

AIML in Healthcare Lab Experiment No. 7

Aim: Design and implement explainable AI technique using a Healthcare Data.

Theory:

In this experiment, a predictive model was developed to assess stroke risk in patients based on their medical history and health metrics. Stroke is a critical medical condition, and early prediction can help in timely medical interventions. The aim of the project is not only to achieve high prediction accuracy but also to make the model's decision-making process transparent using Explainable AI (XAI) techniques. The Random Forest algorithm was used as the predictive model, and SHAP (SHapley Additive exPlanations) was applied for model interpretability.

Random Forest:

Random Forest is a popular and powerful ensemble learning method used for classification and regression tasks. It is based on the idea of aggregating multiple decision trees to improve predictive performance and reduce overfitting.

Random Forest works by building multiple decision trees during training and combining their outputs to make a final prediction. The key features of Random Forest are:

1. **Bootstrapping (Bagging):** The algorithm creates several subsets of the dataset by sampling with replacement (bootstrapping). Each decision tree in the forest is trained on a different subset.
2. **Random Feature Selection:** At each split in the tree, only a random subset of features is considered, which helps to decorrelate the trees, leading to more diverse trees and reducing overfitting.
3. **Aggregation:** In the case of classification, Random Forest uses majority voting to combine the predictions of individual trees. For regression, it takes the average of the predictions.

Advantages:

1. **High Accuracy:** Random Forest often provides better accuracy than individual decision trees due to its ensemble approach.
2. **Robustness to Overfitting:** By averaging the results of multiple trees, Random Forest mitigates the risk of overfitting, especially in large datasets.
3. **Handles Missing Data:** It can handle missing values effectively by computing approximate answers from trees trained on different subsets of the data.
4. **Feature Importance:** It provides an estimate of feature importance, making it useful for feature selection and interpretability.

Limitations:

1. **Interpretability:** Although it reduces overfitting compared to single decision trees, the ensemble nature makes the model more difficult to interpret.

2. **Computationally Intensive:** Training a large number of decision trees can be resource-intensive in terms of memory and computation.

Evaluation Metrics:

1. **Accuracy:** Accuracy measures the proportion of correct predictions (both true positives and true negatives) out of all predictions made.
2. **Precision:** Precision measures the proportion of true positive predictions among all instances predicted as positive.
3. **Recall:** Recall measures the proportion of actual positives that are correctly identified.
4. **F1-score:** The F1-score is the harmonic mean of precision and recall, providing a balance between them. It is especially useful when dealing with imbalanced classes.

SHAP (SHapley Additive exPlanations):

SHAP is a unified approach to explain the output of machine learning models. It is based on cooperative game theory, specifically the Shapley value, which assigns each feature an importance value for a particular prediction.

Key concepts:

1. **Shapley Values:** Derived from cooperative game theory, Shapley values represent the contribution of each player (in this case, each feature) to the overall prediction. The sum of the SHAP values across all features gives the difference between the model's prediction and the average prediction.
2. **Global Interpretability:** SHAP can show which features are most important across all predictions in the dataset, helping to explain the model's overall behavior.
3. **Local Interpretability:** SHAP can also explain individual predictions by showing how each feature contributes to pushing the model's prediction higher or lower compared to the average output.

Visualization Tools:

SHAP provides several visualization tools that offer both global and local interpretability:

1. **Summary Plot:**

Displays feature importance and how features affect the model output across the entire dataset. The SHAP summary plot shows the overall importance of each feature in the model across all predictions. Features such as Age, Average Glucose Level, and Hypertension were shown to have a significant impact on stroke prediction. The plot highlights the contribution of each feature and how it influences the model's decision.

2. **Force Plot:**

Provides a visual explanation for individual predictions, showing how each feature contributes to pushing the prediction towards a higher or lower value. The force plot provides a local explanation for individual predictions. It visualizes how the features

push the model's prediction towards either stroke (positive) or no stroke (negative). For example, in one prediction, older age and high glucose level strongly pushed the prediction towards a higher risk of stroke.

3. **Dependence Plot:**

Shows the relationship between a single feature and the target while accounting for interactions with other features. A dependence plot was used to show how one feature (e.g., Age) affects the model's predictions, while taking into account interactions with other features (e.g., Hypertension). This visualizes how changes in individual features impact the model output.

4. **Decision Plot:**

Explains how the model's decisions are influenced by the cumulative contribution of features. The decision plot illustrates the cumulative contribution of each feature to the model's decision for an individual prediction. This provides a step-by-step explanation of how the model arrived at a decision, offering full transparency in healthcare decisions.

Dataset description:

The dataset used for this project is the **Stroke Prediction Dataset**, which contains the following features:

1. **Age:** The age of the patient
2. **Gender:** Gender of the patient (Male/Female)
3. **Hypertension:** Whether the patient has a history of hypertension (1 = Yes, 0 = No)
4. **Heart Disease:** Whether the patient has a history of heart disease (1 = Yes, 0 = No)
5. **Smoking Status:** Patient's smoking history (never smoked, formerly smoked, smokes)
6. **BMI:** Body Mass Index
7. **Average Glucose Level:** The average glucose level of the patient

The target variable in the dataset is whether the patient had a stroke (1 = Stroke, 0 = No Stroke).

Pre-processing steps:

To prepare the data for modeling:

1. **Handling Missing Values:** Upon inspecting the dataset, no missing values were found.

2. **Encoding Categorical Variables:** Categorical features like Gender, Smoking Status, and Work Type were label-encoded to convert them into numerical values, as required by the Random Forest model.
3. **Correlation Analysis:** A correlation matrix was generated to identify the strength of relationships between features and the target variable. Features such as Age, Hypertension, and Average Glucose Level showed higher correlation with the target, indicating their importance in predicting stroke risk.

Output:

Data preprocessing:

```
Loading and PreProcessing the dataset

[ ] import pandas as pd
    df = pd.read_csv('healthcare-dataset-stroke-data.csv')

df.head()
```

| | 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 |

```
df.drop(['id'], axis=1, inplace=True)
df.head()
```

| | 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 |

```
[ ] df = pd.get_dummies(df, columns=['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status'])
```

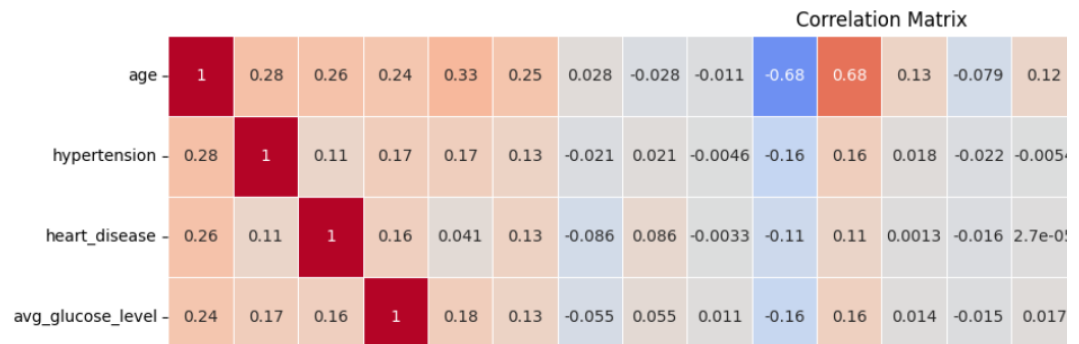
```

import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix = df.corr()

# Plotting the heatmap
plt.figure(figsize=(20, 20))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()

```



```

[ ] from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    df[['age', 'avg_glucose_level', 'bmi']] = scaler.fit_transform(df[['age', 'avg_glucose_level', 'bmi']])

```

Training the model:

Splitting the dataset for training and testing

```

from sklearn.model_selection import train_test_split
X = df.drop(['stroke'], axis=1)
y = df['stroke']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

Model training

```

[ ] from sklearn.ensemble import RandomForestClassifier
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)

```



RandomForestClassifier

RandomForestClassifier(random_state=42)

Model Evaluation:

Evaluation of the model

```
[ ] from sklearn.metrics import accuracy_score
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
➦ Accuracy: 0.9403131115459883
```

```
▶ from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
➦ [[960  0]
   [ 61  1]]
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.94 | 1.00 | 0.97 | 960 |
| 1 | 1.00 | 0.02 | 0.03 | 62 |
| accuracy | | | 0.94 | 1022 |
| macro avg | 0.97 | 0.51 | 0.50 | 1022 |
| weighted avg | 0.94 | 0.94 | 0.91 | 1022 |

Explainable AI:

Explainable Ai part

```
▶ pip install shap
```

```
➦ Requirement already satisfied: shap in /usr/local/lib/python3.10/dist-packages (0.46.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from shap) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from shap) (1.13.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from shap) (1.5.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from shap) (2.1.4)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-packages (from shap) (4.66.5)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packages (from shap) (24.1)
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.10/dist-packages (from shap) (0.0.8)
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba->shap) (0.43.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2024.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
```

```
[ ] import shap

# Initialize the SHAP TreeExplainer for RandomForest
explainer = shap.TreeExplainer(rf_model)

# Calculate SHAP values for the test set
shap_values = explainer.shap_values(X_test)

[ ] print("Shape of SHAP values:", shap_values.shape)
    print("Shape of X_test:", X_test.shape)
```

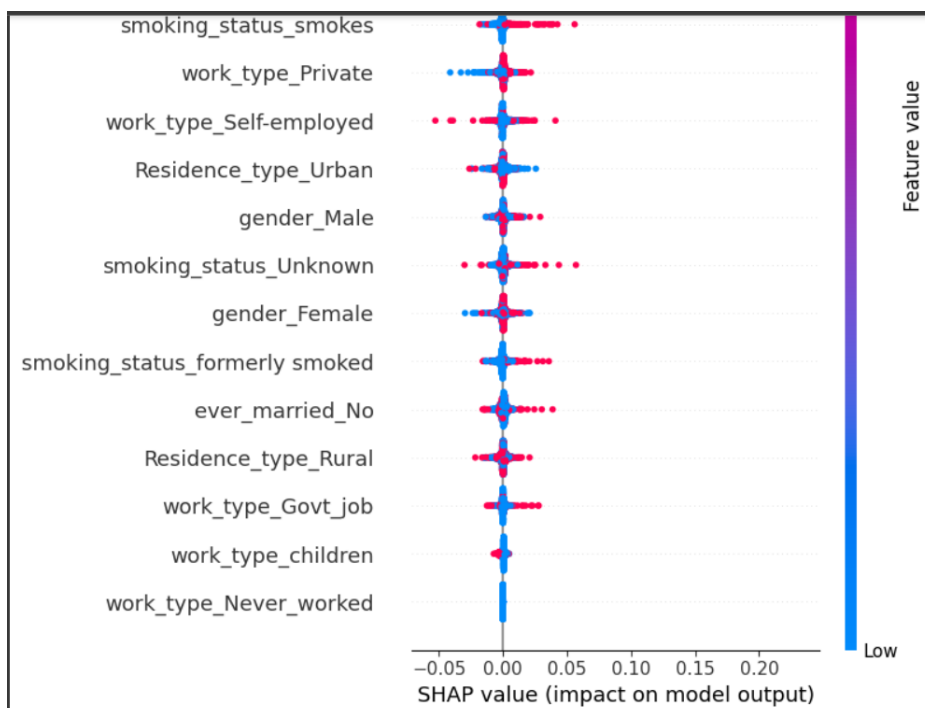
⇒ Shape of SHAP values: (1022, 21, 2)
Shape of X_test: (1022, 21)

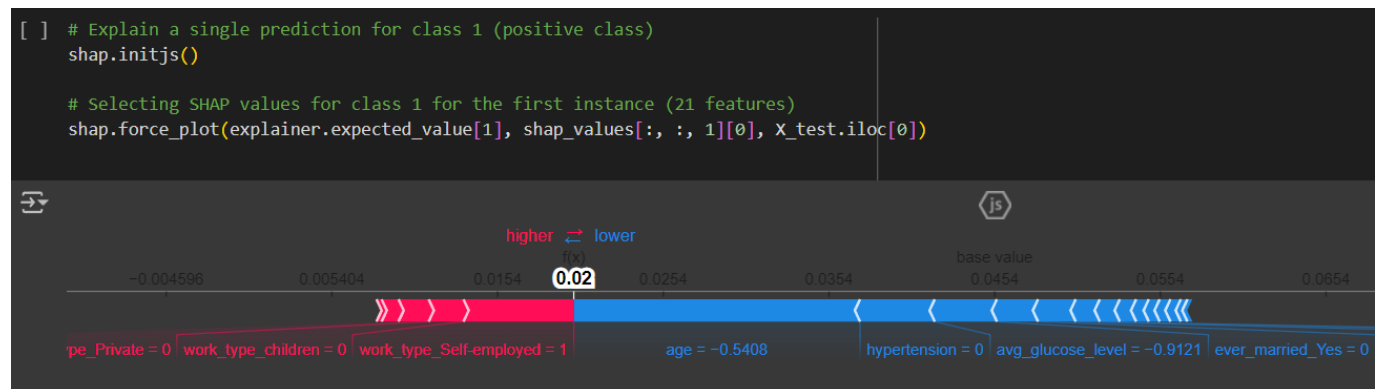
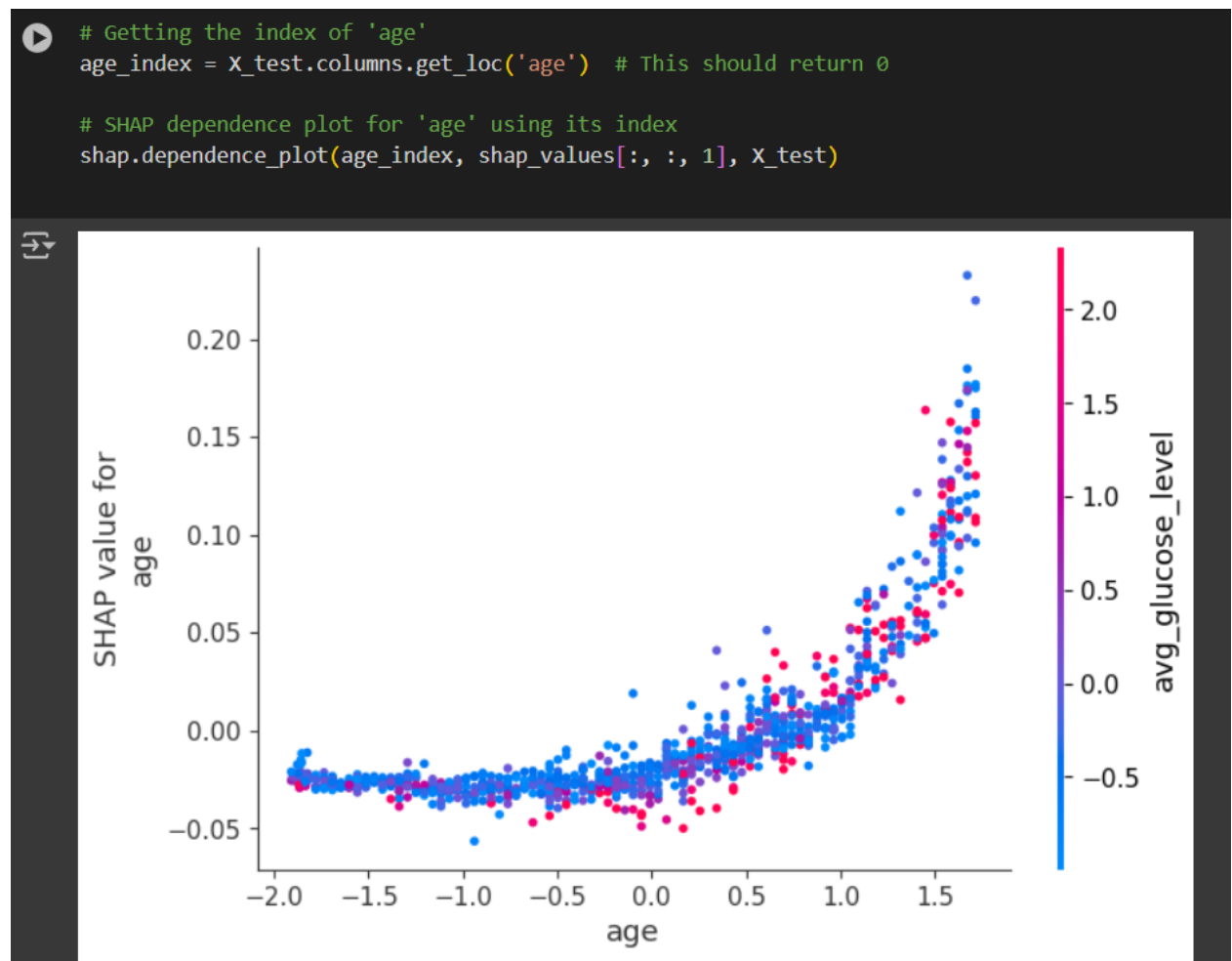
Summary plot:

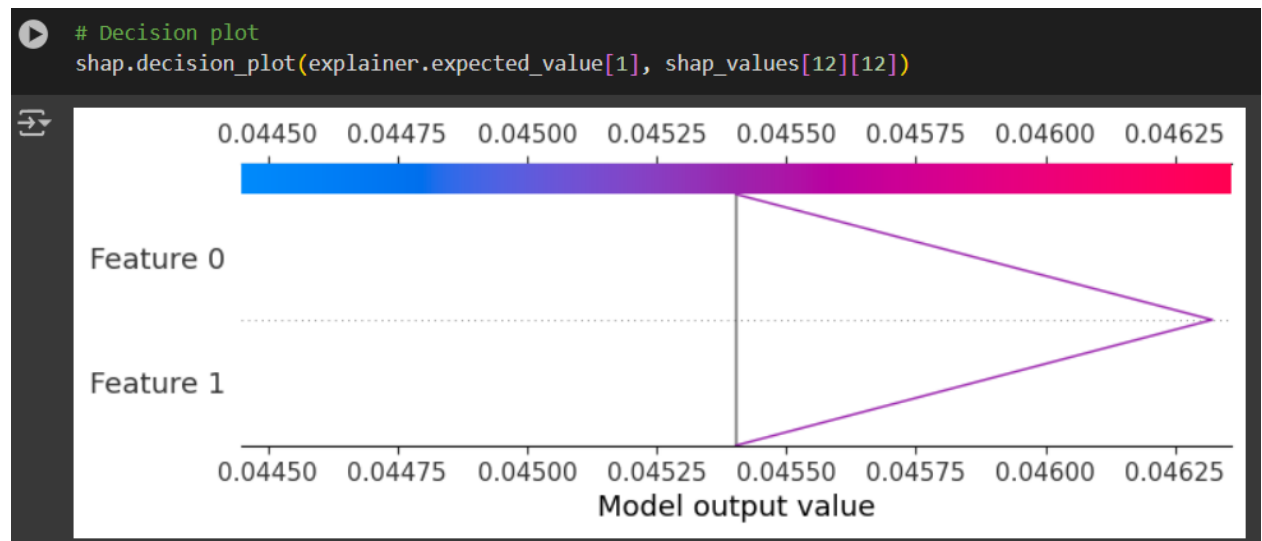
```
[ ] # For binary classification, shap_values[0] corresponds to class 0 and shap_values[1] corresponds to class 1.

# If you want to visualize SHAP values for class 1 (positive class)
shap.summary_plot(shap_values[:, :, 1], X_test)

# If you want to visualize SHAP values for class 0 (negative class)
# shap.summary_plot(shap_values[:, :, 0], X_test)
```



Force Plot:**Dependence plot:**

Decision Plot:**Giving Input to the model:**

```
import pandas as pd

# Example input data (replace with your actual input data)
input_data = {
    'age': [60],
    'hypertension': [1],
    'heart_disease': [1],
    'avg_glucose_level': [85.5],
    'bmi': [26.1],
    'gender': ['Male'],
    'ever_married': ['Yes'],
    'work_type': ['Private'],
    'Residence_type': ['Urban'],
    'smoking_status': ['smoked']
}

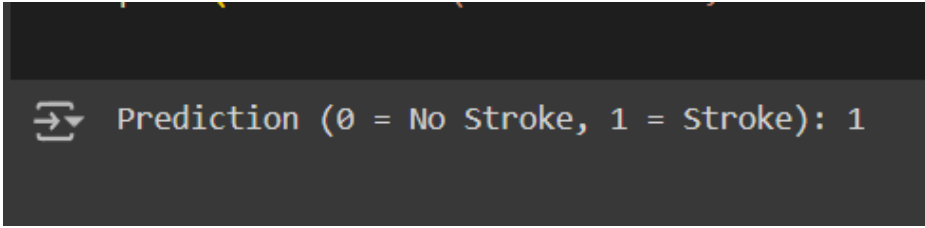
# Convert input data to DataFrame
input_df = pd.DataFrame(input_data)

# One-hot encode the input data to match training data encoding
input_encoded = pd.get_dummies(input_df)

# Align input_encoded with training data's columns (replace 'X_train.columns' with your actual column list)
input_encoded = input_encoded.reindex(columns=X_train.columns, fill_value=0)

# Make prediction using the trained model
prediction = rf_model.predict(input_encoded)

# Output the prediction result (0 = No Stroke, 1 = Stroke)
print("Prediction (0 = No Stroke, 1 = Stroke):", prediction[0])
```

**Conclusion:**

In this project, we developed a stroke prediction model using Random Forest based on key health metrics. The model was evaluated using accuracy, precision, recall, and F1-score to ensure reliable performance. To enhance the interpretability of the predictions, we applied SHAP, which provided clear insights into feature importance and individual predictions through visualizations like summary, dependence, decision and force plots.

This project demonstrates the potential of machine learning in healthcare, combining accurate predictions with explainability to support better decision-making in stroke prevention.