```
# Завантаження бібліотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import xgboost as xgb

# Підготовка даних
# Завантажимо приклад датасету (можна замінити на власний)
data = load_boston()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Розділення на тренувальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-ec01dcf68288> in <cell line: 0>()
     11 # Підготовка даних
     12 # Завантажимо приклад датасету (можна замінити на власний)
---> 13 data = load_boston()
     14 X = pd.DataFrame(data.data, columns=data.feature_names)
     15 y = pd.Series(data.target)

NameError: name 'load_boston' is not defined
```

```
# 1. Реалізація дерева рішень
dt_model = DecisionTreeRegressor(max_depth=3, random_state=42)
dt_model.fit(X_train, y_train)

# Оцінка моделі дерева рішень
y_pred_dt = dt_model.predict(X_test)
mse_dt = mean_squared_error(y_test, y_pred_dt)
mae_dt = mean_absolute_error(y_test, y_pred_dt)
r2_dt = r2_score(y_test, y_pred_dt)

# Виведення результатів для дерева рішень
print(f"Decision Tree MSE: {mse_dt}")
print(f"Decision Tree MAE: {mae_dt}")
print(f"Decision Tree R²: {r2_dt}")

# Візуалізація дерева рішень
plt.figure(figsize=(12, 8))
plot_tree(dt_model, filled=True, feature_names=X.columns, fontsize=10)
plt.show()
```

```
# 2. Реалізація Random Forest
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Оцінка моделі Random Forest
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# Виведення результатів для Random Forest
print(f"Random Forest MSE: {mse_rf}")
print(f"Random Forest MAE: {mae_rf}")
print(f"Random Forest R²: {r2_rf}")
```

```
# 3. Реалізація XGBoost
xg_model = xgb.XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=42)
xg_model.fit(X_train, y_train)

# Оцінка моделі XGBoost
y_pred_xg = xg_model.predict(X_test)
mse_xg = mean_squared_error(y_test, y_pred_xg)
mae_xg = mean_absolute_error(y_test, y_pred_xg)
r2_xg = r2_score(y_test, y_pred_xg)

# Виведення результатів для XGBoost
```

```
print(f"XGBoost MSE: {mse_xg}")
print(f"XGBoost MAE: {mae_xg}")
print(f"XGBoost R²: {r2_xg}")



# Порівняння важливості ознак
feature_importance_dt = dt_model.feature_importances_
feature_importance_rf = rf_model.feature_importances_
feature_importance_xg = xg_model.feature_importances_

# Візуалізація важливості ознак
features = X.columns
plt.figure(figsize=(12, 8))
plt.barh(features, feature_importance_dt, alpha=0.6, label="Decision Tree")
plt.barh(features, feature_importance_rf, alpha=0.6, label="Random Forest")
plt.barh(features, feature_importance_xg, alpha=0.6, label="XGBoost")
plt.xlabel("Feature Importance")
plt.title("Feature Importance Comparison")
plt.legend()
plt.show()

# Визначення найважливіших факторів (за важливістю ознак)
print("\nImportant features for Decision Tree:")
print(pd.DataFrame(feature_importance_dt, index=features, columns=["Importance"]).sort_values(by="Importance", ascending=False))

print("\nImportant features for Random Forest:")
print(pd.DataFrame(feature_importance_rf, index=features, columns=["Importance"]).sort_values(by="Importance", ascending=False))

print("\nImportant features for XGBoost:")
print(pd.DataFrame(feature_importance_xg, index=features, columns=["Importance"]).sort_values(by="Importance", ascending=False))
```