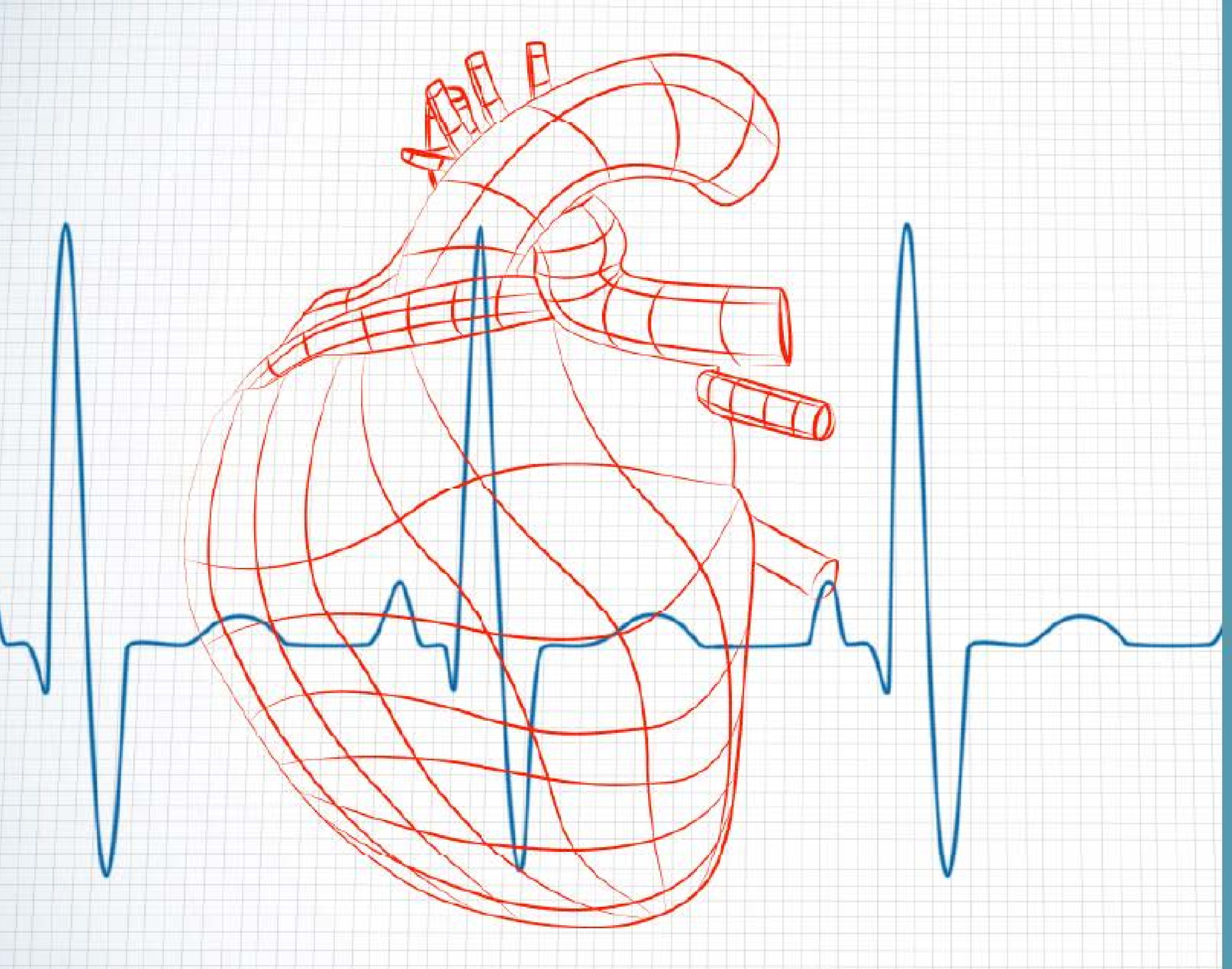


Stroke Prediction

All Code created by:

- Jesse Kranyak
- Jeff Boczkaja
- Mohamed Altoobi
- Siriesha Mandava

2024

Project Purpose Description

Utilizing the Stroke Prediction Dataset from Kaggle we set out to make a machine learning program that will be able to accurately predict whether or not someone will have a stroke. We try out different models that provided us with varying results. We show our results using a few different metrics including balanced accuracy score, F1 scores, precision, and recall.

What do the metrics measure?

Precision measures the accuracy of positive predictions. It is the ratio of true positive predictions to the total number of positive predictions made. In other words, it answers the question, "Of all the instances the model predicted as positive, how many are actually positive?" Precision is particularly important in scenarios where the cost of a false positive is high.

Formula: Precision = True Positives / (True Positives + False Positives)

Precision

What do the metrics measure?

Recall

Recall, also known as sensitivity or true positive rate, measures the ability of a model to find all the relevant cases within a dataset. It is the ratio of true positive predictions to the total number of actual positives. Recall answers the question, "Of all the actual positives, how many did the model successfully identify?" Recall is crucial in situations where missing a positive instance is costly.

Formula: Recall = True Positives / (True Positives + False Negatives)

What do the metrics measure?

F1 Score

The F1 Score is the mean of precision and recall. It provides a single metric that balances both the precision and recall of a classification model, which is particularly useful when you want to compare two or more models. The F1 Score is especially valuable when the distribution of class labels is imbalanced. A high F1 Score indicates that the model has low false positives and low false negatives, so it's correctly identifying real positives and negatives.

Formula: F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

What do the metrics measure?

Balanced Accuracy Score

Balanced Accuracy Score is defined as the average of recall obtained on each class, meaning it considers both the true positive rate and the true negative rate. It calculates the accuracy of the model by taking into account the balance between classes. For a binary classification problem, it would be the average of the proportion of correctly predicted positive observations to the total positive observations and the proportion of correctly predicted negative observations to the total negative observations.

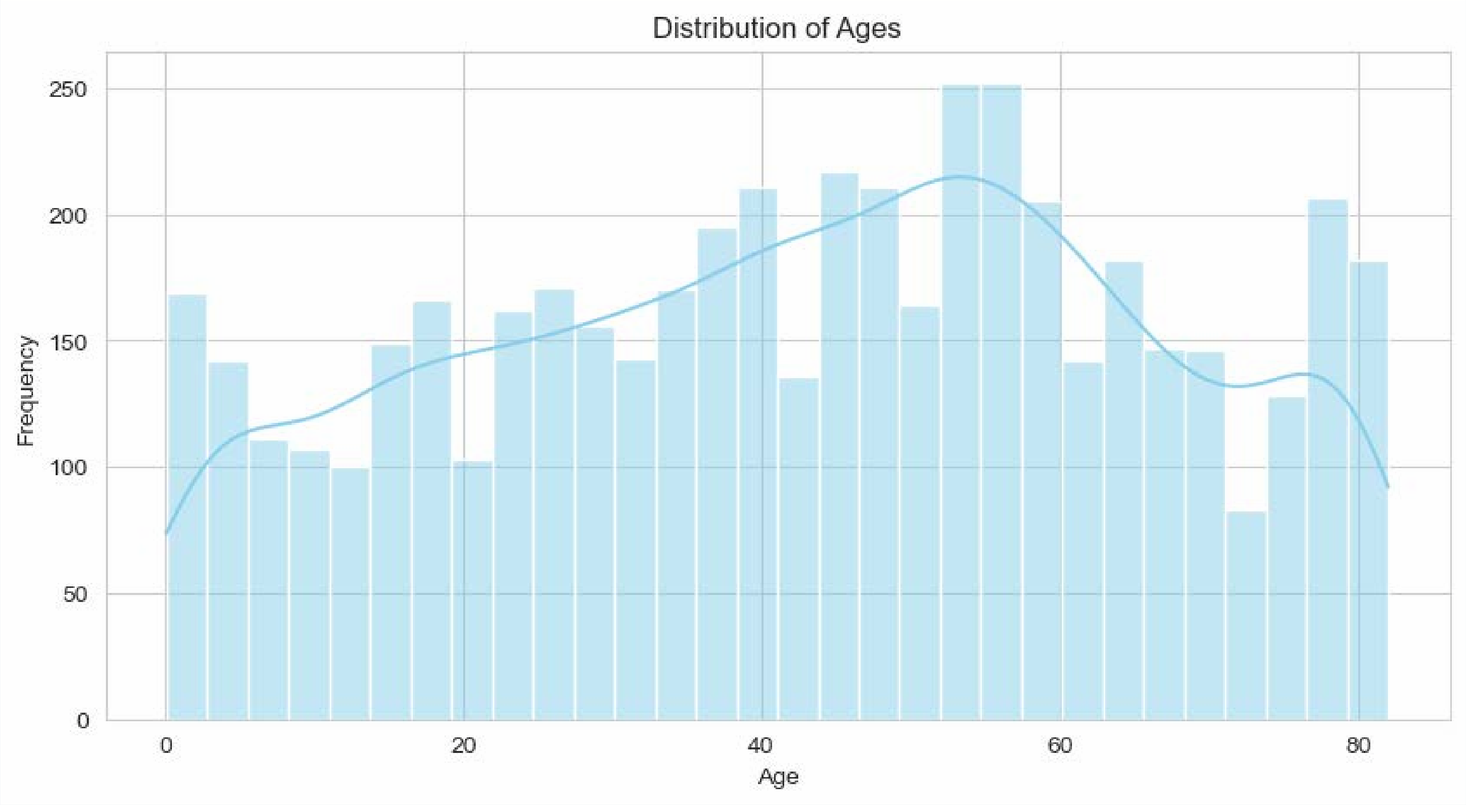
Formula: Balanced Accuracy Score = (2/1) * ((TP / (TP + FN)) + (TN / (TN + FP)))

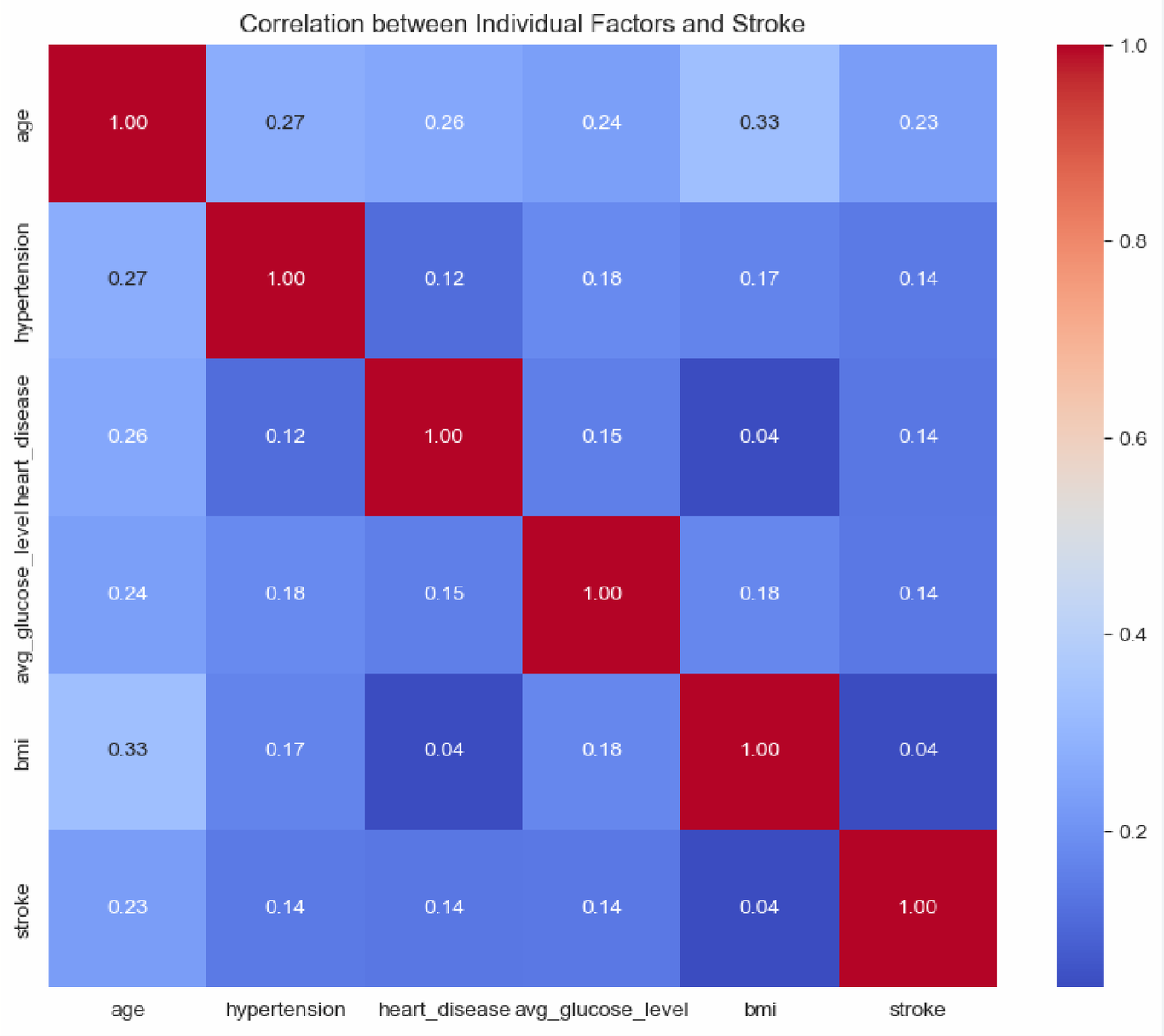
Importing Data

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import SMOTENC
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.metrics import balanced_accuracy_score, classification_report, confusion_matrix, precision_recall_fscore_support
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV

df = pd.read_csv("healthcare-dataset-stroke-data.csv")
df
```

Analyzing and Exploring our Data

```python
#Lets go ahead and loop through all of our columns and see what data they reveal

def describe_df(df: pd.DataFrame):
    print(f"The dataset contains {df.shape[1]} columns and {len(df)} rows")
    for col in df.columns:
        col_dtype = df[col].dtype
        print(f"\nColumn: {col} ({col_dtype})")
        if col_dtype == 'object':
            print(f"--- Percentage of NaNs: {df[col].isna().sum() / len(df[col]) * 100}")
            print(f"--- Unique values:\n {df[col].unique()}")
        else:
            print(f"--- Summary statistics:\n {df[col].describe()}")
describe_df(df)
```




Correlation between Individual Factors and Stroke
age
hypertension
heart_disease
avg_glucose_level
bmi
stroke
1.00 0.27 0.26 0.24 0.33 0.23
0.27 1.00 0.12 0.18 0.17 0.14
0.26 0.12 1.00 0.15 0.04 0.14
0.24 0.18 0.15 1.00 0.18 0.14
0.33 0.17 0.04 0.18 1.00 0.04
0.23 0.14 0.14 0.14 0.04 1.00
1.0
0.8
0.6
0.4
0.2

Encoding our data for use in machine learning

In machine learning, encoding data is essential for preparing categorical variables to be used as input in algorithms. Since most machine learning models require numerical data, categorical variables such as gender, smoking status, or work type need to be encoded into numerical form. This process ensures that the model can effectively interpret and learn from these features, enabling it to make accurate predictions or classifications based on the input data.

Scaling method

Normalization

rescales the features to a fixed range, usually 0 to 1.

Standardization

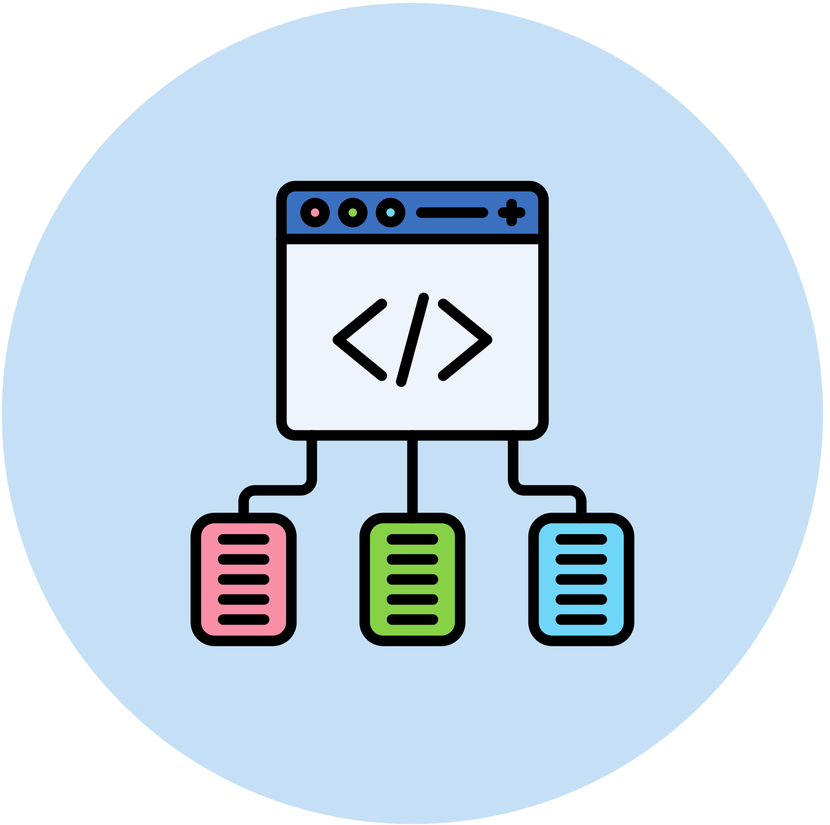
rescales data so that it has a mean of 0 and a standard deviation of 1.

Decision Tree

A decision tree is a hierarchical model that helps in making decisions by mapping out possible outcomes based on different conditions. It's a visual representation where each branch represents a decision based on features in the data, ultimately leading to a prediction or classification.

Random Forest

A Random Forest is a machine learning method used in both classification and regression tasks. It operates by constructing a multitude of decision trees during training time and outputs the mode or average prediction of the individual trees.

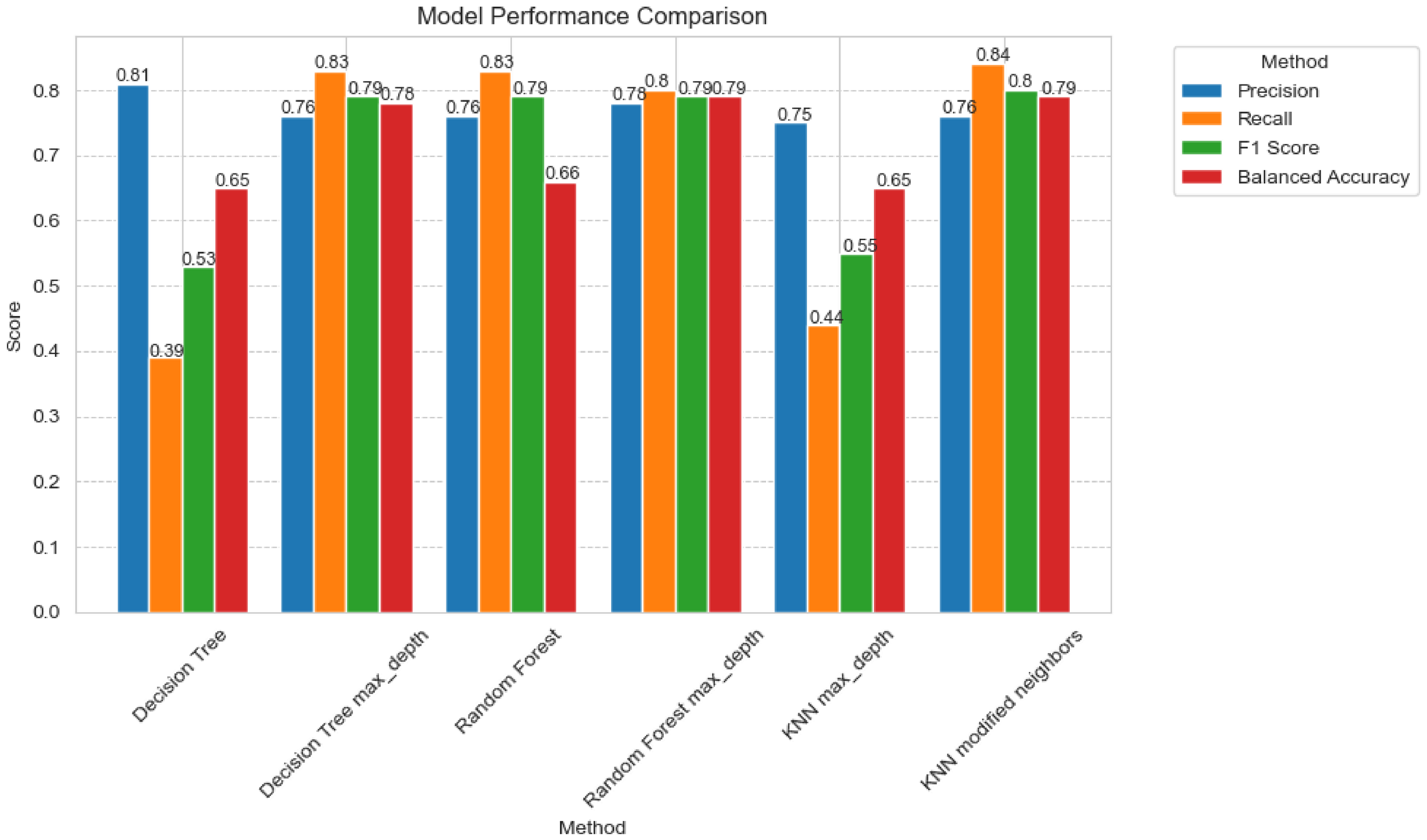


K Nearest Neighbors

The k-nearest neighbors algorithm predicts the label of a data point based on the labels of its 'k' closest neighbors in the dataset. To classify a new instance, KNN calculates the distance between the instance and all points in the training set, identifies the 'k' nearest points, and then uses a majority vote among these neighbors to determine the instance's label. For regression tasks, it averages the values of these neighbors instead.

Conclusion

Despite the challenges presented by lifestyle data, the Random Forest Classifier was the standout model upon tuning, specifically when adjusted to the optimal max depth. This model achieved a balanced accuracy score of %80, marking it as the most effective among the classifiers we tested for predicting stroke potential. The Random Forest Classifier with an appropriate max depth is what we would recommended as a tool for stroke prediction, emphasizing the model's utility in clinical settings for early stroke risk assessment.


Model Performance Comparison
Score
Method
Precision
Recall
F1 Score
Balanced Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.81
0.39
0.53
0.65
0.76
0.83
0.79
0.78
0.76
0.83
0.79
0.66
0.78
0.8
0.79
0.79
0.75
0.44
0.55
0.65
0.76
0.84
0.8
0.79
Decision Tree
Decision Tree max_depth
Random Forest
Random Forest max_depth
KNN max_depth
KNN modified neighbors

References

Credits - original code written by: Sirisha Mandava, Jeff Boczkaja, Mohamed Altoobli, Jesse Kranyak

> [Our dataset](https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset) comes from kaggle

> [inspiration 1 for project] (https://www.kaggle.com/code/thomaskonstantin/analyzing-and-modeling-stroke-data) comes from kaggle

> [inspiration 2 for project] (https://www.stat.cmu.edu/capstoneresearch/spring315/2021files/team16.html) comes from kaggle

All images : https://www.freepik.com/