Coronary Heart Disease Prediction
Machine Learning Project

Pengxi Chen
March 14, 2024

[1]:
```
#(2)
import pandas as pd
dataset_path = "framingham.csv"
data = pd.read_csv(dataset_path)

#(3)
# Initial exploration
data_structure = {
    "Number of features": data.shape[1],
    "Number of observations": data.shape[0],
    "Variable types": data.dtypes
}

data_structure
```

[1]:
```
{'Number of features': 16,
'Number of observations': 4238,
'Variable types': male               int64
age                  int64
education          float64
currentSmoker        int64
cigsPerDay         float64
BPMeds             float64
prevalentStroke      int64
prevalentHyp         int64
diabetes             int64
totChol            float64
sysBP              float64
diaBP              float64
BMI                float64
heartRate          float64
glucose            float64
TenYearCHD           int64
dtype: object}
```

[2]:
```
#(4)
# Generate summary statistics for the dataset
```

```
summary_statistics = data.describe(include='all')
categorical_variables = ['male', 'currentSmoker', 'prevalentStroke',␣
↪'prevalentHyp', 'diabetes', 'TenYearCHD']
categorical_summary =  data[categorical_variables].describe()

summary_statistics, categorical_summary
```

[2]:
```
(                male          age    education  currentSmoker   cigsPerDay  \
count  4238.000000  4238.000000  4133.000000    4238.000000  4209.000000
mean      0.429212    49.584946     1.978950       0.494101     9.003089
std       0.495022     8.572160     1.019791       0.500024    11.920094
min       0.000000    32.000000     1.000000       0.000000     0.000000
25%       0.000000    42.000000     1.000000       0.000000     0.000000
50%       0.000000    49.000000     2.000000       0.000000     0.000000
75%       1.000000    56.000000     3.000000       1.000000    20.000000
max       1.000000    70.000000     4.000000       1.000000    70.000000

             BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
count  4185.000000      4238.000000   4238.000000  4238.000000  4188.000000
mean      0.029630         0.005899      0.310524     0.025720   236.721585
std       0.169584         0.076587      0.462763     0.158316    44.590334
min       0.000000         0.000000      0.000000     0.000000   107.000000
25%       0.000000         0.000000      0.000000     0.000000   206.000000
50%       0.000000         0.000000      0.000000     0.000000   234.000000
75%       0.000000         0.000000      1.000000     0.000000   263.000000
max       1.000000         1.000000      1.000000     1.000000   696.000000

              sysBP        diaBP          BMI    heartRate      glucose  \
count  4238.000000  4238.000000  4219.000000  4237.000000  3850.000000
mean    132.352407    82.893464    25.802008    75.878924    81.966753
std      22.038097    11.910850     4.080111    12.026596    23.959998
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.070000    68.000000    71.000000
50%     128.000000    82.000000    25.400000    75.000000    78.000000
75%     144.000000    89.875000    28.040000    83.000000    87.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

          TenYearCHD
count  4238.000000
mean      0.151958
std       0.359023
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000 ,
                male  currentSmoker  prevalentStroke  prevalentHyp  \
                diabetes
```

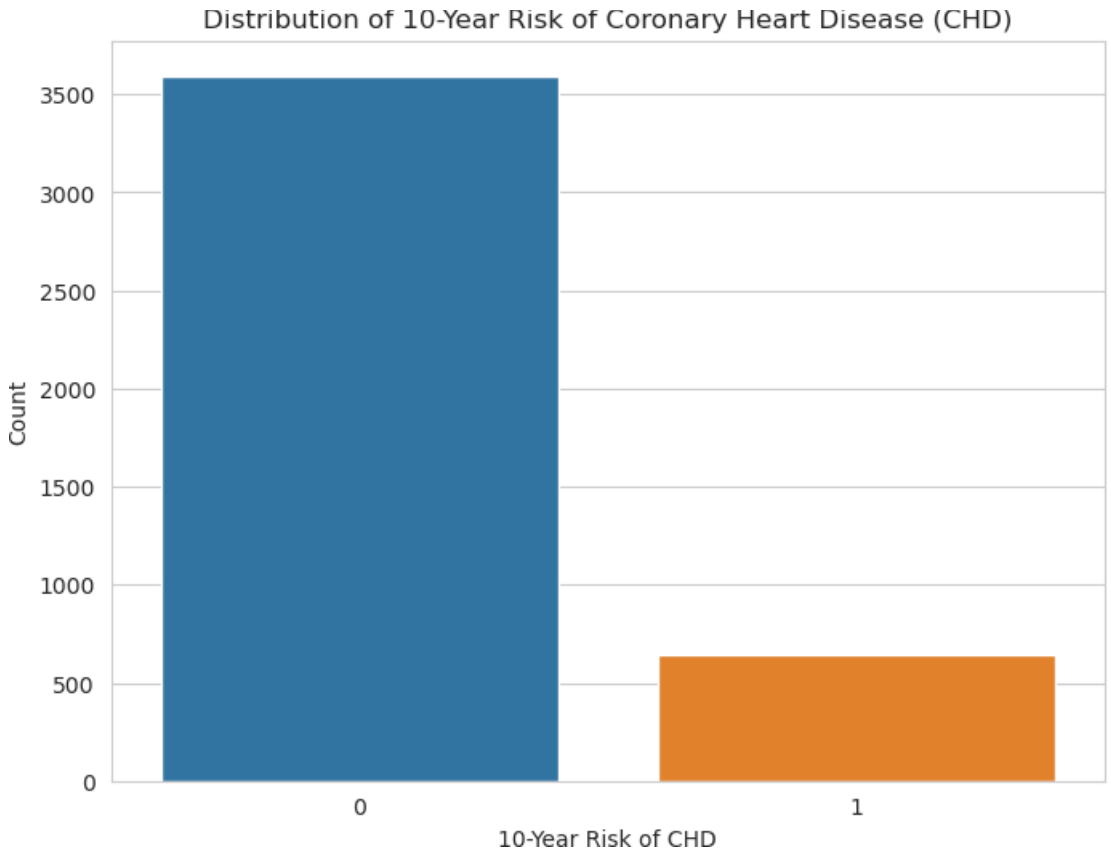
```
\
       male          age    education  currentSmoker     cigsPerDay  \
count  4238.000000  4238.000000  4238.000000  4238.000000  4238.000000
mean      0.429212    49.494101     0.005899     0.310524     0.025720
std       0.495022     0.500024     0.076587     0.462763     0.158316
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       1.000000     1.000000     0.000000     1.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

          TenYearCHD
count  4238.000000
mean      0.151958
std       0.359023
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000  )
```

[3]:
```python
#(5)
import matplotlib.pyplot as plt
import seaborn as sns

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Visualize the distribution of TenYearCHD
plt.figure(figsize=(8, 6))
sns.countplot(data=data, x="TenYearCHD")
plt.title("Distribution of 10-Year Risk of Coronary Heart Disease (CHD)")
plt.xlabel("10-Year Risk of CHD")
plt.ylabel("Count")
plt.show()
```



[11]:
```
# 6
# Recognize 'NA' as missing values
data = pd.read_csv(dataset_path, na_values="NA")
missing_values_corrected = data.isnull().sum()

# Drop observations with missing values
data_cleaned_corrected = data.dropna()
missing_values_corrected, data_cleaned_corrected.shape
```

[11]:
```
(male                0
 age                 0
 education         105
 currentSmoker       0
 cigsPerDay         29
 BPMeds             53
 prevalentStroke     0
 prevalentHyp        0
 diabetes            0
 totChol            50
 sysBP               0
```

```
diaBP              0
BMI               19
heartRate          1
glucose          388
TenYearCHD         0
dtype: int64,
(3656, 16))
```

[19]:
```
#(7)
from sklearn.preprocessing import StandardScaler

# Identify numerical columns
numerical_cols = data_cleaned_corrected.columns.drop('TenYearCHD')

# Standardize the numerical columns
scaler = StandardScaler()
data_cleaned_corrected[numerical_cols] = scaler.
↪fit_transform(data_cleaned_corrected[numerical_cols])
```

[19]:
```
       male       age  education  currentSmoker  cigsPerDay     BPMeds  \
0  1.119825 -1.233351   1.975752      -0.978352   -0.757068 -0.176951
1 -0.892997 -0.415591   0.019795      -0.978352   -0.757068 -0.176951
2  1.119825 -0.181945  -0.958183       1.022127    0.921174 -0.176951
3 -0.892997  1.336754   0.997773       1.022127    1.760294 -0.176951
4 -0.892997 -0.415591   0.997773       1.022127    1.172910 -0.176951

   prevalentStroke  prevalentHyp  diabetes   totChol     sysBP     diaBP  \
0        -0.076008     -0.672698 -0.166831 -0.949714 -1.193695 -1.078415
1        -0.076008     -0.672698 -0.166831  0.297729 -0.514637 -0.159695
2        -0.076008     -0.672698 -0.166831  0.184325 -0.220378 -0.243215
3        -0.076008      1.486551 -0.166831 -0.269291  0.798209  1.009584
4        -0.076008     -0.672698 -0.166831  1.091556 -0.107202  0.090864

        BMI  heartRate   glucose  TenYearCHD
0  0.291688   0.356340 -0.203127           0
1  0.724614   1.608289 -0.244956           0
2 -0.109261  -0.060977 -0.495930           0
3  0.687717  -0.895610  0.884427           1
4 -0.660258   0.773656  0.131505           0
```

[22]:
```
#(8)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Assign predictors to X and response variable to y
X =data_cleaned_corrected.drop('TenYearCHD', axis=1)
```

```python
y =data_cleaned_corrected['TenYearCHD']

# Split the dataset into training and testing sets using stratified sampling
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪stratify=y, random_state=42)



#(9)
# Create an instance of the logistic regression model
logreg = LogisticRegression(max_iter=1000, random_state=42)


#(10)
# Train the model on the training set
logreg.fit(X_train, y_train)
# Make predictions on the test set
y_pred = logreg.predict(X_test)



#(11)
# Calculate the accuracy of the model on the test set
accuracy = accuracy_score(y_test, y_pred)

y_pred,accuracy
```

[22]:
```
(array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
```

```
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
 0.862144420131291)
```

[23]:
```
#(12)
from sklearn.metrics import confusion_matrix

# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
tp = conf_matrix[1, 1]
tn = conf_matrix[0, 0]
fp = conf_matrix[0, 1]
fn = conf_matrix[1, 0]



#(13)
# Calculate sensitivity and specificity
sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)

conf_matrix, sensitivity, specificity
```

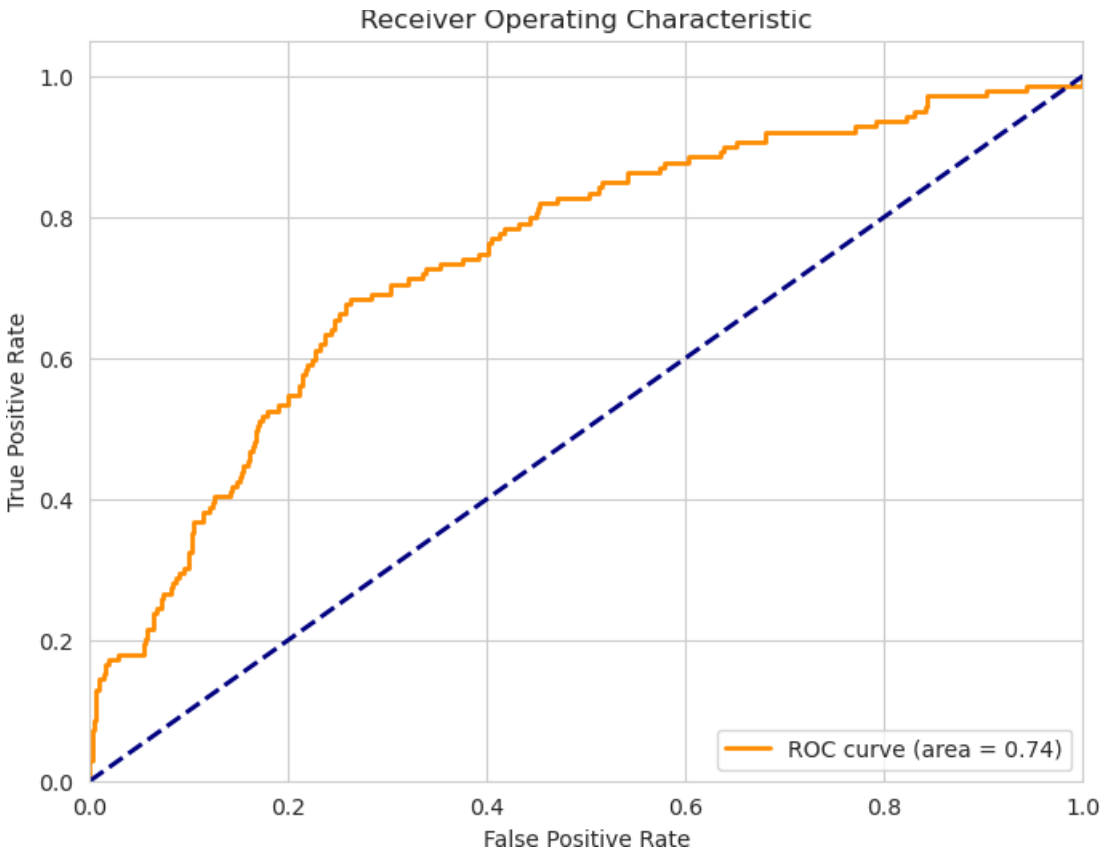
[23]:
```
(array([[770,   5],
        [121,  18]]),
 0.12949640287769784,
 0.9935483870967742)
```

[24]:
```
#(14)
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Compute ROC curve and ROC area
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %␣
↪roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# AUC score
roc_auc
```



[24]: 0.7440241355302853

[25]:
```
#(15)
selected_predictors = ['male', 'age', 'cigsPerDay', 'diabetes', 'sysBP',␣
↪'heartRate']
X_selected = data_cleaned_corrected[selected_predictors]
X_selected.head(), X_selected.shape
```

[25]:
```
(      male       age  cigsPerDay   diabetes      sysBP  heartRate
 0  1.119825 -1.233351   -0.757068 -0.166831 -1.193695   0.356340
 1 -0.892997 -0.415591   -0.757068 -0.166831 -0.514637   1.608289
 2  1.119825 -0.181945    0.921174 -0.166831 -0.220378  -0.060977
 3 -0.892997  1.336754    1.760294 -0.166831  0.798209  -0.895610
 4 -0.892997 -0.415591    1.172910 -0.166831 -0.107202   0.773656,
 (3656, 6))
```

[26]:
```
#(16)
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
```

```
# Split the design matrix and response variables into training and testing sets
X_train_sel, X_test_sel, y_train_sel, y_test_sel =
    train_test_split( X_selected, y, test_size=0.25, stratify=y,
    random_state=42)

X_train_sel_const = sm.add_constant(X_train_sel)

logit_model = sm.Logit(y_train_sel, X_train_sel_const).fit()

# Display the summaries
```

```
Optimization terminated successfully.
         Current function value: 0.381593
         Iterations 6
```

[26]:

Dep. Variable:	TenYearCHD	**No. Observations:**	2742
Model:	Logit	**Df Residuals:**	2735
Method:	MLE	**Df Model:**	6
Date:	Thu, 14 Mar 2024	**Pseudo R-squ.:**	0.1062
Time:	05:49:48	**Log-Likelihood:**	-1046.3
converged:	True	**LL-Null:**	-1170.6
Covariance Type:	nonrobust	**LLR p-value:**	8.191e-51

	coef	std err	z	P> |z|	[0.025	0.975]
const	-1.9716	0.065	-30.540	0.000	-2.098	-1.845
male	0.2128	0.060	3.521	0.000	0.094	0.331
age	0.5742	0.062	9.189	0.000	0.452	0.697
cigsPerDay	0.2158	0.058	3.704	0.000	0.102	0.330
diabetes	0.1039	0.045	2.310	0.021	0.016	0.192
sysBP	0.4253	0.054	7.826	0.000	0.319	0.532
heartRate	-0.0284	0.057	-0.499	0.618	-0.140	0.083