**Pattern Recognition and Anomaly Detection**

**ASSIGNMENT 1**

|  |  |  |  |
| --- | --- | --- | --- |
| S. No. | Name | Type | Description |
| 1 | aPr | string | Ask Price that seller willing to take for the scrip |
| 2 | bPr | string | Bid Price that buyer willing to pay for the scrip |
| 3 | aSz | string | Ask size/quantity for trading |
| 4 | bSz | string | Bid size/quantity for trading |
| 5 | sym | string | Actual symbol name of the scrip |
| 6 | avgPr | string | Average trading price of the equity or derivative |
| 7 | c | string | Close value of market snapshot |
| 8 | h | string | High value of market snapshot |
| 9 | l | string | Low value of market snapshot |
| 19 | o | string | Opening price of a market snapshot |
| 11 | oI | string | Open interest is the total number of outstanding derivative contracts that have not been settled |
| 12 | oIChg | string | open interest changed value |
| 13 | ch | string | Change value is the difference between the current value and the previous day's market close |
| 14 | chPer | string | Percentage of change between the current value and the previous day's market close |
| 15 | lTrdT | string | Time of the last transaction |
| 16 | ltp | string | Price at which last transaction / trade is done |
| 17 | ltq | string | Quantity of last transaction |
| 18 | ltt | string | Last transaction time in milliseconds |
| 19 | lttUTC | string | Last transaction time in UTC time zone format |
| 20 | tBQ | string | Total quantity of BUY transaction |
| 21 | tSQ | string | Total quantity of SELL transaction |
| 22 | ttv | string | Total volume of trading done |
| 23 | vol | string | Total amount of a security traded Today |
| 24 | yH | string | 52 week high |
| 25 | yL | string | 52 week low |
| 26 | streaming\_type | string | Streaming type. Added for future use.Pass this value as “quote” always |
| 27 | aPr | string | Ask Price that seller willing to take for the scrip |
| 28 | bPr | string | Bid Price that buyer willing to pay for the scrip |

**Item 2:**

"aPr": "44561.00",

"bPr": "44554.00",

"aSz": "1",

"bSz": "1",

"sym": "2885\_NSE",

"avgPr": "44371.85",

"c": "44119.00",

"h": "44786.00",

"l": "43969.00",

"o": "44105.00",

"oI": "361290",

"oIChg": "22350.00",

"ch": "435.00",

"chPer": "0.99",

"lTrdT": "01 Oct 2019, 06:47:03 PM",

"ltp": "44554.00",

"ltq": "1",

"ltt": "01 Oct 2019, 06:47:03 PM",

"lttUTC": "01 Oct 2019, 01:17:03 PM",

"tBQ": "973",

"tSQ": "610",

"ttv": "782763833.41",

"vol": "782763833.41",

"yH": "44786.00",

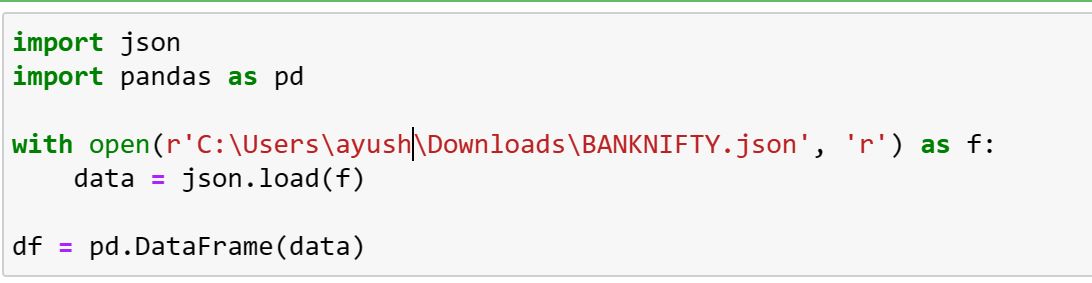
"yL": "0.00",

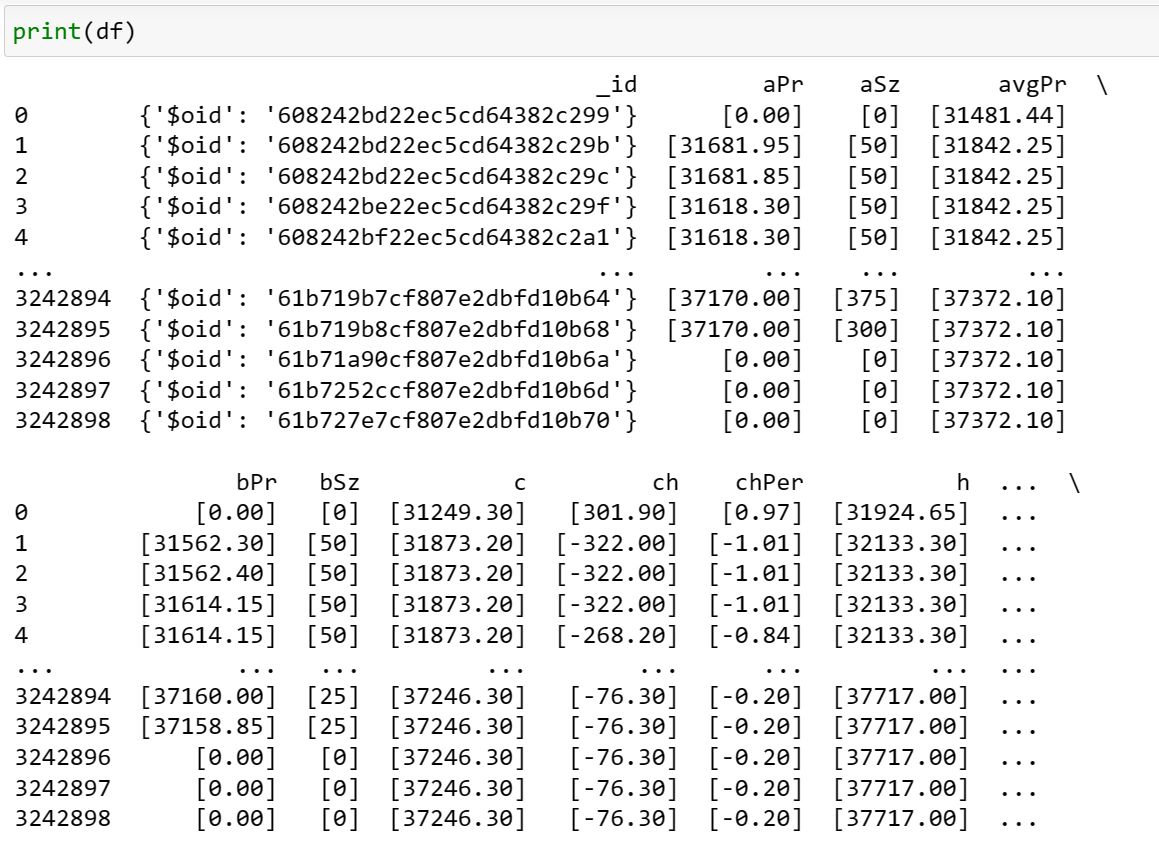
"streaming\_type": "quote"

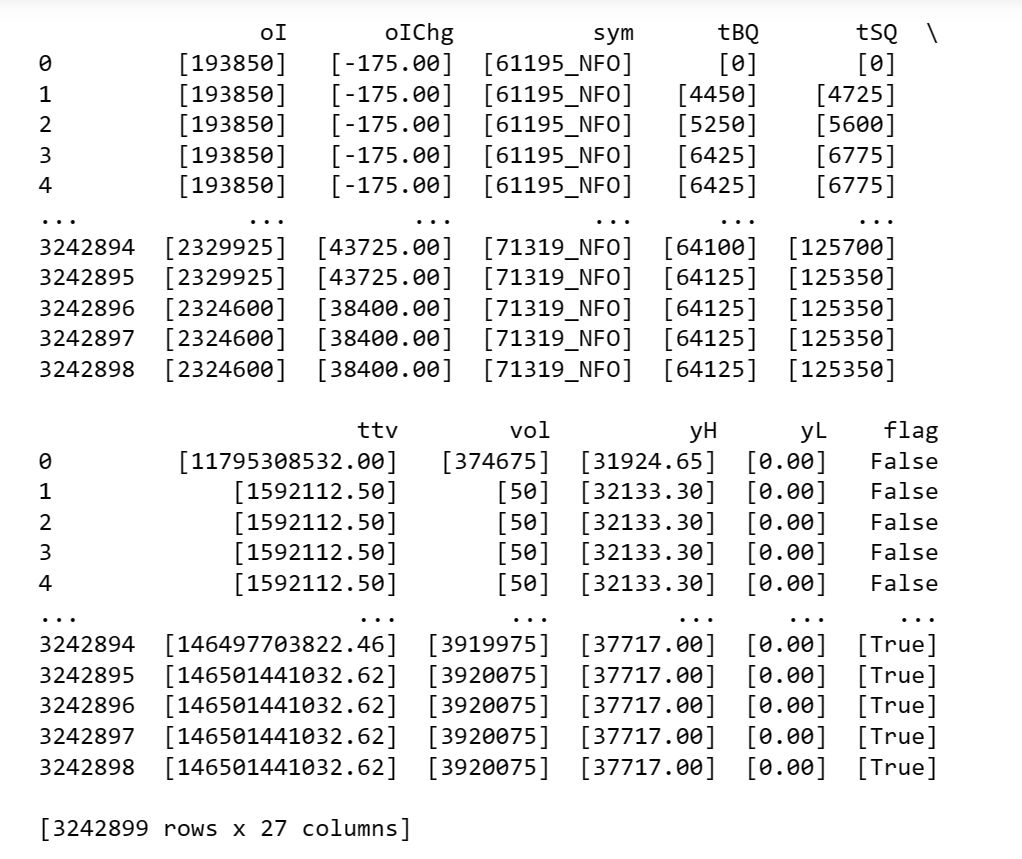
Explanation:

This is the sample item extracted from crawling the data. This contains attributes as mentioned above and by exploring further, we concluded that there are only 4-5 attributes whose value is changing, others are same.

CODE:







**ASSIGNMENT 2**

**OUTLIERS DETECTION**

We used Local Outlier factor and used hyper-parameters n-neighbours as 20 and contamination as 0.005.

Local Outlier Factor (LOF) is a popular algorithm for detecting outliers in a dataset. It measures the local density of a data point compared to its neighbors, and identifies data points that have significantly lower local density as potential outliers. Here's a step-by-step guide on how to apply LOF on any data:

1. Load and pre-process the data: Load the dataset into your programming environment, and pre-process it as needed. This may involve handling missing values, normalizing or scaling features, and encoding categorical variables.

2. Choose the number of neighbours: LOF requires you to specify the number of neighbours to consider when computing local densities. You can choose this value based on your domain knowledge or using techniques such as cross-validation to find an optimal value. Generally, values between 10 to 30 are commonly used.

3. Compute distances: Compute the pairwise distances between data points in the dataset. You can use standard distance metrics such as Euclidean distance or Manhattan distance, depending on the characteristics of your data.

4. Find k-nearest neighbours: For each data point, find its k-nearest neighbours based on the computed distances. The value of k is the number of neighbours chosen in step 2.

5. Compute local densities: For each data point, compute its local density by comparing it with the densities of its k-nearest neighbours. One common approach is to use the inverse of the average distance of a point to its k-nearest neighbours as a measure of local density. Points with lower local densities are potential outliers.

6. Compute local outlier factor: Compute the Local Outlier Factor (LOF) for each data point by comparing its local density with the local densities of its neighbours. LOF is computed as the average ratio of the local density of a point to the local densities of its neighbours. Points with LOF values significantly higher than 1 are considered as outliers.

7. Set an outlier threshold: Set a threshold value to identify outliers. Points with LOF values above this threshold are flagged as outliers.

8. Interpret results: Once you have computed LOF for all data points, you can interpret the results by visualizing the LOF values on a plot or by further analysing the flagged outliers based on domain knowledge or additional investigations.

9. Repeat and refine: You can iterate and refine the LOF process by adjusting the parameters, such as the number of neighbours or the threshold, and repeating the analysis to improve the detection of outliers.

It's important to note that LOF is just one approach for outlier detection, and its effectiveness may vary depending on the characteristics of the data and the specific problem we are trying to solve. It's always recommended to carefully interpret and validate the results obtained from any outlier detection technique.