Train a model with bike rental data using XGBoost algorithm

Model is trained with XGBoost installed in notebook instance

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

In [1]:
```
# Install xgboost in notebook instance.
#### Command to install xgboost
!pip install xgboost
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.co
m
Collecting xgboost
  Downloading xgboost-1.7.6-py3-none-manylinux2014_x86_64.whl (200.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.3/200.3 MB 3.1 MB/s eta 0:00:000
0:0100:01
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/python3/lib/
python3.10/site-packages (from xgboost) (1.22.3)
Requirement already satisfied: scipy in /home/ec2-user/anaconda3/envs/python3/lib/
python3.10/site-packages (from xgboost) (1.10.1)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.6
```

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
```

In [3]:
```
column_list_file = 'bike_train_column_list.txt'
train_file = 'bike_train.csv'
validation_file = 'bike_validation.csv'
test_file = 'bike_test.csv'
```

In [4]:
```
columns = ''
with open(column_list_file,'r') as f:
    columns = f.read().split(',')
```

In [5]:
```
columns
```

Out[5]:
```
['count',
 'season',
 'holiday',
 'workingday',
 'weather',
 'temp',
 'atemp',
 'humidity',
 'windspeed',
 'year',
 'month',
 'day',
 'dayofweek',
 'hour']
```

In [6]:
```
# Specify the column names as the file does not have column header
df_train = pd.read_csv(train_file,names=columns)
df_validation = pd.read_csv(validation_file,names=columns)
```

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

Out[7]:

	count	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	mon
0	87	3	0	0	2	26.24	30.305	73	7.0015	2011	
1	248	3	0	1	1	32.80	34.850	33	7.0015	2012	
2	334	4	0	0	1	15.58	19.695	40	11.0014	2011	
3	623	3	0	1	1	32.80	37.880	55	12.9980	2012	
4	70	2	0	1	1	13.94	17.425	76	7.0015	2011	

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

Out[8]:

	count	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	mon
0	443	3	0	1	2	28.70	33.335	79	12.9980	2011	
1	387	2	0	0	1	32.80	37.880	55	12.9980	2011	
2	2	1	0	1	1	14.76	16.665	40	19.9995	2011	
3	48	1	0	1	1	9.02	9.090	47	36.9974	2011	
4	55	4	0	0	1	10.66	15.150	87	0.0000	2011	

In [9]:
```
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 [10]:
```
# XGBoost Training Parameter Reference:
#   https://github.com/dmlc/xgboost/blob/master/doc/parameter.md
```

```
#regressor = xgb.XGBRegressor(max_depth=5,eta=0.1,subsample=0.7,num_round=150)
#DWB# Note. 5 is the max depth - how many nodes any tree can go down.
#DWB#       150 is the number of trees
regressor = xgb.XGBRegressor(max_depth=5,n_estimators=150)
```

In [11]:
```
regressor
```

Out[11]:
```
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,
```

In [12]:
```
regressor.fit(X_train,y_train, eval_set = [(X_train, y_train), (X_validation, y_val
```

```
[0]    validation_0-rmse:200.21253    validation_1-rmse:198.50750
[1]    validation_0-rmse:158.44940    validation_1-rmse:156.82238
[2]    validation_0-rmse:130.70633    validation_1-rmse:129.74683
[3]    validation_0-rmse:113.91983    validation_1-rmse:113.36164
[4]    validation_0-rmse:97.49929     validation_1-rmse:97.96390
[5]    validation_0-rmse:84.68191     validation_1-rmse:86.42600
[6]    validation_0-rmse:75.20273     validation_1-rmse:77.72003
[7]    validation_0-rmse:71.41857     validation_1-rmse:74.25260
[8]    validation_0-rmse:64.23005     validation_1-rmse:67.80524
[9]    validation_0-rmse:61.87001     validation_1-rmse:65.64181
[10]   validation_0-rmse:60.00386     validation_1-rmse:63.93544
[11]   validation_0-rmse:57.38252     validation_1-rmse:61.66847
[12]   validation_0-rmse:55.40470     validation_1-rmse:59.70897
[13]   validation_0-rmse:53.46252     validation_1-rmse:58.07008
[14]   validation_0-rmse:50.16573     validation_1-rmse:55.00782
[15]   validation_0-rmse:49.58626     validation_1-rmse:54.48380
[16]   validation_0-rmse:49.10568     validation_1-rmse:53.95741
[17]   validation_0-rmse:46.31593     validation_1-rmse:51.42871
[18]   validation_0-rmse:45.25657     validation_1-rmse:50.72695
[19]   validation_0-rmse:44.66090     validation_1-rmse:50.27214
[20]   validation_0-rmse:43.69099     validation_1-rmse:49.51648
[21]   validation_0-rmse:43.06359     validation_1-rmse:49.09360
[22]   validation_0-rmse:42.79748     validation_1-rmse:48.97205
[23]   validation_0-rmse:42.22981     validation_1-rmse:48.57886
[24]   validation_0-rmse:42.07683     validation_1-rmse:48.46537
[25]   validation_0-rmse:41.79092     validation_1-rmse:48.40560
[26]   validation_0-rmse:40.22368     validation_1-rmse:47.06446
[27]   validation_0-rmse:40.02085     validation_1-rmse:46.96824
[28]   validation_0-rmse:39.61362     validation_1-rmse:46.72801
[29]   validation_0-rmse:39.32625     validation_1-rmse:46.58315
[30]   validation_0-rmse:39.17388     validation_1-rmse:46.47925
[31]   validation_0-rmse:38.45699     validation_1-rmse:46.02084
[32]   validation_0-rmse:38.15526     validation_1-rmse:45.90458
[33]   validation_0-rmse:37.93132     validation_1-rmse:45.77935
[34]   validation_0-rmse:37.61136     validation_1-rmse:45.62779
[35]   validation_0-rmse:37.41721     validation_1-rmse:45.47065
[36]   validation_0-rmse:37.33951     validation_1-rmse:45.43344
[37]   validation_0-rmse:36.62579     validation_1-rmse:44.90997
[38]   validation_0-rmse:36.42528     validation_1-rmse:44.84257
[39]   validation_0-rmse:36.06161     validation_1-rmse:44.76581
[40]   validation_0-rmse:35.93213     validation_1-rmse:44.74343
[41]   validation_0-rmse:35.88741     validation_1-rmse:44.71638
[42]   validation_0-rmse:35.50912     validation_1-rmse:44.60798
[43]   validation_0-rmse:35.11191     validation_1-rmse:44.35004
[44]   validation_0-rmse:35.00737     validation_1-rmse:44.29069
[45]   validation_0-rmse:34.78141     validation_1-rmse:44.23079
[46]   validation_0-rmse:34.73855     validation_1-rmse:44.23772
[47]   validation_0-rmse:34.65160     validation_1-rmse:44.17048
[48]   validation_0-rmse:34.35805     validation_1-rmse:44.09884
[49]   validation_0-rmse:34.19989     validation_1-rmse:44.09992
[50]   validation_0-rmse:34.14433     validation_1-rmse:44.11222
[51]   validation_0-rmse:33.91166     validation_1-rmse:44.04286
[52]   validation_0-rmse:33.74896     validation_1-rmse:44.04637
[53]   validation_0-rmse:33.58468     validation_1-rmse:43.98733
[54]   validation_0-rmse:33.51054     validation_1-rmse:43.98548
[55]   validation_0-rmse:33.31730     validation_1-rmse:44.02941
```

```
[56]    validation_0-rmse:33.23237      validation_1-rmse:44.02083
[57]    validation_0-rmse:33.19658      validation_1-rmse:44.00654
[58]    validation_0-rmse:33.07861      validation_1-rmse:43.99136
[59]    validation_0-rmse:32.86804      validation_1-rmse:43.89020
[60]    validation_0-rmse:32.72255      validation_1-rmse:43.82265
[61]    validation_0-rmse:32.31547      validation_1-rmse:43.49294
[62]    validation_0-rmse:32.29336      validation_1-rmse:43.46725
[63]    validation_0-rmse:32.07836      validation_1-rmse:43.38681
[64]    validation_0-rmse:31.90100      validation_1-rmse:43.32034
[65]    validation_0-rmse:31.86057      validation_1-rmse:43.32324
[66]    validation_0-rmse:31.70479      validation_1-rmse:43.18750
[67]    validation_0-rmse:31.59029      validation_1-rmse:43.19710
[68]    validation_0-rmse:31.39416      validation_1-rmse:43.18815
[69]    validation_0-rmse:31.12865      validation_1-rmse:43.09848
[70]    validation_0-rmse:31.07494      validation_1-rmse:43.09148
[71]    validation_0-rmse:31.00557      validation_1-rmse:43.09359
[72]    validation_0-rmse:30.87338      validation_1-rmse:43.00188
[73]    validation_0-rmse:30.56048      validation_1-rmse:42.88081
[74]    validation_0-rmse:30.48123      validation_1-rmse:42.82766
[75]    validation_0-rmse:30.32723      validation_1-rmse:42.81791
[76]    validation_0-rmse:30.23960      validation_1-rmse:42.79342
[77]    validation_0-rmse:30.20228      validation_1-rmse:42.78964
[78]    validation_0-rmse:29.99587      validation_1-rmse:42.77126
[79]    validation_0-rmse:29.75669      validation_1-rmse:42.64365
[80]    validation_0-rmse:29.66938      validation_1-rmse:42.59289
[81]    validation_0-rmse:29.62375      validation_1-rmse:42.59439
[82]    validation_0-rmse:29.50415      validation_1-rmse:42.54275
[83]    validation_0-rmse:29.42191      validation_1-rmse:42.49808
[84]    validation_0-rmse:29.36916      validation_1-rmse:42.49365
[85]    validation_0-rmse:29.27202      validation_1-rmse:42.43840
[86]    validation_0-rmse:29.13929      validation_1-rmse:42.42627
[87]    validation_0-rmse:29.08375      validation_1-rmse:42.40068
[88]    validation_0-rmse:29.02746      validation_1-rmse:42.39586
[89]    validation_0-rmse:28.88187      validation_1-rmse:42.35033
[90]    validation_0-rmse:28.75088      validation_1-rmse:42.28242
[91]    validation_0-rmse:28.64258      validation_1-rmse:42.29661
[92]    validation_0-rmse:28.51315      validation_1-rmse:42.24072
[93]    validation_0-rmse:28.43576      validation_1-rmse:42.21895
[94]    validation_0-rmse:28.34130      validation_1-rmse:42.23728
[95]    validation_0-rmse:28.24365      validation_1-rmse:42.19883
[96]    validation_0-rmse:28.14093      validation_1-rmse:42.15161
[97]    validation_0-rmse:28.08991      validation_1-rmse:42.13850
[98]    validation_0-rmse:28.05213      validation_1-rmse:42.12963
[99]    validation_0-rmse:28.01991      validation_1-rmse:42.13131
[100]   validation_0-rmse:27.97190      validation_1-rmse:42.12578
[101]   validation_0-rmse:27.95016      validation_1-rmse:42.13953
[102]   validation_0-rmse:27.84402      validation_1-rmse:42.10889
[103]   validation_0-rmse:27.80204      validation_1-rmse:42.07974
[104]   validation_0-rmse:27.79910      validation_1-rmse:42.08551
[105]   validation_0-rmse:27.75465      validation_1-rmse:42.08251
[106]   validation_0-rmse:27.62789      validation_1-rmse:42.08866
[107]   validation_0-rmse:27.48359      validation_1-rmse:42.00880
[108]   validation_0-rmse:27.31545      validation_1-rmse:41.93489
[109]   validation_0-rmse:27.23284      validation_1-rmse:41.92242
[110]   validation_0-rmse:27.09928      validation_1-rmse:41.90098
[111]   validation_0-rmse:27.00993      validation_1-rmse:41.87125
```

```
[112]   validation_0-rmse:27.00165      validation_1-rmse:41.87185
[113]   validation_0-rmse:26.93704      validation_1-rmse:41.84040
[114]   validation_0-rmse:26.80303      validation_1-rmse:41.80264
[115]   validation_0-rmse:26.73493      validation_1-rmse:41.80721
[116]   validation_0-rmse:26.64247      validation_1-rmse:41.77972
[117]   validation_0-rmse:26.57824      validation_1-rmse:41.79133
[118]   validation_0-rmse:26.46564      validation_1-rmse:41.71431
[119]   validation_0-rmse:26.39325      validation_1-rmse:41.67695
[120]   validation_0-rmse:26.37278      validation_1-rmse:41.68118
[121]   validation_0-rmse:26.28281      validation_1-rmse:41.66746
[122]   validation_0-rmse:26.19166      validation_1-rmse:41.64710
[123]   validation_0-rmse:26.00082      validation_1-rmse:41.58643
[124]   validation_0-rmse:25.85092      validation_1-rmse:41.59726
[125]   validation_0-rmse:25.79182      validation_1-rmse:41.58297
[126]   validation_0-rmse:25.68885      validation_1-rmse:41.61713
[127]   validation_0-rmse:25.58066      validation_1-rmse:41.60252
[128]   validation_0-rmse:25.52994      validation_1-rmse:41.56733
[129]   validation_0-rmse:25.45470      validation_1-rmse:41.56769
[130]   validation_0-rmse:25.39346      validation_1-rmse:41.54351
[131]   validation_0-rmse:25.37824      validation_1-rmse:41.53871
[132]   validation_0-rmse:25.24823      validation_1-rmse:41.49660
[133]   validation_0-rmse:25.19883      validation_1-rmse:41.47297
[134]   validation_0-rmse:25.12123      validation_1-rmse:41.47223
[135]   validation_0-rmse:25.07267      validation_1-rmse:41.46833
[136]   validation_0-rmse:25.01747      validation_1-rmse:41.44283
[137]   validation_0-rmse:24.91436      validation_1-rmse:41.39120
[138]   validation_0-rmse:24.90468      validation_1-rmse:41.39363
[139]   validation_0-rmse:24.81354      validation_1-rmse:41.36809
[140]   validation_0-rmse:24.78586      validation_1-rmse:41.36993
[141]   validation_0-rmse:24.74158      validation_1-rmse:41.36148
[142]   validation_0-rmse:24.62951      validation_1-rmse:41.32304
[143]   validation_0-rmse:24.57401      validation_1-rmse:41.32358
[144]   validation_0-rmse:24.51089      validation_1-rmse:41.28476
[145]   validation_0-rmse:24.42649      validation_1-rmse:41.25937
[146]   validation_0-rmse:24.39769      validation_1-rmse:41.25983
[147]   validation_0-rmse:24.30934      validation_1-rmse:41.26104
[148]   validation_0-rmse:24.20446      validation_1-rmse:41.19978
[149]   validation_0-rmse:24.10522      validation_1-rmse:41.11284
```

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,
```

In [13]:

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

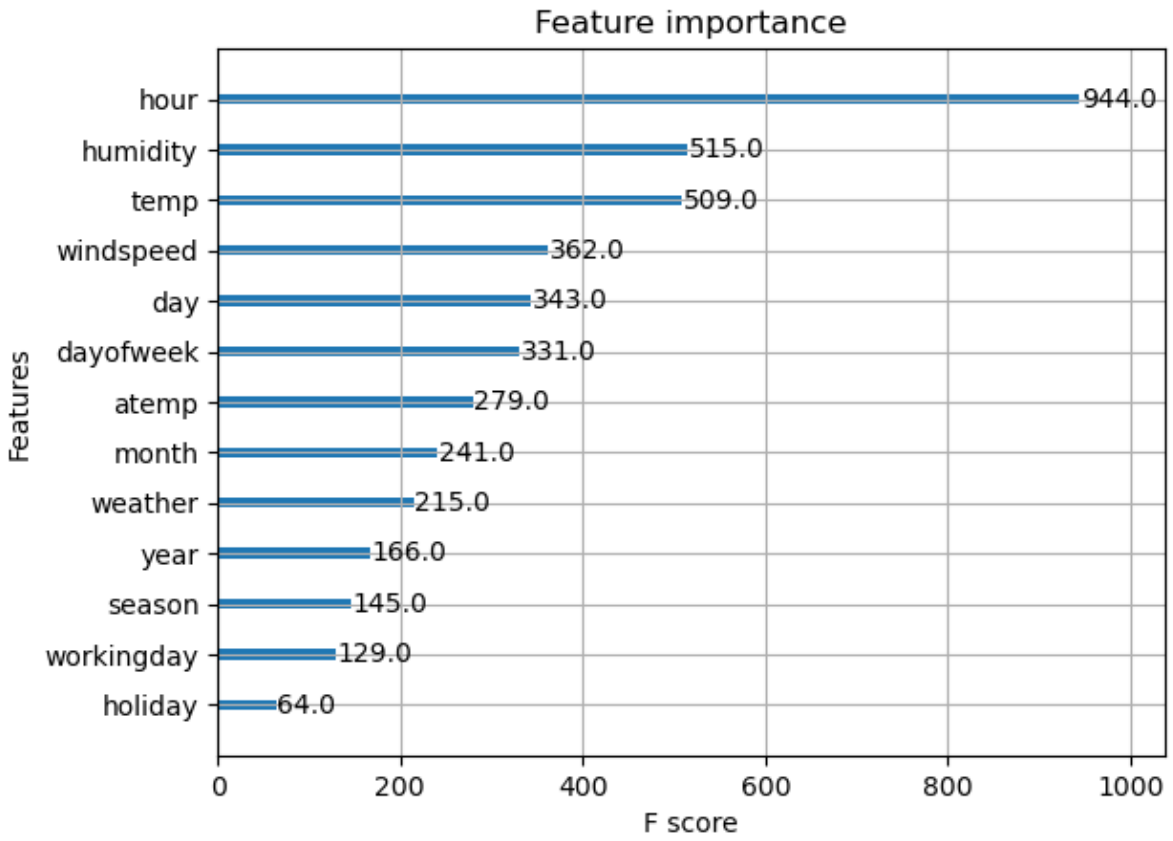
In [14]:
```
training_rounds = range(len(eval_result['validation_0']['rmse']))
```

In [15]:
```
print(training_rounds)
```

```
range(0, 150)
```

In [16]:
```
plt.scatter(x=training_rounds,y=eval_result['validation_0']['rmse'],label='Training
plt.scatter(x=training_rounds,y=eval_result['validation_1']['rmse'],label='Validati
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()
```



In [18]:
```
# Verify Quality using Validation dataset
# Compare actual vs predicted performance with dataset not seen by the model before
df = pd.read_csv(validation_file,names=columns)
```

In [19]:
```
df.head()
```

Out[19]:

	count	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	mon
0	443	3	0	1	2	28.70	33.335	79	12.9980	2011	
1	387	2	0	0	1	32.80	37.880	55	12.9980	2011	
2	2	1	0	1	1	14.76	16.665	40	19.9995	2011	
3	48	1	0	1	1	9.02	9.090	47	36.9974	2011	
4	55	4	0	0	1	10.66	15.150	87	0.0000	2011	

In [20]:
```
df.shape
```

Out[20]:
```
(3266, 14)
```

In [21]:
```
#DWB# The [:,1:] excludes the datetime variable
X_test = df.iloc[:,1:]
print(X_test[:5])
```

```
   season  holiday  workingday  weather   temp   atemp  humidity  windspeed  \
0       3        0           1        2  28.70  33.335        79    12.9980
1       2        0           0        1  32.80  37.880        55    12.9980
2       1        0           1        1  14.76  16.665        40    19.9995
3       1        0           1        1   9.02   9.090        47    36.9974
4       4        0           0        1  10.66  15.150        87     0.0000

   year  month  day  dayofweek  hour
0  2011      7    7          3     8
1  2011      6   11          5    13
2  2011      2   14          0     2
3  2011      2    8          1    10
4  2011     12    4          6     8
```

In [22]:
```
result = regressor.predict(X_test)
```

In [23]:
```
result[:5]
```

Out[23]:
```
array([452.154    , 373.7294   ,   0.7550393,  64.58522  ,  83.32642  ],
      dtype=float32)
```

In [24]:
```
df['count_predicted'] = result
```

In [25]:
```
df.head()
```

Out[25]:

	count	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	mon
0	443	3	0	1	2	28.70	33.335	79	12.9980	2011	
1	387	2	0	0	1	32.80	37.880	55	12.9980	2011	
2	2	1	0	1	1	14.76	16.665	40	19.9995	2011	
3	48	1	0	1	1	9.02	9.090	47	36.9974	2011	
4	55	4	0	0	1	10.66	15.150	87	0.0000	2011	

In [26]:
```
# Negative Values are predicted
df['count_predicted'].describe()
```

Out[26]:
```
count    3266.000000
mean      190.070770
std       174.655914
min       -95.306847
25%        43.720430
50%       150.537590
75%       284.134521
max       901.711853
Name: count_predicted, dtype: float64
```

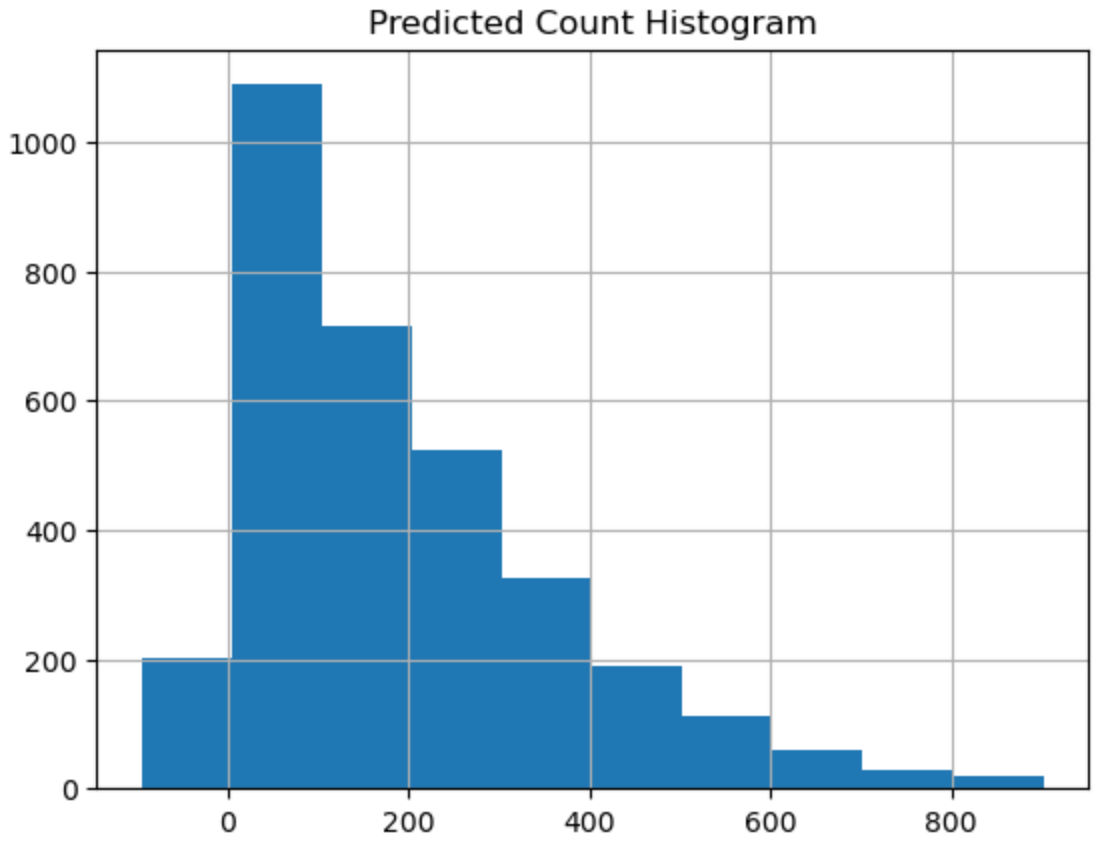
In [27]:
```
df[df['count_predicted'] < 0]
```

Out[27]:

	count	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	r
99	71	2	0	1	3	22.96	26.515	88	7.0015	2012	
103	11	3	0	1	2	27.88	31.820	83	12.9980	2012	
117	2	4	0	1	1	8.20	12.880	80	0.0000	2011	
137	9	1	0	1	1	15.58	19.695	54	7.0015	2012	
158	45	1	0	0	2	12.30	13.635	100	19.9995	2011	
...	...	...	...	...	...	...	...	...	...	...	...
3129	44	1	0	1	3	13.12	16.665	70	8.9981	2012	
3176	16	4	0	1	2	12.30	14.395	52	16.9979	2012	
3199	8	4	0	1	1	13.94	15.910	81	15.0013	2011	
3252	11	3	0	0	1	25.42	30.305	61	0.0000	2011	
3259	4	1	0	1	1	8.20	12.880	61	0.0000	2012	

127 rows × 15 columns

In [28]:
```
df['count_predicted'].hist()
plt.title('Predicted Count Histogram')
plt.show()
```



In [29]:
```
def adjust_count(x):
    if x < 0:
        return 0
    else:
        return x
```

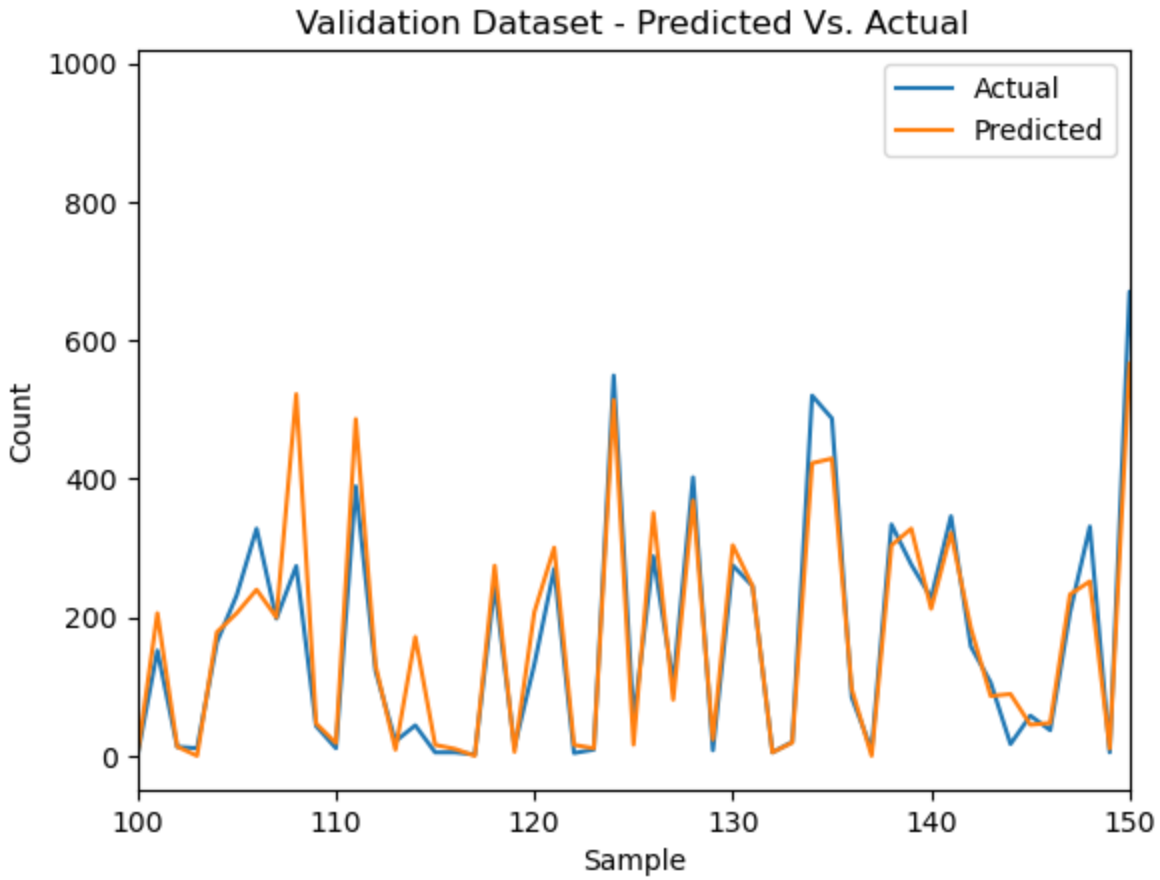
In [30]:
```
df['count_predicted'] = df['count_predicted'].map(adjust_count)
```

In [31]:
```
df[df['count_predicted'] < 0]
```

Out[31]:

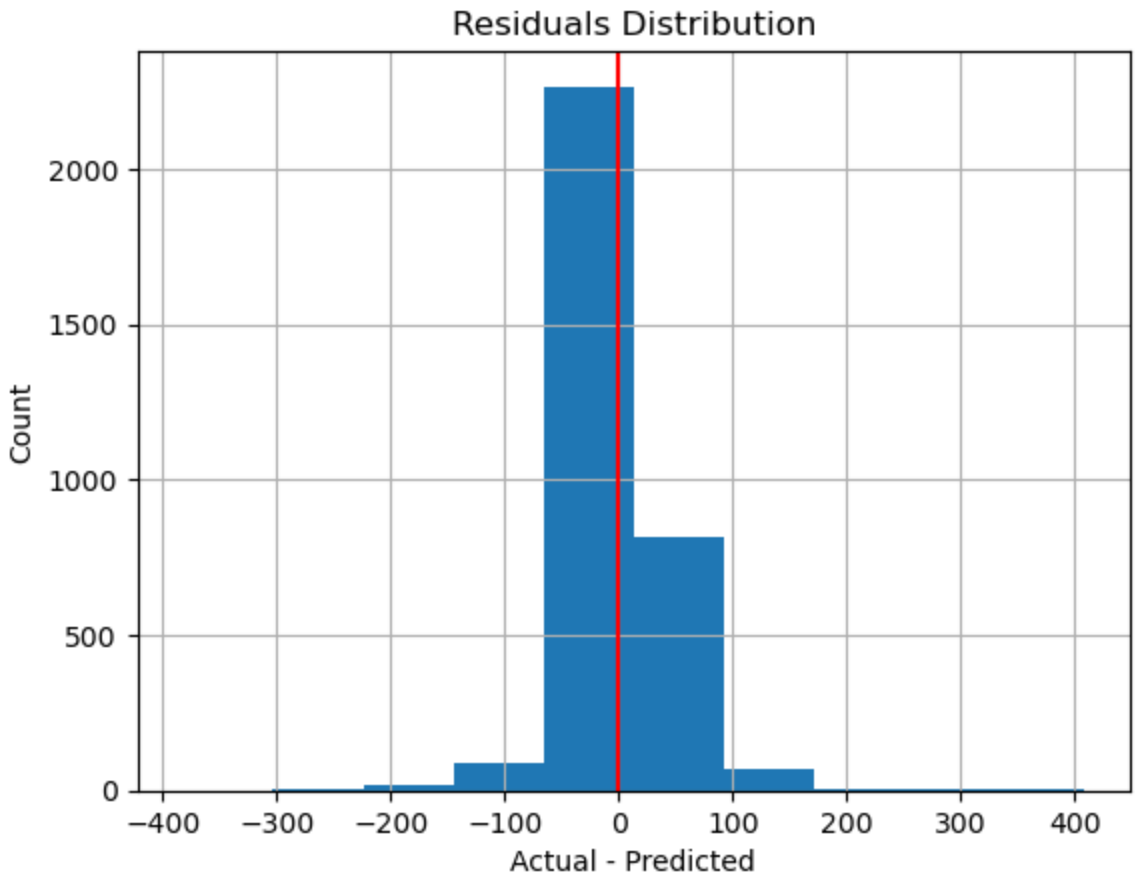
count	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	month

In [32]:
```
# Actual Vs Predicted
plt.plot(df['count'], label='Actual')
plt.plot(df['count_predicted'],label='Predicted')
plt.xlabel('Sample')
plt.ylabel('Count')
plt.xlim([100,150])
plt.title('Validation Dataset - Predicted Vs. Actual')
plt.legend()
plt.show()
```



In [33]:
```
# Over prediction and Under Prediction needs to be balanced
# Training Data Residuals
residuals = (df['count'] - df['count_predicted'])
```

```
plt.hist(residuals)
plt.grid(True)
plt.xlabel('Actual - Predicted')
plt.ylabel('Count')
plt.title('Residuals Distribution')
plt.axvline(color='r')
plt.show()
```



In [34]:
```
value_counts = (residuals > 0).value_counts(sort=False)
print(' Under Estimation: {0:0.2f}'.format(value_counts[True]/len(residuals)))
print(' Over  Estimation: {0:0.2f}'.format(value_counts[False]/len(residuals)))
```

```
 Under Estimation: 0.50
 Over  Estimation: 0.50
```

In [35]:
```
print("RMSE: {0:0.2f}".format(mean_squared_error(df['count'],df['count_predicted'])
```

```
RMSE: 40.89
```

In [36]:
```
# RMSLE - Root Mean Squared Log Error
# RMSLE Metric is used by Kaggle for this competition

# RMSE Cost Function - Magnitude of difference matters

# RMSLE cost function - "Only Percentage difference matters"

# Reference:Katerina Malahova, Khor SoonHin
# https://www.slideshare.net/KhorSoonHin/rmsle-cost-function
```

```
def compute_rmsle(y_true, y_pred):
    if type(y_true) != np.ndarray:
        y_true = np.array(y_true)

    if type(y_pred) != np.ndarray:
        y_pred = np.array(y_pred)

    return(np.average((np.log1p(y_pred) - np.log1p(y_true))**2)**.5)
```

In [37]:
```
print('RMSLE')
print(compute_rmsle(100,50),
      compute_rmsle(1000,500),
      compute_rmsle(10000,5000))
```

```
RMSLE
0.683294884116934 0.6921486782303559 0.6930471955576127
```

In [38]:
```
print('RMSLE')
print(compute_rmsle(100,25),
      compute_rmsle(1000,250),
      compute_rmsle(10000,2500))
```

```
RMSLE
1.3570239788197775 1.383301840183437 1.3859944360988976
```

In [39]:
```
print('RMSE')
print(mean_squared_error([100],[50])**.5,
      mean_squared_error([1000],[500])**.5,
      mean_squared_error([10000],[5000])**.5)
```

```
RMSE
50.0 500.0 5000.0
```

In [40]:
```
print('RMSE')
print(mean_squared_error([100],[25])**.5,
      mean_squared_error([1000],[250])**.5,
      mean_squared_error([10000],[2500])**.5)
```

```
RMSE
75.0 750.0 7500.0
```

In [41]:
```
print("RMSLE: {0}".format(compute_rmsle(df['count'],df['count_predicted'])))
```

```
RMSLE: 0.5999730473932068
```

In [42]:
```
# Prepare Data for Submission to Kaggle
df_test = pd.read_csv(test_file,parse_dates=['datetime'])
```

In [43]:
```
df_test.head()
```

Out[43]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	m
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027	2011	
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011	
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011	
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011	
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011	

In [44]:
```
X_test =  df_test.iloc[:,1:] # Exclude datetime for prediction
```

In [45]:
```
X_test.head()
```

Out[45]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	month	day
0	1	0	1	1	10.66	11.365	56	26.0027	2011	1	20
1	1	0	1	1	10.66	13.635	56	0.0000	2011	1	20
2	1	0	1	1	10.66	13.635	56	0.0000	2011	1	20
3	1	0	1	1	10.66	12.880	56	11.0014	2011	1	20
4	1	0	1	1	10.66	12.880	56	11.0014	2011	1	20

In [46]:
```
result = regressor.predict(X_test)
```

In [47]:
```
result[:5]
```

Out[47]:
```
array([ 12.3928995,  -3.7081811, -10.777083 ,  -4.4427557,  -4.4427557],
      dtype=float32)
```

In [48]:
```
df_test["count"] = result
```

In [49]:
```
df_test.head()
```

Out[49]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	n
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027	2011	
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011	
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011	
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011	
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011	

In [50]:
```
df_test[df_test["count"] < 0]
```

Out[50]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011
25	2011-01-21 01:00:00	1	0	1	2	9.84	11.365	70	16.9979	2011
...	...	...	...	...	...	...	...	...	...	...
6210	2012-12-20 03:00:00	4	0	1	2	12.30	15.910	70	6.0032	2012
6211	2012-12-20 04:00:00	4	0	1	2	12.30	15.910	70	6.0032	2012
6235	2012-12-21 04:00:00	1	0	1	2	14.76	15.910	71	32.9975	2012
6284	2012-12-23 05:00:00	1	0	0	1	8.20	12.880	51	0.0000	2012
6451	2012-12-30 06:00:00	1	0	0	2	9.84	9.850	52	27.9993	2012

285 rows × 15 columns

In [51]:
```
df_test["count"] = df_test["count"].map(adjust_count)
```

In [52]:
```
df_test[['datetime','count']].to_csv('predicted_count.csv',index=False)
```

In [ ]:
```
# RMSLE (Kaggle) Score
# Test 1: 0.62
```

In [ ]:
```

```