Import Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
```

Read Data

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

```
df = pd.read_csv('dataset-stroke.csv')
```

Preview Dataset

```
df.info()
```

```
<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
```

```
df.isna().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.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12

Clean Data

```
# Drop out ID column
df = df.drop('id',axis=1)
```

```
# Replace Missing Values in column 'bmi' with the average bmi
df.loc[df['bmi'].isnull(),'bmi'] = df.bmi.mean()
```

```
# Clean gender column
df['gender'].value_counts()
```

```
Female    2994
Male      2115
Other        1
Name: gender, dtype: int64
```

```
df = df.drop(df[df['gender'] == 'Other'].index, axis=0)
```

```
# Convert categorical string into numeric
cols = ['gender','ever_married','work_type','Residence_type','smoking_status']

for col in cols:
    print(f"---{col}--- \n{df[col].value_counts()}")
```

```
---gender---
Female    2994
Male      2115
Name: gender, dtype: int64
---ever_married---
Yes    3353
No     1756
Name: ever_married, dtype: int64
---work_type---
Private          2924
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: work_type, dtype: int64
---Residence_type---
Urban    2596
```

```
Rural    2513
Name: Residence_type, dtype: int64
---smoking_status---
```

```
for col in cols:
    label = LabelEncoder().fit_transform(df[col])
    df.drop(col, axis= 1, inplace=True)
    df[col] = label
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5109 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   age                5109 non-null   float64
 1   hypertension       5109 non-null   int64  
 2   heart_disease      5109 non-null   int64  
 3   avg_glucose_level  5109 non-null   float64
 4   bmi                5109 non-null   float64
 5   stroke             5109 non-null   int64  
 6   gender             5109 non-null   int64  
 7   ever_married       5109 non-null   int64  
 8   work_type          5109 non-null   int64  
 9   Residence_type     5109 non-null   int64  
 10  smoking_status     5109 non-null   int64  
dtypes: float64(3), int64(8)
memory usage: 479.0 KB
```

Split Data

```
X = df.drop(['stroke'],axis=1)
y = df['stroke']
# split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state = 66)
```

Fit models to data

```
logit_model = LogisticRegression()
logit_model.fit(X_train, y_train)

p = logit_model.predict(X_test)
logit_model.score(X_test, y_test)
```

```
0.9500978473581213

/opt/python/envs/default/lib/python3.8/site-packages/sklearn/linear_model/_logi
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
  n_iter_i = _check_optimize_result(
```

```
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)

p = dt_model.predict(X_test)
dt_model.score(X_test, y_test)
```

```
0.9060665362035225
```

```
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

p = rf_model.predict(X_test)
rf_model.score(X_test, y_test)
```

```
0.949119373776908
```

```
from sklearn.metrics import make_scorer, precision_score, recall_score, accuracy_
```

```
print(classification_report(y_test, p, target_names=['stroke','not']))
```

```
              precision    recall  f1-score   support

      stroke       0.95      1.00      0.97       971
         not       0.00      0.00      0.00        51

    accuracy                           0.95      1022
   macro avg       0.48      0.50      0.49      1022
weighted avg       0.90      0.95      0.93      1022
```