In [1]:
```
import pandas as pd
import numpy as np
df = pd.read_excel('StockX Click Dataset.xlsx')
```

In [2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from scipy import stats
from sklearn.preprocessing import LabelEncoder
```

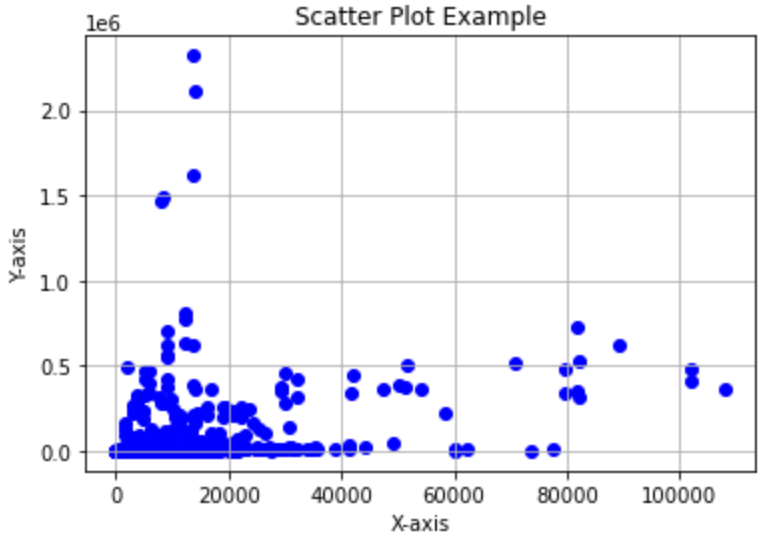
In [3]:
```
df = pd.DataFrame(df)
dff = df.fillna(0)
dff = dff[dff['Media Cost'] >= 2]
dff = dff[dff['Media Cost'] >= 1]
dff
```

Out[3]:

	Campaign Name	Objective	Platform	Media Cc
0	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	107906.7000
1	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	102058.1599
2	C3_DABA_FBIG_US_CO_CatalogSales_AlwaysOn_Socia...	CatalogSales	FBIG	101864.2999
3	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	89141.4499
4	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	81975.4600
...	...	...	...	
605	C3_BrandTraffic_TikTok_US_AW_Traffic_AlwaysOn_...	Traffic	TikTok	49.5600
606	C3_REM-DSA_Google_US_Brand+Nonbrand_CPA_DSA_Se...	CPA	Google	46.5800
607	C3_Electronics_Google_US_Nonbrand_CPA_Search_S...	CPA	Google	43.4800
608	C3_REM-Accessories_Google_US_Nonbrand_CPA_Sear...	CPA	Google	39.7700
609	c3_us_DOOHviaDV360_all_psp_awareness_both_8.15.22	awareness	DV360	29.0301

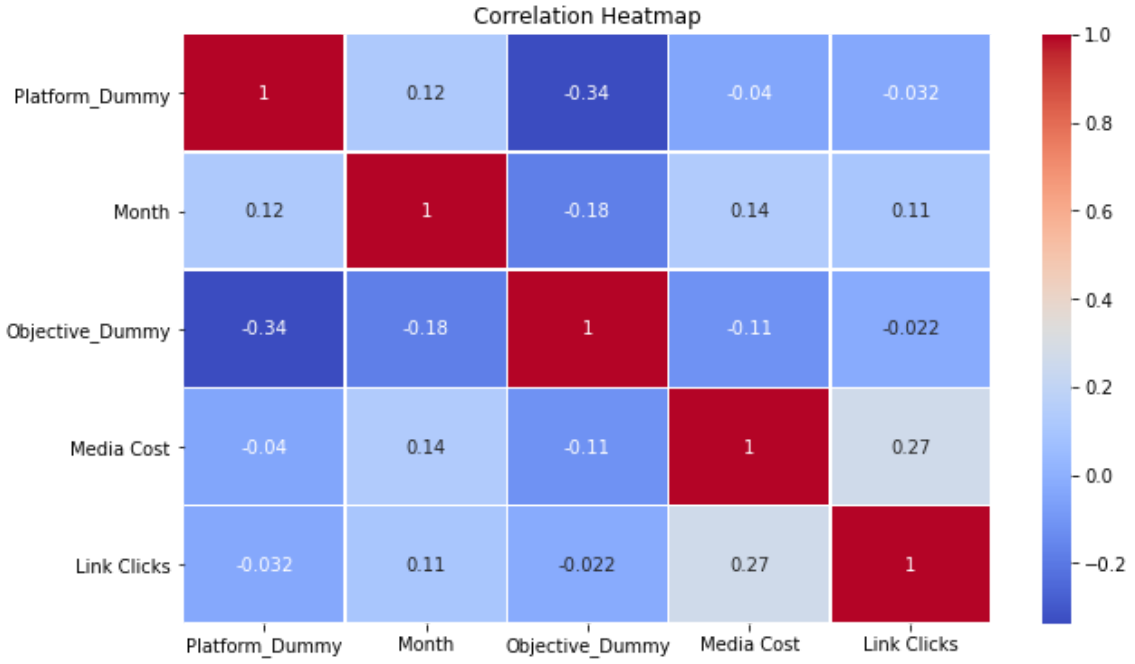
610 rows × 6 columns

In [4]:
```
plt.scatter(dff['Media Cost'], dff['Link Clicks'], color='blue', mark
plt.title('Scatter Plot Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)  # Adding a grid

# Display the plot
plt.show()
```

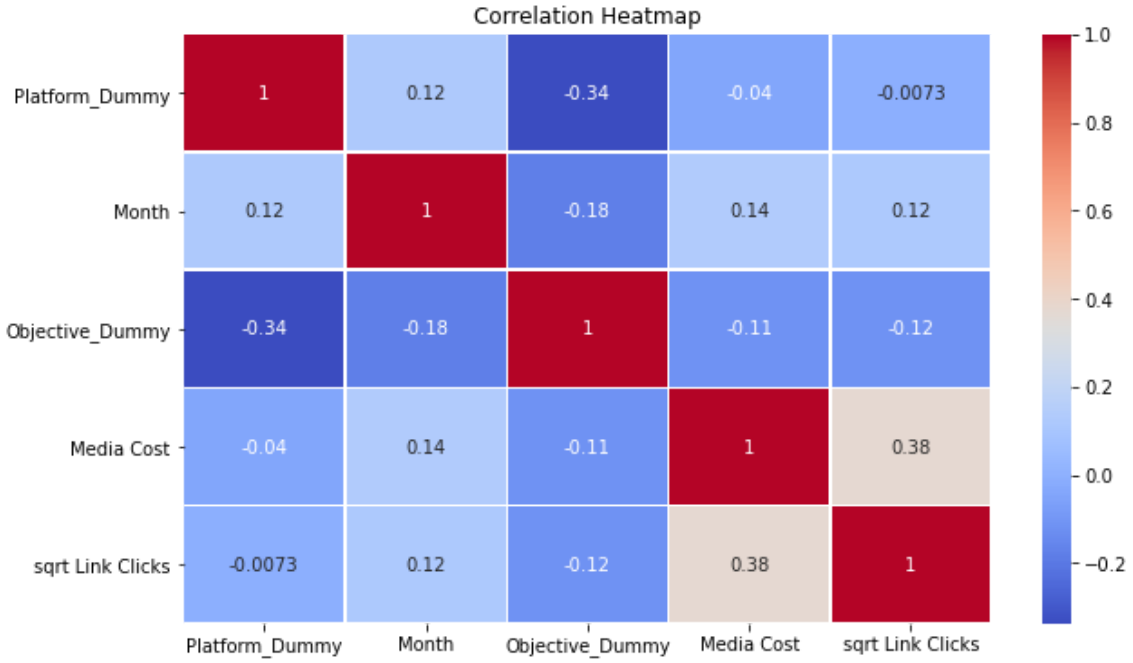


In [5]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
# Use LabelEncoder to convert 'Category' to integer labels
label_encoder = LabelEncoder()
dff['Platform_Dummy'] = label_encoder.fit_transform(dff['Platform'])
dff['Objective_Dummy'] = label_encoder.fit_transform(dff['Objective']
```

In [6]:
```
dff = dff.fillna(0)
dff.reset_index(drop=True, inplace=True)  # Reset the index to ensure

corr_df = dff.loc[1:609,['Platform_Dummy','Month','Objective_Dummy','

df = pd.DataFrame(corr_df)

# Compute the correlation matrix
correlation_matrix = df.corr()

plt.figure(figsize=(10, 6))  # Adjust the figure size as needed

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidt
plt.title('Correlation Heatmap')
plt.show()
```



In [7]:
```
dff['sqrt Link Clicks'] = np.sqrt(dff['Link Clicks'])

dff = dff.fillna(0)
dff.reset_index(drop=True, inplace=True)  # Reset the index to ensure

corr_df = dff.loc[1:609,['Platform_Dummy','Month','Objective_Dummy','

df = pd.DataFrame(corr_df)

# Compute the correlation matrix
correlation_matrix = df.corr()

plt.figure(figsize=(10, 6))  # Adjust the figure size as needed

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidt
plt.title('Correlation Heatmap')
plt.show()
```



In [8]:
```
import statsmodels.api as sm

dff = dff.fillna(0)

dff['Month'] = pd.to_numeric(dff['Month'], errors='coerce')

# Define your dependent variable (y) and independent variables (X)
y = dff['sqrt Link Clicks']
#X = dff[['Platform_Dummy','Month','Objective_Dummy','Spend','Country
X = dff[['Platform_Dummy','Month','Objective_Dummy','Media Cost']]
```

In [9]:
```
dff2 = dff[['sqrt Link Clicks','Platform_Dummy','Month','Objective_Du
```

In [10]:
```
from statsmodels.stats.outliers_influence import variance_inflation_f

# Assuming 'new_X' is your new DataFrame with predictor variables
vif_data_new = pd.DataFrame()
vif_data_new["Predictor"] = dff2.columns
vif_data_new["VIF"] = [variance_inflation_factor(dff2.values, i) for

# Check for high VIF values (generally VIF > 5 indicates multicolline
print(vif_data_new)
```

```
          Predictor       VIF
0  sqrt Link Clicks  1.878831
1    Platform_Dummy  3.228178
2             Month  3.585139
3   Objective_Dummy  2.257360
4        Media Cost  1.952456
```

In [11]:
```
import statsmodels.api as sm

# Function to perform forward stepwise variable selection
def forward_stepwise_selection(X, y):
    included = []
    remaining = list(X.columns)
    best_model = None
    prev_score = float('-inf')

    while remaining:
        scores_with_candidates = []

        for candidate in remaining:
            X_i = X[included + [candidate]]
            X_i = sm.add_constant(X_i)  # Add a constant term (interc
            model = sm.OLS(y, X_i).fit()
            score = model.rsquared_adj  # Adjusted R-squared (you can
            scores_with_candidates.append((score, candidate))

        scores_with_candidates.sort(reverse=True)
        best_score, best_candidate = scores_with_candidates[0]

        if best_score > prev_score:
            included.append(best_candidate)
            remaining.remove(best_candidate)
            prev_score = best_score
            best_model = model  # Move this line outside the if-else
        else:
            break

    return included, best_model

# Perform forward stepwise variable selection
selected_features, best_regression_model = forward_stepwise_selection

if best_regression_model is not None:
    print("Selected Features:", selected_features)
    print(best_regression_model.summary())
else:
    print("No model was created.")
```

```
Selected Features: ['Media Cost', 'Objective_Dummy', 'Month']
                            OLS Regression Results
==============================================================================
==========
Dep. Variable:       sqrt Link Clicks   R-squared:
0.155
Model:                            OLS   Adj. R-squared:
0.151
Method:                 Least Squares   F-statistic:
37.14
Date:                Thu, 07 Dec 2023   Prob (F-statistic):
4.84e-22
Time:                        14:06:10   Log-Likelihood:
-4089.8
No. Observations:                 610   AIC:
8188.
Df Residuals:                     606   BIC:
8205.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
==============
                     coef    std err          t      P>|t|      [0.
025      0.975]
------------------------------------------------------------------------------
--------------
const            104.9905     24.310      4.319      0.000      57.
248     152.733
Media Cost         0.0054      0.001      9.723      0.000       0.
004       0.006
Objective_Dummy   -5.2299      3.098     -1.688      0.092     -11.
315       0.855
Month              3.8339      2.498      1.535      0.125      -1.
072       8.740
==============================================================================
==========
Omnibus:                      386.502   Durbin-Watson:
1.591
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
3879.494
Skew:                           2.729   Prob(JB):
0.00
Kurtosis:                      14.084   Cond. No.
5.86e+04
==============================================================================
==========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
[2] The condition number is large, 5.86e+04. This might indicate tha
t there are
strong multicollinearity or other numerical problems.
```

In [12]:
```
#Now a non linear model
import pandas as pd
import numpy as np
df = pd.read_excel('StockX Click Dataset.xlsx')
```

In [13]:
```
df = pd.DataFrame(df)
dff = df.fillna(0)
dff = dff[dff['Media Cost'] >= 2]
dff = dff[dff['Media Cost'] >= 1]
dff
```

Out[13]:

	Campaign Name	Objective	Platform	Media Cc
0	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	107906.7000
1	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	102058.1599
2	C3_DABA_FBIG_US_CO_CatalogSales_AlwaysOn_Socia...	CatalogSales	FBIG	101864.2999
3	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	89141.4499
4	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	81975.4600
...	...	...	...	
605	C3_BrandTraffic_TikTok_US_AW_Traffic_AlwaysOn_...	Traffic	TikTok	49.5600
606	C3_REM-DSA_Google_US_Brand+Nonbrand_CPA_DSA_Se...	CPA	Google	46.5800
607	C3_Electronics_Google_US_Nonbrand_CPA_Search_S...	CPA	Google	43.4800
608	C3_REM-Accessories_Google_US_Nonbrand_CPA_Sear...	CPA	Google	39.7700
609	c3_us_DOOHviaDV360_all_psp_awareness_both_8.15.22	awareness	DV360	29.0301

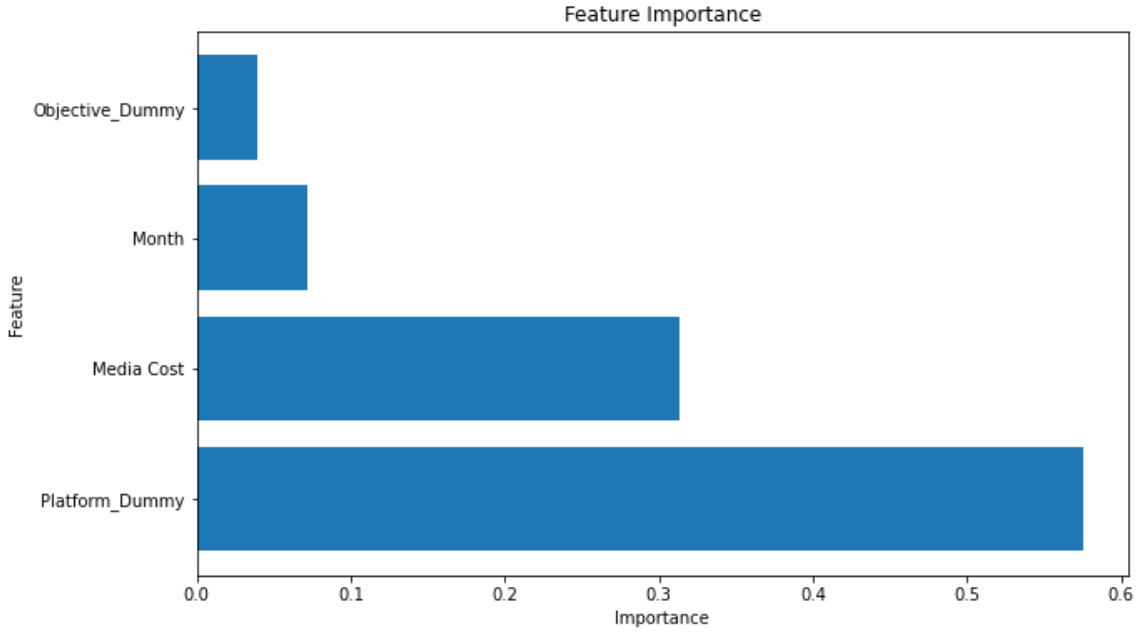
610 rows × 6 columns

In [14]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
# Use LabelEncoder to convert 'Category' to integer labels
label_encoder = LabelEncoder()
dff['Platform_Dummy'] = label_encoder.fit_transform(dff['Platform'])
dff['Objective_Dummy'] = label_encoder.fit_transform(dff['Objective']
```

In [15]:
```
dff['sqrt Link Clicks'] = np.sqrt(dff['Link Clicks'])
dff = dff.fillna(0)
dff.reset_index(drop=True, inplace=True)  # Reset the index to ensure

import statsmodels.api as sm

dff = dff.fillna(0)

dff['Month'] = pd.to_numeric(dff['Month'], errors='coerce')

# Define your dependent variable (y) and independent variables (X)
y = dff['sqrt Link Clicks']
#X = dff[['Platform_Dummy','Month','Objective_Dummy','Spend','Country
X = dff[['Platform_Dummy','Month','Objective_Dummy','Media Cost']]
```

In [16]:
```
from sklearn.ensemble import RandomForestRegressor


rf = RandomForestRegressor(n_estimators=100, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

rf.fit(X_train, y_train)

feature_names = X_train.columns

importances = rf.feature_importances_

# Create a DataFrame to store feature names and their importances
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Impo

# Sort the DataFrame by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importa
```

In [17]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Imp
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importance')
plt.show()
```



In [18]:
```
dff.reset_index(drop=True, inplace=True)  # Reset the index to ensure
X = dff[['Platform_Dummy','Media Cost']]
y = dff['sqrt Link Clicks']

#y = y.reshape(-1, 1)
#X = X.reshape(-1, 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0


from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor

# Create a DecisionTreeRegressor model
reg_tree = DecisionTreeRegressor()

# Define hyperparameters and their possible values for tuning
param_grid = {
    'max_depth': [None, 5, 10, 15],
    'min_samples_leaf': [5, 10, 15],
    'min_samples_split': [5, 10, 15],
    #'max_features': ['auto', 'sqrt', 'log2'],
    'min_impurity_decrease': [0.0, 0.01, 0.05]
}

# Create a GridSearchCV object for hyperparameter tuning
grid_search = GridSearchCV(reg_tree, param_grid, cv=5, scoring='neg_m

# Fit the model to the data and find the best hyperparameters
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
```

In [19]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Evaluate the model on the test data
y_pred = grid_search.best_estimator_.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error on Test Data: {mse}")
print(f"R-squared on Test Data: {r2}")

# Retrain the model on the entire dataset with the best hyperparamete
best_regressor = grid_search.best_estimator_
best_regressor.fit(X, y)
```

```
Mean Squared Error on Test Data: 19830.885425685818
R-squared on Test Data: 0.6029126552510202
```

Out[19]:
```
DecisionTreeRegressor(max_depth=5, min_samples_leaf=5, min_samples_s
plit=5)
```

In [ ]:

In [20]:
```
#pip install xgboost
```

In [21]:
```
#Now a non linear model
import pandas as pd
import numpy as np
df = pd.read_excel('StockX Click Dataset.xlsx')
```

In [22]:
```
df = pd.DataFrame(df)
dff = df.fillna(0)
dff = dff[dff['Media Cost'] >= 2]
dff = dff[dff['Media Cost'] >= 1]
dff
```

Out[22]:

	Campaign Name	Objective	Platform	Media Cc
0	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	107906.7000
1	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	102058.1599
2	C3_DABA_FBIG_US_CO_CatalogSales_AlwaysOn_Socia...	CatalogSales	FBIG	101864.2999
3	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	89141.4499
4	C3_Dynamic-Remarketing_FBIG_US_CO_CatalogSales...	CatalogSales	FBIG	81975.4600
...	...	...	...	
605	C3_BrandTraffic_TikTok_US_AW_Traffic_AlwaysOn_...	Traffic	TikTok	49.5600
606	C3_REM-DSA_Google_US_Brand+Nonbrand_CPA_DSA_Se...	CPA	Google	46.5800
607	C3_Electronics_Google_US_Nonbrand_CPA_Search_S...	CPA	Google	43.4800
608	C3_REM-Accessories_Google_US_Nonbrand_CPA_Sear...	CPA	Google	39.7700
609	c3_us_DOOHviaDV360_all_psp_awareness_both_8.15.22	awareness	DV360	29.0301

610 rows × 6 columns

In [23]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
# Use LabelEncoder to convert 'Category' to integer labels
label_encoder = LabelEncoder()
dff['Platform_Dummy'] = label_encoder.fit_transform(dff['Platform'])
dff['Objective_Dummy'] = label_encoder.fit_transform(dff['Objective']
```

In [24]:
```
dff['sqrt Link Clicks'] = np.sqrt(dff['Link Clicks'])
dff = dff.fillna(0)
dff.reset_index(drop=True, inplace=True)  # Reset the index to ensure

import statsmodels.api as sm

dff = dff.fillna(0)

dff['Month'] = pd.to_numeric(dff['Month'], errors='coerce')

# Define your dependent variable (y) and independent variables (X)
y = dff['sqrt Link Clicks']
#X = dff[['Platform_Dummy','Month','Objective_Dummy','Spend','Country
X = dff[['Platform_Dummy','Month','Objective_Dummy','Media Cost']]
```

In [25]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import xgboost as xgb

dff.reset_index(drop=True, inplace=True)  # Reset the index to ensure
X = dff[['Platform_Dummy','Media Cost']]
y = dff['sqrt Link Clicks']


# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# Create an XGBoost regressor
xg_reg = xgb.XGBRegressor(objective ='reg:squarederror', colsample_by
                max_depth = 5, alpha = 10, n_estimators = 100)

# Fit the model to the training data
xg_reg.fit(X_train, y_train)

# Predict on the test set
preds = xg_reg.predict(X_test)

# Calculate RMSE (Root Mean Squared Error)
rmse = mean_squared_error(y_test, preds, squared=False)
print("RMSE:", rmse)

from sklearn.metrics import r2_score

# Predict on the test set
preds = xg_reg.predict(X_test)

# Calculate R-squared
r_squared = r2_score(y_test, preds)
print("R-squared:", r_squared)
```

```
RMSE: 121.56521265297903
R-squared: 0.5470391424069768
```

In [ ]: