About Data

There are certain factors which influence the chances of getting a stroke. This dataset contains a person's information like gender, age, hypertension, heart_disease, ever_married, work_type, Residence_type, avg_glucose_level, bmi, smoking_status and we have to predict whether they will get a stroke or not.

Import libraries

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

importing Data

In [3]:
```
stroke = pd.read_csv(r"C:\Users\hp\Downloads\healthcare-dataset-stroke-data.csv")
```

In [4]:
```
stroke.head()
```

Out[4]:

	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

Dropping unnecessary columns

In [5]:
```
# Drop the "id" column as it is irrelavent in the prediction

stroke = stroke.drop("id", axis=1)
stroke.head()
```

Out[5]:

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

missing values

There are "unknown"values in the smoking_status column. we will treat them as missing values

In [6]:
```
stroke["smoking_status"].replace("Unknown", np.nan, inplace=True)
```

In [7]:
```
stroke.isna().sum()
```

Out[7]:
```
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      1544
stroke                 0
dtype: int64
```

Filling missing Values

In [9]:
```
stroke["bmi"].fillna(stroke["bmi"].mean(), inplace=True)
stroke["smoking_status"].fillna(stroke["smoking_status"].mode()[0], inplace = True)

stroke.isna().sum()
```

Out[9]:
```
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
```

Using LabelEncoder()

we will encode target labels with values between 0 and n_classes-1

In [22]:
```
from sklearn.preprocessing import LabelEncoder
def label_encoded(feat):
    le = LabelEncoder()
    le.fit(feat)
    print(feat.name,le.classes_)


 #   print(le.classes_)
    return le.transform(feat)
```

In [23]:
```
for col in stroke.columns:
    stroke[str(col)] = label_encoded(stroke[str(col)])
```

```
 44.5        44.6        44.7        44.8        44.9        45.
 45.1        45.2        45.3        45.4        45.5        45.7
 45.8        45.9        46.         46.1        46.2        46.3
 46.4        46.5        46.6        46.8        46.9        47.1
 47.3        47.4        47.5        47.6        47.8        47.9
 48.         48.1        48.2        48.3        48.4        48.5
 48.7        48.8        48.9        49.2        49.3        49.4
 49.5        49.8        49.9        50.1        50.2        50.3
 50.4        50.5        50.6        50.8        50.9        51.
 51.5        51.7        51.8        51.9        52.3        52.5
 52.7        52.8        52.9        53.4        53.5        53.8
 53.9        54.         54.1        54.2        54.3        54.6
 54.7        54.8        55.         55.1        55.2        55.7
 55.9        56.         56.1        56.6        57.2        57.3
 57.5        57.7        57.9        58.1        59.7        60.2
 60.9        61.2        61.6        63.3        64.4        64.8
 66.8        71.9        78.         92.         97.6       ]
smoking_status ['formerly smoked' 'never smoked' 'smokes']
stroke [0 1]
```

In [24]:
```
stroke
```

0	1	88	0	1	1	2	1	3850	240	0	1
1	0	82	0	0	1	3	0	3588	162	1	1
2	1	101	0	1	1	2	0	2483	199	1	1
3	0	70	0	0	1	2	1	3385	218	2	1
4	0	100	1	0	1	3	0	3394	113	1	1
...	...	...	...	...	...	...	...	...	...	...	...
5105	0	101	1	0	1	2	1	1360	162	1	0
5106	0	102	0	0	1	3	1	3030	274	1	0
5107	0	56	0	0	1	3	0	1314	180	1	0
5108	1	72	0	0	1	2	0	3363	129	0	0
5109	0	65	0	0	1	0	1	1454	135	1	0

5110 rows × 11 columns

In [25]:
```
# people who have not had a stroke

stroke_False = stroke[stroke["stroke"] == 0]

stroke_False.head()
```

Out[25]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
249	1	24	0	0	0	4	0	1986	53	1	0
250	1	79	1	0	1	2	1	1604	266	1	0
251	0	29	0	0	0	2	1	2663	49	1	0
252	0	91	0	0	1	2	0	528	233	0	0
253	1	35	0	0	0	1	0	3340	64	1	0

In [26]:
```
print("people who have not had a stroke in percentage-",len(stroke_False)/len(stroke)*100,"%")
```

```
people who have not had a stroke in percentage- 95.12720156555773 %
```

In [28]:
```
# prople who have had a stroke
stroke_True = stroke[stroke["stroke"] == 1]

stroke_True.head()
```

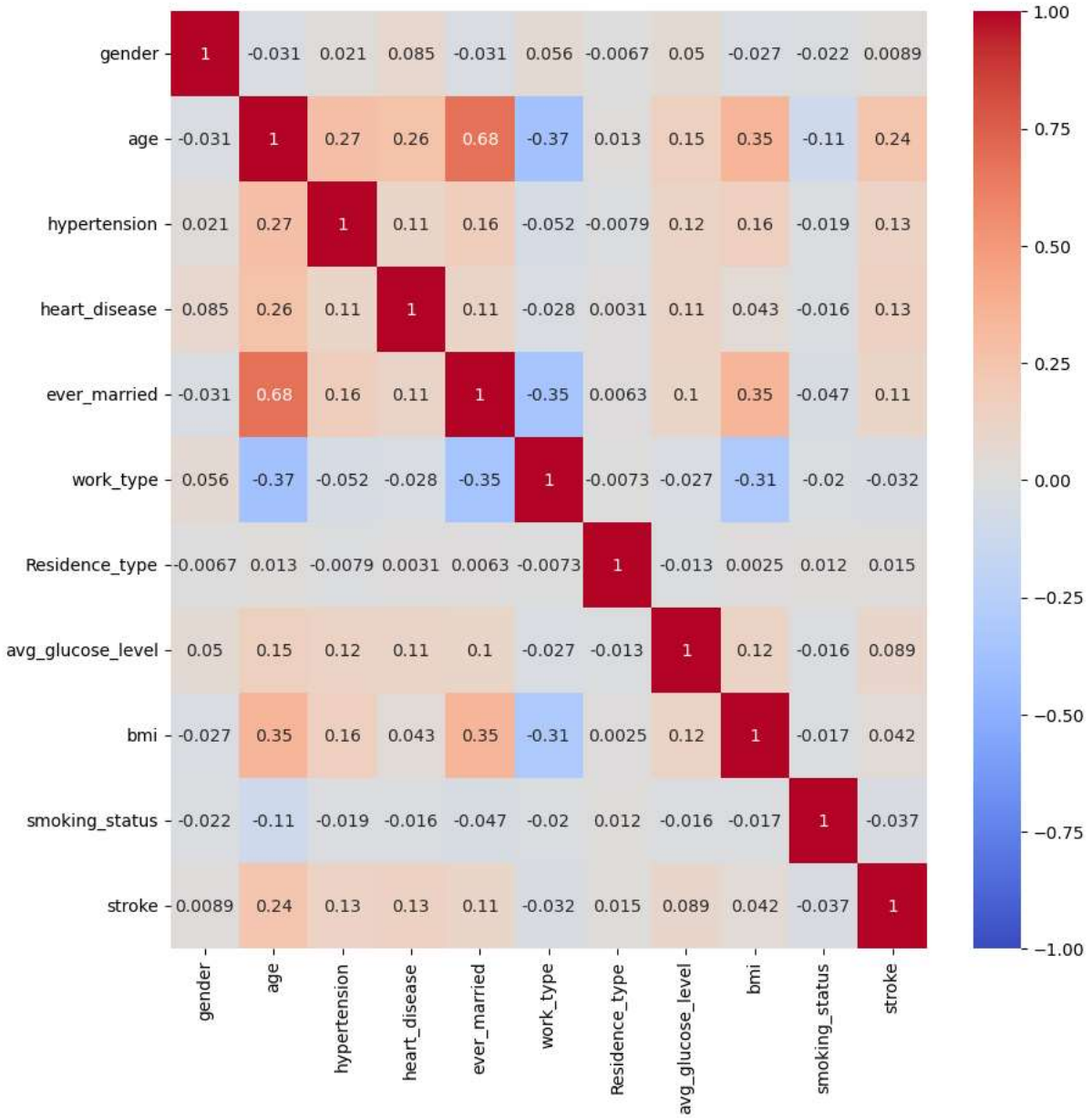
Out[28]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	88	0	1	1	2	1	3850	240	0	1
1	0	82	0	0	1	3	0	3588	162	1	1
2	1	101	0	1	1	2	0	2483	199	1	1
3	0	70	0	0	1	2	1	3385	218	2	1
4	0	100	1	0	1	3	0	3394	113	1	1

In [29]:
```
print("People who have had a stroke in percentage-",len(stroke_True)/len(stroke)*100, "%")
```

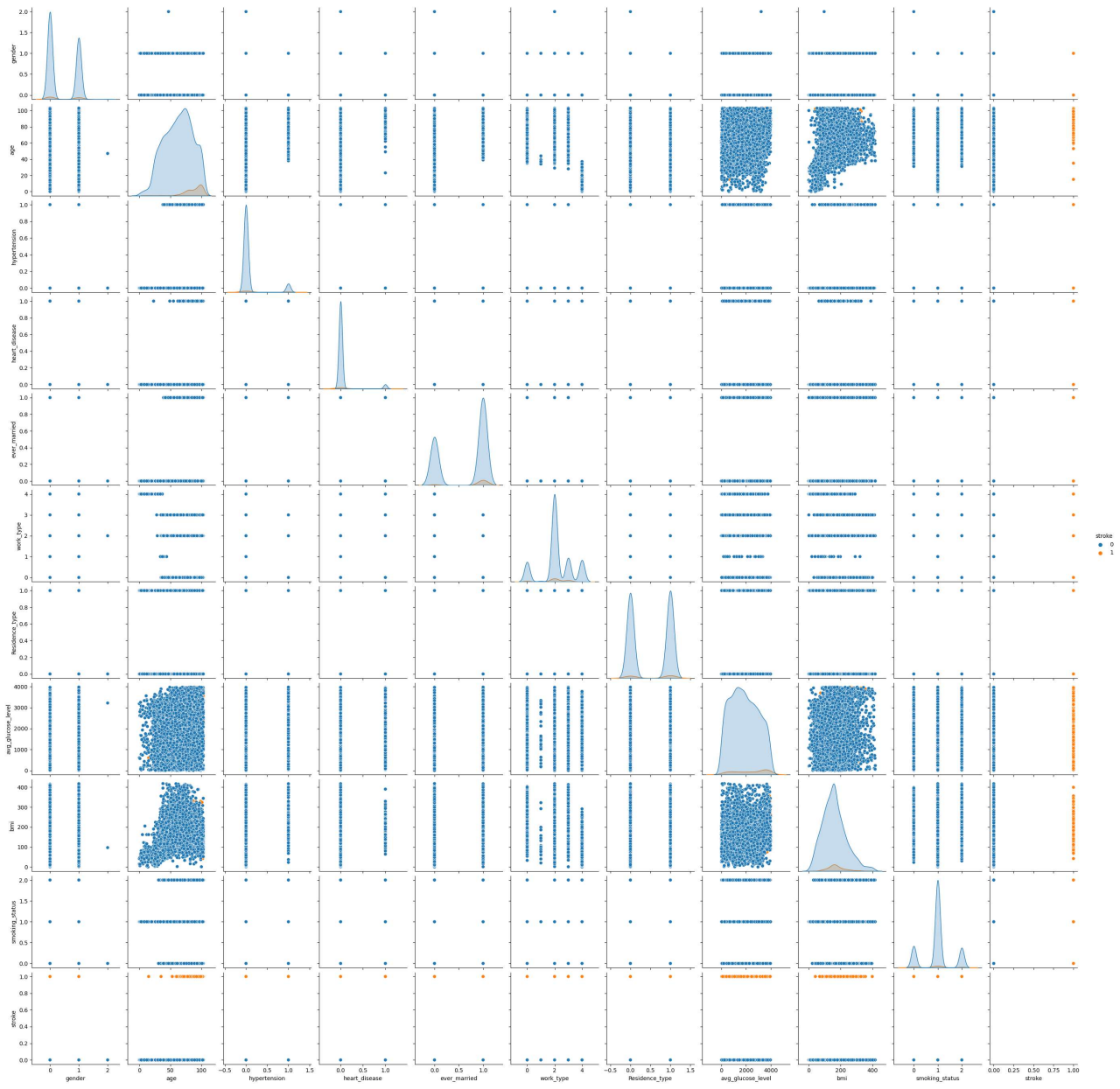
```
People who have had a stroke in percentage- 4.87279843444227 %
```

In [30]:
```
# correlation
plt.figure(figsize=(10,10))
sns.heatmap(stroke.corr(), vmin=-1, cmap="coolwarm", annot=True);
```



In [31]:
```
'pertension','heart_disease','ever_married','work_type','Residence_type','avg_glucose_level','bmi','smoking_status','stroke'])
```

Out[31]: <seaborn.axisgrid.PairGrid at 0x1b4e01a3580>



Testing and Training Data

In [32]:
```
stroke
```

Out[32]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	88	0	1	1	2	1	3850	240	0	1
1	0	82	0	0	1	3	0	3588	162	1	1
2	1	101	0	1	1	2	0	2483	199	1	1
3	0	70	0	0	1	2	1	3385	218	2	1
4	0	100	1	0	1	3	0	3394	113	1	1
...	...	...	...	...	...	...	...	...	...	...	...
5105	0	101	1	0	1	2	1	1360	162	1	0
5106	0	102	0	0	1	3	1	3030	274	1	0
5107	0	56	0	0	1	3	0	1314	180	1	0
5108	1	72	0	0	1	2	0	3363	129	0	0
5109	0	65	0	0	1	0	1	1454	135	1	0

5110 rows × 11 columns

In [33]:
```
# Drop the target label i.e the stroke column
x = stroke.drop(["stroke"],axis = 1)

x
```

Out[33]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	1	88	0	1	1	2	1	3850	240	0
1	0	82	0	0	1	3	0	3588	162	1
2	1	101	0	1	1	2	0	2483	199	1
3	0	70	0	0	1	2	1	3385	218	2
4	0	100	1	0	1	3	0	3394	113	1
...	...	...	...	...	...	...	...	...	...	...
5105	0	101	1	0	1	2	1	1360	162	1
5106	0	102	0	0	1	3	1	3030	274	1
5107	0	56	0	0	1	3	0	1314	180	1
5108	1	72	0	0	1	2	0	3363	129	0
5109	0	65	0	0	1	0	1	1454	135	1

5110 rows × 10 columns

In [34]:
```
y = stroke["stroke"]

y
```

Out[34]:
```
0       1
1       1
2       1
3       1
4       1
       ..
5105    0
5106    0
5107    0
5108    0
5109    0
Name: stroke, Length: 5110, dtype: int64
```

In [35]:
```
# Now we need to splot trained data and split data i.e divide data into training and testing sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

Training Data Using DecisionTreeClassifier

In [36]:
```
# sample data point fr training
x_train.shape
y_train.shape
```

Out[36]: (3577,)

In [37]:
```
# sample data for testing
x_test.shape
y_test.shape
```

Out[37]: (1533,)

In [38]:
```
# Train a decision tree classifier
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
##Instantiate an object out of our class
decision_tree.fit(x_train, y_train)
```

Out[38]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

Evolution model

In [40]:
```
from sklearn.metrics import classification_report , confusion_matrix
from sklearn.metrics import accuracy_score
```

In [44]:
```
# plot the confusion  metrics for the testing data
y_predict_test = decision_tree.predict(x_test)

y_predict_test
```

Out[44]: array([0, 0, 0, ..., 1, 0, 0], dtype=int64)

In [42]:
```
y_test
```

Out[42]:
```
1092    0
3874    0
1799    0
3146    0
524     0
       ..
2679    0
3432    0
386     0
1101    0
913     0
Name: stroke, Length: 1533, dtype: int64
```

In [46]:
```
cm = confusion_matrix(y_test, y_predict_test)
```

In [47]:
```
ax= plt.subplot()
sns.heatmap(cm, annot=True, ax = ax, fmt = 'g'); #annot=True to annotate cells
# labels, title and ticks
ax.set_xlabel('True', fontsize=20)
ax.xaxis.set_label_position('top')
ax.xaxis.set_ticklabels(['Did not suffer stroke','suffered stroke'], fontsize = 10)
ax.xaxis.tick_top()

ax.set_ylabel('Predicted', fontsize=20)
ax.yaxis.set_ticklabels(['Did not suffer stroke', 'suffered stroke'], fontsize = 10)
plt.show()
```

In [48]:
```
print(classification_report(y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94      1446
           1       0.13      0.15      0.14        87

    accuracy                           0.90      1533
   macro avg       0.54      0.55      0.54      1533
weighted avg       0.90      0.90      0.90      1533
```

In [49]:
```
# Training model using RandomForestClasifier
```

In [51]:
```
# Random Forest classifier to improve the model
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier(n_estimators = 150)
RandomForest.fit(x_train, y_train)
```

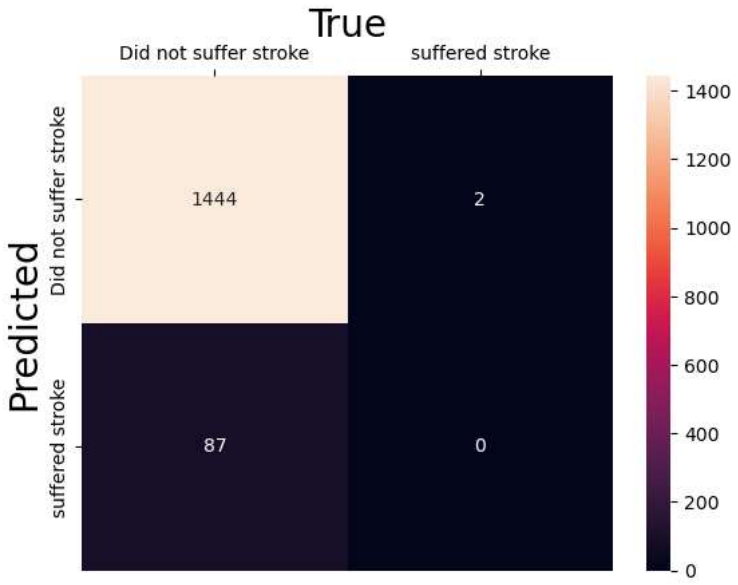
Out[51]:
```
RandomForestClassifier
RandomForestClassifier(n_estimators=150)
```

In [52]:
```
# predicting on test data
y_predict_test = RandomForest.predict(x_test)
```

In [53]:
```
# creating confusion matrix for test prediction
cm = confusion_matrix(y_test, y_predict_test)
```

In [54]:
```
ax= plt.subplot()
sns.heatmap(cm, annot=True, ax = ax, fmt = 'g'); #annot=True to annotate cells
# labels, title and ticks
ax.set_xlabel('True', fontsize=20)
ax.xaxis.set_label_position('top')
ax.xaxis.set_ticklabels(['Did not suffer stroke','suffered stroke'], fontsize = 10)
ax.xaxis.tick_top()

ax.set_ylabel('Predicted', fontsize=20)
ax.yaxis.set_ticklabels(['Did not suffer stroke', 'suffered stroke'], fontsize = 10)
plt.show()
```



In [55]:
```
print(classification_report(y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1446
           1       0.00      0.00      0.00        87

    accuracy                           0.94      1533
   macro avg       0.47      0.50      0.49      1533
weighted avg       0.89      0.94      0.92      1533
```

The accuracy improved from 91% to 95% when RandomForestClassifier was used instead of DecisionTreeClassifier and thus, the accuracy score of RandomForestClassifier is better than that of DecisionTreeClassifier

In [ ]:
```

```