1. Data Loading

I started by importing the dataset:

```
df = pd.read_csv('Housing.csv')
```

- This loaded all housing data into a Data Frame.

---

2. Data Preprocessing

I cleaned and prepared the dataset for modelling:

- Used LabelEncoder to convert categorical variables (like 'furnishing status', 'main road', etc.) into numeric values.

```
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

- Split the data into:
    - **Features (X)** → All columns **except price**
    - **Target (y)** → price

---

3. Train-Test Split

Divided dataset into training and test sets (80/20 split):

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

---

4. Model 1: Linear Regression

I built a simple linear regression model:

```
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
LR.fit(X_train, y_train)
y_pred = LR.predict(X_test)
```

Evaluation:

```
from sklearn.metrics import r2_score, mean_squared_error
print("R² Score:", r2_score(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
```

- Accuracy: Around **61% R²**, indicating moderate performance.

---

5. Model 2: Random Forest Regressor

You also tried a more advanced model:

```
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor(random_state=42)
RF.fit(X_train, y_train)
y_pred_rf= RF.predict(X_test)
```

Evaluation:

- Accuracy: Around **64% R²**, slightly better than linear regression.

Random Forest is better at capturing complex patterns.

📊 6. Feature Importance (Random Forest)

You visualized which features were most important:

```
import matplotlib.pyplot as plt
plt.barh(X.columns, rf_model.feature_importances_)
plt.title("Feature Importances")
plt.show()
```