Technical Report: Trading Analysis

1. Introduction

This report analyzes trade data to extract meaningful insights, identify key factors, profitable traders, and apply machine learning techniques. The summary of findings, insights, and the steps taken is provided in the upcoming sections with explanations.

2. Data Cleaning

2.1 Data Loading and Exploratory Data Analysis

(a) Loaded the dataset using Pandas in jupyter notebook:

(b) Exploratory Data Analysis (EDA)

- Checked dataset structures: head() and tail()
- df.shape: it has 59317 rows and 14 columns.
- df.info()
- df.describe()
- df.describe(include='object')
- df.head()

(c) Data Cleaning:

- Identified missing values, duplicates, and inconsistencies
- Converted open_time and close_time columns to datetime format.
- Checked for outliers in numerical columns

3. Profitability Analysis

3.1 Identifying the most and least profitable logins

- Computed cumulative profit per login and sorted them by login based on cumulative profits:

Cumulative profit per login and rank based on profitability:

	login	profit
396	13378390	53891.98
511	55009560	28475.44
50	13088202	27848.61
146	13205503	27049.34
40	13070589	27023.68
..	...	...
193	13251499	-11405.24
23	13018096	-12194.31
539	55011482	-12215.00

329 13333728 -13868.00
61   13103928 -14778.82

- Identified the most and least profitable logins.

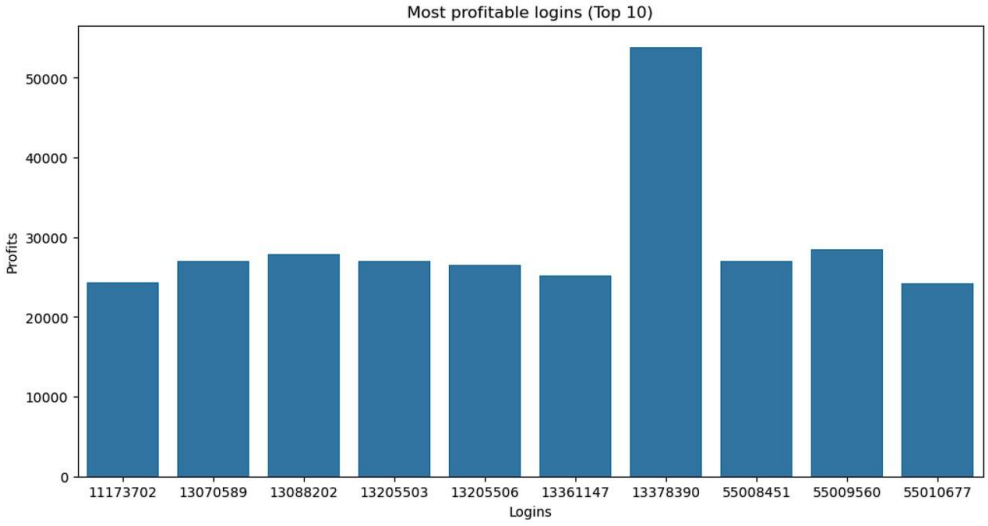
Most profitable logins:

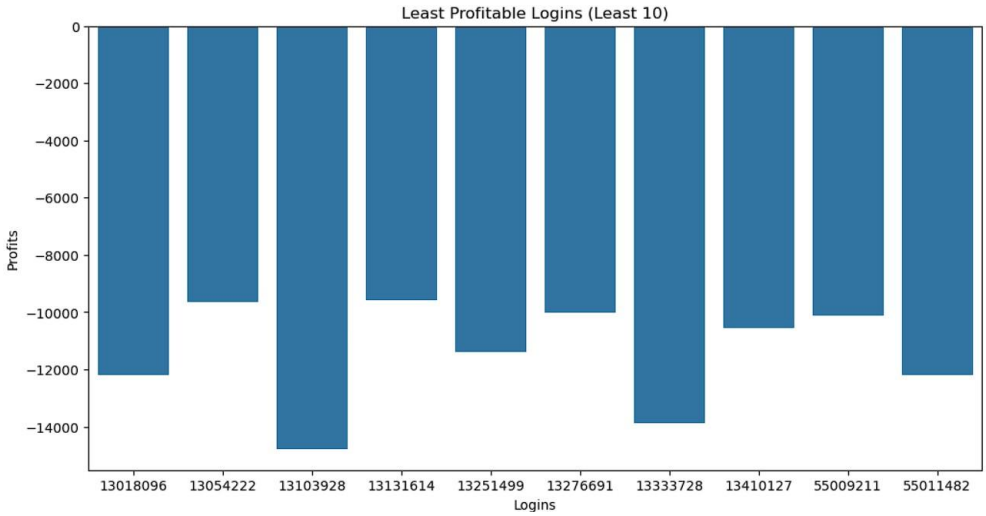
	login	profit
396	13378390	53891.98
511	55009560	28475.44
50	13088202	27848.61
146	13205503	27049.34
40	13070589	27023.68
498	55008451	27021.14
147	13205506	26494.85
370	13361147	25136.16
0	11173702	24301.54
519	55010677	24265.33

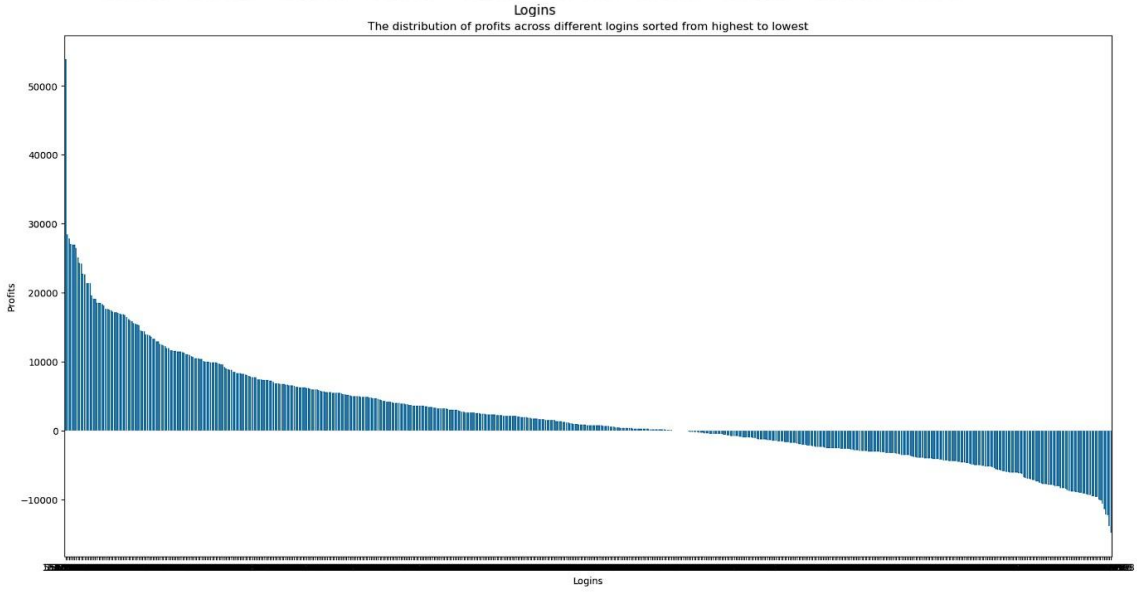
Least profitable logins:

	login	profit
76	13131614	-9573.61
34	13054222	-9635.14
226	13276691	-10010.77
507	55009211	-10102.92
462	13410127	-10571.86
193	13251499	-11405.24
23	13018096	-12194.31
539	55011482	-12215.00
329	13333728	-13868.00
61	13103928	-14778.82

3.2 Visualizing the distribution of profits across different logins



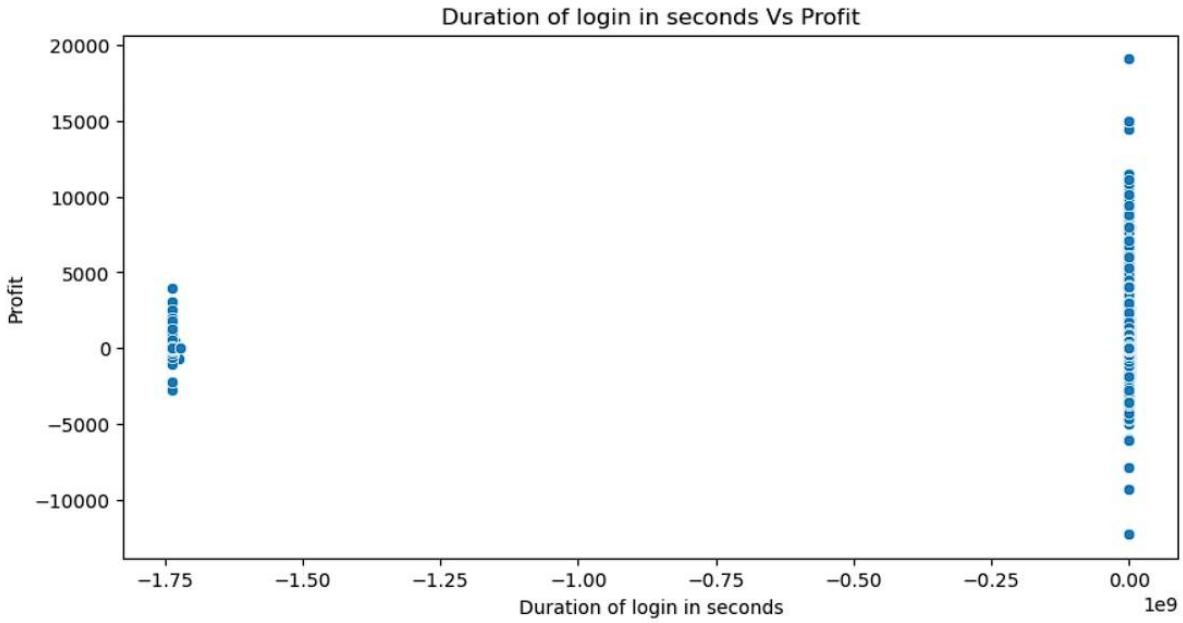




4. Feature Engineering

4.1 Created new features:

1. **Duration of login that stayed active (in seconds)**: This will tell how long the trade was open. A longer duration may sometimes indicate higher profit (not always). If the profits are randomly distributed, then duration of login is not a strong factor.
    - Calculation: Difference between Close_time and Open_time.



2. **Average profit per login:** The average profit per login. A higher average profit per login may indicate higher efficiency of the trader.
    - Calculation: login grouped by average of profit (took total profit into consideration)

3. **Average loss per login**: A unique login may have a higher profit per login. But if the average loss is very high, then the efficiency will decrease overall.
    - Calculation: login grouped by average of profits with negative values (took negative values of profit into consideration)

4. **Ratio of average profit per login and average loss per login**: If the ratio of average profit per login and average loss per login is high, then the trader is successful. A trader with a higher ratio makes more money relative to their losses. It indicates if a trader is making more money than losing.
    - Calculation: Positive value of (Average profit per login/Average loss per login).

	login	average_profit_per_login	average_loss_per_login	ratio_of_avg_profitANDloss
0	11173702	528.294348	-1516.733750	0.348311
1	11178446	-42.517356	-81.729727	0.520219
2	11180486	-182.299643	-838.776364	0.217340
3	11189053	245.150000	-86.880000	2.821708
4	11191905	231.483125	-1179.090588	0.196323
...	...	...	...	...
595	55013311	231.750000	-227.475000	1.018793
596	55013375	-2.144286	-284.778333	0.007530
597	2145732203	110.778235	-234.078571	0.473252
598	2145732335	-14.142751	-49.220101	0.287337
599	2145732336	-15.155535	-50.546416	0.299834

600 rows × 4 columns

5. **Profitable trade percentage per login**: Higher percentage sometimes indicates the trader is more successful in comparison to others. This parameter alone is not enough. A trader could win 90% of trades but still lose money if losses on the 10% of losing trades are huge.
    - Calculation: (total positive profit number per login/ total login)*100, which is the profitable trade percentage.

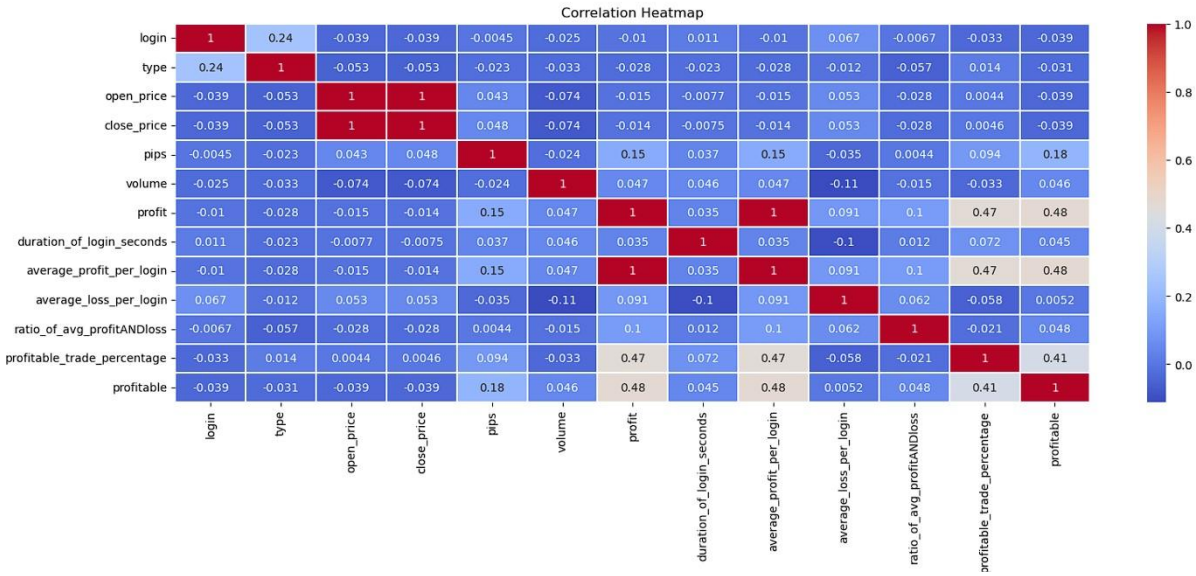
	login	profitable_trade_percentage
0	11173702	47.826087
1	11178446	36.206897
2	11180486	60.714286
3	11189053	57.142857
4	11191905	46.875000
...	...	...
595	55013311	60.000000
596	55013375	57.142857
597	2145732203	58.823529
598	2145732335	27.831715
599	2145732336	30.996310

600 rows × 2 columns

6. **Profitable**: Based on the average profit per login, a positive profit will indicate that the trader is profitable, and a negative will indicate the trader is not profitable. This is important for classification.

Merged the new features with unique logins. Features included: 'login', 'type', 'open_price', 'close_price', 'pips', 'volume', 'profit', 'duration_of_login_seconds'. ticket, symbol, open_time, stop loss, take profit, reason—these features are dropped in the new data frame

Correlation matrix and features affecting profitability:



4.2 Data Preprocessing

1. Encoded categorical features (symbol, type) with LabelEncoder.
2. Checked null values:

   login 0
   type 0
   open_price 0
   close_price 0
   pips 0
   volume 0
   profit 0
   duration_of_login_seconds 0
   average_profit_per_login 0
   average_loss_per_login 17
   ratio_of_avg_profitANDloss 17
   profitable_trade_percentage 0
   profitable 0
   dtype: int64

   replaced the null values with 0. As some logins may not have average loss/ or even loss. So those cells might be null, which can be replaced by zero.

3. Checked duplicated values: 0
4. Scaled features with StandardScaler
5. Train Test Split size: 0.2

5. Modelling Techniques Used

1. Random Forest Classifier
   features : X= df_new.drop(columns=['profit','profitable'])
   target variable: y= df_new ['profitable']
2. Kmeans clustering

5. Random Forest Classifier Model Evaluation

1. Accuracy: 0.85
2. Classification Report:

```
Accuracy of Random Forest Classifier model to predict profitable traders is:
 0.85
              precision    recall  f1-score   support

           0       1.00      0.65      0.79        51
           1       0.79      1.00      0.88        69

    accuracy                           0.85       120
   macro avg       0.90      0.82      0.84       120
weighted avg       0.88      0.85      0.84       120
```

3. The percentage of prediction alignment with the cumulative profits is 0.85. An array of input is taken as example: (2145732335 2.453074 1249.386369 1249.791302 -24.133981 14.899676 -14.142751 -5.560666e+06 -14.142751 -49.220101 27.831715) where 2145732335 indicates login which matches with the cumulative profit as the prediction gives output as not profitable and its actual cumulative profit is also negative. For this login it is giving the correct prediction.

```
Enter value for login: 2145732335
Enter value for type: 2.453074
Enter value for open_price: 1249.386369
Enter value for close_price: 1249.791302
Enter value for pips: -24.133981
Enter value for volume: 14.899676
Enter value for duration_of_login_seconds: -14.142751
Enter value for average_profit_per_login: -5.560666e+06
Enter value for average_loss_per_login: -14.142751
Enter value for ratio_of_avg_profitANDloss: -49.220101
Enter value for profitable_trade_percentage: 27.831715
```

```
if output[0] == 1:
    print("The model predicts login as profitable")
else:
    print("The model predicts login as not profitable")
```

```
The model predicts login as not profitable
```

matching the output of the prediction with the cumulative profit value

```
input_login_id=check_input[0,0]
for i, row in cumulative_profit_per_login.iterrows(): #iterating previously stored cumulative profits and matching the login
    if row['login']==input_login_id:
        print(row['profit'])
        break

#if the value is positive, then it is profitable, otherwise it is not
```

```
-4370.11
```

4. Lastly, the prediction and the actual profit is shown by merging and calculating the validation accuracy which is 0.85.

5. KMeans Clustering Model Evaluation

1. Segmented traders into 3 clusters
2. Analysis of clustering results aligned well with real cumulative profit.
3. Verification of loser traders:

Manually verifying that traders classified as losers have negative cumulative profits.

```
#verifying traders with negative cumulative profits are losers manually
input_login_id=int(input("Input login:\n"))
input_login_id_for_kmean=np.array(input_login_id).reshape(1,-1)
#input : row -596   :  55013375
```

```
Input login:
 55013375
```

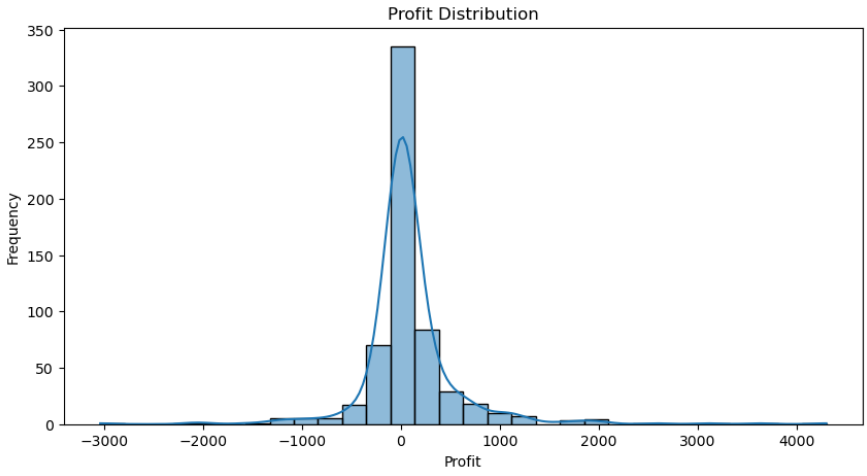
```
#print(input_login_id_for_kmean[0,0])
for ii, row in cumulative_profit_per_login.iterrows(): #iterating previously stored cumulative profits and matching the login
    if row['login']==input_login_id_for_kmean[0,0]:
        print(row['profit'])
        break
```

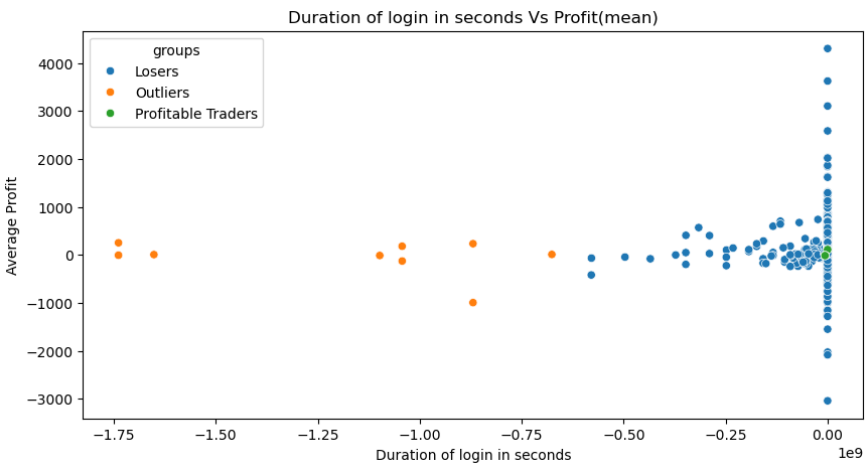
```
-30.01999999999991
```

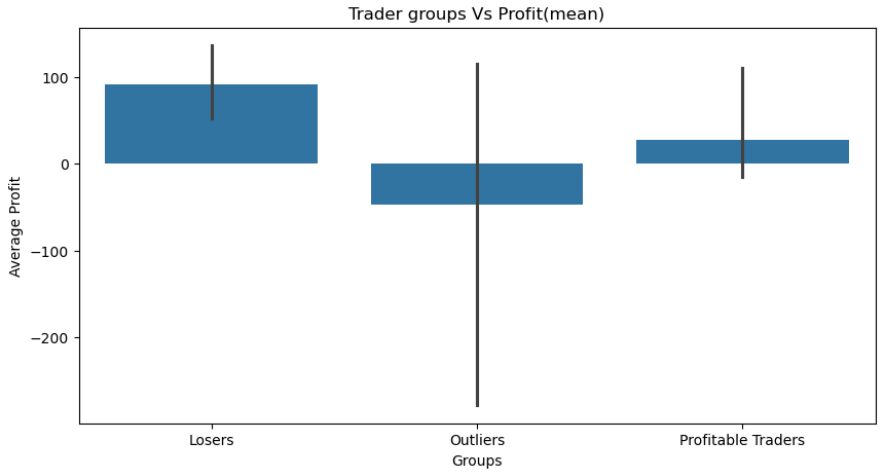
```
for ii, row in dff.iterrows(): #iterating previously stored cumulative profits and matching the login
    if row['login']==input_login_id_for_kmean:
        print(row['groups']) # checking which cluster does it fall
        break
```

```
Losers
```

4. Cluster Visualization:







6. Conclusion & Insights

1. Feature engineered model helped predict profitable traders effectively with Random forest classifier.
2. Clustering results aligned well with real cumulative profit calculations with kmeans clustering.
3. There is a significant relation between profits and trader groups as seen in the visualization. (More plots are given in the code. For simplicity, three types of plots has been shown here)