Dataset_Stoke

May 30, 2021

1 Step one: Handle null data

1.1 have a glance at the data

```
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          Yes
1  51676  Female  61.0             0              0          Yes
2  31112    Male  80.0             0              1          Yes
3  60182  Female  49.0             0              0          Yes
4   1665  Female  79.0             1              0          Yes

       work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
0        Private          Urban             228.69  36.6  formerly smoked
1  Self-employed          Rural             202.21   NaN     never smoked
2        Private          Rural             105.92  32.5     never smoked
3        Private          Urban             171.23  34.4           smokes
4  Self-employed          Rural             174.12  24.0     never smoked

   stroke
0       1
1       1
2       1
3       1
4       1

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
```

```
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

1.2 Find null datas

```
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
```

1.3 To deal with NaN data, I want to evaluate these bmi data with age and gender, using decision tree

having a look at the predicted bmi data

```
1       29.879487
8       30.556098
13      27.247222
19      30.841860
27      33.146667
          …
5039    32.716000
5048    28.313636
5093    31.459322
5099    28.313636
5105    28.476923
Length: 201, dtype: float64
```

1.4 so now we have all the datas

```
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
```

2 Step two: Now, let's find the relation ship among every predictors

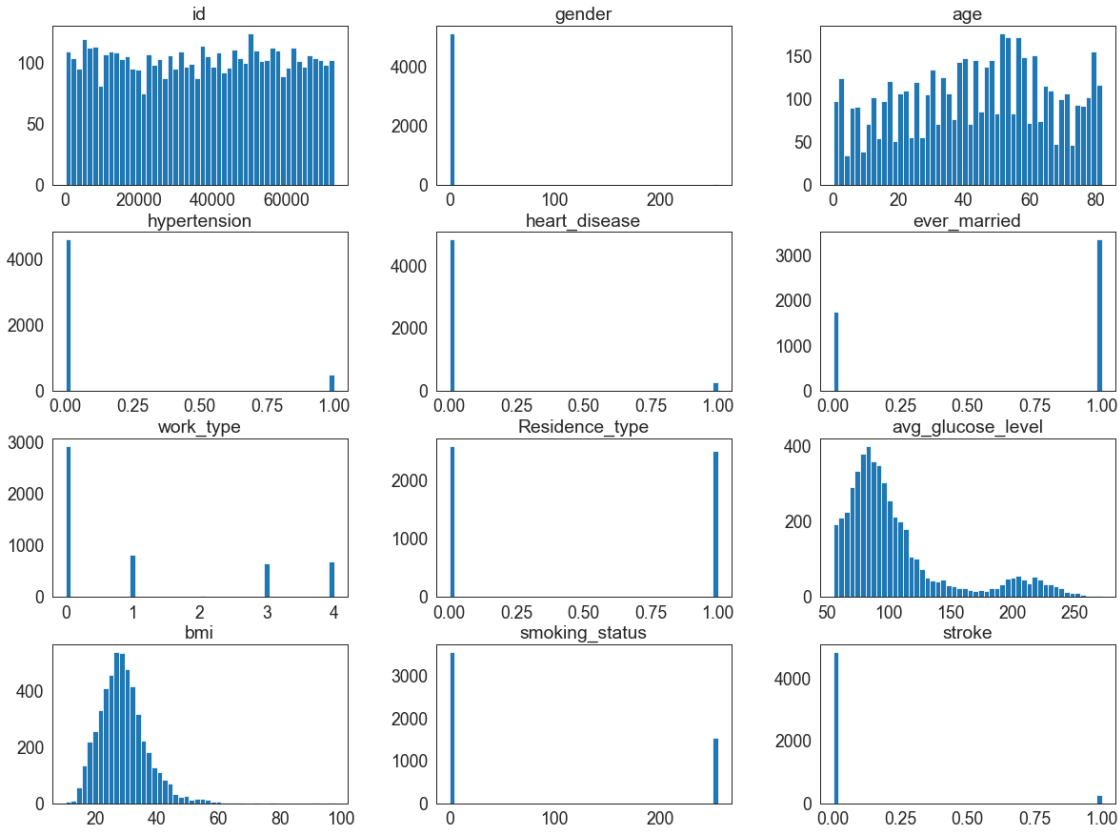
2.1 change these string value to districted integers

```
         id  gender   age  hypertension  heart_disease  ever_married  \
0      9046       0  67.0             0              1             1
1     51676       1  61.0             0              0             1
2     31112       0  80.0             0              1             1
3     60182       1  49.0             0              0             1
4      1665       1  79.0             1              0             1
…      …     …    …              …              …             …
5105  18234       1  80.0             1              0             1
5106  44873       1  81.0             0              0             1
5107  19723       1  35.0             0              0             1
5108  37544       0  51.0             0              0             1
5109  44679       1  44.0             0              0             1

      work_type  Residence_type  avg_glucose_level        bmi  smoking_status  \
0             0               0             228.69  36.600000               5
1             1               1             202.21  29.879487               0
2             0               1             105.92  32.500000               0
3             0               0             171.23  34.400000               3
4             1               1             174.12  24.000000               0
…         …             …               …        …             …
5105          0               0              83.75  28.476923               0
5106          1               0             125.20  40.000000               0
5107          1               1              82.99  30.600000               0
5108          0               1             166.29  25.600000               5
5109          3               0              85.28  26.200000             255

      stroke
0          1
1          1
2          1
3          1
4          1
```

```
...       ...
5105        0
5106        0
5107        0
5108        0
5109        0

[5110 rows x 12 columns]
```

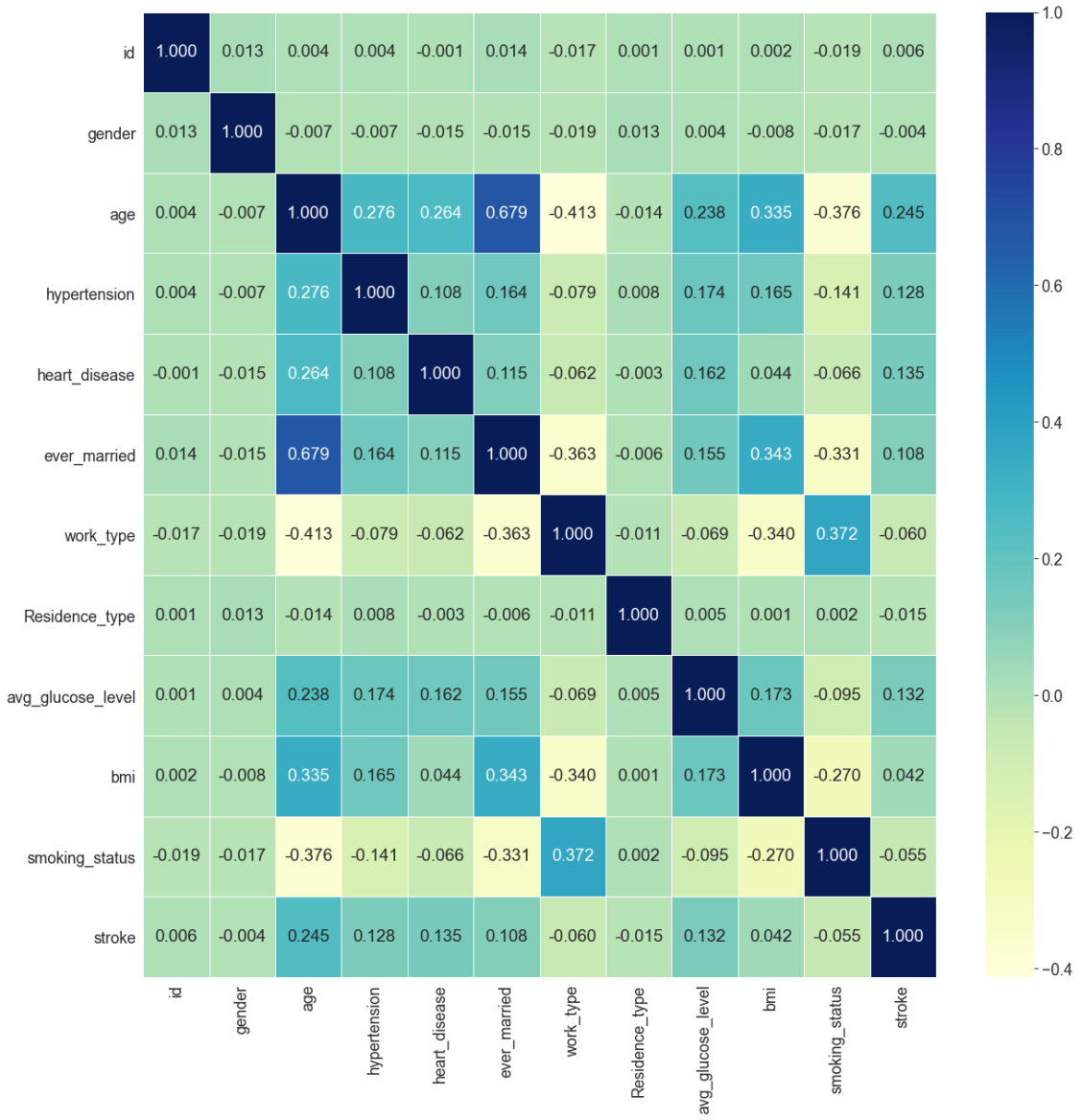
2.2 plot the distribution of every predictors with histogtram

```
array([[<AxesSubplot:title={'center':'id'}>,
        <AxesSubplot:title={'center':'gender'}>,
        <AxesSubplot:title={'center':'age'}>],
       [<AxesSubplot:title={'center':'hypertension'}>,
        <AxesSubplot:title={'center':'heart_disease'}>,
        <AxesSubplot:title={'center':'ever_married'}>],
       [<AxesSubplot:title={'center':'work_type'}>,
        <AxesSubplot:title={'center':'Residence_type'}>,
        <AxesSubplot:title={'center':'avg_glucose_level'}>],
       [<AxesSubplot:title={'center':'bmi'}>,
        <AxesSubplot:title={'center':'smoking_status'}>,
        <AxesSubplot:title={'center':'stroke'}>]], dtype=object)
```



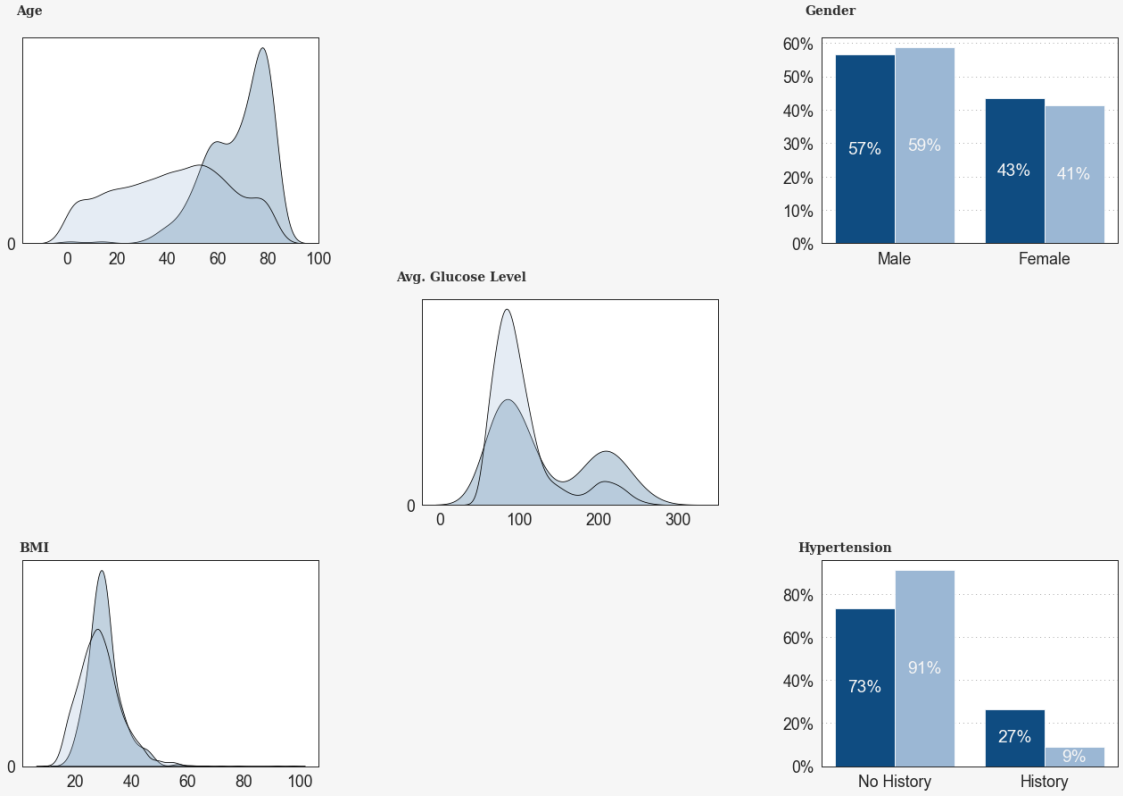
2.3 plot the heat map

```
<AxesSubplot:>
```



According to the upper picture, we found that the propability of stroke has little related to work_type, Residence_type and smoking_status.

Let's go further.

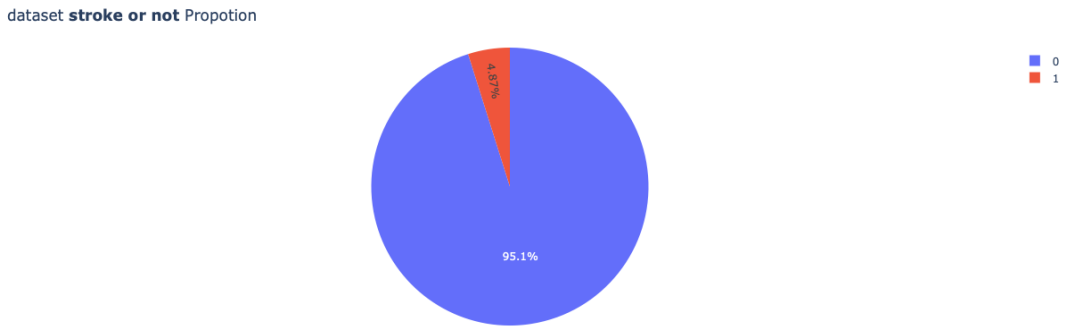


So we build model with upper predictors

3 Step three: Before building models, we affirm the training and testing datas first.

3.1 We first look at the number of the data who has stroke and who has't

```
number of people who has stroked 249
number of people who hasn't stroked 4861
```



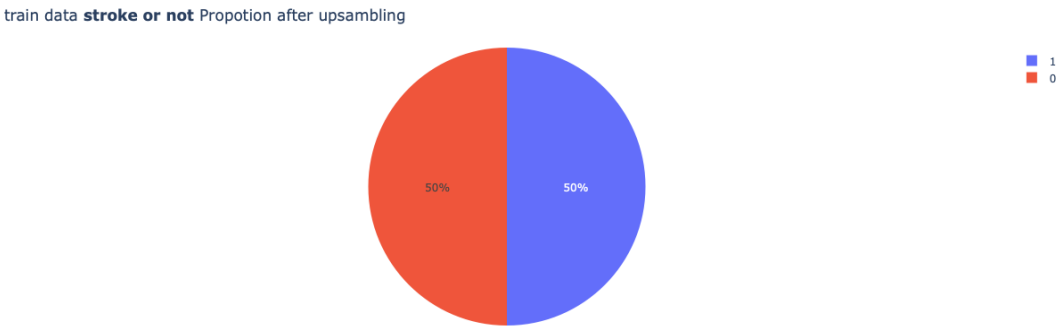
We find that there are too many data of who hasn't attacked by stroke, so we should balance these numbers of datas.

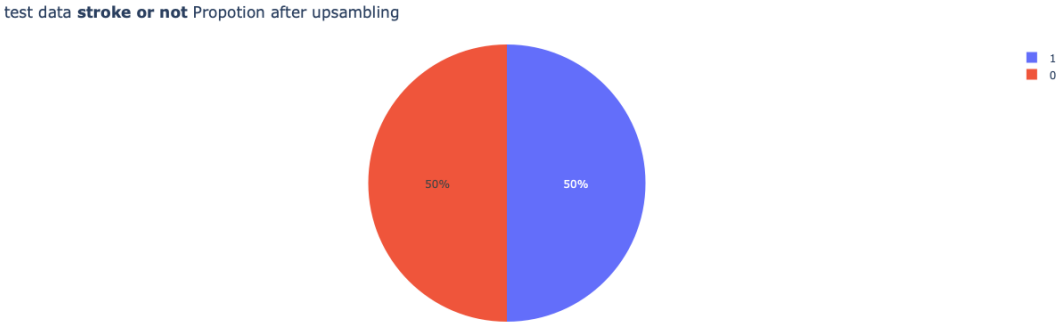
We now solve over-sambling problem.

3.2 We use SMOTE(Synthetic Minority Over-sampling Technique) to generate more datas.

The traditional over-sampling methode, just duplicating these minority sample and making no new infomation to the dataset, which will easily causes over-fitting problem. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Aliking the equation: x_new = x + rand(0,1) * (ˆx - x)

```
986 986
3875 3875
```





4 Step four: Build models and analyse the result

I will try following models:

Random Forest

Logistic Regression

Decision Tree

MLP Neural Network with L-BFGS

MLP Neural Network with SGD

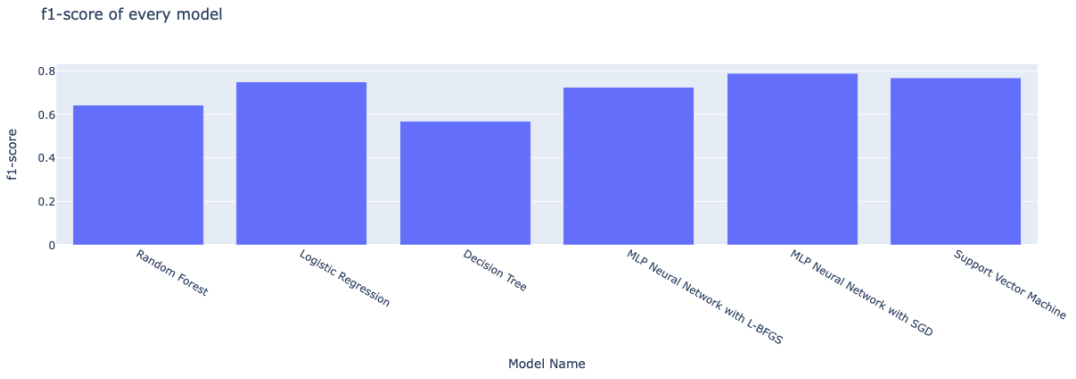
Support Vector Machine

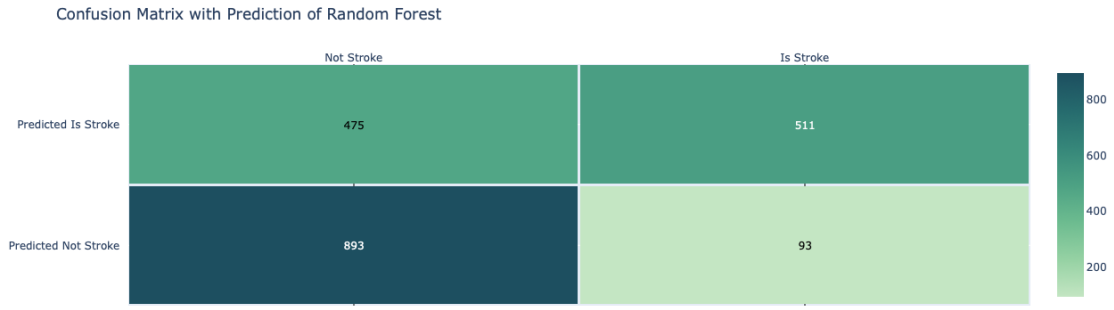
```
The correction rate of cross validation of Random Forest:  0.9254277042396863
The correction rate of cross validation of Logistic Regression:
0.8125404112385366
The correction rate of cross validation of Decision Tree:  0.9000791816875742
The correction rate of cross validation of MuliLayer Perception with L-BFGS
algorithm:  0.8193992873591979
The correction rate of cross validation of MuliLayer Perception with Stochastic
Gradient Descent algorithm:  0.81339693090295
The correction rate of cross validation of Support Vector Machine :
0.8224945320864562
```

From what we have seen above, we can conclusion easly that the model Random Forest preform best.

But one who is illed wouldn't like to get a wrong answer, to make it more reasonable, we next calculate the Recall Rate

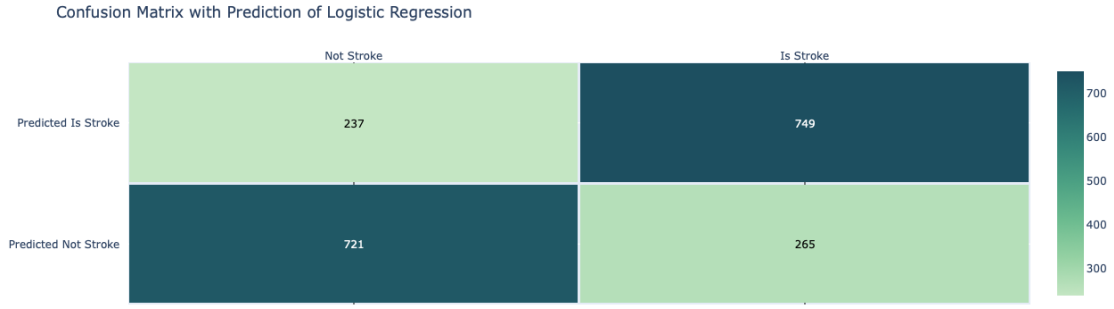
```
Pipeline(steps=[('scale', StandardScaler()), ('SVM', SVC(random_state=30))])
```





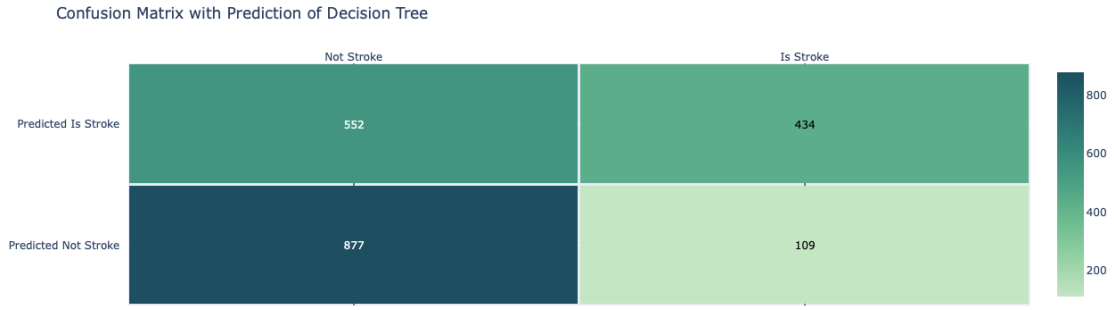
	precision	recall	f1-score	support
0	0.65	0.91	0.76	986
1	0.85	0.52	0.64	986
accuracy			0.71	1972
macro avg	0.75	0.71	0.70	1972
weighted avg	0.75	0.71	0.70	1972

Accuracy Score: 0.7119675456389453



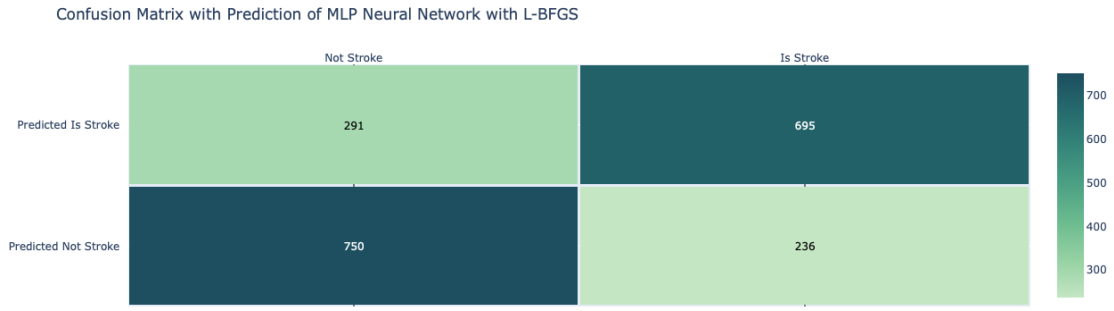
	precision	recall	f1-score	support
0	0.75	0.73	0.74	986
1	0.74	0.76	0.75	986
accuracy			0.75	1972
macro avg	0.75	0.75	0.75	1972
weighted avg	0.75	0.75	0.75	1972

```
Accuracy Score:  0.7454361054766734
```



	precision	recall	f1-score	support
0	0.61	0.89	0.73	986
1	0.80	0.44	0.57	986
accuracy			0.66	1972
macro avg	0.71	0.66	0.65	1972
weighted avg	0.71	0.66	0.65	1972

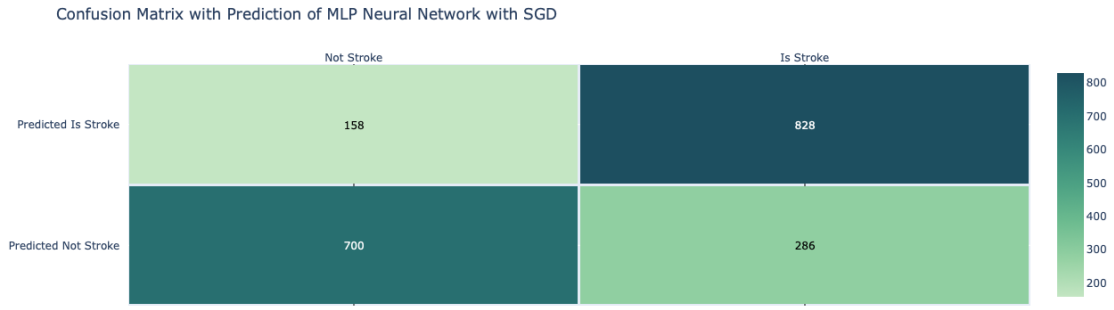
```
Accuracy Score:  0.6648073022312373
```



	precision	recall	f1-score	support
0	0.72	0.76	0.74	986
1	0.75	0.70	0.73	986
accuracy			0.73	1972

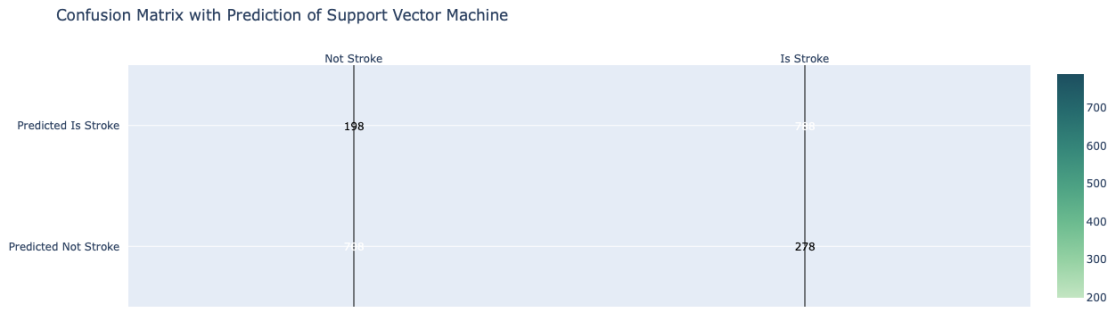
macro avg	0.73	0.73	0.73	1972
weighted avg	0.73	0.73	0.73	1972

Accuracy Score: 0.7327586206896551



	precision	recall	f1-score	support
0	0.82	0.71	0.76	986
1	0.74	0.84	0.79	986
accuracy			0.77	1972
macro avg	0.78	0.77	0.77	1972
weighted avg	0.78	0.77	0.77	1972

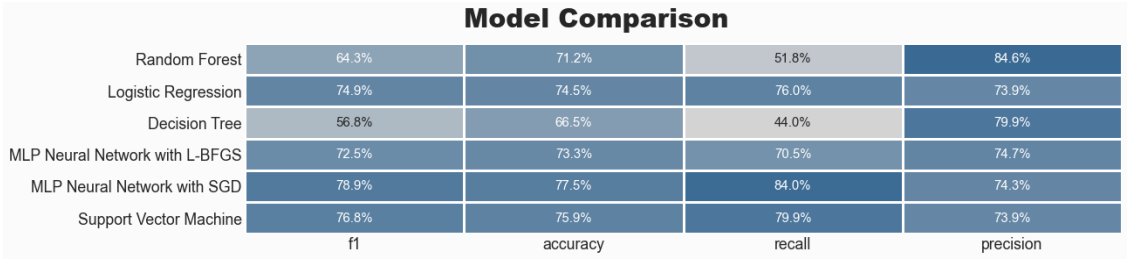
Accuracy Score: 0.7748478701825557



	precision	recall	f1-score	support
0	0.78	0.72	0.75	986
1	0.74	0.80	0.77	986

```
    accuracy                           0.76      1972
   macro avg       0.76      0.76      0.76      1972
weighted avg       0.76      0.76      0.76      1972

Accuracy Score:  0.7586206896551724
```

5 Step five: Model Selection



5.0.1 So obviously, we find that the model of MLP Neural Network with SGD algorithm has the highest recall rate of class_1 (is_stroke) which means that it will misclassify least cases of patient who has stoke.

6 Step six: Find the best parameters

6.1 Find the iterater times of largest recall rate

```
recall rate:  0.8539553752535497 index:  24
```

6.2 Replot the accuracy rate of this parameter

```
recall rate: 0.8529411764705882
```

```
              precision    recall  f1-score   support

           0       0.83      0.71      0.76       986
           1       0.74      0.85      0.79       986

    accuracy                           0.78      1972
   macro avg       0.79      0.78      0.78      1972
weighted avg       0.79      0.78      0.78      1972

Accuracy Score:  0.7794117647058824
```

6.3 Conclusion

Due to the randomness of the method stochastic gradient descend and we can do early stopping when we training the data with MultiLayer Network, we can control when to stop by the will of my own. So in this case, I can gain a higher recall rate by sacrificing the accuracy rate. But if we want the best performance of accuracy, (balanced) random forest is still the best choice.

7 Last step: Train the model with all data

```
Pipeline(steps=[('scale', StandardScaler()),
                ('MLP',
                 MLPClassifier(hidden_layer_sizes=(5, 2), max_iter=40,
                               random_state=30, warm_start=True))])
```

7.1 Test with several data

```
array([1])
```

```
array([1])
```

7.2 Store the optimal model

```
['./data/model_MLP_SGD_fitted.pkl']
```