Fall 2022 MAT265 Outliers

- Shaokang Li

Introduction:

For this week's assignment, I try to find outliers of different kinds.

- Using standard deviation of checkout times to find the most popular and unpopular items within CD category
- Using both purchase number and checkout times as the indicator of popularity, applying algorithms find out the outliers.
- Since itemNumber is auto incremented when entering the library, this attribute should be consecutive. I want to find out if the data follows such pattern. If not, what's the distribution looks like? What's the proportion of item that are never appear in the database?

Query 01:

For the first query, I tried to find out the most popular and unpopular items within CD category. The checkout times is used as the indicator of popularity.

QUERY:

This query is used to find out the quartiles of each bibNumber by their checkout numbers. However, the window function NTILE only works in MySQL 8.0. Thus, I port this to Python to do further processing.

```sql
SELECT
    count(cout) as checkouts,
    bibNumber,
    NTILE(4) OVER (
        order by
            checkouts
    ) as quartile
FROM spl_2016.inraw
where
    itemtype in (
        'arcd',
        'nacd',
        'jrcd',
        'accd',
        'cacd',
        'cccd',
        'jccd',
        'nccd'
    )
    and cout > '2020-01-01'
group by bibNumber;
```

Use query below to get the raw data for further processing.

```
SELECT
    count(cout) as checkouts,
    bibNumber,
    title
FROM spl_2016.inraw
where
    itemtype in (
        'arcd',
        'nacd',
        'jrcd',
        'accd',
        'cacd',
        'cccd',
        'jccd',
        'nccd'
    )
group by bibNumber;
```

Python Script:

Below Python script is used to find out outliers, outliers are defined as the article below. We filter out data that are larger than the upper limit and lower than the lower limit. The upper limit is defined as

$$upper_{limit} = (checkout, 75\%) + 1.5 * ((checkout, 75\%) - (checkout, 25\%))$$

$$lower_{limit} = (checkout, 25\%) - 1.5 * ((checkout, 75\%) - (checkout, 25\%))$$

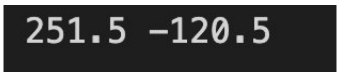
https://dataschool.com/how-to-teach-people-sql/how-to-find-outliers-with-sql/

```
import pandas as pd
import numpy as np

df = pd.read_csv('CD_popularity.csv')
df.head()
print(df.shape)

diff = df.checkouts.quantile(0.75)-df.checkouts.quantile(0.25)
upper_limit = df.checkouts.quantile(0.75) + 1.5*diff
lower_limit = df.checkouts.quantile(0.25) - 1.5*diff
print(upper_limit, lower_limit)

filtered_data = df[df.checkouts>upper_limit].sort_values(by=
['checkouts'],ascending=False)
ratio = df.shape[0]/filtered_data.shape[0]
```

```python
print(ratio)
filtered_data.head()
```

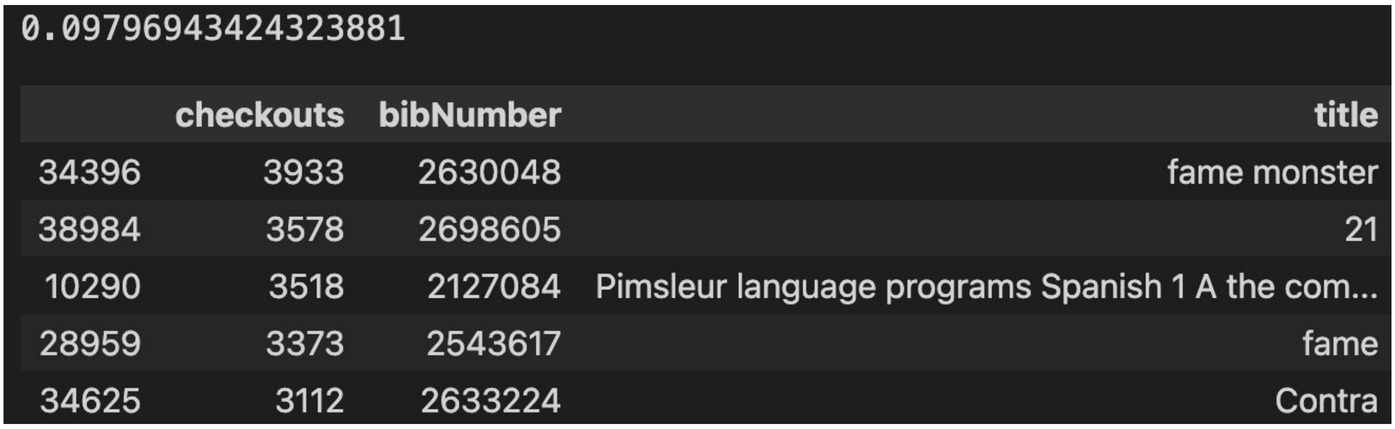
RESULT:

The upper limit and lower limit for outliers are, it is worth noting that the lower limit is actually negative. So we only have to filter out data that are larger than the upper limit.



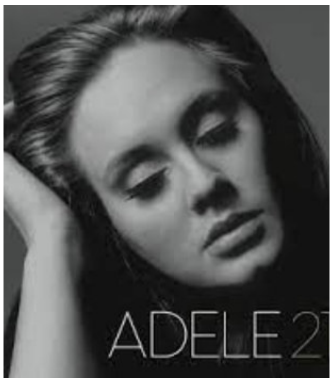
```
251.5 -120.5
```

The number 0.097 means the outliers are nearly 10% of all CDs. And The most popular one is Fame Monster by Lady Gaga and 21 by Adele, also a Spanish Learning CD.



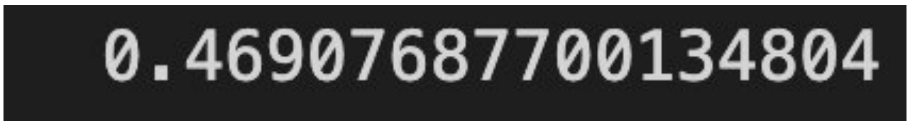
```
0.09796943424323881
```

	checkouts	bibNumber	title
34396	3933	2630048	fame monster
38984	3578	2698605	21
10290	3518	2127084	Pimsleur language programs Spanish 1 A the com...
28959	3373	2543617	fame
34625	3112	2633224	Contra



If we take a closer look at these outliers, they made up of 46.9% of total checkouts.

```python
sum_filtered = filtered_data['checkouts'].sum()
sum_all = df['checkouts'].sum()

checkouts_ratio = sum_filtered/sum_all
print(checkouts_ratio)
```



```
0.46907687700134804
```

Query 02:

For the second query, I'm trying to use 2 columns as the indicator of popularity. The purchase number and the checkout times. Then I'm trying to apply outlier detection algorithm in Python, this may also give us some insights about the popularity distribution of CD data.

QUERY:

Below query returns the CD data with its checkout times and its number of copies.

```sql
SELECT
    count(cout) as checkouts,
    count(distinct(itemNumber)) as copies,
    bibNumber,
    title
FROM spl_2016.inraw
where
    itemtype in (
        'arcd',
        'nacd',
        'jrcd',
        'accd',
        'cacd',
        'cccd',
        'jccd',
        'nccd'
    )
    and cout > '2010-01-01'
group by bibNumber, title;
```

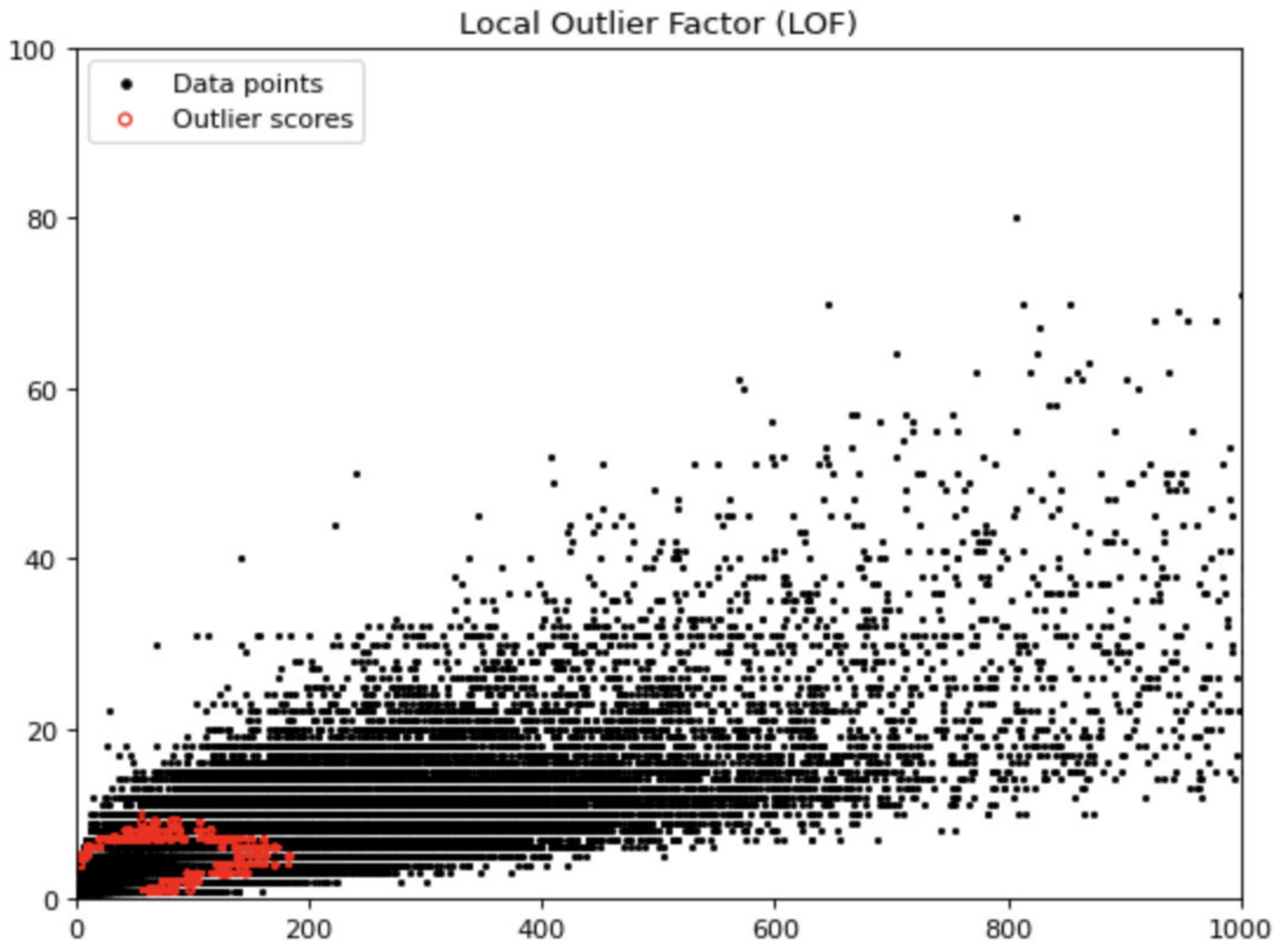
Python Script:

To check ouliters with 2 dimensional data, I used sci-kit learn package for outlier classification. The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors. In this case, a CD that has very high purchase and very low borrow amount.

```python

import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import LocalOutlierFactor
from matplotlib.pyplot import figure
figure(figsize=(8, 6), dpi=80)

df_new = pd.read_csv('CD_popularity_2D.csv')
df_new.head()
arr = np.array(df_new[['checkouts','copies']])
print(arr)

# Generate train data
X = arr

# fit the model for outlier detection (default)
clf = LocalOutlierFactor(n_neighbors=20, contamination=0.1)
# use fit_predict to compute the predicted labels of the training samples
# (when LOF is used for outlier detection, the estimator has no predict,
# decision_function and score_samples methods).
y_pred = clf.fit_predict(X)
X_scores = clf.negative_outlier_factor_

plt.title("Local Outlier Factor (LOF)")
plt.scatter(X[:, 0], X[:, 1], color="k", s=3.0, label="Data points")
# plot circles with radius proportional to the outlier scores
radius = (X_scores.max() - X_scores) / (X_scores.max() - X_scores.min())

plt.scatter(
    X[:, 0],
    X[:, 1],
    s=10 * radius,
    edgecolors="r",
    facecolors="none",
    label="Outlier scores",
)


plt.axis("tight")
plt.xlim((0, 1000))
plt.ylim((0, 100))
legend = plt.legend(loc="upper left")
legend.legendHandles[0]._sizes = [10]
legend.legendHandles[1]._sizes = [20]
plt.show()
```

RESULT:

The x axis is the checkout times, the y axis is the purchase amount. The red dots are the CDs considered as outliers. In this case, there are some CDs has very low checkout times and comparatively adequate stock. (A CD has 2 checkouts and 5 purchase times). There are also some CDs has a high checkout times but limited stock.



The outlier ratio can be computed as below:

```
num_outlier = len(radius[radius>0.01])
outlier_ratio = num_outlier / len(X)
```

This give a result of 0.03372, which means around 3% of the books are either overstocked or understocked.

Query 03:

If we consider all the bibNumber to be consecutive, the number missing can be items that no one borrow. I first filtered all bibNumber that is smaller than 10k.

```sql
select
    distinct bibNumber,
    title
from spl_2016.inraw
where `bibNumber` < 10000
order by bibNumber;
```

There are 2973 entries, which means there are only 29.7% bibNumber being borrowed in the first 10000 bibNumber.

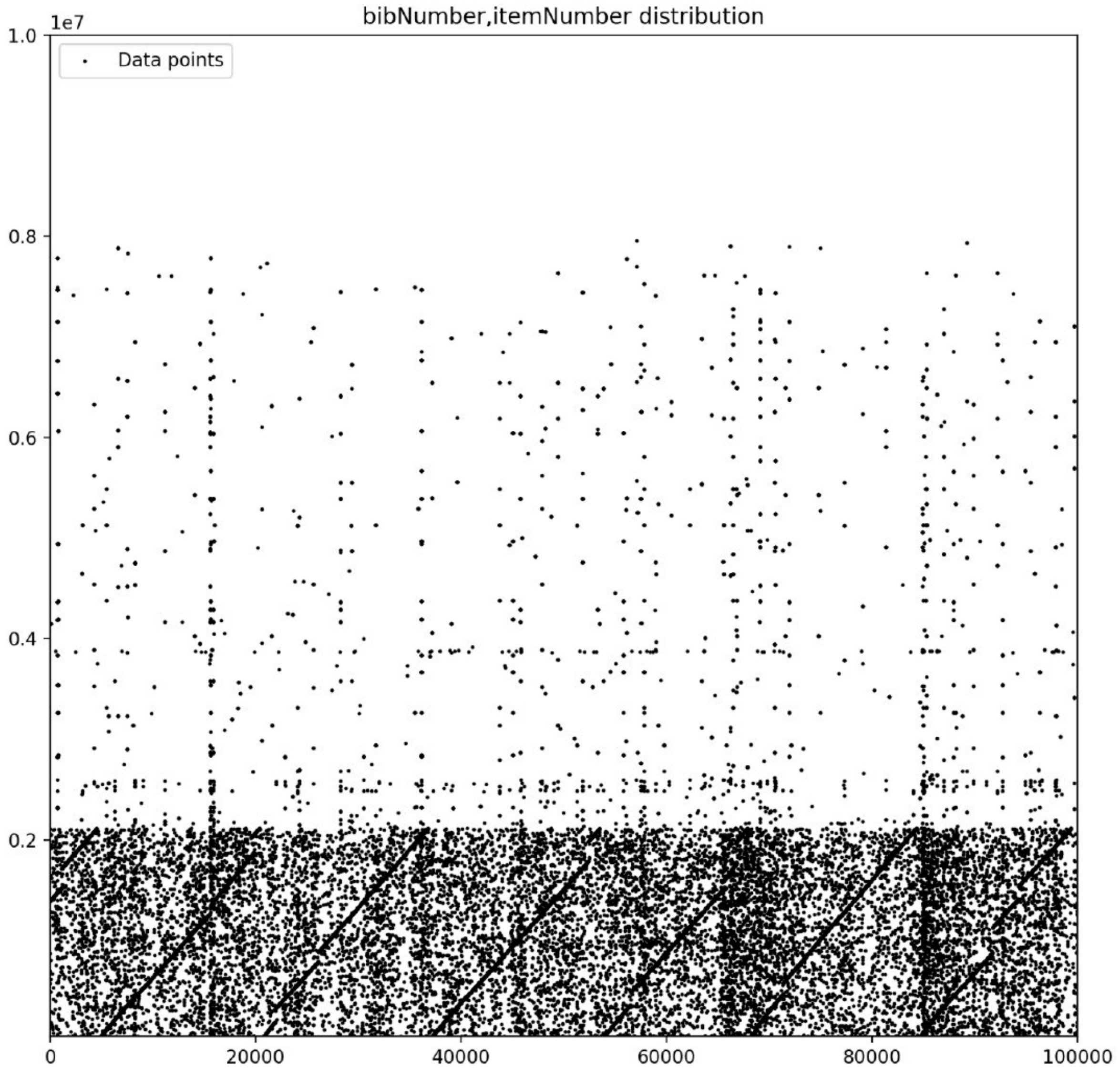
I try to furthur visualize the bibNumber and itemNumber distribution on the 2D plane in the first 100k bibNumber.

```sql
select
    distinct bibNumber,
    itemNumber
    title
from spl_2016.inraw
where `bibNumber` < 100000
order by bibNumber;
```

Python Script:

```python
arr_dist = np.array(df_dist[['bibNumber','itemNumber']])
figure(figsize=(10, 10), dpi=200)
# Generate train data
X = arr_dist

plt.title("bibNumber,itemNumber distribution")
plt.scatter(X[:, 0], X[:, 1], color="k", s=1.0, label="Data points")
# plot circles with radius proportional to the outlier scores

plt.axis("tight")
plt.xlim((0, 100000))
plt.ylim((50000, 10000000))
legend = plt.legend(loc="upper left")
# legend.legendHandles[0]._sizes = [10]
# legend.legendHandles[1]._sizes = [20]
plt.show()
plt.savefig('img.png')
```

The distribution looks as below, there are bibNumber that has a wide range of itemNumber, which means it maybe restocked over a very long period of time. Also, in the lower section of the graph, there are some points forming a sloped line. Points on these lines means the itemNumber and bibNumber grows proportionately. For example, everytime the library purchase a new book, they purchase a fixed amount of copies. Note that these lines appear periodically, which is interesting to see.



Conclusion:

After using different algorithms with outliers, here's my conclusions.

- If we use the quantile method with threshold above, there are 10% outliers making up 46.9% of total checkouts.
- If we use local detection filter algorithm to the dataset, we may find that around 3% of the books are outliers and they are likely either overstocked or understocked.
- In the first 10k bibNumbers, only 29.7% appears in the database.
- There are items that being restocked over a long period of time.
- There are periodic patterns that shows proportionate growth between itemNumber and bibNumber.