# Data Preparation

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

# Data Understanding

# Data Preprocessing

# Model 1 (Logistic Regression):

## Model Preparation

* Split x to y parameters

|  |
| --- |
| y = numerical\_columns['stroke']  X = numerical\_columns.drop(columns=['stroke']) |

* Split to train, test, val

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.051, random\_state=5) |

* Normalize x value

|  |
| --- |
| scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_test\_scaled = scaler.transform(X\_test) |

* Create model

|  |
| --- |
| lr\_model = LogisticRegression(class\_weight='balanced')  lr\_model.fit(X\_train\_scaled, y\_train) |

## Model Train

|  |
| --- |
| lr\_model.score( X\_test\_scaled, y\_test) |

## Model Result

Model accuracy: 0.7290836653386454

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.99 | 0.73 | 0.84 | 245 |
| Stroke | 0.07 | 0.83 | 0.13 | 6 |
| accuracy |  |  | 0.73 | 251 |
| macro avg | 0.53 | 0.78 | 0.48 | 251 |
| weighted avg | 0.97 | 0.73 | 0.82 | 251 |

A graph with blue squares

AI-generated content may be incorrect.

## Optimize Model

Logistic Regression tidak optimal untuk data yang imbalance. Karena daset dengan label “Stroke” (6 data) dan “No Stroke” (245 data) memiliki jumlah yang tidak balance, sehingga perlu dilakukan balancing data sebelum train&test model.

* Tahap balancing data menggunakan SMOTE (Synthetic Minority Over-sampling Technique)

Code:

|  |
| --- |
| from imblearn.over\_sampling import SMOTE  sm = SMOTE()  X\_resampled, y\_resampled = sm.fit\_resample(X, y) |

## Model Train after optimized

|  |
| --- |
| lr\_model.score( X\_test\_scaled, y\_test) |

## Model Result after optimized

Model Accuracy: 0.8916666666666667

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.91 | 0.87 | 0.89 | 236 |
| Stroke | 0.88 | 0.91 | 0.90 | 244 |
| accuracy |  |  | 0.89 | 480 |
| macro avg | 0.89 | 0.89 | 0.89 | 480 |
| weighted avg | 0.89 | 0.89 | 0.89 | 480 |

A blue squares with numbers and a number of labels

AI-generated content may be incorrect.

## Reference:

# Model 2 (Random Forest)

## Model Preparation

* Split x to y parameters

|  |
| --- |
| y = numerical\_columns['stroke']  X = numerical\_columns.drop(columns=['stroke']) |

* Split to train, test, val

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.051, random\_state=1) |

* Normalize x value

|  |
| --- |
| scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_test\_scaled = scaler.transform(X\_test) |

* Create model

|  |
| --- |
| rforest\_model = RandomForestClassifier(class\_weight='balanced')  rforest\_model.fit(X\_train\_scaled, y\_train) |

## Model Train

|  |
| --- |
| rforest\_model.score(X\_test\_scaled, y\_test) |

## Model Result

Model Accuracy: 0.9800796812749004

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.98 | 1.0 | 0.99 | 246 |
| Stroke | 0.00 | 0.00 | 0.00 | 5 |
| accuracy |  |  | 0.98 | 251 |
| macro avg | 0.49 | 0.50 | 0.49 | 251 |
| weighted avg | 0.96 | 0.98 | 0.97 | 251 |

A graph with numbers and a blue square

AI-generated content may be incorrect.

## Optimize Model

Dari hasil akurasi, meskipun akurasi yang didapat memiliki nilai yang tinggi, namun model randomforest tidak dapat mendeteksi/menganalisa value label 1 (Stroke). Hal ini disebabkan karena dataset yang imbalance antara value 1(stroke) dan 0 (Non stroke).

* Balancing dataset dapat dilakukan dengan SMOTE (Synthetic Minority Over-sampling Technique)

Code:

|  |
| --- |
| from imblearn.over\_sampling import SMOTE  sm = SMOTE()  X\_resampled, y\_resampled = sm.fit\_resample(X, y) |

## Model Train after optimized

|  |
| --- |
| rforest\_model.score(X\_test\_scaled, y\_test) |

## Model Result after optimized

Model Accuracy: 0.970

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.96 | 0.98 | 0.97 | 233 |
| Stroke | 0.98 | 0.96 | 0.97 | 247 |
| accuracy |  |  | 0.97 | 480 |
| macro avg | 0.97 | 0.97 | 0.97 | 480 |
| weighted avg | 0.97 | 0.97 | 0.97 | 480 |

A blue squares with numbers and a number on it

AI-generated content may be incorrect.

A blue squares with white text

AI-generated content may be incorrect.

## Reference:

# Model 3 (Deep Learning Using Sequential Models)

## Model Preparation

* Split x to y parameters

|  |
| --- |
| y = numerical\_columns['stroke']  X = numerical\_columns.drop(columns=['stroke']) |

* Split to train, test, val

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.63, shuffle=True, random\_state=21) |

* Normalize x value

|  |
| --- |
| scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_test\_scaled = scaler.transform(X\_test) |

* Create model

|  |
| --- |
| model = Sequential()  model.add(Dense(16, activation='relu', input\_dim=X\_train\_scaled.shape[1]))  model.add(Dense(50, activation="relu"))  model.add(Dense(20, activation="relu"))  model.add(Dropout(0.25))  model.add(Dense(1, activation="sigmoid"))  print (model.summary()) |

A screenshot of a computer

AI-generated content may be incorrect.

## Model Train

|  |
| --- |
| Epoch 1/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **2s** 18ms/step - accuracy: 0.8936 - loss: 0.4938 - val\_accuracy: 0.9532 - val\_loss: 0.3116  Epoch 2/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - accuracy: 0.9549 - loss: 0.2999 - val\_accuracy: 0.9532 - val\_loss: 0.2337  Epoch 3/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - accuracy: 0.9553 - loss: 0.2241 - val\_accuracy: 0.9532 - val\_loss: 0.2142  Epoch 4/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - accuracy: 0.9699 - loss: 0.1559 - val\_accuracy: 0.9532 - val\_loss: 0.2036  Epoch 5/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - accuracy: 0.9609 - loss: 0.1705 - val\_accuracy: 0.9532 - val\_loss: 0.1918  Epoch 6/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - accuracy: 0.9626 - loss: 0.1576 - val\_accuracy: 0.9532 - val\_loss: 0.1855  Epoch 7/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - accuracy: 0.9551 - loss: 0.1627 - val\_accuracy: 0.9532 - val\_loss: 0.1822  Epoch 8/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - accuracy: 0.9618 - loss: 0.1431 - val\_accuracy: 0.9532 - val\_loss: 0.1806  Epoch 9/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - accuracy: 0.9518 - loss: 0.1736 - val\_accuracy: 0.9532 - val\_loss: 0.1801  Epoch 10/10  **18/18** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - accuracy: 0.9567 - loss: 0.1428 - val\_accuracy: 0.9532 - val\_loss: 0.1801 |

## Model Result

Loss Graph

A graph of a line

AI-generated content may be incorrect.

Accuracy Graph:

A green and red line graph

AI-generated content may be incorrect.

Model accuracy: 0.9586162567138672

Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.96 | 1.00 | 0.98 | 2965 |
| Stroke | 0.00 | 0.00 | 0.00 | 128 |
| accuracy |  |  | 0.96 | 3093 |
| macro avg | 0.48 | 0.50 | 0.49 | 3093 |
| weighted avg | 0.92 | 0.96 | 0.94 | 3093 |

Confusion Matrix:

A blue and white graph

AI-generated content may be incorrect.

## Optimize Model

Meskipun model memiliki akurasi yang tinggi, namun model belum dapat mendeteksi Stroke. Hasil grafik accuracy juga menunjukkan tidak ada peningkatan akurasi. Hal ini dikarenakan imbalance dataset yang membuat model hanya memprediksi label yang lebih besar (label No Stroke).

Langkah optimasi:

* Menggunakan SMOTE untuk menyamakan ukuran data label Stroke dan No Stroke

|  |
| --- |
| from imblearn.over\_sampling import SMOTE  sm = SMOTE()  X\_resampled, y\_resampled = sm.fit\_resample(X, y) |

* Menaikkan/menambah Dense layer

|  |
| --- |
| model = Sequential()  model.add(Dense(16, activation='relu', input\_dim=X\_train\_scaled.shape[1]))  model.add(Dense(50, activation="relu"))  model.add(Dense(20, activation="relu"))  model.add(Dropout(0.25))  model.add(Dense(23, activation="relu"))  model.add(Dense(1, activation="sigmoid"))  print (model.summary()) |

A screenshot of a computer

AI-generated content may be incorrect.

* Mengurangi test size, randomstate

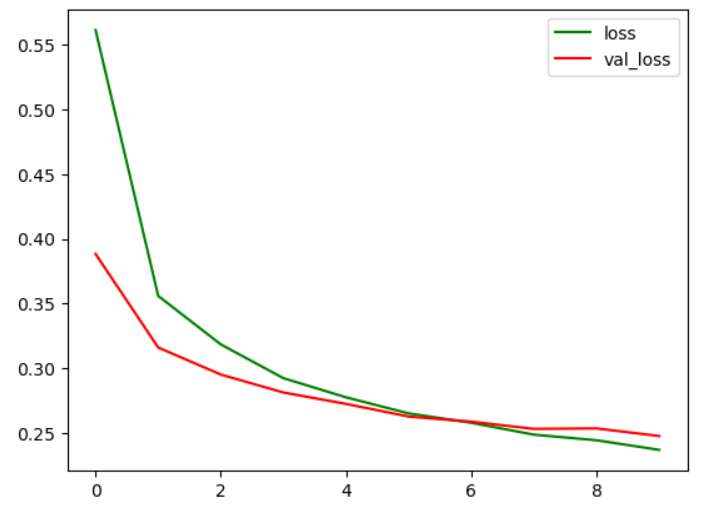
|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_resampled, y\_resampled, test\_size=0.04, shuffle=True, random\_state=12) |

## Model Train after optimized

|  |
| --- |
| Epoch 1/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **2s** 6ms/step - accuracy: 0.6433 - loss: 0.6376 - val\_accuracy: 0.8263 - val\_loss: 0.3883  Epoch 2/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.8331 - loss: 0.3729 - val\_accuracy: 0.8596 - val\_loss: 0.3159  Epoch 3/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.8548 - loss: 0.3290 - val\_accuracy: 0.8706 - val\_loss: 0.2950  Epoch 4/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.8772 - loss: 0.3015 - val\_accuracy: 0.8789 - val\_loss: 0.2811  Epoch 5/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.8792 - loss: 0.3004 - val\_accuracy: 0.8850 - val\_loss: 0.2723  Epoch 6/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.8928 - loss: 0.2665 - val\_accuracy: 0.8881 - val\_loss: 0.2626  Epoch 7/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.9026 - loss: 0.2572 - val\_accuracy: 0.8900 - val\_loss: 0.2586  Epoch 8/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.9027 - loss: 0.2542 - val\_accuracy: 0.8994 - val\_loss: 0.2530  Epoch 9/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.9039 - loss: 0.2462 - val\_accuracy: 0.8983 - val\_loss: 0.2534  Epoch 10/10  **85/85** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - accuracy: 0.9097 - loss: 0.2335 - val\_accuracy: 0.9017 - val\_loss: 0.2475 |

## Model Result after optimized

Loss Graph:



Accuracy Graph:

A graph with a green line

AI-generated content may be incorrect.

Model accuracy: 0.9095744490623474

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.92 | 0.90 | 0.91 | 190 |
| Stroke | 0.90 | 0.92 | 0.91 | 186 |
| accuracy |  |  | 0.91 | 376 |
| macro avg | 0.91 | 0.91 | 0.91 | 376 |
| weighted avg | 0.91 | 0.91 | 0.91 | 376 |

Confusion Matrix:

A blue squares with white text

AI-generated content may be incorrect.

## Reference:

# Model 4 (Deep Learning Using TabNetClassifier)

## Model Preparation

* Split x and y parameters

|  |
| --- |
| X = numerical\_columns.drop(columns=['stroke'])  y = data['stroke'] |

* Split to train, test, val

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.87, random\_state=45, stratify=y) |

* Normalize x value

|  |
| --- |
| scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_test\_scaled = scaler.transform(X\_test) |

* Create model

|  |
| --- |
| clf = TabNetClassifier(verbose=3, seed=100) |

* Calculate class weight

|  |
| --- |
| class\_weights = compute\_class\_weight(class\_weight='balanced', classes=np.unique(y\_train\_np), y=y\_train\_np)  weights\_dict = {i: class\_weights[i] for i in range(len(class\_weights))} |

* Create train model

|  |
| --- |
| clf.fit(  X\_train\_np, y\_train\_np,  eval\_set=[(X\_test\_np, y\_test\_np)],  max\_epochs=60,  patience=10,  batch\_size=400,  virtual\_batch\_size=200,  weights=weights\_dict,  eval\_metric=['accuracy']  ) |

## Model Train

|  |
| --- |
| epoch 0 | loss: 0.73233 | val\_0\_accuracy: 0.50995 | 0:00:00s  epoch 3 | loss: 0.55627 | val\_0\_accuracy: 0.56872 | 0:00:00s  epoch 6 | loss: 0.47698 | val\_0\_accuracy: 0.6207 | 0:00:01s  epoch 9 | loss: 0.42488 | val\_0\_accuracy: 0.66191 | 0:00:02s  epoch 12 | loss: 0.3445 | val\_0\_accuracy: 0.67853 | 0:00:03s  epoch 15 | loss: 0.402 | val\_0\_accuracy: 0.70311 | 0:00:03s  epoch 18 | loss: 0.31994 | val\_0\_accuracy: 0.7174 | 0:00:04s  epoch 21 | loss: 0.23884 | val\_0\_accuracy: 0.7195 | 0:00:05s  epoch 24 | loss: 0.24585 | val\_0\_accuracy: 0.75018 | 0:00:05s  epoch 27 | loss: 0.27238 | val\_0\_accuracy: 0.78881 | 0:00:06s  epoch 30 | loss: 0.20849 | val\_0\_accuracy: 0.78928 | 0:00:07s  epoch 33 | loss: 0.18627 | val\_0\_accuracy: 0.75205 | 0:00:07s  epoch 36 | loss: 0.2489 | val\_0\_accuracy: 0.75252 | 0:00:08s  Early stopping occurred at epoch 38 with best\_epoch = 28 and best\_val\_0\_accuracy = 0.79653 |

## Model Result

Model accuracy: 0.7965347693748537

A graph with a red line

AI-generated content may be incorrect.

Loss:

A green line graph with numbers and a white background

AI-generated content may be incorrect.

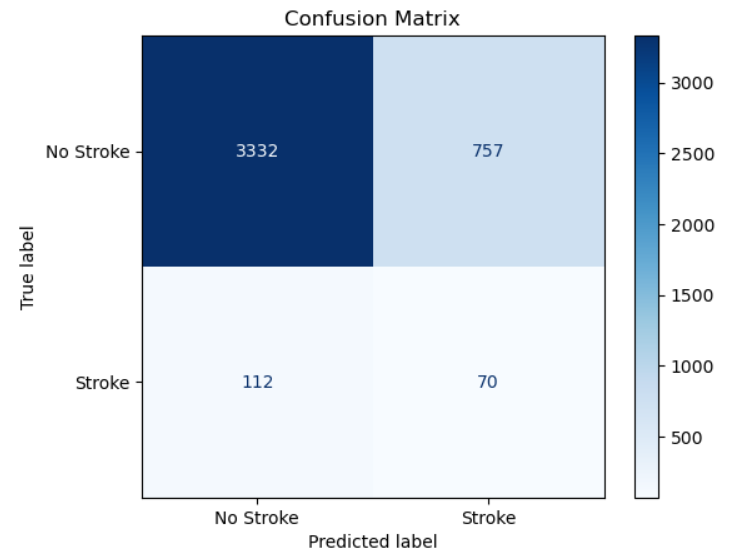
Loss & Val:

A graph showing a line graph

AI-generated content may be incorrect.

Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.97 | 0.81 | 0.88 | 4089 |
| Stroke | 0.08 | 0.38 | 0.14 | 182 |
| accuracy |  |  | 0.80 | 4271 |
| macro avg | 0.53 | 0.60 | 0.51 | 4271 |
| weighted avg | 0.93 | 0.80 | 0.85 | 4271 |



## Optimize Model

Dari akurasi yang didapatkan, model masih dapat ditingkatkan akurasinya. Akurasi yang didapatkan sudah baik (0.8 / 80%). Akan tetapi data akurasi dan data confusion matrix mendapatkan hasil yang kurang baik. Perbedaan persebaran data yang terlalu besar yaitu 4000 data ‘No Stroke’ dan 182 data ‘Stroke’. Untuk meningkatkan akurasi dari model, dapat dilakukan dengan beberapa cara yaitu:

* Menggunakan SMOTE untuk mengolah data yang ada supaya balance
* Mengurangi jumlah dataset

## Model Train after optimized

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_resampled, y\_resampled, test\_size=0.02, random\_state=45, stratify=y\_resampled) |

|  |
| --- |
| epoch 0 | loss: 0.46002 | val\_0\_accuracy: 0.82447 | 0:00:00s  epoch 3 | loss: 0.3322 | val\_0\_accuracy: 0.8883 | 0:00:03s  epoch 6 | loss: 0.27599 | val\_0\_accuracy: 0.87234 | 0:00:05s  epoch 9 | loss: 0.24425 | val\_0\_accuracy: 0.93617 | 0:00:08s  epoch 12 | loss: 0.23396 | val\_0\_accuracy: 0.95213 | 0:00:11s  epoch 15 | loss: 0.23857 | val\_0\_accuracy: 0.90426 | 0:00:13s  epoch 18 | loss: 0.22647 | val\_0\_accuracy: 0.91489 | 0:00:16s  epoch 21 | loss: 0.22596 | val\_0\_accuracy: 0.93085 | 0:00:18s  Early stopping occurred at epoch 22 with best\_epoch = 12 and best\_val\_0\_accuracy = 0.95213 |

## Model Result after optimized

Model accuracy: 95.21

Val\_Accuracy

A graph with a line drawn on it

AI-generated content may be incorrect.

Val\_Loss:

A graph with a line

AI-generated content may be incorrect.

Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.97 | 0.94 | 0.95 | 94 |
| Stroke | 0.94 | 0.97 | 0.95 | 94 |
| accuracy |  |  | 0.95 | 188 |
| macro avg | 0.53 | 0.60 | 0.51 | 188 |
| weighted avg | 0.93 | 0.80 | 0.85 | 188 |

A diagram of a diagram

AI-generated content may be incorrect.

## Reference:

# Model 5 (Xai SHAP LogisticRegression)

## Model Preparation

* Shap explainer

|  |
| --- |
| explainer = shap.Explainer(lr\_model, X\_train\_scaled)  shap\_values = explainer(X\_test\_scaled) |

## Model Result

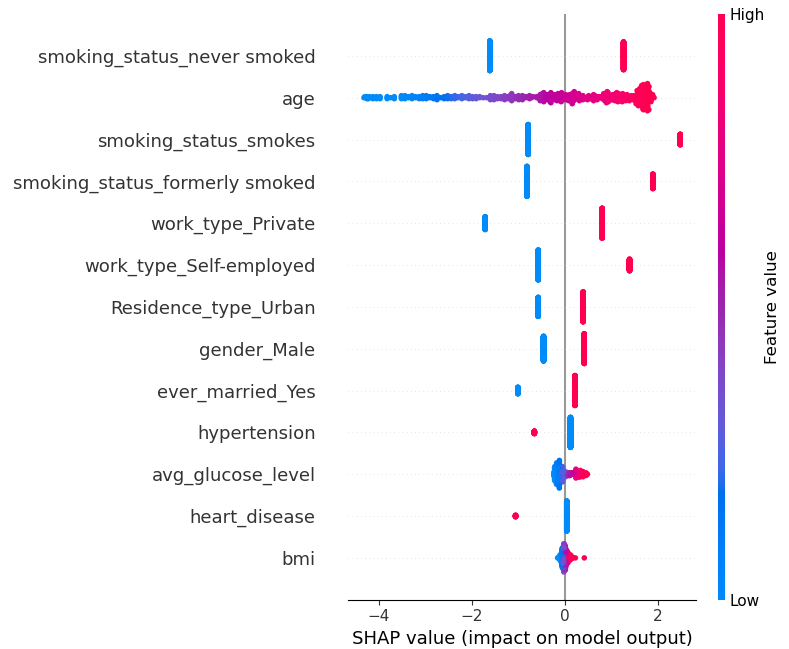
Model Accuracy: 0.8791666666666667

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| No Stroke | 0.89 | 0.86 | 0.88 | 236 |
| Stroke | 0.87 | 0.89 | 0.88 | 244 |
| accuracy |  |  | 0.88 | 480 |
| macro avg | 0.88 | 0.88 | 0.88 | 480 |
| weighted avg | 0.88 | 0.88 | 0.88 | 480 |

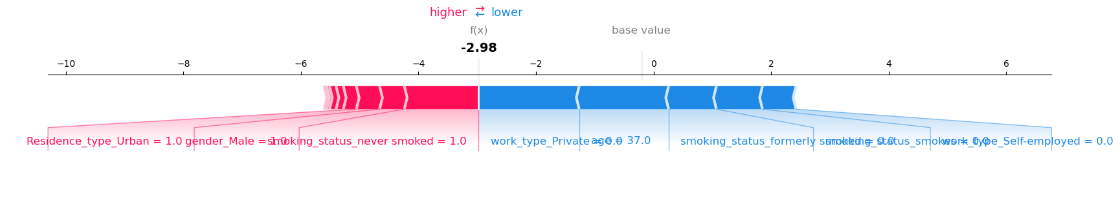
A graph of a graph with blue squares

AI-generated content may be incorrect.

* Shap Summary Plot:



* Shap Force Plot:



* Shap Coeficient:

A graph with blue bars

AI-generated content may be incorrect.

## Analyze SHAP

* Model Coefficient Analyze

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Coefficient | Absolute\_Coefficient | Odds\_Ratio |
| age | 1.692851 | 1.692851 | 5.434952 |
| smoking\_status\_never smoked | 1.43751 | 1.43751 | 4.210199 |
| smoking\_status\_smokes | 1.385832 | 1.385832 | 3.998151 |
| smoking\_status\_formerly smoked | 1.260527 | 1.260527 | 3.527281 |
| work\_type\_Private | 1.1532 | 1.1532 | 3.168317 |
| work\_type\_Self-employed | 0.879443 | 0.879443 | 2.409556 |
| ever\_married\_Yes | 0.47965 | 0.47965 | 1.615509 |
| Residence\_type\_Urban | 0.468729 | 0.468729 | 1.597962 |
| gender\_Male | 0.436685 | 0.436685 | 1.547569 |
| heart\_disease | -0.23734 | 0.237336 | 0.788726 |
| hypertension | -0.2209 | 0.220903 | 0.801795 |
| avg\_glucose\_level | 0.186979 | 0.186979 | 1.205602 |
| bmi | 0.059518 | 0.059518 | 1.061325 |

* Coeficeient tertinggi merupakan parameter yang dapat menyebabkan stroke.  
  Dari table coefficient, Age memiliki nilai koefisien tertinggi (Usia menyebabkan stroke). Sedangkan nilai negative merupakan koefisien terendah/ tidak berkorelasi secara langsung terhadap akurasi penyebab stroke.

## Reference:

https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability

# Model 6 (Xai Lime)

## Model Preparation

* Lime Explainer

|  |
| --- |
| explainer = lime.lime\_tabular.LimeTabularExplainer(  X\_train\_scaled,  feature\_names=X.columns,  class\_names=['No Stroke', 'Stroke'],  mode='classification',  discretize\_continuous=True  ) |

* Lime Model Creation

|  |
| --- |
| exp = explainer.explain\_instance(X\_test\_scaled[0], lr\_model.predict\_proba, num\_features=10) |

## Model Result

A screenshot of a graph

AI-generated content may be incorrect.

## Analyze Lime

* Untuk summary keseluruhan

A screenshot of a computer

AI-generated content may be incorrect.

* Untuk 1 sample data missdetection:

A screenshot of a graph

AI-generated content may be incorrect.

## Reference:

https://medium.com/data-science/lime-explain-machine-learning-predictions-af8f18189bfe