Walmart_Sales_Prediction_Analysis - ML

```
import numpy as np
import pandas as pd

df = pd.read_csv("Walmart_sales.csv")
print(df.head())
```

```
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature
Fuel_Price  \
0      1  05-02-2010    1643690.90             0        42.31
2.572
1      1  12-02-2010    1641957.44             1        38.51
2.548
2      1  19-02-2010    1611968.17             0        39.93
2.514
3      1  26-02-2010    1409727.59             0        46.63
2.561
4      1  05-03-2010    1554806.68             0        46.50
2.625

          CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
```

```
df.head()
```

```
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature
Fuel_Price  \
0      1  05-02-2010    1643690.90             0        42.31
2.572
1      1  12-02-2010    1641957.44             1        38.51
2.548
2      1  19-02-2010    1611968.17             0        39.93
2.514
3      1  26-02-2010    1409727.59             0        46.63
2.561
4      1  05-03-2010    1554806.68             0        46.50
2.625

          CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
```

```
df.isna().sum()
```

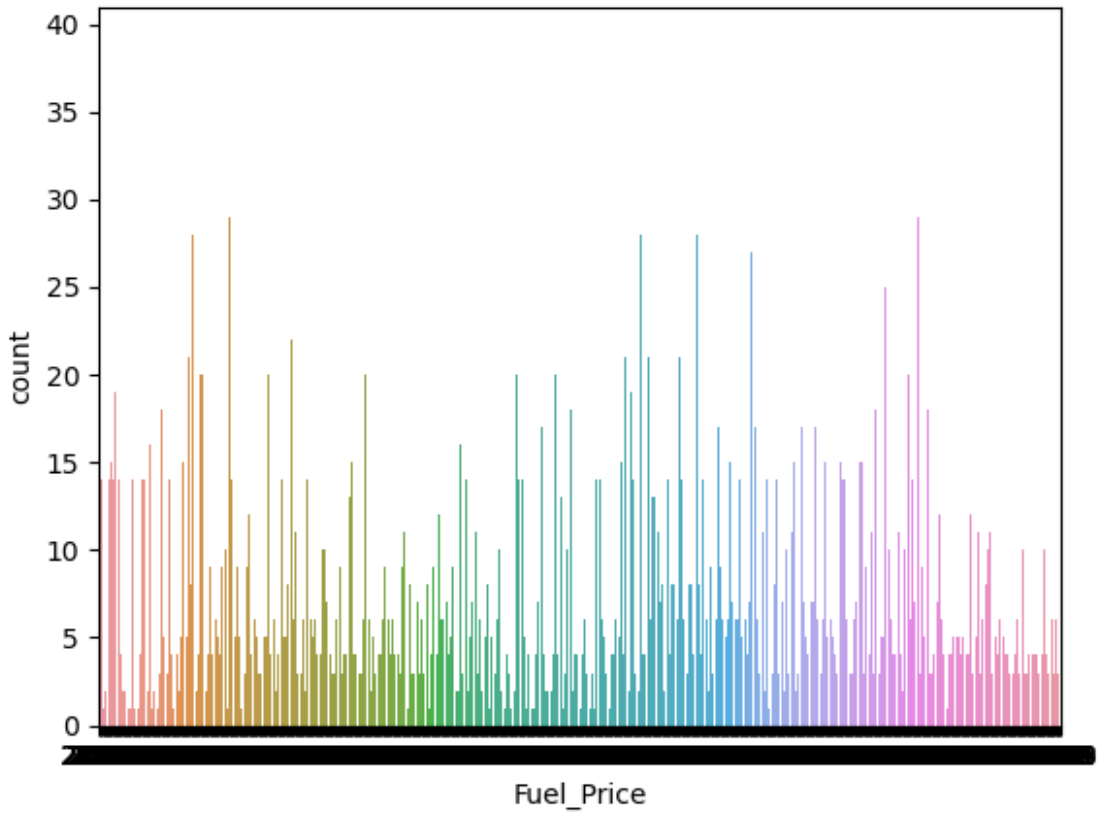
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(x="Fuel_Price" , data=df)
plt.show()
```




```
df.head()
```

```
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature
Fuel_Price  \
0      1  05-02-2010    1643690.90             0        42.31
2.572
1      1  12-02-2010    1641957.44             1        38.51
2.548
2      1  19-02-2010    1611968.17             0        39.93
2.514
3      1  26-02-2010    1409727.59             0        46.63
2.561
4      1  05-03-2010    1554806.68             0        46.50
2.625

           CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingRegressor
from statsmodels.graphics.tsaplots import plot_acf
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
df['Store'].unique()
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45])
```

```
data= df['Weekly_Sales'].sum()
print(f" Total sales in walmart: $",data)
```

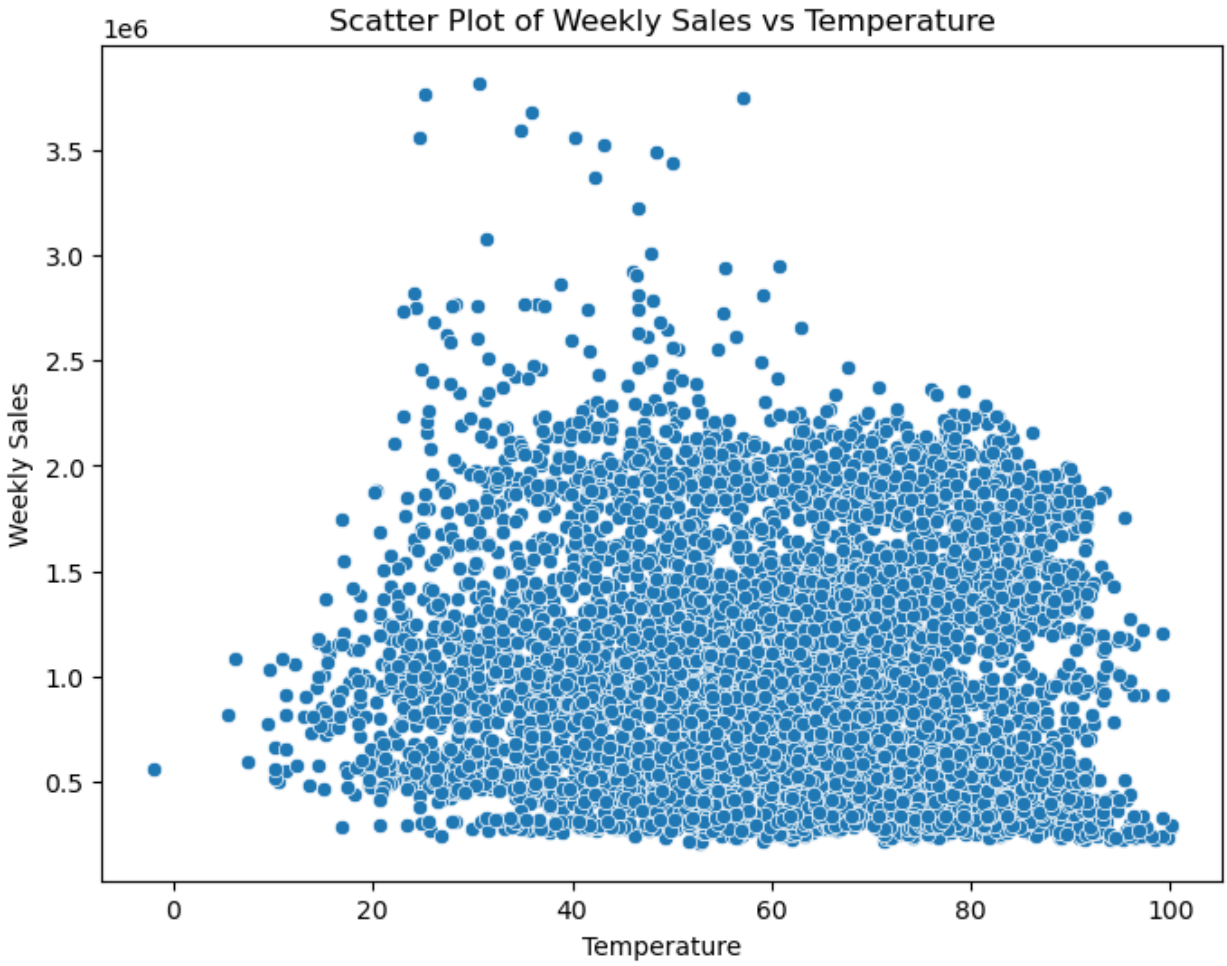
```
 Total sales in walmart: $ 6737218987.11
```

```
df.describe()
```

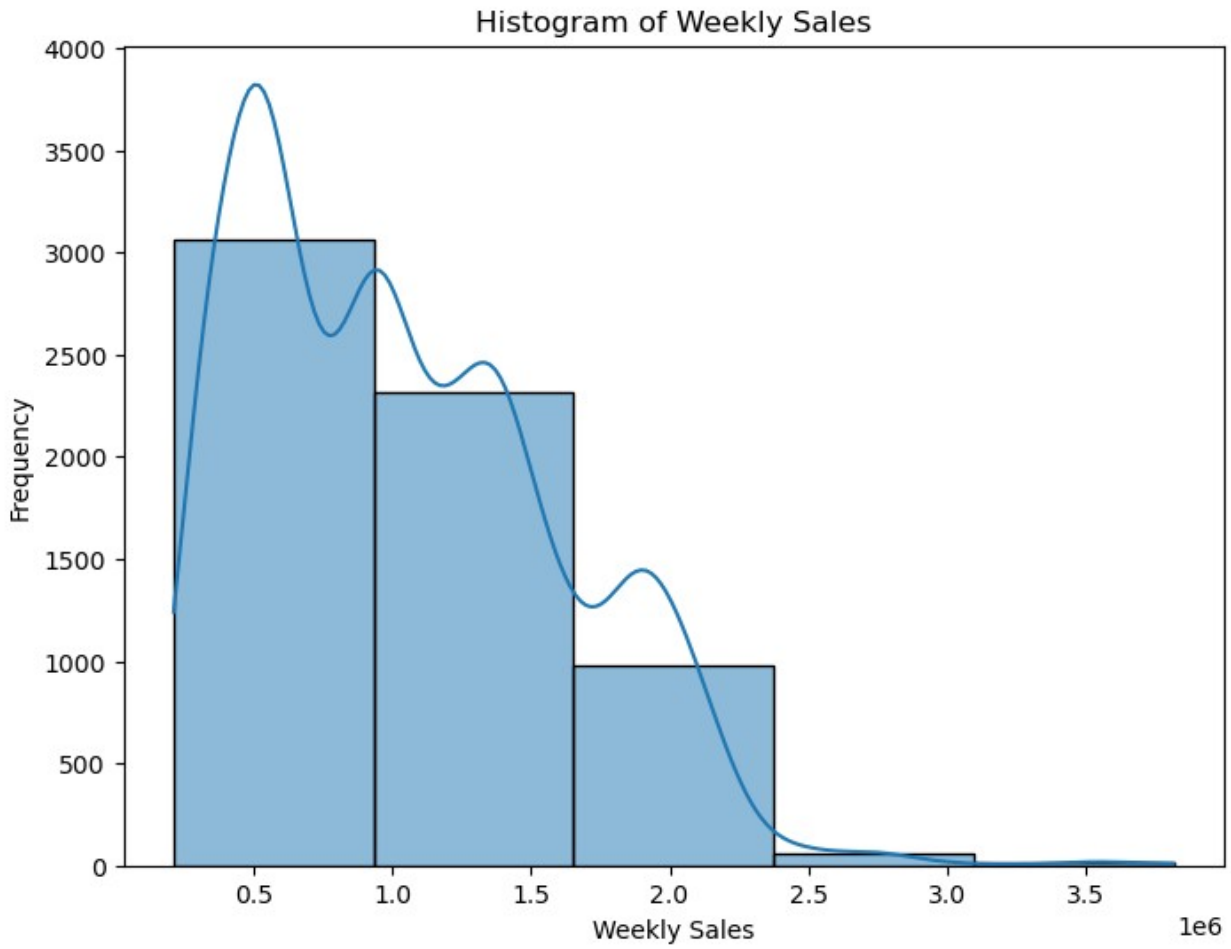
```
              Store  Weekly_Sales  Holiday_Flag  Temperature
Fuel_Price  \
count  6435.000000  6.435000e+03   6435.000000  6435.000000
6435.000000
mean     23.000000  1.046965e+06      0.069930    60.663782
3.358607
std      12.988182  5.643666e+05      0.255049    18.444933
0.459020
min       1.000000  2.099862e+05      0.000000    -2.060000
2.472000
25%      12.000000  5.533501e+05      0.000000    47.460000
2.933000
50%      23.000000  9.607460e+05      0.000000    62.670000
3.445000
75%      34.000000  1.420159e+06      0.000000    74.940000
3.735000
max      45.000000  3.818686e+06      1.000000   100.140000
4.468000

                CPI  Unemployment
count  6435.000000   6435.000000
mean    171.578394      7.999151
std      39.356712      1.875885
min     126.064000      3.879000
25%     131.735000      6.891000
50%     182.616521      7.874000
75%     212.743293      8.622000
max     227.232807     14.313000
```

```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='Temperature', y='Weekly_Sales')
plt.title('Scatter Plot of Weekly Sales vs Temperature')
plt.xlabel('Temperature')
plt.ylabel('Weekly Sales')
plt.show()
```



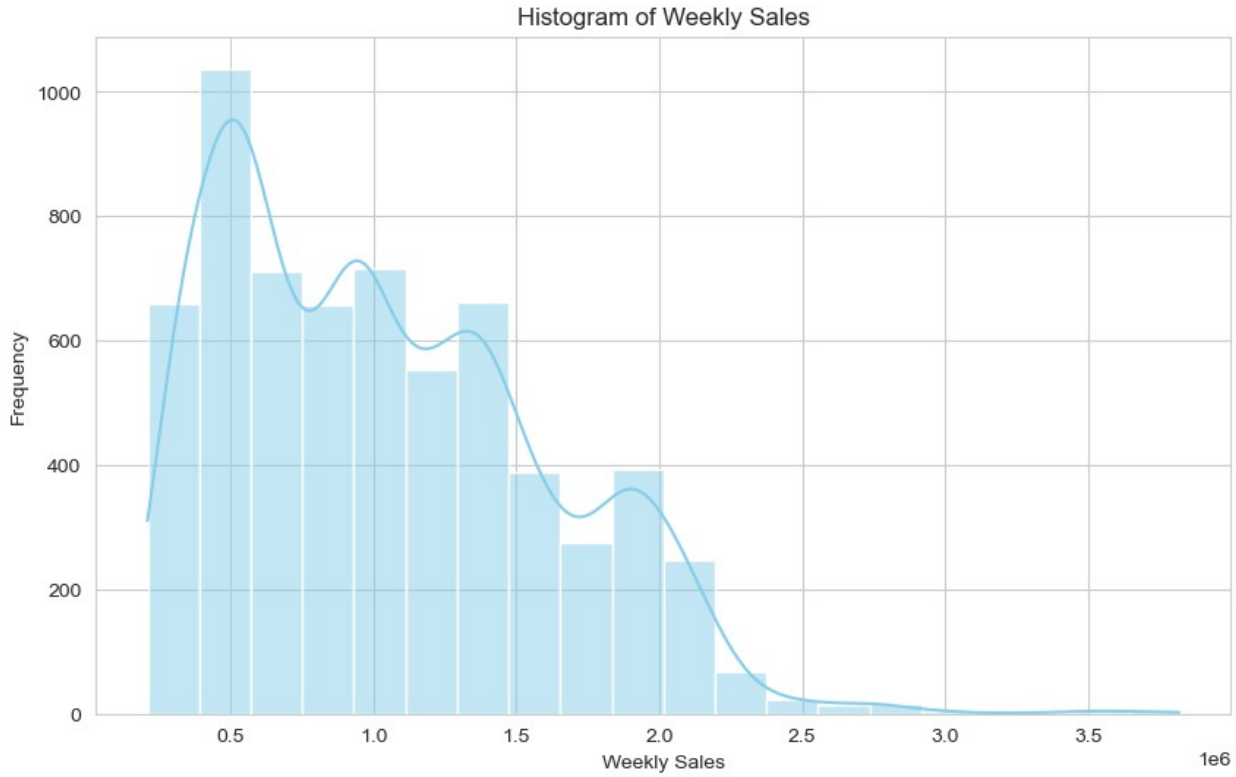

```
plt.figure(figsize=(8, 6))
sns.histplot(df['Weekly_Sales'], bins=5, kde = True)
plt.title('Histogram of Weekly Sales')
plt.xlabel('Weekly Sales')
plt.ylabel('Frequency')
plt.show()
```




```
df.drop(columns=['Date'], inplace=True)

sns.set_style("whitegrid")

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Weekly_Sales', bins=20, kde=True,
color='skyblue')
plt.title('Histogram of Weekly Sales')
plt.xlabel('Weekly Sales')
plt.ylabel('Frequency')
plt.show()
```




```
df.head()
```

```
   Store  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price
CPI  \
0      1    1643690.90             0        42.31       2.572
211.096358
1      1    1641957.44             1        38.51       2.548
211.242170
2      1    1611968.17             0        39.93       2.514
211.289143
3      1    1409727.59             0        46.63       2.561
211.319643
4      1    1554806.68             0        46.50       2.625
211.350143

   Unemployment
0         8.106
1         8.106
2         8.106
3         8.106
4         8.106
```

```
X = df[['Temperature', 'Fuel_Price', 'CPI', 'Unemployment']]
y = df['Weekly_Sales']
```

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

model_names = ['Linear Regression', 'Decision Tree', 'Random Forest',
'Gradient Regressor']

scaler = StandardScaler()
selector = SelectKBest(score_func=f_regression, k='all')

models = {
    'Linear Regression' : LinearRegression(),
    'Decision Tree' : DecisionTreeRegressor(),
    'Random Forest' : RandomForestRegressor(),
    'Gradient Regressor' : GradientBoostingRegressor()
}

score_list = []

for names in model_names:
    model = models[names]
    X_train_scaled = scaler.fit_transform(X_train)
    X_train_selected = selector.fit_transform(X_train_scaled, y_train)

    scores = cross_val_score(model, X_train_selected, y_train, cv=4)
    score_list.append(scores)
    print(f"{names}:")
    print(f"Mean R^2 Score: {scores.mean():.4f}, Std Dev:
{scores.std():.4f}\n")
```

```
Linear Regression:
Mean R^2 Score: 0.0241, Std Dev: 0.0053

Decision Tree:
Mean R^2 Score: -0.2769, Std Dev: 0.0273

Random Forest:
Mean R^2 Score: 0.1532, Std Dev: 0.0148

Gradient Regressor:
Mean R^2 Score: 0.2190, Std Dev: 0.0038
```

```
best_model_index = max(range(len(score_list)), key=lambda i:
score_list[i].mean())
best_model_name = model_names[best_model_index]
best_model = models[best_model_name]

print(best_model)
```

```
GradientBoostingRegressor()
```

```
best_model.fit(X_train_selected, y_train)
```

```
GradientBoostingRegressor()
```

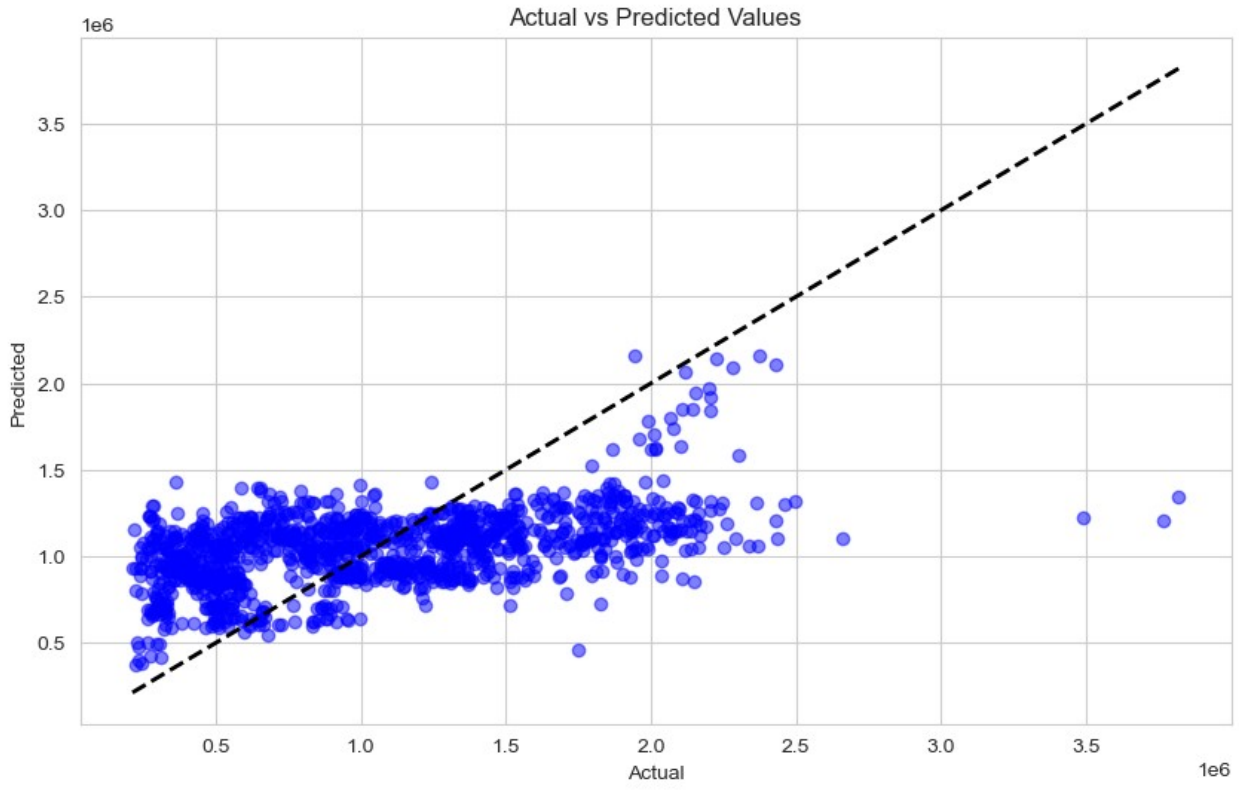
```
X_test_scaled = scaler.transform(X_test)
X_test_selected = selector.transform(X_test_scaled)

y_pred = best_model.predict(X_test_selected)

r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
print("r^2 score:", r2)
print("MAE:", mae)
print("RMSE:", np.sqrt(mse))
```

```
r^2 score: 0.23447899179860932
MAE: 411907.15795929066
RMSE: 496604.55808932683
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'k--', lw=2)  # Plotting the diagonal line
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted Values')
plt.show()
```


Actual vs Predicted Values
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
Predicted
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Actual
1e6