Experiment 11: Heart Disease Prediction using different Machine Learning Models

In [ ]:
```
#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.impute import SimpleImputer
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import (
    confusion_matrix, classification_report, roc_curve, roc_auc_score
)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [74]:
```
# Load the dataset
df = pd.read_csv('heart_disease.csv')
df
```

Out[74]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slo
0	52	1	0	125	212	0	1	168	0	1.0	
1	53	1	0	140	203	1	0	155	1	3.1	
2	70	1	0	145	174	0	1	125	1	2.6	
3	61	1	0	148	203	0	1	161	0	0.0	
4	62	0	0	138	294	1	1	106	0	1.9	
...	...	...	...	...	...	...	...	...	...	...	
1020	59	1	1	140	221	0	1	164	1	0.0	
1021	60	1	0	125	258	0	0	141	1	2.8	
1022	47	1	0	110	275	0	0	118	1	1.0	
1023	50	0	0	110	254	0	0	159	0	0.0	
1024	54	1	0	120	188	0	1	113	0	1.4	

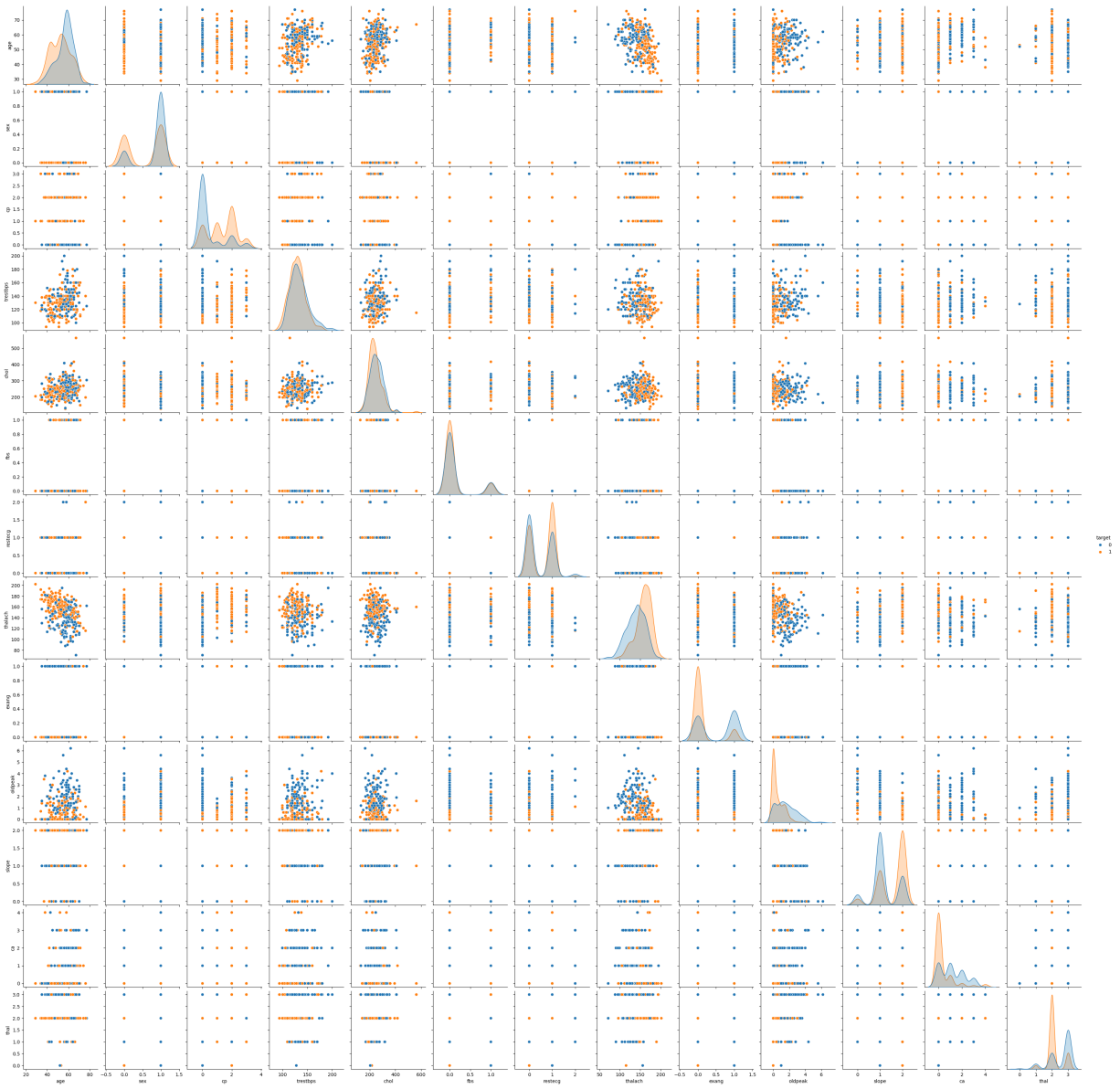
1025 rows × 14 columns

In [75]:
```
df.describe()
```

Out[75]:

	age	sex	cp	trestbps	chol	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.0
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.1
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.3
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.0
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.0
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.0
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.0
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.0

In [76]:
```
sns.pairplot(df, hue='target')
plt.show()
```



In [77]:
```
# Split the data into features (X) and target variable (y)
X = df.drop('target', axis=1)
y = df['target']
```

In [78]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar
```

In [79]:
```
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [105…
```
models = {
    'Logistic Regression': LogisticRegression(solver='liblinear'),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC(probability=True),
    'KNN': KNeighborsClassifier(),
    'Naive Bayes': GaussianNB(),
```

```
}


for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {accuracy:.2f}")
    print(classification_report(y_test, y_pred))
```

```
Training Logistic Regression...
Logistic Regression Accuracy: 0.81
              precision    recall  f1-score   support

           0       0.86      0.75      0.80       159
           1       0.76      0.87      0.81       149

    accuracy                           0.81       308
   macro avg       0.81      0.81      0.80       308
weighted avg       0.81      0.81      0.80       308

Training Decision Tree...
Decision Tree Accuracy: 0.97
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       159
           1       1.00      0.94      0.97       149

    accuracy                           0.97       308
   macro avg       0.97      0.97      0.97       308
weighted avg       0.97      0.97      0.97       308

Training Random Forest...
Random Forest Accuracy: 0.98
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       159
           1       1.00      0.96      0.98       149

    accuracy                           0.98       308
   macro avg       0.98      0.98      0.98       308
weighted avg       0.98      0.98      0.98       308

Training SVM...
SVM Accuracy: 0.89
              precision    recall  f1-score   support

           0       0.92      0.86      0.89       159
           1       0.86      0.92      0.89       149

    accuracy                           0.89       308
   macro avg       0.89      0.89      0.89       308
weighted avg       0.89      0.89      0.89       308

Training KNN...
KNN Accuracy: 0.84
              precision    recall  f1-score   support

           0       0.87      0.82      0.84       159
           1       0.82      0.87      0.84       149

    accuracy                           0.84       308
   macro avg       0.84      0.84      0.84       308
weighted avg       0.84      0.84      0.84       308

Training Naive Bayes...
```

```
Naive Bayes Accuracy: 0.81
              precision    recall  f1-score   support

           0       0.88      0.74      0.81       159
           1       0.76      0.89      0.82       149

    accuracy                           0.81       308
   macro avg       0.82      0.82      0.81       308
weighted avg       0.82      0.81      0.81       308
```

In [85]:

```python
for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {accuracy:.2f}")
    print(classification_report(y_test, y_pred))
    print("\n")
```

```
Training Logistic Regression...
Logistic Regression Accuracy: 0.81
              precision    recall  f1-score   support

           0       0.86      0.75      0.80       159
           1       0.76      0.87      0.81       149

    accuracy                           0.81       308
   macro avg       0.81      0.81      0.80       308
weighted avg       0.81      0.81      0.80       308



Training Decision Tree...
Decision Tree Accuracy: 0.97
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       159
           1       1.00      0.94      0.97       149

    accuracy                           0.97       308
   macro avg       0.97      0.97      0.97       308
weighted avg       0.97      0.97      0.97       308



Training Random Forest...
Random Forest Accuracy: 1.00
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       159
           1       1.00      1.00      1.00       149

    accuracy                           1.00       308
   macro avg       1.00      1.00      1.00       308
weighted avg       1.00      1.00      1.00       308



Training SVM...
SVM Accuracy: 0.89
              precision    recall  f1-score   support

           0       0.92      0.86      0.89       159
           1       0.86      0.92      0.89       149

    accuracy                           0.89       308
   macro avg       0.89      0.89      0.89       308
weighted avg       0.89      0.89      0.89       308



Training KNN...
KNN Accuracy: 0.84
              precision    recall  f1-score   support
```

```
              0       0.87      0.82      0.84       159
              1       0.82      0.87      0.84       149

       accuracy                           0.84       308
      macro avg       0.84      0.84      0.84       308
   weighted avg       0.84      0.84      0.84       308



Training Naive Bayes...
Naive Bayes Accuracy: 0.81
                precision    recall  f1-score   support

              0       0.88      0.74      0.81       159
              1       0.76      0.89      0.82       149

       accuracy                           0.81       308
      macro avg       0.82      0.82      0.81       308
   weighted avg       0.82      0.81      0.81       308
```

In [121…

```python
# Definitions for the accuracy variables
LRA = 0.81  # Logistic Regression accuracy
KNNA = 0.84  # K-Nearest Neighbour accuracy
SVMA = 0.89  # Support Vector Machine accuracy
GNBA = 0.81  # Gaussian Naive Bayes accuracy
DTA = 0.97  # Decision Tree accuracy
RFA = 1.00  # Random Forest accuracy

# Create the DataFrame with defined accuracy values
compare = pd.DataFrame({
    'Model': ['Logistic Regression', 'K-Nearest Neighbour', 'Support Vector
              'Gaussian Naive Bayes', 'Decision Tree', 'Random Forest'],
    'Accuracy': [LRA * 100, KNNA * 100, SVMA * 100, GNBA * 100, DTA * 100, R
})

# --- Create Accuracy Comparison Table ---
styled_compare = compare.style.set_properties(**{'font-family': 'Segoe UI'})
styled_compare
```

Out[121…

Model	Accuracy
Logistic Regression	81.000000
K-Nearest Neighbour	84.000000
Support Vector Machine	89.000000
Gaussian Naive Bayes	81.000000
Decision Tree	97.000000
Random Forest	100.000000

In [111…

```python
# Dictionary for model abbreviations
model_abbreviations = {
    'Decision Tree': 'DT',
    'Logistic Regression': 'LR',
    'Random Forest': 'RF',
    'SVM': 'SVM',
    'KNN': 'KNN',
    'Naive Bayes': 'NB'
}

# Plot ROC Curve
plt.figure(figsize=(10, 6))

for model_name, model in models.items():
    # Fit the model
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]  # Probability estimates

    # Metrics
    accuracy = accuracy_score(y_test, y_pred)  # Calculate accuracy
    cm = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    f1 = report['1']['f1-score']
    precision = report['1']['precision']
    sensitivity = report['1']['recall']  # also known as True Positive Rate
    specificity = cm[0][0] / (cm[0][0] + cm[0][1])  # True Negative Rate
    roc_auc = roc_auc_score(y_test, y_prob)

    # Store results with abbreviation
    short_name = model_abbreviations.get(model_name, model_name)  # Get abbr
    results[short_name] = {
        'Accuracy': round(accuracy, 3),
        'Precision': round(precision, 3),
        'F1 Score': round(f1, 3),
        'Sensitivity': round(sensitivity, 3),
        'Specificity': round(specificity, 3),
        'ROC AUC': round(roc_auc, 3)
    }

    # ROC Curve
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.plot(fpr, tpr, label=f'{short_name} (AUC = {roc_auc:.3f})')

# Plot ROC Curve
plt.plot([0, 1], [0, 1], 'k--')
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()
plt.show()
```