ASSETTE

Detecting Anomalies

Questrom School of Business MSBA Capstone Project Spring 2022

Team B2:
Aash Gohil
Chunxiaqiu(Tommy) Yang
Phyllis(Fangfei) Cao
Zihan Cui
Zixing Li

Table of Contents

Methodology

I. Code Repository 2

II. List of tools & Documentation Developed 2

Key findings and Deliverables

I. Key Findings

..... 3

II. Deliverable

..... 3

Business Impact and Implications

III. Impact and Implications

..... 4

Limitation and avenues for further improvement

IV. Limitations

..... 5

V. Moving Forward

..... 5

Project Summary

..... 6

The business application deliverables, including the coding, slides, and the poster can be found at the **GitHub repository**:

https://github.com/ChunxiaqiuY/Capstone_TeamB2_ASSEETE

Methodologies

First, we managed to employ modified z scores on the primary account performance factor and the comparison between its corresponding benchmark history. The reason we chose the modified z score is the standard z scores are limited when the data are not normally distributed, and it is sensitive to extreme values. The modified z scores can be more robust than the standard z score because it relies on the median for calculating (It is calculated by subtracting the median and dividing by the constant scale factor times the mean absolute deviation d or the median absolute deviation). We implemented the modified z score in the python within two functions. The results were stored in separate data frames.

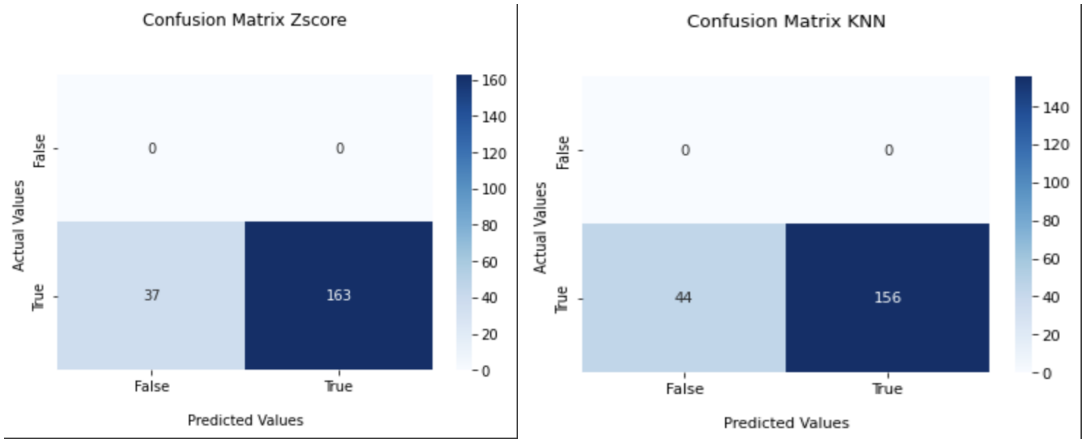
Next, we decided to utilize the K nearest neighbors algorithm to help us detect additional outliers. KNN method detects outliers by exploiting the relationship among neighborhoods in data points. The fundamental theorem behind the nearest-neighbor algorithm is that it is based on the Euclidean distance of K number of neighbors and assigns these points in different categories. Generally similar observations are in proximity to each other, and outliers are usually lonely observations. In addition to the KNN method, we also considered other space distance algorithm such as DBSCAN and LOF; however, the research indicates that the DBSCAN method is not appropriate for the 1-dimension data point because DBSCAN doesn't take advantage of some significant speed-ups that are possible with a 1D space. Also, both DBSCAN and LOF were not performing well as we expected (detected 80 outliers out of 4800 samples). As a result, we choose the KNN method. We utilized the Sklearn and PyOD libraries to implement the KNN method. Finally, we built a class in a python environment to combine four models and finally return the data frame which contains all outcomes from four models.

The class would take three inputs from the business team, including the account id, confidence interval level, and the voting number. The account ID would locate the exact product that the business team would like to investigate. The confidence interval would be the threshold for detecting the outliers, the default value is 1.96. The voting number represents how risky the business team would like to define the outliers. If the voting number equals to 3, that means the algorithm would define the outlier if the total votes (with four models) are greater or equal to 3. The process can be repeated for all accounts.

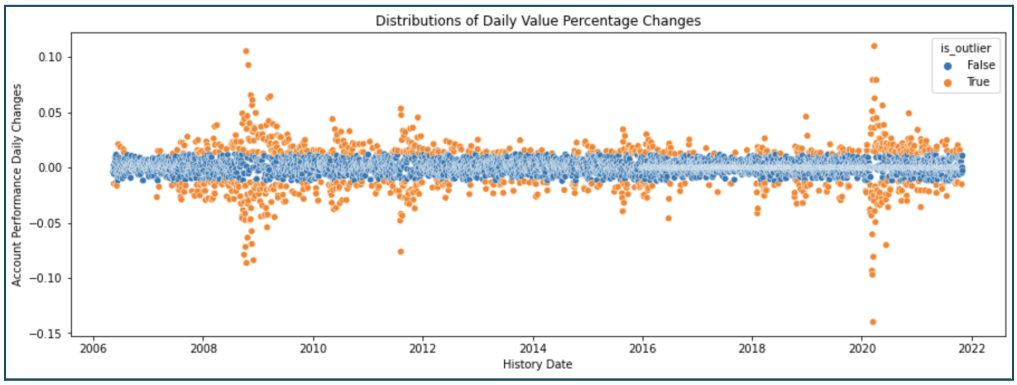
```
## Class to run data through the various models.
class models():
    """ Utilizing different models for anomaly detection. Returns multiple boolean
    arrays with True if points are outliers and False otherwise """

    def __init__(self,accountid = int(acc),classid = 8, vote = int(vot), thesh=float(zsco)):
        '''set up init values for account, modified_z_score and difference'''
        self.account = account_perf[(account_perf['AccountID']==accountid) & (account_perf['AssetClassID']==classid)].MonthlyValue
        self.modified_z_score = []
        self.z_score = []
        self.diff = []
        self.diff_acc_bench = []
        self.model_1 = []
        self.model_2 = []
        self.model_3 = []
        self.model_4 = []
        self.final_sum = []
        self.result = []
```

To test our models' accuracy, we managed to create artificial anomalies and conduct litmus test We randomly took 200 monthly-change-percentage values, multiplied by 10, as human error and labeled those data as anomalies. Next we ran through Modified Z-score and kNN models to test whether they could detect the anomalies we created. The result of the litmus test will be explained in the next section.

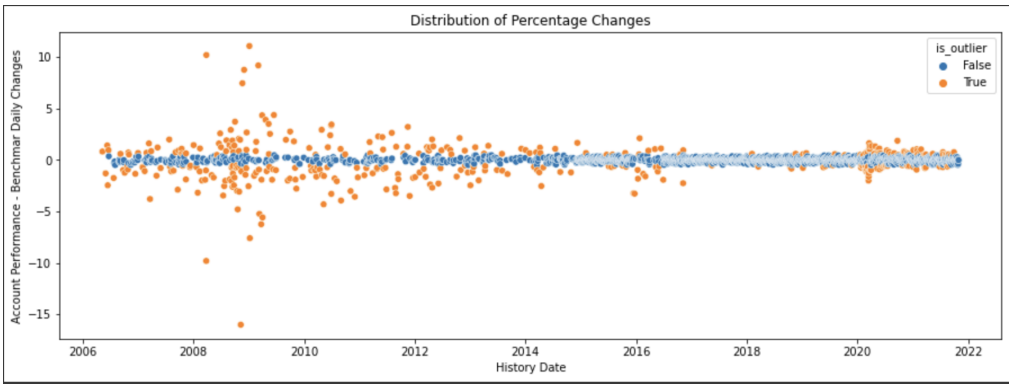
Key Findings and Deliverables



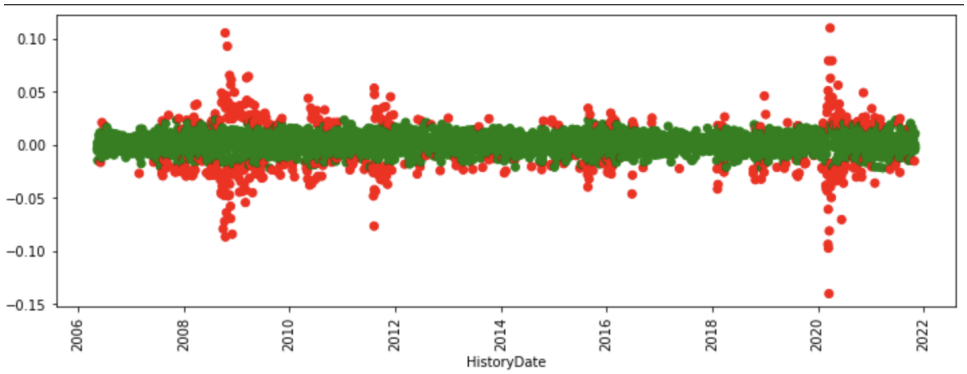
Based on the litmus test performed, explained in the previous section, we found that the Modified Z score method performs better than the KNN method, in most cases we observed Z score performed 10% better than the KNN method, in that the modified Z score method identified 10 % more of the synthetically created anomalies.



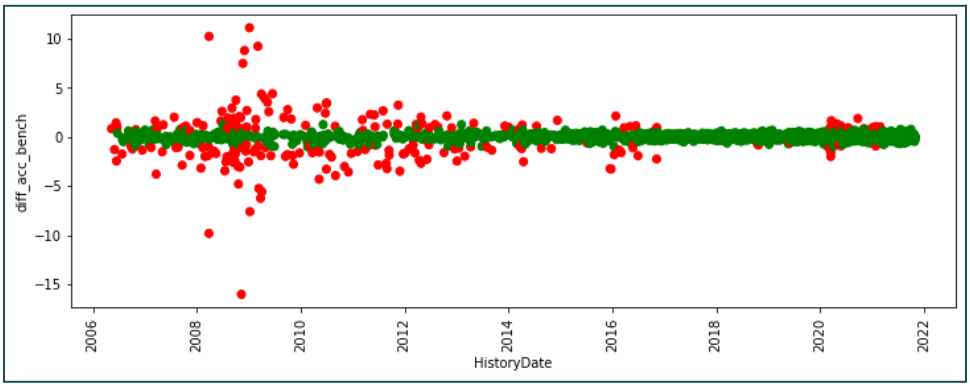
Modified Z-Score: Account Performance Anomalies



Modified Z-Score: Benchmark Comparison Anomalies



KNN: Account Performance Anomalies

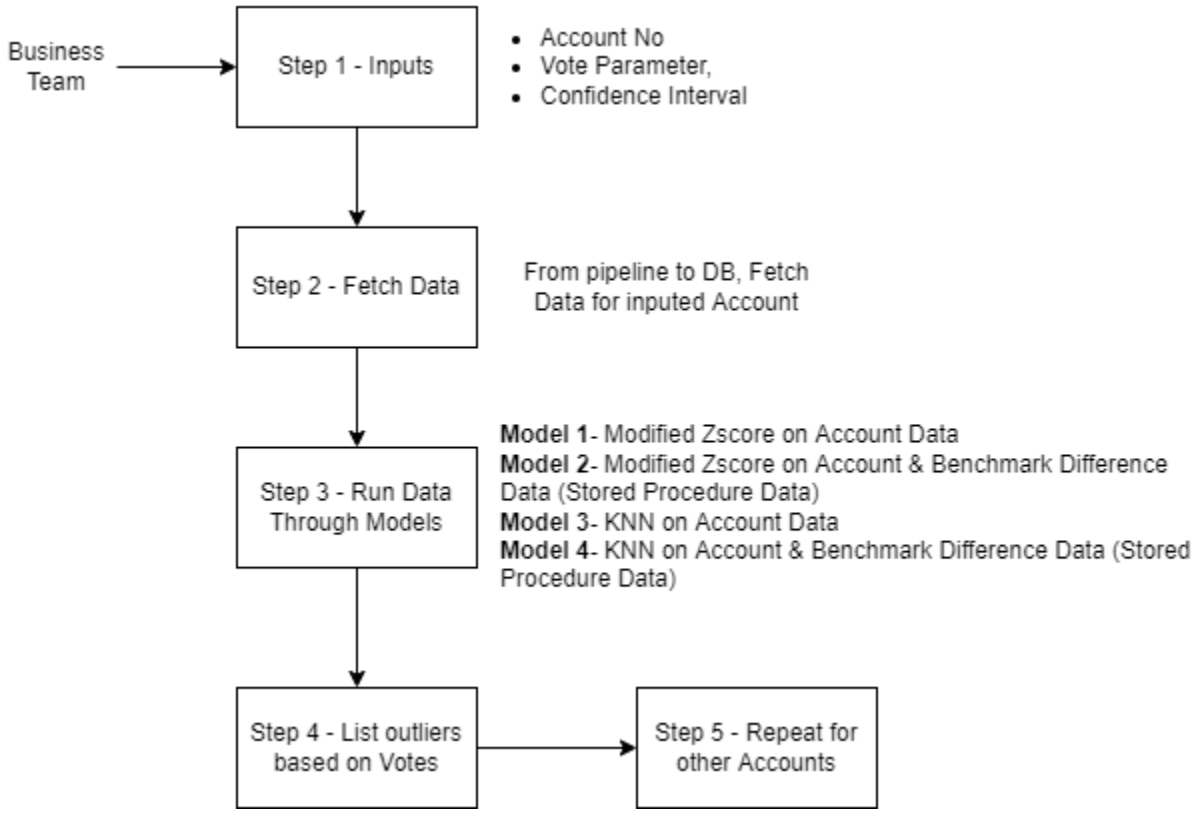


KNN: Benchmark Comparison Anomalies

Additionally based on the anomalies flagged in the original data, for multiple accounts, we found both the modified Z score and the KNN method majorly flagged anomalies around times of financial crisis, namely the 2008 subprime financial crisis and the 2020 covid crash. We also found that when analyzing anomalies based on comparison of the difference of the account return and benchmark return, versus just the account return both models flagged more anomalies for the latter. This makes sense since comparing the account and benchmark return is a more accurate representation of the market, in that the overall trend of the market is taken into account.

In terms of the deliverables, we built a reusable python script, taking in the account to be analyzed. Once the account is inputted, we fetch data for the account by establishing a data pipeline to the database, using the SQL Alchemy library in python which has support for the azure cloud database. After the data is fetched based on the preprocessing steps mentioned in the previous section, the data is run through the models and anomalies are flagged, these anomalies can then be mailed to the team in question and be cross verified manually.

This tool that was created using the reusable python script can be run manually or be scheduled as a cron job. The tool provides an end to end automated solution to flag anomalies for an Assette data warehouse. Below is a diagram of the entire process.



Business Impact and Implications

From the preliminary research, we learned that Assette, a software company, is equipped with data warehouses which are used to store clients' data. The warehouses refresh its client databases on a batch basis, which means that there are possibilities of Assette receiving data that has not been properly curated. Because it is not feasible to cross-check every data point, Assette has yet to formulate and incorporate any anomaly detection tools into its processes. Error detection after data ingestion can also be inefficient due to the high cost of time and resources, which can also cause issues with the compliance if the data errors are not handled properly in time. This project then offers a solution to this problem with a series of statistical validations that detect anomalies in data submitted by asset managers during the ingestion process.

As a team, our goal is to provide a solution that is explainable and maneuverable. By choosing the modified Z-score and KNN methods, we want the managers and clients to be able to understand the decision process of our model. Unlike Black Box technologies, our methodologies are easily understandable even by those not familiar with statistics or machine learning. Additionally, due to the potential variations between accounts, we want to offer the Assette managers more control over the model's outputs through customized hyper-parameters.

This level of flexibility allows the managers who are familiar with the accounts to decide how conservative the anomaly estimates should be. The model's maneuverability also potentially allows it to be implemented into the automated data-fetching process, where the model can be a "one-size-fit-all" error detection process for all incoming data.

Limitations

Our project has two main limitations. First of all, the outliers we detected need to be manually verified by the company, which is not fully automated in a practical sense.
Another limitation is that our outlier results are not really combined with business cognition. Are the outliers we detected truly abnormal in business understanding? Are those anomaly situations truly unreasonable in the customer data? Within the modified z-score method, for the outliers which have big variance, perhaps it is a normal thing to have this outcome for a specific point. We should obtain more rules or regulations from a more practical sense, rather than decide based on the model outputs alone. We need to understand the company's range of outliers tolerance on their operations aspect.

Moving Forward

According to our limitations, in further we can focus on the following steps. First, after getting the outputs of four models, ask the company to manually add labels to these outliers, to identify which are really abnormal in their business perspective, and which are the problems only from model monitoring. After obtaining the labeled data, a set of models or rules can be established and used to screen out problematic data points that were been defined by the company in each outlier result.This part of outputs can then be called high-recall outliers.The remaining outlier could be filtered manually afterwards when having sufficient human resources.These steps can help to enable the business accuracy, further automation, and gain more efficiency.

Project Summary

Data overview

In this project, we heavily use benchmark and account performance datasets that Assette provided to do anomaly detection. We heavily use 'MonthlyValue' in each table to detect anomalies, since it's easy to see if the value is inside or outside the normal range. We created two kinds of datasets based on benchmark and account performance, the primary dataset for the model and the comparison dataset for the model test.

Problem statement - Data outliers detection

Assette is a data warehouse that refreshes its client databases on a batch process. Sometimes the data has not been properly curated and detecting errors after data ingestion has a high cost on time, and it can also cause issues with compliance if it is not handled properly in time. In this project, we are committed to detecting those anomalies in the dataset that Assette provided and our mission is to develop a series of statistical validations to detect anomalies in data submitted during the ingestion process.

Proposed solution

- Two methods include modified z-score and k-nearest neighbor(KNN).

The modified z-score is a standardized score that measures outlier strength or how much a particular score differs from the typical score. We run the model implemented on a sample account 911 percentage of monthly values changes. Modified Z-score implementation on the primary dataset got 824 anomalies. Modified Z-score implementation on comparison dataset got 363 anomalies. The KNN algorithm detects anomalies using the distances of k-nearest. Model implements on sample account 911 percentage of monthly values changes. KNN model implementation on the primary dataset got 460 anomalies. KNN model implementation on different dataset got 222 anomalies.

- Manually create anomalies to test model

Conduct a litmus test to test model accuracy. Randomly take 200 monthly-change-percentage values, multiply by 10, as human error and label those data as anomalies. Run through Modified Z-score and kNN models to test whether they could detect the anomalies we created. The Modified Z-score performs better for getting more True Positive results.

Conclusion & Limitations

The litmus test shows that Z-Score is potentially more accurate than the KNN method. The validity of both methods can also be tested by observing the proportionally larger number of outliers during economic events, such as the financial crisis in the late 2000s and the COVID-19 pandemic in recent years. The sample account models one through four each produced 824, 363, 460, and 222 outliers. This shows that using benchmark differences could potentially be more conservative than using monthly values alone. As for the limitations, there was no labeled data so anomalies need to be manually verified so that the next steps can focus on experimenting with other different methods.