Analyzing House Price Data Using Random Forests and Decision Trees - Project by Steve Matindi (Stevemats)

March 30, 2024

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Load the dataset
data = pd.read_csv("./Dataset/Real estate.csv")
print(data.columns)
```

```
Index(['No', 'X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'Y house price of unit area'],
      dtype='object')
```

[3]:
```
# Splitting features and target variable
X = data.drop(columns=['No', 'Y house price of unit area'])
Y = data['Y house price of unit area']
```

[4]:
```
# Split 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)
```

[5]:
```
# Train Random Forest model
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, Y_train)
```

[5]:
```
RandomForestRegressor(random_state=42)
```

[6]:
```
# Train Decision Tree model
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, Y_train)
```

[6]:
```
DecisionTreeRegressor(random_state=42)
```

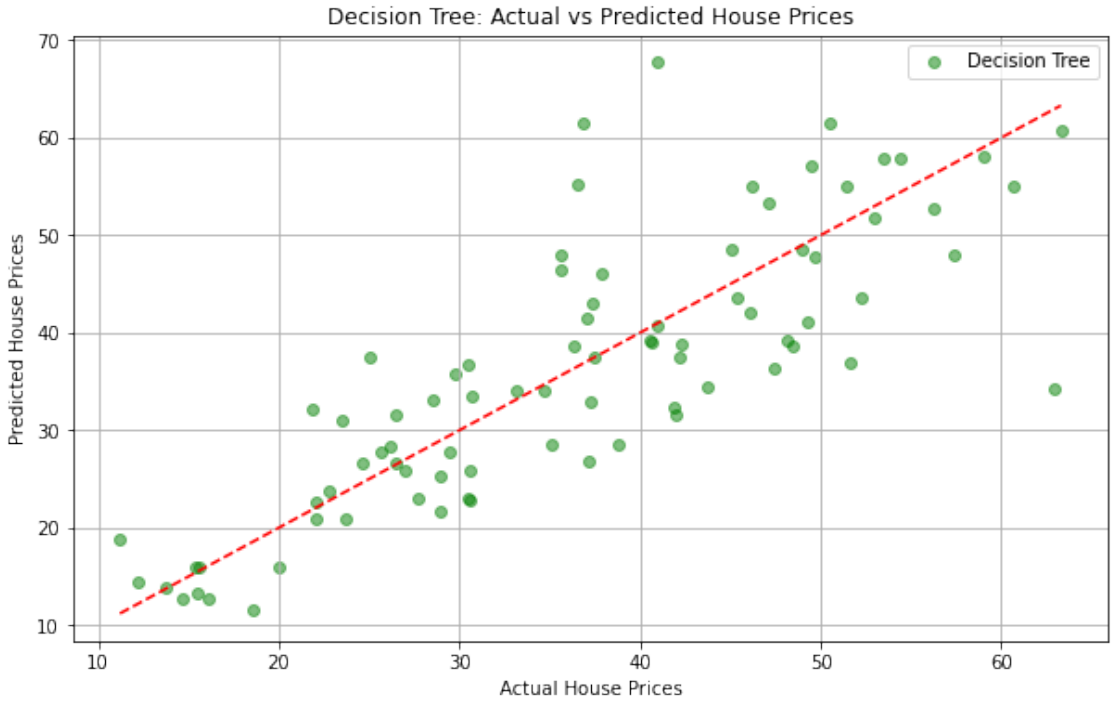
[7]:
```
# Function to calculate RMSE
def rmse(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))
```

[8]:
```
# Evaluate Random Forest model
rf_pred = rf_model.predict(X_test)
rf_mse = mean_squared_error(Y_test, rf_pred)
rf_mae = mean_absolute_error(Y_test, rf_pred)
rf_rmse = rmse(Y_test, rf_pred)
```

[9]:
```
# Evaluate Decision Tree model
dt_pred = dt_model.predict(X_test)
dt_mse = mean_squared_error(Y_test, dt_pred)
dt_mae = mean_absolute_error(Y_test, dt_pred)
dt_rmse = rmse(Y_test, dt_pred)
```

[10]:
```
# Visualize predicted vs actual house prices for Random Forest
plt.figure(figsize=(10, 6))
plt.scatter(Y_test, rf_pred, color='blue', label='Random Forest', alpha=0.5)
plt.plot([min(Y_test), max(Y_test)], [min(Y_test), max(Y_test)], color='red',␣
 ↪linestyle='--')
plt.xlabel("Actual House Prices")
plt.ylabel("Predicted House Prices")
plt.title("Random Forest: Actual vs Predicted House Prices")
plt.legend()
plt.grid(True)
plt.show()
```

[11]:
```
# Visualize predicted vs actual house prices for Decision Tree
plt.figure(figsize=(10, 6))
plt.scatter(Y_test, dt_pred, color='green', label='Decision Tree', alpha=0.5)
plt.plot([min(Y_test), max(Y_test)], [min(Y_test), max(Y_test)], color='red',␣
↪linestyle='--')
plt.xlabel("Actual House Prices")
plt.ylabel("Predicted House Prices")
plt.title("Decision Tree: Actual vs Predicted House Prices")
plt.legend()
plt.grid(True)
plt.show()
```



[12]:
```
# Compare the performance of Random Forest and Decision Tree models
print("\nModel Comparison:")
print("Random Forest - Mean Squared Error:", rf_mse)
print("Decision Tree - Mean Squared Error:", dt_mse)

if rf_mse < dt_mse:
    print("\nRandom Forest performs better in terms of Mean Squared Error. It␣
 ↪is preferred for analyzing the data.")
elif rf_mse > dt_mse:
    print("\nDecision Tree performs better in terms of Mean Squared Error. It␣
 ↪is preferred for analyzing the data.")
else:
    print("\nBoth models have similar performance in terms of Mean Squared␣
 ↪Error.")
```

```
Model Comparison:
Random Forest - Mean Squared Error: 32.41107270244575
Decision Tree - Mean Squared Error: 66.47144578313254

Random Forest performs better in terms of Mean Squared Error. It is preferred
for analyzing the data.
```

[ ]: