Ex4.IsolationForest

January 16, 2024

1 4th exercise: Use Isolation Forest for anomaly detection

- Course: AML
- Lecturer: Gernot Heisenberg
- Author of notebook: Finn Heydemann
- Date: 28.10.2023

GENERAL NOTE 1: Please make sure you are reading the entire notebook, since it contains a lot of information on your tasks (e.g. regarding the set of certain paramaters or a specific computational trick), and the written mark downs as well as comments contain a lot of information on how things work together as a whole.
GENERAL NOTE 2: * Please, when commenting source code, just use English language only. * When describing an observation please use English language, too * This applies to all exercises throughout this course.

1.0.1 DESCRIPTION:

This notebook allows you for using the Isolation Forest algorithm for anomaly detection. Isolation Forest is an unsupervised learning algorithm that belongs to the ensemble decision trees family. The following paper explains the details on its theory and implementation.

1.0.2 TASKS:

The tasks that you need to work on within this notebook are always indicated below as bullet points. If a task is more challenging and consists of several steps, this is indicated as well. Make sure you have worked down the task list and commented your doings. This should be done by using markdown. Make sure you don't forget to specify your name and your matriculation number in the notebook.

YOUR TASKS in this exercise are as follows: 1. import the notebook to Google Colab or use your local machine. 2. make sure you specified you name and your matriculation number in the header below my name and date. * set the date too and remove mine. 3. read the entire notebook carefully * add comments whereever you feel it necessary for better understanding * run the notebook for the first time.

4. take the three data sets from exercize 1 and apply the isolation forest to them.

5.

1.1 implement an appropriate visualisation (chart) that renders the result (anomaly={yes,no}) for every data point TOGETHER with the original data point in your data set.

1.1.1 First example

```
[1]: from sklearn.ensemble import IsolationForest
     import numpy as np
     from numpy.random import randn

     np.random.seed(1)
     random_data = np.random.randn(50000,2)  * 20 + 20
```

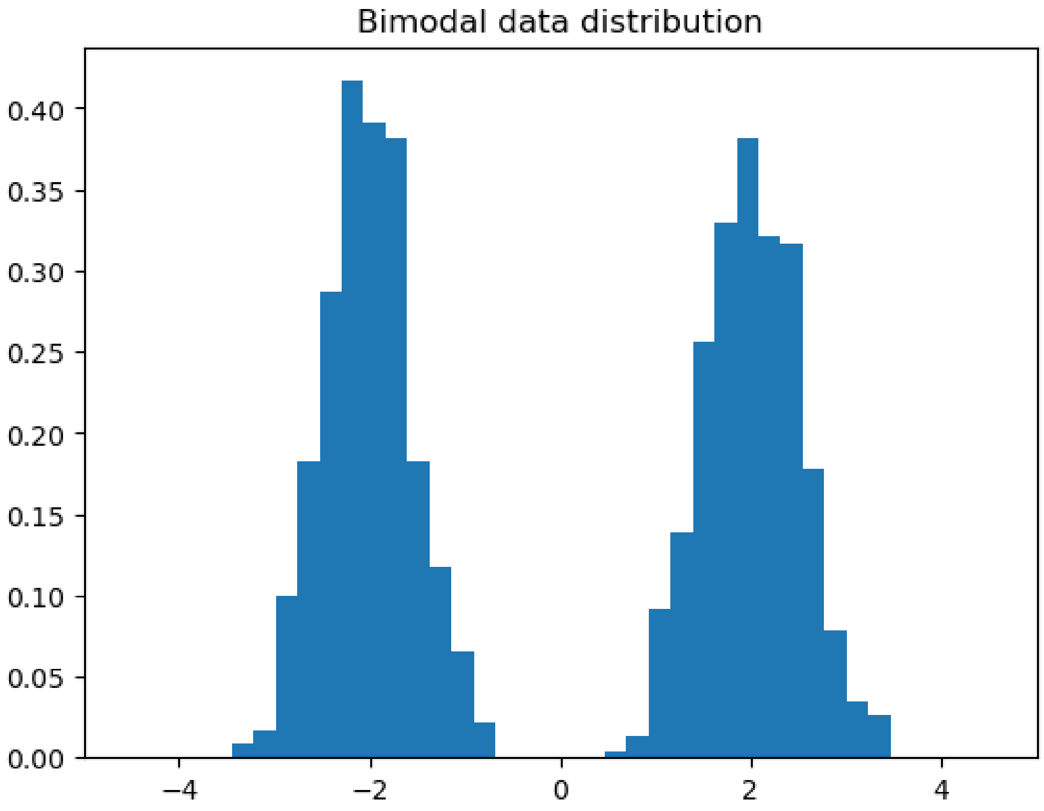
This code will output the predictions for each data point in an array. If the result is -1, it means that this specific data point is an outlier. If the result is 1, then it means that the data point is not an outlier.

```
[6]: #https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.
      ↪IsolationForest.html
     isolation_forest1 = IsolationForest(max_samples=100, random_state = 1,␣
      ↪contamination= 'auto')

     # Note, that fit and predict is called mutually (compare with code below!)
     # Performs a fit on data and returns labels for that data
     outlier_labels = isolation_forest1.fit_predict(random_data)

     outlier_labels[0:200]
```

```
[6]: array([-1,  1, -1, -1,  1, -1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
             1,  1, -1,  1, -1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,
            -1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
             1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1, -1, -1,  1,  1,  1,  1,
             1,  1,  1,  1,  1,  1,  1, -1, -1,  1,  1,  1,  1,  1,  1,  1, -1,
             1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1,  1,  1,  1, -1,
            -1, -1, -1,  1,  1,  1,  1,  1, -1, -1, -1,  1,  1,  1, -1,  1,  1,
            -1,  1,  1,  1,  1, -1,  1, -1,  1, -1,  1, -1,  1,  1,  1,  1,  1,
             1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,
             1,  1,  1, -1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1, -1,
             1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1, -1,  1,  1,  1,
            -1, -1, -1,  1, -1,  1,  1,  1, -1,  1,  1, -1, -1])
```

1.1.2 Second example

[15]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest

# create a bimodal data distribution and visualize it
bimod_data = np.concatenate((np.random.normal(loc=-2,scale=.5,size=500),
                             np.random.normal(loc=2, scale=.5,size=500)))

plt.hist(bimod_data, density=True, bins=30)
plt.xlim([-5, 5])
plt.title("Bimodal data distribution")
plt.show()
```

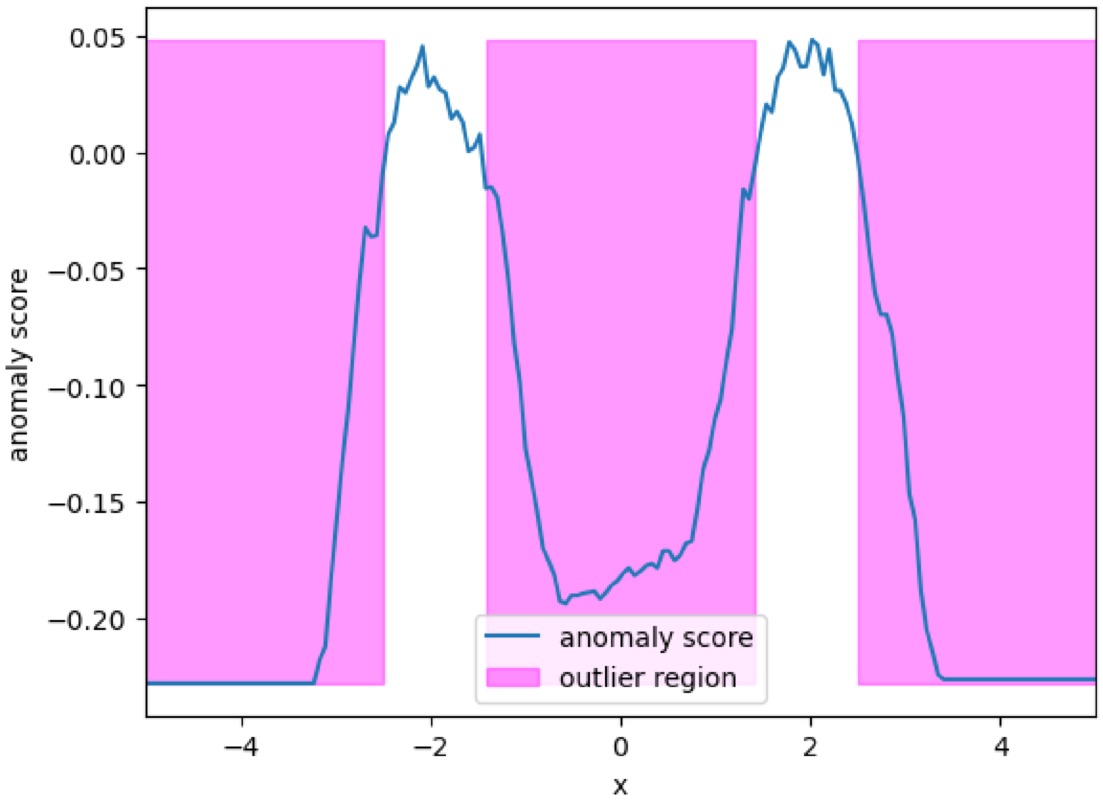


Note, that there are three regions where the data has low probability to appear: * one on the right side of the distribution * another one on the left * and another around zero.

Let's see if the IsolationForest is able to identify these three regions

[8]:
```
# create the IF with n=100 estimators (remember IF is a ensemble method)
isolation_forest2 = IsolationForest(n_estimators=100)
```

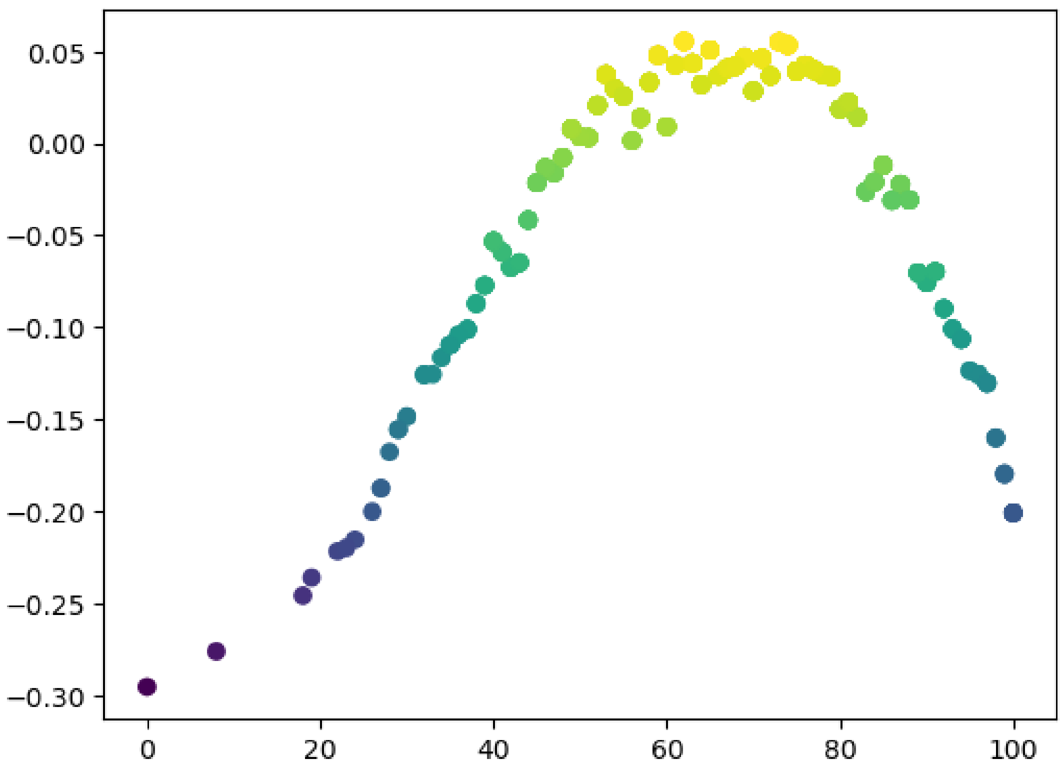
```
isolation_forest2.fit(bimod_data.reshape(-1, 1))

# create a test data set for rasterizing the x-axis the get the anomaly score␣
 ↪for it.
anomaly_score_test_data = np.linspace(-6, 6, 200).reshape(-1,1)

# get the anomaly score (y) for each testing data point on x
anomaly_score = isolation_forest2.decision_function(anomaly_score_test_data)
# classify into an outlier or not (asc < 0 -> outlier)
outlier_labels = isolation_forest2.predict(anomaly_score_test_data)
# visalize both: anomaly_score and outlier
plt.plot(anomaly_score_test_data, anomaly_score, label='anomaly score')
plt.fill_between(anomaly_score_test_data.T[0], np.min(anomaly_score), np.
 ↪max(anomaly_score),
                 where=outlier_labels==-1, color='#FF00FF',
                 alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('x')
plt.xlim([-5, 5])
plt.show()
```



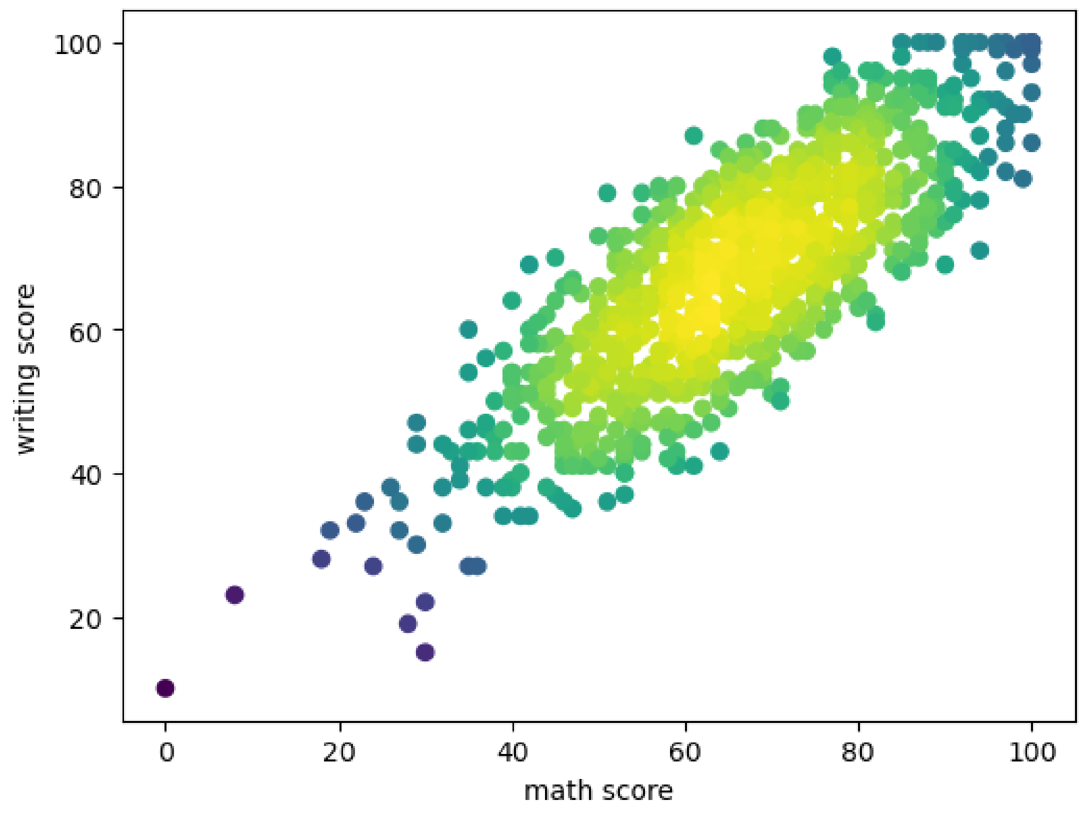
```
[114]: import pandas as pd

       data = pd.read_csv("data/exercise_1/StudentsPerformance.csv")

       data.head()
```

```
[114]:    gender race/ethnicity parental level of education         lunch  \
       0  female        group B           bachelor's degree      standard
       1  female        group C                some college      standard
       2  female        group B             master's degree      standard
       3    male        group A          associate's degree  free/reduced
       4    male        group C                some college      standard

         test preparation course  math score  reading score  writing score
       0                    none          72             72             74
       1               completed          69             90             88
       2                    none          90             95             93
       3                    none          47             57             44
       4                    none          76             78             75
```

```
[116]: def outliers_isolation_forest(data):
           if isinstance(data, pd.Series):
               data = data.to_numpy()
           if data.ndim == 1:
               data = data[:, np.newaxis]
           isolation_forest = IsolationForest(n_estimators=100)
           isolation_forest.fit(data)
           return isolation_forest.predict(data), isolation_forest.
         ↪decision_function(data)

       pred, score = outliers_isolation_forest(data["math score"])
       plt.scatter(data["math score"], score, c=score)
       plt.show()
```



[117]:
```
pred, score = outliers_isolation_forest(np.stack((data["math score"],␣
  ↪data["writing score"]), 1))
plt.scatter(data["math score"], data["writing score"], c=score)
plt.xlabel("math score")
plt.ylabel("writing score")
plt.show()
```

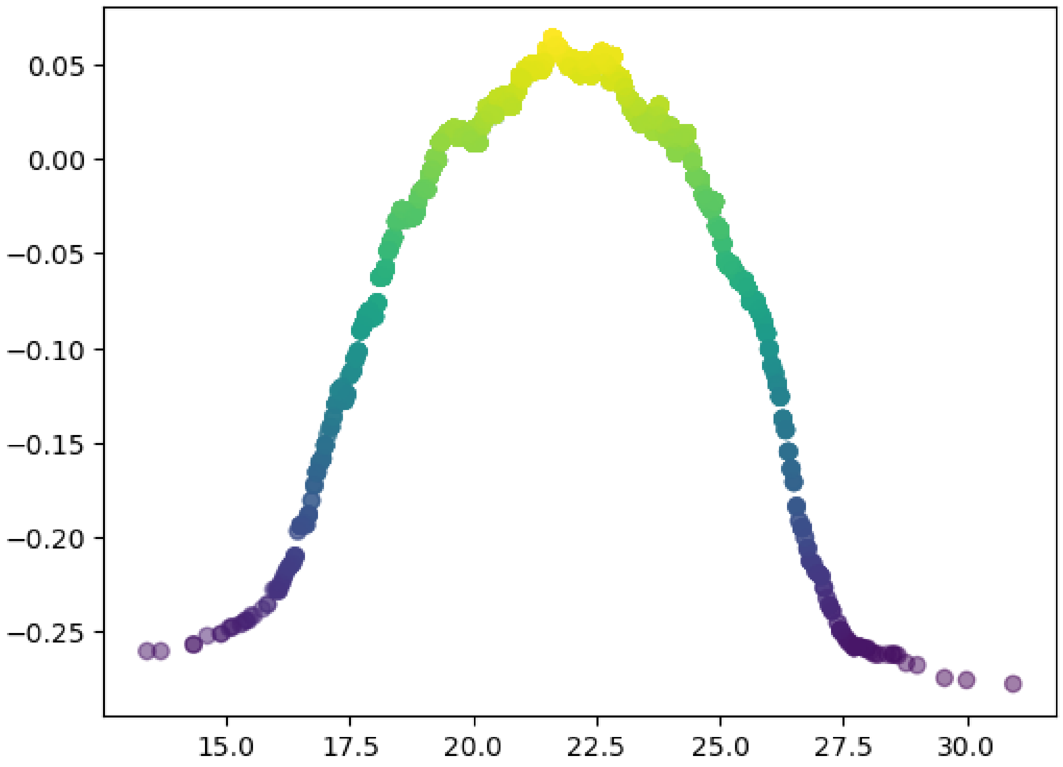


Isolation Forest also return a score how close a point is to be labeled as outlier or not

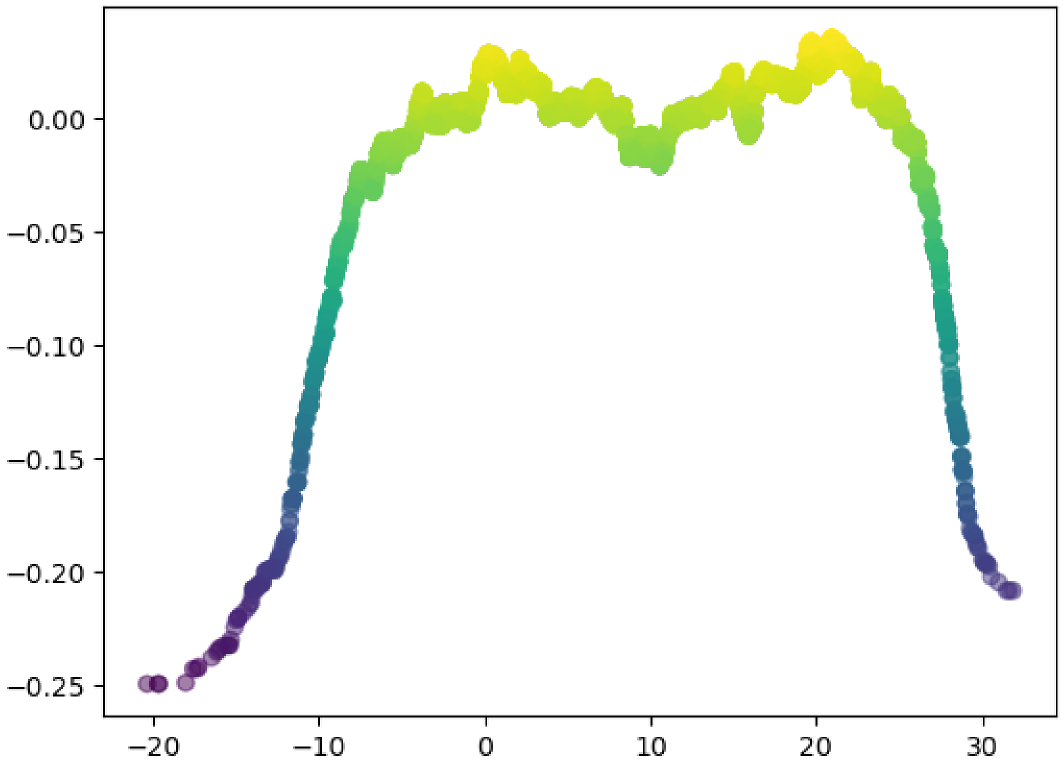
[118]:
```
data = pd.read_csv("data/exercise_1/city_temperature.csv")
data.drop(data[data["AvgTemperature"] == -99].index, inplace=True)
data["AvgTemperature"] = (data["AvgTemperature"] - 32) * 5/9
data["date"] = pd.to_datetime(data[["Year", "Month", "Day"]])
data = data.set_index("date").groupby(["City", pd.
 ↪Grouper(freq="D")])["AvgTemperature"].mean()
data = data.reset_index()
```

```
/tmp/ipykernel_63576/3201813435.py:1: DtypeWarning: Columns (2) have mixed
types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("data/exercise_1/city_temperature.csv")
```

[119]:
```
p_data = data[data["City"] == "Brasilia"]["AvgTemperature"]
pred, score = outliers_isolation_forest(p_data)
plt.scatter(p_data, score, c=score, alpha=.5)
plt.show()
```



[120]:
```
p_data = data[data["City"] == "Pyongyang"]["AvgTemperature"]
pred, score = outliers_isolation_forest(p_data)
plt.scatter(p_data, score, c=score, alpha=.5)
plt.show()
```


0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−20
−10
0
10
20
30

```
[ ]:
```