Stroke Prediction of Adult Women

Matthew Galvez
Rayna Sevilla

What is the Dataset about?

This dataset is used to predict whether a patient is likely to get a stroke based on the input parameters like gender, age, various diseases, and smoking status.

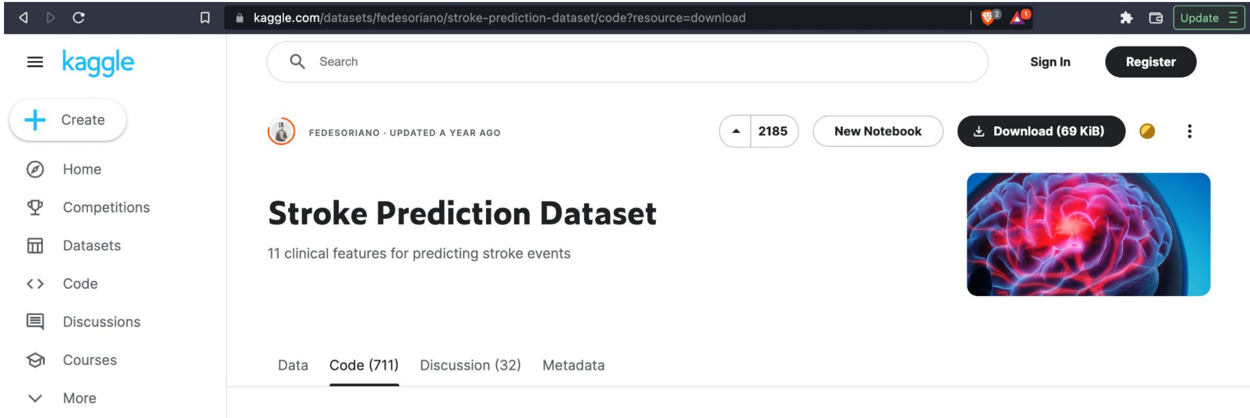
Each row in the data provides relevant information about the patient.

Research Question:

What are the factors for adult women that are most likely to result in a stroke?

Data Overview

- 5110 patients
- Patient information
  - Age
  - Bmi
  - Heart Disease
  - Smoking Status
  - Etc…
- If the patient suffered a stroke (ischemic or hemorrhagic)



kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/code?resource=download

FEDESORIANO · UPDATED A YEAR AGO

Stroke Prediction Dataset

11 clinical features for predicting stroke events

www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/code?resource=download

Risk Factors for Stroke[1]

Risk Factors Include, but not limited to:

- **High blood pressure. (Hypertension)**
- **Heart disease.** Heart disease and stroke have many of the same risk factors.
- **Smoking.** Smoking almost doubles your risk for an ischemic stroke.
- **Obesity**
- **Older age.** For each decade of life after age 55, your chance of having a stroke more than doubles.
- **Gender.** Stroke occurs more often in men.

Expectations:

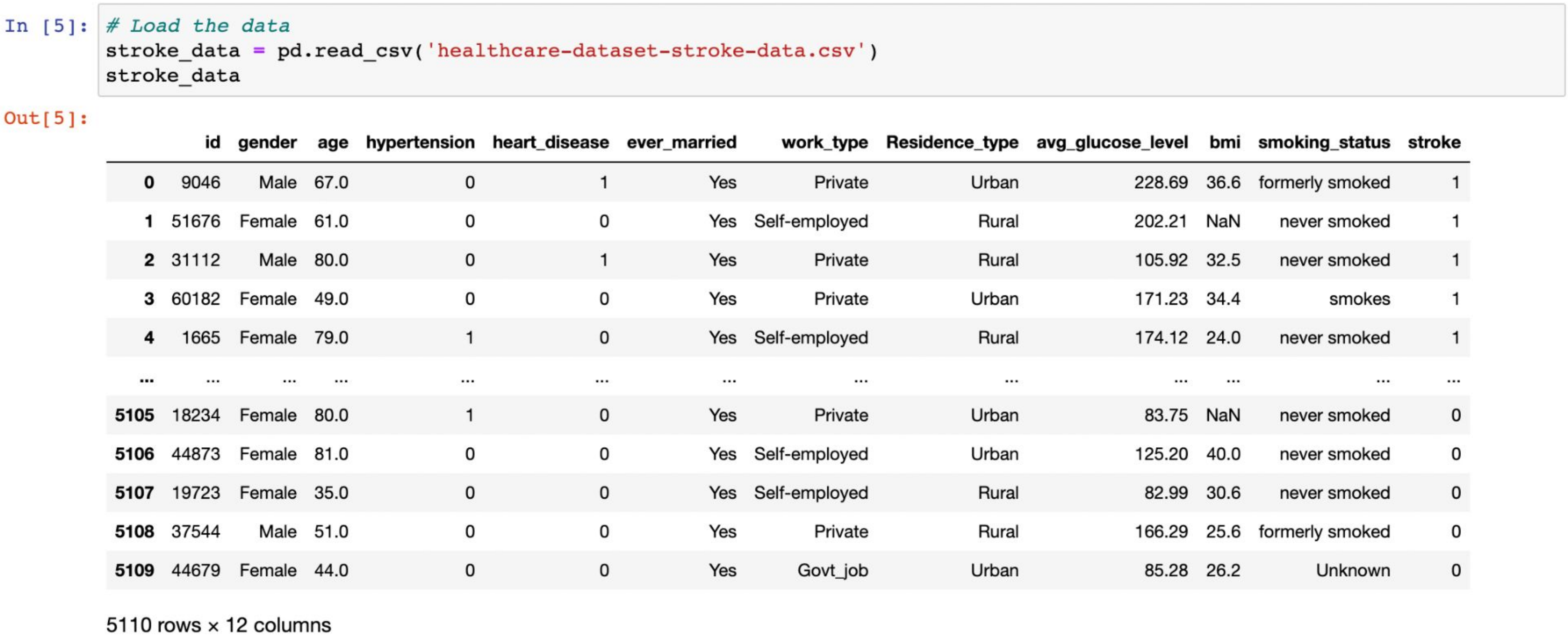
We believe 'Age', 'bmi', 'heart_disease', 'hypertension', and smoking 'smoking_status' to produce the best logistic regression model

Evaluation Process:

We will split our data into train/test sets and then see if our logistic regression models produce a high accuracy

We expect our 'ideal' group of independent variables to produce a high accuracy

Loading the Data



In [5]:

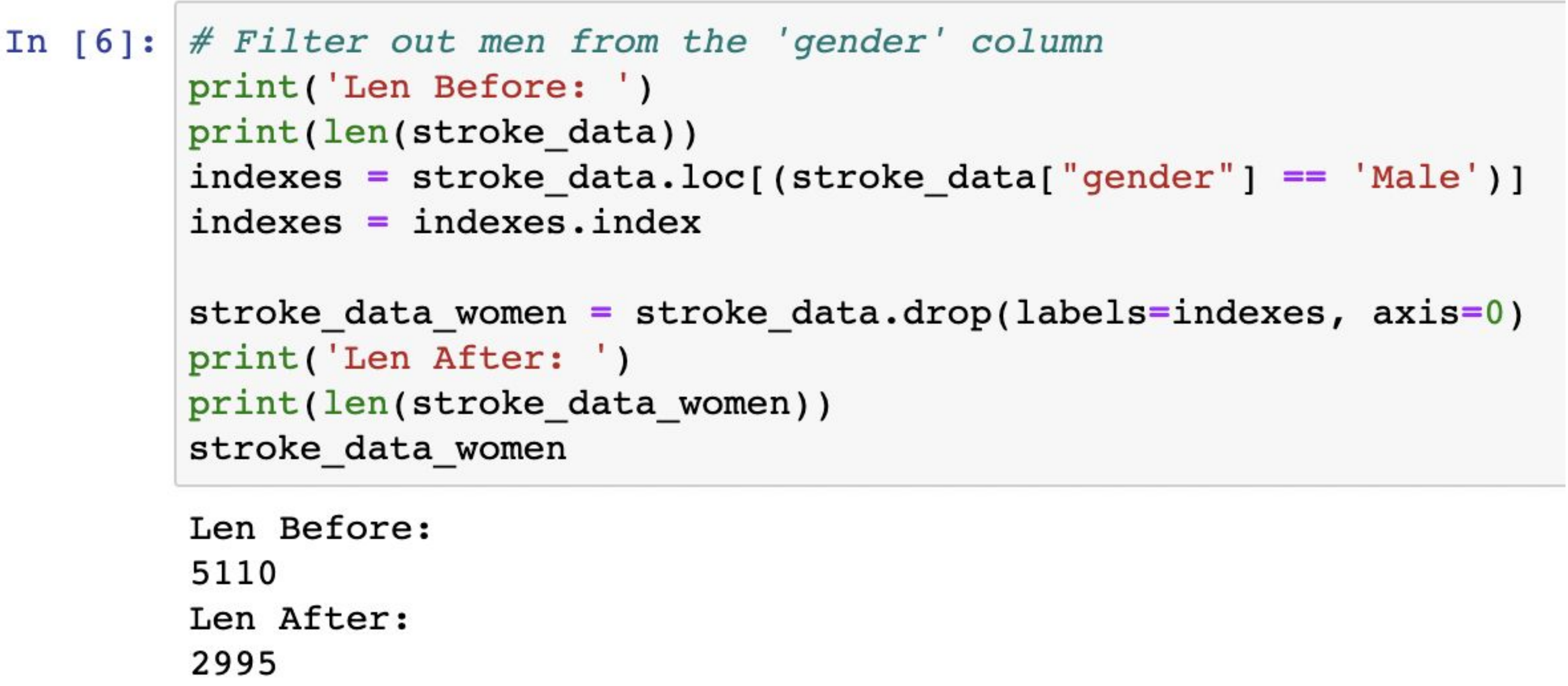
```python
# Load the data
stroke_data = pd.read_csv('healthcare-dataset-stroke-data.csv')
stroke_data
```

Out[5]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

5110 rows × 12 columns

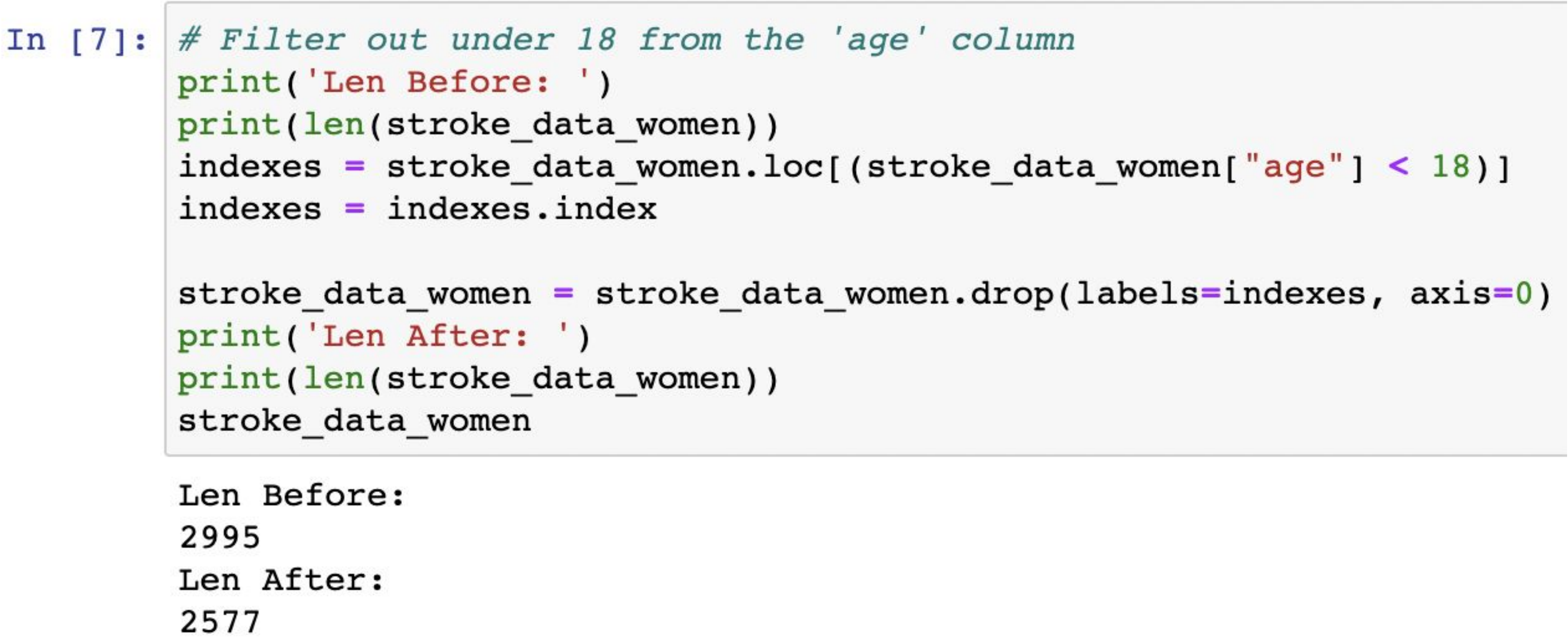
Data Cleanup



```
In [6]: # Filter out men from the 'gender' column
        print('Len Before: ')
        print(len(stroke_data))
        indexes = stroke_data.loc[(stroke_data["gender"] == 'Male')]
        indexes = indexes.index

        stroke_data_women = stroke_data.drop(labels=indexes, axis=0)
        print('Len After: ')
        print(len(stroke_data_women))
        stroke_data_women
```

```
Len Before:
5110
Len After:
2995
```

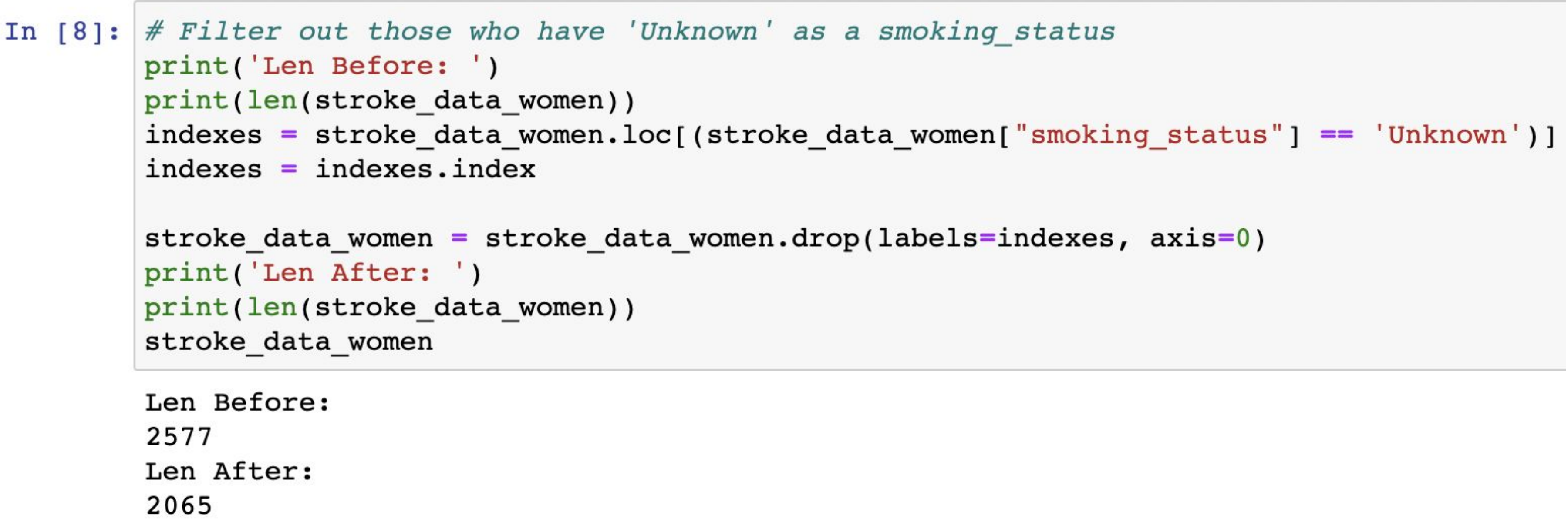
Data Cleanup - II



```
In [7]: # Filter out under 18 from the 'age' column
        print('Len Before: ')
        print(len(stroke_data_women))
        indexes = stroke_data_women.loc[(stroke_data_women["age"] < 18)]
        indexes = indexes.index

        stroke_data_women = stroke_data_women.drop(labels=indexes, axis=0)
        print('Len After: ')
        print(len(stroke_data_women))
        stroke_data_women
```

```
Len Before:
2995
Len After:
2577
```

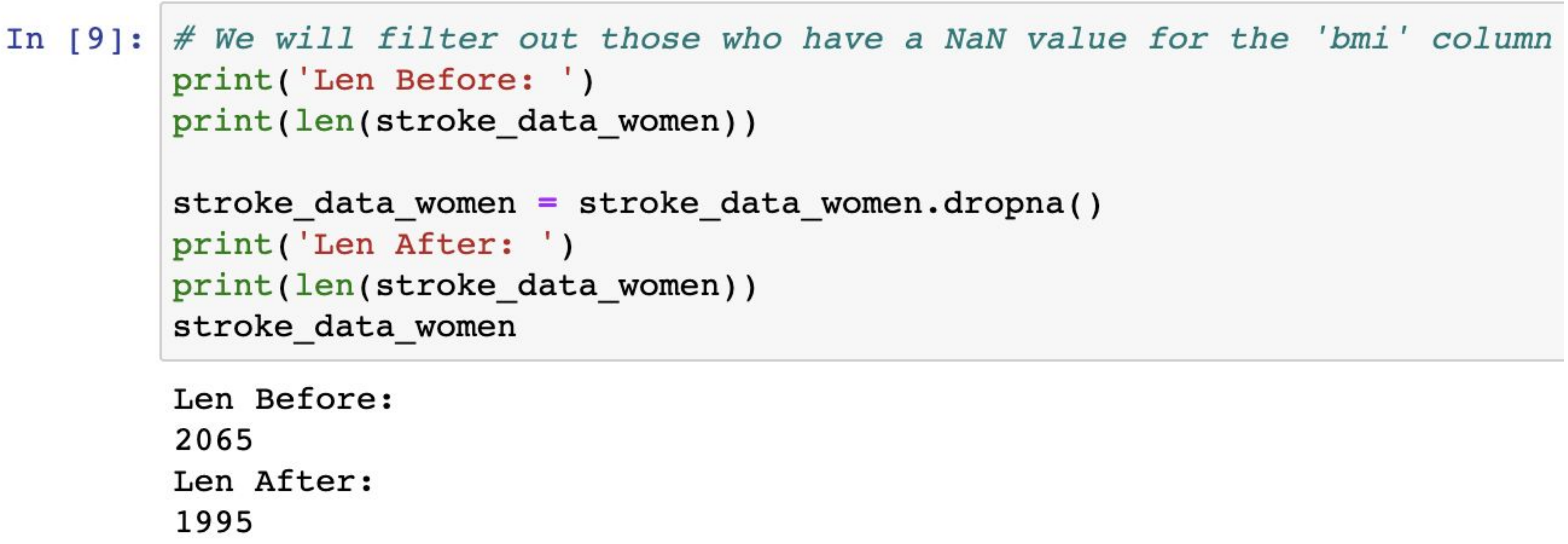
Data Cleanup - III



```
# Filter out those who have 'Unknown' as a smoking_status
print('Len Before: ')
print(len(stroke_data_women))
indexes = stroke_data_women.loc[(stroke_data_women["smoking_status"] == 'Unknown')]
indexes = indexes.index

stroke_data_women = stroke_data_women.drop(labels=indexes, axis=0)
print('Len After: ')
print(len(stroke_data_women))
stroke_data_women
```

```
Len Before:
2577
Len After:
2065
```

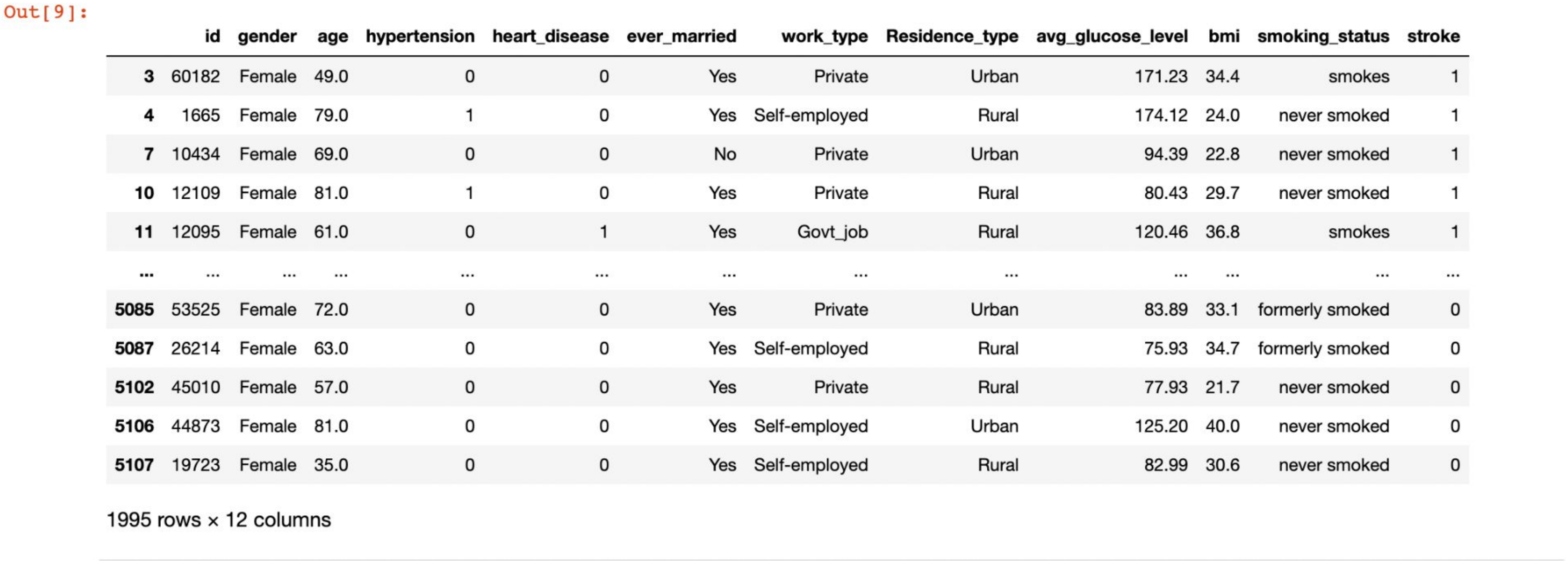
Data Cleanup - IV



```
# We will filter out those who have a NaN value for the 'bmi' column
print('Len Before: ')
print(len(stroke_data_women))

stroke_data_women = stroke_data_women.dropna()
print('Len After: ')
print(len(stroke_data_women))
stroke_data_women
```

```
Len Before:
2065
Len After:
1995
```

Final Product



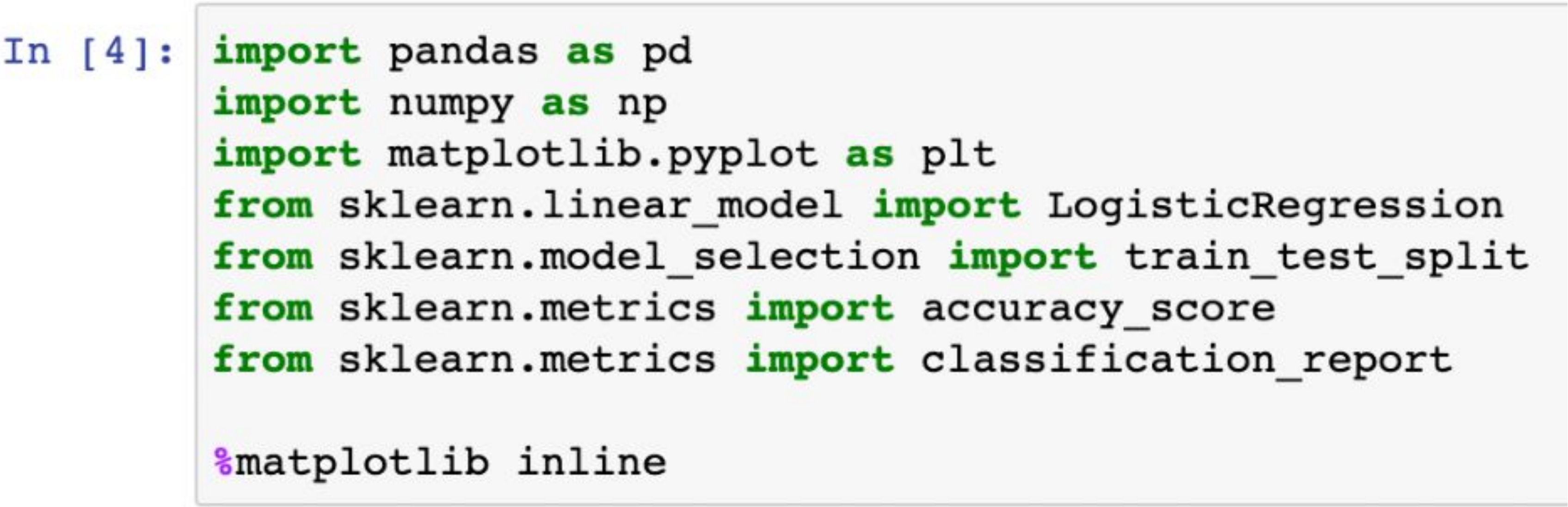
Out[9]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
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
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
10	12109	Female	81.0	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
11	12095	Female	61.0	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5085	53525	Female	72.0	0	0	Yes	Private	Urban	83.89	33.1	formerly smoked	0
5087	26214	Female	63.0	0	0	Yes	Self-employed	Rural	75.93	34.7	formerly smoked	0
5102	45010	Female	57.0	0	0	Yes	Private	Rural	77.93	21.7	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0

1995 rows × 12 columns

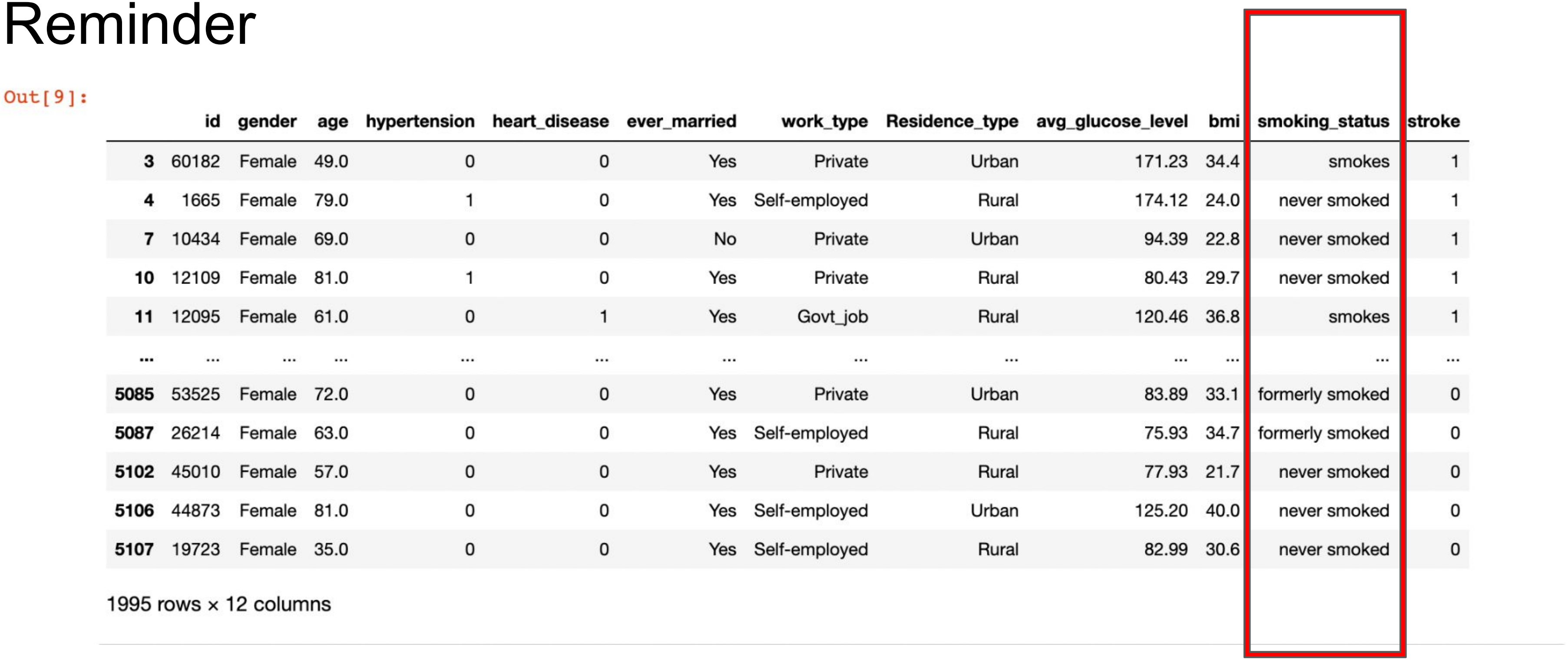
Implementation

Imported Libraries



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

%matplotlib inline
```

Reminder

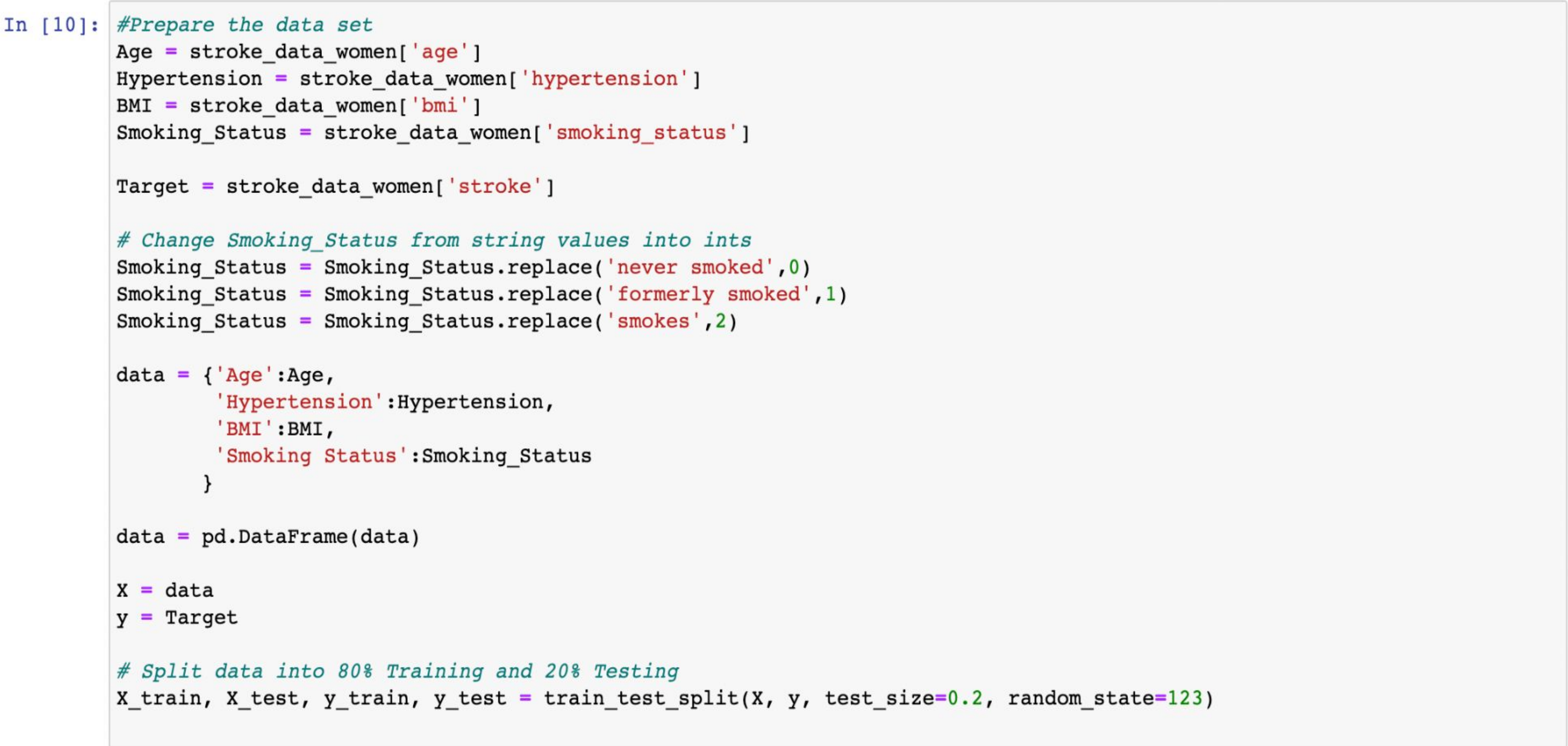


Out[9]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
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
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
10	12109	Female	81.0	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
11	12095	Female	61.0	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5085	53525	Female	72.0	0	0	Yes	Private	Urban	83.89	33.1	formerly smoked	0
5087	26214	Female	63.0	0	0	Yes	Self-employed	Rural	75.93	34.7	formerly smoked	0
5102	45010	Female	57.0	0	0	Yes	Private	Rural	77.93	21.7	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0

1995 rows × 12 columns

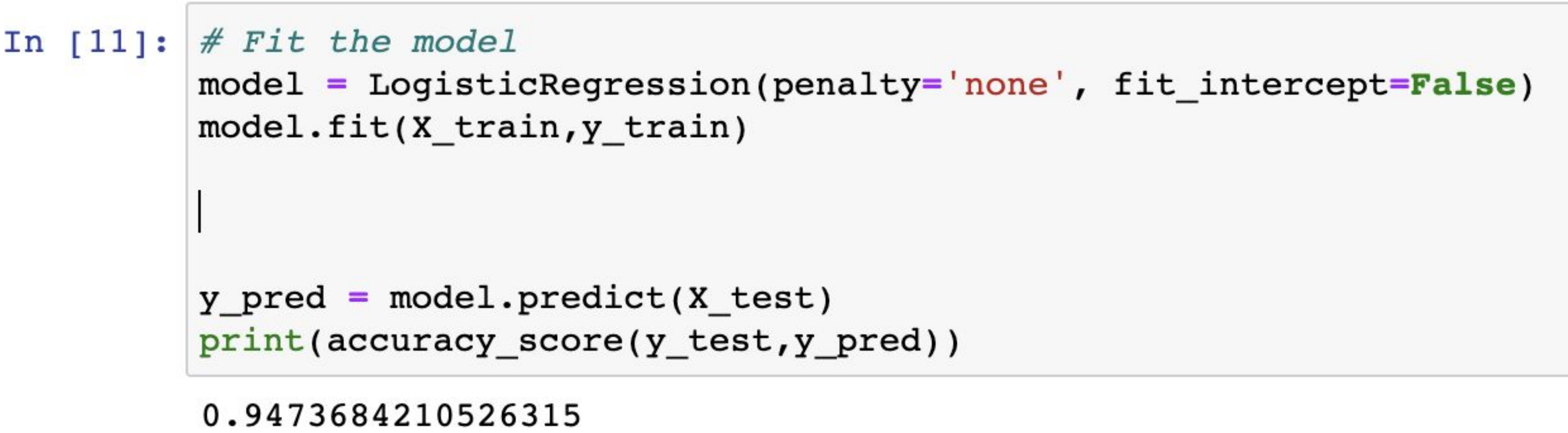
Fitting our Logistic Regression Model



In [10]:

```
#Prepare the data set
Age = stroke_data_women['age']
Hypertension = stroke_data_women['hypertension']
BMI = stroke_data_women['bmi']
Smoking_Status = stroke_data_women['smoking_status']

Target = stroke_data_women['stroke']

# Change Smoking_Status from string values into ints
Smoking_Status = Smoking_Status.replace('never smoked',0)
Smoking_Status = Smoking_Status.replace('formerly smoked',1)
Smoking_Status = Smoking_Status.replace('smokes',2)

data = {'Age':Age,
        'Hypertension':Hypertension,
        'BMI':BMI,
        'Smoking Status':Smoking_Status
       }

data = pd.DataFrame(data)

X = data
y = Target

# Split data into 80% Training and 20% Testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

Fitting our Logistic Regression Model - II

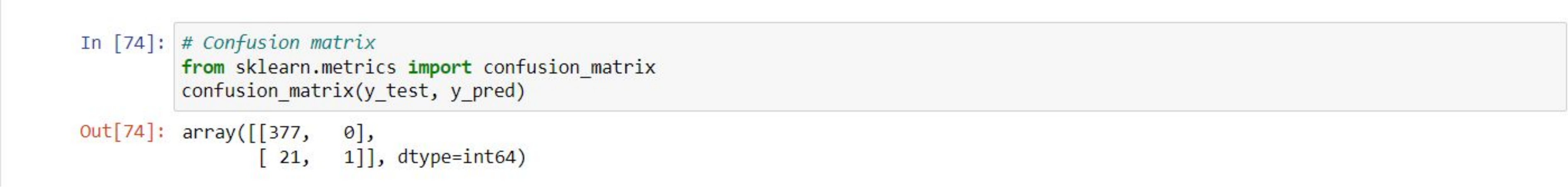


In [11]:

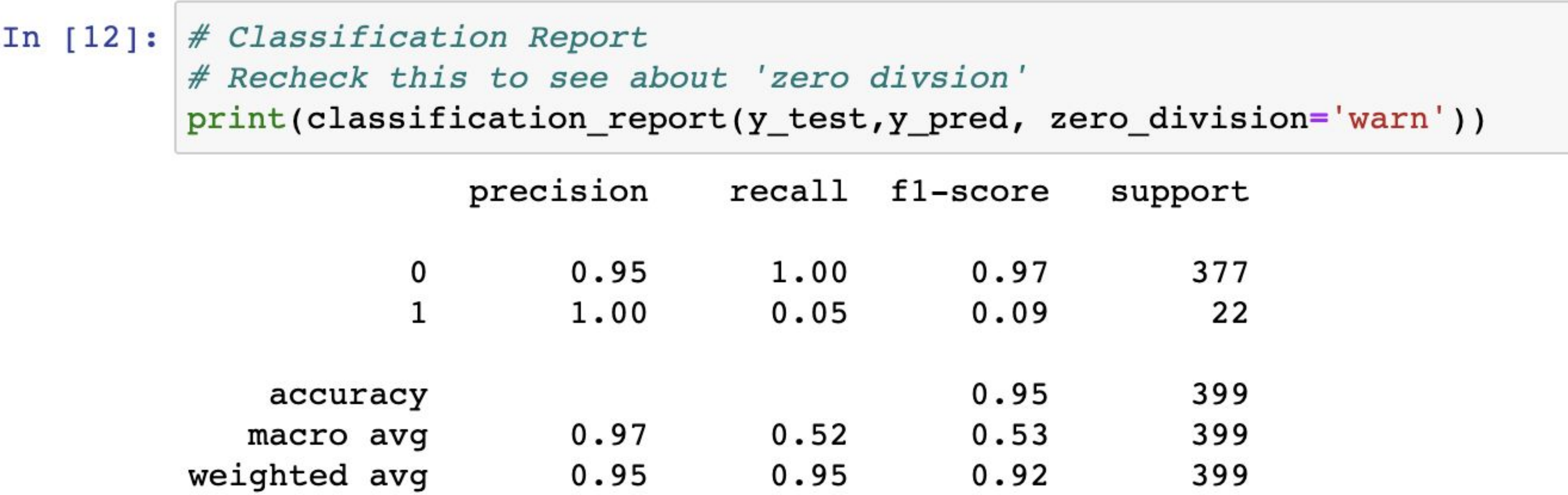
```
# Fit the model
model = LogisticRegression(penalty='none', fit_intercept=False)
model.fit(X_train,y_train)



y_pred = model.predict(X_test)
print(accuracy_score(y_test,y_pred))
```

```
0.9473684210526315
```

Confusion Matrix & Classification Report

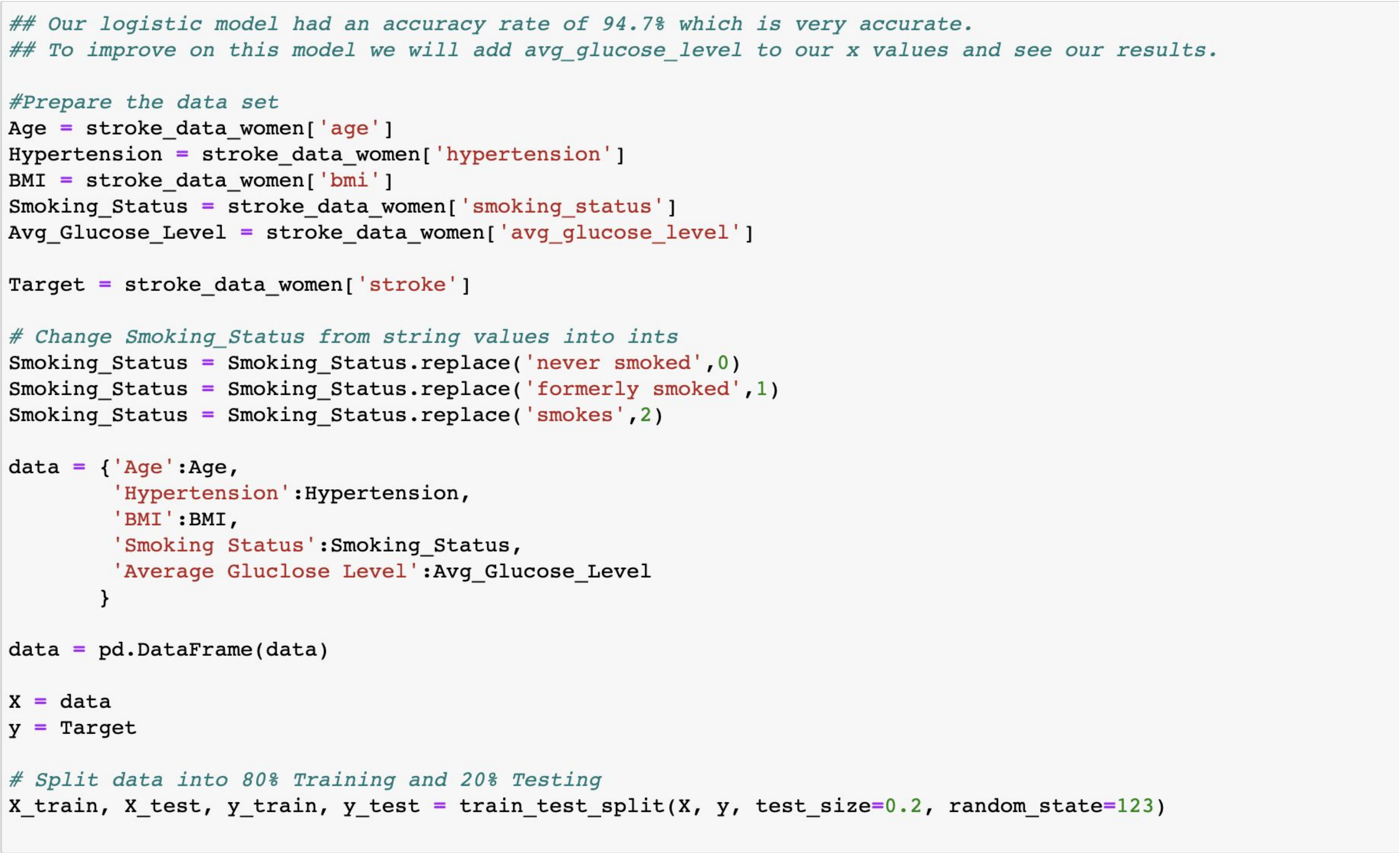


```
In [74]: # Confusion matrix
         from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test, y_pred)

Out[74]: array([[377,   0],
                [ 21,   1]], dtype=int64)
```



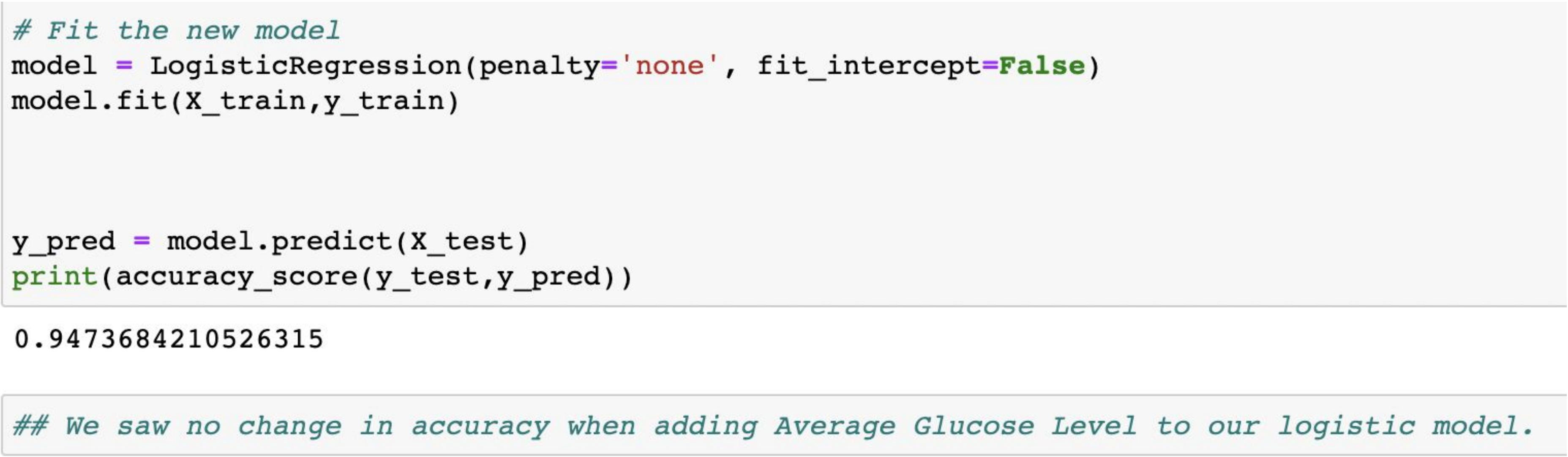
```
In [12]: # Classification Report
         # Recheck this to see about 'zero divsion'
         print(classification_report(y_test,y_pred, zero_division='warn'))

              precision    recall  f1-score   support

           0       0.95      1.00      0.97       377
           1       1.00      0.05      0.09        22

    accuracy                           0.95       399
   macro avg       0.97      0.52      0.53       399
weighted avg       0.95      0.95      0.92       399
```

Improving The Model - I



```
## Our logistic model had an accuracy rate of 94.7% which is very accurate.
## To improve on this model we will add avg_glucose_level to our x values and see our results.

#Prepare the data set
Age = stroke_data_women['age']
Hypertension = stroke_data_women['hypertension']
BMI = stroke_data_women['bmi']
Smoking_Status = stroke_data_women['smoking_status']
Avg_Glucose_Level = stroke_data_women['avg_glucose_level']

Target = stroke_data_women['stroke']

# Change Smoking_Status from string values into ints
Smoking_Status = Smoking_Status.replace('never smoked',0)
Smoking_Status = Smoking_Status.replace('formerly smoked',1)
Smoking_Status = Smoking_Status.replace('smokes',2)

data = {'Age':Age,
        'Hypertension':Hypertension,
        'BMI':BMI,
        'Smoking Status':Smoking_Status,
        'Average Gluclose Level':Avg_Glucose_Level
       }

data = pd.DataFrame(data)

X = data
y = Target

# Split data into 80% Training and 20% Testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

Improving The Model - II

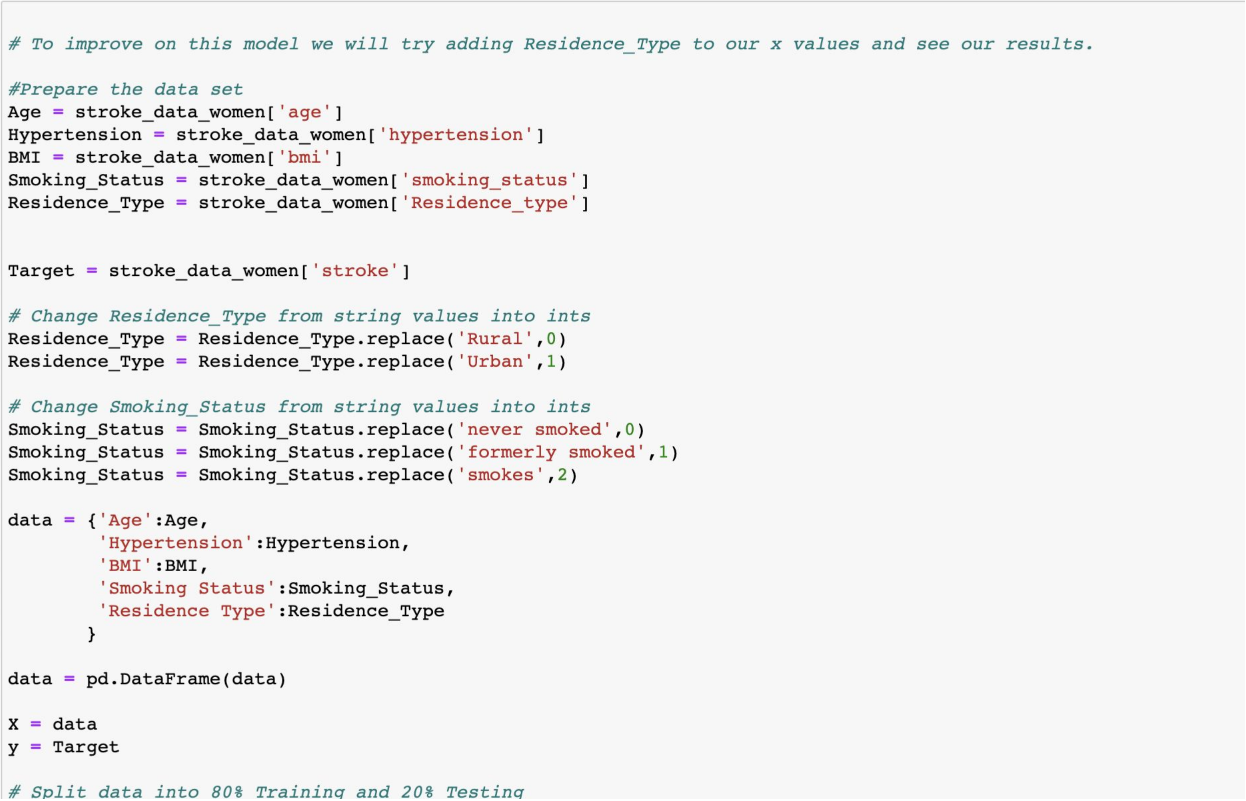


```
# Fit the new model
model = LogisticRegression(penalty='none', fit_intercept=False)
model.fit(X_train,y_train)



y_pred = model.predict(X_test)
print(accuracy_score(y_test,y_pred))
```

```
0.9473684210526315
```

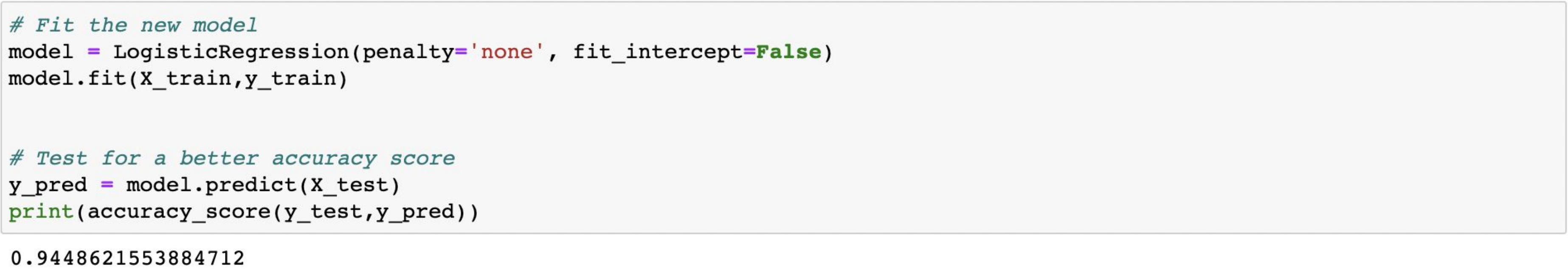
```
## We saw no change in accuracy when adding Average Glucose Level to our logistic model.
```

Improving The Model - III



```
# To improve on this model we will try adding Residence_Type to our x values and see our results.

#Prepare the data set
Age = stroke_data_women['age']
Hypertension = stroke_data_women['hypertension']
BMI = stroke_data_women['bmi']
Smoking_Status = stroke_data_women['smoking_status']
Residence_Type = stroke_data_women['Residence_type']


Target = stroke_data_women['stroke']

# Change Residence_Type from string values into ints
Residence_Type = Residence_Type.replace('Rural',0)
Residence_Type = Residence_Type.replace('Urban',1)

# Change Smoking_Status from string values into ints
Smoking_Status = Smoking_Status.replace('never smoked',0)
Smoking_Status = Smoking_Status.replace('formerly smoked',1)
Smoking_Status = Smoking_Status.replace('smokes',2)

data = {'Age':Age,
        'Hypertension':Hypertension,
        'BMI':BMI,
        'Smoking Status':Smoking_Status,
        'Residence Type':Residence_Type
       }

data = pd.DataFrame(data)

X = data
y = Target

# Split data into 80% Training and 20% Testing
```

Improving The Model - IV



```
# Fit the new model
model = LogisticRegression(penalty='none', fit_intercept=False)
model.fit(X_train,y_train)


# Test for a better accuracy score
y_pred = model.predict(X_test)
print(accuracy_score(y_test,y_pred))
```

```
0.9448621553884712
```

We saw no change in accuracy when adding Residence_Type to our logistic model.

Conclusion

- Our logistic model had an accuracy rate of 94.7% which is very accurate, when using age, hypertension, bmi, and smoking_status as independent variables.

- We have no false positives and very few false negatives for predicting that a patient WILL have a stroke.

References

1. *Risk factors for stroke*. Johns Hopkins Medicine. (2021, November 15). Retrieved July 8, 2022, from https://www.hopkinsmedicine.org/health/conditions-and-diseases/stroke/risk-factors-for-stroke