Christian Campbell

Final Project

In [9]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
```

In [5]:
```
path = r"C:\Users\chris\Documents\Bellevue University\7 - Predictive Analytics\Final Project\heart_statlc
heart_disease = pd.read_csv(path)
heart_disease.head()
```

Out[5]:

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target
0	40	1	2	140	289	0	0	172	0	0.0	1	0
1	49	0	3	160	180	0	0	156	0	1.0	2	1
2	37	1	2	130	283	0	1	98	0	0.0	1	0
3	48	0	4	138	214	0	0	108	1	1.5	2	1
4	54	1	3	150	195	0	0	122	0	0.0	1	0

Logistic regression model

In [6]:
```
# Preprocess the data

# Separates features and target
X = heart_disease.drop(columns='target')
y = heart_disease['target']

# Standardizes the numeric features
numeric_features = ['age', 'resting bp s', 'cholesterol', 'max heart rate', 'oldpeak']
scaler = StandardScaler()
X[numeric_features] = scaler.fit_transform(X[numeric_features])

# Splits the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Builds and trains the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

Out[6]:
```
LogisticRegression(max_iter=1000)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [7]:
```
# The code chunks in this cell evaluates the model

# Predictions
y_pred = model.predict(X_test)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Precision
precision = precision_score(y_test, y_pred)
print("Precision:", precision)

# F1 score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)

# Cross-validation
cv_scores = cross_val_score(model, X, y, cv=10, scoring='accuracy')
print("Cross-validation Accuracy Scores:", cv_scores)
print("Mean Cross-validation Accuracy:", cv_scores.mean())
```

```
Confusion Matrix:
[[ 90  17]
 [ 16 115]]
Accuracy: 0.8613445378151261
Precision: 0.8712121212121212
F1 Score: 0.8745247148288973
Cross-validation Accuracy Scores: [0.80672269 0.89915966 0.86554622 0.76470588 0.84033613 0.86554622
 0.78151261 0.77310924 0.81512605 0.78151261]
Mean Cross-validation Accuracy: 0.819327731092437
```

Random Forest Model

In [10]:
```
# Preprocessing the data

# Separates features and target
X = heart_disease.drop("target", axis=1)
y = heart_disease["target"]

# Splits the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initializes the random forest classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Trains the model
rf_model.fit(X_train, y_train)
```

Out[10]:
```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [11]:

```
# The code chunks in this cell evaluates the model

# Makes predictions on the test set
y_pred = rf_model.predict(X_test)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Precision
precision = precision_score(y_test, y_pred)
print("Precision:", precision)

# F1 score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)

# Perform cross-validation and get the scores
cv_accuracy = cross_val_score(rf_model, X, y, cv=5, scoring='accuracy')
cv_precision = cross_val_score(rf_model, X, y, cv=5, scoring='precision')
cv_f1 = cross_val_score(rf_model, X, y, cv=5, scoring='f1')

print("Cross-validated Accuracy:", np.mean(cv_accuracy))
print("Cross-validated Precision:", np.mean(cv_precision))
print("Cross-validated F1 Score:", np.mean(cv_f1))
```

```
Confusion Matrix:
 [[ 98   9]
 [  4 127]]
Accuracy: 0.9453781512605042
Precision: 0.9338235294117647
F1 Score: 0.951310861423221
Cross-validated Accuracy: 0.9285714285714286
Cross-validated Precision: 0.9312755672648905
Cross-validated F1 Score: 0.9332100467879512
```

Random Rorest Feature Importance

In [14]:
```
# Fits Random Forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Gets the feature importances
feature_importance_rf = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf.feature_importances_
}).sort_values(by='Importance', ascending=False)

print(feature_importance_rf)
```

```
                Feature  Importance
10             ST slope    0.194539
2       chest pain type    0.134759
9               oldpeak    0.116515
7        max heart rate    0.116115
4           cholesterol    0.111434
0                   age    0.094971
3          resting bp s    0.078512
8       exercise angina    0.068288
1                   sex    0.038240
6           resting ecg    0.027517
5   fasting blood sugar    0.019110
```

In [ ]:
```

```