Stroke Prediction

Submitted By: Lashika Arunachalam

Date of Submission: 23-01-2023

University of Roehampton London

Table of Contents

University of Roehampton London

1. Source of Data:

A serious cerebrovascular condition called a stroke is brought on by a blockage in the supply of blood to and from the brain which prevents the brain from getting the oxygen and nutrients it needs to function properly. The brain is in severe condition as brain cells will soon start to die within a matter of minutes. The majority of characteristics line up with clinical medical data. The characteristics shows the dataset offers pertinent information regarding the chance of patients having a stroke disease. It is understandable that a patient with high blood sugar and BMI, who has had heart disease and/or hypertension, is more prone to experience a stroke. The source of the data set is https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

2. Pre-processing involved:

Cleaning and organising raw data is a technique used in machine learning to prepare it for the creation and development of machine learning techniques. Data cleaning is done here. Data cleaning is the process of adding missing details and removing false or pointless information from a data set. Dating cleansing is the most important pre-processing step because it ensures that the data is ready for your future requirements. In the beginning, look for identical and empty values. This dataset contains no values that are duplicates. The bmi column has some incomplete data.

Data Pre Processing

Checking of Null values

```
df.isnull().sum()
```

```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                201
smoking_status       0
stroke               0
dtype: int64
```

```
[ ] df.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

The average of the bmi column is filled using the null values in bmi column. After performing the data pre-processing, the dataset became clean.

Treating Null values

```
[ ] df['bmi'].value_counts()
```

```
28.7    41
28.4    38
26.7    37
27.6    37
26.1    37
        ..
48.7     1
49.2     1
51.0     1
49.4     1
14.9     1
Name: bmi, Length: 418, dtype: int64
```

```
[ ] df['bmi'].describe()
```

```
count    4909.000000
mean       28.893237
std         7.854067
min        10.300000
25%        23.500000
50%        28.100000
75%        33.100000
max        97.600000
Name: bmi, dtype: float64
```

```
[ ] df['bmi'].fillna(df['bmi'].mean(),inplace=True)
```

```
df.isnull().sum()
```

```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

3. Discussion on Hypothesis:

The major goal is to determine whether an individual is likely to experience a brain haemorrhage, and then to take necessary action to stop extra harm to the damaged area of the brain and other issues in other bodily systems. In a countable number of healthy and unwell people, there are some risk factors for stroke, diabetes, cardiovascular disease, smoking, and alcoholic use.

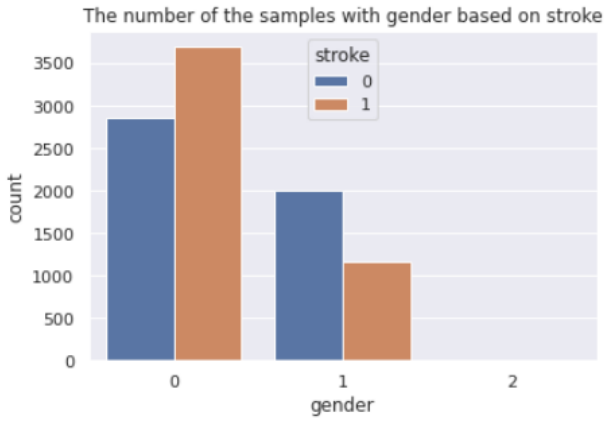
Another goal is to determine which gender suffers most of the stroke whether female or male because the study says female are with a ratio of 57.42% followed by male with a ratio of 42.58% and to determine which stroke patient is suffered whether the person who never smoked or who has a habit of smoking. It's found the person who never Smoked with a ratio of 40.19% followed by the person smoked in the past with a ratio of 27.27%.

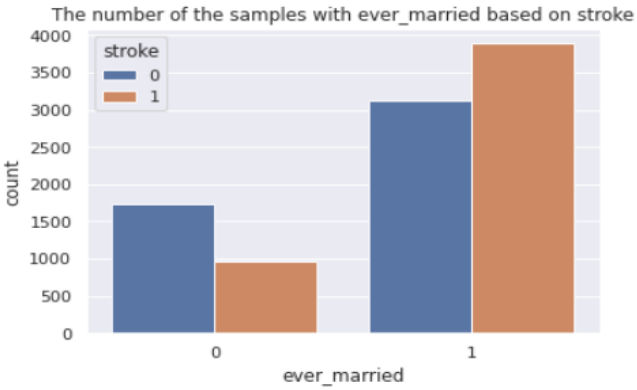
The hypothesis is followed by the patient who is married or unmarried and the person who is working is different job sectors such as private, government and self-employed because these can be found that the patients who is experiencing more stress results in fluctuation of glucose levels so that we can come to the conclusion, they are affected by stroke.

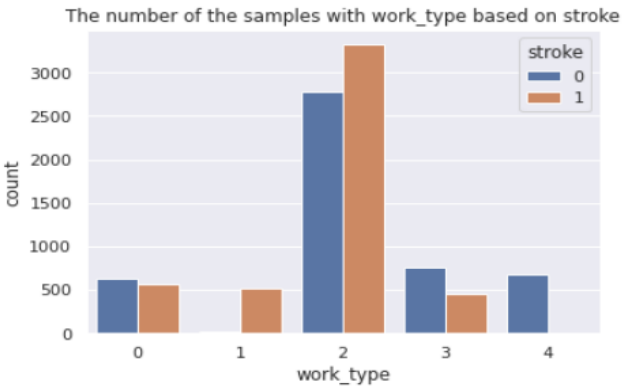
University of Roehampton London

Visualization

```
df_as = pd.concat([X, y], axis = 1)

sns.set_theme(style = 'darkgrid')
for i in df_as.columns[:-1]:  # exclude stroke column
    if (df_as[i].dtype == 'object') or (df_as[i].dtype == 'int64'):
            sns.countplot(data = df_as, x = i, hue = 'stroke')
            plt.title('The number of the samples with {} based on stroke'.format(i))
            plt.show()
```







4. Technique used:

The technique used here are kNN, Logistic Regression, Decision Tree Classifier, Random Forest. The k Nearest Neighbours methodology, sometimes known as kNN, is the most straightforward machine learning technique. It is a non-parametric approach being used regression and classification problems. Non-parametric indicates that no data distribution hypothesis is necessary. Therefore, kNN does not involve the use of any core assumptions. The k closest training samples in the feature space provide the input for the regression and classification tasks respectively. Whenever kNN is applied for cataloguing or regression determines the results.

The outcome of kNN classification is a class membership. Depending on the type of its immediate neighbours, the given data point is categorised. The data point's k closest neighbours decide which group it belongs to most frequently. K is typically a little positive integer. The data object is merely allocated to the class of its lone nearest neighbour if k=1. The result of a kNN regression is just a property value for the object. The mean of the values of the k closest neighbours makes up this number.

The kNN algorithm idea is really easy to grasp. It only determines the separation between each training piece of information and the actual target value. The separation may be Manhattan distance or Euclidean distance. The next step is to choose the k closest data points, wherein k is an integer. The sample data point is then assigned to the class that the most of the other k data points correspond to.

If conditional independence would have a significant negative impact on categorization, K-NN should be chosen instead of Naive Bayes. When an attribute's conditional probability is zero, Naive Bayes can experience the zero-possibility issue and will not be able to make any predictions at all. A Laplacian estimator may be used to remedy this, although K-NN might eventually wind up becoming the simpler option. Just elliptic, parabolic, or linear decision boundaries are compatible with naive Bayes. Alternatively, pick K-NN. With K-NN, you simply load the dataset and let it go; no training is necessary. Naive Bayes, on the other hand, does call for instruction.

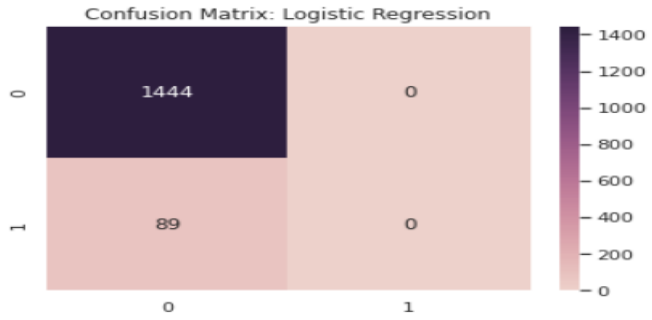
In contrast to LR, which only provides linear solutions, KNN allows non-linear services. After tuning of hyperparameter, comparing with Logistic regression, kNN and Random forest technique gives the same accuracy.

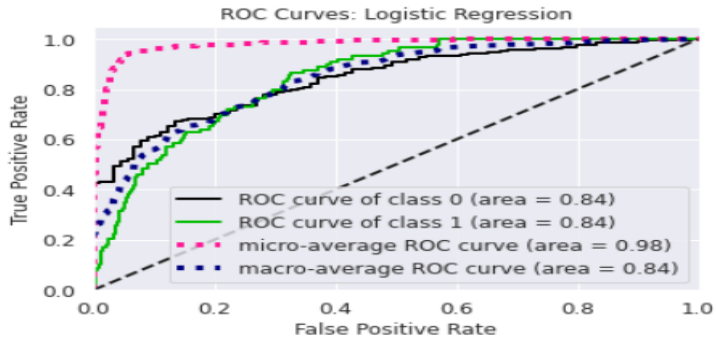
Model Training

```
[49] models = [('Logistic Regression', LogisticRegression()),
               ('Random Forest', RandomForestClassifier()),
               ('KNN', KNeighborsClassifier())]

     models_score = []
     for name, model in models:
         model = model
         model.fit(X_train, y_train)
         model.predict(X_test)
         models_score.append([name, accuracy_score(y_test, model.predict(X_test))])

         print(name)
         print('Validation Accuracy: ', accuracy_score(y_test, model.predict(X_test)))
         print('Training Accuracy: ', accuracy_score(y_train, model.predict(X_train)))

         plt.figure()
         cf_matrix = confusion_matrix(y_test, model.predict(X_test))
         plt.title('Confusion Matrix: {}'.format(name))
         sns.heatmap(cf_matrix, annot = True, fmt = 'g', cmap = sns.cubehelix_palette(as_cmap=True))
         plt.show()

         import scikitplot as skplt

         skplt.metrics.plot_roc(y_test, model.predict_proba(X_test))
         plt.title('ROC Curves: {}'.format(name))
         plt.show()
```

```
Logistic Regression
Validation Accuracy:  0.9419439008480104
Training Accuracy:  0.9552697791445345
```

```
Logistic Regression
Validation Accuracy:  0.9419439008480104
Training Accuracy:  0.9552697791445345
```





Tuning of Hyperparameter

```
[51] grid_models = [(LogisticRegression(),[{'C' : [0.3, 0.7, 1], 'random_state' : [42]}]),
                    (RandomForestClassifier(),[{'n_estimators' : [100, 200, 300], 'criterion' : ['gini','entropy'], 'random_state' : [42]}]),
                    (KNeighborsClassifier(),[{'n_neighbors' : [4, 6, 8, 10], 'metric' : ['euclidean', 'manhattan', 'chebyshev','minkowski']}])]
```

```
for model, param_grid  in grid_models:
    cv = GridSearchCV(estimator = model, param_grid = param_grid, scoring = 'accuracy', cv = 5)
    cv.fit(X_train, y_train)
    best_accuracy = cv.best_score_
    best_params = cv.best_params_
    print('{}: \nBest Accuracy: {:.2f}%'.format(model, best_accuracy*100))
    print('Best Parameters: ',best_params)
    print('******************************')
```

```
LogisticRegression():
Best Accuracy: 95.53%
Best Parameters:  {'C': 0.3, 'random_state': 42}
******************************
RandomForestClassifier():
Best Accuracy: 95.55%
Best Parameters:  {'criterion': 'entropy', 'n_estimators': 300, 'random_state': 42}
******************************
KNeighborsClassifier():
Best Accuracy: 95.55%
Best Parameters:  {'metric': 'manhattan', 'n_neighbors': 4}
******************************
```