Linear Regression Predicting Disease Spread

Problem description

You are provided the following set of information on a (year, weekofyear) timescale:

(Where appropriate, units are provided as a _unit suffix on the feature name.)

City and date indicators
city – City abbreviations: sj for San Juan and iq for Iquitos
week_start_date – Date given in yyyy-mm-dd format

NOAA's GHCN daily climate data weather station measurements

station_max_temp_c – Maximum temperature
station_min_temp_c – Minimum temperature
station_avg_temp_c – Average temperature
station_precip_mm – Total precipitation
station_diur_temp_rng_c – Diurnal temperature range

PERSIANN satellite precipitation measurements (0.25x0.25 degree scale)

precipitation_amt_mm – Total precipitation

NOAA's NCEP Climate Forecast System Reanalysis measurements (0.5x0.5 degree scale)

reanalysis_sat_precip_amt_mm – Total precipitation
reanalysis_dew_point_temp_k – Mean dew point temperature
reanalysis_air_temp_k – Mean air temperature
reanalysis_relative_humidity_percent – Mean relative humidity
reanalysis_specific_humidity_g_per_kg – Mean specific humidity

reanalysis_precip_amt_kg_per_m2 – Total precipitation
reanalysis_max_air_temp_k – Maximum air temperature
reanalysis_min_air_temp_k – Minimum air temperature
reanalysis_avg_temp_k – Average air temperature
reanalysis_tdtr_k – Diurnal temperature range

Satellite vegetation - Normalized difference vegetation index (NDVI) - NOAA's CDR Normalized Difference Vegetation Index (0.5x0.5 degree scale) measurements

ndvi_se – Pixel southeast of city centroid
ndvi_sw – Pixel southwest of city centroid
ndvi_ne – Pixel northeast of city centroid
ndvi_nw – Pixel northwest of city centroid

Your goal is to predict the number of dengue cases that are going to be found in given city, year, week_of_year and some extra features of weather.

There are two cities, “San Juan” and “Iquitos” with test data for each city spanning 5 and 3 years respectively. The data for each city have been concatenated along with a city column indicating the source: 'sj' for San Juan and 'iq' for Iquitos. Throughout, missing values have been filled as NaNs.
Download data from:
[https://s3.amazonaws.com/drivendata/data/44/public/dengue_features_train.csv (https://s3.amazonaws.com/drivendata/data/44/public/dengue_features_train.csv)](https://s3.amazonaws.com/drivendata/data/44/public/dengue_features_train.csv)
[https://s3.amazonaws.com/drivendata/data/44/public/dengue_labels_train.csv (https://s3.amazonaws.com/drivendata/data/44/public/dengue_labels_train.csv)](https://s3.amazonaws.com/drivendata/data/44/public/dengue_labels_train.csv)

Sort the data by year and week_of_year and choose last 400 rows as test data.

Task 1: Do all the sorts of pre-processing required along with proper analysis and make data ready for models.
Task 2: You are allowed to use only linear regression, report the mean absolute error on the test data (you are free to use all sorts of feature conversion techniques, you can add/delete/modify features as you want) Performance metric Mean absolute error

In [1]:
```
import datetime
import time

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import StandardScaler
```

```
C:\Users\GauravP\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was deprec
ated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are mov
ed. Also note that the interface of the new CV iterators are different from that of this module. This module will be re
moved in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [2]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [3]:
```
data = pd.read_csv("dengue_features_train.csv")
data.shape
```

Out[3]:
```
(1456, 24)
```

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

Out[4]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_ten
0	sj	1990	18	1990-04-30	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	297.742
1	sj	1990	19	1990-05-07	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	298.442
2	sj	1990	20	1990-05-14	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	298.878
3	sj	1990	21	1990-05-21	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	299.228
4	sj	1990	22	1990-05-28	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	299.664

In [5]:
```
labels = pd.read_csv("dengue_labels_train.csv")
labels.shape
```

Out[5]: (1456, 4)

In [6]:
```
labels.head()
```

Out[6]:

	city	year	weekofyear	total_cases
0	sj	1990	18	4
1	sj	1990	19	5
2	sj	1990	20	4
3	sj	1990	21	3
4	sj	1990	22	6

In [7]:
```
#adding labels column to our main data
data["Labels"] = labels["total_cases"]
```

In [8]:
```
data.shape
```

Out[8]: (1456, 25)

In [9]:
```
data.head()
```

Out[9]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_tem
0	sj	1990	18	1990-04-30	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	297.742
1	sj	1990	19	1990-05-07	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	298.442
2	sj	1990	20	1990-05-14	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	298.878
3	sj	1990	21	1990-05-21	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	299.228
4	sj	1990	22	1990-05-28	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	299.664

In [10]:
```
columnNamesData = data.dtypes.index
columnNamesData
```

Out[10]:
```
Index(['city', 'year', 'weekofyear', 'week_start_date', 'ndvi_ne', 'ndvi_nw',
       'ndvi_se', 'ndvi_sw', 'precipitation_amt_mm', 'reanalysis_air_temp_k',
       'reanalysis_avg_temp_k', 'reanalysis_dew_point_temp_k',
       'reanalysis_max_air_temp_k', 'reanalysis_min_air_temp_k',
       'reanalysis_precip_amt_kg_per_m2',
       'reanalysis_relative_humidity_percent', 'reanalysis_sat_precip_amt_mm',
       'reanalysis_specific_humidity_g_per_kg', 'reanalysis_tdtr_k',
       'station_avg_temp_c', 'station_diur_temp_rng_c', 'station_max_temp_c',
       'station_min_temp_c', 'station_precip_mm', 'Labels'],
      dtype='object')
```

In [11]:
```
print(columnNamesData.shape)
```

```
(25,)
```

Task 1: Do all sorts of pre-processing required along with proper analysis and make data ready for models.

In [12]:
```
#finding for how many values are missing in all the columns in data
for i in columnNamesData:
    count = 0
    s = data[i].isin(["NaN"])
    for j in range(len(s)):
        if s[j]:
            count += 1
    print("Feature "+str(i)+" has "+str(count)+" missing values.\n")
```

```
Feature city has 0 missing values.

Feature year has 0 missing values.

Feature weekofyear has 0 missing values.

Feature week_start_date has 0 missing values.

Feature ndvi_ne has 194 missing values.

Feature ndvi_nw has 52 missing values.

Feature ndvi_se has 22 missing values.

Feature ndvi_sw has 22 missing values.

Feature precipitation_amt_mm has 13 missing values.

Feature reanalysis_air_temp_k has 10 missing values.

Feature reanalysis_avg_temp_k has 10 missing values.

Feature reanalysis_dew_point_temp_k has 10 missing values.

Feature reanalysis_max_air_temp_k has 10 missing values.

Feature reanalysis_min_air_temp_k has 10 missing values.

Feature reanalysis_precip_amt_kg_per_m2 has 10 missing values.

Feature reanalysis_relative_humidity_percent has 10 missing values.
```

```
Feature reanalysis_sat_precip_amt_mm has 13 missing values.

Feature reanalysis_specific_humidity_g_per_kg has 10 missing values.

Feature reanalysis_tdtr_k has 10 missing values.

Feature station_avg_temp_c has 43 missing values.

Feature station_diur_temp_rng_c has 43 missing values.

Feature station_max_temp_c has 20 missing values.

Feature station_min_temp_c has 14 missing values.

Feature station_precip_mm has 22 missing values.

Feature Labels has 0 missing values.
```

In [13]:
```
missingDataColumnNames = columnNamesData[4:24]
```

In [14]:
```
missingDataColumnNames
```

Out[14]:
```
Index(['ndvi_ne', 'ndvi_nw', 'ndvi_se', 'ndvi_sw', 'precipitation_amt_mm',
       'reanalysis_air_temp_k', 'reanalysis_avg_temp_k',
       'reanalysis_dew_point_temp_k', 'reanalysis_max_air_temp_k',
       'reanalysis_min_air_temp_k', 'reanalysis_precip_amt_kg_per_m2',
       'reanalysis_relative_humidity_percent', 'reanalysis_sat_precip_amt_mm',
       'reanalysis_specific_humidity_g_per_kg', 'reanalysis_tdtr_k',
       'station_avg_temp_c', 'station_diur_temp_rng_c', 'station_max_temp_c',
       'station_min_temp_c', 'station_precip_mm'],
      dtype='object')
```

In [15]:
```
len(missingDataColumnNames)
```

Out[15]:
```
20
```

In [16]:
```
for i in range(len(data["week_start_date"])):
    mydate = data["week_start_date"][i]
    s = time.mktime(datetime.datetime.strptime(mydate, "%Y-%m-%d").timetuple())
    data["week_start_date"][i] = s
```

```
C:\Users\GauravP\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-co
py (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
```

The date in the column "week_start_date" has been converted into unix timestamp

In [17]:
```
data.sort_values("week_start_date", axis = 0, inplace = True)
```

In [18]:
```
print(data.shape)
```

```
(1456, 25)
```

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

Out[19]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_tem
0	sj	1990	18	6.41426e+08	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	297.742
1	sj	1990	19	6.42031e+08	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	298.442
2	sj	1990	20	6.42636e+08	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	298.878
3	sj	1990	21	6.43241e+08	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	299.228
4	sj	1990	22	6.43846e+08	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	299.664

In [21]:
```
data.reset_index(drop=True, inplace = True)
```

In [22]:
```
data["precipitation_amt_mm"].value_counts()[0]
```

Out[22]:
```
239
```

In [23]:
```
for j in range(len(data["precipitation_amt_mm"])):
    a = data["precipitation_amt_mm"][j]
    b = str(a)
    if b == "nan":
        data["precipitation_amt_mm"][j] = 0.00
```

```
C:\Users\GauravP\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-co
py (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  """
```

During data pre-processing stage it has been found out that the column "precipitation_amt_mm" has 13 missing values and furthermore, most occuring value in this column is 0 which is occuring 239 times out of total 1456. Hence, it has been decided to replace all the missing values of this column with 0.

In [24]:
```
data.shape
```

Out[24]:
```
(1456, 25)
```

In [28]:
```
data.head()
```

Out[28]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_ten
0	sj	1990	18	6.41426e+08	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	297.742
1	sj	1990	19	6.42031e+08	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	298.442
2	sj	1990	20	6.42636e+08	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	298.878
3	sj	1990	21	6.43241e+08	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	299.228
4	sj	1990	22	6.43846e+08	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	299.664

In [28]:
```
print("Rows with missing values in more than 10 columns")
missingColumnIndex = []
for i in range(len(data)):
    count = 0
    for j in missingDataColumnNames:
        a =  data.iloc[i][j]
        b = str(a)
        if b == "nan":
            count += 1
    if count > 10:
        missingColumnIndex.append(i)
missingColumnIndex
```

```
Rows with missing values in more than 10 columns
```

Out[28]:
```
[87, 139, 399, 451, 893, 894, 997, 998, 1378, 1430]
```

It has been found out that there are 10 rows where more than 10 out of 25 feature values are missing. Therefore, it has been decided that these 10 rows will be removed from the dataset.

In [29]:
```
data.drop(data.index[missingColumnIndex], inplace=True)
data.shape
```

Out[29]:
```
(1446, 25)
```

In [30]:
```
data.reset_index(drop=True, inplace = True)
```

In [31]:
```
data.shape
```

Out[31]:
```
(1446, 25)
```

In [32]:
```
count = 0
for i in range(len(data["Labels"])):
    if data["Labels"][i] < 34:
        count += 1
perc = (count/len(data["Labels"])) * 100
print("Percentage of rows with labels less than 34 = "+str(perc)+"%")
```

```
Percentage of rows with labels less than 34 = 80.56708160442601%
```

It has been found out that more than 80% of the rows have dengue cases less than 34. It means all the labels above 34 will be outliers and hence they will impact our model. Therefore, it has been decided that all of the rows with dengue cases above 34 shall be removed.

In [33]:
```
RowsWithHighLabels = []
for i in range(len(data["Labels"])):
    if data["Labels"][i] > 34:
        RowsWithHighLabels.append(i)
len(RowsWithHighLabels)
```

Out[33]:
```
268
```

In [34]:
```
data.drop(data.index[RowsWithHighLabels], inplace=True)
data.shape
```

Out[34]:
```
(1178, 25)
```

In [35]:
```
Data = data.reset_index(drop=True, inplace = False)
```

In [36]:
```
Data.head()
```

Out[36]:

	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_ten
0	sj	1990	18	6.41426e+08	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	297.742
1	sj	1990	19	6.42031e+08	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	298.442
2	sj	1990	20	6.42636e+08	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	298.878
3	sj	1990	21	6.43241e+08	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	299.228
4	sj	1990	22	6.43846e+08	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	299.664

In [37]:
```
Data.shape
```

Out[37]: (1178, 25)

In [38]:
```
Data_cle = Data.drop("city", axis = 1, inplace = False)
```

In [39]:
```
Data_city = Data["city"]
```

In [40]:
```
Data_cle.head()
```

Out[40]:

	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_temp_k
0	1990	18	6.41426e+08	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	297.742857
1	1990	19	6.42031e+08	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	298.442857
2	1990	20	6.42636e+08	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	298.878571
3	1990	21	6.43241e+08	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	299.228571
4	1990	22	6.43846e+08	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	299.664286

In [41]:
```
Data_cle.shape
```

Out[41]: (1178, 24)

In [42]:
```
columnNamesData2 = Data_cle.dtypes.index
columnNamesData2
```

Out[42]:
```
Index(['year', 'weekofyear', 'week_start_date', 'ndvi_ne', 'ndvi_nw',
       'ndvi_se', 'ndvi_sw', 'precipitation_amt_mm', 'reanalysis_air_temp_k',
       'reanalysis_avg_temp_k', 'reanalysis_dew_point_temp_k',
       'reanalysis_max_air_temp_k', 'reanalysis_min_air_temp_k',
       'reanalysis_precip_amt_kg_per_m2',
       'reanalysis_relative_humidity_percent', 'reanalysis_sat_precip_amt_mm',
       'reanalysis_specific_humidity_g_per_kg', 'reanalysis_tdtr_k',
       'station_avg_temp_c', 'station_diur_temp_rng_c', 'station_max_temp_c',
       'station_min_temp_c', 'station_precip_mm', 'Labels'],
      dtype='object')
```

In [43]:
```
missingImputation = Imputer(missing_values = "NaN", strategy = "median", axis = 0, copy = False)

mi = missingImputation.fit_transform(Data_cle)
```

All of the missing values in dataset have been imputed using median strategy as medians are less impacted by outliers

In [44]:
```
Data_New = pd.DataFrame(mi, columns=columnNamesData2)
Data_New.head(20)
```

Out[44]:

	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_k	reanalysis_avg_temp_
0	1990.0	18.0	641426400.0	0.122600	0.103725	0.198483	0.177617	12.42	297.572857	297.74285
1	1990.0	19.0	642031200.0	0.169900	0.142175	0.162357	0.155486	22.82	298.211429	298.44285
2	1990.0	20.0	642636000.0	0.032250	0.172967	0.157200	0.170843	34.54	298.781429	298.87857
3	1990.0	21.0	643240800.0	0.128633	0.245067	0.227557	0.235886	15.36	298.987143	299.22857
4	1990.0	22.0	643845600.0	0.196200	0.262200	0.251200	0.247340	7.52	299.518571	299.66428
5	1990.0	23.0	644450400.0	0.158056	0.174850	0.254314	0.181743	9.58	299.630000	299.76428
6	1990.0	24.0	645055200.0	0.112900	0.092800	0.205071	0.210271	3.48	299.207143	299.22142
7	1990.0	25.0	645660000.0	0.072500	0.072500	0.151471	0.133029	151.12	299.591429	299.52857
8	1990.0	26.0	646264800.0	0.102450	0.146175	0.125571	0.123600	19.32	299.578571	299.55714
9	1990.0	27.0	646869600.0	0.158056	0.121550	0.160683	0.202567	14.41	300.154286	300.27857
10	1990.0	28.0	647474400.0	0.192875	0.082350	0.191943	0.152929	22.27	299.512857	299.59285
11	1990.0	29.0	648079200.0	0.291600	0.211800	0.301200	0.280667	59.17	299.667143	299.75000
12	1990.0	30.0	648684000.0	0.150567	0.171700	0.226900	0.214557	16.48	299.558571	299.63571
13	1990.0	31.0	649288800.0	0.158056	0.247150	0.379700	0.381357	32.66	299.862857	299.95000
14	1990.0	32.0	649893600.0	0.158056	0.064333	0.164443	0.138857	28.80	300.391429	300.47857
15	1990.0	33.0	650498400.0	0.158056	0.128033	0.206957	0.168243	90.75	299.958571	299.95714
16	1990.0	34.0	651103200.0	0.190233	0.168800	0.167657	0.172286	32.40	300.332857	300.41428
17	1990.0	35.0	651708000.0	0.252900	0.330750	0.264171	0.284314	40.94	300.118571	300.22142
18	1990.0	36.0	652312800.0	0.235400	0.200025	0.283817	0.230443	28.86	300.530000	300.63571
19	1990.0	37.0	652917600.0	0.127967	0.437100	0.123400	0.148283	64.56	300.674286	300.79285

In [45]:
```
Data_New.shape
```

Out[45]:
```
(1178, 24)
```

Task 2: You are allowed to use only linear regression, report the mean absolute error on the test data (you are free to use all sorts of feature conversion techniques, you can add/delete/modify features as you want) Performance metric Mean absolute error

In [46]:
```
Final_Data = Data_New.drop("Labels", axis = 1, inplace = False)
Final_Data_Labels = Data_New["Labels"]
```

In [47]:
```
print(Final_Data.shape)
print(Final_Data_Labels.shape)
```

```
(1178, 23)
(1178,)
```

In [48]:
```
Final_Data_std = StandardScaler().fit_transform(Final_Data)
Final_Data_std.shape
```

Out[48]:
```
(1178, 23)
```

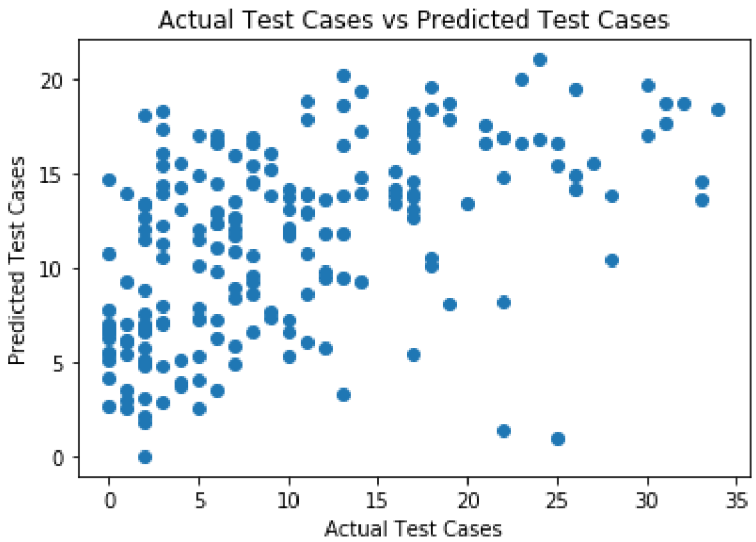
In [62]:
```
X_Train, X_Test, Y_Train, Y_Test = train_test_split(Final_Data_std, Final_Data_Labels, test_size = 0.17, random_state = 5
X_Train.shape, X_Test.shape, Y_Train.shape, Y_Test.shape
```

Out[62]:
```
((977, 23), (201, 23), (977,), (201,))
```

In [63]:
```
clf = LinearRegression()
clf.fit(X_Train, Y_Train)

Y_pred = clf.predict(X_Test)

plt.scatter(Y_Test, Y_pred)
plt.xlabel("Actual Test Cases")
plt.ylabel("Predicted Test Cases")
plt.title("Actual Test Cases vs Predicted Test Cases")
plt.show()
```



In [64]:
```
print("(Y_true, Predicted Y)")
for xy in zip(Y_Test, Y_pred):
    print(xy)
```

```
(Y_true, Predicted Y)
(8.0, 8.57869780568274)
(7.0, 12.49564321685605)
(2.0, 6.537481530084217)
(21.0, 17.54969375210083)
(2.0, 12.684129602640269)
(2.0, 7.039845271811534)
(8.0, 16.908274594381698)
(3.0, 11.292796879565483)
(3.0, 2.8930220793398274)
(5.0, 2.4901349893894995)
(0.0, 4.165682046902185)
(7.0, 8.42440800370681)
(11.0, 10.700621609146683)
(32.0, 18.748397739149297)
(22.0, 8.218447946267995)
(2.0, 18.040185339605795)
(14.0, 14.781597211545524)
(18.0, 18.344617006042306)
```

In [65]:
```
print("Mean Absolute Error = "+str(mean_absolute_error(Y_Test, Y_pred)))
```

```
Mean Absolute Error = 5.758645447993808
```