Heart Attack Prediction Using Random Forest Supervised Machine Learning

April 29, 2025

[222]:

```
# Importing the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
  ↪classification_report
from sklearn.tree import DecisionTreeClassifier

# Loading dataset into Jupyter Notebook
data = pd.read_csv('heart-attack-risk-prediction-dataset.csv')

# Exploring dataset
print(data.describe())

print(data.info())

# There is 9651 patients in the dataset.
# Some of the the patients data is not complete so there is less data to␣
  ↪classify their risk levels.
# For this we will replace the null entry with the average in the data set of␣
  ↪what is oberved
```

```
               Age  Cholesterol   Heart rate     Diabetes  Family History  \
count  9651.000000  9651.000000  9651.000000  9377.000000     9377.000000
mean      0.450254     0.499780     0.050756     0.652554        0.488749
std       0.231154     0.284461     0.024922     0.476184        0.499900
min       0.000000     0.000000     0.000000     0.000000        0.000000
25%       0.258427     0.264286     0.035289     0.000000        0.000000
50%       0.460674     0.499780     0.050412     1.000000        0.000000
75%       0.640449     0.739286     0.065995     1.000000        1.000000
max       1.000000     1.000000     1.000000     1.000000        1.000000
```

```
           Smoking      Obesity  Alcohol Consumption  Exercise Hours Per Week  \
count  9377.000000  9377.000000          9377.000000              9651.000000
mean      0.902421     0.500160             0.600192                 0.502110
std       0.296761     0.500027             0.489885                 0.284830
min       0.000000     0.000000             0.000000                 0.000000
25%       1.000000     0.000000             0.000000                 0.259793
50%       1.000000     1.000000             1.000000                 0.502110
75%       1.000000     1.000000             1.000000                 0.747086
max       1.000000     1.000000             1.000000                 1.000000

              Diet  …  Triglycerides  Physical Activity Days Per Week  \
count  9651.000000  …    9651.000000                      9377.000000
mean      1.057093  …       0.503603                         3.501866
std       0.868418  …       0.286183                         2.283833
min       0.000000  …       0.000000                         0.000000
25%       0.000000  …       0.262338                         2.000000
50%       1.000000  …       0.503603                         3.000000
75%       2.000000  …       0.748052                         6.000000
max       3.000000  …       1.000000                         7.000000

       Sleep Hours Per Day  Heart Attack Risk (Binary)  Blood sugar  \
count          9651.000000                 9651.000000  9651.000000
mean              0.504621                    0.345146     0.227018
std               0.327482                    0.475440     0.075577
min               0.000000                    0.000000     0.000000
25%               0.166667                    0.000000     0.227018
50%               0.500000                    0.000000     0.227018
75%               0.833333                    1.000000     0.227018
max               1.000000                    1.000000     1.000000

             CK-MB     Troponin  Heart Attack Risk (Text)  \
count  9651.000000  9651.000000               9651.000000
mean      0.048229     0.036512                  0.398093
std       0.075959     0.059556                  0.737488
min       0.000000     0.000000                  0.000000
25%       0.048229     0.036512                  0.000000
50%       0.048229     0.036512                  0.000000
75%       0.048229     0.036512                  0.000000
max       1.000000     1.000000                  2.000000

       Systolic blood pressure  Diastolic blood pressure
count              9651.000000               9651.000000
mean                  0.449982                  0.497553
std                   0.170344                  0.172033
min                   0.000000                  0.000000
25%                   0.303226                  0.348837
50%                   0.445161                  0.500000
75%                   0.600000                  0.651163
```

```
max                   1.000000                 1.000000

[8 rows x 26 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9651 entries, 0 to 9650
Data columns (total 27 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Age                              9651 non-null   float64
 1   Cholesterol                      9651 non-null   float64
 2   Heart rate                       9651 non-null   float64
 3   Diabetes                         9377 non-null   float64
 4   Family History                   9377 non-null   float64
 5   Smoking                          9377 non-null   float64
 6   Obesity                          9377 non-null   float64
 7   Alcohol Consumption              9377 non-null   float64
 8   Exercise Hours Per Week          9651 non-null   float64
 9   Diet                             9651 non-null   int64
 10  Previous Heart Problems          9377 non-null   float64
 11  Medication Use                   9377 non-null   float64
 12  Stress Level                     9377 non-null   float64
 13  Sedentary Hours Per Day          9651 non-null   float64
 14  Income                           9651 non-null   float64
 15  BMI                              9651 non-null   float64
 16  Triglycerides                    9651 non-null   float64
 17  Physical Activity Days Per Week  9377 non-null   float64
 18  Sleep Hours Per Day              9651 non-null   float64
 19  Heart Attack Risk (Binary)       9651 non-null   float64
 20  Blood sugar                      9651 non-null   float64
 21  CK-MB                            9651 non-null   float64
 22  Troponin                         9651 non-null   float64
 23  Heart Attack Risk (Text)         9651 non-null   int64
 24  Gender                           9651 non-null   object
 25  Systolic blood pressure          9651 non-null   float64
 26  Diastolic blood pressure         9651 non-null   float64
dtypes: float64(24), int64(2), object(1)
memory usage: 2.0+ MB
None
```

[223]:

```
#Clean dataset by replacing null entries with the mean values.
data.fillna(data.median(numeric_only=True), inplace=True)
```

[224]:

```
Checking for Duplicate rows
print(data.duplicated().sum())

Ensuring correct data type
print(data.dtypes)
```

```
  Cell In[224], line 1
    Checking for Duplicate rows
             ^
SyntaxError: invalid syntax
```

[225]:
```
data['Gender'] = data['Gender'].map({'Female': 0, 'Male': 1})

# See the problematic rows
print(data[data['Gender'].isna()])

# Then drop them (recommended)
data = data.dropna(subset=['Gender'])

# Check again
print(data['Gender'].unique())
```

```
           Age  Cholesterol  Heart rate  Diabetes  Family History  Smoking  \
9377  0.516854      0.49978    0.065995       1.0             0.0      1.0
9378  0.516854      0.49978    0.065995       1.0             0.0      1.0
9379  0.516854      0.49978    0.065995       1.0             0.0      1.0
9380  0.382022      0.49978    0.105408       1.0             0.0      1.0
9381  0.370787      0.49978    0.042163       1.0             0.0      1.0
…        …          …         …       …             …      …
9646  0.404494      0.49978    0.091659       1.0             0.0      1.0
9647  0.235955      0.49978    0.049496       1.0             0.0      1.0
9648  0.348315      0.49978    0.088909       1.0             0.0      1.0
9649  0.370787      0.49978    0.067828       1.0             0.0      1.0
9650  0.797753      0.49978    0.084326       1.0             0.0      1.0

      Obesity  Alcohol Consumption  Exercise Hours Per Week  Diet  …  \
9377      1.0                  1.0                  0.50211     3  …
9378      1.0                  1.0                  0.50211     3  …
9379      1.0                  1.0                  0.50211     3  …
9380      1.0                  1.0                  0.50211     3  …
9381      1.0                  1.0                  0.50211     3  …
…       …                  …                      …   …  …
9646      1.0                  1.0                  0.50211     3  …
9647      1.0                  1.0                  0.50211     3  …
9648      1.0                  1.0                  0.50211     3  …
9649      1.0                  1.0                  0.50211     3  …
9650      1.0                  1.0                  0.50211     3  …

      Physical Activity Days Per Week  Sleep Hours Per Day  \
9377                              3.0             0.504621
9378                              3.0             0.504621
```

```
9379                              3.0             0.504621
9380                              3.0             0.504621
9381                              3.0             0.504621
…                               …                  …
9646                              3.0             0.504621
9647                              3.0             0.504621
9648                              3.0             0.504621
9649                              3.0             0.504621
9650                              3.0             0.504621

      Heart Attack Risk (Binary)  Blood sugar     CK-MB  Troponin  \
9377                         0.0     0.525692  0.004268  0.000388
9378                         0.0     0.112648  0.005002  0.000680
9379                         0.0     0.140316  0.024723  0.003690
9380                         0.0     0.128458  0.315234  0.000291
9381                         0.0     0.482213  1.000000  0.000583
…                          …          …       …       …
9646                         0.0     0.081028  0.002029  0.006894
9647                         0.0     0.084980  0.003500  0.000194
9648                         0.0     0.330040  0.009540  0.000194
9649                         0.0     0.197628  0.119858  0.025439
9650                         0.0     0.156126  0.006237  0.005923

      Heart Attack Risk (Text)  Gender  Systolic blood pressure  \
9377                         1     NaN                 0.554839
9378                         1     NaN                 0.458065
9379                         2     NaN                 0.554839
9380                         2     NaN                 0.212903
9381                         2     NaN                 0.445161
…                        …     …                      …
9646                         2     NaN                 1.000000
9647                         1     NaN                 0.445161
9648                         1     NaN                 0.225806
9649                         2     NaN                 0.258065
9650                         2     NaN                 0.322581

      Diastolic blood pressure
9377                  0.418605
9378                  0.313953
9379                  0.418605
9380                  0.209302
9381                  0.174419
…                        …
9646                  1.000000
9647                  0.186047
9648                  0.302326
9649                  0.453488
9650                  0.313953
```

```
[274 rows x 27 columns]
[1. 0.]
```

[226]:

```
X = data.drop(columns=["Heart Attack Risk (Text)"])  # or whatever your target␣
 ↪column is called
y = data["Heart Attack Risk (Text)"]

# Train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)


if 'Risk_Level' in data.columns:
    data = data.drop(columns=['Risk_Level'])

print(y_train.head())
print(y_train.dtype)
```

```
8093    2
7645    1
5639    0
4689    0
7603    0
Name: Heart Attack Risk (Text), dtype: int64
int64
```

[227]:

```
# Making a Random Forest Classifier

from sklearn.metrics import classification_report, accuracy_score

y_pred = rf_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.996268656716418

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00      1476
           1       0.96      0.99      0.98       144
           2       1.00      0.98      0.99       256

    accuracy                           1.00      1876
   macro avg       0.99      0.99      0.99      1876
weighted avg       1.00      1.00      1.00      1876
```

[229]:

```
# Using Bagging Classifier to decrease variance and bias

from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

# Base estimator (can be a decision tree or any classifier)
base_estimator = DecisionTreeClassifier(random_state=42)

# Initialize and train Bagging
bagging_model = BaggingClassifier(base_estimator=base_estimator,␣
  ↪n_estimators=100, random_state=42)
bagging_model.fit(X_train, y_train)

# Predict and evaluate
bagging_preds = bagging_model.predict(X_test)
print("Bagging Classifier Classification Report:")
print(classification_report(y_test, bagging_preds))
print("Accuracy:", accuracy_score(y_test, bagging_preds))
```

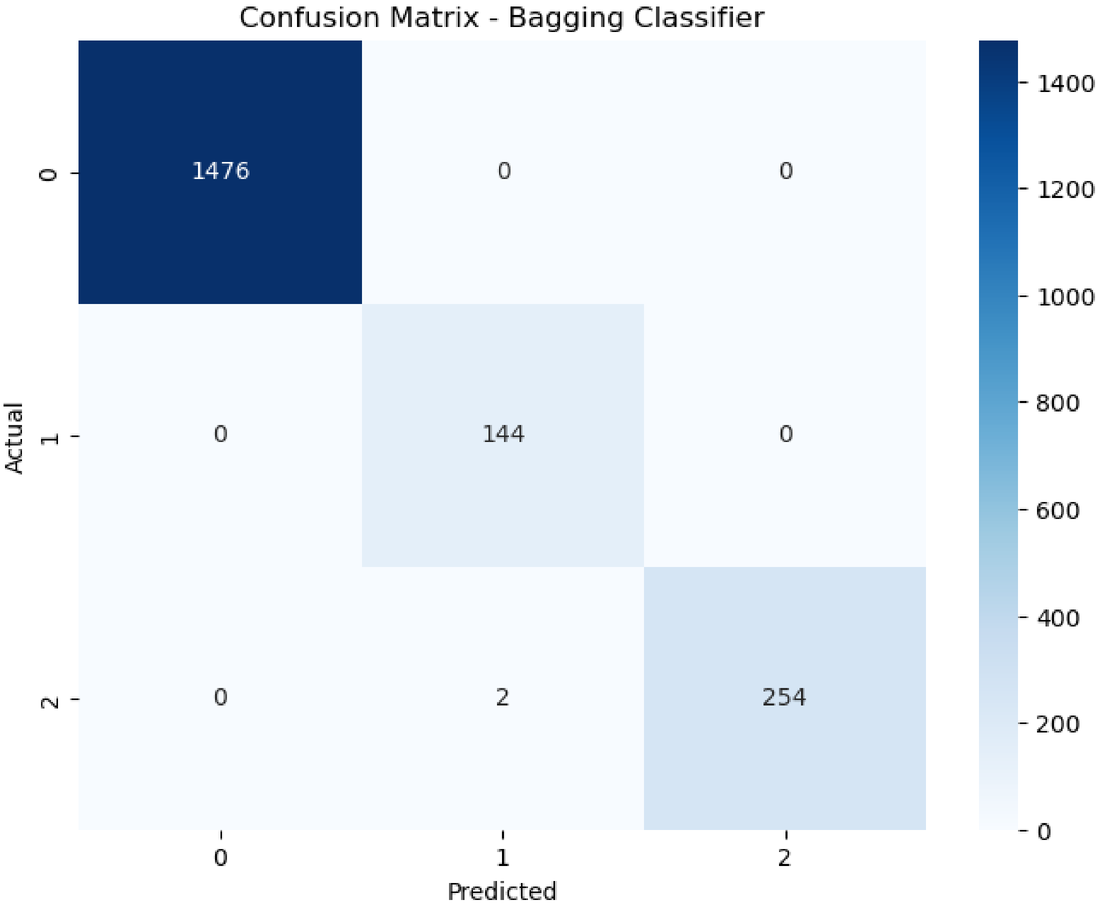
```
/Users/ajastarkey/anaconda3/lib/python3.10/site-
packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was
renamed to `estimator` in version 1.2 and will be removed in 1.4.
  warnings.warn(
```

```
Bagging Classifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1476
           1       0.99      1.00      0.99       144
           2       1.00      0.99      1.00       256

    accuracy                           1.00      1876
   macro avg       1.00      1.00      1.00      1876
weighted avg       1.00      1.00      1.00      1876

Accuracy: 0.9989339019189766
```

[231]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

```
# Replace with bagging_model or rf_model as needed
y_pred = bagging_model.predict(X_test)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
labels = bagging_model.classes_  # or rf_model.classes_

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Bagging Classifier")
plt.show()
```



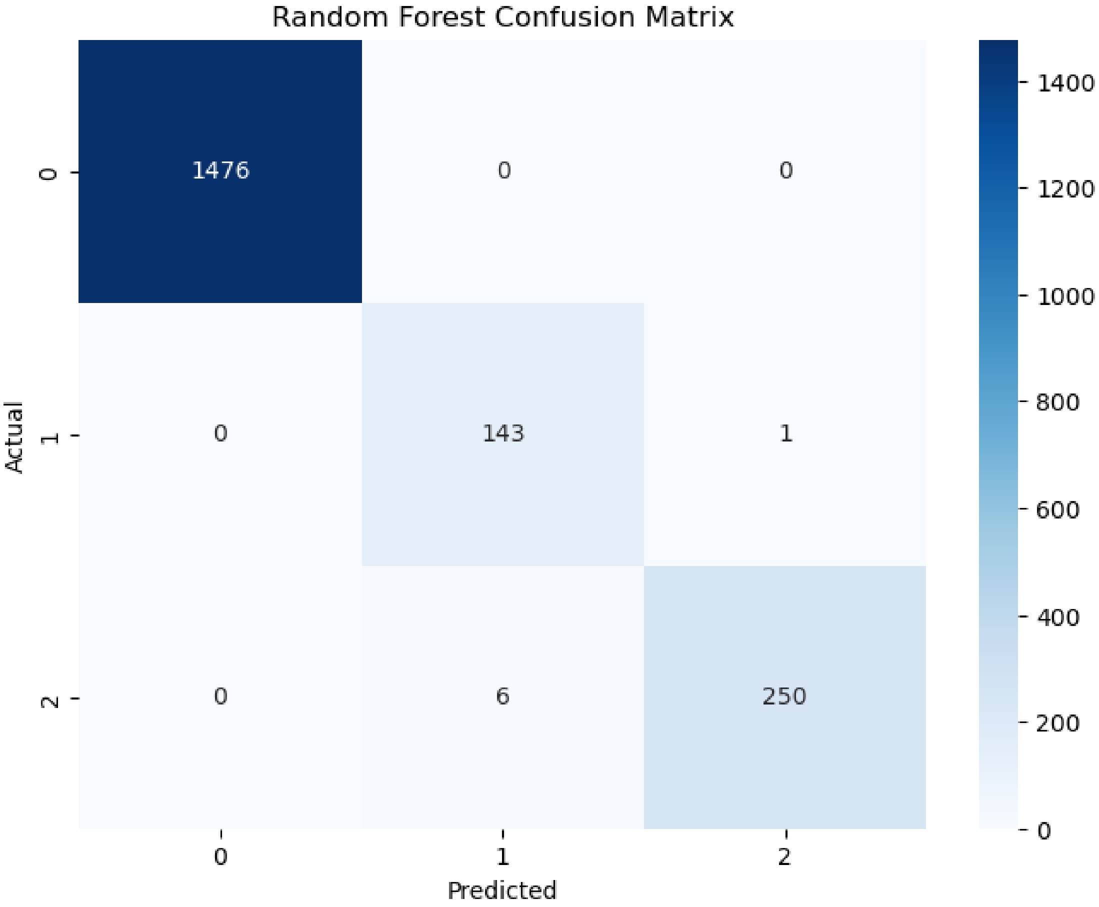
```
[ ]: # The confusion matrix above indicates that the model correctly classified all␣
      ↪of the
```

```
# high risk patients, however, accidently classified 2 of the moderate risk as␣
 ↪low risk.
# Only misclassifying two patients is a fairly good model, and with health care␣
 ↪data it is better to err on the side of more risk
# Which we can see that the model did
```

[232]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

# Train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict on the test set
rf_preds = rf_model.predict(X_test)

# Confusion matrix
rf_cm = confusion_matrix(y_test, rf_preds)
print("Random Forest Classification Report:")
print(classification_report(y_test, rf_preds))
print("Accuracy:", accuracy_score(y_test, rf_preds))

# Plot the confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(rf_cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=rf_model.classes_, yticklabels=rf_model.classes_)
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1476
           1       0.96      0.99      0.98       144
           2       1.00      0.98      0.99       256

    accuracy                           1.00      1876
   macro avg       0.99      0.99      0.99      1876
weighted avg       1.00      1.00      1.00      1876

Accuracy: 0.996268656716418
```



[ ]:
```
# The random forest classifier correctly labeled all of the high risk patients,␣
  ↪however, it mixed up 6 of the low risk patients and categorized them in␣
  ↪moderate risk.
# It also incorrectly labeled one patient as low risk when they should have␣
  ↪been moderate.
```

[ ]:
```
# Based off of the two models performance for classifying patients, I would␣
  ↪recommend using the bagging classifier over the random forest method.
```

[ ]:
```
# Github link: https://github.com/AjaStarkey/
  ↪Masters-in-Data-Science-Class-Projects
```