Anomaly Detection using Isolation Forest

Team:

- Anantha Krishna A
- K Venkat Ramnan
- Ruthu Shankar
- Khushi Sahay

Anomaly Detection

Anomaly Detection otherwise called outlier detection is a step in data mining that identifies data points, events, and/or observations that deviate from a dataset's normal behavior. Anomalous data can indicate critical incidents, such as a technical glitch, or potential opportunities, for instance a change in consumer behavior.

Anomaly Detection is very important in the field of meterology. Anomalies in weather include sudden increase in downpour(rainfall) , decrease of pressure in certain region leading to storms ,etc. Finding these anomalies help in predicting these natural disasters. Also finding anomalies in weather data is important to understand the meterological geography of the given place.

Isolation Forest

As labeling data is a difficult task , unsupervised anomaly detection becomes more practical.There are numerous methods for outlier detection including LOF , etc.

The usual methods are not optimized to detect anomalies, instead, they are optimized to find normal instances, because of which the result of anomaly detection either contains too many false positives or might detect too few anomalies. Many of these methods are computationally complex and hence suit low dimensional and/or small-sized data.

Swamping is a situation of wrongly identifying the normal instances as anomalous, which can happen when normal and anomalous instances are close to each other. Masking is the situation of wrongly identifying anomalous instances as normal, frequently happening when they are collectively situated in a dense area, so that they "conceal" their own presence. The sub-sampling in the iForest allows it to build a partial model that is resistant to these two effects.

This is where isolation forest comes in as it is capable of addressing both of these above concerns.

Algorithm

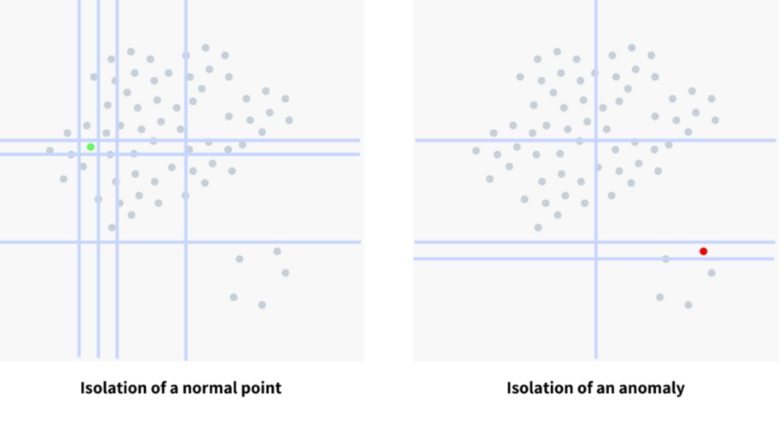
The main principle behind the Isolation forest is to 'isolate' anomalies by creating decision trees over random attributes. The random partitioning produces noticeable shorter paths for anomalies since

- fewer instances (of anomalies) result in smaller partitions
- distinguishable attribute values are more likely to be separated in early partitioning

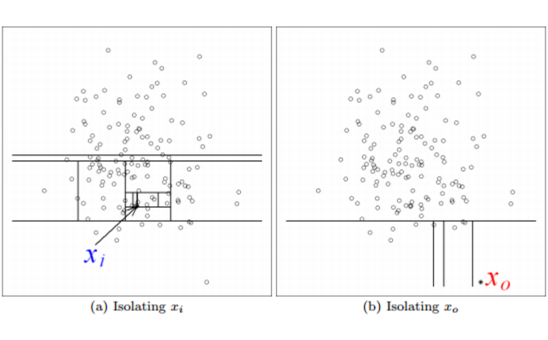
Hence, when a forest of random trees collectively produces shorter path lengths for some particular points, then they are highly likely to be anomalies.

The following steps define the working of Isolation tree :

1. Random and recursive partition of data is carried out, which is represented as a tree (random forest). This is the training stage where the user defines the parameters of the subsample and the number of trees.The convergence is reached as the number of tree increases. However, fine tuning may be required on the case basis.



2.The end of the tree is reached once the recursive partition of data is finished. It is expected that the distance taken to reach the outlier is far less than that for the normal data (see the below figure).



1. Then the distance of the path is averaged and normalised to calculate the anomaly score.

Desision Tree for Isolation Forest

The decision tree is constructed by splitting the sub-sample points/instances over a split value of a randomly selected attribute such that the instances whose corresponding attribute value is smaller than the split value goes left and the others go right, and the process is continued recursively until the tree is fully constructed.

Anomaly Score

$$Anomaly\ Score\ (S) = 2^{\frac{-E(h(k,m,N))}{c(n)}}$$
$$,where\ c(n) = 2(\ln(n-1) + 0.5772156649) - 2\left(\frac{n-1}{n}\right)$$
$$,where\ n\ is\ a\ number\ of\ data\ points\ in\ a\ chosen\ sample$$
$$,where\ E(h(k,m,N)) = \frac{\sum_{i=1}^{N}\begin{cases} if\ k == 1, \sum_{j=1}^{M} 1 \\ else,\ \sum_{j=1}^{M} 1 + c(k)\end{cases}}{N}$$
$$,where\ N\ is\ a\ total\ number\ of\ trees$$
$$,where\ M\ is\ a\ total\ number\ of\ binary\ splits$$
$$,where\ k\ is\ a\ total\ number\ of\ data\ points\ in\ the\ final\ node\ (exit\ node)$$

We calculate this anomaly score for each tree and average them out across different trees and get the final anomaly score for an entire forest for a given data point

Mathematically, an outlier gets a score closer to 1.

More info can be found in the paper 'Isolation Forest;Fei Tony Liu, Kai Ming Ting ;Zhi-Hua Zhou'

New York City dataset

In [4]:
```
import pandas as pd
```

In [5]:
```
import numpy as np
```

In [6]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [7]:
```
from sklearn.ensemble import IsolationForest
```

In [8]:
```
df=pd.read_csv('NewYork.csv')
```

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

Out[9]:

	datetime	Humidity	Pressure	Temperature	Wind_Direction	Wind_Speed
0	01-10-2012 13:00	58	1012	288.220000	260	7
1	01-10-2012 14:00	57	1012	288.247676	260	7
2	01-10-2012 15:00	57	1012	288.326940	260	7
3	01-10-2012 16:00	57	1012	288.406203	260	7
4	01-10-2012 17:00	57	1012	288.485467	261	6

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43609 entries, 0 to 43608
Data columns (total 6 columns):
datetime          43609 non-null object
Humidity          43609 non-null int64
Pressure          43609 non-null int64
Temperature       43609 non-null float64
Wind_Direction    43609 non-null int64
Wind_Speed        43609 non-null int64
dtypes: float64(1), int64(4), object(1)
memory usage: 2.0+ MB
```

In [17]:
```
# Removing the Nan Data
```

In [18]:
```
df.dropna(inplace=True)
```

Understanding the outliers

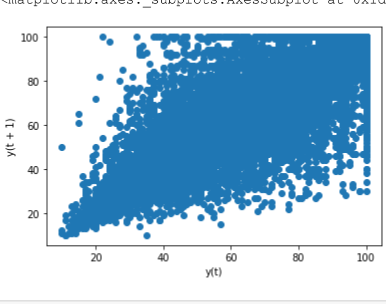
Lag Plots

Outliers are easily discernible on a lag plot.

In [21]:
```
from pandas.plotting import lag_plot
```

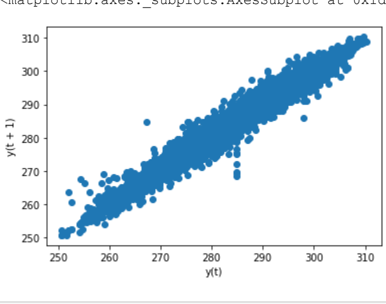
In [25]:
```
lag_plot(df.Humidity)
```

Out[25]: `<matplotlib.axes._subplots.AxesSubplot at 0x1d18abf0ac8>`



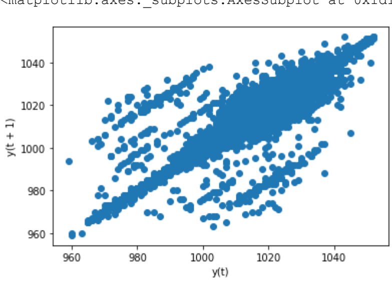
In [24]:
```
lag_plot(df.Temperature)
```

Out[24]: `<matplotlib.axes._subplots.AxesSubplot at 0x1d18a136748>`



In [26]:
```
lag_plot(df.Pressure)
```

Out[26]: `<matplotlib.axes._subplots.AxesSubplot at 0x1d18a1fa708>`



Isolation forest using Scikit Learn

In [13]:
```
features=df[['Humidity','Pressure','Temperature','Wind_Direction','Wind_Speed']]
```

In [14]:
```
model=IsolationForest(n_estimators=100, max_samples='auto', contamination=float(0.1),max_features=1.0)
model.fit(features)
```

Out[14]: `IsolationForest(contamination=0.1)`

In [15]:
```
model.predict(features)
```

Out[15]: `array([1, 1, 1, ..., 1, 1, 1])`

In [16]:
```
df['scores']=model.decision_function(features)
df['anomaly']=model.predict(features)
```

In [17]:
```
df
```

Out[17]:

	datetime	Humidity	Pressure	Temperature	Wind_Direction	Wind_Speed	scores	anomaly
0	01-10-2012 13:00	58	1012	288.220000	260	7	0.067877	1
1	01-10-2012 14:00	57	1012	288.247676	260	7	0.068237	1
2	01-10-2012 15:00	57	1012	288.326940	260	7	0.068237	1
3	01-10-2012 16:00	57	1012	288.406203	260	7	0.068237	1
4	01-10-2012 17:00	57	1012	288.485467	261	6	0.091416	1
...	...	...	...	...	...	...	...	...
43604	27-10-2017 20:00	36	1019	289.980000	0	3	0.036953	1
43605	27-10-2017 21:00	38	1019	289.480000	0	1	0.025145	1
43606	27-10-2017 22:00	54	1019	287.920000	196	2	0.118365	1
43607	27-10-2017 23:00	62	1020	285.830000	171	3	0.129102	1
43608	28-10-2017 00:00	58	1020	284.980000	0	2	0.084043	1

43609 rows × 8 columns

In [18]:
```
# Closer look at anomaly regions
```

In [19]:
```
df.loc[df['anomaly'] == 1] # Normal Values
```

Out[19]:

	datetime	Humidity	Pressure	Temperature	Wind_Direction	Wind_Speed	scores	anomaly
0	01-10-2012 13:00	58	1012	288.220000	260	7	0.067877	1
1	01-10-2012 14:00	57	1012	288.247676	260	7	0.068237	1
2	01-10-2012 15:00	57	1012	288.326940	260	7	0.068237	1
3	01-10-2012 16:00	57	1012	288.406203	260	7	0.068237	1
4	01-10-2012 17:00	57	1012	288.485467	261	6	0.091416	1
...	...	...	...	...	...	...	...	...
43604	27-10-2017 20:00	36	1019	289.980000	0	3	0.036953	1
43605	27-10-2017 21:00	38	1019	289.480000	0	1	0.025145	1
43606	27-10-2017 22:00	54	1019	287.920000	196	2	0.118365	1
43607	27-10-2017 23:00	62	1020	285.830000	171	3	0.129102	1
43608	28-10-2017 00:00	58	1020	284.980000	0	2	0.084043	1

39248 rows × 8 columns

In [20]:
```
df.loc[df['anomaly'] == -1]
```

Out[20]:

	datetime	Humidity	Pressure	Temperature	Wind_Direction	Wind_Speed	scores	anomaly
42	03-10-2012 11:00	100	1016	289.56	0	0	-0.017761	-1
62	04-10-2012 07:00	100	1016	290.13	0	0	-0.014786	-1
81	05-10-2012 04:00	100	1020	293.44	0	0	-0.016069	-1
82	05-10-2012 05:00	100	1020	293.34	0	0	-0.016069	-1
83	05-10-2012 06:00	100	1020	292.86	0	0	-0.016599	-1
...	...	...	...	...	...	...	...	...
43484	22-10-2017 20:00	33	1029	296.48	0	2	-0.001093	-1
43527	24-10-2017 15:00	78	1011	295.98	167	9	-0.011196	-1
43531	24-10-2017 19:00	83	1008	295.15	170	10	-0.035314	-1
43532	24-10-2017 20:00	88	1008	294.69	180	9	-0.018766	-1
43533	24-10-2017 21:00	88	1008	294.73	170	9	-0.022540	-1

4361 rows × 8 columns

In [21]:
```
outliers=df.loc[df['anomaly']==-1]
outlier_index=list(outliers.index)
```

In [22]:
```
print(df['anomaly'].value_counts()) # Count of anomalies
```

```
 1    39248
-1     4361
Name: anomaly, dtype: int64
```

Isolation forest for single parameters

Temperature

In [23]:
```
df_temp=df
```

In [24]:
```
del df_temp['scores']
del df_temp['anomaly']
```

In [25]:
```
df_temp
```

Out[25]:

	datetime	Humidity	Pressure	Temperature	Wind_Direction	Wind_Speed
0	01-10-2012 13:00	58	1012	288.220000	260	7
1	01-10-2012 14:00	57	1012	288.247676	260	7
2	01-10-2012 15:00	57	1012	288.326940	260	7
3	01-10-2012 16:00	57	1012	288.406203	260	7
4	01-10-2012 17:00	57	1012	288.485467	261	6
...	...	...	...	...	...	...
43604	27-10-2017 20:00	36	1019	289.980000	0	3
43605	27-10-2017 21:00	38	1019	289.480000	0	1
43606	27-10-2017 22:00	54	1019	287.920000	196	2
43607	27-10-2017 23:00	62	1020	285.830000	171	3
43608	28-10-2017 00:00	58	1020	284.980000	0	2

43609 rows × 6 columns

In [26]:
```
model=IsolationForest(n_estimators=100, max_samples='auto', contamination=float(0.1),max_features=1.0)
model.fit(np.array(df_temp['Temperature']).reshape(-1,1))
```

Out[26]: `IsolationForest(contamination=0.1)`

In [27]:
```
model.predict(np.array(df_temp['Temperature']).reshape(-1,1))
```

Out[27]: `array([1, 1, 1, ..., 1, 1, 1])`

In [28]:
```
df_temp['anomaly']=model.predict((np.array(df_temp['Temperature']).reshape(-1,1)))
```

In [29]:
```
df_temp
```

Out[29]:

	datetime	Humidity	Pressure	Temperature	Wind_Direction	Wind_Speed	anomaly
0	01-10-2012 13:00	58	1012	288.220000	260	7	1
1	01-10-2012 14:00	57	1012	288.247676	260	7	1
2	01-10-2012 15:00	57	1012	288.326940	260	7	1
3	01-10-2012 16:00	57	1012	288.406203	260	7	1
4	01-10-2012 17:00	57	1012	288.485467	261	6	1
...	...	...	...	...	...	...	...
43604	27-10-2017 20:00	36	1019	289.980000	0	3	1
43605	27-10-2017 21:00	38	1019	289.480000	0	1	1
43606	27-10-2017 22:00	54	1019	287.920000	196	2	1
43607	27-10-2017 23:00	62	1020	285.830000	171	3	1
43608	28-10-2017 00:00	58	1020	284.980000	0	2	1

43609 rows × 7 columns

In [32]:
```
outliers=df_temp.loc[df_temp['anomaly']==-1]
outlier_index=list(outliers.index)
print(df_temp['anomaly'].value_counts()) # Count of anomalies
```

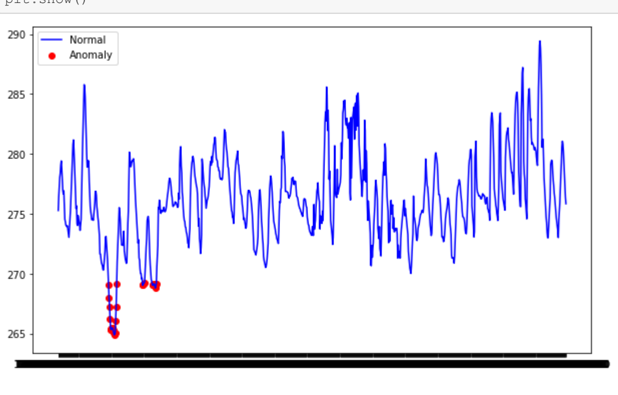
```
 1    39256
-1     4353
Name: anomaly, dtype: int64
```

Visualization temperature anomaly

In [30]:
```
df_temp_2=df_temp.iloc[3000:4000]
```

In [31]:
```
#df_temp_2.loc[df_temp_2['anomaly'] == -1]
```

In [104]:
```
fig,ax=plt.subplots(figsize=(10,6))
a=df_temp_2.loc[df_temp_2['anomaly'] == -1 , ['datetime','Temperature']]
ax.plot(df_temp_2['datetime'],df_temp_2['Temperature'],color='blue',label='Normal')
ax.scatter(a['datetime'],a['Temperature'],color='red',label='Anomaly')
plt.legend()
plt.show()
```



Humidity

In [33]:
```
df_hum=df_temp
del df_hum['anomaly']
```

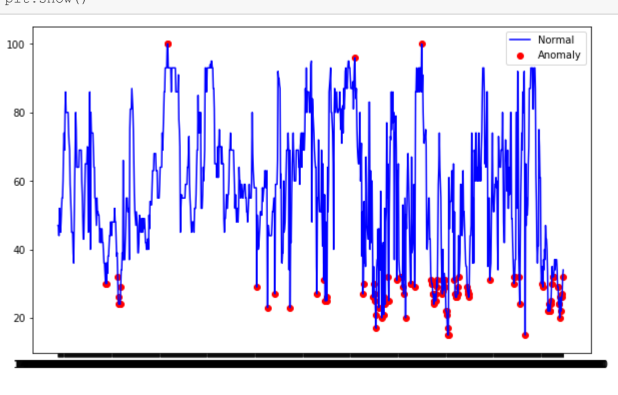
In [34]:
```
model=IsolationForest(n_estimators=100, max_samples='auto', contamination=float(0.1),max_features=1.0)
model.fit(np.array(df_hum['Humidity']).reshape(-1,1))

model.predict(np.array(df_hum['Humidity']).reshape(-1,1))

df_hum['anomaly']=model.predict((np.array(df_hum['Humidity']).reshape(-1,1)))
```

In [35]:
```
outliers=df_hum.loc[df_hum['anomaly']==-1]
outlier_index=list(outliers.index)
print(df_hum['anomaly'].value_counts()) # Count of anomalies
```

```
 1    39275
-1     4334
Name: anomaly, dtype: int64
```

Visualization humidity anomaly

In [110]:
```
df_hum_2=df_hum.iloc[3000:4000]
#df_temp_2.loc[df_temp_2['anomaly'] == -1]
fig,ax=plt.subplots(figsize=(10,6))
a=df_hum_2.loc[df_hum_2['anomaly'] == -1 , ['datetime','Humidity']]
ax.plot(df_hum_2['datetime'],df_hum_2['Humidity'],color='blue',label='Normal')
ax.scatter(a['datetime'],a['Humidity'],color='red',label='Anomaly')
plt.legend()
plt.show()
```



Pressure

In [36]:
```
df_pre=df_temp
del df_pre['anomaly']
```

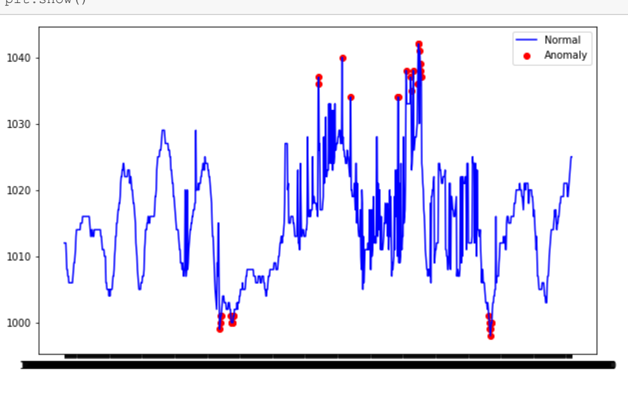
In [37]:
```
model=IsolationForest(n_estimators=100, max_samples='auto', contamination=float(0.1),max_features=1.0)
model.fit(np.array(df_pre['Pressure']).reshape(-1,1))

model.predict(np.array(df_pre['Pressure']).reshape(-1,1))

df_pre['anomaly']=model.predict((np.array(df_pre['Pressure']).reshape(-1,1)))
```

In [38]:
```
outliers=df_hum.loc[df_pre['anomaly']==-1]
outlier_index=list(outliers.index)
print(df_pre['anomaly'].value_counts()) # Count of anomalies
```

```
 1    39525
-1     4084
Name: anomaly, dtype: int64
```

Visualization Pressure anomaly

In [115]:
```
df_pre_2=df_pre.iloc[3000:4000]
#df_temp_2.loc[df_temp_2['anomaly'] == -1]
fig,ax=plt.subplots(figsize=(10,6))
a=df_pre_2.loc[df_pre_2['anomaly'] == -1 , ['datetime','Pressure']]
ax.plot(df_pre_2['datetime'],df_pre_2['Pressure'],color='blue',label='Normal')
ax.scatter(a['datetime'],a['Pressure'],color='red',label='Anomaly')
plt.legend()
plt.show()
```



Wind Speed

In [39]:
```
df_wd=df_temp
del df_wd['anomaly']
```

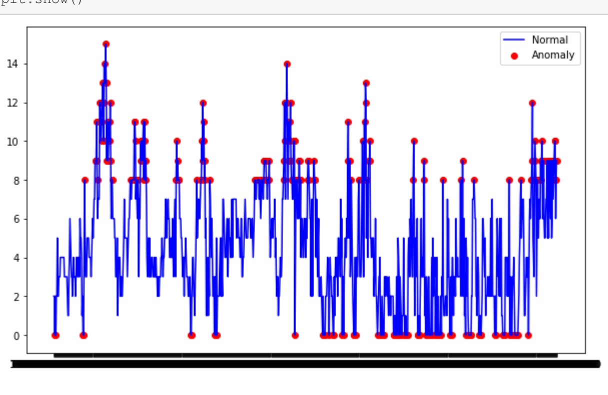
In [44]:
```
model=IsolationForest(n_estimators=100, max_samples='auto', contamination=float(0.1),max_features=1.0)
model.fit(np.array(df_wd['Wind_Speed']).reshape(-1,1))

model.predict(np.array(df_wd['Wind_Speed']).reshape(-1,1))

df_wd['anomaly']=model.predict((np.array(df_wd['Wind_Speed']).reshape(-1,1)))
```

In [45]:
```
outliers=df_wd.loc[df_wd['anomaly']==-1]
outlier_index=list(outliers.index)
print(df_wd['anomaly'].value_counts()) # Count of anomalies
```

```
 1    39988
-1     3621
Name: anomaly, dtype: int64
```

Visualization Wind Speed Anomaly

In [46]:
```
df_wd_2=df_wd.iloc[3000:4000]
#df_temp_2.loc[df_temp_2['anomaly'] == -1]
fig,ax=plt.subplots(figsize=(10,6))
a=df_wd_2.loc[df_wd_2['anomaly'] == -1 , ['datetime','Wind_Speed']]
ax.plot(df_wd_2['datetime'],df_wd_2['Wind_Speed'],color='blue',label='Normal')
ax.scatter(a['datetime'],a['Wind_Speed'],color='red',label='Anomaly')
plt.legend()
plt.show()
```



Final Results

Data	Number of Outliers
All Data together	4361
Temperature	4353
Humidity	4334
Pressure	4084
Wind Speed	3621

Conclusion

- Isolation forest works very well even with a small data set to detect outliers -
- With a huge increase in computational speed, the accuracy of detecting outliers didn't get affected
- There are still challenges to implement this algorithm in a potentially automated reporting AI system

Future Scopes

- application of Deep learning
- Usage of Extended Isolation Forest