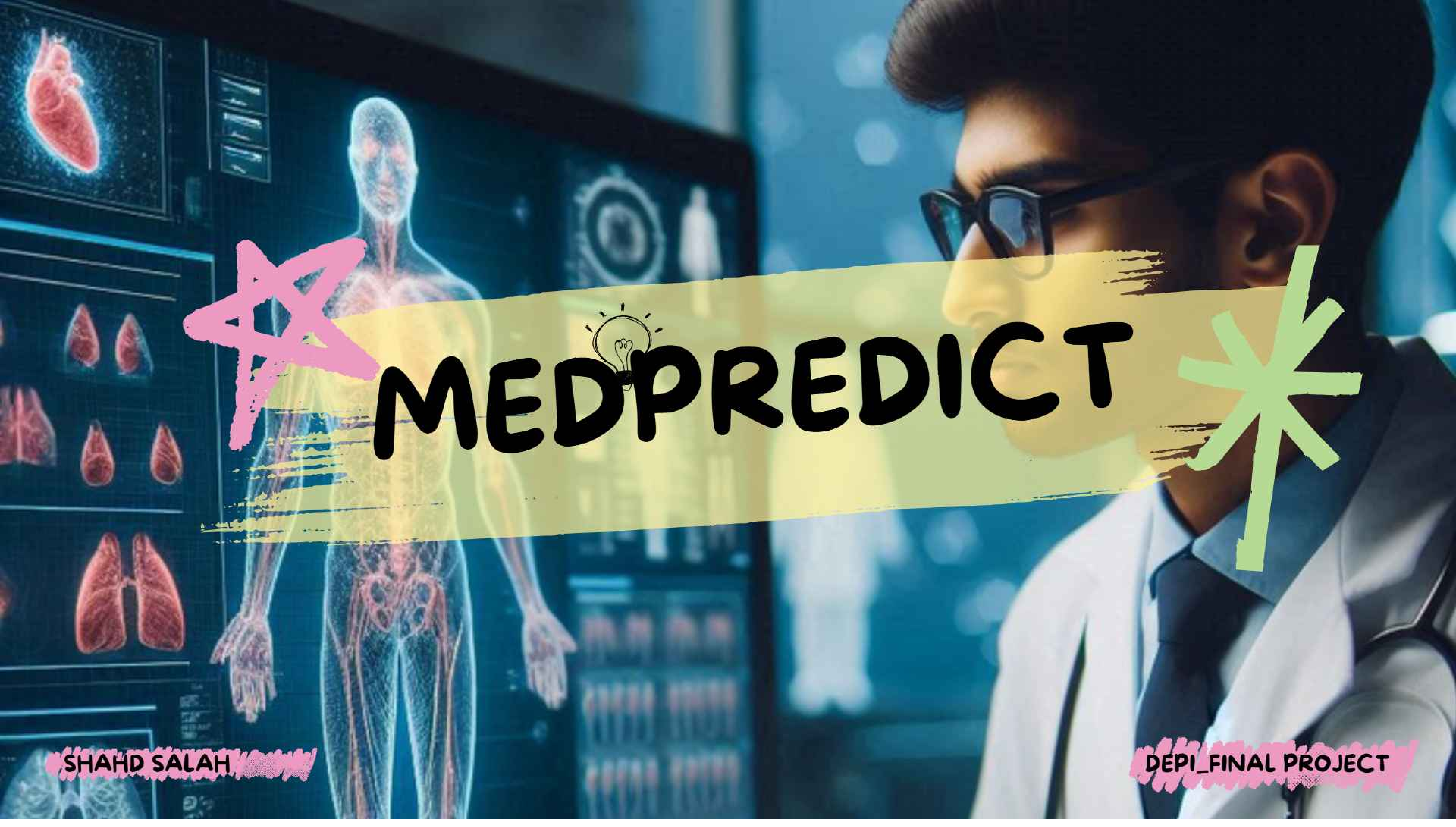


MEDPREDICT

SHAHD SALAH

DEPI_FINAL PROJECT

TABLE OF CONTENTS

INTRODUCTION

DATA LINKS

THE PROCESS

1-BREAST CANCER

2-DIABETES

3-STROKE

WHAT IS IN THE CODE

CONCLUSION

INTRODUCTION

I HAVE DEVELOPED THREE SEPARATE MACHINE LEARNING-POWERED WEBSITES DESIGNED TO PREDICT STROKE, DIABETES, AND BREAST CANCER. EACH WEBSITE INTEGRATES USER-FRIENDLY INTERFACES BUILT WITH GRADIO, ALLOWING USERS TO INPUT THEIR PERSONAL HEALTH DATA AND RECEIVE A PREDICTION FOR EACH DISEASE. THE MODELS BEHIND THE SITES EMPLOY ADVANCED TECHNIQUES LIKE ENSEMBLE METHODS AND STACKING CLASSIFIERS TO ENSURE ACCURACY AND RELIABILITY.

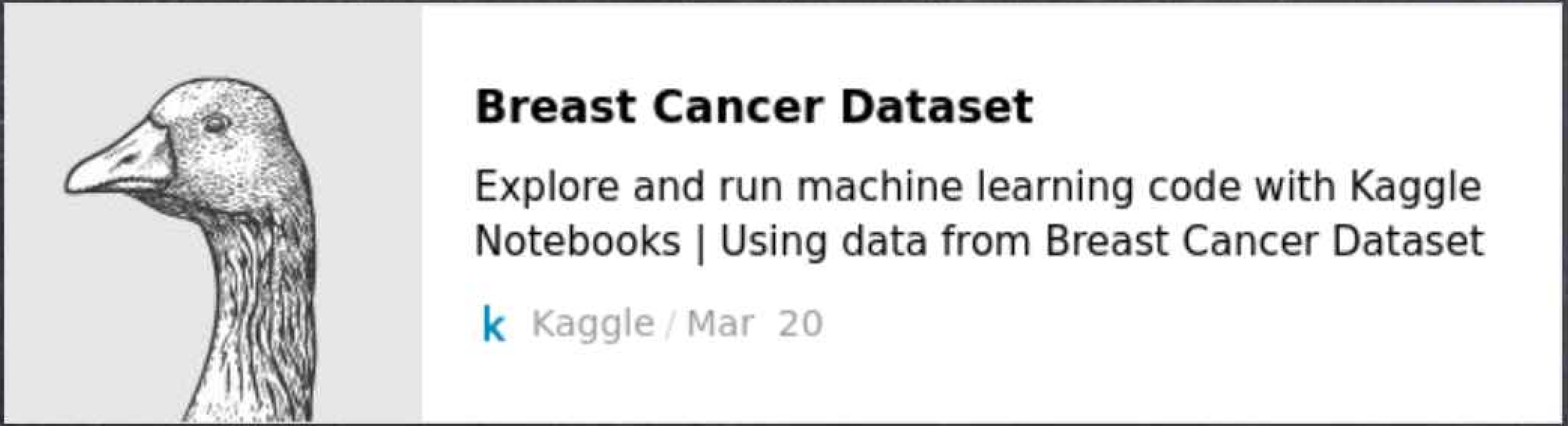
these websites allow users to input relevant medical and lifestyle data, offering predictions based on machine learning models that include ensemble methods and deep learning architectures. Each model has been carefully trained and fine-tuned to provide high accuracy, making the applications practical tools for preliminary health risk assessments. The user interface is designed to be intuitive, ensuring accessibility to a wide audience while maintaining robust predictive performance.

DATA LINKS

diabetes.csv

Kaggle is the world's largest data science community with powerful tools and resources to help you achieve your data science goals.

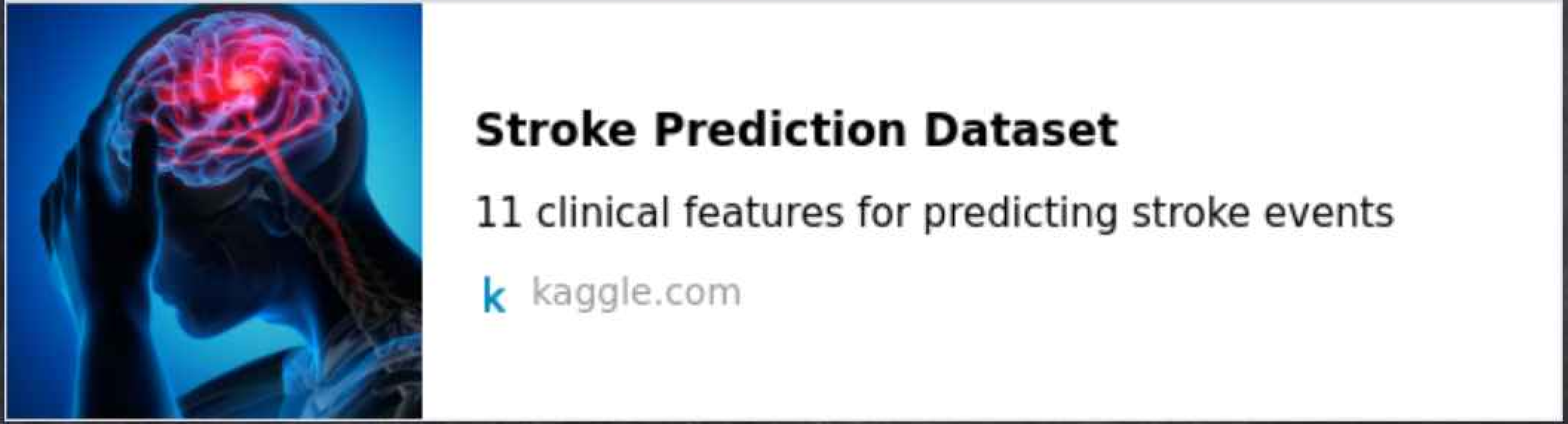
k kaggle.com



Breast Cancer Dataset

Explore and run machine learning code with Kaggle Notebooks | Using data from Breast Cancer Dataset

k Kaggle / Mar 20



Stroke Prediction Dataset

11 clinical features for predicting stroke events

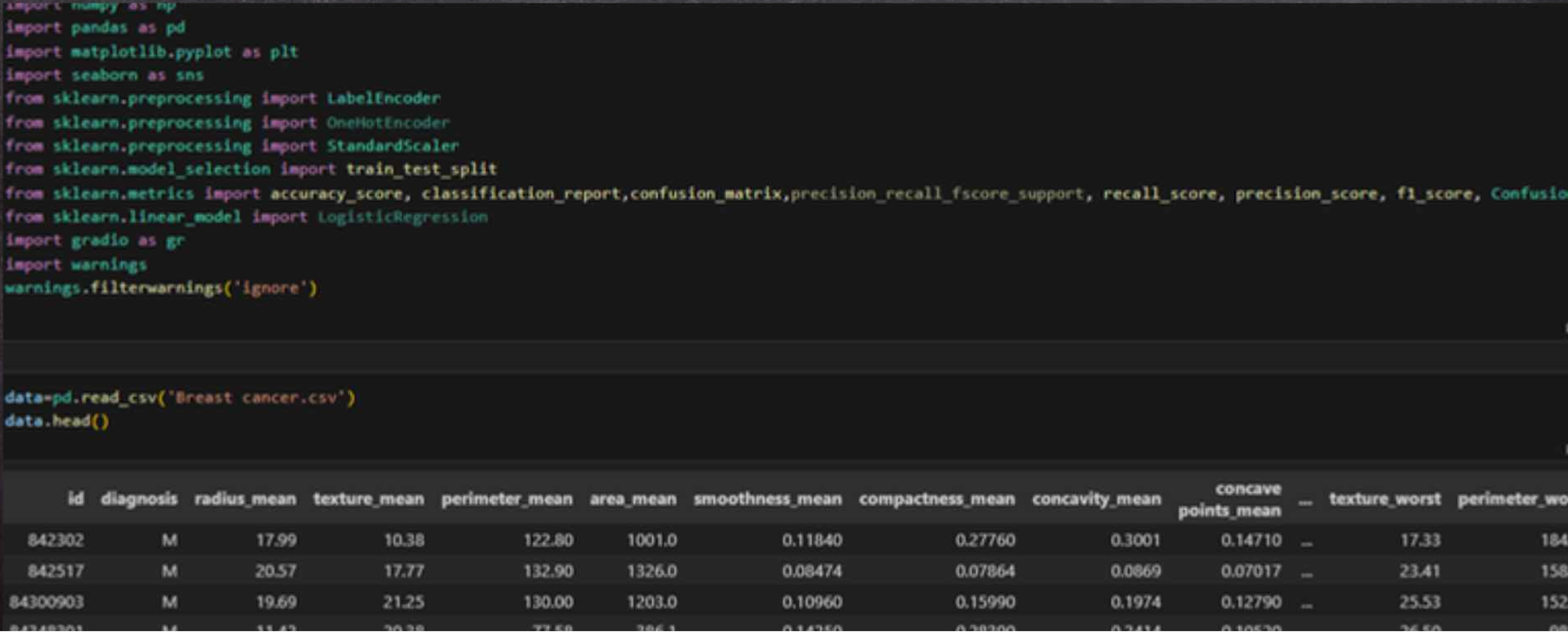
k kaggle.com

THE PROCESS

- DATA PREPROCESSING
- MODELS USED
- INTERFACE

I-BREAST CANCER

ADD LIBRARIES AND FILE

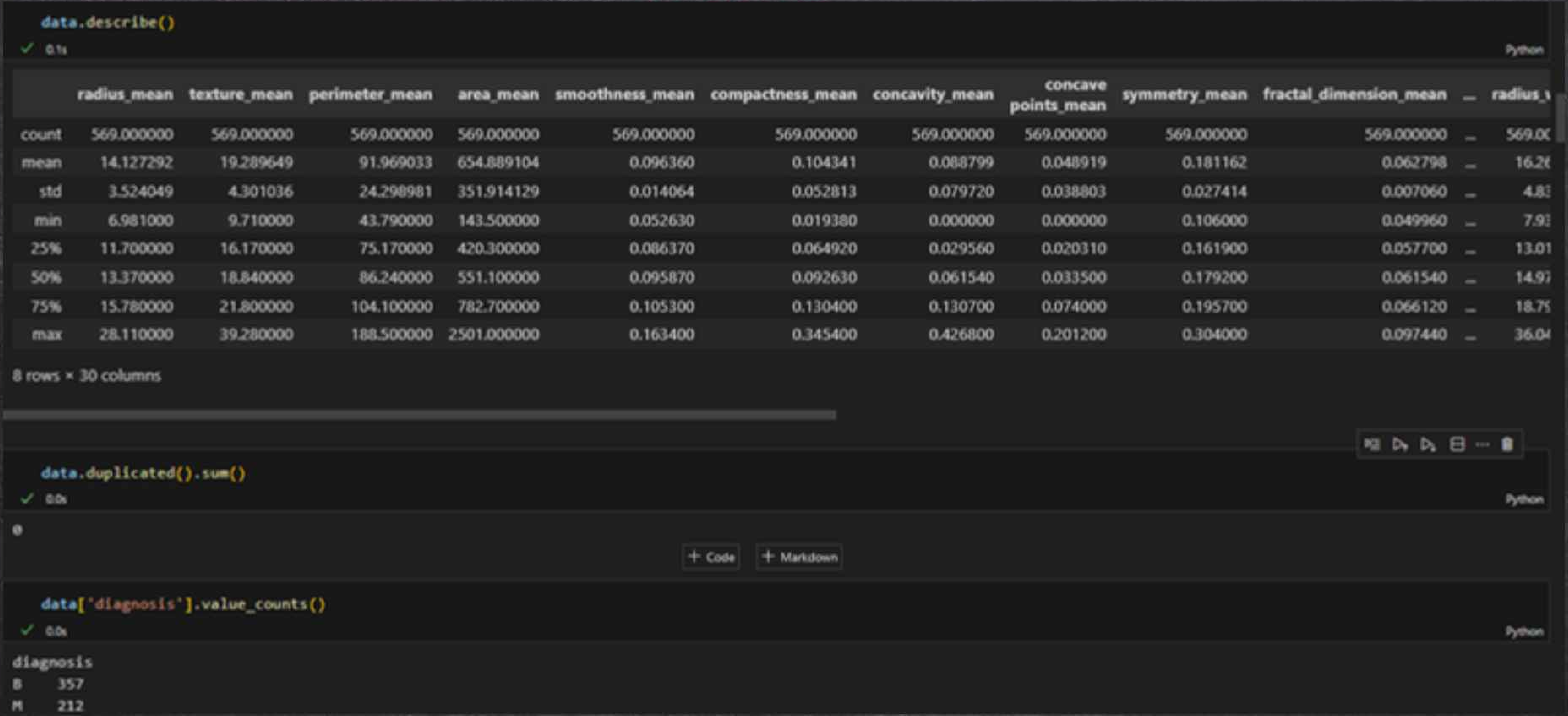


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,confusion_matrix,precision_recall_fscore_support, recall_score, precision_score, f1_score, Confusio
from sklearn.linear_model import LogisticRegression
import gradio as gr
import warnings
warnings.filterwarnings('ignore')
```

```
data=pd.read_csv('Breast cancer.csv')
data.head()
```

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_wo
842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	17.33	184
842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	23.41	158
84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	25.53	152
84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	26.50	98

2-ANALYZE DATA USING INFO, DESCRIPTION, VALUE_COUNTS AND DUPLICATED



```python
data.describe()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean	...	radius_
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.0
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	16.2
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	4.8
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	7.9
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	13.01
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	14.9
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	18.7
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	36.0

8 rows × 30 columns

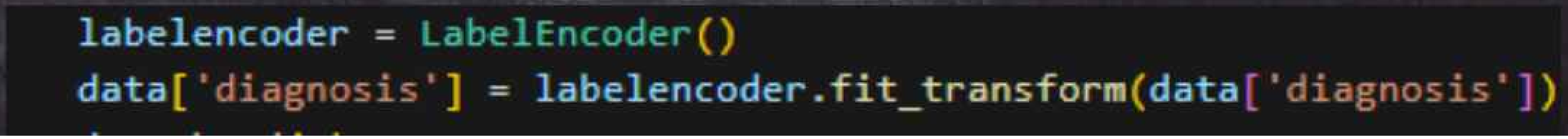
```python
data.duplicated().sum()
```

```
0
```

```python
data['diagnosis'].value_counts()
```

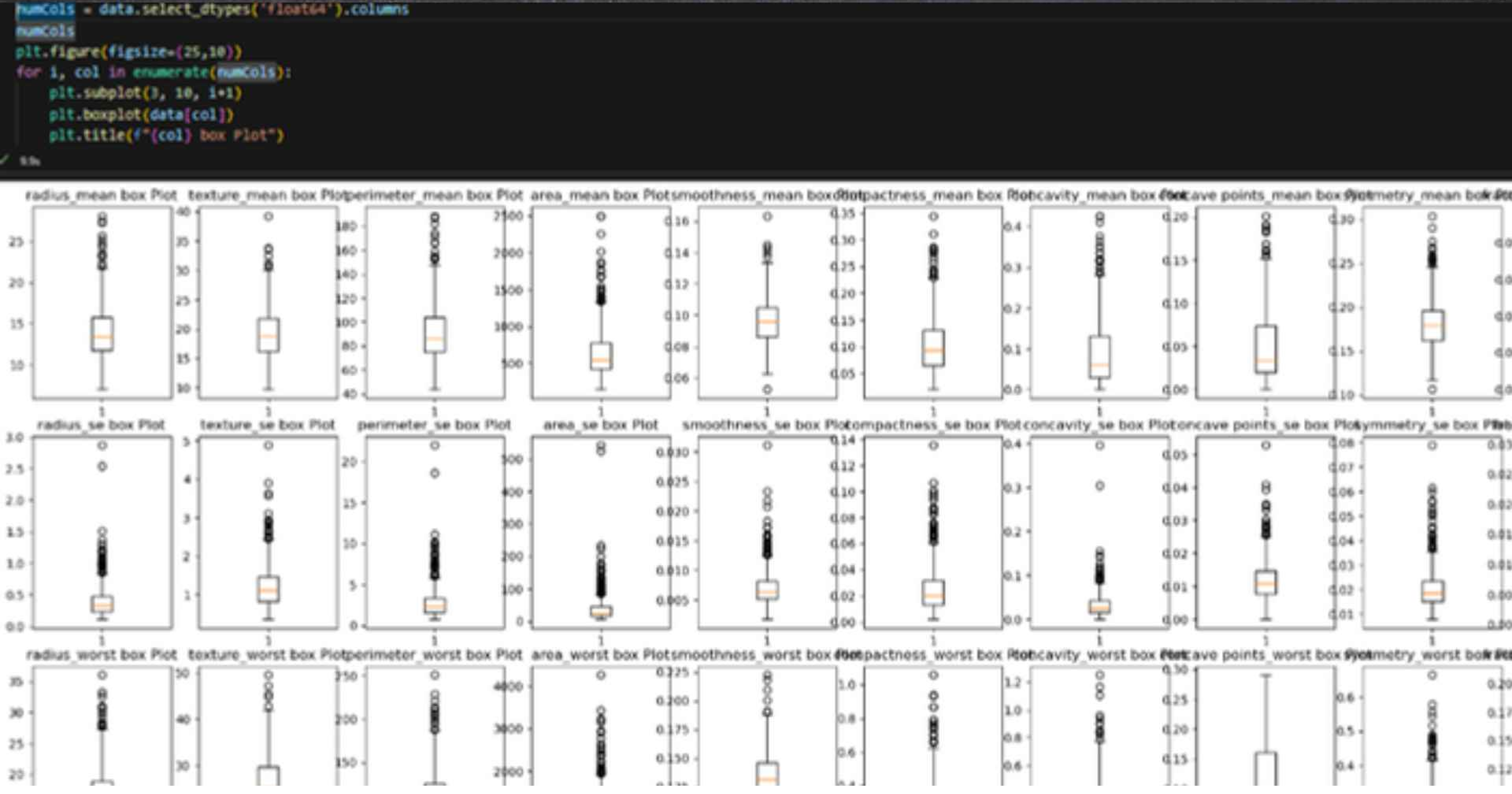
```
diagnosis
B    357
M    212
```

3-LABEL ENCODE TO BE 0 ,1



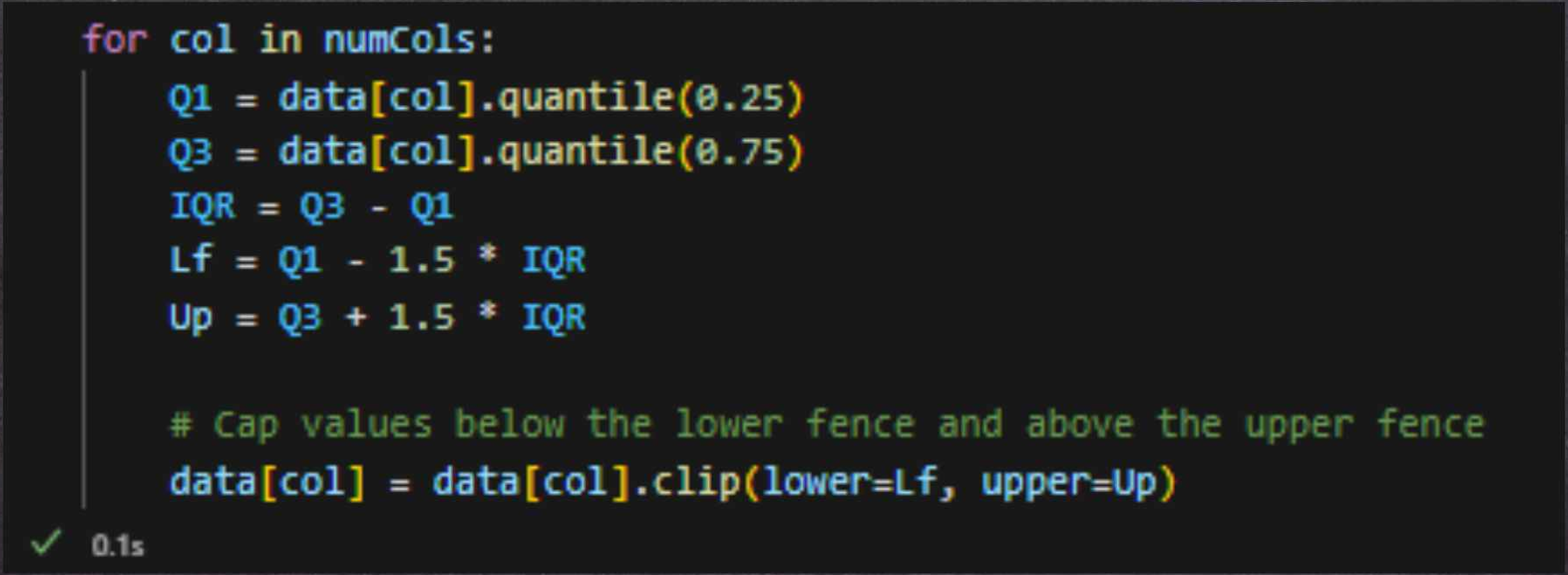
```python
labelencoder = LabelEncoder()
data['diagnosis'] = labelencoder.fit_transform(data['diagnosis'])
```

4-BOX PLOT



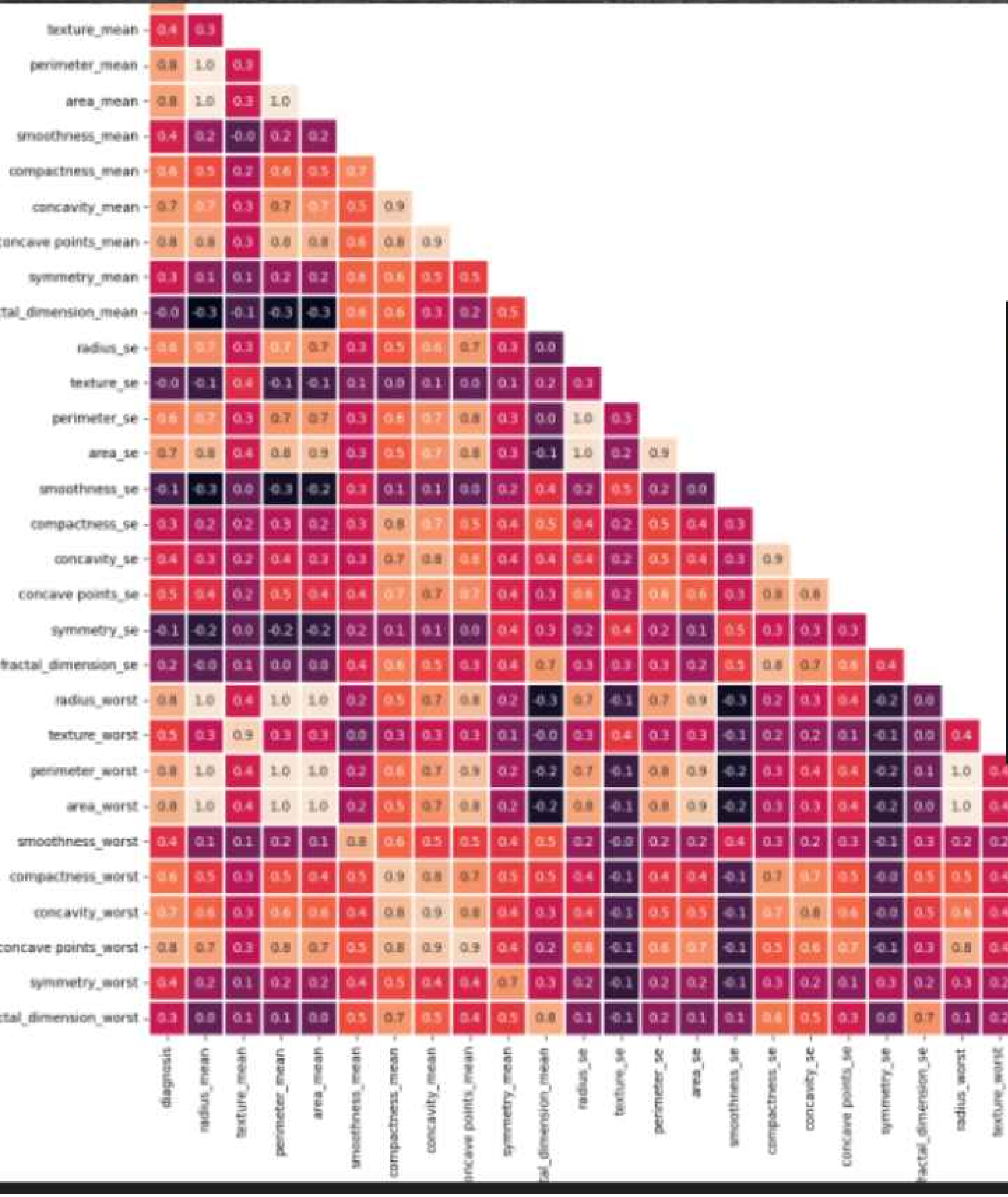
```
numCols = data.select_dtypes('float64').columns
numCols
plt.figure(figsize=(25,10))
for i, col in enumerate(numCols):
    plt.subplot(3, 10, i+1)
    plt.boxplot(data[col])
    plt.title(f"{col} box Plot")
```

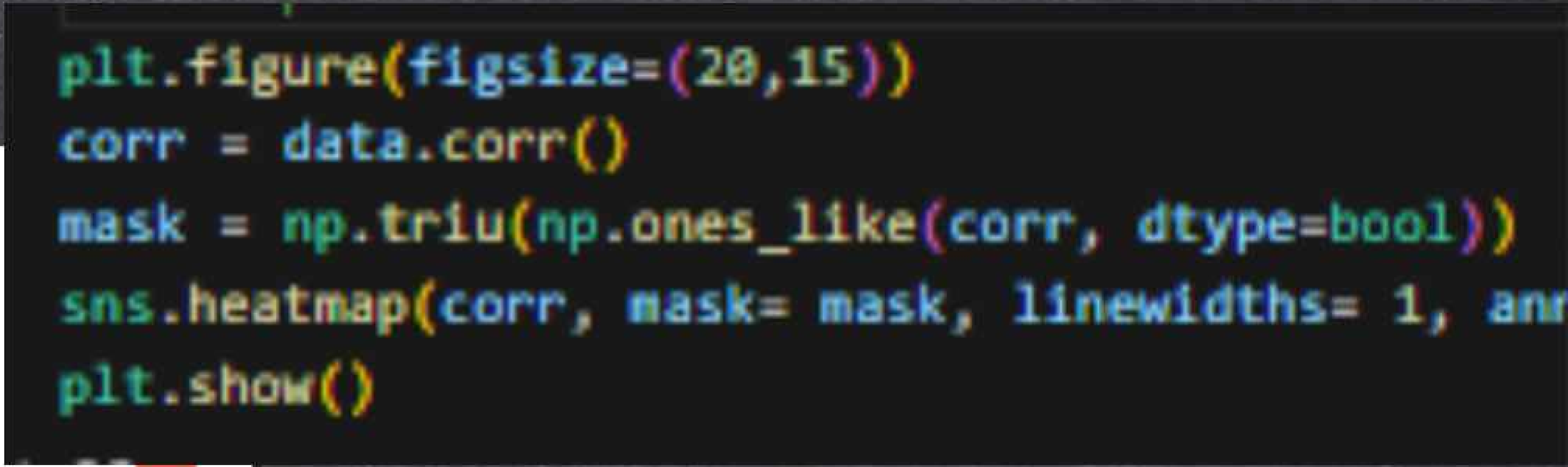
5-SOLVEING THE PROBLEM OF OUT LAYERS



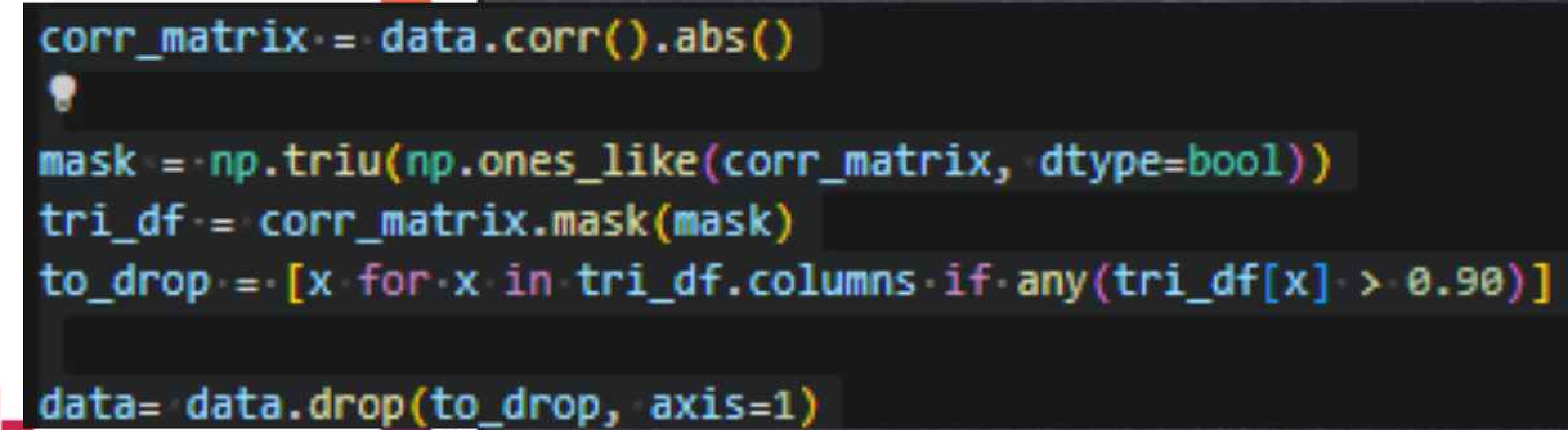
```
for col in numCols:
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    Lf = Q1 - 1.5 * IQR
    Up = Q3 + 1.5 * IQR

    # Cap values below the lower fence and above the upper fence
    data[col] = data[col].clip(lower=Lf, upper=Up)
```

6-DELETING SOME FEATURES



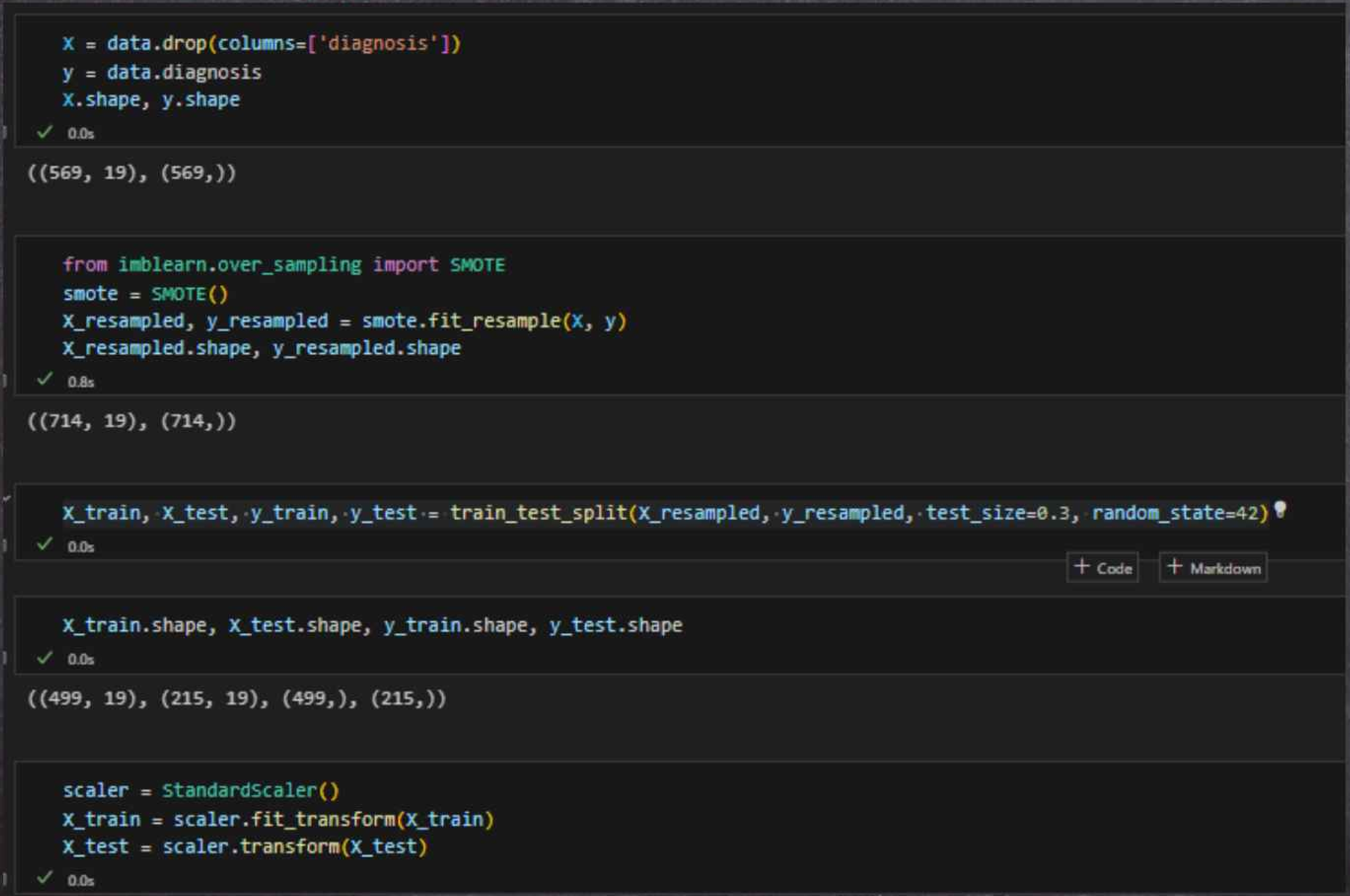


```
plt.figure(figsize=(20,15))
corr = data.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
sns.heatmap(corr, mask= mask, linewidths= 1, ann
plt.show()
```



```
corr_matrix = data.corr().abs()

mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
tri_df = corr_matrix.mask(mask)
to_drop = [x for x in tri_df.columns if any(tri_df[x] > 0.90)]

data= data.drop(to_drop, axis=1)
```

7- FINAL TOUCHES BEFORE MODELS



```python
X = data.drop(columns=['diagnosis'])
y = data.diagnosis
X.shape, y.shape
```

```
((569, 19), (569,))
```

```python
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
X_resampled.shape, y_resampled.shape
```

```
((714, 19), (714,))
```

```python
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42)
```

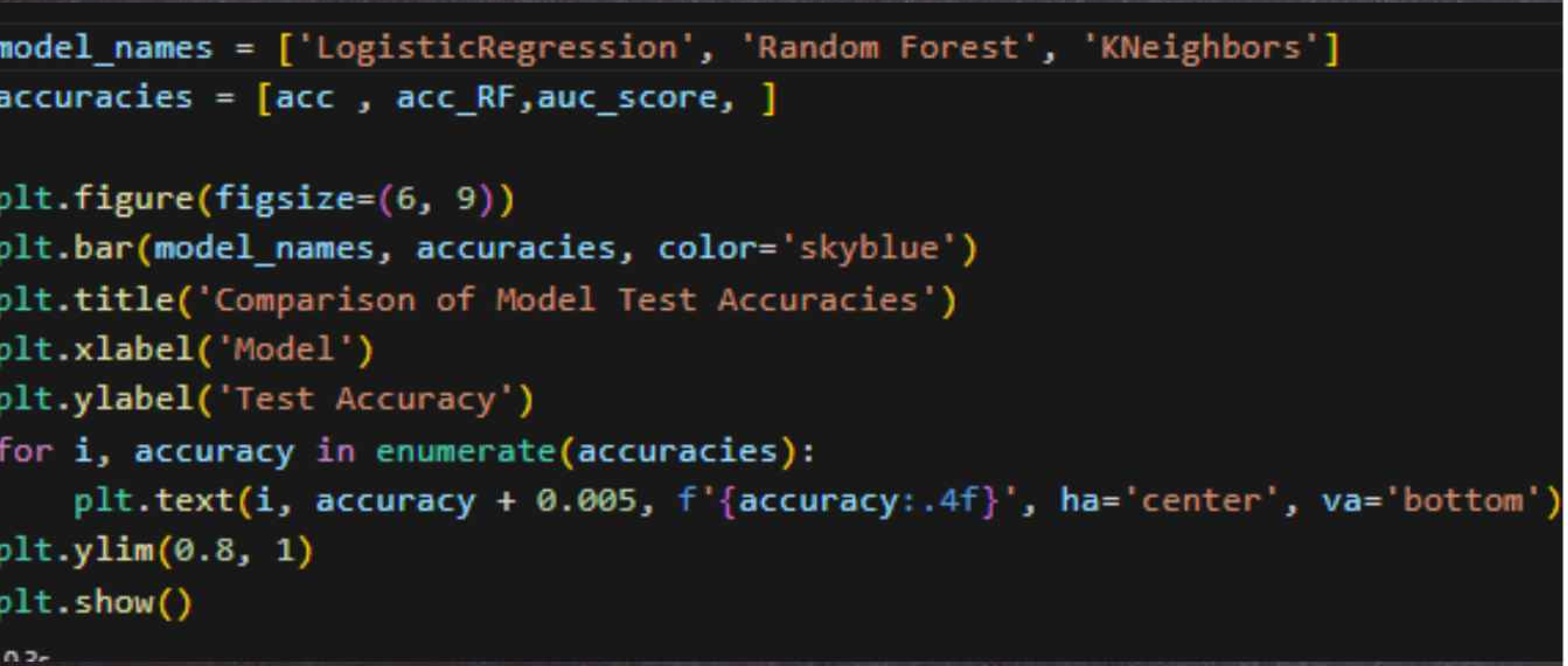
```python
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((499, 19), (215, 19), (499,), (215,))
```

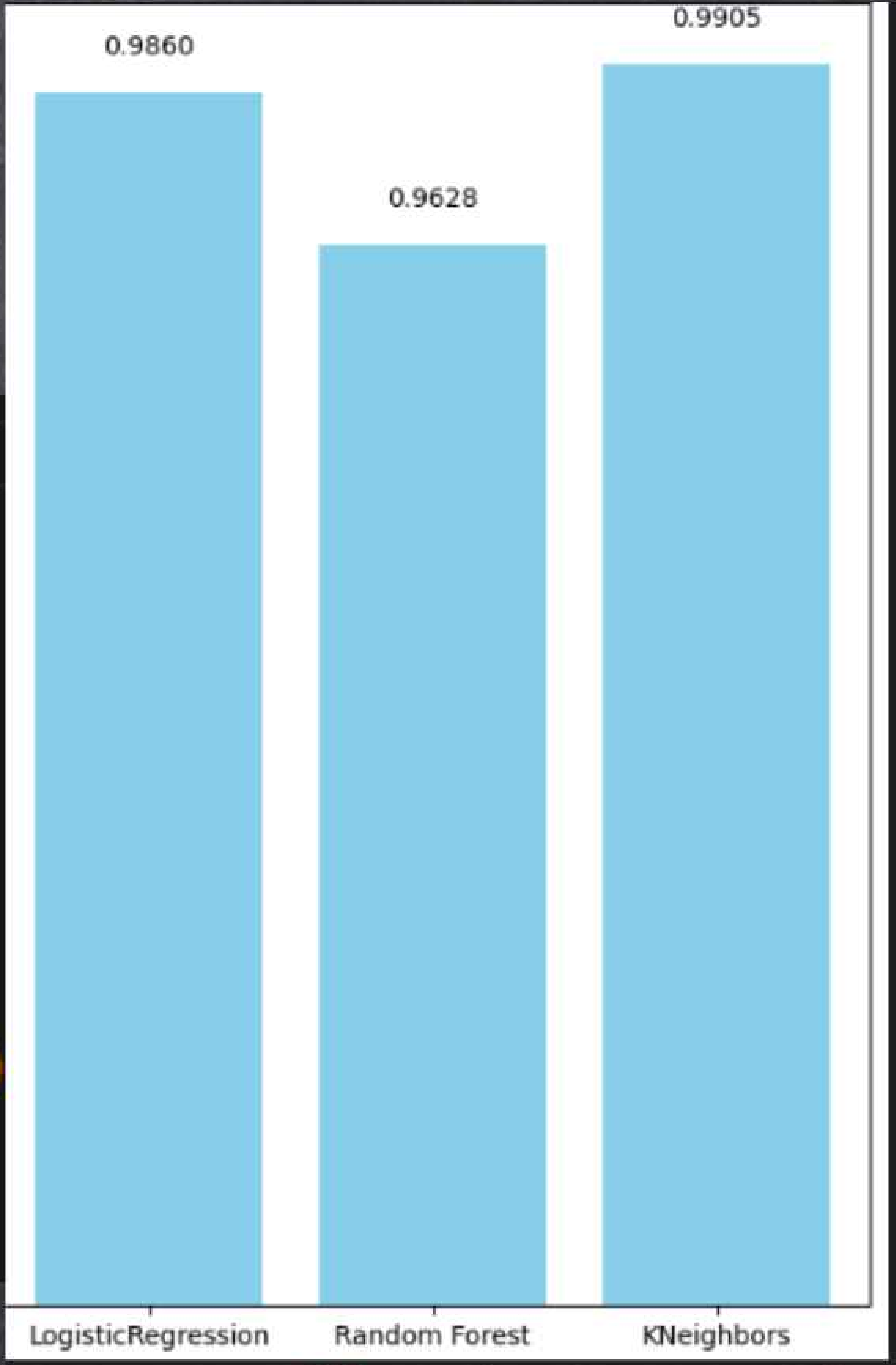
```python
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

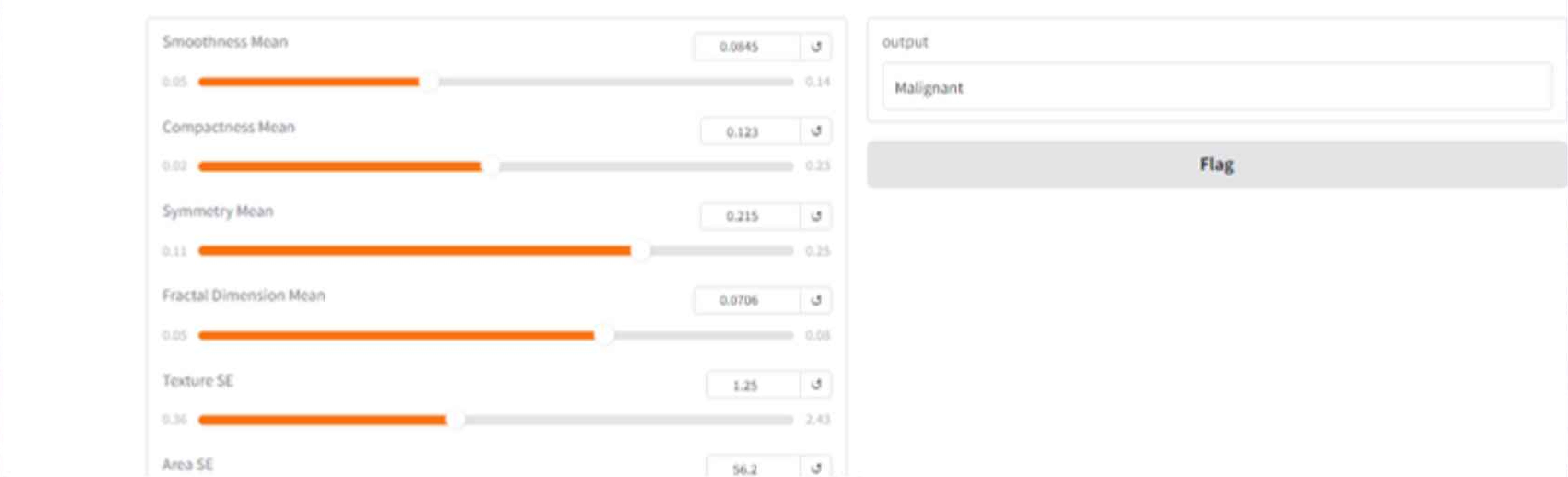
NOTE :
SMOTE (SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE)

MODELS ANALYSIS



```
model_names = ['LogisticRegression', 'Random Forest', 'KNeighbors']
accuracies = [acc , acc_RF,auc_score, ]

plt.figure(figsize=(6, 9))
plt.bar(model_names, accuracies, color='skyblue')
plt.title('Comparison of Model Test Accuracies')
plt.xlabel('Model')
plt.ylabel('Test Accuracy')
for i, accuracy in enumerate(accuracies):
    plt.text(i, accuracy + 0.005, f'{accuracy:.4f}', ha='center', va='bottom')
plt.ylim(0.8, 1)
plt.show()
```





Smoothness Mean 0.0845

0.05 0.14

Compactness Mean 0.123

0.02 0.23

Symmetry Mean 0.215

0.11 0.25

Fractal Dimension Mean 0.0706

0.05 0.08

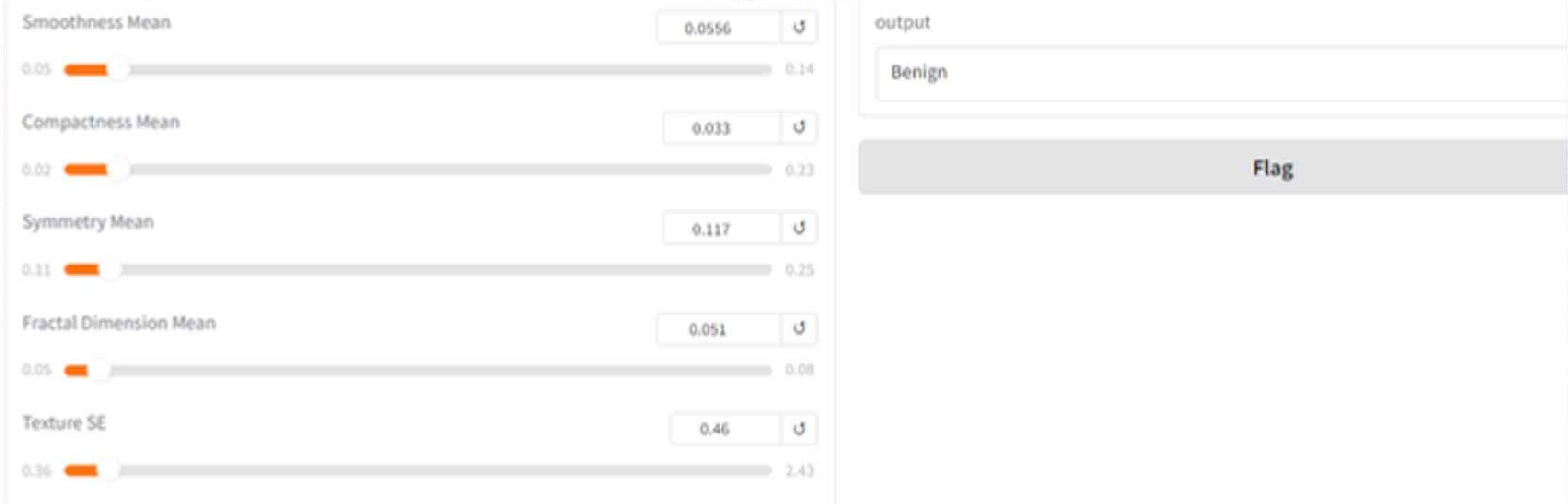
Texture SE 1.25

0.36 2.43

Area SE 56.2

output

Malignant



Smoothness Mean 0.0556

0.05 0.14

Compactness Mean 0.033

0.02 0.23

Symmetry Mean 0.117

0.11 0.25

Fractal Dimension Mean 0.051

0.05 0.08

Texture SE 0.46

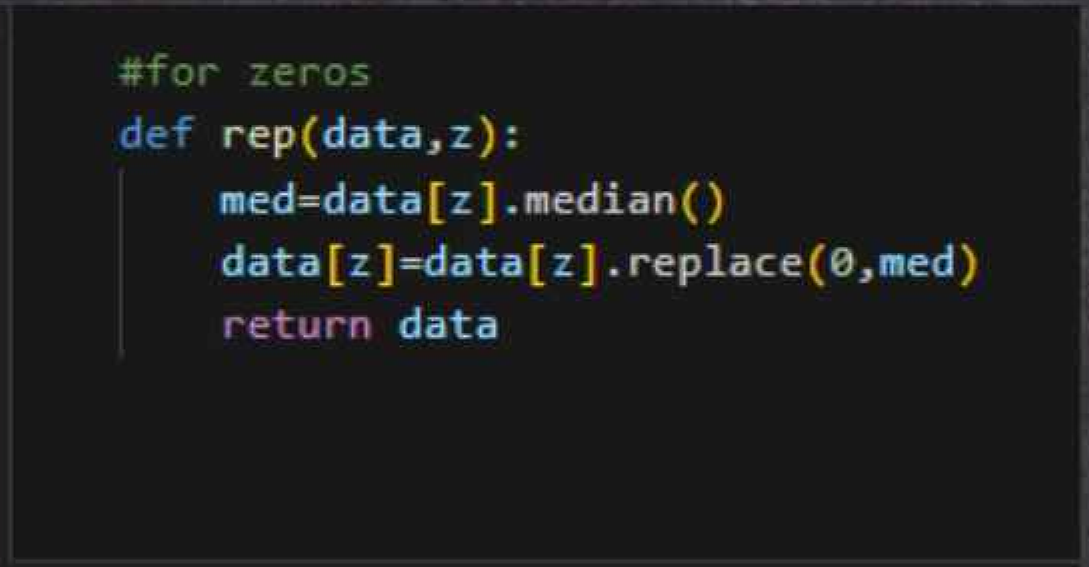
0.36 2.43

output

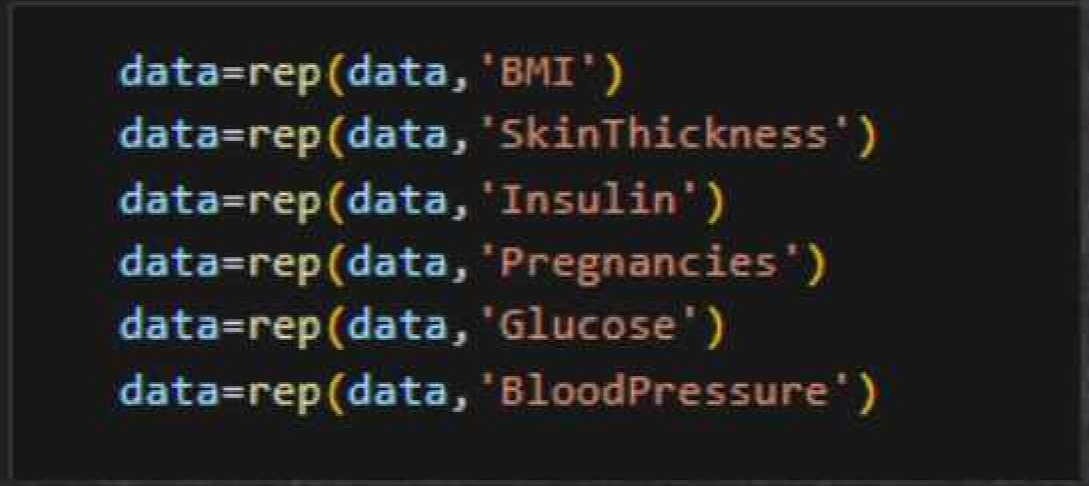
Benign

2-DIABETES

PROCESS ZERO VALUES

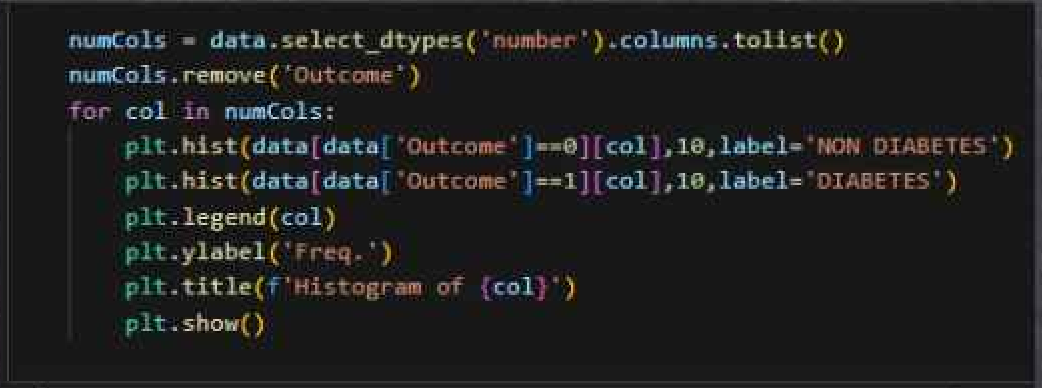


```
#for zeros
def rep(data,z):
    med=data[z].median()
    data[z]=data[z].replace(0,med)
    return data
```

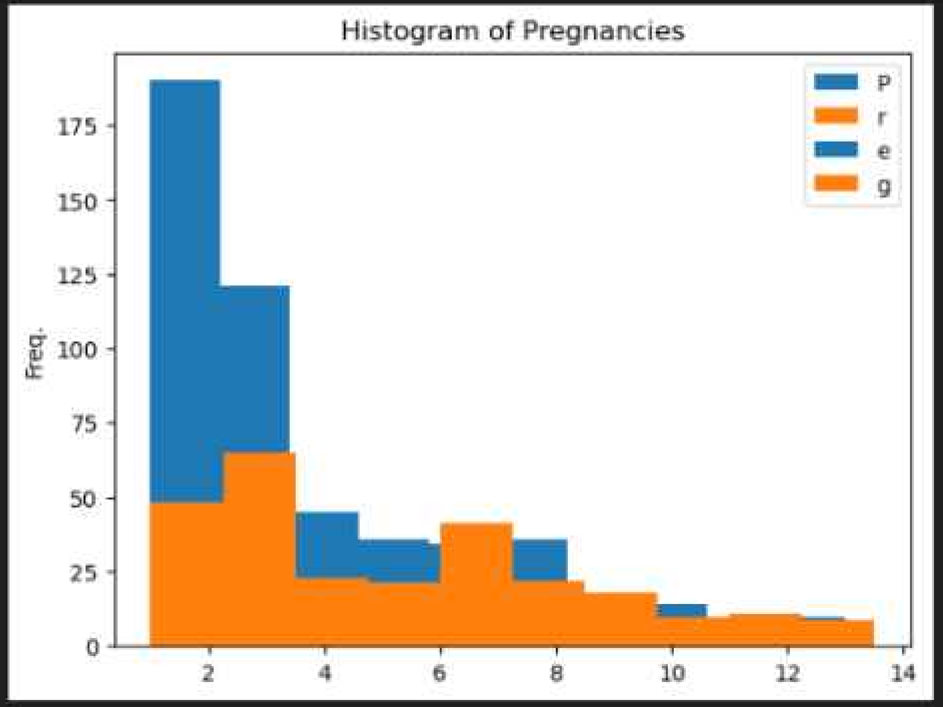


```
data=rep(data,'BMI')
data=rep(data,'SkinThickness')
data=rep(data,'Insulin')
data=rep(data,'Pregnancies')
data=rep(data,'Glucose')
data=rep(data,'BloodPressure')
```

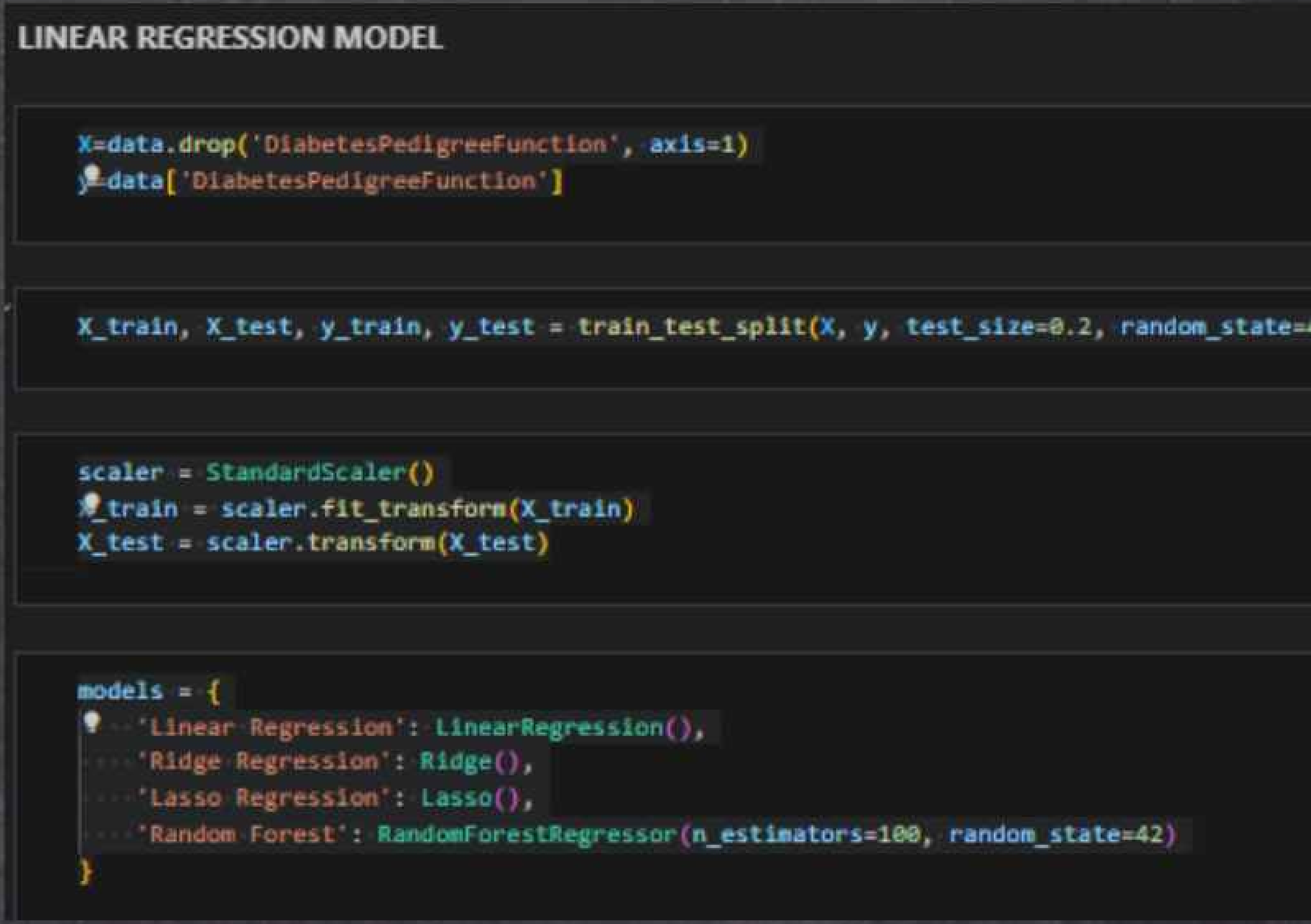
VISUALIZE



```
numCols = data.select_dtypes('number').columns.tolist()
numCols.remove('Outcome')
for col in numCols:
    plt.hist(data[data['Outcome']==0][col],10,label='NON DIABETES')
    plt.hist(data[data['Outcome']==1][col],10,label='DIABETES')
    plt.legend(col)
    plt.ylabel('Freq.')
    plt.title(f'Histogram of {col}')
    plt.show()
```



LINEAR REGRESSION



LINEAR REGRESSION MODEL

```
X=data.drop('DiabetesPedigreeFunction', axis=1)
y=data['DiabetesPedigreeFunction']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
```

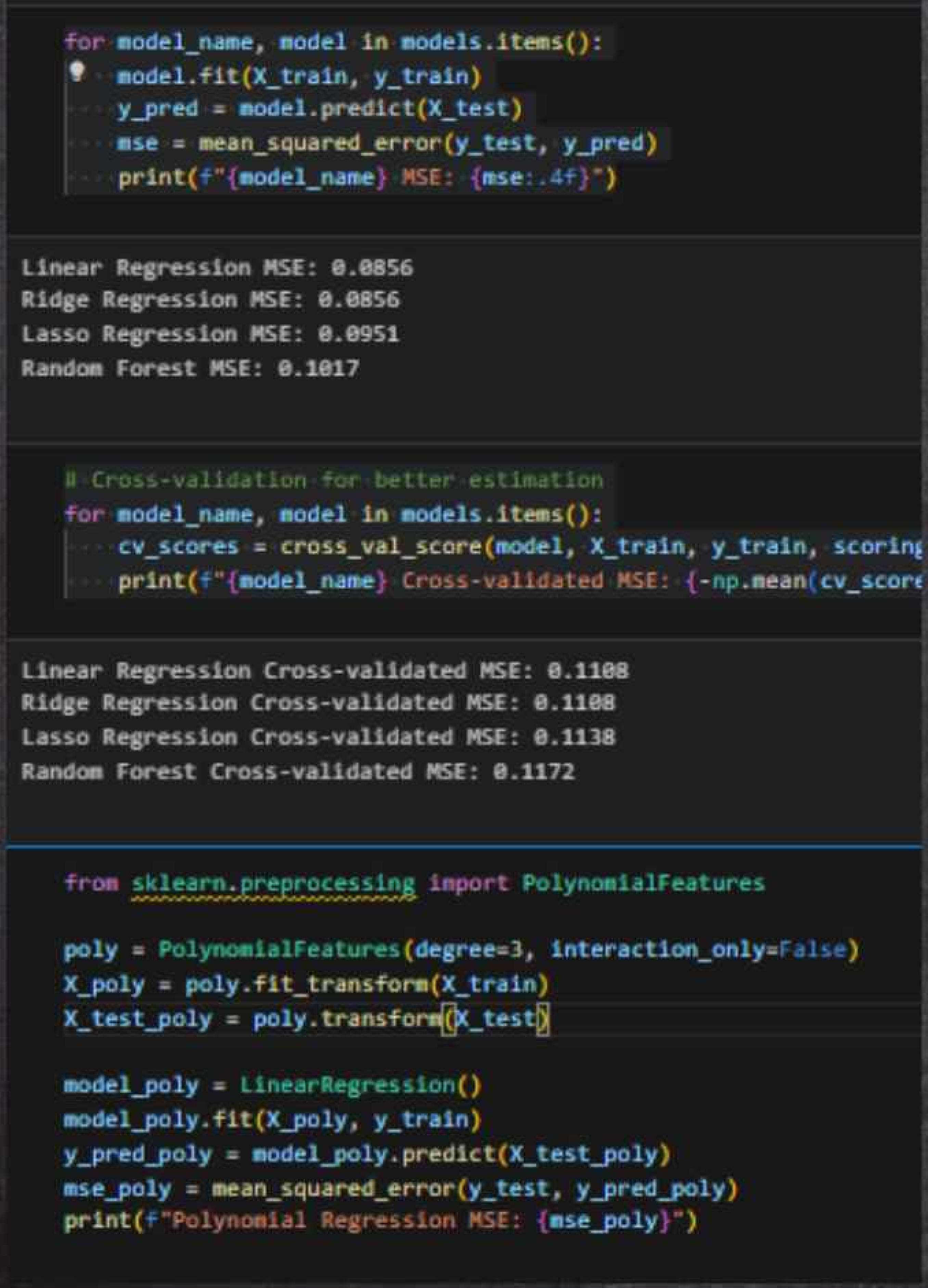
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42)
}
```

SHAHD SALAH

DEPI_FINAL PROJECT

(Fit models and evaluate)



```
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(f"{model_name} MSE: {mse:.4f}")
```

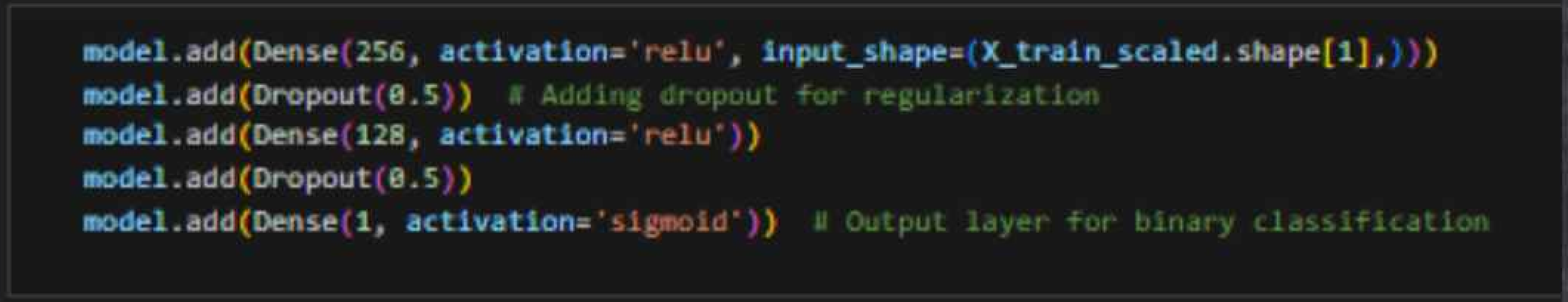
```
Linear Regression MSE: 0.0856
Ridge Regression MSE: 0.0856
Lasso Regression MSE: 0.0951
Random Forest MSE: 0.1017
```

```
# Cross-validation for better estimation
for model_name, model in models.items():
    cv_scores = cross_val_score(model, X_train, y_train, scoring
    print(f"{model_name} Cross-validated MSE: {-np.mean(cv_score
```

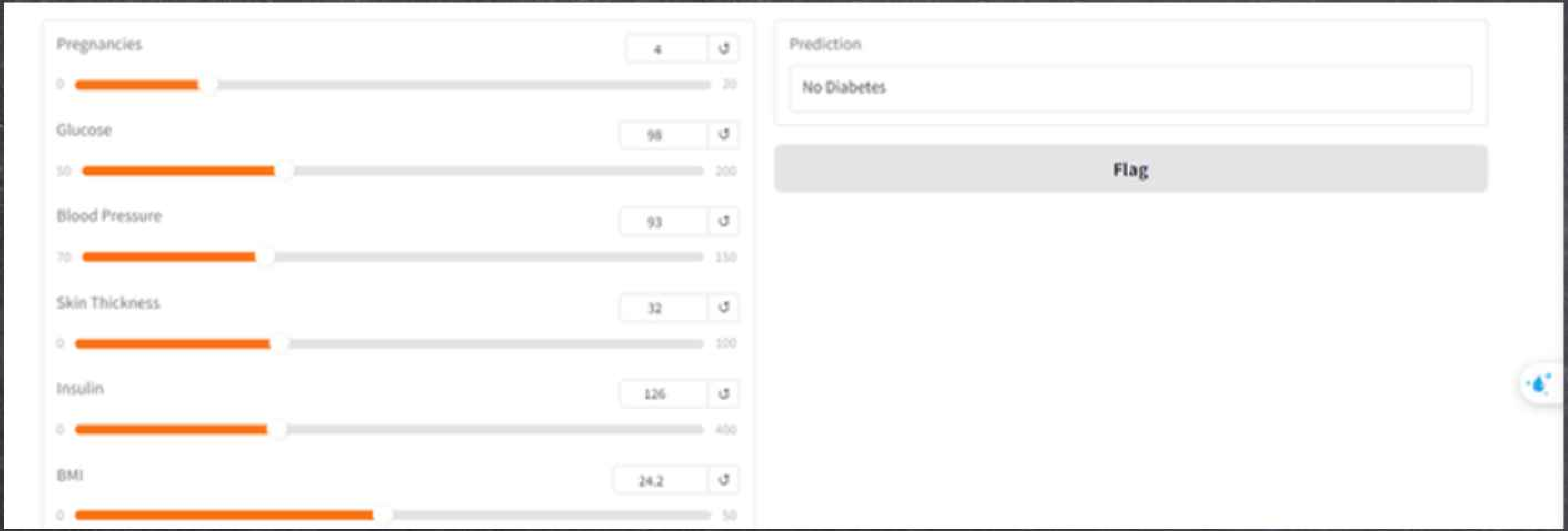
```
Linear Regression Cross-validated MSE: 0.1108
Ridge Regression Cross-validated MSE: 0.1108
Lasso Regression Cross-validated MSE: 0.1138
Random Forest Cross-validated MSE: 0.1172
```

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=3, interaction_only=False)
X_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model_poly = LinearRegression()
model_poly.fit(X_poly, y_train)
y_pred_poly = model_poly.predict(X_test_poly)
mse_poly = mean_squared_error(y_test, y_pred_poly)
print(f"Polynomial Regression MSE: {mse_poly}")
```

LAYERS



```
model.add(Dense(256, activation='relu', input_shape=(X_train_scaled.shape[1],)))
model.add(Dropout(0.5))  # Adding dropout for regularization
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))  # Output layer for binary classification
```



Pregnancies 4

0 20

Glucose 98

50 200

Blood Pressure 93

70 150

Skin Thickness 32

0 100

Insulin 126

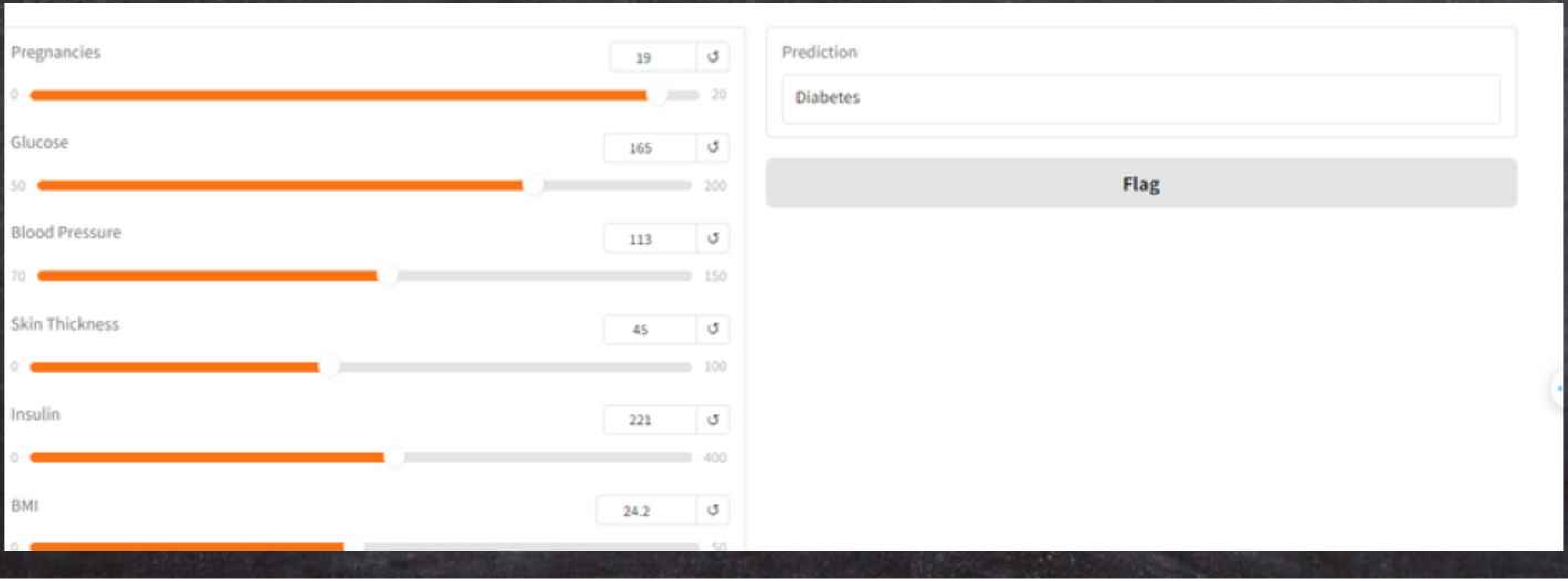
0 400

BMI 24.2

0 50

Prediction

No Diabetes



Pregnancies 19

0 20

Glucose 165

50 200

Blood Pressure 113

70 150

Skin Thickness 45

0 100

Insulin 221

0 400

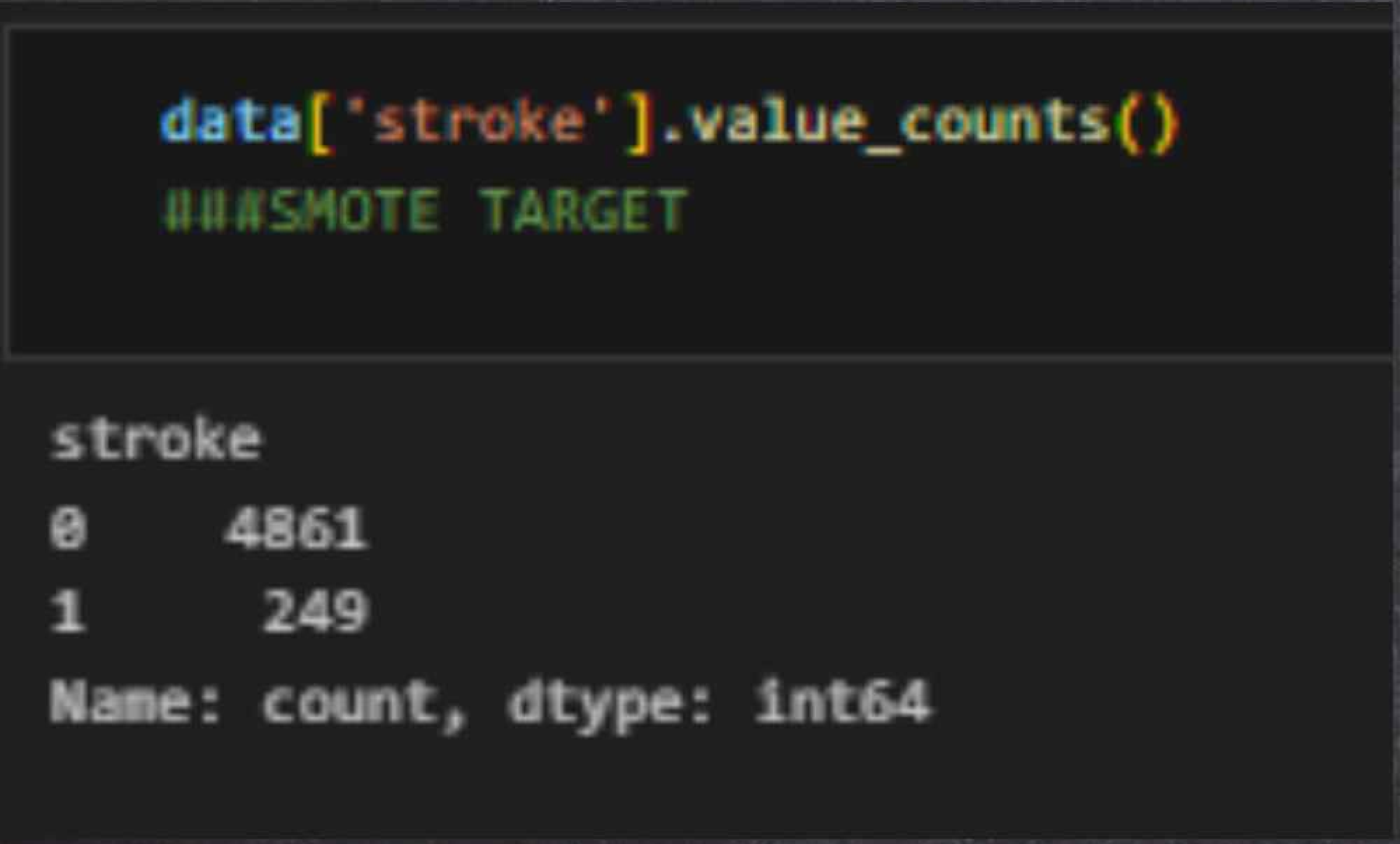
BMI 24.2

0 50

Prediction

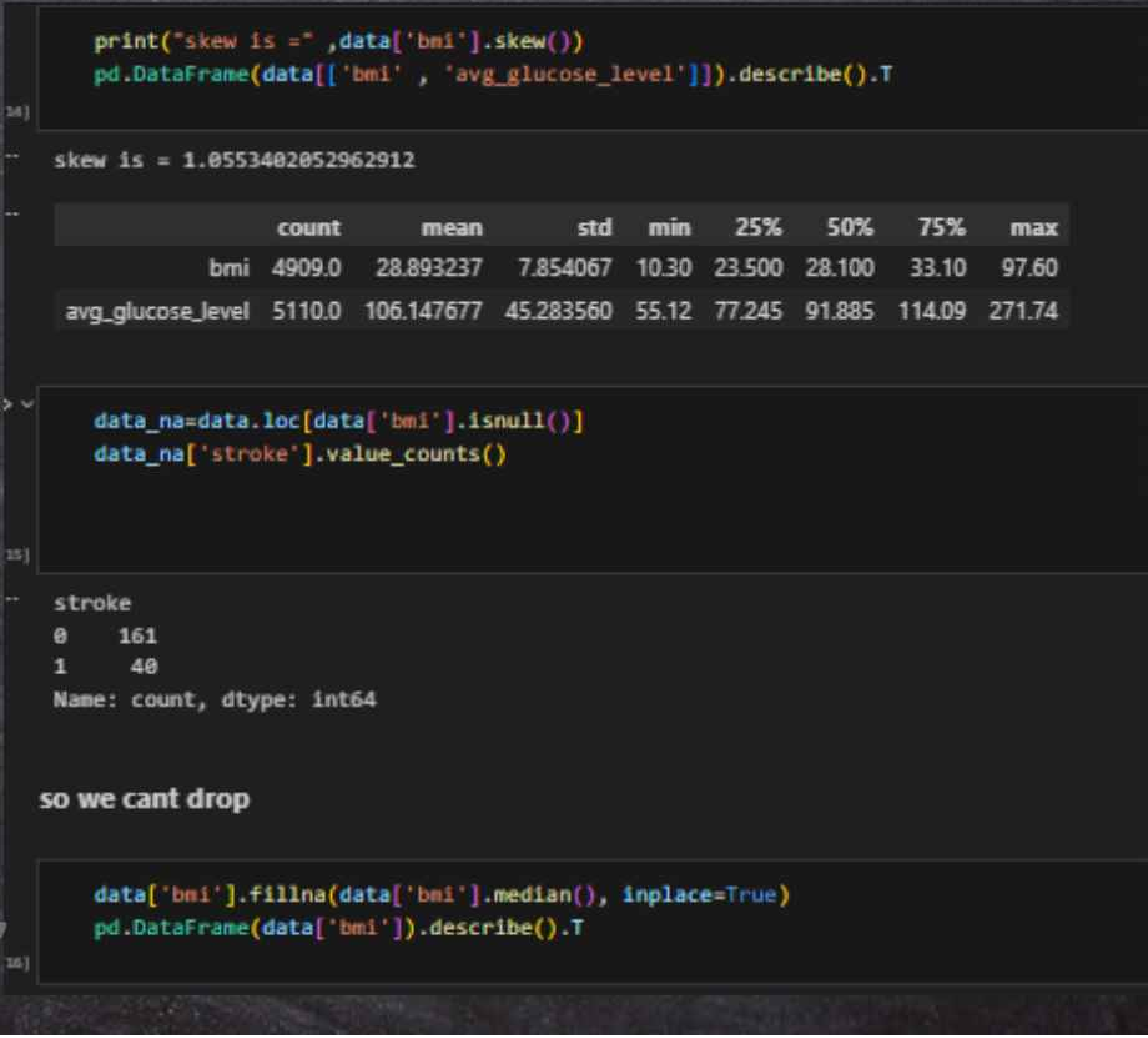
Diabetes

3-STROKE



```
data['stroke'].value_counts()
###SMOTE TARGET
```

```
stroke
0    4861
1     249
Name: count, dtype: int64
```



```
print("skew is =" ,data['bmi'].skew())
pd.DataFrame(data[['bmi' , 'avg_glucose_level']]).describe().T
```

```
skew is = 1.0553402052962912
```

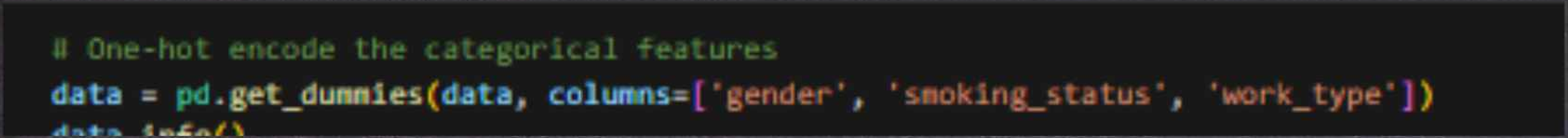
	count	mean	std	min	25%	50%	75%	max
bmi	4909.0	28.893237	7.854067	10.30	23.500	28.100	33.10	97.60
avg_glucose_level	5110.0	106.147677	45.283560	55.12	77.245	91.885	114.09	271.74

```
data_na=data.loc[data['bmi'].isnull()]
data_na['stroke'].value_counts()
```

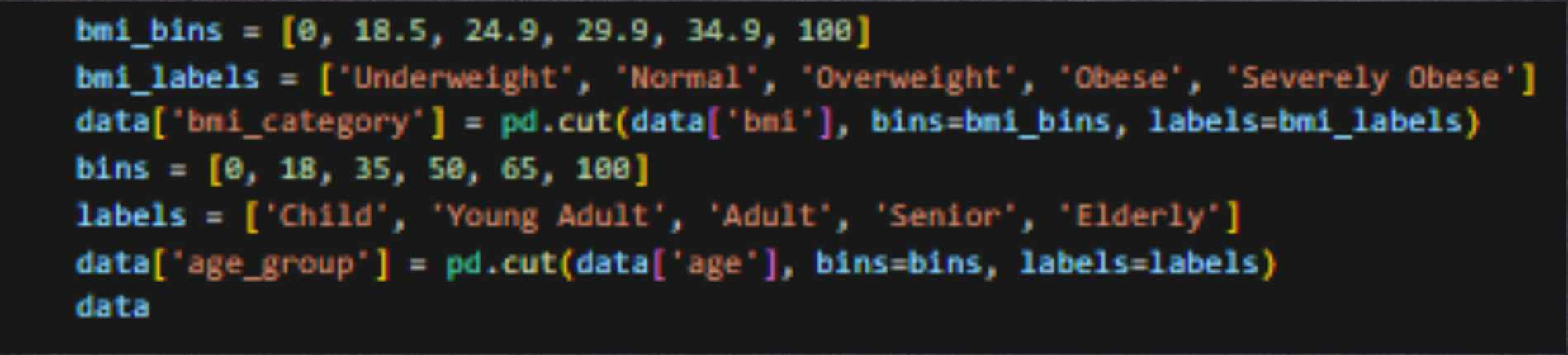
```
stroke
0    161
1     40
Name: count, dtype: int64
```

so we cant drop

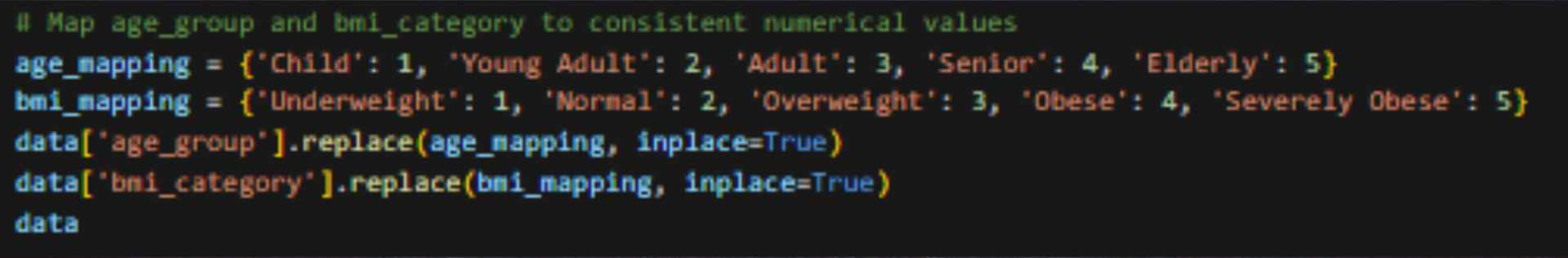
```
data['bmi'].fillna(data['bmi'].median(), inplace=True)
pd.DataFrame(data['bmi']).describe().T
```



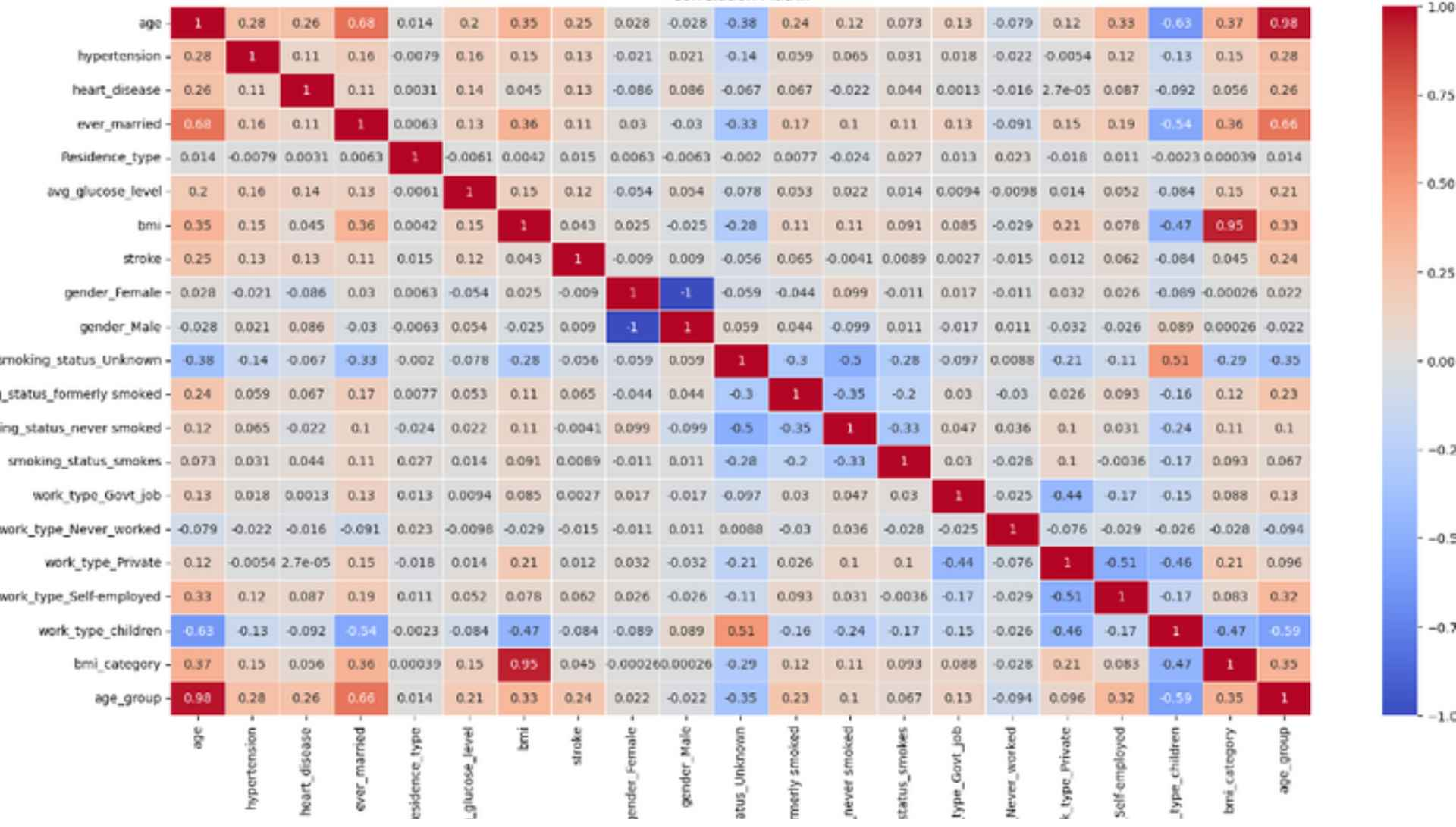
```python
# One-hot encode the categorical features
data = pd.get_dummies(data, columns=['gender', 'smoking_status', 'work_type'])
data.info()
```

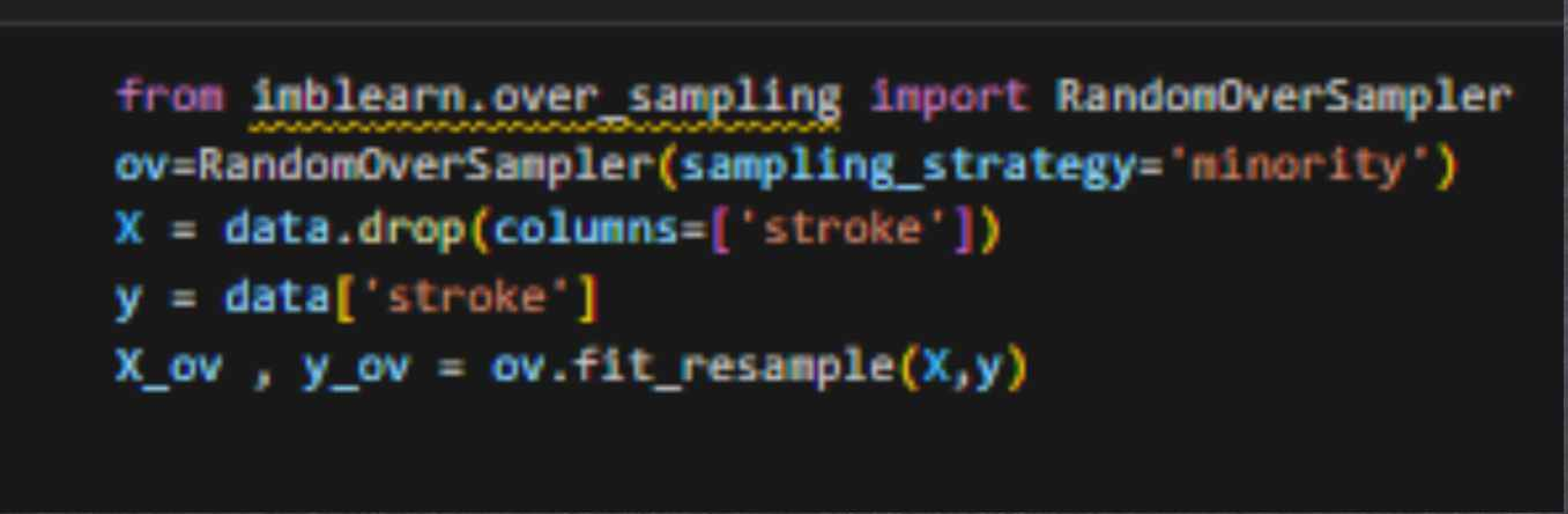


```python
bmi_bins = [0, 18.5, 24.9, 29.9, 34.9, 100]
bmi_labels = ['Underweight', 'Normal', 'Overweight', 'Obese', 'Severely Obese']
data['bmi_category'] = pd.cut(data['bmi'], bins=bmi_bins, labels=bmi_labels)
bins = [0, 18, 35, 50, 65, 100]
labels = ['Child', 'Young Adult', 'Adult', 'Senior', 'Elderly']
data['age_group'] = pd.cut(data['age'], bins=bins, labels=labels)
data
```

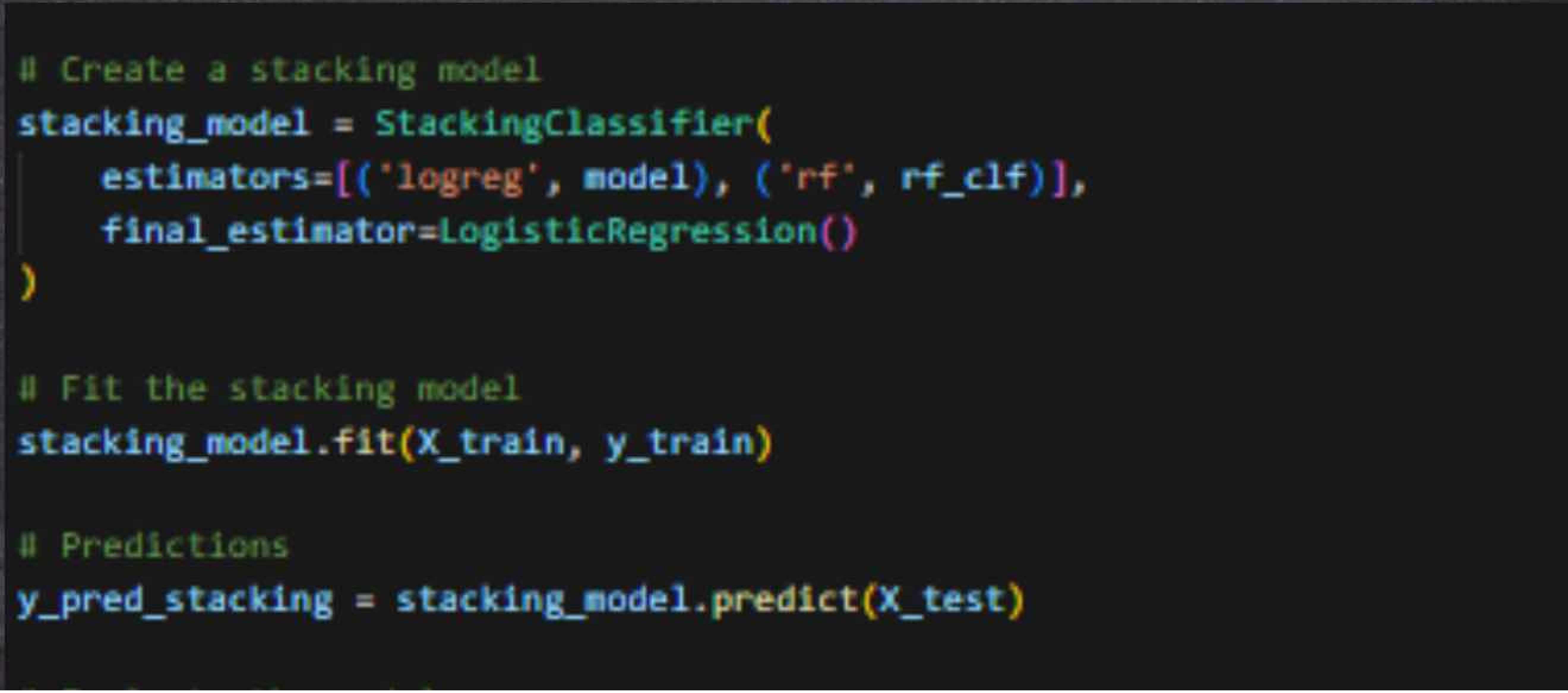


```python
# Map age_group and bmi_category to consistent numerical values
age_mapping = {'Child': 1, 'Young Adult': 2, 'Adult': 3, 'Senior': 4, 'Elderly': 5}
bmi_mapping = {'Underweight': 1, 'Normal': 2, 'Overweight': 3, 'Obese': 4, 'Severely Obese': 5}
data['age_group'].replace(age_mapping, inplace=True)
data['bmi_category'].replace(bmi_mapping, inplace=True)
data
```

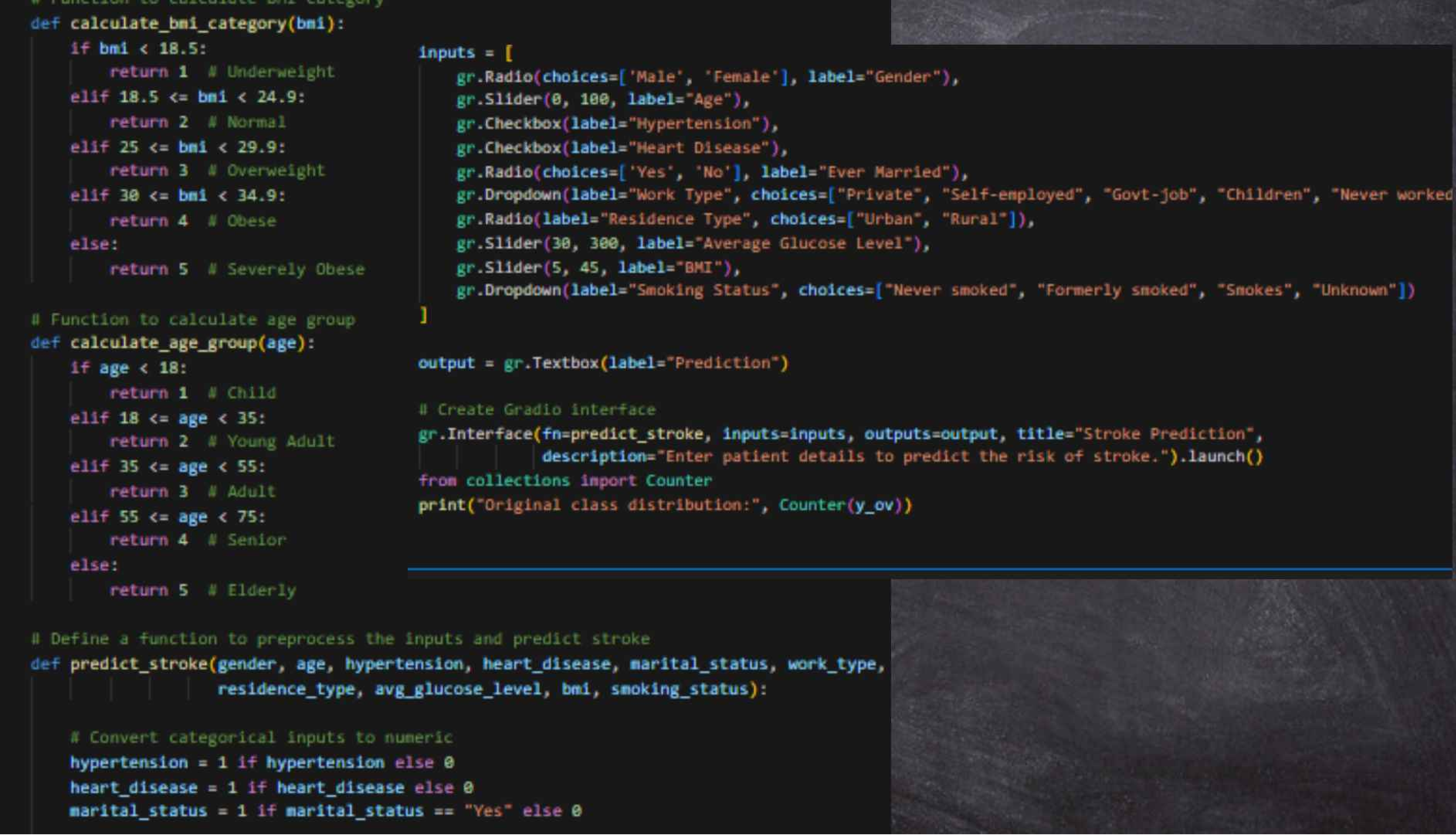

age 1 0.28 0.26 0.68 0.014 0.2 0.35 0.25 0.028 -0.028 -0.38 0.24 0.12 0.073 0.13 -0.079 0.12 0.33 -0.63 0.37 0.98
hypertension 0.28 1 0.11 0.16 -0.0079 0.16 0.15 0.13 -0.021 0.021 -0.14 0.059 0.065 0.031 0.018 -0.022 -0.0054 0.12 -0.13 0.15 0.28
heart_disease 0.26 0.11 1 0.11 0.0031 0.14 0.045 0.13 -0.086 0.086 -0.067 0.067 -0.022 0.044 0.0013 -0.016 2.7e-05 0.087 -0.092 0.056 0.26
ever_married 0.68 0.16 0.11 1 0.0063 0.13 0.36 0.11 0.03 -0.03 -0.33 0.17 0.1 0.11 0.13 -0.091 0.15 0.19 -0.54 0.36 0.66
Residence_type 0.014 -0.0079 0.0031 0.0063 1 -0.0061 0.0042 0.015 0.0063 -0.0063 -0.002 0.0077 -0.024 0.027 0.013 0.023 -0.018 0.011 -0.0023 0.00039 0.014
avg_glucose_level 0.2 0.16 0.14 0.13 -0.0061 1 0.15 0.12 -0.054 0.054 -0.078 0.053 0.022 0.014 0.0094 -0.0098 0.014 0.052 -0.084 0.15 0.21
bmi 0.35 0.15 0.045 0.36 0.0042 0.15 1 0.043 0.025 -0.025 -0.28 0.11 0.11 0.091 0.085 -0.029 0.21 0.078 -0.47 0.95 0.33
stroke 0.25 0.13 0.13 0.11 0.015 0.12 0.043 1 -0.009 0.009 -0.056 0.065 -0.0041 0.0089 0.0027 -0.015 0.012 0.062 -0.084 0.045 0.24
gender_Female 0.028 -0.021 -0.086 0.03 0.0063 -0.054 0.025 -0.009 1 -1 -0.059 -0.044 0.099 -0.011 0.017 -0.011 0.032 0.026 -0.089 -0.00026 0.022
gender_Male -0.028 0.021 0.086 -0.03 -0.0063 0.054 -0.025 0.009 -1 1 0.059 0.044 -0.099 0.011 -0.017 0.011 -0.032 -0.026 0.089 0.00026 -0.022
moking_status_Unknown -0.38 -0.14 -0.067 -0.33 -0.002 -0.078 -0.28 -0.056 -0.059 0.059 1 -0.3 -0.5 -0.28 -0.097 0.0088 -0.21 -0.11 0.51 -0.29 -0.35
_status_formerly smoked 0.24 0.059 0.067 0.17 0.0077 0.053 0.11 0.065 -0.044 0.044 -0.3 1 -0.35 -0.2 0.03 -0.03 0.026 0.093 -0.16 0.12 0.23
ing_status_never smoked 0.12 0.065 -0.022 0.1 -0.024 0.022 0.11 -0.0041 0.099 -0.099 -0.5 -0.35 1 -0.33 0.047 0.036 0.1 0.031 -0.24 0.11 0.1
smoking_status_smokes 0.073 0.031 0.044 0.11 0.027 0.014 0.091 0.0089 -0.011 0.011 -0.28 -0.2 -0.33 1 0.03 -0.028 0.1 -0.0036 -0.17 0.093 0.067
work_type_Govt_job 0.13 0.018 0.0013 0.13 0.013 0.0094 0.085 0.0027 0.017 -0.017 -0.097 0.03 0.047 0.03 1 -0.025 -0.44 -0.17 -0.15 0.088 0.13
work_type_Never_worked -0.079 -0.022 -0.016 -0.091 0.023 -0.0098 -0.029 -0.015 -0.011 0.011 0.0088 -0.03 0.036 -0.028 -0.025 1 -0.076 -0.029 -0.026 -0.028 -0.094
work_type_Private 0.12 -0.0054 2.7e-05 0.15 -0.018 0.014 0.21 0.012 0.032 -0.032 -0.21 0.026 0.1 0.1 -0.44 -0.076 1 -0.51 -0.46 0.21 0.096
work_type_Self-employed 0.33 0.12 0.087 0.19 0.011 0.052 0.078 0.062 0.026 -0.026 -0.11 0.093 0.031 -0.0036 -0.17 -0.029 -0.51 1 -0.17 0.083 0.32
work_type_children -0.63 -0.13 -0.092 -0.54 -0.0023 -0.084 -0.47 -0.084 -0.089 0.089 0.51 -0.16 -0.24 -0.17 -0.15 -0.026 -0.46 -0.17 1 -0.47 -0.59
bmi_category 0.37 0.15 0.056 0.36 0.00039 0.15 0.95 0.045 -0.00026 0.00026 -0.29 0.12 0.11 0.093 0.088 -0.028 0.21 0.083 -0.47 1 0.35
age_group 0.98 0.28 0.26 0.66 0.014 0.21 0.33 0.24 0.022 -0.022 -0.35 0.23 0.1 0.067 0.13 -0.094 0.096 0.32 -0.59 0.35 1
age
hypertension
heart_disease
ever_married
esidence_type
_glucose_level
bmi
stroke
ender_Female
gender_Male
tatus_Unknown
merly smoked
never smoked
status_smokes
type_Govt_job
Never_worked
k_type_Private
Self-employed
_type_children
bmi_category
age_group
1.00
0.75
0.50
0.25
0.00
-0.2
-0.5
-0.7
-1.0



```python
from imblearn.over_sampling import RandomOverSampler
ov=RandomOverSampler(sampling_strategy='minority')
X = data.drop(columns=['stroke'])
y = data['stroke']
X_ov , y_ov = ov.fit_resample(X,y)
```

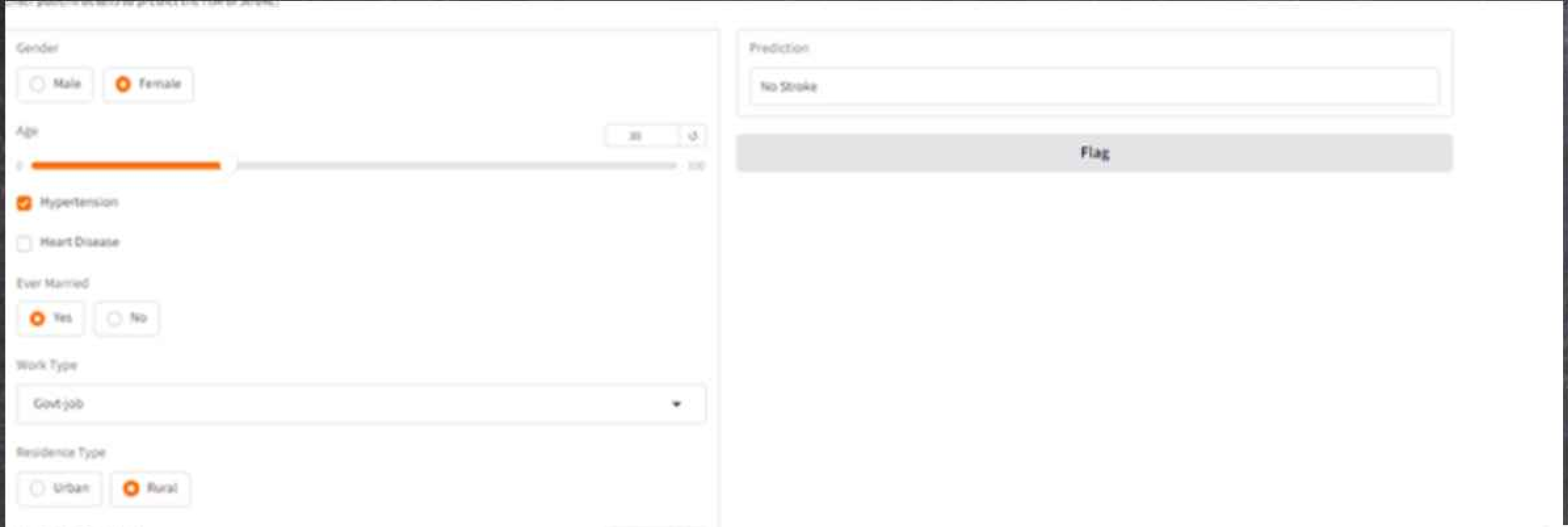


```python
# Create a stacking model
stacking_model = StackingClassifier(
    estimators=[('logreg', model), ('rf', rf_clf)],
    final_estimator=LogisticRegression()
)

# Fit the stacking model
stacking_model.fit(X_train, y_train)

# Predictions
y_pred_stacking = stacking_model.predict(X_test)
```



```
# Function to calculate bmi category
def calculate_bmi_category(bmi):
    if bmi < 18.5:
        return 1  # Underweight
    elif 18.5 <= bmi < 24.9:
        return 2  # Normal
    elif 25 <= bmi < 29.9:
        return 3  # Overweight
    elif 30 <= bmi < 34.9:
        return 4  # Obese
    else:
        return 5  # Severely Obese

# Function to calculate age group
def calculate_age_group(age):
    if age < 18:
        return 1  # Child
    elif 18 <= age < 35:
        return 2  # Young Adult
    elif 35 <= age < 55:
        return 3  # Adult
    elif 55 <= age < 75:
        return 4  # Senior
    else:
        return 5  # Elderly

# Define a function to preprocess the inputs and predict stroke
def predict_stroke(gender, age, hypertension, heart_disease, marital_status, work_type,
                   residence_type, avg_glucose_level, bmi, smoking_status):

    # Convert categorical inputs to numeric
    hypertension = 1 if hypertension else 0
    heart_disease = 1 if heart_disease else 0
    marital_status = 1 if marital_status == "Yes" else 0
```

```
inputs = [
    gr.Radio(choices=['Male', 'Female'], label="Gender"),
    gr.Slider(0, 100, label="Age"),
    gr.Checkbox(label="Hypertension"),
    gr.Checkbox(label="Heart Disease"),
    gr.Radio(choices=['Yes', 'No'], label="Ever Married"),
    gr.Dropdown(label="Work Type", choices=["Private", "Self-employed", "Govt-job", "Children", "Never worked
    gr.Radio(label="Residence Type", choices=["Urban", "Rural"]),
    gr.Slider(30, 300, label="Average Glucose Level"),
    gr.Slider(5, 45, label="BMI"),
    gr.Dropdown(label="Smoking Status", choices=["Never smoked", "Formerly smoked", "Smokes", "Unknown"])
]

output = gr.Textbox(label="Prediction")

# Create Gradio interface
gr.Interface(fn=predict_stroke, inputs=inputs, outputs=output, title="Stroke Prediction",
             description="Enter patient details to predict the risk of stroke.").launch()
from collections import Counter
print("Original class distribution:", Counter(y_ov))
```



Gender

○ Male ● Female

Age

30

☑ Hypertension

□ Heart Disease

Ever Married

● Yes ○ No

Work Type

Govt job

Residence Type

○ Urban ● Rural

Average Glucose Level

104

BMI

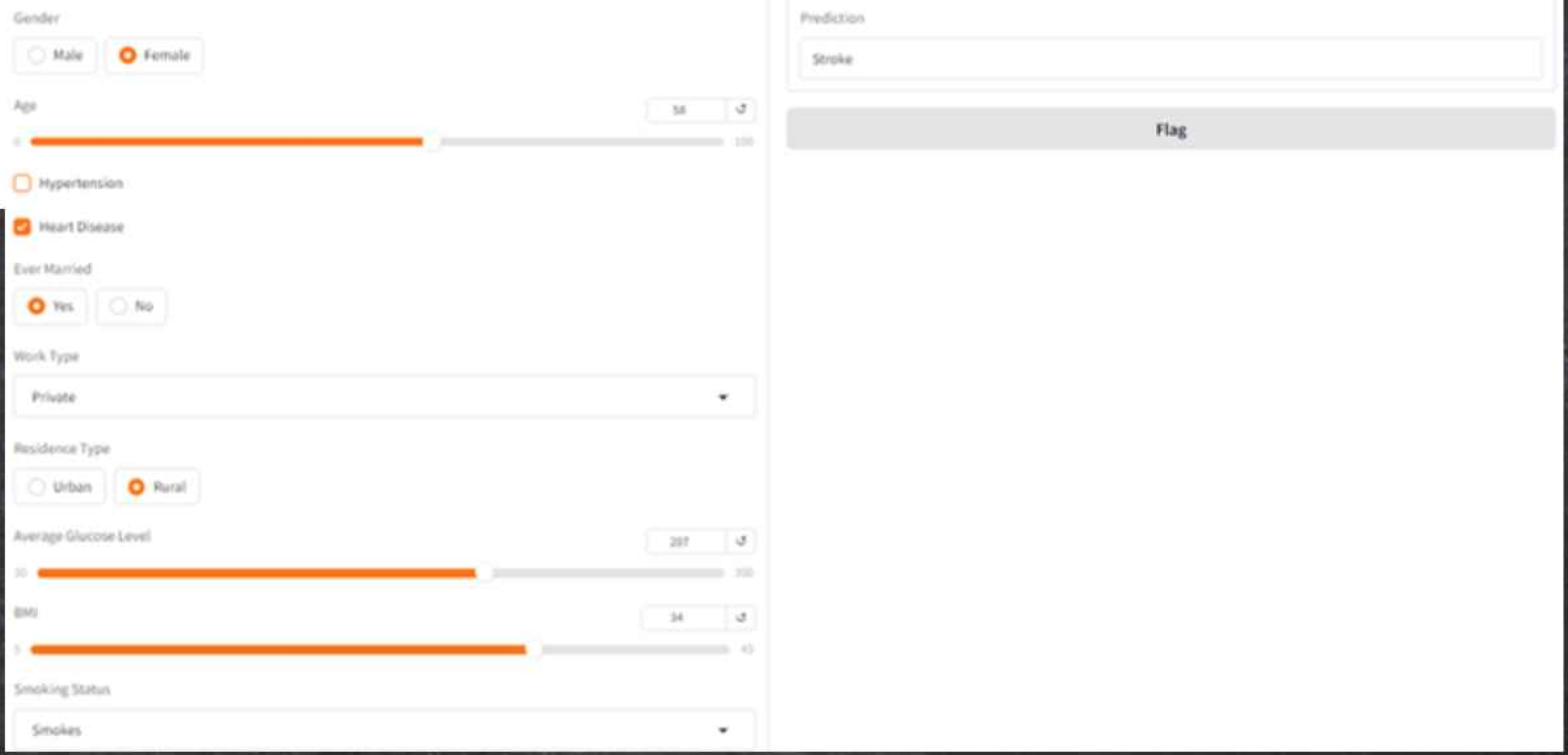
23.5

Smoking Status

Formerly smoked

Prediction

No Stroke



Gender

○ Male ● Female

Age

58

□ Hypertension

☑ Heart Disease

Ever Married

● Yes ○ No

Work Type

Private

Residence Type

○ Urban ● Rural

Average Glucose Level

207

BMI

34

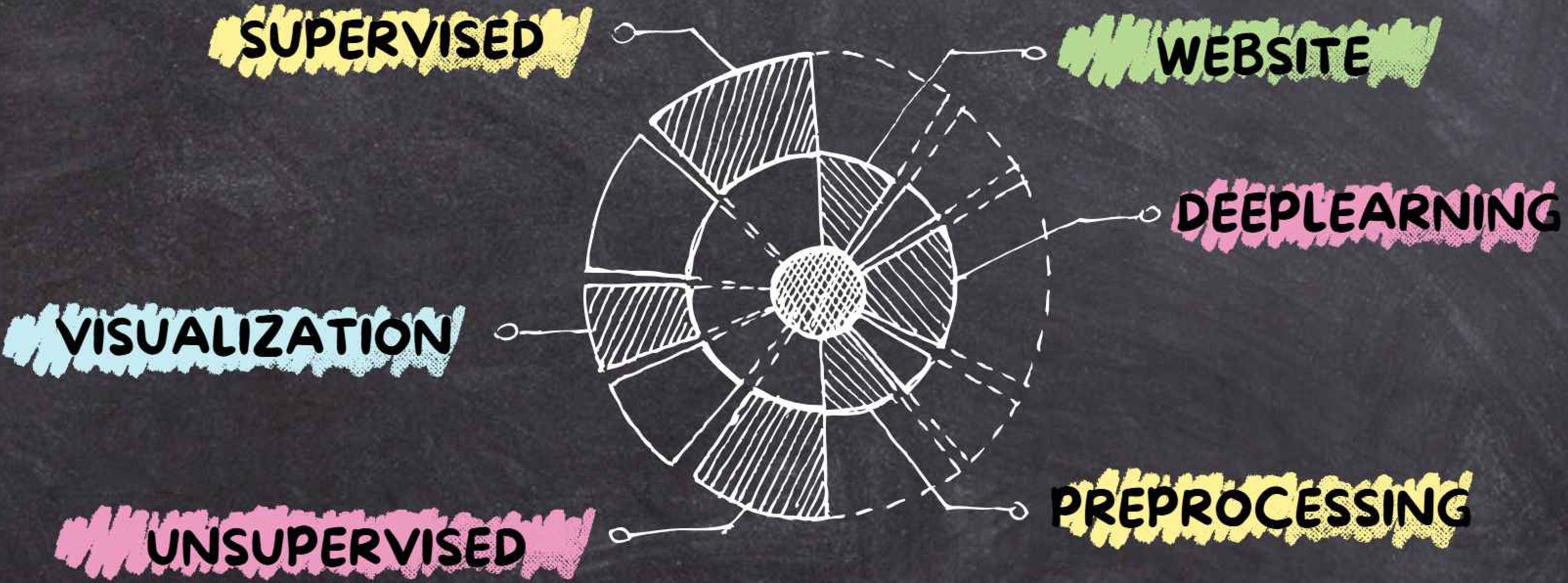
Smoking Status

Smokes

Prediction

Stroke

WHAT IS IN THE CODE



SUPERVISED

VISUALIZATION

UNSUPERVISED

WEBSITE

DEEPLEARNING

PREPROCESSING

CONCLUSION

SUMMARIZE THE PRESENTATION

these projects underscore the transformative impact of machine learning in the healthcare domain. By effectively utilizing data-driven approaches, we can develop predictive models that not only improve diagnostic accuracy but also facilitate timely interventions, leading to enhanced patient care. These models serve as valuable tools for healthcare professionals, allowing them to make informed decisions and prioritize patient health more effectively. As we continue to refine these models and explore new data sources, the potential for advancements in disease prediction and prevention remains significant.