challenge

August 2, 2019

1 Challenge

1.1 Identifying Outliers using Standard Deviation

[1]:
```
# initial imports
import pandas as pd
import numpy as np
import random
from sqlalchemy import create_engine
```

[2]:
```
# create a connection to the database
engine = create_engine("postgresql://postgres:postgres@localhost:5432/
 ↪fraud_detection")
```

[3]:
```
def find_outliers_sd(card_holder=1):
    query = (
        "SELECT t.date, t.amount, t.card "
        + "FROM transaction AS t "
        + "JOIN credit_card AS cc ON cc.card = t.card "
        + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
        + "WHERE ch.id = "
        + str(card_holder)
        + " ORDER BY date"
    )
    data = pd.read_sql(query, engine)
    elements = data["amount"]
    mean = np.mean(elements, axis=0)
    sd = np.std(elements, axis=0)
    # 2 standard deviations are taken for analysis purposes
    low_transactions = [x for x in elements if (x < mean - 2 * sd)]
    high_transaction = [x for x in elements if (x > mean + 2 * sd)]
    final_list = low_transactions + high_transaction
    if len(final_list) > 0:
        query = (
            "SELECT t.date, t.amount, t.card "
            + "FROM transaction AS t "
            + "JOIN credit_card AS cc ON cc.card = t.card "
            + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
            + "WHERE ch.id = "
            + str(card_holder)
            + " AND t.amount IN ("
            + str(final_list)[1:-1]
            + ") "
            + "ORDER BY date"
        )
        data = pd.read_sql(query, engine)
        return data
    else:
        return "There are no fraudulent transactions identified for this card␣
↪holder"
```

[4]:
```
# find anomalous transactions for 3 random card holders
for i in range(1, 4):
    card_holder = random.randint(1, 25)
    print("*" * 60)
    print(f"Looking for fraudulent transactions for card holder id␣
↪{card_holder}")
    print(find_outliers_sd(card_holder))
```

```
************************************************************
Looking for fraudulent transactions for card holder id 13
                 date  amount              card
0 2018-11-08 02:10:03   22.78  5135837688671496
************************************************************
Looking for fraudulent transactions for card holder id 21
There are no fraudulent transactions identified for this card holder
************************************************************
Looking for fraudulent transactions for card holder id 16
                 date  amount              card
0 2018-01-22 08:07:03  1131.0  5570600642865857
1 2018-02-17 01:27:19  1430.0  5570600642865857
2 2018-05-29 02:55:08  1203.0  5570600642865857
3 2018-06-17 15:59:45  1103.0  5570600642865857
4 2018-07-26 23:02:51  1803.0  5570600642865857
5 2018-11-13 17:07:25  1911.0  5570600642865857
6 2018-12-03 02:38:52  1014.0  5570600642865857
7 2018-12-24 15:55:06  1634.0  5570600642865857
```

1.2 Identifying Outliers Using Interquartile Range

[5]:
```
def find_outliers_iqr(card_holder=1):
    query = (
        "SELECT t.date, t.amount, t.card "
        + "FROM transaction AS t "
        + "JOIN credit_card AS cc ON cc.card = t.card "
        + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
        + "WHERE ch.id = "
        + str(card_holder)
        + " ORDER BY date"
    )
    data = pd.read_sql(query, engine)
    # calculate interquartile range
    q25, q75 = np.percentile(data["amount"], 25), np.percentile(data["amount"],␣
↪75)
    iqr = q75 - q25
    # calculate the outlier cutoff
    cut_off = iqr * 1.5
    lower, upper = q25 - cut_off, q75 + cut_off
    # identify outliers
    outliers = [x for x in data["amount"] if x < lower or x > upper]
    if len(outliers) > 0:
        query = (
            "SELECT t.date, t.amount, t.card "
            + "FROM transaction AS t "
            + "JOIN credit_card AS cc ON cc.card = t.card "
            + "JOIN card_holder AS ch ON ch.id = cc.id_card_holder "
            + "WHERE ch.id = "
            + str(card_holder)
            + " AND t.amount IN ("
            + str(outliers)[1:-1]
            + ") "
            + "ORDER BY date"
        )
        data = pd.read_sql(query, engine)
        return data
    else:
        return "There are no fraudulent transactions identified for this card␣
↪holder"
```

[6]:
```
# find anomalous transactions for 3 random card holders
for i in range(1, 4):
    card_holder = random.randint(1, 25)
    print("*" * 60)
    print(f"Looking for fraudulent transactions for card holder id␣
↪{card_holder}")
    print(find_outliers_iqr(card_holder))
```

```
************************************************************
Looking for fraudulent transactions for card holder id 16
                 date  amount              card
0 2018-01-11 13:20:31   229.0  5570600642865857
1 2018-01-22 08:07:03  1131.0  5570600642865857
```

```
2  2018-02-17 01:27:19  1430.0  5570600642865857
3  2018-05-29 02:55:08  1203.0  5570600642865857
4  2018-06-17 15:59:45  1103.0  5570600642865857
5  2018-07-04 17:28:06    89.0  5570600642865857
6  2018-07-26 23:02:51  1803.0  5570600642865857
7  2018-10-19 12:32:37   178.0  5570600642865857
8  2018-10-23 22:47:13   393.0  5570600642865857
9  2018-11-13 17:07:25  1911.0  5570600642865857
10 2018-12-03 02:38:52  1014.0  5570600642865857
11 2018-12-24 15:55:06  1634.0  5570600642865857
************************************************************
Looking for fraudulent transactions for card holder id 22
There are no fraudulent transactions identified for this card holder
************************************************************
Looking for fraudulent transactions for card holder id 17
There are no fraudulent transactions identified for this card holder
```