



Photo credit: Unsplash

Anomaly Detection for Dummies

Unsupervised Anomaly Detection for Univariate & Multivariate Data.



Susan Li [Follow](#)

Jul 1, 2019 · 8 min read

Anomaly detection is the process of identifying unexpected items or events in data sets, which differ from the norm. And anomaly detection is often applied on unlabeled data which is known as unsupervised anomaly detection. Anomaly detection has two basic assumptions:

- Anomalies only occur very rarely in the data.
- Their features differ from the normal instances significantly.

Univariate Anomaly Detection

Before we get to Multivariate anomaly detection, I think its necessary to work through a simple example of Univariate anomaly detection method in which we detect outliers from a distribution of values in a single feature space.

We are using the Super Store Sales data set that can be downloaded from [here](#), and we are going to find patterns in Sales and Profit separately that do not conform to expected behavior. That is, spotting outliers for one variable at a time.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
from sklearn.ensemble import IsolationForest
```

Distribution of the Sales

```
df = pd.read_excel("Superstore.xls")
df['Sales'].describe()
```

count	9994.000000
mean	229.858001
std	623.245101
min	0.444000
25%	17.280000
50%	54.490000
75%	209.940000
max	22638.480000
Name: Sales, dtype: float64	

Figure 1

```
plt.scatter(range(df.shape[0]), np.sort(df['Sales'].values))
plt.xlabel('index')
plt.ylabel('Sales')
plt.title("Sales distribution")
sns.despine()
```

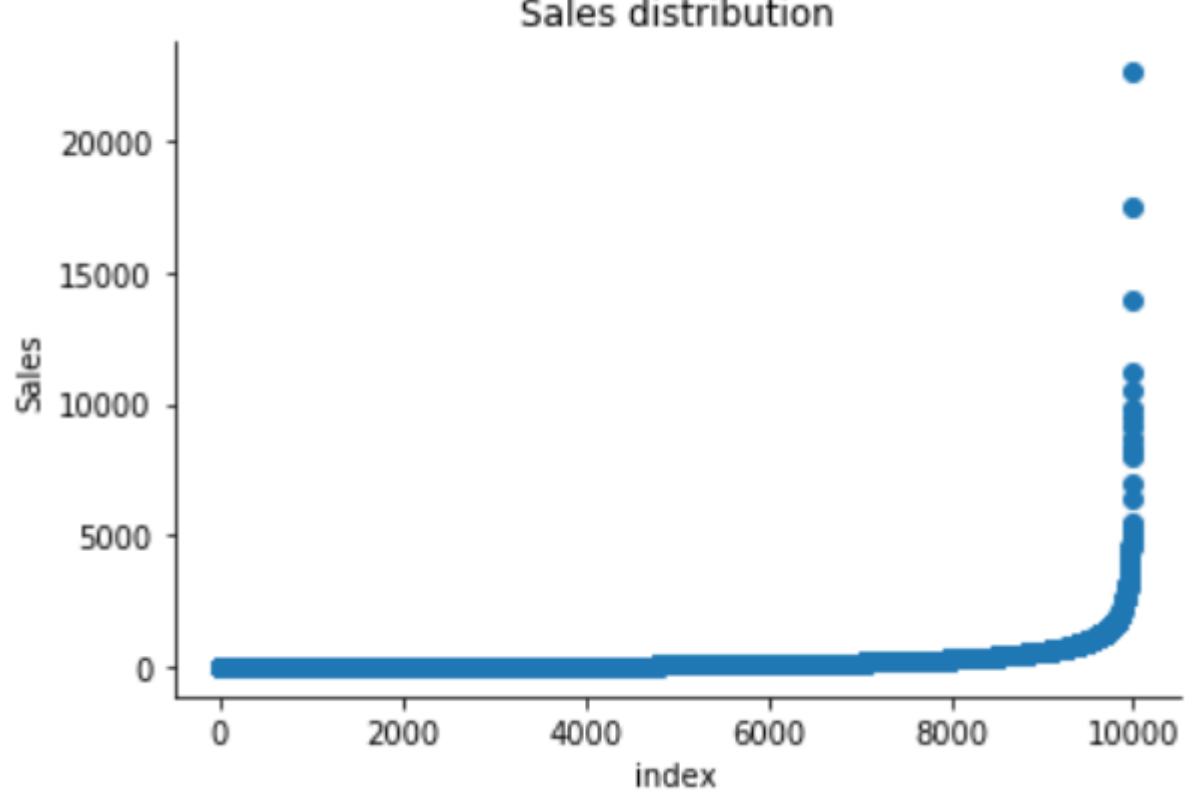


Figure 2

```
sns.distplot(df['Sales'])  
plt.title("Distribution of Sales")  
sns.despine()
```

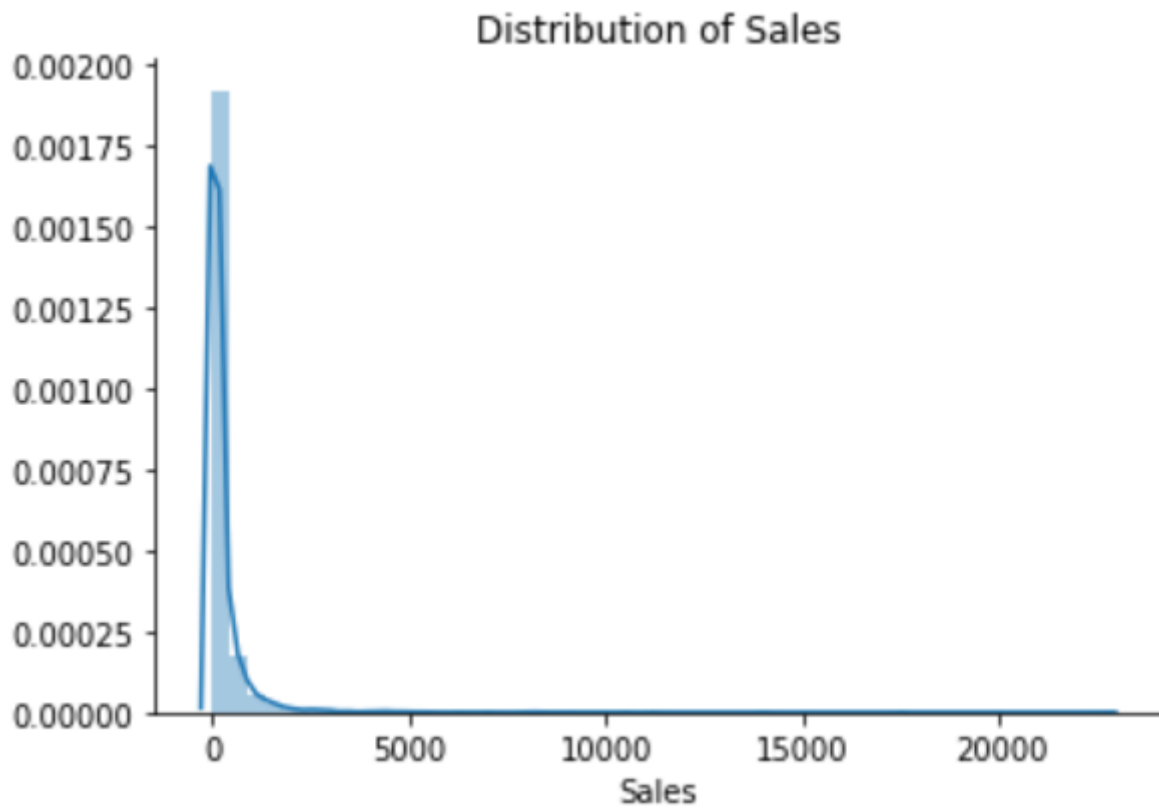


Figure 3

```
print("Skewness: %f" % df['Sales'].skew())
print("Kurtosis: %f" % df['Sales'].kurt())
```

```
Skewness: 12.972752
Kurtosis: 305.311753
```

The Superstore's sales distribution is far from a normal distribution, and it has a positive long thin tail, the mass of the distribution is concentrated on the left of the figure. And the tail sales distribution far exceeds the tails of the normal distribution.

There are one region where the data has low probability to appear which is on the right side of the distribution.

Distribution of the Profit

```
df['Profit'].describe()
```

```
count    9994.000000
mean      28.656896
std       234.260108
min      -6599.978000
25%         1.728750
50%         8.666500
75%        29.364000
max       8399.976000
Name: Profit, dtype: float64
```

Figure 4

```
plt.scatter(range(df.shape[0]), np.sort(df['Profit'].values))
plt.xlabel('index')
plt.ylabel('Profit')
plt.title("Profit distribution")
sns.despine()
```

Profit distribution

8000

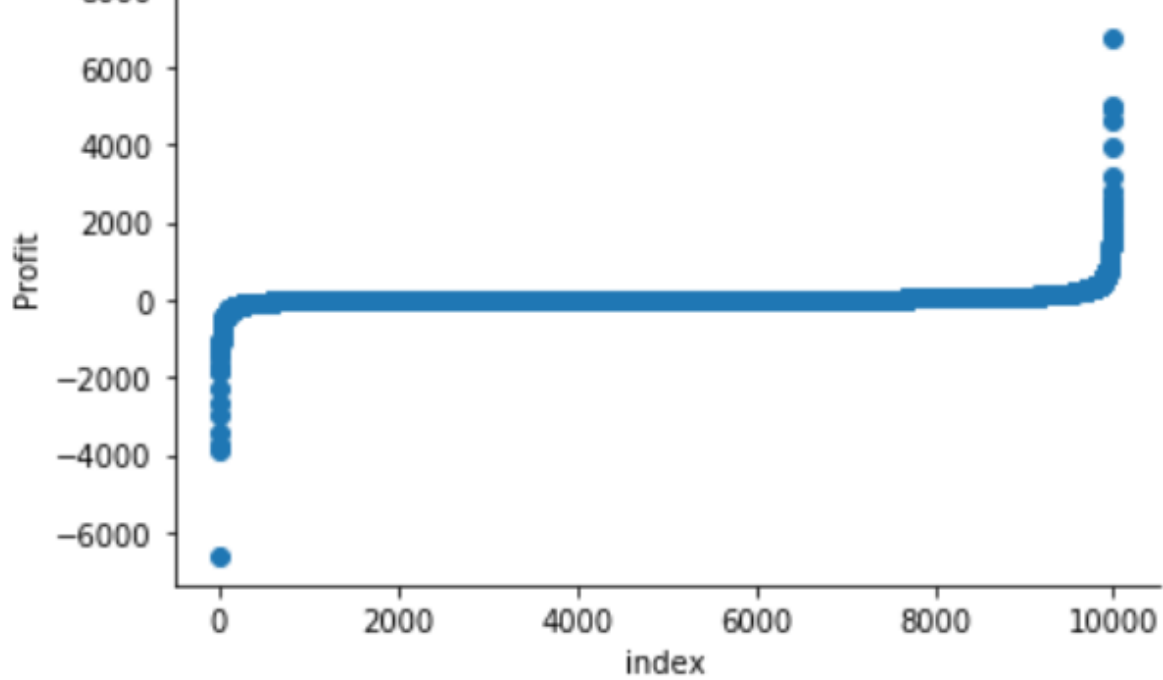


Figure 5

```
sns.distplot(df['Profit'])
plt.title("Distribution of Profit")
sns.despine()
```

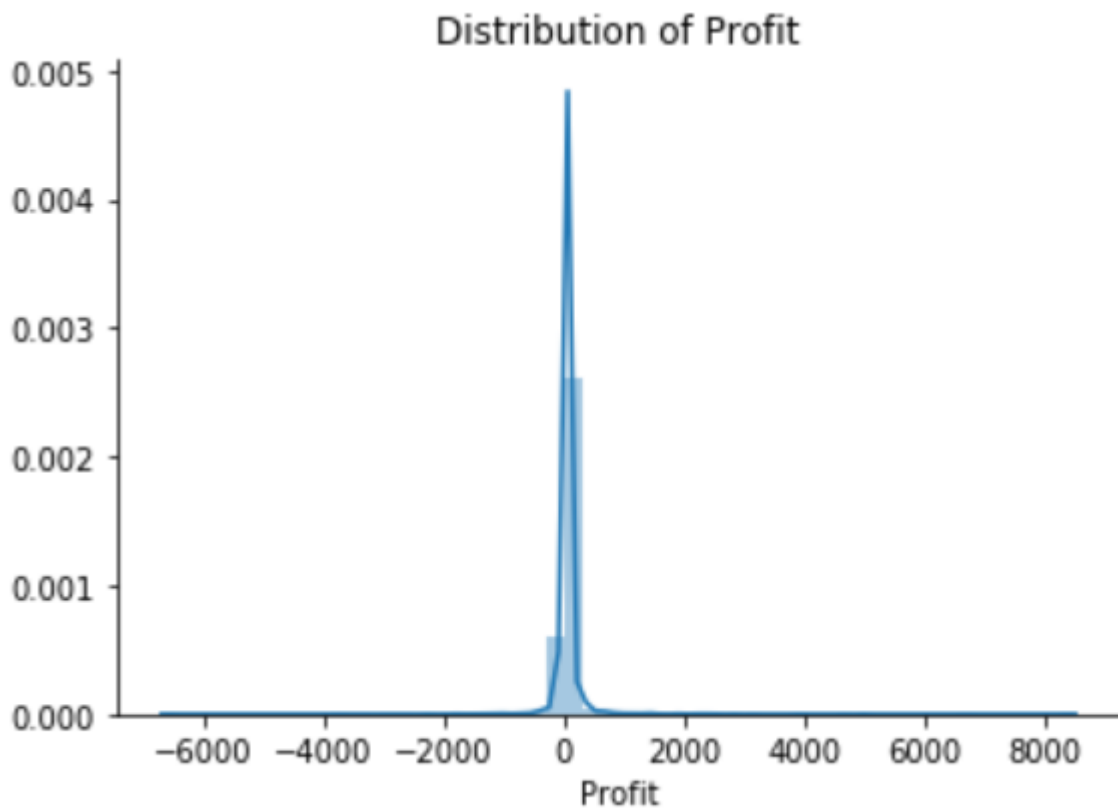


Figure 6

```
print("Skewness: %f" % df['Profit'].skew())
print("Kurtosis: %f" % df['Profit'].kurt())
```

Skewness: 7.561432
Kurtosis: 397.188515

The Superstore's Profit distribution has both a positive tail and negative tail. However, the positive tail is longer than the negative tail. So the distribution is positive skewed, and the data are heavy-tailed or profusion of outliers.

There are two regions where the data has low probability to appear: one on the right side of the distribution, another one on the left.

Univariate Anomaly Detection on Sales

Isolation Forest is an algorithm to detect outliers that returns the anomaly score of each sample using the IsolationForest algorithm which is based on the fact that anomalies are data points that are few and different. Isolation Forest is a tree-based model. In these trees, partitions are created by first randomly selecting a feature and then selecting a random split value between the minimum and maximum value of the selected feature.

The following process shows how IsolationForest behaves in the case of the Susperstore's sales, and the algorithm was implemented in Sklearn and the code was largely borrowed from this tutorial

- Trained IsolationForest using the Sales data.
- Store the Sales in the *NumPy* array for using in our models later.
- Computed the anomaly score for each observation. The anomaly score of an input sample is computed as the mean anomaly score of the trees in the forest.
- Classified each observation as an outlier or non-outlier.
- The visualization highlights the regions where the outliers fall.

```
1 isolation_forest = IsolationForest(n_estimators=100)
2 isolation_forest.fit(df['Sales'].values.reshape(-1, 1))
3 xx = np.linspace(df['Sales'].min(), df['Sales'].max(), len(df)).reshape(-1,1)
4 anomaly_score = isolation_forest.decision_function(xx)
5 outlier = isolation_forest.predict(xx)
6 plt.figure(figsize=(10,4))
7 plt.plot(xx, anomaly_score, label='anomaly score')
8 plt.fill_between(xx.T[0], np.min(anomaly_score), np.max(anomaly_score),
9                  where=outlier== -1, color='r',
10                  alpha=.4, label='outlier region')
```

```

11 plt.legend()
12 plt.ylabel('anomaly score')
13 plt.xlabel('Sales')
14 plt.show();

```

sales_IsolationForest.py hosted with ❤ by GitHub

[view raw](#)

sales_IsolationForest.py

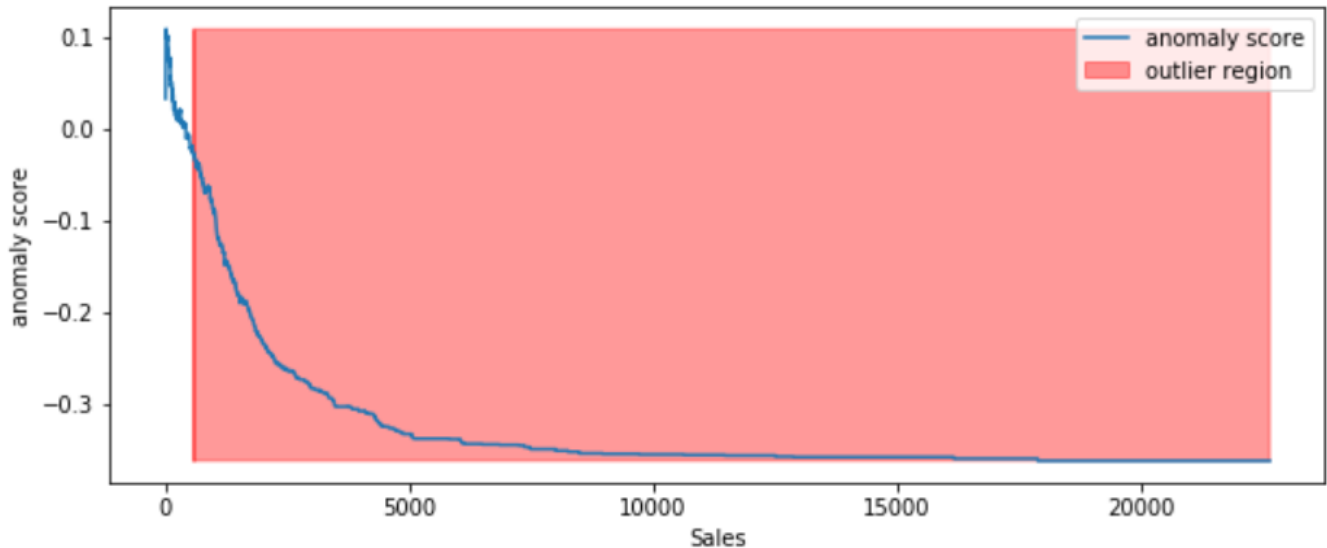


Figure 7

According to the above results and visualization, It seems that Sales that exceeds 1000 would be definitely considered as an outlier.

Visually investigate one anomaly

```
df.iloc[10]
```

Row ID	11
Order ID	CA-2014-115812
Order Date	2014-06-09 00:00:00
Ship Date	2014-06-14 00:00:00
Ship Mode	Standard Class
Customer ID	BH-11710
Customer Name	Brosina Hoffman
Segment	Consumer
Country	United States
City	Los Angeles
State	California
Postal Code	90032
Region	West
Product ID	FUR-TA-10001539

Category	Furniture
Sub-Category	Tables
Product Name	Chromcraft Rectangular Conference Tables
Sales	1706.18
Quantity	9
Discount	0.2
Profit	85.3092
Name: 10, dtype: object	

Figure 8

This purchase seems normal to me expect it was a larger amount of sales compared with the other orders in the data.

Univariate Anomaly Detection on Profit

- Trained IsolationForest using the Profit variable.
- Store the Profit in the *NumPy* array for using in our models later.
- Computed the anomaly score for each observation. The anomaly score of an input sample is computed as the mean anomaly score of the trees in the forest.
- Classified each observation as an outlier or non-outlier.
- The visualization highlights the regions where the outliers fall.

```

1  isolation_forest = IsolationForest(n_estimators=100)
2  isolation_forest.fit(df['Profit'].values.reshape(-1, 1))
3  xx = np.linspace(df['Profit'].min(), df['Profit'].max(), len(df)).reshape(-1,1)
4  anomaly_score = isolation_forest.decision_function(xx)
5  outlier = isolation_forest.predict(xx)
6  plt.figure(figsize=(10,4))
7  plt.plot(xx, anomaly_score, label='anomaly score')
8  plt.fill_between(xx.T[0], np.min(anomaly_score), np.max(anomaly_score),
9                  where=outlier==1, color='r',
10                 alpha=.4, label='outlier region')
11 plt.legend()
12 plt.ylabel('anomaly score')
13 plt.xlabel('Profit')
14 plt.show();

```

profit_IsolationForest.py hosted with ❤ by GitHub

[view raw](#)

profit_IsolationForest.py



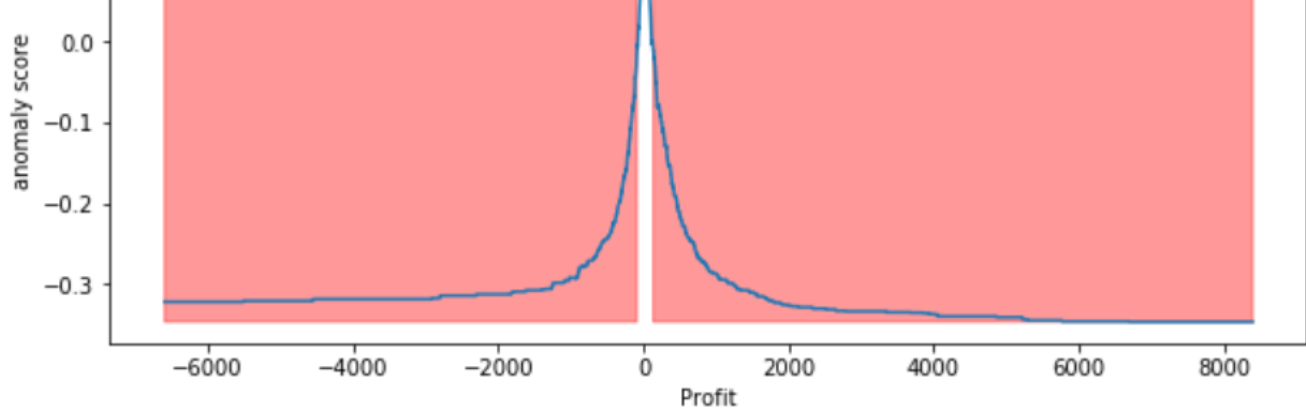


Figure 9

Visually investigate some of the anomalies

According to the above results and visualization, It seems that Profit that below -100 or exceeds 100 would be considered as an outlier, let's visually examine one example each that determined by our model and to see whether they make sense.

```
df.iloc[3]
```

Row ID	4
Order ID	US-2015-108966
Order Date	2015-10-11 00:00:00
Ship Date	2015-10-18 00:00:00
Ship Mode	Standard Class
Customer ID	SO-20335
Customer Name	Sean O'Donnell
Segment	Consumer
Country	United States
City	Fort Lauderdale
State	Florida
Postal Code	33311
Region	South
Product ID	FUR-TA-10000577
Category	Furniture
Sub-Category	Tables
Product Name	Bretford CR4500 Series Slim Rectangular Table
Sales	957.577
Quantity	5
Discount	0.45
Profit	-383.031
outlier	0

Name: 3, dtype: object

Figure 10

Any negative profit would be an anomaly and should be further investigate, this goes without saying

```
df.iloc[1]
```

Row ID	2
Order ID	CA-2016-152156
Order Date	2016-11-08 00:00:00
Ship Date	2016-11-11 00:00:00
Ship Mode	Second Class
Customer ID	CG-12520
Customer Name	Claire Gute
Segment	Consumer
Country	United States
City	Henderson
State	Kentucky
Postal Code	42420
Region	South
Product ID	FUR-CH-10000454
Category	Furniture
Sub-Category	Chairs
Product Name	Hon Deluxe Fabric Upholstered Stacking Chairs,...
Sales	731.94
Quantity	3
Discount	0
Profit	219.582
Name: 1, dtype: object	

Figure 11

Our model determined that this order with a large profit is an anomaly. However, when we investigate this order, it could be just a product that has a relatively high margin.

The above two visualizations show the anomaly scores and highlighted the regions where the outliers are. As expected, the anomaly score reflects the shape of the underlying distribution and the outlier regions correspond to low probability areas.

However, Univariate analysis can only get us thus far. We may realize that some of these anomalies that determined by our models are not the anomalies we expected. When our data is multidimensional as opposed to **univariate**, the approaches to **anomaly detection** become more computationally intensive and more mathematically complex.

Multivariate Anomaly Detection

Most of the analysis that we end up doing are **multivariate** due to complexity of the world we are living in. In **multivariate** anomaly detection, outlier is a combined unusual score on at least two variables.

So, using the Sales and Profit variables, we are going to build an unsupervised **multivariate anomaly detection** method based on several models.

We are using **PyOD** which is a Python library for detecting anomalies in multivariate data. The library was developed by Yue Zhao.

Sales & Profit

When we are in business, we expect that Sales & Profit are positive correlated. If some of the Sales data points and Profit data points are not positive correlated, they would be considered as outliers and need to be further investigated.

```
sns.regplot(x="Sales", y="Profit", data=df)
sns.despine();
```

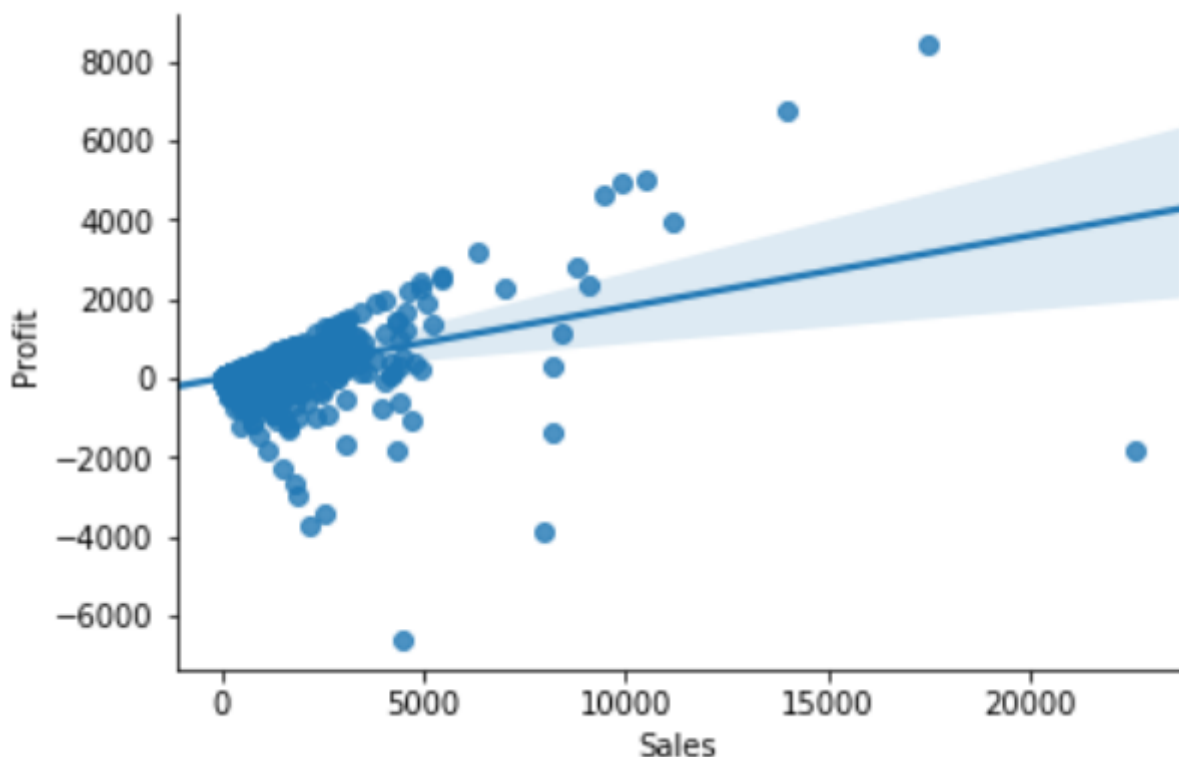


Figure 12

From the above correlation chart, we can see that some of the data points are obvious outliers such as extreme low and extreme high values.

Cluster-based Local Outlier Factor (CBLOF)

The CBLOF calculates the outlier score based on cluster-based local outlier factor. An anomaly score is computed by the distance of each instance to its cluster center multiplied by the instances belonging to its cluster. PyOD library includes the CBLOF implementation.

The following code are borrowed from PyOD tutorial combined with this article.

- Scaling Sales and Profit to between zero and one.
- Arbitrarily set outliers fraction as 1% based on trial and best guess.
- Fit the data to the CBLOF model and predict the results.
- Use threshold value to consider a data point is inlier or outlier.
- Use decision function to calculate the anomaly score for every point.

```
1 outliers_fraction = 0.01
2 xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
3 clf = CBLOF(contamination=outliers_fraction, check_estimator=False, random_state=0)
4 clf.fit(X)
5 scores_pred = clf.decision_function(X) * -1
6 y_pred = clf.predict(X)
7 n_inliers = len(y_pred) - np.count_nonzero(y_pred)
8 n_outliers = np.count_nonzero(y_pred == 1)
9
10 plt.figure(figsize=(8, 8))
11
12 df1 = df
13 df1['outlier'] = y_pred.tolist()
14
15 # sales - inlier feature 1, profit - inlier feature 2
16 inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-1,1)
17 inliers_profit = np.array(df1['Profit'][df1['outlier'] == 0]).reshape(-1,1)
18
19 # sales - outlier feature 1, profit - outlier feature 2
20 outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-1,1)
21 outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-1,1)
22
23 print('OUTLIERS:',n_outliers,'INLIERS:',n_inliers)
24 threshold = percentile(scores_pred, 100 * outliers_fraction)
25 Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
26 Z = Z.reshape(xx.shape)
27
28 plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold, 7), cmap=plt.cm.Blues_r)
29 a = plt.contour(xx, yy, Z, levels=[threshold], linewidths=2, colors='red')
30 plt.contourf(xx, yy, Z, levels=[threshold, Z.max()], colors='orange')
31 b = plt.scatter(inliers_sales, inliers_profit, c='white', s=20, edgecolor='k')
32
```

```

33 c = plt.scatter(outliers_sales, outliers_profit, c='black',s=20, edgecolor='k')
34
35 plt.axis('tight')
36 plt.legend([a.collections[0], b,c], ['learned decision function', 'inliers','outliers'],
37           prop=matplotlib.font_manager.FontProperties(size=20),loc='lower right')
38 plt.xlim((0, 1))
39 plt.ylim((0, 1))
40 plt.title('Cluster-based Local Outlier Factor (CBLOF)')
41 plt.show();

```

CBLOF.py hosted with ❤ by GitHub

[view raw](#)

CBLOF.py

OUTLIERS: 100 INLIERS: 9894

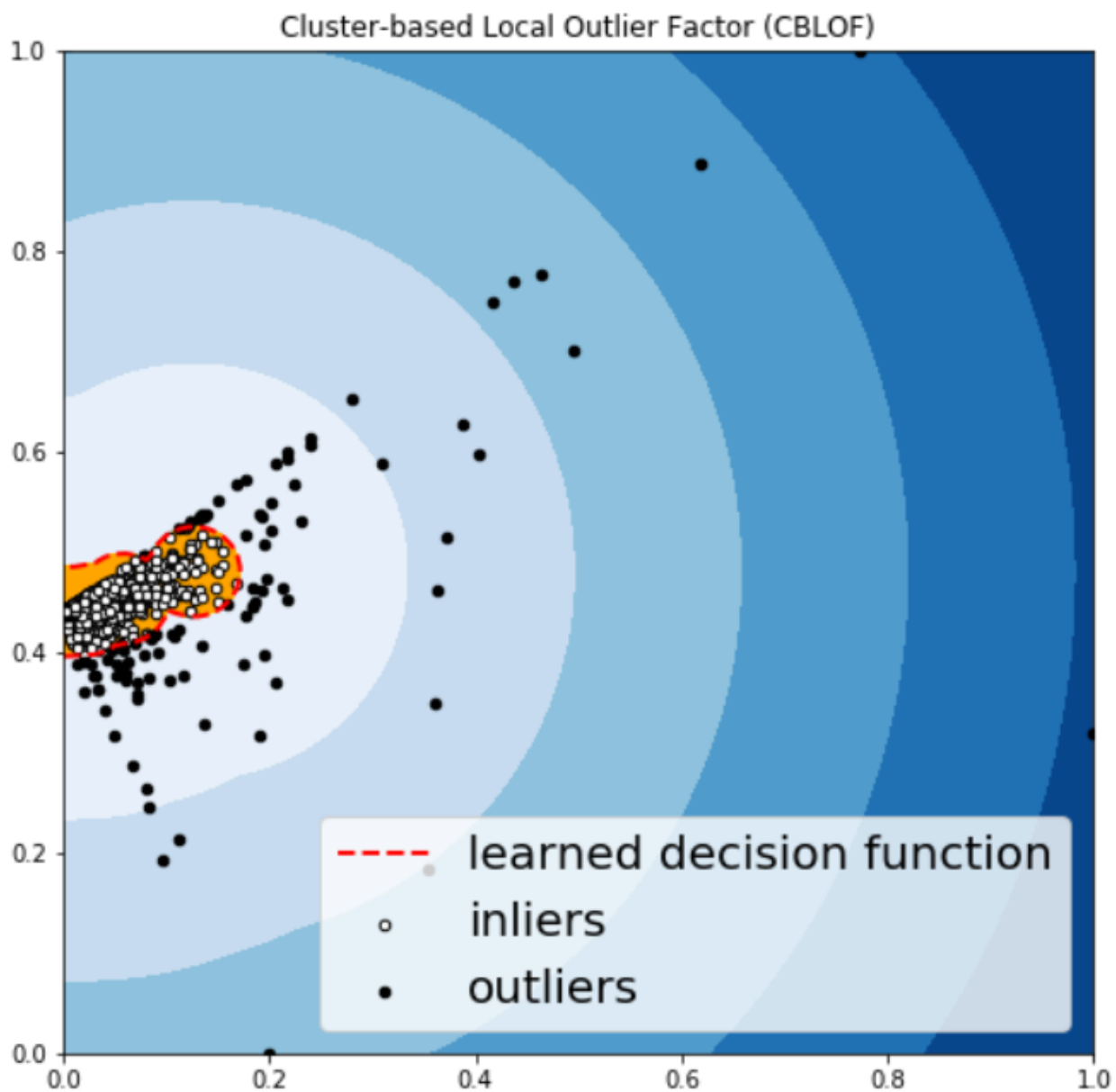


Figure 13

Histogram-based Outlier Detection (HBOS)

HBOS assumes the feature independence and calculates the degree of anomalies by building histograms. In multivariate anomaly detection, a histogram for each single

feature can be computed, scored individually and combined at the end. When using PyOD library, the code are very similar with the CBLOF.

```
1 outliers_fraction = 0.01
2 xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
3 clf = HBOS(contamination=outliers_fraction)
4 clf.fit(X)
5 scores_pred = clf.decision_function(X) * -1
6 y_pred = clf.predict(X)
7 n_inliers = len(y_pred) - np.count_nonzero(y_pred)
8 n_outliers = np.count_nonzero(y_pred == 1)
9 plt.figure(figsize=(8, 8))
10 df1 = df
11 df1['outlier'] = y_pred.tolist()
12
13 # sales - inlier feature 1, profit - inlier feature 2
14 inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-1,1)
15 inliers_profit = np.array(df1['Profit'][df1['outlier'] == 0]).reshape(-1,1)
16
17 # sales - outlier feature 1, profit - outlier feature 2
18 outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-1,1)
19 outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-1,1)
20
21 print('OUTLIERS:',n_outliers,'INLIERS:',n_inliers)
22 threshold = percentile(scores_pred, 100 * outliers_fraction)=
23 Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
24 Z = Z.reshape(xx.shape)
25
26 plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold, 7),cmap=plt.cm.Blues_r)
27 a = plt.contour(xx, yy, Z, levels=[threshold],linewidths=2, colors='red')
28 plt.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='orange')
29 b = plt.scatter(inliers_sales, inliers_profit, c='white',s=20, edgecolor='k')
30
31 c = plt.scatter(outliers_sales, outliers_profit, c='black',s=20, edgecolor='k')
32
33 plt.axis('tight')
34 plt.legend([a.collections[0], b,c], ['learned decision function', 'inliers','outliers'],
35          prop=matplotlib.font_manager.FontProperties(size=20),loc='lower right')
36 plt.xlim((0, 1))
37 plt.ylim((0, 1))
38 plt.title('Histogram-base Outlier Detection (HBOS)')
39 plt.show();
```

HBOS.py hosted with ❤ by GitHub

[view raw](#)

HBOS.py

OUTLIERS: 90 INLIERS: 9904

Histogram-base Outlier Detection (HBOS)

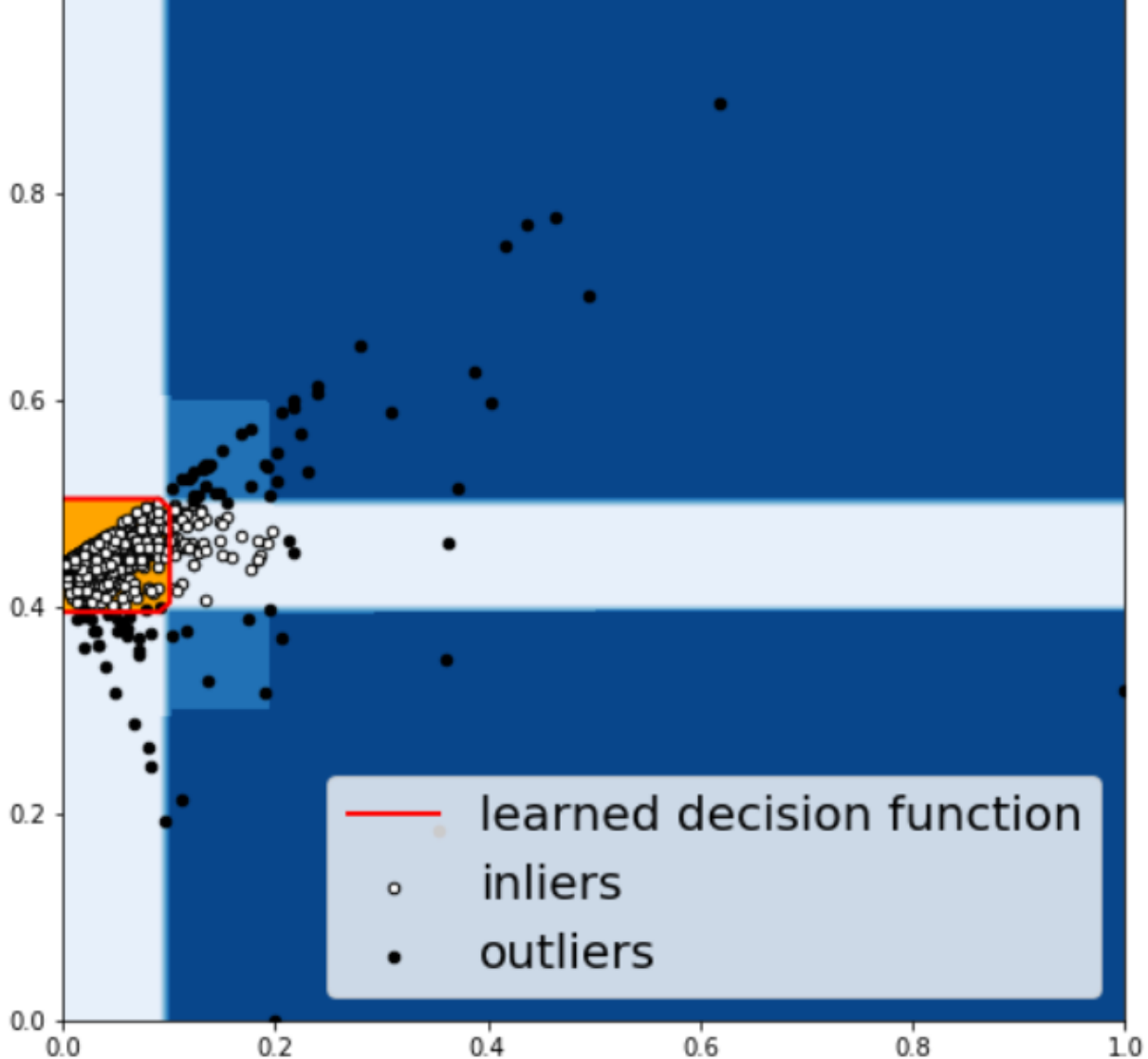


Figure 14

Isolation Forest

Isolation Forest is similar in principle to Random Forest and is built on the basis of decision trees. Isolation Forest isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature.

The PyOD Isolation Forest module is a wrapper of Scikit-learn Isolation Forest with more functionalities.

```

1  outliers_fraction = 0.01
2  xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
3  clf = IForest(contamination=outliers_fraction,random_state=0)
4  clf.fit(X)
5  scores_pred = clf.decision_function(X) * -1
6
7  y_pred = clf.predict(X)
8  n_inliers = len(y_pred) - np.count_nonzero(y_pred)
9  n_outliers = np.count_nonzero(y_pred == 1)
10 plt.figure(figsize=(8, 8))
11

```

```

12 df1 = df
13 df1['outlier'] = y_pred.tolist()
14
15 # sales - inlier feature 1, profit - inlier feature 2
16 inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-1,1)
17 inliers_profit = np.array(df1['Profit'][df1['outlier'] == 0]).reshape(-1,1)
18
19 # sales - outlier feature 1, profit - outlier feature 2
20 outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-1,1)
21 outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-1,1)
22
23 print('OUTLIERS: ',n_outliers,'INLIERS: ',n_inliers)
24
25 threshold = percentile(scores_pred, 100 * outliers_fraction)
26 Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
27 Z = Z.reshape(xx.shape)
28 plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold, 7), cmap=plt.cm.Blues_r)
29 a = plt.contour(xx, yy, Z, levels=[threshold], linewidths=2, colors='red')
30 plt.contourf(xx, yy, Z, levels=[threshold, Z.max()], colors='orange')
31 b = plt.scatter(inliers_sales, inliers_profit, c='white', s=20, edgecolor='k')
32
33 c = plt.scatter(outliers_sales, outliers_profit, c='black', s=20, edgecolor='k')
34
35 plt.axis('tight')
36 plt.legend([a.collections[0], b,c], ['learned decision function', 'inliers','outliers'],
37          prop=matplotlib.font_manager.FontProperties(size=20),loc='lower right')
38 plt.xlim((0, 1))
39 plt.ylim((0, 1))
40 plt.title('Isolation Forest')
41 plt.show();

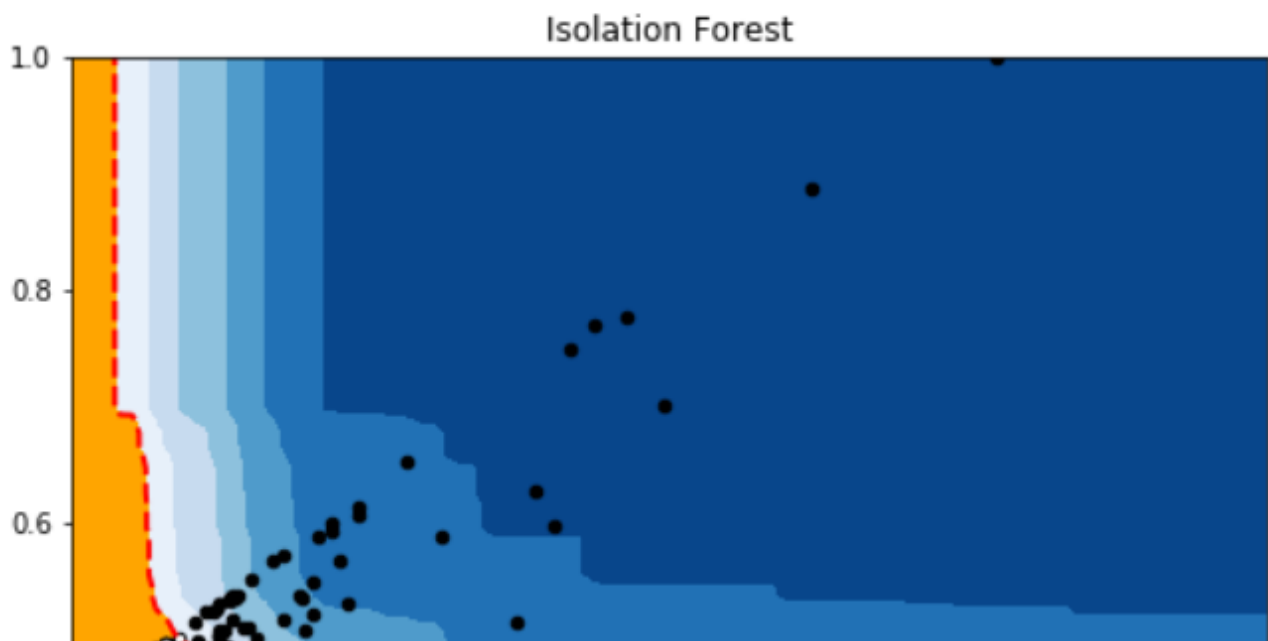
```

IsolationForest.py hosted with ❤ by GitHub

[view raw](#)

IsolationForest.Py

OUTLIERS: 100 INLIERS: 9894



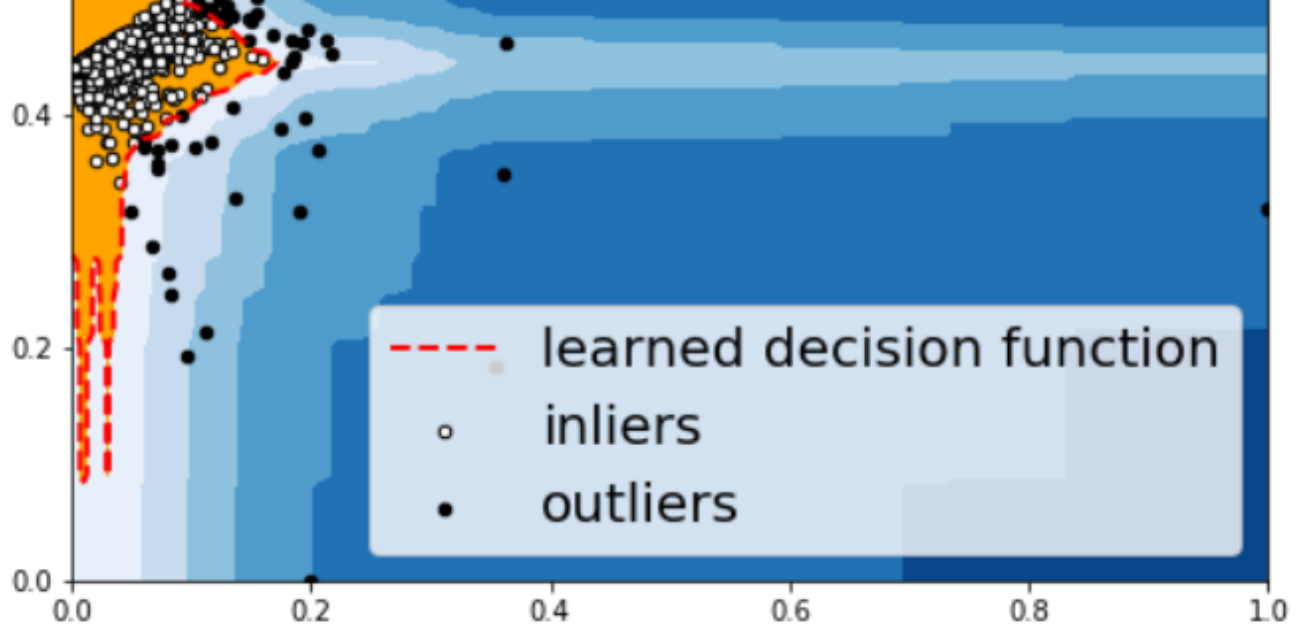


Figure 15

K - Nearest Neighbors (KNN)

KNN is one of the simplest methods in anomaly detection. For a data point, its distance to its k th nearest neighbor could be viewed as the outlier score.

```

1  outliers_fraction = 0.01
2  xx , yy = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
3  clf = KNN(contamination=outliers_fraction)
4  clf.fit(X)
5  scores_pred = clf.decision_function(X) * -1
6  y_pred = clf.predict(X)
7  n_inliers = len(y_pred) - np.count_nonzero(y_pred)
8  n_outliers = np.count_nonzero(y_pred == 1)
9  plt.figure(figsize=(8, 8))
10
11 df1 = df
12 df1['outlier'] = y_pred.tolist()
13
14 # sales - inlier feature 1, profit - inlier feature 2
15 inliers_sales = np.array(df1['Sales'][df1['outlier'] == 0]).reshape(-1,1)
16 inliers_profit = np.array(df1['Profit'][df1['outlier'] == 0]).reshape(-1,1)
17
18 # sales - outlier feature 1, profit - outlier feature 2
19 outliers_sales = df1['Sales'][df1['outlier'] == 1].values.reshape(-1,1)
20 outliers_profit = df1['Profit'][df1['outlier'] == 1].values.reshape(-1,1)
21
22 print('OUTLIERS: ',n_outliers,'INLIERS: ',n_inliers)
23
24 threshold = percentile(scores_pred, 100 * outliers_fraction)
25
26 Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
27 Z = Z.reshape(xx.shape)
28

```

```

29 plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold, 7), cmap=plt.cm.Blues_r)
30 a = plt.contour(xx, yy, Z, levels=[threshold], linewidths=2, colors='red')
31 plt.contourf(xx, yy, Z, levels=[threshold, Z.max()], colors='orange')
32 b = plt.scatter(inliers_sales, inliers_profit, c='white', s=20, edgecolor='k')
33 c = plt.scatter(outliers_sales, outliers_profit, c='black', s=20, edgecolor='k')
34 plt.axis('tight')
35 plt.legend([a.collections[0], b, c], ['learned decision function', 'inliers', 'outliers'],
36           prop=matplotlib.font_manager.FontProperties(size=20), loc='lower right')
37 plt.xlim((0, 1))
38 plt.ylim((0, 1))
39 plt.title('K Nearest Neighbors (KNN)')
40 plt.show();

```

KNN.py hosted with ❤ by GitHub

[view raw](#)

KNN.py

OUTLIERS: 91 INLIERS: 9903

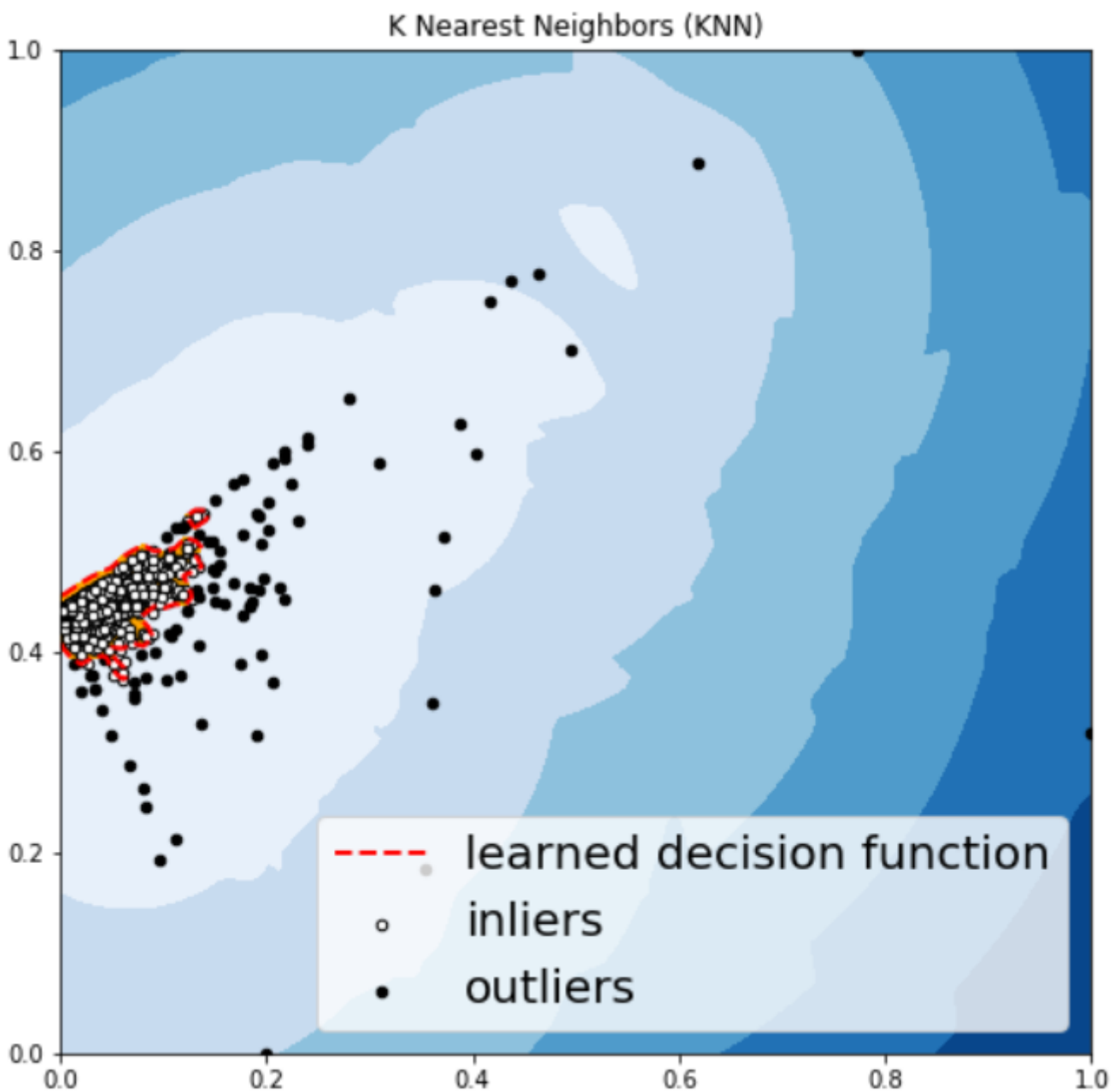


Figure 16

The anomalies predicted by the above four algorithms were not very different.

Visually investigate some of the anomalies

We may want to investigate each of the outliers that determined by our model, for example, let's look in details for a couple of outliers that determined by KNN, and try to understand what make them anomalies.

```
df.iloc[1995]
```

Row ID	1996
Order ID	US-2017-147221
Order Date	2017-12-02 00:00:00
Ship Date	2017-12-04 00:00:00
Ship Mode	Second Class
Customer ID	JS-16030
Customer Name	Joy Smith
Segment	Consumer
Country	United States
City	Houston
State	Texas
Postal Code	77036
Region	Central
Product ID	OFF-AP-10002534
Category	Office Supplies
Sub-Category	Appliances
Product Name	3.6 Cubic Foot Counter Height Office Refrigerator
Sales	294.62
Quantity	5
Discount	0.8
Profit	-766.012
Name: 1995, dtype: object	

Figure 17

For this particular order, a customer purchased 5 products with total price at 294.62 and profit at lower than -766, with 80% discount. It seems like a clearance. We should be aware of the loss for each product we sell.

```
df.iloc[9649]
```

Row ID	9650
Order ID	CA-2016-107104

Order Date	2016-11-26 00:00:00
Ship Date	2016-11-30 00:00:00
Ship Mode	Standard Class
Customer ID	MS-17365
Customer Name	Maribeth Schnelling
Segment	Consumer
Country	United States
City	Los Angeles
State	California
Postal Code	90045
Region	West
Product ID	FUR-BO-10002213
Category	Furniture
Sub-Category	Bookcases
Product Name	DMI Eclipse Executive Suite Bookcases
Sales	3406.66
Quantity	8
Discount	0.15
Profit	160.314
Name: 9649, dtype: object	

Figure 18

For this purchase, it seems to me that the profit at around 4.7% is too small and the model determined that this order is an anomaly.

```
df.iloc[9270]
```

Row ID	9271
Order ID	US-2017-102183
Order Date	2017-08-21 00:00:00
Ship Date	2017-08-28 00:00:00
Ship Mode	Standard Class
Customer ID	PK-19075
Customer Name	Pete Kriz
Segment	Consumer
Country	United States
City	New York City
State	New York
Postal Code	10035
Region	East
Product ID	OFF-BI-10001359

Category	Office Supplies
Sub-Category	Binders
Product Name	GBC DocuBind TL300 Electric Binding System
Sales	4305.55
Quantity	6
Discount	0.2
Profit	1453.12

Name: 9270, dtype: object

Figure 19

For the above order, a customer purchased 6 product at 4305 in total price, after 20% discount, we still get over 33% of the profit. We would love to have more of these kind of anomalies.

Jupyter notebook for the above analysis can be found on Github. Enjoy the rest of the week.

[Data Science](#) [Machine Learning](#) [Anomaly Detection](#) [Machine Learning Tutorial](#)

[Unsupervised Learning](#)

[About](#) [Help](#) [Legal](#)

Get the Medium app

