision Trees, but we found that the ones we chose

6. References

[1] https://www.nhs.uk/conditions/stroke/causes/

[2] https://www.brgeneral.org/news-blog/2021/may/why-are-more-young-people-having-strokes-/

[3] https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?resource=download

[4] https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11

[5] https://www.javatpoint.com/artificial-neural-network

[6] https://statisticsbyjim.com/regression/mean-squared-error-mse/

[7] https://vitalflux.com/hold-out-method-for-training-machine-learning-model/

[8] A. Jung,“Machine Learning: The Basics,” Springer, Singapore, 2022

[9] https://en.wikipedia.org/wiki/Overfitting

[] https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57

[] https://trainindata.medium.com/feature-engineering-for-machine-learning-a-comprehensive-overview-a7ad04c896f8

7. Appendices

Project

October 6, 2022

[1]:
```
# Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import log_loss, precision_score, accuracy_score,␣
 ↪confusion_matrix, mean_squared_error  # evaluation metrics

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

[2]:
```
# Reading and processing of data points
df = pd.read_csv('healthcare-dataset-stroke-data.csv')
df.drop(columns=['id', 'gender', 'ever_married','work_type', 'Residence_type',␣
 ↪'avg_glucose_level'], inplace=True)

df = df.replace(to_replace="smokes",value=1)
df = df.replace(to_replace="never smoked",value=0)
df = df.replace(to_replace="formerly smoked",value=np.nan)
df = df.replace(to_replace="Unknown",value=np.nan)
df = df.dropna(axis = 0) # drop rows with empty values
df
```

[2]:

	age	hypertension	heart_disease	bmi	smoking_status	stroke
2	80.0	0	1	32.5	0.0	1
3	49.0	0	0	34.4	1.0	1
4	79.0	1	0	24.0	0.0	1
6	74.0	1	1	27.4	0.0	1
7	69.0	0	0	22.8	0.0	1
…	…	…	…	…	…	…
5096	57.0	0	0	28.2	0.0	0
5100	82.0	1	0	28.3	0.0	0
5102	57.0	0	0	21.7	0.0	0

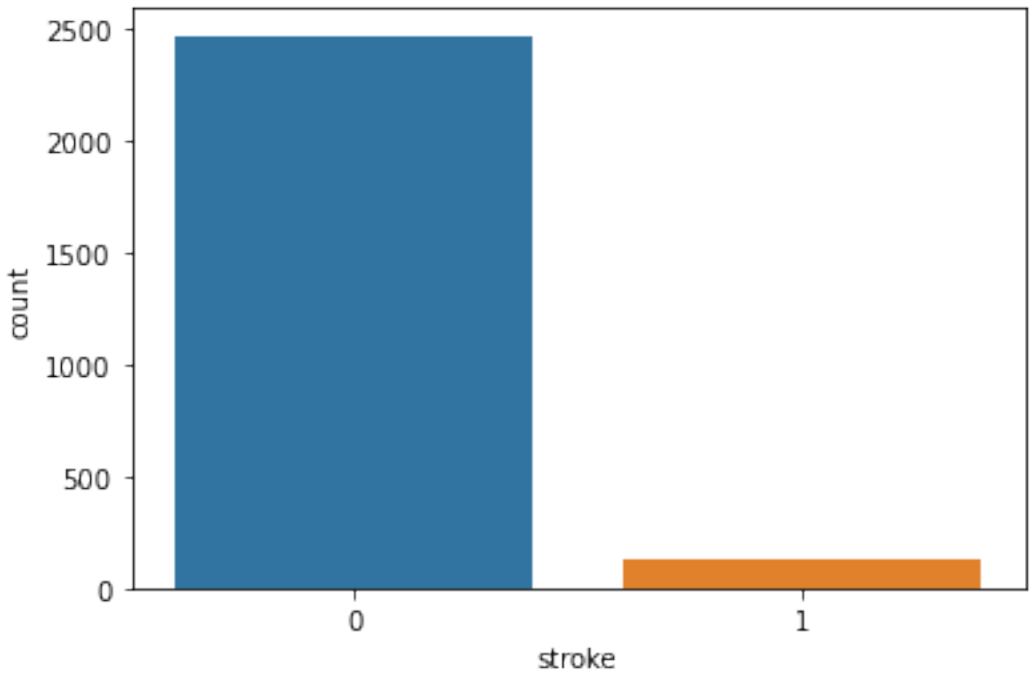
```
5106  81.0             0             0  40.0             0.0       0
5107  35.0             0             0  30.6             0.0       0

[2589 rows x 6 columns]
```

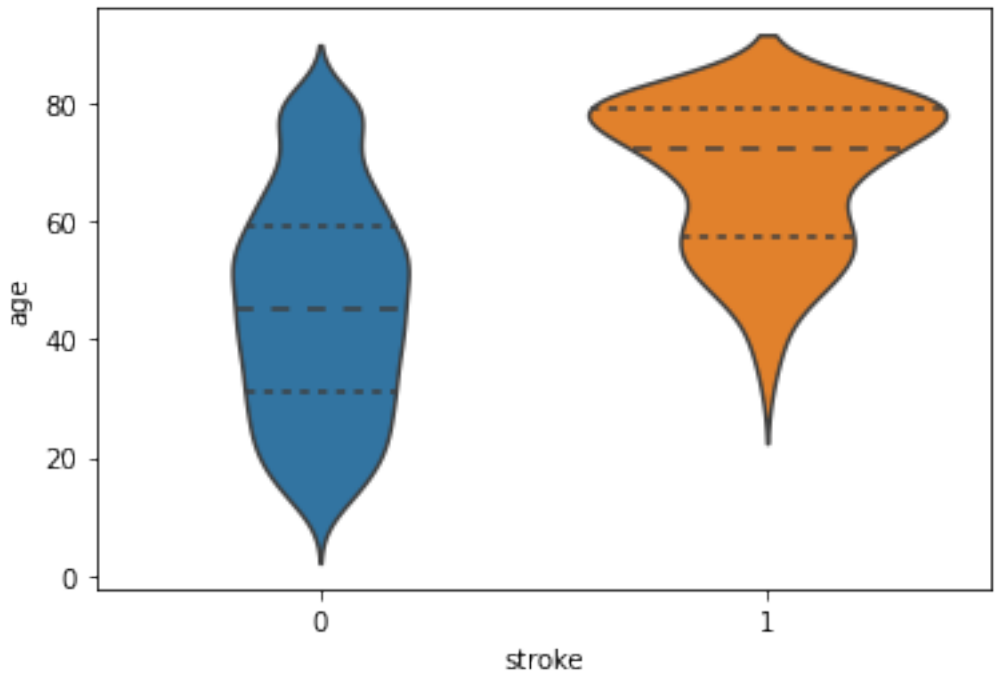
[3]:
```
# Visualizations of some attributes
sns.countplot(df['stroke'])
```

```
/opt/software/lib/python3.9/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

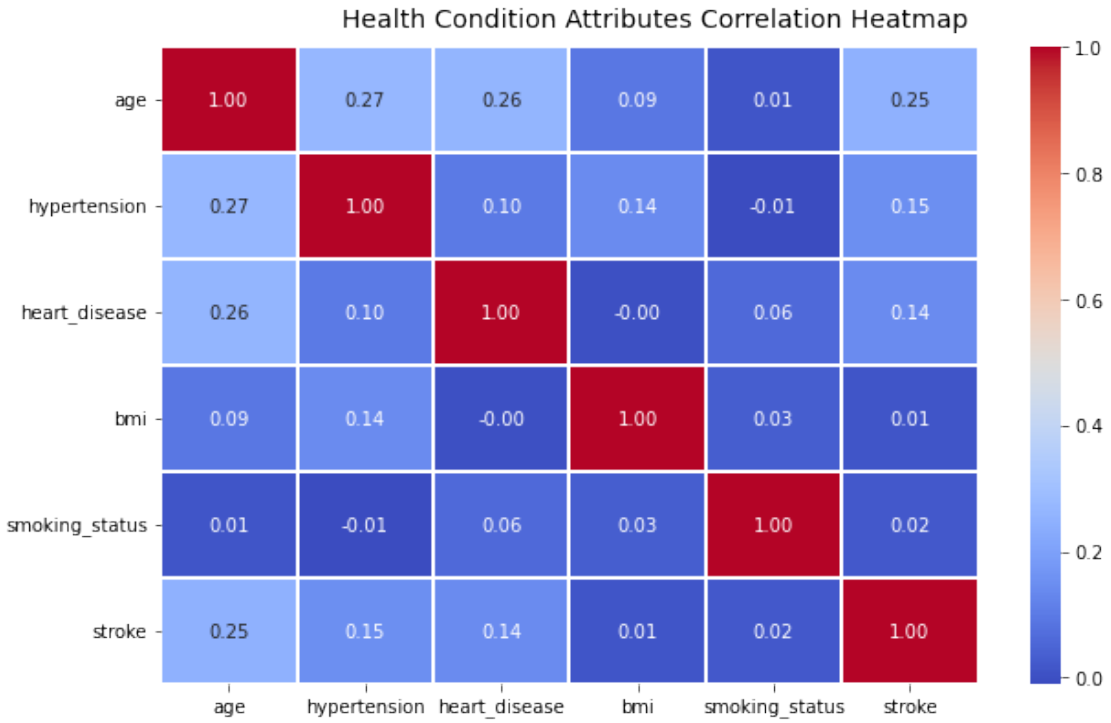
[3]:
```
<AxesSubplot:xlabel='stroke', ylabel='count'>
```



[4]:
```
sns.violinplot(y='age', x='stroke', data=df, inner='quartile')
plt.show()
```



[5]:
```
# Correlation Matrix Heatmap
f, ax = plt.subplots(figsize=(10, 6))
corr = df.corr()
hm = sns.heatmap(round(corr,2), annot=True, ax=ax, cmap="coolwarm",fmt='.2f',
                 linewidths=.05)
f.subplots_adjust(top=0.93)
t= f.suptitle('Health Condition Attributes Correlation Heatmap', fontsize=14)
```



[6]:
```
# Selecting features and labels
features =␣
 ↪df[['age','hypertension','heart_disease','bmi','smoking_status','stroke']].
 ↪to_numpy() # shape (2589,6)
labels = df['stroke'].to_numpy() # shape (2589,)
```

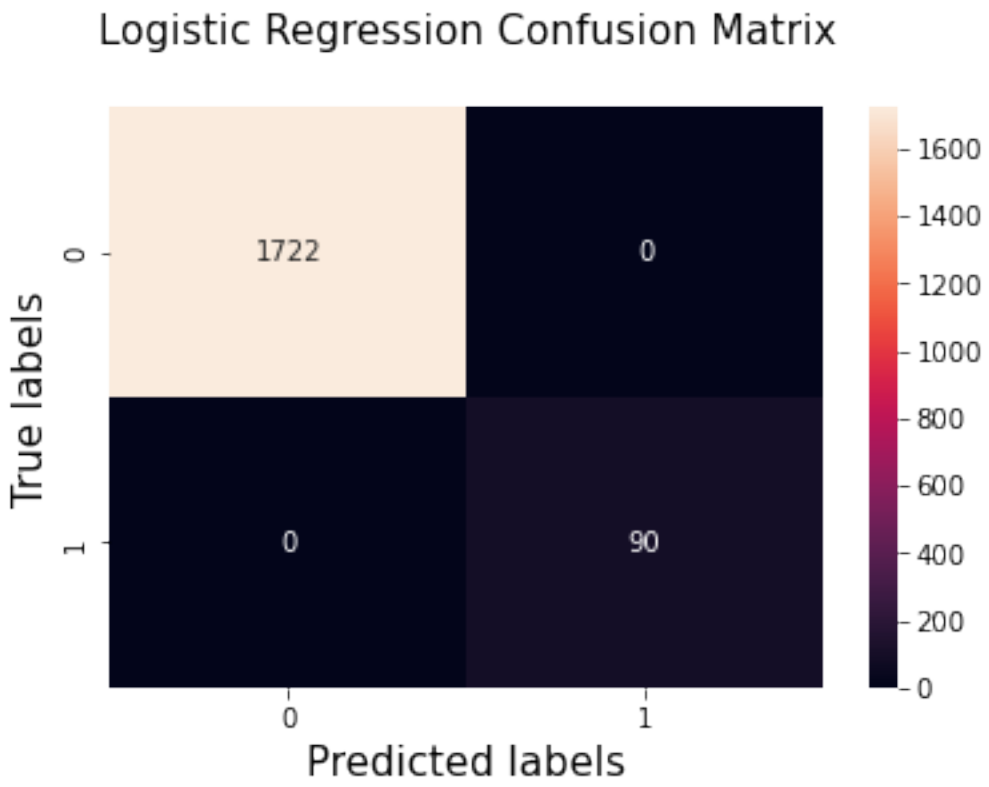
[7]:
```
# Splitting data into training set, validation set and testing set
X_train, X_rem, y_train, y_rem = train_test_split(features, labels, test_size =␣
 ↪0.30, random_state=43)
X_test, X_val, y_test, y_val = train_test_split(X_rem, y_rem,test_size = 0.15,␣
 ↪random_state=43)
```

[8]:
```
# Method 1: Logistic Regression
clf_1 = LogisticRegression()
clf_1.fit(X_train,y_train)

# compute predicted labels for training and validation set
y_pred_train_log = clf_1.predict(X_train)
y_pred_val_log = clf_1.predict(X_val)
```

[9]:
```
# confusion matrix
confmat_log = confusion_matrix(y_train, y_pred_train_log)
```

```
# plot the confusion matrix
ax = plt.subplot()
sns.heatmap(confmat_log,annot=True, fmt='g', ax=ax)
ax.set_title('Logistic Regression Confusion Matrix\n',fontsize=15)
ax.set_xlabel('Predicted labels',fontsize=15)
ax.set_ylabel('True labels',fontsize=15)
plt.show()
```



[10]:
```
# Logistic loss for training and validation data
loss_train = log_loss(y_train, y_pred_train_log)
loss_val = log_loss(y_val, y_pred_val_log)
print(loss_train)
print(loss_val)
```

```
9.992007221626413e-16
9.992007221626413e-16
```

[11]:
```
# Method 2: Multilayer perceptron (MLP)
mlp_regr = MLPRegressor(activation = 'logistic', random_state=42,␣
 ↪max_iter=1000) # Sigmoid function (logistic) used as activation function
mlp_regr.fit(X_train,y_train)     # Train MLP on the training set
```

```
## evaluate the trained MLP on both training set and validation set
y_pred_train_mlp = mlp_regr.predict(X_train)    # predict on the training set
y_pred_val_mlp = mlp_regr.predict(X_val) # predict values for the validation␣
 ↪data
```

[12]:
```
# Training and Validation errors for MLP using Mean Squared Error functions
MSE_train = mean_squared_error(y_train, y_pred_train_mlp)    # training error
MSE_val = mean_squared_error(y_val, y_pred_val_mlp)       # validation error
print(MSE_train)
print(MSE_val)
```

```
0.0002738068349228482
0.00024375237395388673
```

[13]:
```
# Training error for chosen model, method #2 (MLP)
y_pred_test_mlp = mlp_regr.predict(X_test)
MSE_test = mean_squared_error(y_test, y_pred_test_mlp)
print(MSE_test)
```

```
0.0005757804534356
```