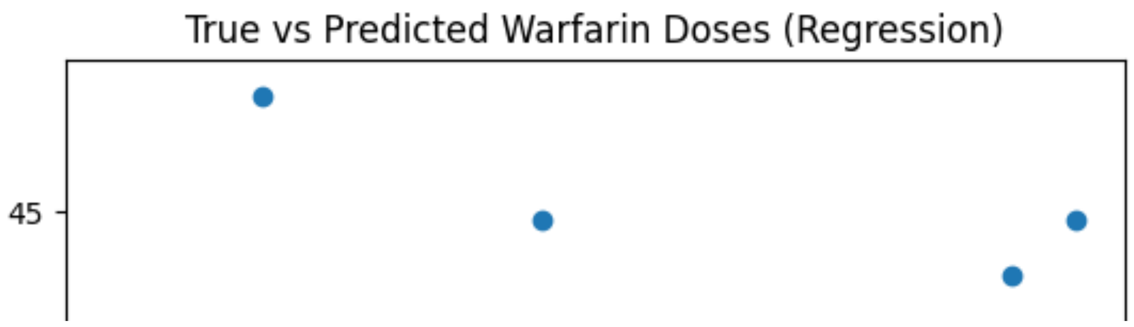
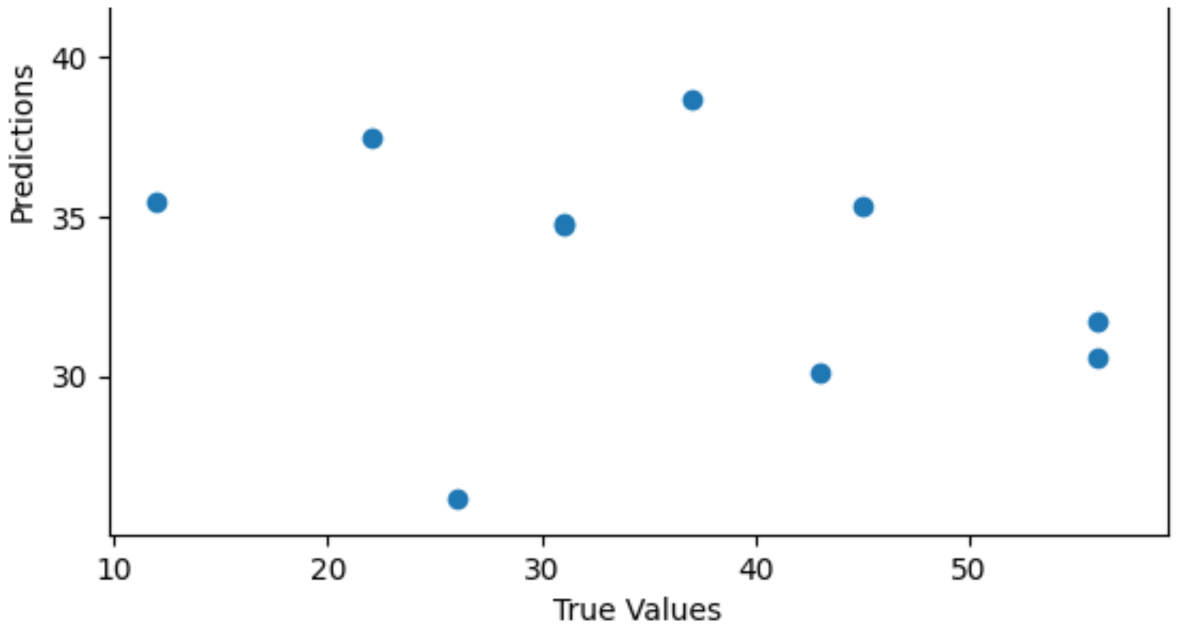
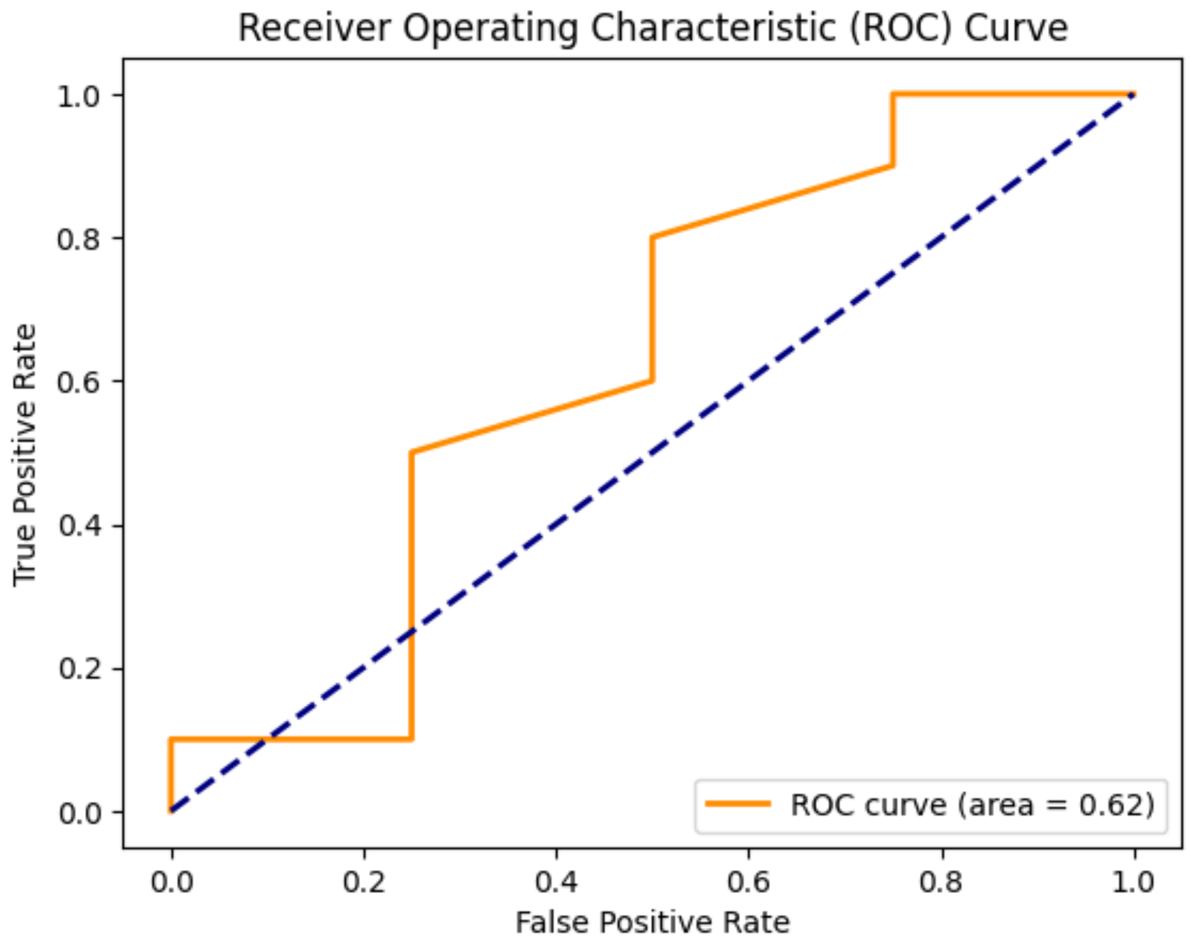
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, accu
from sklearn.metrics import roc_curve, auc
import xgboost as xgb
import gradio as gr
import joblib

# Load Dataset
df = pd.read_excel('warfarin_dataset.xlsx')
print("Columns in the dataset:", df.columns)
features = ['Gender', 'Age', 'Height (cm)', 'Weight (kg)', 'Diabetes', 'Simvastatir
            'INR on Reported Therapeutic Dose of Warfarin', 'Cyp2C9 genotypes', 'VK
target = 'Therapeutic Dose of Warfarin'

# Data Preprocessing
imputer = SimpleImputer(strategy='most_frequent')
df[features] = imputer.fit_transform(df[features])

# Feature engineering
X = df[features]
y = df[target]
X = pd.get_dummies(X, drop_first=True)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Regression model
regressor_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
])
regressor_pipeline.fit(X_train, y_train)
y_pred_regressor = regressor_pipeline.predict(X_test)

print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_regressor))
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_regressor))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_regressor))
print("R2 Score:", r2_score(y_test, y_pred_regressor))

# Classification setup
y_class = (y > 30).astype(int)  # binary target
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(X, y_cl

classifier_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])
classifier_pipeline.fit(X_train_class, y_train_class)
y_pred_class = classifier_pipeline.predict(X_test_class)

print("Accuracy:", accuracy_score(y_test_class, y_pred_class))
print("Classification Report:\n", classification_report(y_test_class, y_pred_class)

# Hyperparameter tuning for regression
param_grid = {
    'regressor__n_estimators': [50, 100, 200],
    'regressor__max_depth': [10, 20, 30],
}
grid_search = GridSearchCV(regressor_pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Best parameters:", grid_search.best_params_)

# Regression plot
plt.scatter(y_test, y_pred_regressor)
plt.xlabel("True Values")
plt.ylabel("Predictions")
plt.title("True vs Predicted Warfarin Doses (Regression)")
plt.show()

# ROC curve for classification
fpr, tpr, _ = roc_curve(y_test_class, classifier_pipeline.predict_proba(X_test_clas
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

```
# Save correct model and features
joblib.dump(classifier_pipeline, 'warfarin_dose_predictor_model.pkl')
joblib.dump(X.columns.tolist(), 'warfarin_feature_columns.pkl')

# Load model & features for Gradio
classifier_pipeline = joblib.load('warfarin_dose_predictor_model.pkl')
feature_names = joblib.load('warfarin_feature_columns.pkl')

# UI setup
gender_options = [('Male', 0), ('Female', 1)]
race_options = [('Asian', 1), ('Black', 2), ('White', 3)]
age_options = [('10-19', 0), ('20-29', 1), ('30-39', 2), ('40-49', 3), ('50-59', 4)

def predict_dose(gender, race, age_group, height, weight, diabetes, simvastatin, am
    input_data = {
        'Gender': gender,
        'Race': race,
        'Age': age_group,
        'Height (cm)': height,
        'Weight (kg)': weight,
        'Diabetes': diabetes,
        'Simvastatin': simvastatin,
        'Amiodarone': amiodarone
    }
    df_input = pd.DataFrame([input_data])
    df_input = pd.get_dummies(df_input)

    # Align columns
    for col in feature_names:
        if col not in df_input.columns:
            df_input[col] = 0
    df_input = df_input[feature_names]

    prob = classifier_pipeline.predict_proba(df_input)[0]
    prediction = classifier_pipeline.predict(df_input)[0]

    labels = ['Low Therapeutic Dose of Warfarin Required', 'High Therapeutic Dose o
    return labels[prediction], {labels[0]: prob[0], labels[1]: prob[1]}

# Gradio interface
interface = gr.Interface(
    fn=predict_dose,
    inputs=[
        gr.Dropdown(gender_options, label="Gender"),
        gr.Dropdown(race_options, label="Race"),
        gr.Dropdown(age_options, label="Age Group"),
        gr.Number(label="Height (cm)"),
        gr.Number(label="Weight (kg)"),
        gr.Radio([0, 1], label="Diabetes"),
        gr.Radio([0, 1], label="Simvastatin"),
        gr.Radio([0, 1], label="Amiodarone"),
        gr.Dropdown(['Random Forest', 'Logistic Regression', 'Decision Tree',
                     'Linear Discriminant Analysis', 'MLP', 'SVM', 'KNN'], label="M
    ],
    outputs=[
        gr.Text(label="Predicted Class"),
        gr.Label(label="Predicted Probability")
    ],
    title="Warfarin Dosage Predictor"
)

interface.launch()
```

```
Columns in the dataset: Index(['Gender', 'Age', 'Height (cm)', 'Weight (kg)',
       'Simvastatin', 'Amiodarone', 'Target INR',
       'INR on Reported Therapeutic Dose of Warfarin', 'Cyp2C9 genotypes',
       'VKORC1 genotype', 'Therapeutic Dose of Warfarin'],
      dtype='object')
Mean Absolute Error: 13.288571428571426
Mean Squared Error: 258.9795571428571
R2 Score: -0.24923075332857514
Accuracy: 0.7142857142857143
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         4
           1       0.71      1.00      0.83        10

    accuracy                           0.71        14
   macro avg       0.36      0.50      0.42        14
weighted avg       0.51      0.71      0.60        14

Best parameters: {'regressor__max_depth': 10, 'regressor__n_estimators': 100}
```







```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the G

Colab notebook detected. To show errors in colab notebook, set debug=True in l
* Running on public URL: https://ce05dae0e2ce8b007c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades
```

0

Diabetes

0 [✓ 1]

Simvastatin

[✓ 0] 1

Amiodarone

0 [✓ 1]

Model

Random Forest

Clear Submit

Predicted Class