Quadratic Regression Dataset - Linear Regression vs XGBoost

Model is trained with XGBoost installed in notebook instance

In the later examples, we will train using SageMaker's XGBoost algorithm.

Training on SageMaker takes several minutes (even for simple dataset).

If algorithm is supported on Python, we will try them locally on notebook instance

This allows us to quickly learn an algorithm, understand tuning options and then finally train on SageMaker Cloud

In this exercise, let's compare XGBoost and Linear Regression for Quadratic regression dataset

In [1]:
```
# Install xgboost in notebook instance.
#### Command to install xgboost
#DWB#Discussed in comments
#DWB#!conda install -y -c conda-forge xgboost
#
#DWB#  I'm not going to uninstall the xgboost that
#DWB#+ I installed in the last lecture; we can see
#DWB#+ what the system will do
!pip install xgboost
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: xgboost in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (1.7.6)
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgb
oost) (1.22.3)
Requirement already satisfied: scipy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgb
oost) (1.10.1)
```

In [2]:
```
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
# XGBoost
import xgboost as xgb
# Linear Regression
from sklearn.linear_model import LinearRegression
```

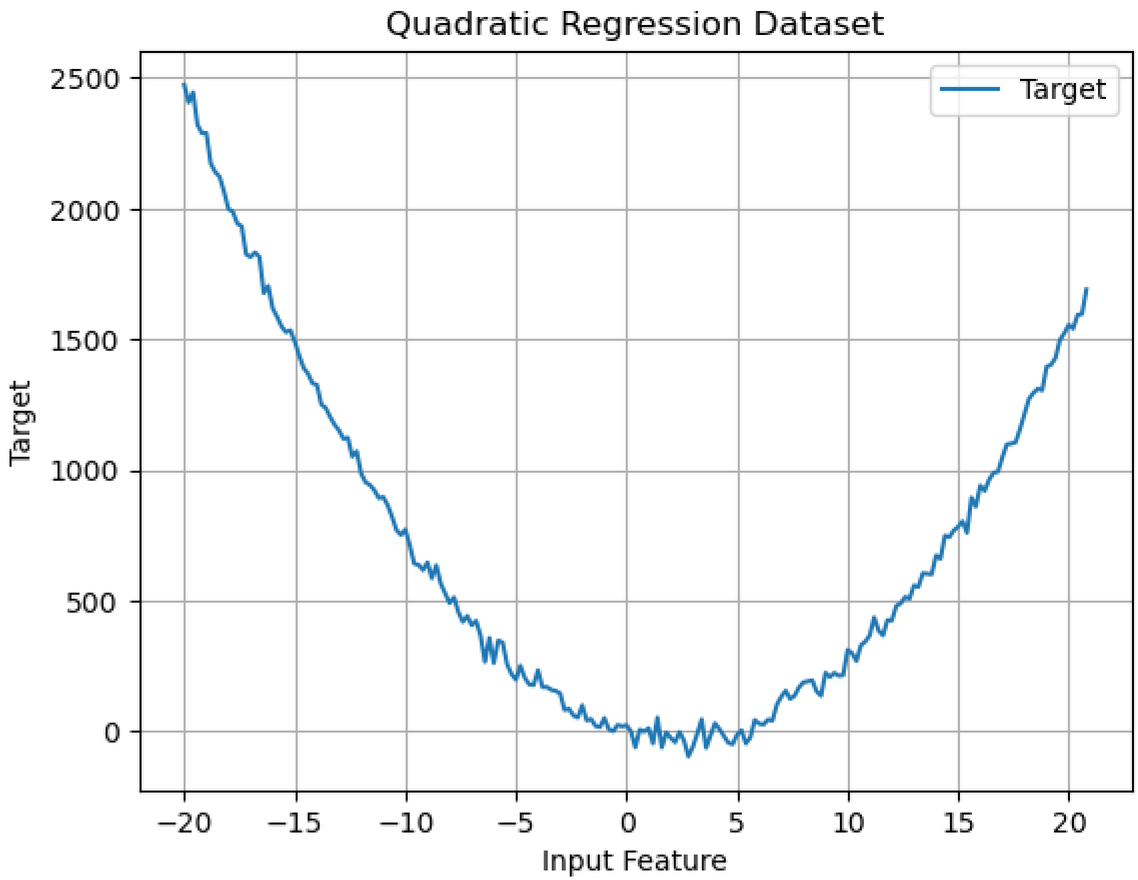
In [3]:
```
df = pd.read_csv('quadratic_all.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	x	y
0	-20.0	2473.236825
1	-19.8	2405.673895
2	-19.6	2444.523136
3	-19.4	2320.437236
4	-19.2	2288.088295

In [5]:
```
plt.plot(df.x,df.y,label='Target')
plt.grid(True)
plt.xlabel('Input Feature')
plt.ylabel('Target')
plt.legend()
plt.title('Quadratic Regression Dataset')
plt.show()
```



In [6]:
```
train_file = 'quadratic_train.csv'
validation_file = 'quadratic_validation.csv'

# Specify the column names as the file does not have column header
df_train = pd.read_csv(train_file,names=['y','x'])
df_validation = pd.read_csv(validation_file,names=['y','x'])
```

In [7]:
```
df_train.head()
```

Out[7]:

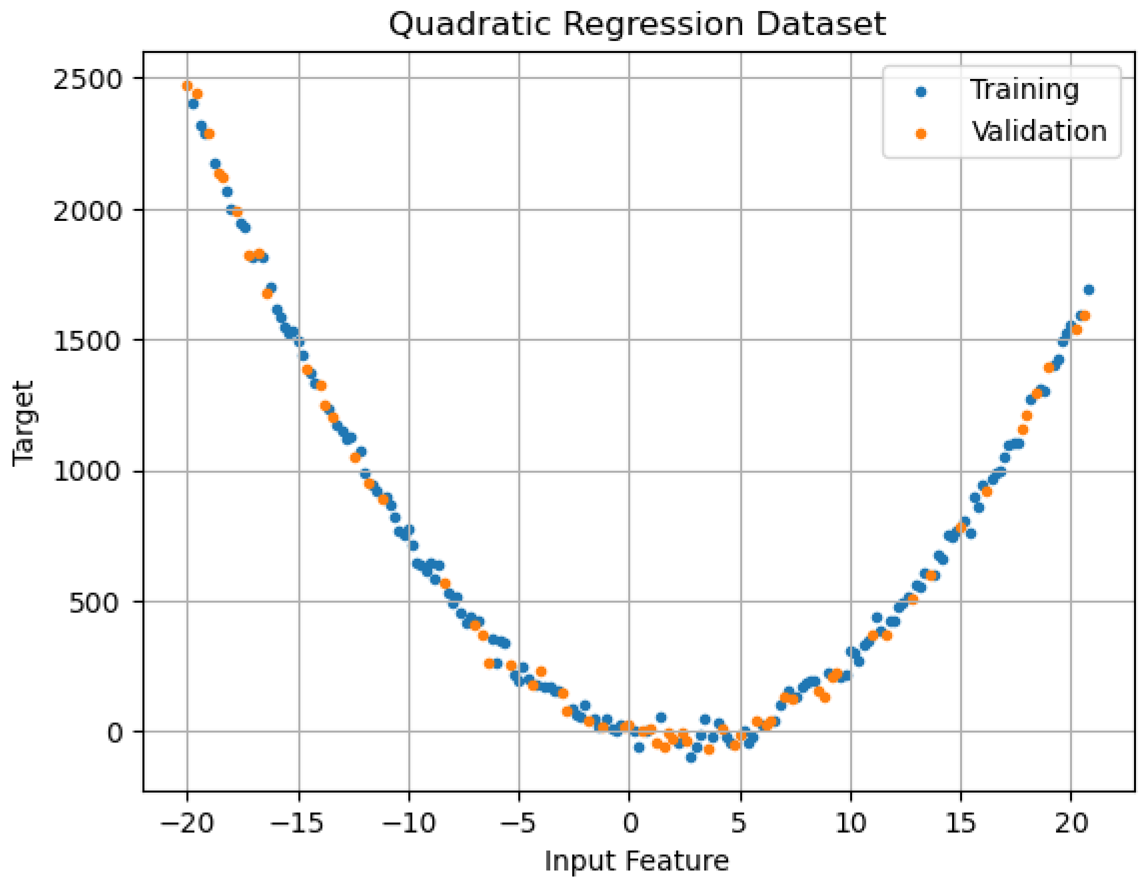
	y	x
0	343.968005	10.8
1	1585.894405	-15.8
2	1497.303317	19.6
3	769.909912	-10.4
4	1173.230755	-13.2

In [8]:
```
df_validation.head()
```

Out[8]:

	y	x
0	1824.856344	-17.2
1	16.997917	-1.2
2	1832.141730	-16.8
3	1395.206684	19.0
4	145.840543	-3.0

In [9]:
```
plt.scatter(df_train.x,df_train.y,label='Training',marker='.')
plt.scatter(df_validation.x,df_validation.y,label='Validation',marker='.')
plt.grid(True)
plt.xlabel('Input Feature')
plt.ylabel('Target')
plt.title('Quadratic Regression Dataset')
plt.legend()
plt.show()
```



In [10]:
```
X_train = df_train.iloc[:,1:] # Features: 1st column onwards
y_train = df_train.iloc[:,0].ravel() # Target: 0th column

X_validation = df_validation.iloc[:,1:]
y_validation = df_validation.iloc[:,0].ravel()
```

In [11]:
```
# Create an instance of XGBoost Regressor
# XGBoost Training Parameter Reference:
#   https://github.com/dmlc/xgboost/blob/master/doc/parameter.md
regressor = xgb.XGBRegressor()
```

In [12]:
```
regressor
```

Out[12]:

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [66]:

```
#DWB# I don't like the output with the scroll bar
print(str(regressor))
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

In [13]:

```
regressor.fit(X_train,y_train, eval_set = [(X_train, y_train), (X_validation, y_validation)])
```

```
[0]	validation_0-rmse:680.75659	validation_1-rmse:759.28186
[1]	validation_0-rmse:496.64975	validation_1-rmse:558.76229
[2]	validation_0-rmse:364.40194	validation_1-rmse:416.74504
[3]	validation_0-rmse:268.61849	validation_1-rmse:314.03880
[4]	validation_0-rmse:198.73166	validation_1-rmse:239.39934
[5]	validation_0-rmse:148.15570	validation_1-rmse:184.01248
[6]	validation_0-rmse:111.41606	validation_1-rmse:143.64577
[7]	validation_0-rmse:85.12823	validation_1-rmse:114.83409
[8]	validation_0-rmse:66.19105	validation_1-rmse:95.02868
[9]	validation_0-rmse:52.48116	validation_1-rmse:80.46168
[10]	validation_0-rmse:42.81858	validation_1-rmse:70.20043
[11]	validation_0-rmse:35.82252	validation_1-rmse:62.60704
[12]	validation_0-rmse:30.72047	validation_1-rmse:57.81083
[13]	validation_0-rmse:27.04723	validation_1-rmse:53.74323
[14]	validation_0-rmse:24.51246	validation_1-rmse:50.83495
[15]	validation_0-rmse:22.54053	validation_1-rmse:48.28755
[16]	validation_0-rmse:20.98229	validation_1-rmse:46.41355
[17]	validation_0-rmse:19.73797	validation_1-rmse:45.18608
[18]	validation_0-rmse:18.49679	validation_1-rmse:44.70341
[19]	validation_0-rmse:17.69560	validation_1-rmse:44.00994
[20]	validation_0-rmse:17.09966	validation_1-rmse:43.28699
[21]	validation_0-rmse:16.64862	validation_1-rmse:42.85340
[22]	validation_0-rmse:15.70011	validation_1-rmse:43.02807
[23]	validation_0-rmse:15.38755	validation_1-rmse:42.75773
[24]	validation_0-rmse:14.72696	validation_1-rmse:42.90691
[25]	validation_0-rmse:14.32629	validation_1-rmse:43.10545
[26]	validation_0-rmse:14.14502	validation_1-rmse:42.99868
[27]	validation_0-rmse:13.84486	validation_1-rmse:43.31779
[28]	validation_0-rmse:13.70856	validation_1-rmse:43.14159
[29]	validation_0-rmse:13.57877	validation_1-rmse:43.00242
[30]	validation_0-rmse:13.10335	validation_1-rmse:43.16242
[31]	validation_0-rmse:12.95686	validation_1-rmse:43.01017
[32]	validation_0-rmse:12.61189	validation_1-rmse:43.07995
[33]	validation_0-rmse:12.26656	validation_1-rmse:43.08501
[34]	validation_0-rmse:11.96268	validation_1-rmse:43.09133
[35]	validation_0-rmse:11.69330	validation_1-rmse:42.94814
[36]	validation_0-rmse:11.36168	validation_1-rmse:42.99236
[37]	validation_0-rmse:10.91526	validation_1-rmse:43.30779
[38]	validation_0-rmse:10.66809	validation_1-rmse:43.35736
[39]	validation_0-rmse:10.53745	validation_1-rmse:43.39308
[40]	validation_0-rmse:10.17034	validation_1-rmse:43.30790
[41]	validation_0-rmse:9.60657	validation_1-rmse:43.47400
```

```
[42]	validation_0-rmse:9.40088	validation_1-rmse:43.53024
[43]	validation_0-rmse:9.05288	validation_1-rmse:43.59005
[44]	validation_0-rmse:8.59284	validation_1-rmse:43.78224
[45]	validation_0-rmse:8.36865	validation_1-rmse:43.82842
[46]	validation_0-rmse:8.21436	validation_1-rmse:43.77688
[47]	validation_0-rmse:8.02011	validation_1-rmse:43.85358
[48]	validation_0-rmse:7.60991	validation_1-rmse:43.99822
[49]	validation_0-rmse:7.31762	validation_1-rmse:44.06958
[50]	validation_0-rmse:7.21769	validation_1-rmse:44.10243
[51]	validation_0-rmse:6.91921	validation_1-rmse:44.25390
[52]	validation_0-rmse:6.68621	validation_1-rmse:44.30911
[53]	validation_0-rmse:6.49543	validation_1-rmse:44.32807
[54]	validation_0-rmse:6.40539	validation_1-rmse:44.35911
[55]	validation_0-rmse:6.19920	validation_1-rmse:44.36742
[56]	validation_0-rmse:6.06104	validation_1-rmse:44.43427
[57]	validation_0-rmse:5.86804	validation_1-rmse:44.38549
[58]	validation_0-rmse:5.62227	validation_1-rmse:44.52854
[59]	validation_0-rmse:5.56722	validation_1-rmse:44.48940
[60]	validation_0-rmse:5.50182	validation_1-rmse:44.46332
[61]	validation_0-rmse:5.45560	validation_1-rmse:44.43528
[62]	validation_0-rmse:5.38756	validation_1-rmse:44.41671
[63]	validation_0-rmse:5.33709	validation_1-rmse:44.44324
[64]	validation_0-rmse:5.21123	validation_1-rmse:44.49044
[65]	validation_0-rmse:5.16929	validation_1-rmse:44.50380
[66]	validation_0-rmse:4.92579	validation_1-rmse:44.61582
[67]	validation_0-rmse:4.76708	validation_1-rmse:44.64403
[68]	validation_0-rmse:4.68881	validation_1-rmse:44.68129
[69]	validation_0-rmse:4.50568	validation_1-rmse:44.78865
[70]	validation_0-rmse:4.47962	validation_1-rmse:44.76700
[71]	validation_0-rmse:4.34575	validation_1-rmse:44.85468
[72]	validation_0-rmse:4.20597	validation_1-rmse:44.91148
[73]	validation_0-rmse:4.14287	validation_1-rmse:44.96600
[74]	validation_0-rmse:4.11514	validation_1-rmse:44.95660
[75]	validation_0-rmse:3.95120	validation_1-rmse:44.95081
[76]	validation_0-rmse:3.82405	validation_1-rmse:44.97570
[77]	validation_0-rmse:3.69686	validation_1-rmse:44.97686
[78]	validation_0-rmse:3.55601	validation_1-rmse:45.05957
[79]	validation_0-rmse:3.45422	validation_1-rmse:45.10153
[80]	validation_0-rmse:3.42310	validation_1-rmse:45.11102
[81]	validation_0-rmse:3.34021	validation_1-rmse:45.13074
[82]	validation_0-rmse:3.23519	validation_1-rmse:45.13293
[83]	validation_0-rmse:3.19969	validation_1-rmse:45.14602
```

```
[84]	validation_0-rmse:3.18501	validation_1-rmse:45.14520
[85]	validation_0-rmse:3.10588	validation_1-rmse:45.13691
[86]	validation_0-rmse:3.04586	validation_1-rmse:45.10998
[87]	validation_0-rmse:2.96672	validation_1-rmse:45.11537
[88]	validation_0-rmse:2.90470	validation_1-rmse:45.11700
[89]	validation_0-rmse:2.85249	validation_1-rmse:45.13259
[90]	validation_0-rmse:2.73530	validation_1-rmse:45.20306
[91]	validation_0-rmse:2.65719	validation_1-rmse:45.24547
[92]	validation_0-rmse:2.60856	validation_1-rmse:45.23182
[93]	validation_0-rmse:2.60290	validation_1-rmse:45.24003
[94]	validation_0-rmse:2.52591	validation_1-rmse:45.24687
[95]	validation_0-rmse:2.45369	validation_1-rmse:45.29898
[96]	validation_0-rmse:2.40118	validation_1-rmse:45.32123
[97]	validation_0-rmse:2.31446	validation_1-rmse:45.37927
[98]	validation_0-rmse:2.29170	validation_1-rmse:45.39479
[99]	validation_0-rmse:2.23361	validation_1-rmse:45.39562
```

Out[13]:

▾ XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [14]:

```
eval_result = regressor.evals_result()
```

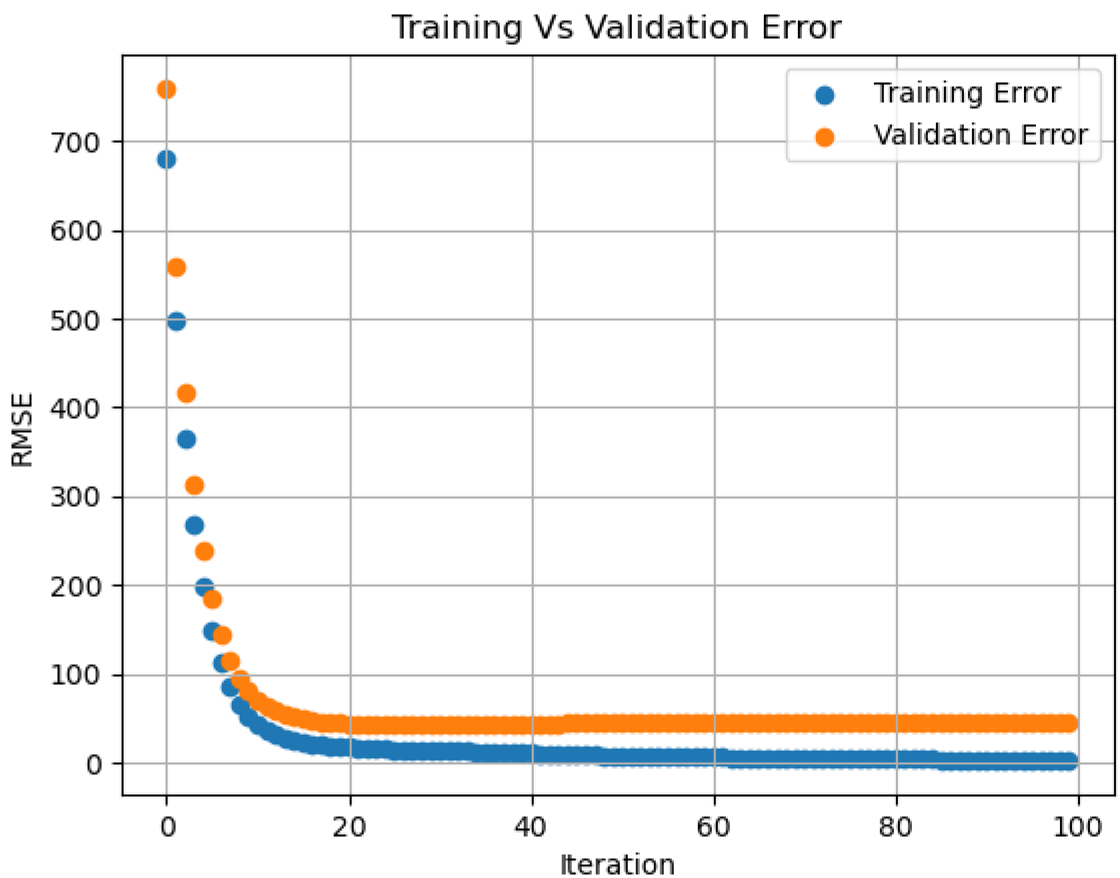
In [15]:

```
training_rounds = range(len(eval_result['validation_0']['rmse']))
```

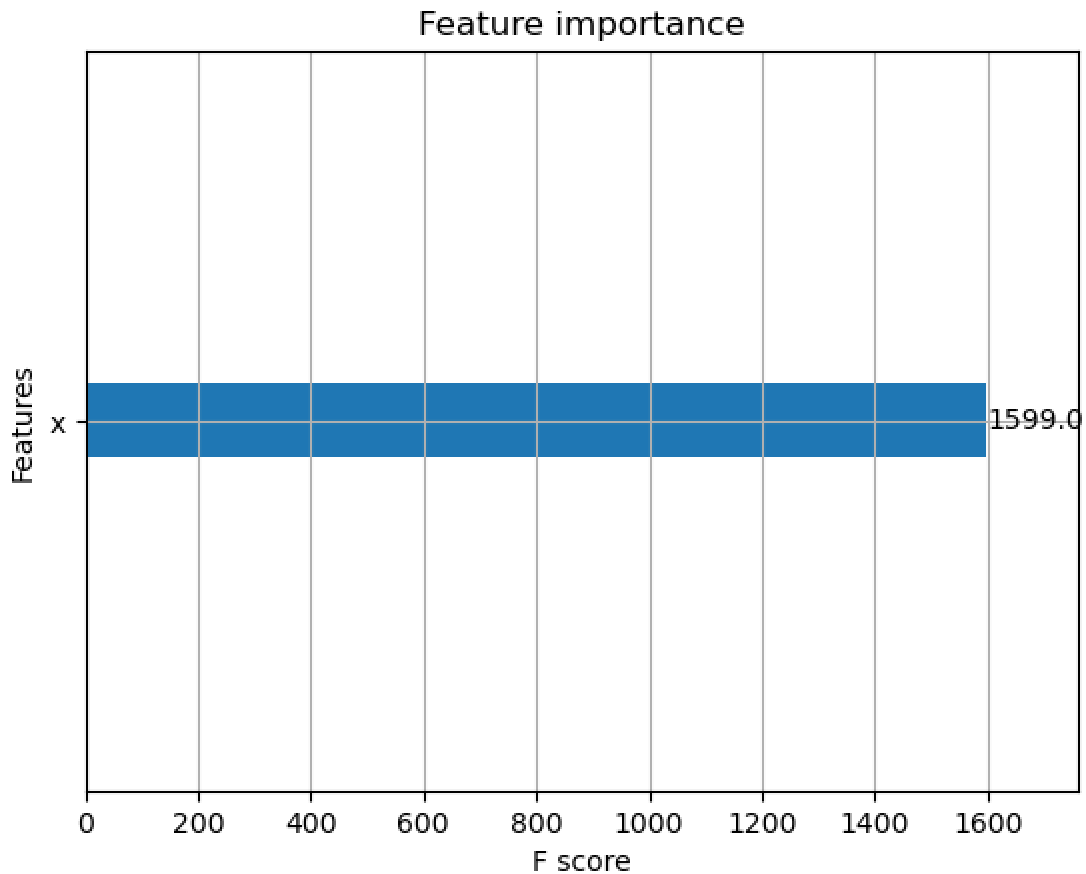
In [16]:

```
plt.scatter(x=training_rounds,y=eval_result['validation_0']['rmse'],label='Training Error')
plt.scatter(x=training_rounds,y=eval_result['validation_1']['rmse'],label='Validation Error')
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('RMSE')
plt.title('Training Vs Validation Error')
```

```
plt.legend()
plt.show()
```



In [17]:
```
xgb.plot_importance(regressor)
plt.show()
```



Validation Dataset Compare Actual and Predicted

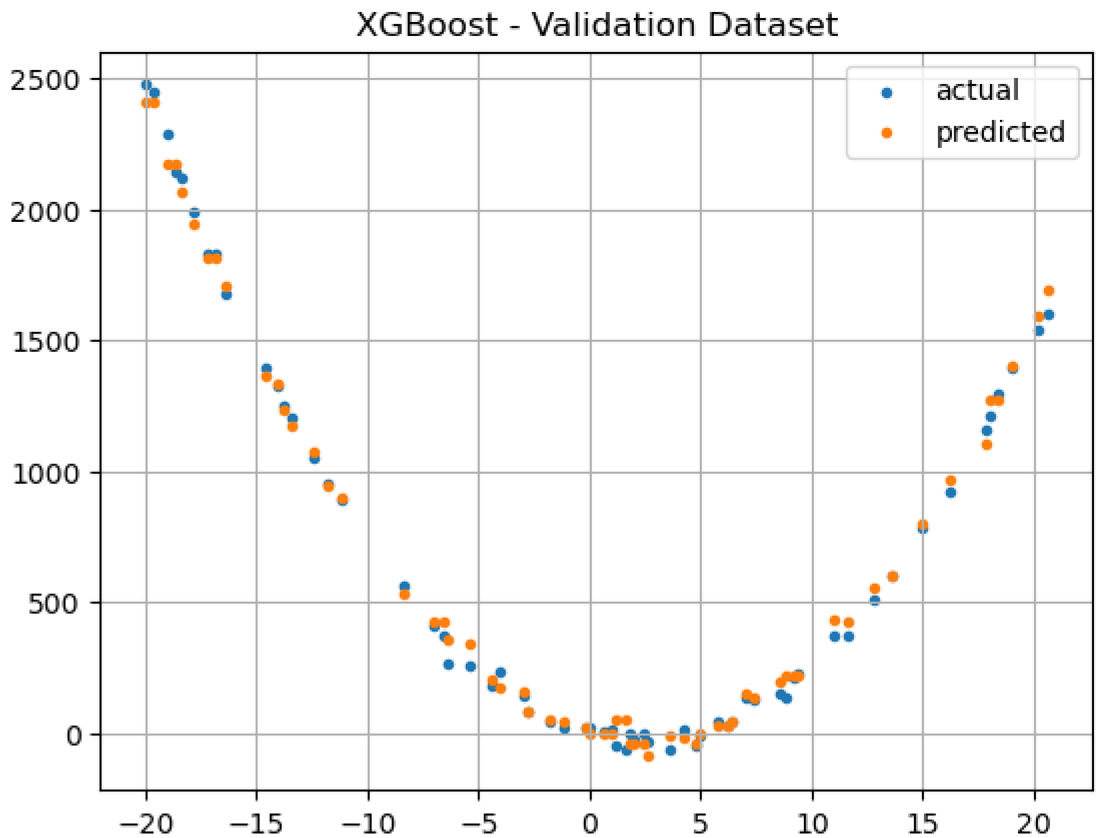
In [18]:
```
result = regressor.predict(X_validation)
```

In [19]:
```
result[:5]
```

Out[19]:
```
array([1815.7225 ,   46.51924, 1815.7225 , 1400.9963 ,  156.46053],
      dtype=float32)
```

In [20]:
```
plt.title('XGBoost - Validation Dataset')
plt.scatter(df_validation.x,df_validation.y,label='actual',marker='.')
plt.scatter(df_validation.x,result,label='predicted',marker='.')
```

```
plt.grid(True)
plt.legend()
plt.show()
```

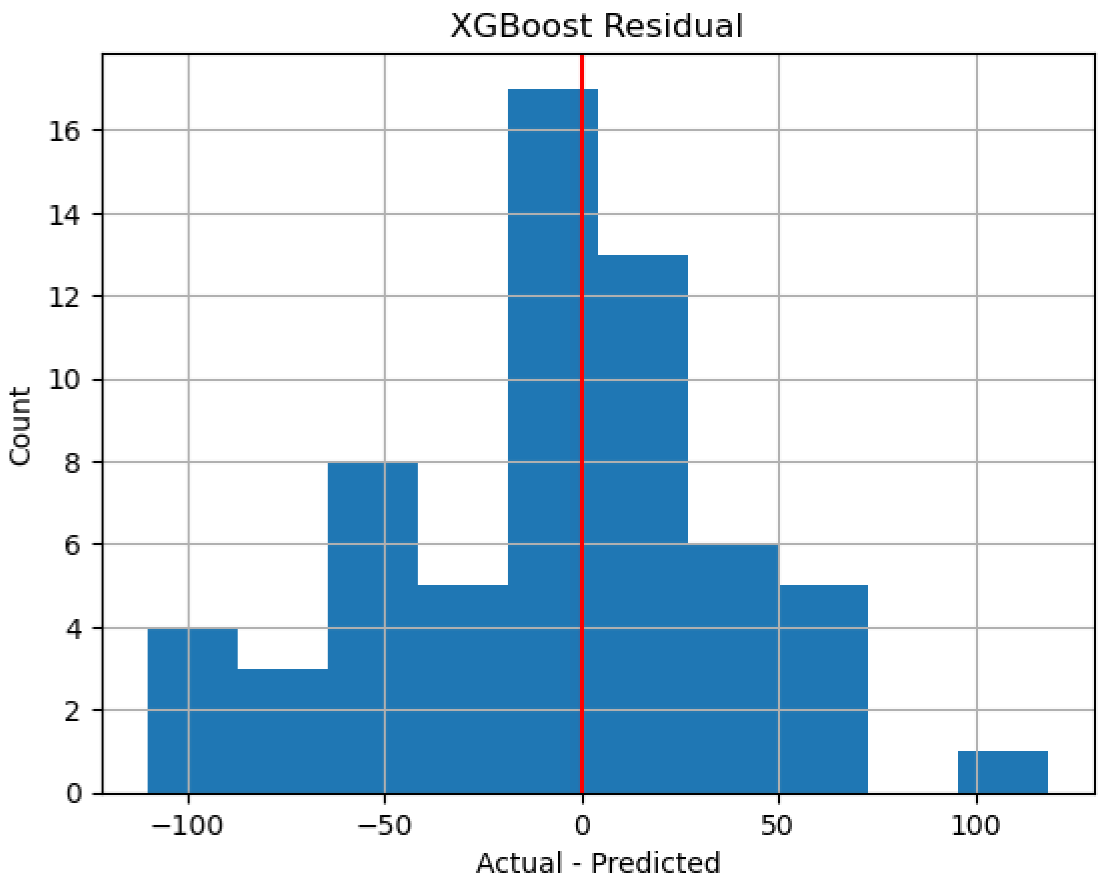


In [21]:
```
# RMSE Metrics
print('XGBoost Algorithm Metrics')
mse = mean_squared_error(df_validation.y,result)
print(" Mean Squared Error: {0:.2f}".format(mse))
print(" Root Mean Square Error: {0:.2f}".format(mse**.5))
```

```
XGBoost Algorithm Metrics
 Mean Squared Error: 2060.76
 Root Mean Square Error: 45.40
```

In [22]:
```
# Residual
# Over prediction and Under Prediction needs to be balanced
```

```
# Training Data Residuals
residuals = df_validation.y - result
plt.hist(residuals)
plt.grid(True)
plt.xlabel('Actual - Predicted')
plt.ylabel('Count')
plt.title('XGBoost Residual')
plt.axvline(color='r')
plt.show()
```

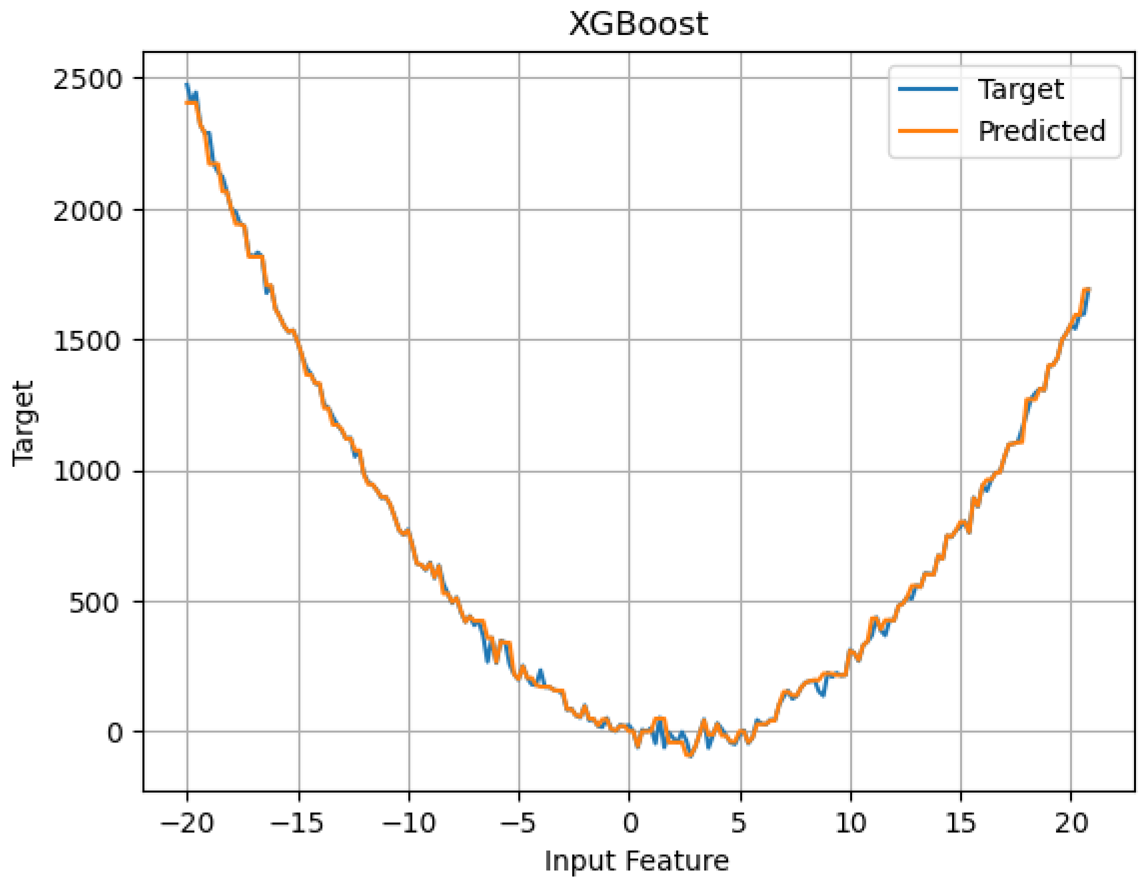


In [23]:
```
# Count number of values greater than zero and less than zero
value_counts = (residuals > 0).value_counts(sort=False)
```

```
print(' Under Estimation: {0}'.format(value_counts[True]))
print(' Over  Estimation: {0}'.format(value_counts[False]))
```

```
 Under Estimation: 27
 Over  Estimation: 35
```

In [24]:
```
# Plot for entire dataset
plt.plot(df.x,df.y,label='Target')
plt.plot(df.x,regressor.predict(df[['x']]) ,label='Predicted')
plt.grid(True)
plt.xlabel('Input Feature')
plt.ylabel('Target')
plt.legend()
plt.title('XGBoost')
plt.show()
```



Linear Regression Algorithm

In [25]:
```
lin_regressor = LinearRegression()
```

In [68]:
```
#DWB#  For kicks and giggles, to match our
#DWB#+    regressor = xgb.XGBRegressor()
#DWB#+ string output
print(str(lin_regressor)) #  didn't do much new, in comparison
                          #+ to the one that can have a scroll
                          #+ bar
```

```
LinearRegression()
```

In [26]:
```
lin_regressor.fit(X_train,y_train)
```

Out[26]:
```
▾ LinearRegression
LinearRegression()
```

Compare Weights assigned by Linear Regression.

Original Function: 5*x2 -23*x + 47 + some noise

Linear Regression Function: -15.08 * x + 709.86

Linear Regression Coefficients and Intercepts are not close to actual

In [27]:
```
lin_regressor.coef_
```

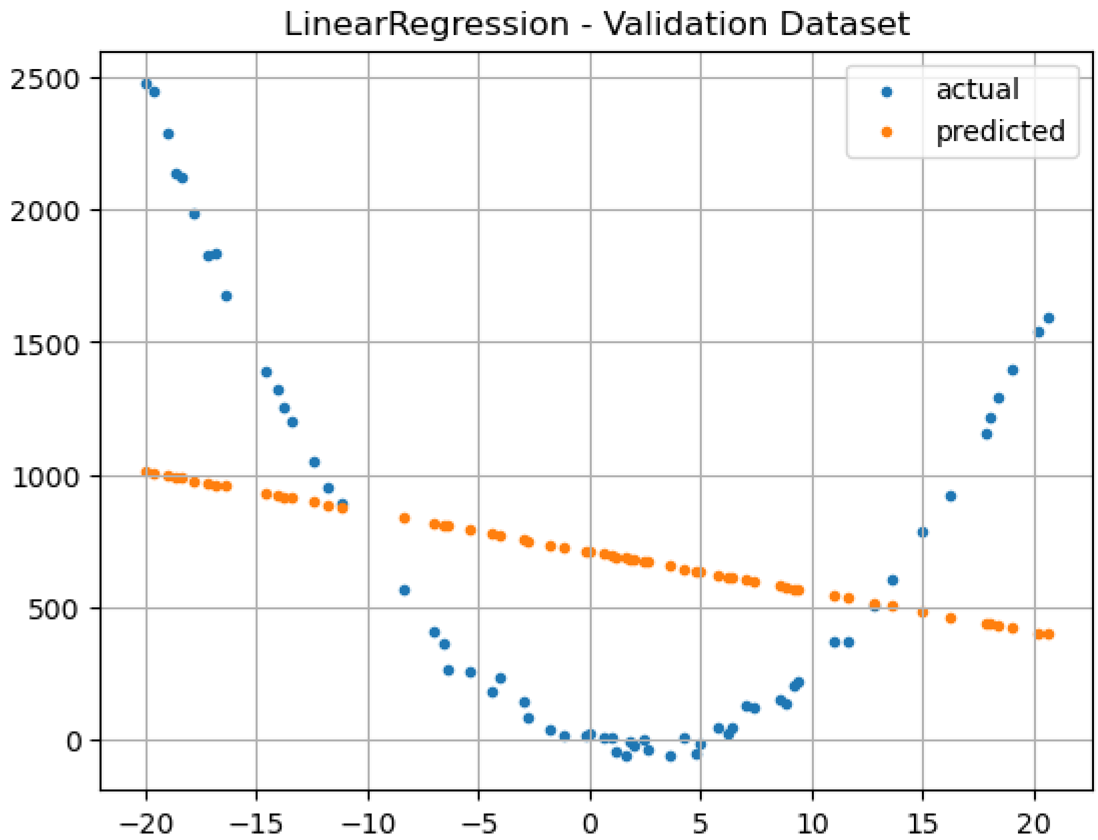
Out[27]:
```
array([-15.07800272])
```

In [28]:
```
lin_regressor.intercept_
```

Out[28]:
```
709.8622001903116
```

In [29]:
```
result = lin_regressor.predict(df_validation[['x']])
```

In [30]:
```
plt.title('LinearRegression - Validation Dataset')
plt.scatter(df_validation.x,df_validation.y,label='actual',marker='.')
plt.scatter(df_validation.x,result,label='predicted',marker='.')
plt.grid(True)
plt.legend()
plt.show()
```

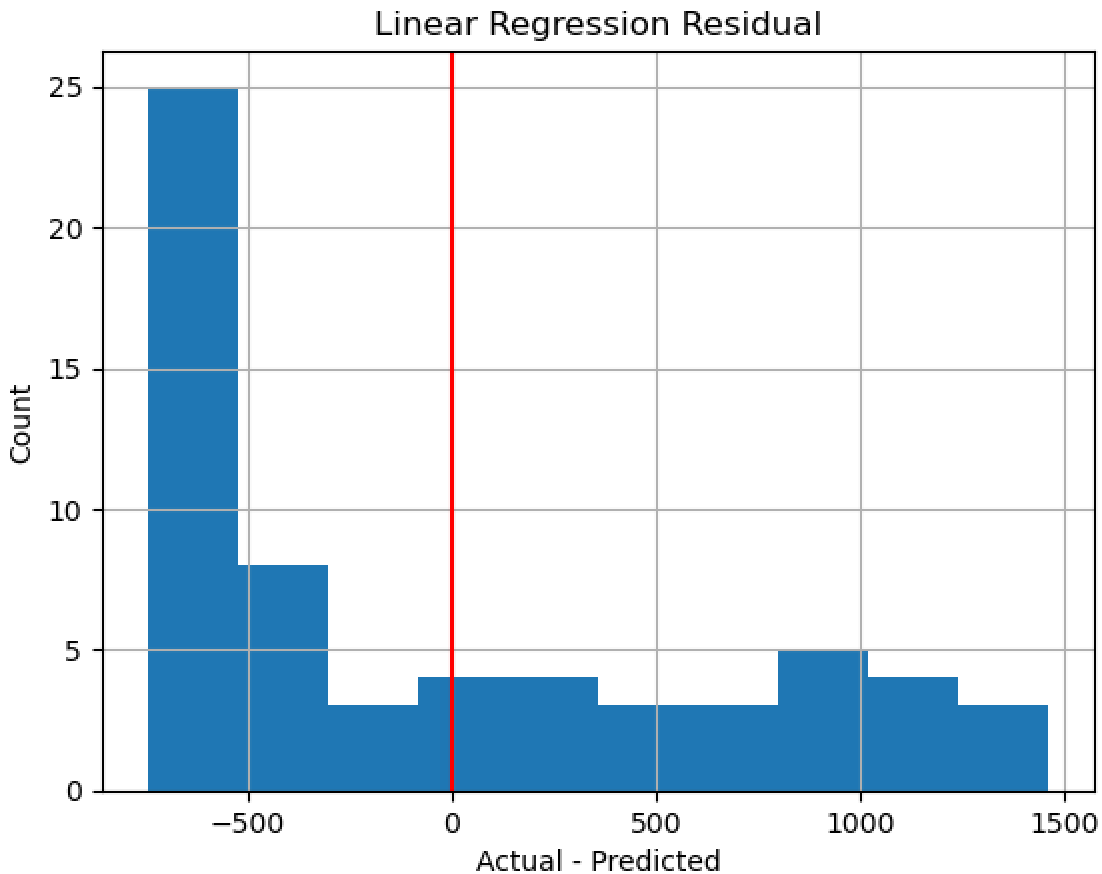


In [31]:
```
# RMSE Metrics
print('Linear Regression Metrics')
mse = mean_squared_error(df_validation.y,result)
print(" Mean Squared Error: {0:.2f}".format(mse))
print(" Root Mean Square Error: {0:.2f}".format(mse**.5))
```

```
Linear Regression Metrics
 Mean Squared Error: 488269.59
 Root Mean Square Error: 698.76
```

In [32]:
```
# Residual
# Over prediction and Under Prediction needs to be balanced
# Training Data Residuals
residuals = df_validation.y - result
plt.hist(residuals)
plt.grid(True)
```

```
plt.xlabel('Actual - Predicted')
plt.ylabel('Count')
plt.title('Linear Regression Residual')
plt.axvline(color='r')
plt.show()
```



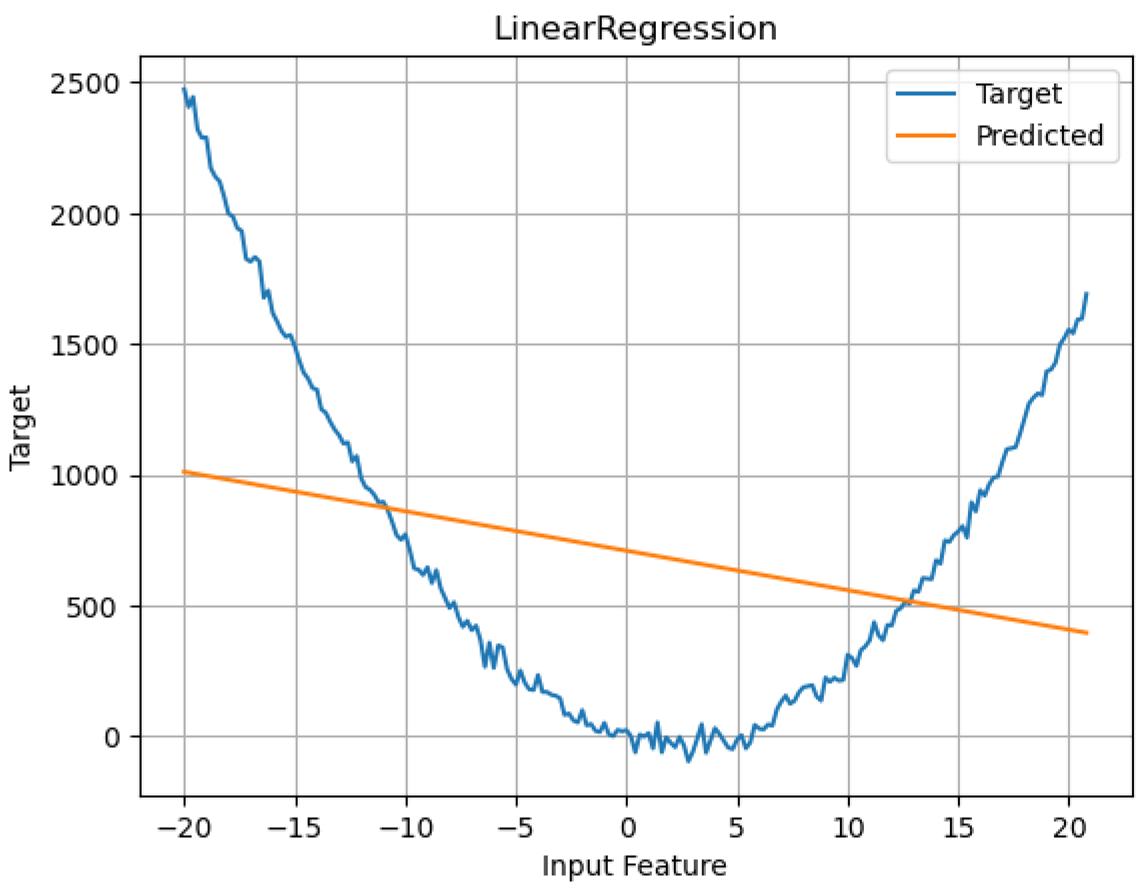
In [33]:

```
# Count number of values greater than zero and less than zero
value_counts = (residuals > 0).value_counts(sort=False)

print(' Under Estimation: {0}'.format(value_counts[True]))
print(' Over  Estimation: {0}'.format(value_counts[False]))
```

```
 Under Estimation: 25
 Over  Estimation: 37
```

In [34]:
```
# Plot for entire dataset
plt.plot(df.x,df.y,label='Target')
plt.plot(df.x,lin_regressor.predict(df[['x']]) ,label='Predicted')
plt.grid(True)
plt.xlabel('Input Feature')
plt.ylabel('Target')
plt.legend()
plt.title('LinearRegression')
plt.show()
```



Linear Regression is showing clear symptoms of under-fitting

Input Features are not sufficient to capture complex relationship

Your Turn

You can correct this under-fitting issue by adding relavant features.

1. What feature will you add and why?
2. Complete the code and Test
3. What performance do you see now?

In [ ]:
```
# Specify the column names as the file does not have column header
df_train = pd.read_csv(train_file,names=['y','x'])
df_validation = pd.read_csv(validation_file,names=['y','x'])
df = pd.read_csv('quadratic_all.csv')
```

Add new features

Place holder to add new features to df_train, df_validation and df if you need help, scroll down to see the answer Add your code

In [45]:
```
# Add your new features
#DWB# cf. 3.28 (Linear/Non-Linear)
#DWB#  I'm adding the feature, x^2, because
#DWB#+ we're trying to predict a quadratic
#DWB#+ function using linear regression, and
#DWB#+ the degree coefficients of a
#DWB#+ polynomial function are linearly
#DWB#+ independent.

def pow_2_func (x):
    return x**2

#repeat_x_ser = pd.Series(np.arange(-20, 21, 0.2))
#x_sq_ser = repeat_x_ser.map(pow_2_func)
#df['x_sq']            = x_sq_ser
#  #  worked, but feels like we're doing too much extra
#  #+ for the other dataframes, I don't know how I'd work it
#  #+ without re-doing everything for them, too.

#df_train['x_sq']      = df_train['x'].map(lambda x: df_train['x']**2)
```

```
#  # didn't work at all, and I'm not sure what is happening here

#df_validation['x_sq'] = df_validation.index.map(lambda x: df_train[x]**2)
#  # didn't work (gave index**2)

# I had the feature correct, so I just looked down for the syntax
df['x_sq']             = df['x']**2
df_train['x_sq']       = df_train['x']**2
df_validation['x_sq']  = df_validation['x']**2

print(  (f"\n  Inspecting df:\n"
         f"df.head() = \n{str(df.head())}\n"
        )
     )
print(  (f"\n  Inspecting df_train:\n"
         f"df_train.head() = \n{str(df_train.head())}\n"
        )
     )
print(  (f"\n  Inspecting df_validation:\n"
         f"df_validation.head() = \n{str(df_validation.head())}\n"
        )
     )
```

```
  Inspecting df:
df.head() =
      x            y    x_sq
0 -20.0  2473.236825  400.00
1 -19.8  2405.673895  392.04
2 -19.6  2444.523136  384.16
3 -19.4  2320.437236  376.36
4 -19.2  2288.088295  368.64


  Inspecting df_train:
df_train.head() =
             y     x    x_sq
0   343.968005  10.8  116.64
1  1585.894405 -15.8  249.64
2  1497.303317  19.6  384.16
3   769.909912 -10.4  108.16
4  1173.230755 -13.2  174.24


  Inspecting df_validation:
df_validation.head() =
             y     x    x_sq
0  1824.856344 -17.2  295.84
1    16.997917  -1.2    1.44
2  1832.141730 -16.8  282.24
3  1395.206684  19.0  361.00
4   145.840543  -3.0    9.00
```

In [46]:

```
X_train = df_train.iloc[:,1:] # Features: 1st column onwards
y_train = df_train.iloc[:,0].ravel() # Target: 0th column

X_validation = df_validation.iloc[:,1:]
y_validation = df_validation.iloc[:,0].ravel()
```

In [47]:

```
lin_regressor.fit(X_train,y_train)
```

Out[47]:
```
▾ LinearRegression
LinearRegression()
```

Original Function: -23*x* + 5*x***2 + 47 + some noise (rewritten with x term first)

In [48]:
```
lin_regressor.coef_
```

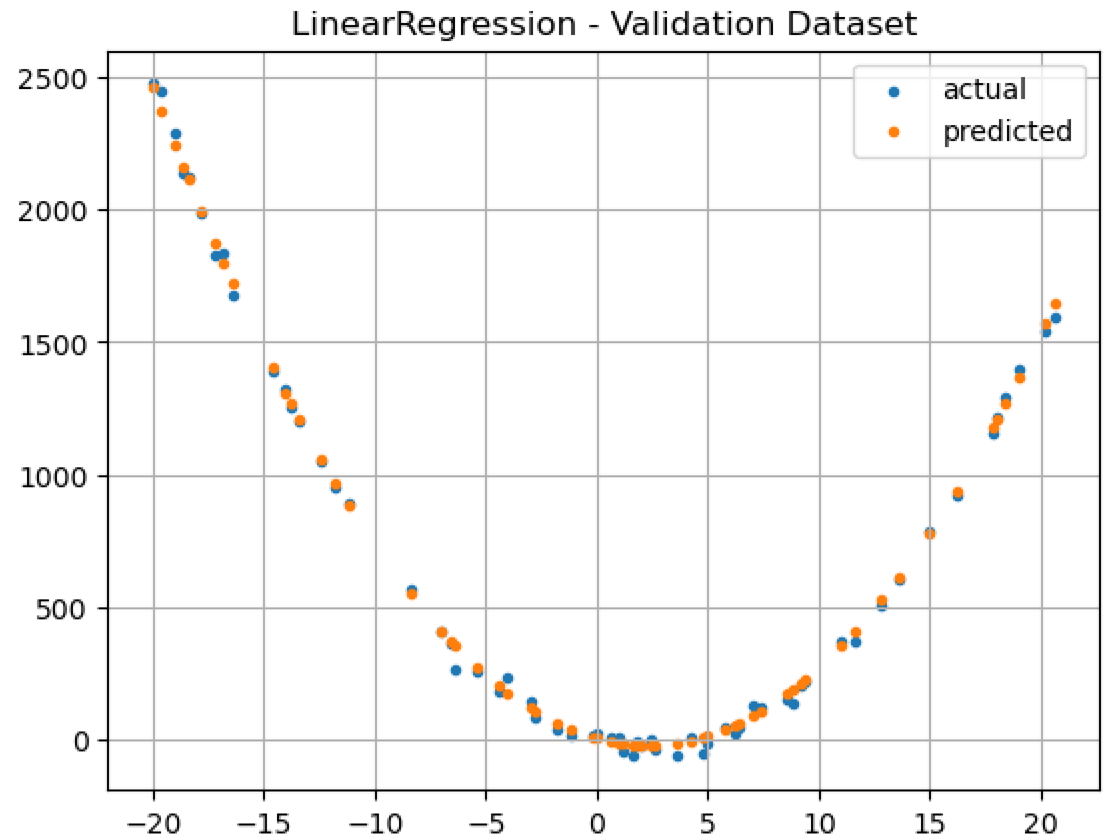
Out[48]:
```
array([-22.98286274,   4.98161803])
```

In [49]:
```
lin_regressor.intercept_
```

Out[49]:
```
5.86810755251679
```

In [50]:
```
result = lin_regressor.predict(X_validation)
```

In [51]:
```
plt.title('LinearRegression - Validation Dataset')
plt.scatter(df_validation.x,df_validation.y,label='actual',marker='.')
plt.scatter(df_validation.x,result,label='predicted',marker='.')
plt.grid(True)
plt.legend()
plt.show()
```



In [52]:
```
# RMSE Metrics
print('Linear Regression Metrics')
mse = mean_squared_error(df_validation.y,result)
print(" Mean Squared Error: {0:.2f}".format(mse))
print(" Root Mean Square Error: {0:.2f}".format(mse**.5))

print("***You should see an RMSE score of 30.45 or less")
```

```
Linear Regression Metrics
 Mean Squared Error: 927.22
 Root Mean Square Error: 30.45
***You should see an RMSE score of 30.45 or less
```

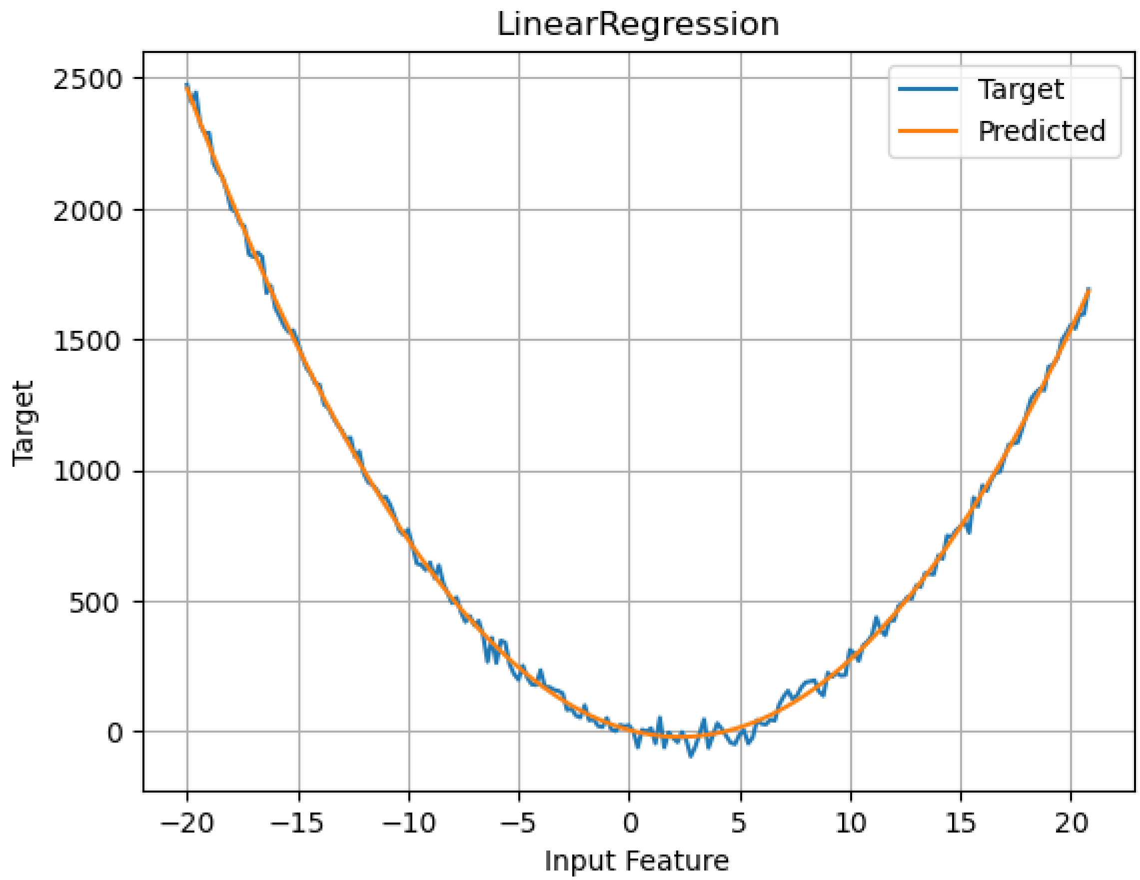
In [53]:
```
df.head()
```

Out[53]:

	x	y	x_sq
0	-20.0	2473.236825	400.00
1	-19.8	2405.673895	392.04
2	-19.6	2444.523136	384.16
3	-19.4	2320.437236	376.36
4	-19.2	2288.088295	368.64

In [56]:

```
# Plot for entire dataset
plt.plot(df.x,df.y,label='Target')
#DWB#Changing to match my column names#
#DWB#plt.plot(df.x,lin_regressor.predict(df[['x','x2']]) ,label='Predicted')
plt.plot(df.x, lin_regressor.predict(df[['x','x_sq']]),
         label='Predicted')
plt.grid(True)
plt.xlabel('Input Feature')
plt.ylabel('Target')
plt.legend()
plt.title('LinearRegression')
plt.show()
```



Solution for under-fitting

add a new X**2 term to the dataframe

syntax:

df_train['x2'] = df_train['x']**2

df_validation['x2'] = df_validation['x']**2

df['x2'] = df['x']**2

Tree Based Algorithms have a lower bound and upper bound for predicted values

In [57]:
```
# True Function
def quad_func (x):
    return 5*x**2 -23*x + 47
```

In [59]:
```
# X is outside range of training samples
# New Feature: Adding X^2 term

X = np.array([-100,-25,25,1000,5000])
y = quad_func(X)
#DWB#Changing to match my column names#
#df_tmp = pd.DataFrame({'x':X,'y':y,'x2':X**2})
df_tmp = pd.DataFrame({'x':X, 'y':y, 'x_sq':X**2})
df_tmp['xgboost']=regressor.predict(df_tmp[['x']])
#DWB#Changing to match my column names#
#df_tmp['linear']=lin_regressor.predict(df_tmp[['x','x2']])
df_tmp['linear']=lin_regressor.predict(df_tmp[['x', 'x_sq']])
```

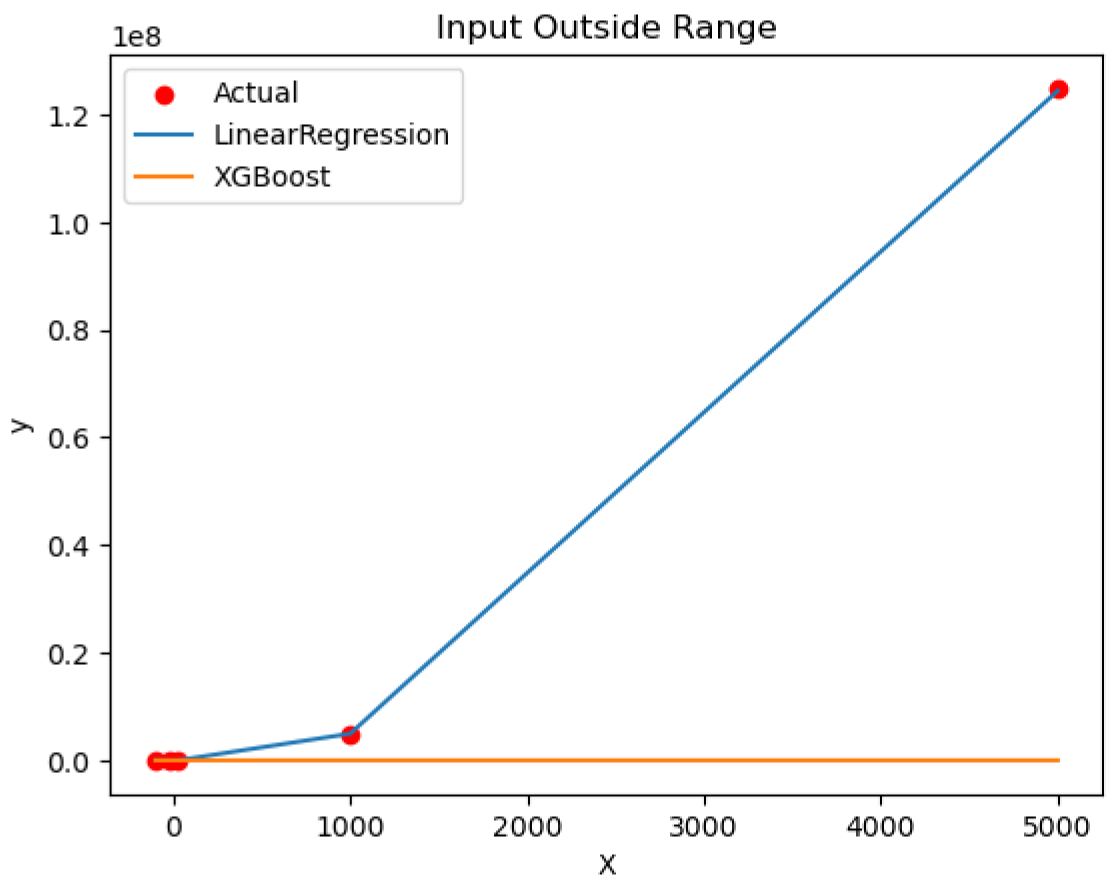
In [60]:
```
df_tmp
```

Out[60]:

	x	y	x_sq	xgboost	linear
0	-100	52347	10000	2404.638428	5.212033e+04
1	-25	3747	625	2404.638428	3.693951e+03
2	25	2597	625	1689.587646	2.544808e+03
3	1000	4977047	1000000	1689.587646	4.958641e+06
4	5000	124885047	25000000	1689.587646	1.244255e+08

In [61]:
```
plt.scatter(df_tmp.x,df_tmp.y,label='Actual',color='r')
plt.plot(df_tmp.x,df_tmp.linear,label='LinearRegression')
plt.plot(df_tmp.x,df_tmp.xgboost,label='XGBoost')
plt.legend()
plt.xlabel('X')
plt.ylabel('y')
```

```
plt.title('Input Outside Range')
plt.show()
```



In [63]:

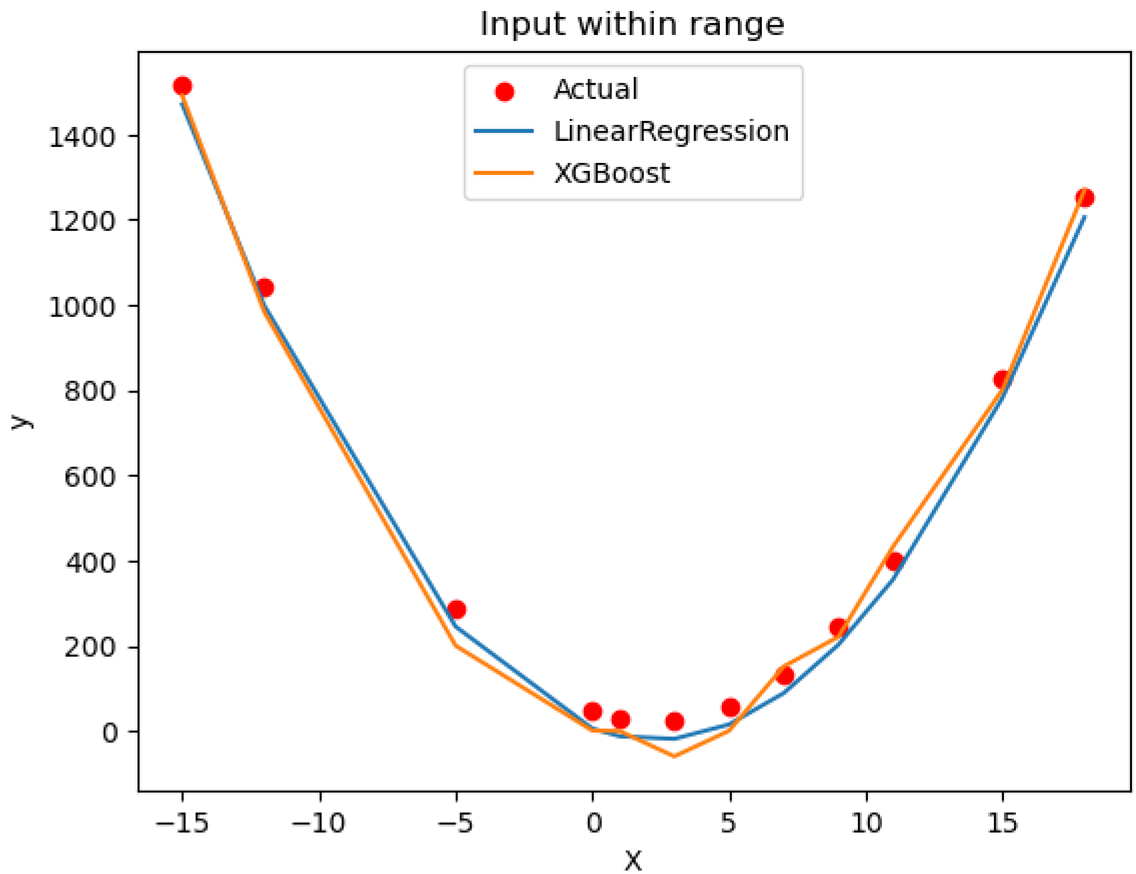
```
# X is inside range of training samples
X = np.array([-15,-12,-5,0,1,3,5,7,9,11,15,18])
y = quad_func(X)
#DWB#Changing to match my column names#
#df_tmp = pd.DataFrame({'x':X,'y':y,'x2':X**2})
df_tmp = pd.DataFrame({'x':X, 'y':y, 'x_sq':X**2})
df_tmp['xgboost'] = regressor.predict(df_tmp[['x']])
#DWB#Changing to match my column names#
#df_tmp['linear']=lin_regressor.predict(df_tmp[['x','x2']])
df_tmp['linear'] = lin_regressor.predict(df_tmp[['x', 'x_sq']])
```

In [64]:
```
df_tmp
```

Out[64]:

	x	y	x_sq	xgboost	linear
0	-15	1517	225	1491.868652	1471.475106
1	-12	1043	144	983.951050	999.015457
2	-5	287	25	200.439957	245.322872
3	0	47	0	1.164244	5.868108
4	1	29	1	0.122412	-12.133137
5	3	23	9	-59.463448	-18.245918
6	5	57	25	0.623751	15.494245
7	7	131	49	151.693665	89.087352
8	9	245	81	221.317764	202.533404
9	11	399	121	432.898956	355.832399
10	15	827	225	801.118469	781.989224
11	18	1253	324	1269.712769	1206.220821

In [65]:
```
# XGBoost Predictions have an upper bound and lower bound
# Linear Regression Extrapolates
plt.scatter(df_tmp.x,df_tmp.y,label='Actual',color='r')
plt.plot(df_tmp.x,df_tmp.linear,label='LinearRegression')
plt.plot(df_tmp.x,df_tmp.xgboost,label='XGBoost')
plt.legend()
plt.xlabel('X')
plt.ylabel('y')
plt.title('Input within range')
plt.show()
```


Input within range
Actual
LinearRegression
XGBoost
1400
1200
1000
800
600
400
200
0
y
−15
−10
−5
0
5
10
15
X

Summary

1. In this exercise, we compared performance of XGBoost model and Linear Regression on a quadratic dataset
2. The relationship between input feature and target was non-linear.
3. XGBoost handled it pretty well; whereas, linear regression was under-fitting
4. To correct the issue, we had to add additional features for linear regression
5. With this change, linear regression performed much better

XGBoost can detect patterns involving non-linear relationship; whereas, algorithms like linear regression may need complex feature engineering