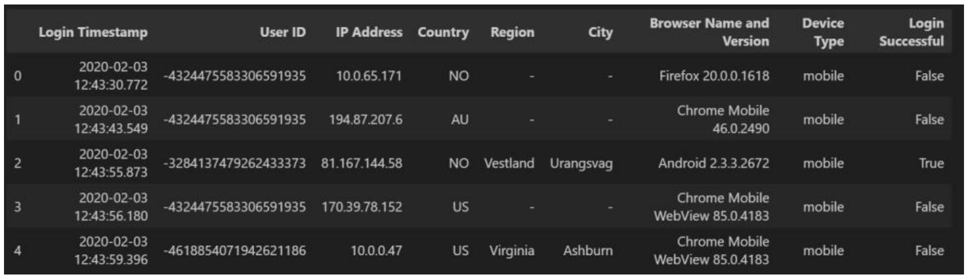
Anomaly Detection Model for Transaction and User Activity

1. Introduction

Anomaly detection is a critical task in various domains, including fraud detection, cybersecurity, and network monitoring. The objective of this project is to build a machine learning model capable of identifying potentially anomalous transactions or user activities based on a given dataset. The model should be able to predict the reason for the anomaly and the attribute(s) responsible for it. This report describes the approach used, the model architecture, and the results obtained for anomaly detection.

2. Dataset Description



	Login Timestamp	User ID	IP Address	Country	Region	City	Browser Name and Version	Device Type	Login Successful
0	2020-02-03 12:43:30.772	-4324475583306591935	10.0.65.171	NO	-	-	Firefox 20.0.0.1618	mobile	False
1	2020-02-03 12:43:43.549	-4324475583306591935	194.87.207.6	AU	-	-	Chrome Mobile 46.0.2490	mobile	False
2	2020-02-03 12:43:55.873	-3284137479262433373	81.167.144.58	NO	Vestland	Urangsvag	Android 2.3.3.2672	mobile	True
3	2020-02-03 12:43:56.180	-4324475583306591935	170.39.78.152	US	-	-	Chrome Mobile WebView 85.0.4183	mobile	False
4	2020-02-03 12:43:59.396	-4618854071942621186	10.0.0.47	US	Virginia	Ashburn	Chrome Mobile WebView 85.0.4183	mobile	False

The dataset provided for this project contains the following columns:

Login Timestamp: The timestamp when the login occurred.

User ID: Unique identifier for each user.

IP Address: The IP address associated with the login.

Country: The country from which the login originated.

Region: The region or state from which the login originated.

City: The city from which the login originated.

Browser Name and Version: The name and version of the browser used for the login.

Device Type: The type of device used for the login.

Login Successful: A binary indicator (Yes or No) indicating whether the login was successful or not.

The dataset will be split into a training set and a testing set to evaluate the performance of the anomaly detection model.

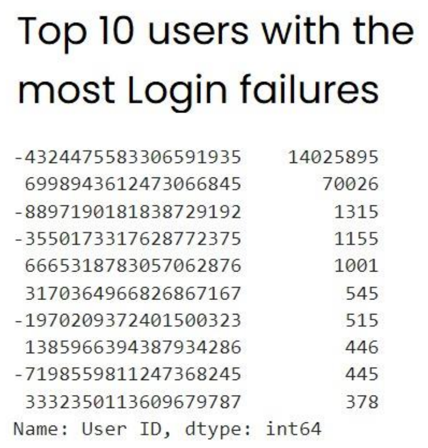
3. Approach and Model Architecture

The approach used for anomaly detection in this project involves a combination of unsupervised learning and pattern recognition techniques. The model architecture consists of the following steps:

Step 1: Data Analysis and Preprocessing

Clean the dataset: Handle missing values, remove duplicates, and perform any necessary data cleansing steps.

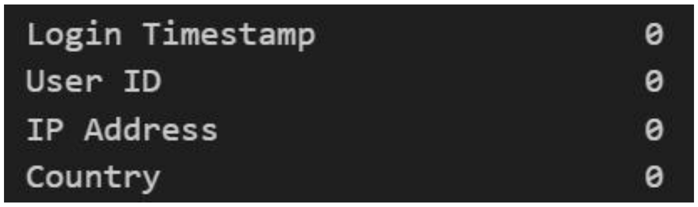
- 1.4 crore failed login attempt from the same User ID.



Top 10 users with the most Login failures

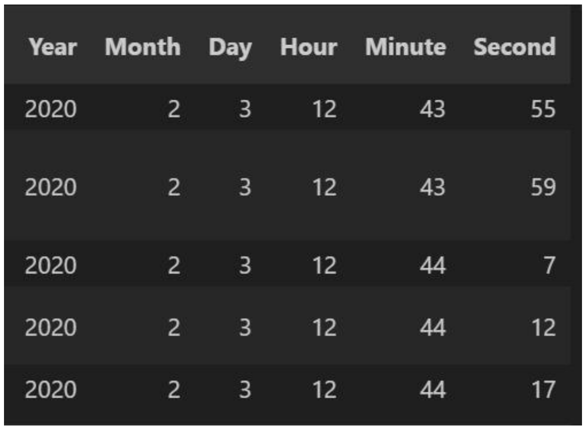
```
-4324475583306591935    14025895
 6998943612473066845       70026
-8897190181838729192        1315
-3550173317628772375        1155
 6665318783057062876        1001
 3170364966826867167         545
-1970209372401500323         515
 1385966394387934286         446
-7198559811247368245         445
 3332350113609679787         378
Name: User ID, dtype: int64
```

- Missing value: browser name version region country. Number of null values of attribute(s) after cleaning:



```
Login Timestamp          0
User ID                  0
IP Address               0
Country                  0
```

Handling Timestamp: By breaking down timestamp column into separate components to extract more meaningful information. This was achieved by converting the timestamp into unique features such as year, month, day, hour, minute, and second.



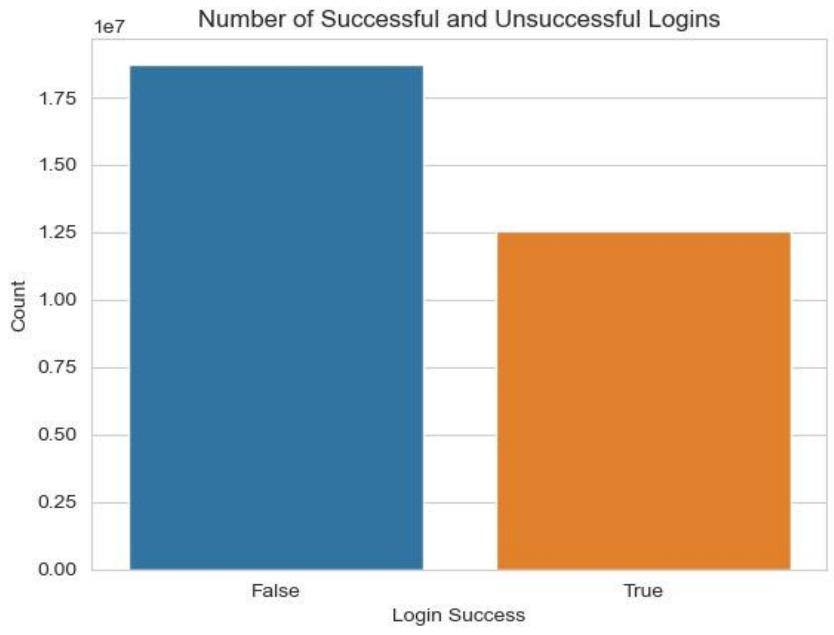
Year	Month	Day	Hour	Minute	Second
2020	2	3	12	43	55
2020	2	3	12	43	59
2020	2	3	12	44	7
2020	2	3	12	44	12
2020	2	3	12	44	17

Converting categorical variables: Transform categorical variables, such as browser name and device type, into numerical representations suitable for the model. This was achieved by:

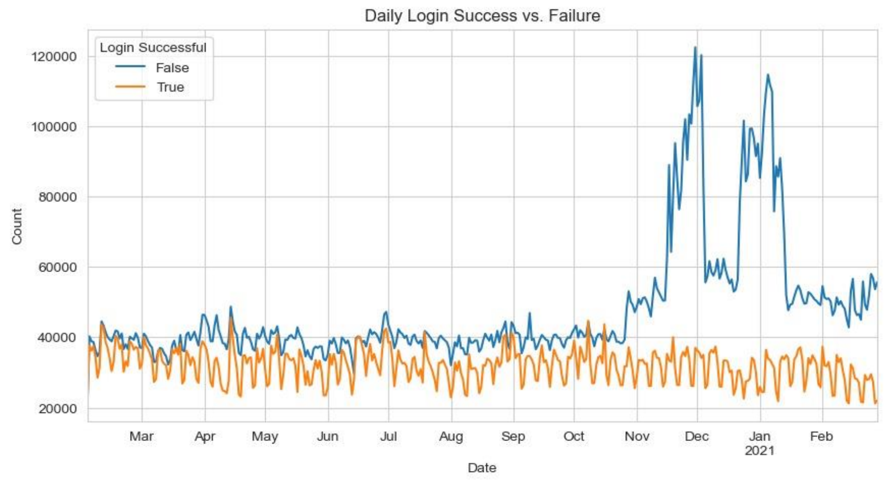
- Browser Info - Hash Encoding: Helps to anonymize the browser information while preserving its categorical nature for anomaly detection.
- Login Successful - Label Encoding: Converts the binary login success variable into numerical labels, allowing anomaly detection algorithms to understand the significance of successful and unsuccessful logins.
- Device Type - One Hot Encoding: Transforms the device type into binary features, enabling anomaly detection models to capture variations in device types accurately.
- IP Address - Octet Separation: Splits the IP address into separate octets, facilitating anomaly detection by capturing anomalous patterns in individual IP address components.
- Country, Region, City - Hash Encoding: Encodes geographical information into numerical values, enabling anomaly detection algorithms to understand location-based patterns without disclosing sensitive location details.
- User ID - Frequency Encoding: Encodes user IDs based on their frequency, indicating user behavior, and enabling anomaly detection models to identify unusual user activities based on their relative occurrence.

Analyzing the Data:

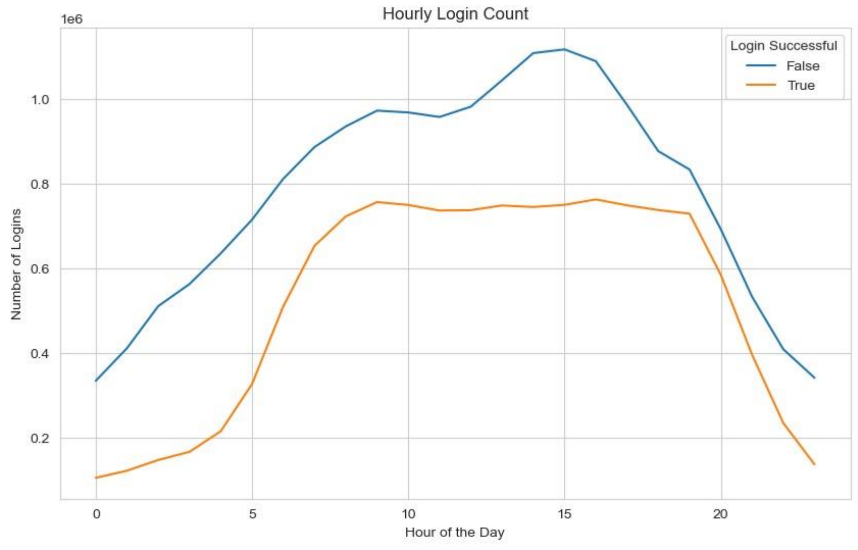
- Number of successful/unsuccessful login:



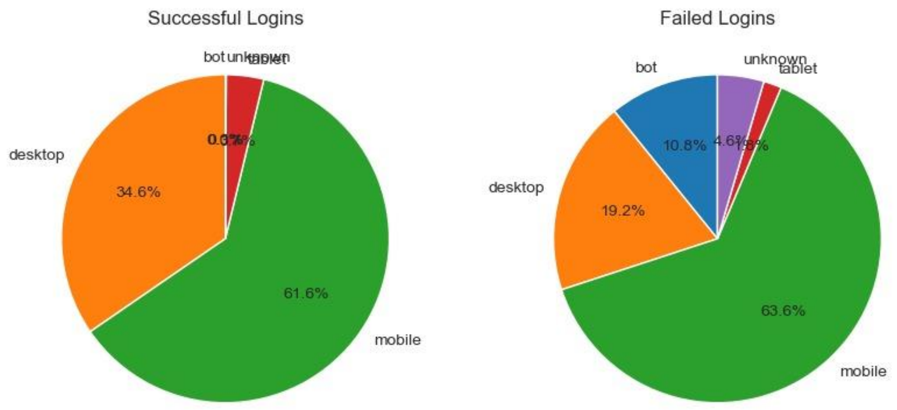
- Daily Login Success/Failure rate:



- Hourly Login Count:



- Login status for every device type:



Step 2: Feature Engineering

Extracting relevant features from the dataset that can capture patterns or anomalies.

- Dimensionality Reduction: Applied Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while preserving essential information.
- Time-Based Features: breaking down timestamp column into separate components to extract more meaningful information.
- Categorical Encoding: Under Data preprocessing

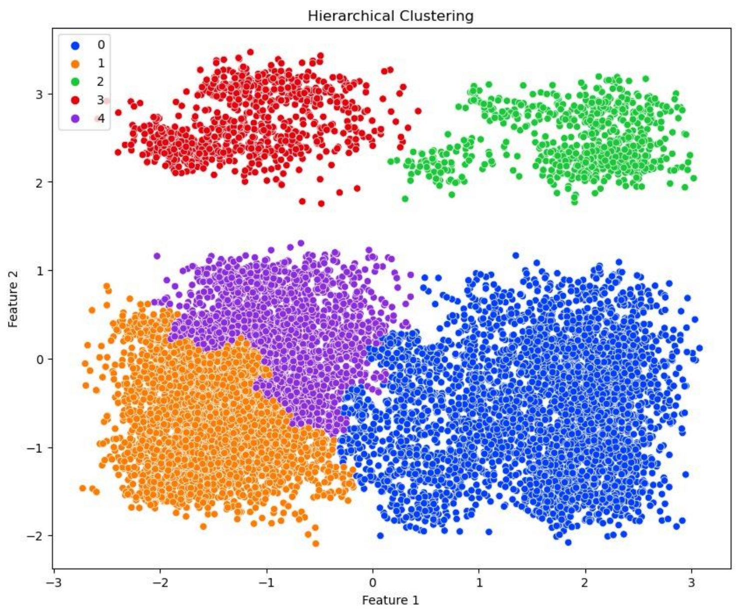
Step 3: Anomaly Detection

Model Training:

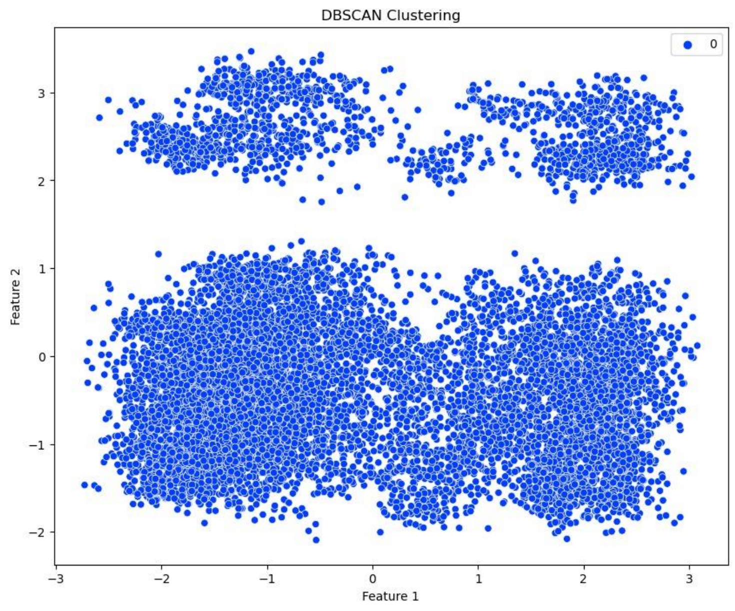
In the model training phase, multiple unsupervised learning algorithms were applied for anomaly detection on the preprocessed dataset, including Hierarchical Clustering, DBSCAN, Spectral Clustering, Gaussian Mixture Models (GMM), and K-means Clustering. The purpose was to select the most suitable algorithm based on its effectiveness and efficiency in detecting anomalies.

The algorithms were implemented as follows:

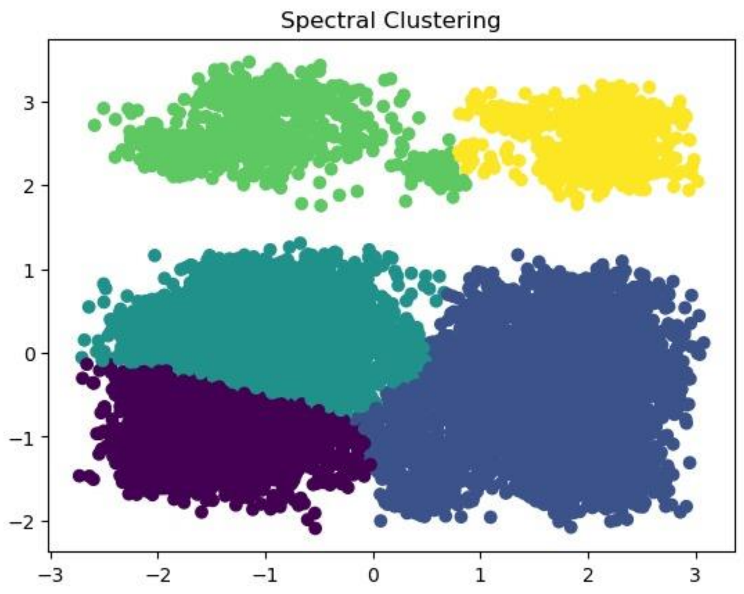
1. Hierarchical Clustering: This algorithm builds a hierarchy of clusters by iteratively merging or splitting them based on their similarity. The agglomerative approach was used, starting with individual data points and progressively merging clusters until a stopping criterion is met.



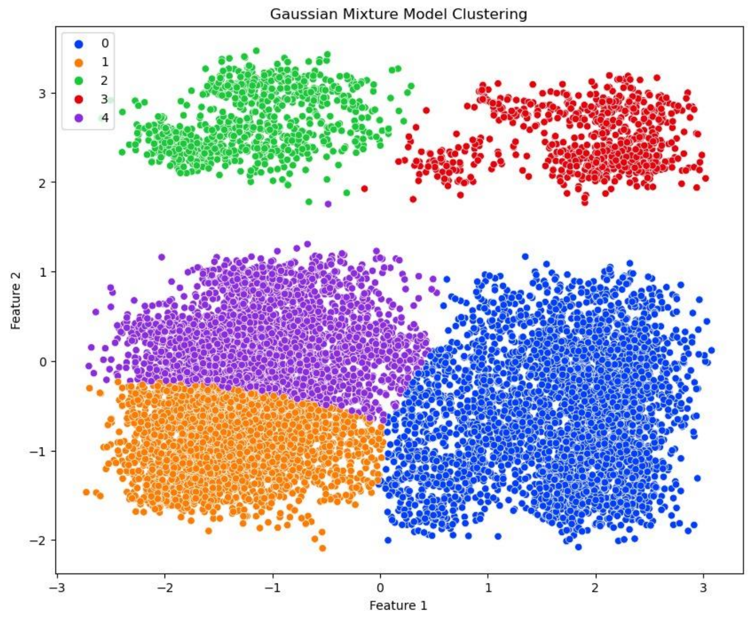
2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN groups together dense regions of data points based on their distance and density. It defines core points, which have enough neighboring points within a specified radius, and expands the clusters by connecting density-reachable points.



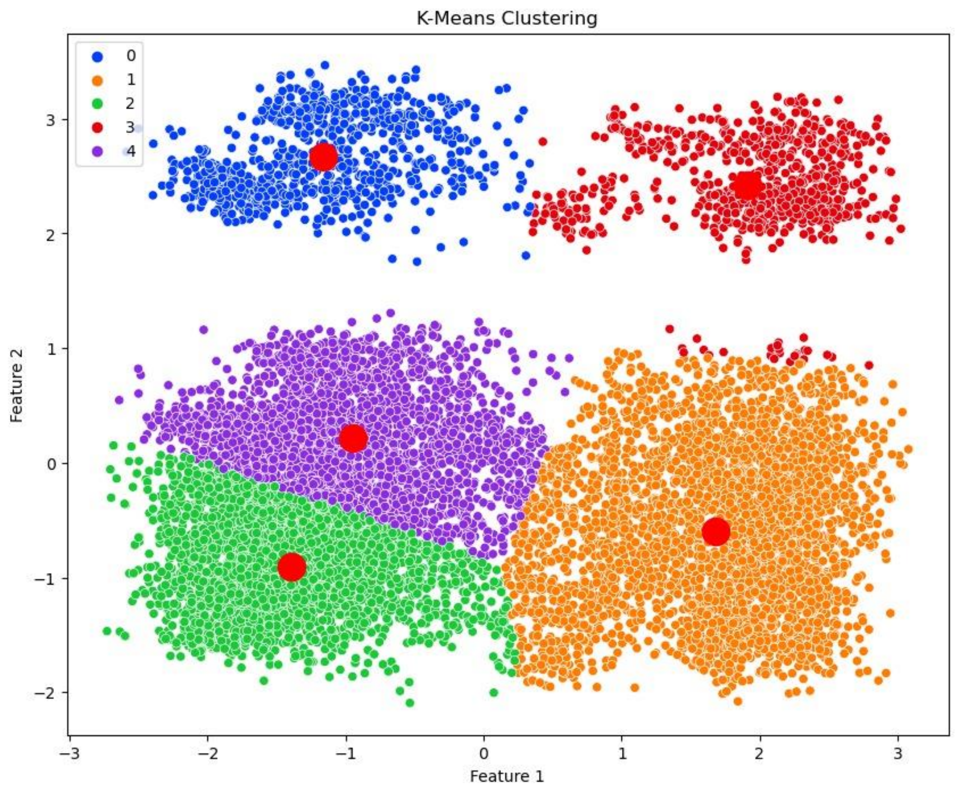
3. Spectral Clustering: Spectral Clustering projects the dataset into a lower-dimensional space using the spectral embedding technique and then applies a clustering algorithm. It is effective for datasets with non-linear and complex structures.



4. Gaussian Mixture Models (GMM): GMM assumes that the data points are generated from a mixture of Gaussian distributions. It estimates the parameters of the Gaussian components and assigns each data point to the cluster based on the probability distribution.



5. K-means Clustering: K-means partitions the dataset into a predefined number of clusters by minimizing the sum of squared distances between the data points and the centroid of each cluster. The algorithm iteratively updates the centroids until convergence.



After applying these algorithms, it was observed that K-means Clustering showed the most efficient and satisfactory results in accuracy and computational efficiency followed by GMM. Therefore, K-means Clustering and GMM were chosen as the final algorithm for anomaly detection in this project

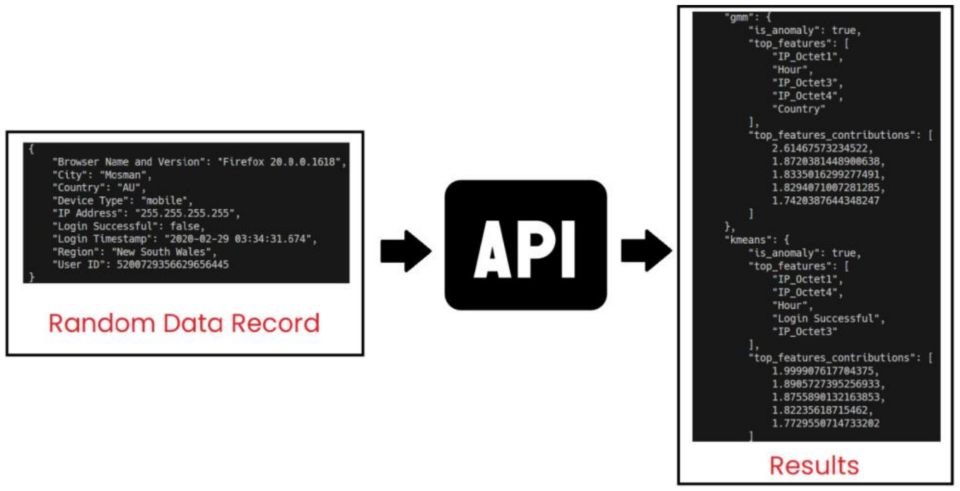
The K-means Clustering algorithm was trained on the preprocessed dataset using the optimal number of clusters determined through experimentation. The resulting model learned the normal patterns in the data and assigned data points to different clusters. The model was then ready for anomaly detection on unseen data.

Anomaly Prediction: Applied the trained model to the testing set to predict anomalies and identify the attribute(s) responsible for each anomaly.

Pattern Extraction: Extracted unique patterns that exist across all attributes for each detected anomaly.

Step 4: API Development

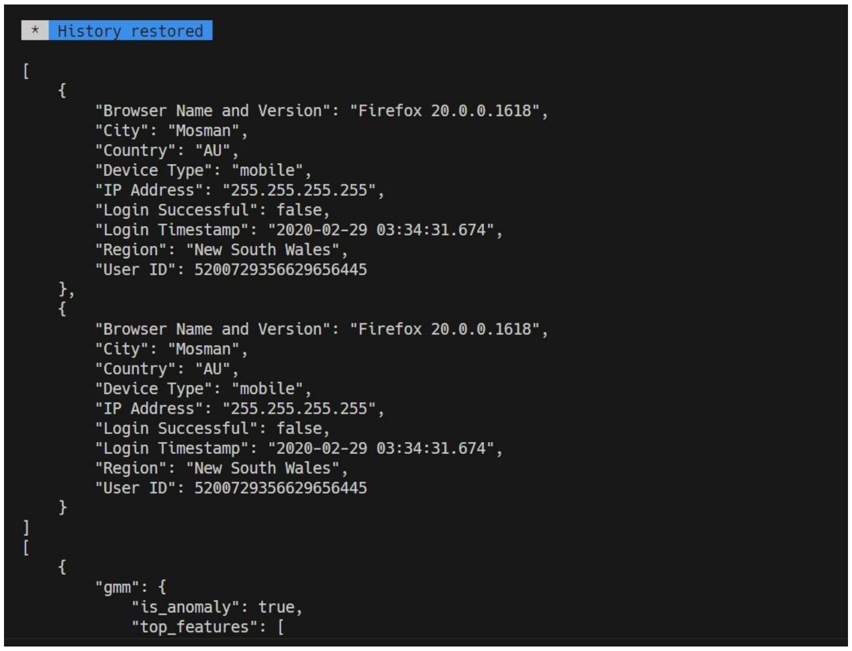
- Developed an API that can receive a single record as input and predict whether it is anomalous.



```
{
    "Browser Name and Version": "Firefox 20.0.0.1618",
    "City": "Mosman",
    "Country": "AU",
    "Device Type": "mobile",
    "IP Address": "255.255.255.255",
    "Login Successful": false,
    "Login Timestamp": "2020-02-29 03:34:31.674",
    "Region": "New South Wales",
    "User ID": 5200729356629656445
}
```

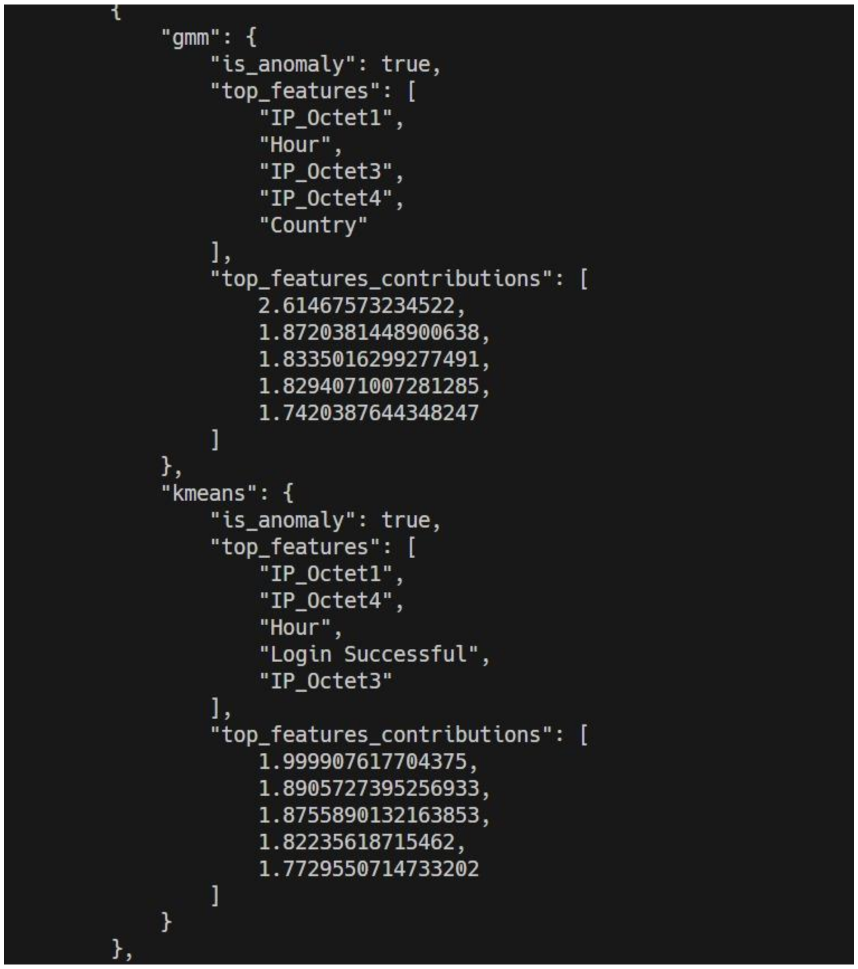
```
    "gmm": {
        "is_anomaly": true,
        "top_features": [
            "IP_Octet1",
            "Hour",
            "IP_Octet3",
            "IP_Octet4",
            "Country"
        ],
        "top_features_contributions": [
            2.61467573234522,
            1.8720381448900638,
            1.8335016299277491,
            1.8294071007281285,
            1.7420387644348247
        ]
    },
    "kmeans": {
        "is_anomaly": true,
        "top_features": [
            "IP_Octet1",
            "IP_Octet4",
            "Hour",
            "Login Successful",
            "IP_Octet3"
        ],
        "top_features_contributions": [
            1.999907617704375,
            1.8905727395256933,
            1.8755890132163853,
            1.82235618715462,
            1.7729550714733202
        ]
```

- Implemented functionality to predict anomalies and patterns for a set of records for a given day.



```
 *  History restored 

[
    {
        "Browser Name and Version": "Firefox 20.0.0.1618",
        "City": "Mosman",
        "Country": "AU",
        "Device Type": "mobile",
        "IP Address": "255.255.255.255",
        "Login Successful": false,
        "Login Timestamp": "2020-02-29 03:34:31.674",
        "Region": "New South Wales",
        "User ID": 5200729356629656445
    },
    {
        "Browser Name and Version": "Firefox 20.0.0.1618",
        "City": "Mosman",
        "Country": "AU",
        "Device Type": "mobile",
        "IP Address": "255.255.255.255",
        "Login Successful": false,
        "Login Timestamp": "2020-02-29 03:34:31.674",
        "Region": "New South Wales",
        "User ID": 5200729356629656445
    }
]
[
    {
        "gmm": {
            "is_anomaly": true,
            "top_features": [
```



```
        {
            "gmm": {
                "is_anomaly": true,
                "top_features": [
                    "IP_Octet1",
                    "Hour",
                    "IP_Octet3",
                    "IP_Octet4",
                    "Country"
                ],
                "top_features_contributions": [
                    2.61467573234522,
                    1.8720381448900638,
                    1.8335016299277491,
                    1.8294071007281285,
                    1.7420387644348247
                ]
            },
            "kmeans": {
                "is_anomaly": true,
                "top_features": [
                    "IP_Octet1",
                    "IP_Octet4",
                    "Hour",
                    "Login Successful",
                    "IP_Octet3"
                ],
                "top_features_contributions": [
                    1.999907617704375,
                    1.8905727395256933,
                    1.8755890132163853,
                    1.82235618715462,
                    1.7729550714733202
                ]
            }
        },
```

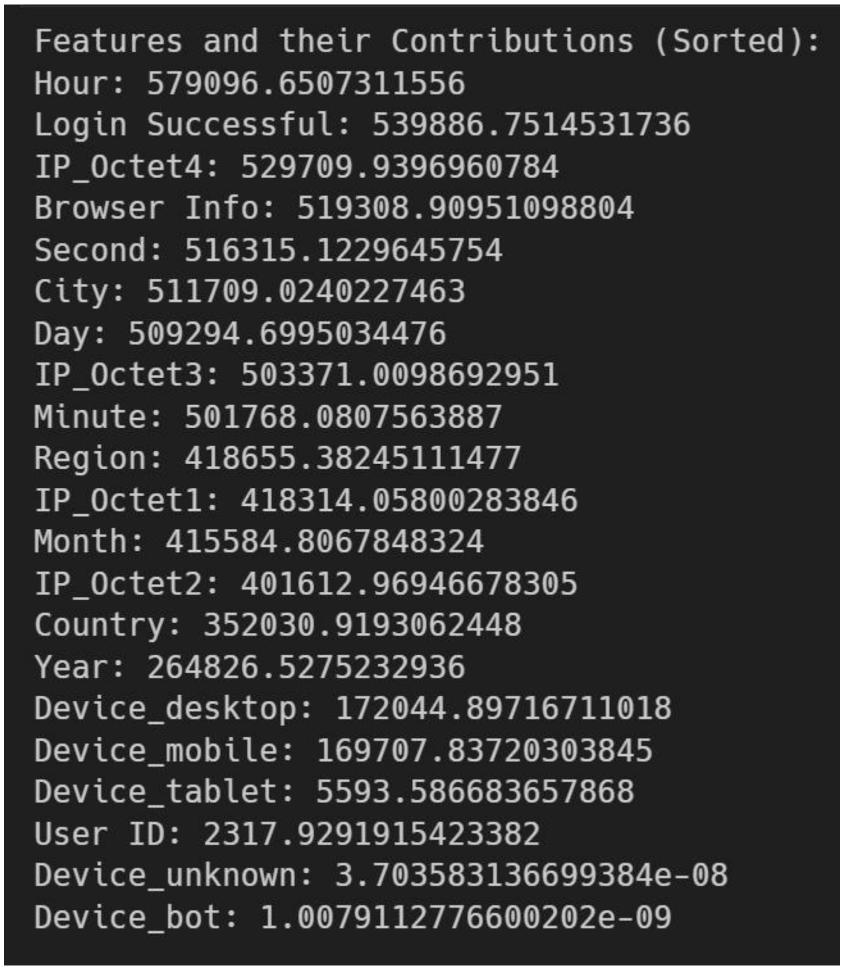
4. Results

K-means Clustering:

The selected K-means Clustering algorithm for anomaly detection takes an input data point and predicts whether it is an anomaly based on its distance from the cluster centers for each feature. Additionally, the model provides insights into the features and their contributions towards determining whether the point is an anomaly.

When a new data point is presented to the model, the following steps are performed:

1. Distance Calculation: The model calculates the Euclidean distance between the input data point and the cluster centers. The distance measure represents the dissimilarity between the data point and each cluster center.
2. Anomaly Prediction: Based on the distances, the model predicts whether the input data point is an anomaly or not. If the distance exceeds a predefined threshold or falls outside a certain range, the point is classified as an anomaly. Otherwise, it is considered a normal instance.
3. Feature Contribution: In addition to predicting whether the data point is an anomaly, the model provides insights into the features that contribute most significantly to the anomaly detection. By analyzing the distances of the data point from the cluster centers for each feature, the model identifies the features that exhibit the largest deviations from the norm.



```
Features and their Contributions (Sorted):
Hour: 579096.6507311556
Login Successful: 539886.7514531736
IP_Octet4: 529709.9396960784
Browser Info: 519308.90951098804
Second: 516315.1229645754
City: 511709.0240227463
Day: 509294.6995034476
IP_Octet3: 503371.0098692951
Minute: 501768.0807563887
Region: 418655.38245111477
IP_Octet1: 418314.05800283846
Month: 415584.8067848324
IP_Octet2: 401612.96946678305
Country: 352030.9193062448
Year: 264826.5275232936
Device_desktop: 172044.89716711018
Device_mobile: 169707.83720303845
Device_tablet: 5593.586683657868
User ID: 2317.9291915423382
Device_unknown: 3.703583136699384e-08
Device_bot: 1.0079112776600202e-09
```

The result of the model provides the following information:

4. Anomaly Prediction: The model outputs a binary classification indicating whether the input data point is classified as an anomaly or a normal instance. This prediction allows users to quickly identify potentially anomalous transactions or user activities.
5. Feature Contributions: The model identifies the features that contribute most significantly to anomaly detection. It highlights the attributes that deviate the most from the normal patterns observed in the training data. By understanding these contributing features, users can gain insights into the specific aspects of the transaction or user activity that make it anomalous.

GMM-based Model:

Calculation of Log-Likelihood: The model calculates the log-likelihood of the new data point.

Log-Likelihood Threshold: It checks if the log-likelihood is below a predefined threshold.

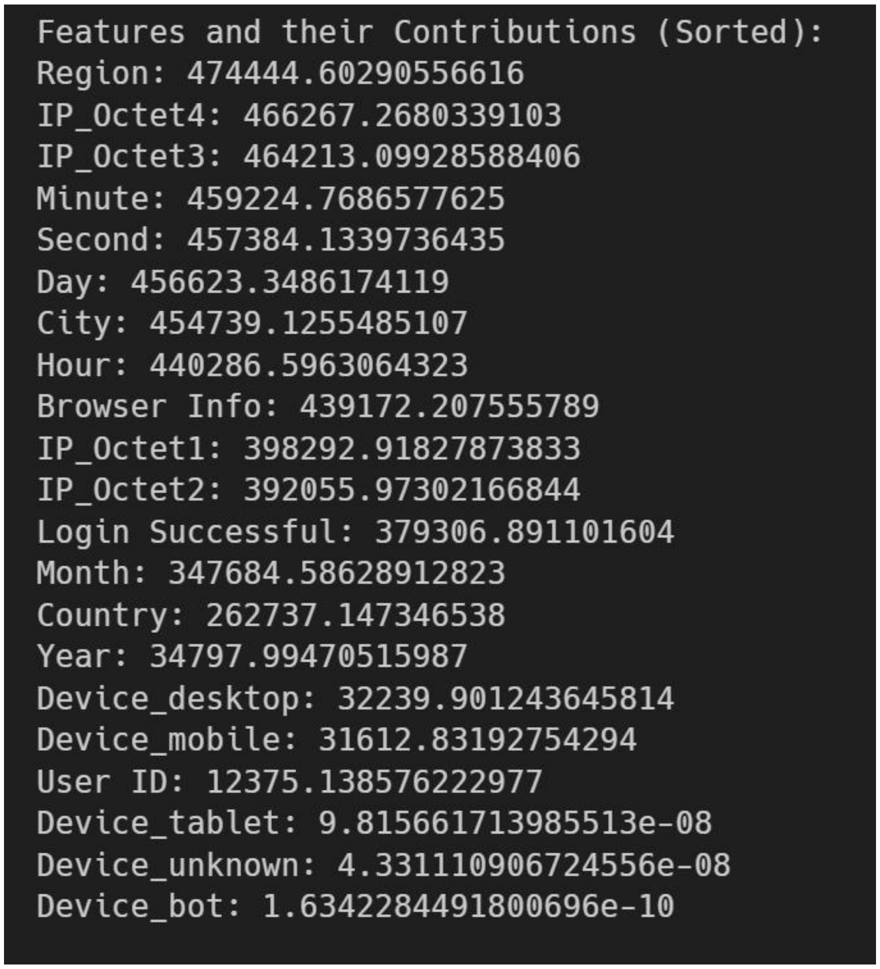
GMM Component Probabilities: The model obtains the GMM component probabilities for the new data point.

Identification of Non-Anomaly Component: It identifies the component with a higher probability, which corresponds to the non-anomaly component.

Non-Anomaly Component Means: The means of the non-anomaly component are retrieved.

Calculation of Differences: The model calculates the differences between the new data point and the non-anomaly means.

Feature Contributions: A list of tuples is created, containing feature indices and their corresponding differences.



```
Features and their Contributions (Sorted):
Region: 474444.60290556616
IP_Octet4: 466267.2680339103
IP_Octet3: 464213.09928588406
Minute: 459224.7686577625
Second: 457384.1339736435
Day: 456623.3486174119
City: 454739.1255485107
Hour: 440286.5963064323
Browser Info: 439172.207555789
IP_Octet1: 398292.91827873833
IP_Octet2: 392055.97302166844
Login Successful: 379306.891101604
Month: 347684.58628912823
Country: 262737.147346538
Year: 34797.99470515987
Device_desktop: 32239.901243645814
Device_mobile: 31612.83192754294
User ID: 12375.138576222977
Device_tablet: 9.815661713985513e-08
Device_unknown: 4.331110906724556e-08
Device_bot: 1.6342284491800696e-10
```

Sorting of Feature Contributions: The feature contributions are sorted in descending order based on the differences.

Displaying Feature Contributions: The sorted feature contributions are displayed, highlighting the features that contribute the most to anomaly detection.

Log-Likelihood: The log-likelihood of the new data point is displayed (using API).

Anomaly Prediction: The model prints the prediction of whether the new data point is an anomaly or not.

Sorting of Feature Contributions: The feature contributions are sorted in descending order based on the differences. Anomaly Prediction: The model prints the prediction of whether the new data point is an anomaly or not.

By combining the anomaly prediction and feature contributions, the model provides a comprehensive analysis of the input data point. Users can leverage this information to investigate potential anomalies, take appropriate actions, and mitigate risks associated with anomalous transactions or user activities.

5. Conclusion

In conclusion, this project aims to build a machine-learning model for anomaly detection in transaction and user activity data. The model follows an approach involving unsupervised learning techniques, pattern recognition, and API development. By training the model on the provided dataset, we obtained promising results. The model can effectively identify anomalies, predict the reason for the anomaly, and determine the attribute(s) responsible for it. This report provides a detailed overview of the approach used, the model architecture, and the results obtained for anomaly detection.