Introduction to Isolation Forest

Salvatore Di Carlo

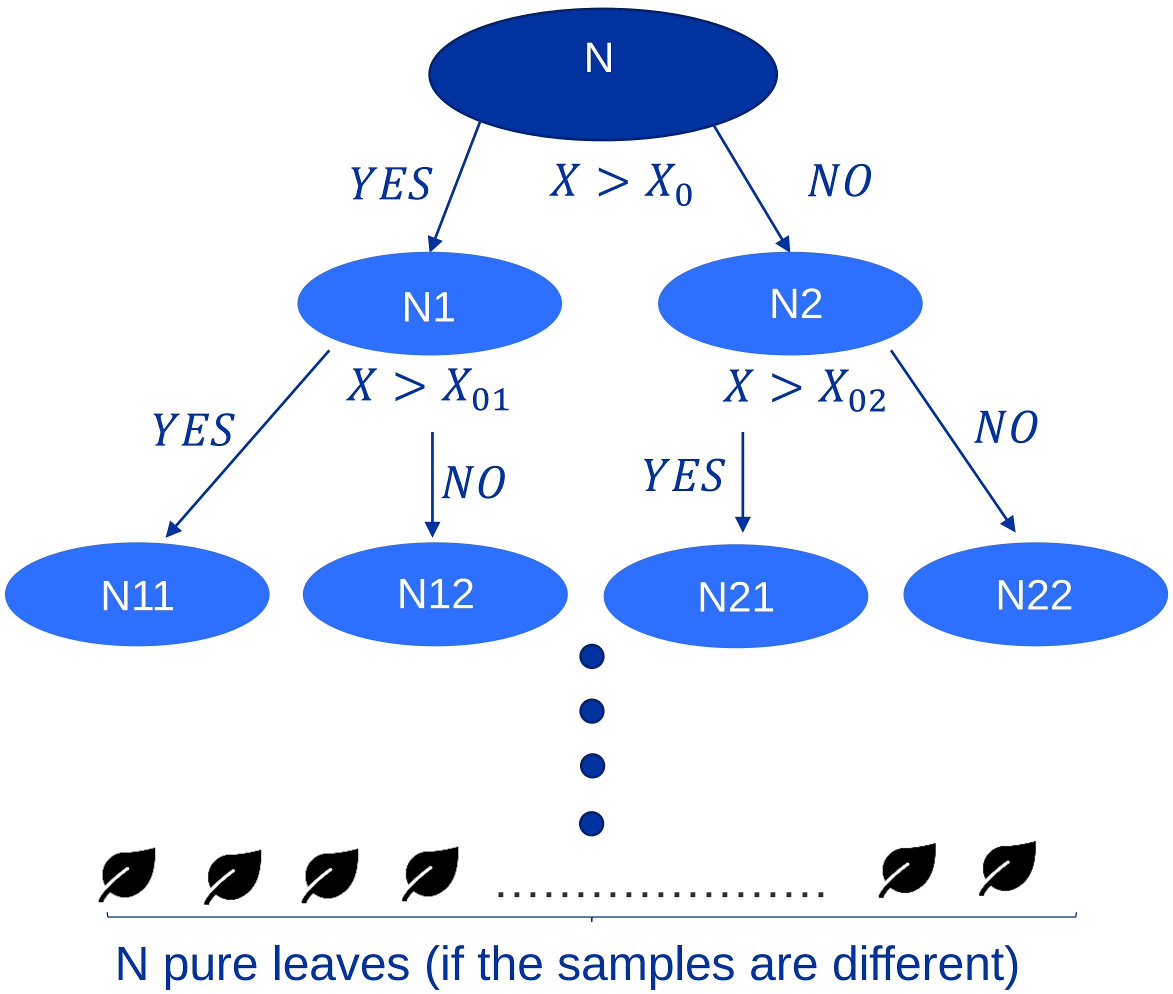
16/12/2019

Outline

- **What is the Isolation forest technique:**
  - Decision trees
  - Random forest
  - Isolation forest
- **Why it is useful:**
  - Example: wire scanner calibrations
  - Outliers detection
- **How to use it:**
  - Code snippet
- **Conclusion**

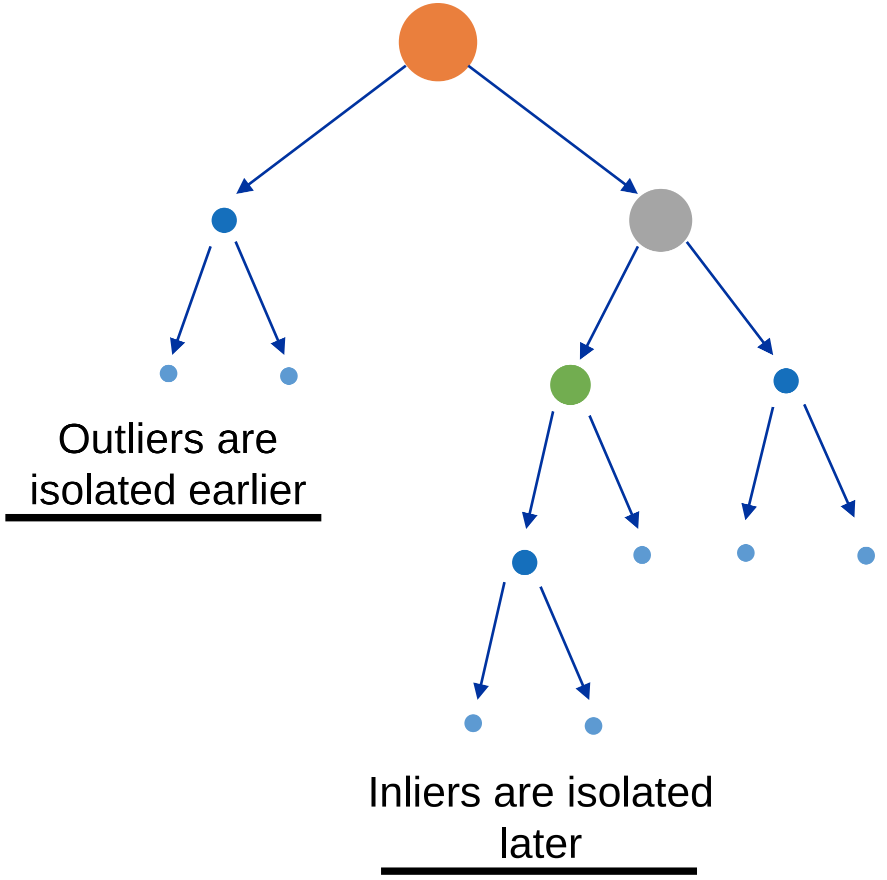
Decision trees

- **Used for classification/regression.**
- **Learns a hierarchy of if/else tests.**
- **The N samples will be split until each leaf contains different values of the feature X.**
- **The argument generalizes to dataset containing several features X1,X2,X3…**
- **The algorithms chooses the split that minimizes the variance within the leaves after the split.**
- **Random forests are ensembles of several (different) decision trees and the prediction is the average results.**



The Isolation Forest technique

- The Isolation Forest is a technique for the detection of outlier samples.
- Since outliers have features X that differ significantly from most of the samples, they are isolated earlier in the hierarchy of a decision tree.
- Outliers are detected by setting a threshold on the mean length (number of splits) from the top of the tree downwards.
- The Scikit-learn implementation provides a score for each sample that increases from -1 to +1 with the number of splits.



Example: Wire Scanners calibrations

Wire scanners are calibrated by using a laser in order to transform the angular coordinate α into the cartesian coordinate y.

$$Polynomial\ fitting$$

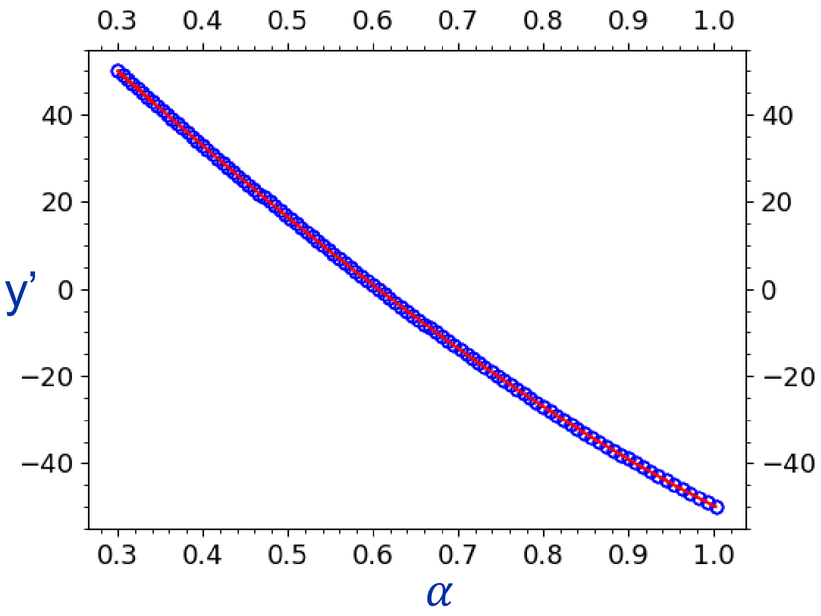
$$y' = x0 + x1\ \alpha + \ldots\ x5\ \alpha^5$$

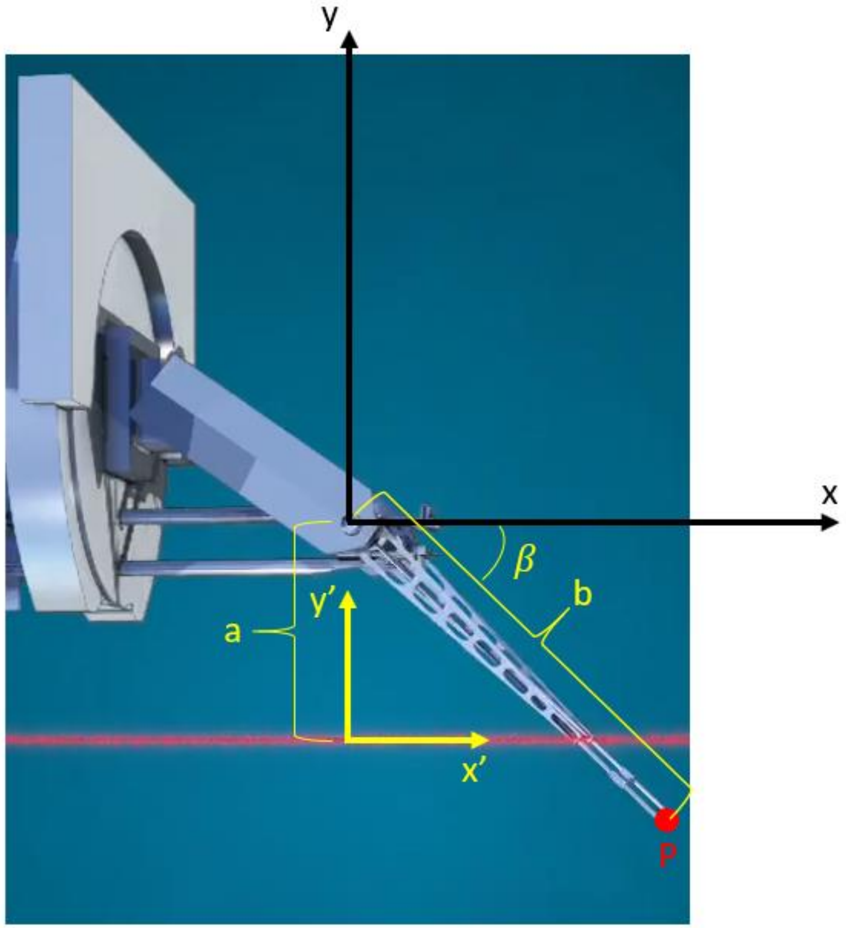
$$Sinusoidal\ fitting$$

$$y' = a + b \sin(\beta)$$

$$\beta = c\alpha + d$$

b represents the fork length

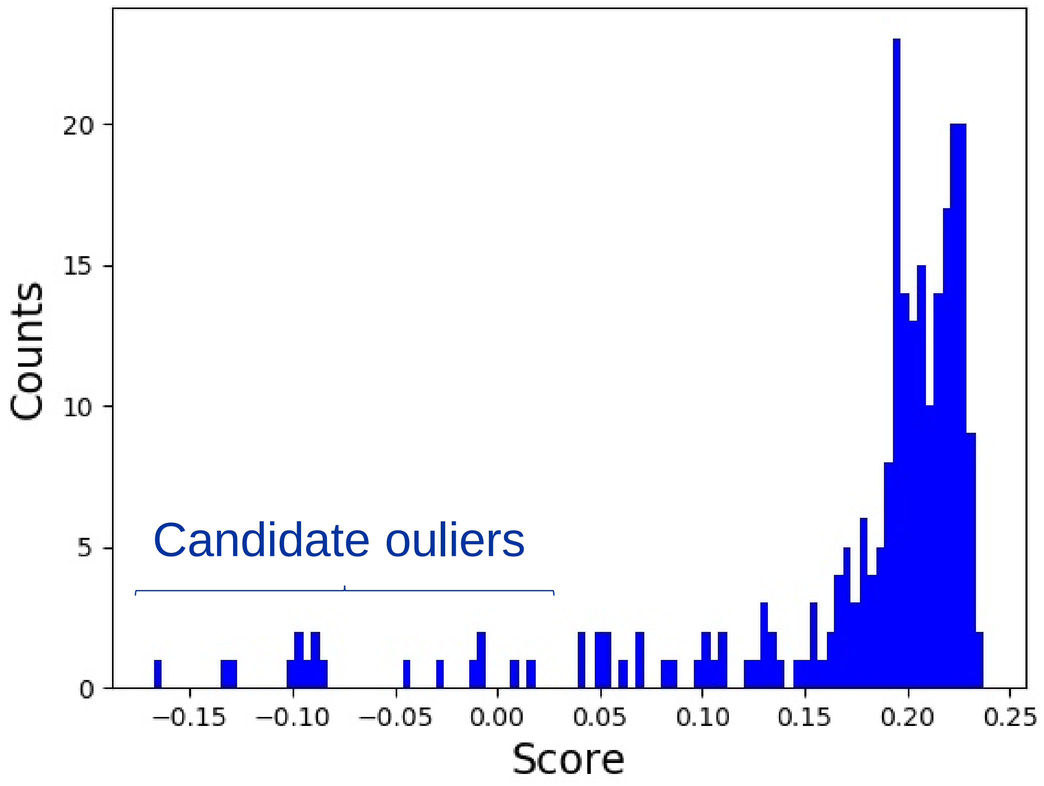




	Av.Speed [rads-1]	Scan	Serial	a	b	c	d	res_rms_fun	res_rms_pol	x0	x1	x2	x3	x4	x5	Serial_Scan
0	126	0	0	-6.949080	45.151601	2.807945	-2.511018	1.778010	14.142109	174.585574	-46.592303	-140.338860	80.131103	-17.117674	1.790317	0
1	120	1	0	-7.276314	44.814874	2.835180	-2.556390	1.783502	15.824297	131.131409	82.308607	-291.433152	167.413441	-41.982387	4.590144	1
2	54	0	1	83.243813	148.229295	1.008467	1.561936	0.003196	0.003142	84.618943	-150.028218	1.121386	22.677435	1.832081	-1.676659	2
3	55	0	1	83.404492	148.389936	1.007407	1.555081	0.001505	0.001317	85.807706	-150.079806	0.712955	22.599064	1.789800	-1.620751	2
4	57	0	1	83.103338	148.068460	1.009334	1.561043	0.003933	0.003834	84.647074	-150.362429	2.392140	20.445845	3.650663	-2.230798	2

Outliers detection

- **The algorithm is applied to our dataset of calibrations and returns a score for each sample.**
- **The sample with lower score are likely to be outliers.**
- **The outlier threshold on the score must be set by the user.**
- **In our case, we can set the threshold in a way that the parameter b from the fitting for the inliers (total dataset without the outliers) is reasonably close to the nominal fork length.**



Inliers/Outliers comparison

- The initial dataset contained 242 samples;
- Choosing a threshold score of 0.04 provides 18 candidate outliers
- The mean of each feature for the inliers is significantly different from the corresponding value for the outliers;
- b (fork length) can should be either about 150 mm or about 180 mm.

INLIERS

	Av.Speed [rads-1]	a	b	c	d	res_rms_fun	res_rms_pol
count	224.000000	224.000000	224.000000	224.000000	224.000000	224.000000	224.000000
mean	93.879464	90.783892	161.658324	1.009472	1.545312	0.012232	0.011963
std	27.826798	9.015463	15.786919	0.009238	0.146737	0.017091	0.016969

OUTLIERS

	Av.Speed [rads-1]	Scan	Serial	a	b	c	d	res_rms_fun	res_rms_pol
0	126	IN	PSB_PXBWSRA005-CR000002	-6.949080	45.151601	2.807945	-2.511018	1.778010	14.142109
1	120	OUT	PSB_PXBWSRA005-CR000002	-7.276314	44.814874	2.835180	-2.556390	1.783502	15.824297
66	123	IN	PSB_PXBWSRA005-CR000006	56.274344	117.166833	1.212458	1.339496	0.122250	1.371147
74	53	IN	PSB_PXBWSRA005-CR000007	12.296238	57.868795	2.381853	0.349101	1.480598	4.753168
75	53	IN	PSB_PXBWSRA005-CR000007	-16.377113	36.291998	4.099571	-1.572101	6.806953	14.810047
81	102	IN	PSB_PXBWSRA005-CR000007	-21.596523	37.521772	4.212057	-1.873593	9.329634	25.449834
86	112	IN	PSB_PXBWSRA005-CR000007	-21.127661	37.151954	4.423333	-2.016384	9.497816	29.187143
92	59	OUT	PSB_PXBWSRA005-CR000007	-16.379993	36.291681	4.099854	-1.571310	6.809331	14.815095
98	106	OUT	PSB_PXBWSRA005-CR000007	-21.563820	37.374712	4.318452	-1.951819	9.523586	27.950754
103	122	OUT	PSB_PXBWSRA005-CR000007	-21.615759	37.527720	4.212631	-1.871267	9.341669	25.516649
142	110	IN	PS_PXBWSRB011-CR000001	100.594327	179.214612	1.012507	0.799985	0.076210	0.075414
146	57	OUT	PS_PXBWSRB011-CR000001	97.175192	175.315330	1.028967	0.770325	0.023056	0.022058
148	107	OUT	PS_PXBWSRB011-CR000001	112.164005	191.693287	0.964516	0.895065	0.098010	0.092833
149	115	OUT	PS_PXBWSRB011-CR000001	102.035065	180.732285	1.006146	0.817653	0.019974	0.019211
152	116	IN	PS_PXBWSRB011-CR000003	95.899272	173.410579	1.036613	1.527444	0.058205	0.055469
155	127	OUT	PS_PXBWSRB011-CR000003	96.537273	174.022595	1.034126	1.533044	0.059705	0.057539
227	111	OUT	PS_PXBWSRB011-CR000013	126.051426	207.808798	0.904873	1.675835	0.051006	0.044033
237	112	OUT	PS_PXBWSRB011-CR000013	97.445857	175.353764	1.030407	1.537276	0.053843	0.050050

Code snippet

```python
import numpy as np
import pandas as pd

%matplotlib notebook
import matplotlib.pyplot as plt
import matplotlib.ticker as tck

from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import IsolationForest

df = pd.read_csv('calibration_table_cos.csv')
df = df.drop('Unnamed: 0', 1)
X = df.copy()

X['Serial_Scan'] = X['Serial'] + "_" + X['Scan']

cat_cols = ['Scan', 'Serial', 'Serial_Scan']

label_encoder = LabelEncoder()

for col in cat_cols:
    X[col] = label_encoder.fit_transform(X[col])

if_model = IsolationForest(behaviour='new',
                           bootstrap=True,
                           contamination=0.06,
                           max_features=1.0,
                           max_samples=len(X),
                           n_estimators=100,
                           n_jobs=-1,
                           random_state=1,
                           verbose=0,
                           warm_start=False)

if_model.fit(X)

y_pred = if_model.predict(X)
y_scores = if_model.decision_function(X)

outliers_index = np.where(y_scores < 0.04)[0]

df_inliers = df[~df.index.isin(outliers_index)]
df_outliers = df[df.index.isin(outliers_index)]
```

Main model parameters:

- behaviour: this parameter has not effect, is deprecated, and will be removed;
- bootstrap: randomly re-sample initial sample with replacement;
- contamination: % of outliers (if known), no effect on score;
- max_features: # of randomly selected features at each node;
- max_samples: # of samples to draw from the initial samples for each tree;
- n_estimators: # of trees in the forest;
- n_jobs: # of cores used for parallelization, -1 means max available;
- random_state: sets the seed for reproducibility;
- verbose: control the verbosity of the output;
- warm_start: reuse the solution from the previous call and add more trees to the old ensemble.

Conclusion

- Decision trees/forests provide an intuitive and useful tool in machine learning that can be used both for data classification and regression.
- The isolation forest technique is useful to isolate outliers, i.e., samples that significantly differ from the rest of the data.
- As an example we isolated some outlier using the wire scanner calibrations dataset, and the corresponding code snippet was provided.
- A certain degree of arbitrariness remains in setting the threshold score for outliers. Plotting a histogram of the score can help in deciding the optimal value.

home.cern