# Analyzing Market Sentiment and Price Trends Using Fear and Greed Index

- Ayush Dhoble
- Indian Institute of Information Technology, Nagpur
- ayush.dhoble2004@gmail.com

## **Objective** :

Analyze how trading behavior (profitability, risk, volume, leverage) aligns or diverges from overall market sentiment (fear vs greed). Identify hidden trends or signals that could influence smarter trading strategies.

## **Datasets** :

1) Bitcoin Market Sentiment Dataset - tells about sentiment in market "Fear" and "Greed" Data Shape is (2644, 4)

| | |
|---|---|
| Timestamp | - Time given in Unix format |
| Value | - Numerical value of Sentiment in people. Range (0-100) |
| Classification | - Types of sentiments |
| Date | - Date in standard format |

2) Historical Trader Data from Hyperliquid - trades placed by different users and its related information. Data Shape is (211224, 16)

| | |
|---|---|
| Account | - Alphanumeric string symbolising a unique account |
| Coin | - Token/assets that people have invested in. |
| Execution Price | - Trading price |
| Size Tokens | - Trading amount |
| Size USD | - Trading Volume |
| Side | - Buy or Sell |
| Timestamp IST | - Date time in Indian Standard Time |
| Start Position | - Money in account before opening a trade |
| Direction | - The trade intent or action, (i.e. close long, open short, etc) |
| Closed PnL | - Realised Profit or Loss made from a trade |
| Transaction Hash | - Unique string denoting the record of a transaction |
| Order ID | - Unique identifier assigned to the order by the trading platform |
| Crossed | - Indicates if the trade crossed the spread. |
| Fee | - Trading fee charged for executing the trade |
| Trade ID | - Unique identifier for the specific trade instance |
| Timestamp | - Raw timestamp of the trade in UTC |

## Data Preprocessing and Normalization :

Given the differing timestamps of the two datasets the primary challenge is to align them for meaningful joint analysis. We extend the “fear & greed index - data” to match the market sentiment of the nearest timestamp from “Historical trade - data”.

In the Historical trade data there are given features like *'Transaction Hash', 'Order ID', 'Trade ID' and 'Timestamp'* that don’t have much meaning for analysis, so we can drop them

Now by joining the two data frames we have a joint dataset “merged_df” of the shape (211224, 16) that has no NULL values. Below is an overview of the obtained dataset.

```
#   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Account          211224 non-null  object
 1   Coin             211224 non-null  object
 2   Execution Price  211224 non-null  float64
 3   Size Tokens      211224 non-null  float64
 4   Size USD         211224 non-null  float64
 5   Side             211224 non-null  object
 6   Timestamp IST    211224 non-null  object
 7   Start Position   211224 non-null  float64
 8   Direction        211224 non-null  object
 9   Closed PnL       211224 non-null  float64
 10  Crossed          211224 non-null  bool
 11  Fee              211224 non-null  float64
 12  Timestamp        211224 non-null  datetime64[ns]
 13  timestamp        211224 non-null  int64
 14  value            211224 non-null  int64
 15  classification   211224 non-null  object
```

- There are 32 unique Accounts ,
- 246 different Coins/Assets,
- 2 Sides (i.e. Buy and Sell),
- 12 Directions - 'Buy', 'Sell' ,'Open Long' ,'Close Long' ,'Spot Dust Conversion' ,'Open Short', 'Close Short' ,'Long > Short' ,'Short > Long', 'Auto-Deleveraging', 'Liquidated Isolated Short' ,'Settlement'
- 5 Classifications - 'Extreme Greed', 'Extreme Fear', 'Fear', 'Greed', 'Neutral'

By grouping the dataset based on “Account” we can identify the people that have made most and least money.

- **Account with Maximum Daily PnL:**

| | |
|---|---|
| Account | 0xb1231a4a2dd02f2276fa3c5e2a2f3436e6bfed23 |
| Daily PnL | 2143382.597689 |
| Avg Daily Return % | 6.237844 |
| Volume USD | 56543565.23 |
| Trades | 14733 |
| Execution Price | 5610.012701 |
| Sentiment Value | 75 |
| Sentiment Label | Extreme Greed |

- **Account with Maximum Avg Daily Return %:**

| | |
|---|---|
| Account | 0xa520ded057a32086c40e7dd6ed4eb8efb82c00e0 |
| Daily PnL | 72846.484272 |
| Avg Daily Return % | 27.606413 |
| Volume USD | 861809.4 |
| Trades | 417 |
| Execution Price | 498.145845 |
| Sentiment Value | 37 |
| Sentiment Label | Fear |

**- Account with Minimum Daily PnL:**

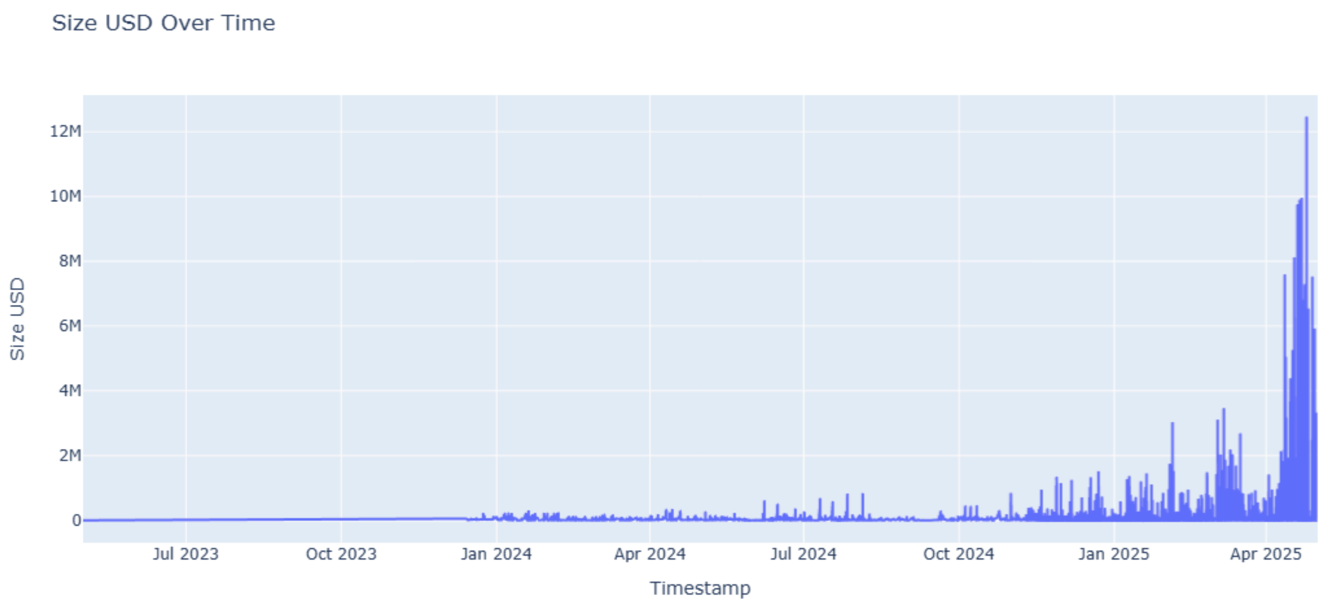
| | |
|---|---|
| Account | 0x8170715b3b381dffb7062c0298972d4727a0a63b |
| Daily PnL | -167621.124781 |
| Avg Daily Return % | 4.094751 |
| Volume USD | 10143758.13 |
| Trades | 4601 |
| Execution Price | 3742.612574 |
| Sentiment Value | 87 |
| Sentiment Label | Extreme Greed |

- **Account with Minimum Avg Daily Return %:**

| | |
|---|---|
| Account | 0x3998f134d6aaa2b6a5f723806d00fd2bbbbce891 |
| Daily PnL | -31203.599986 |
| Avg Daily Return % | -46.263735 |
| Volume USD | 1409902.0 |
| Trades | 815 |
| Execution Price | 258.788891 |
| Sentiment Value | 63 |
| Sentiment Label | Greed |

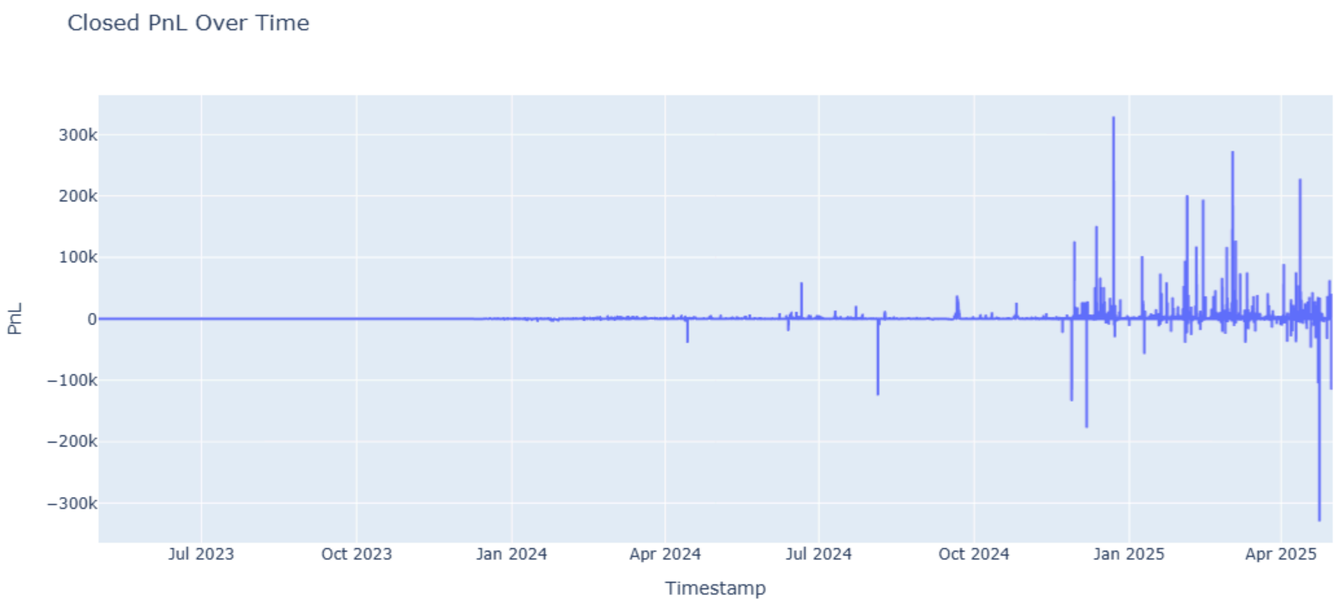
## Trades over Time

- Following graph shows the increase of Size USD (Volume) of assets being traded -



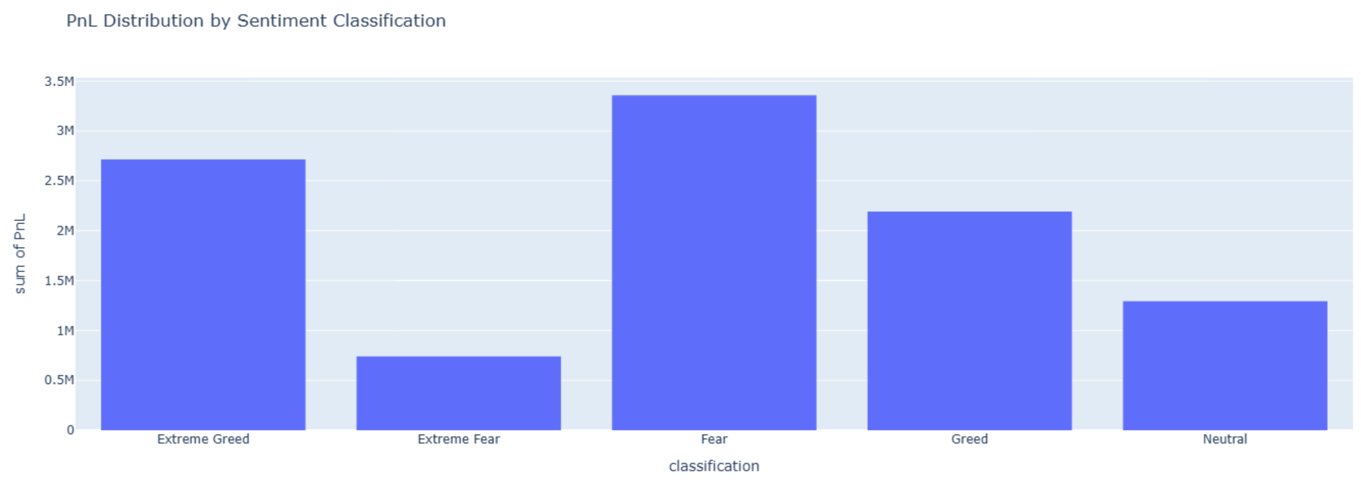
Traders have started investing with higher amounts since the beginning of 2025.

- Following graph shows the distribution of PnL achieved by traders over time



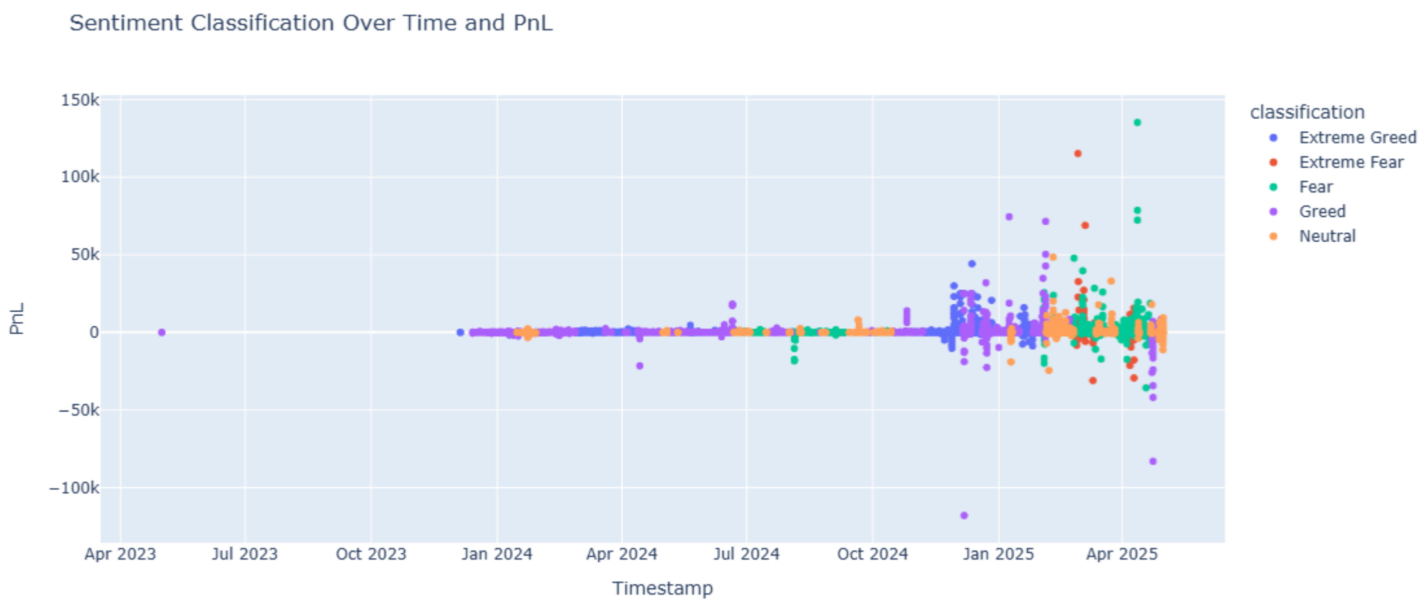
Traders are making progress mostly while losing money rarely , although the disparity has grown since 2025

- Following graph shows returns achieved by traders during different market sentiments



Traders are making the highest returns during Fear and Extreme Greed sentiment in the market.

- Following graph shows returns over time and with different sentiments



Traders are making most money during Fear and Extreme Greed sentiment in the market for the rest of the time the market remains mostly calm.

## Correlations Analysis :

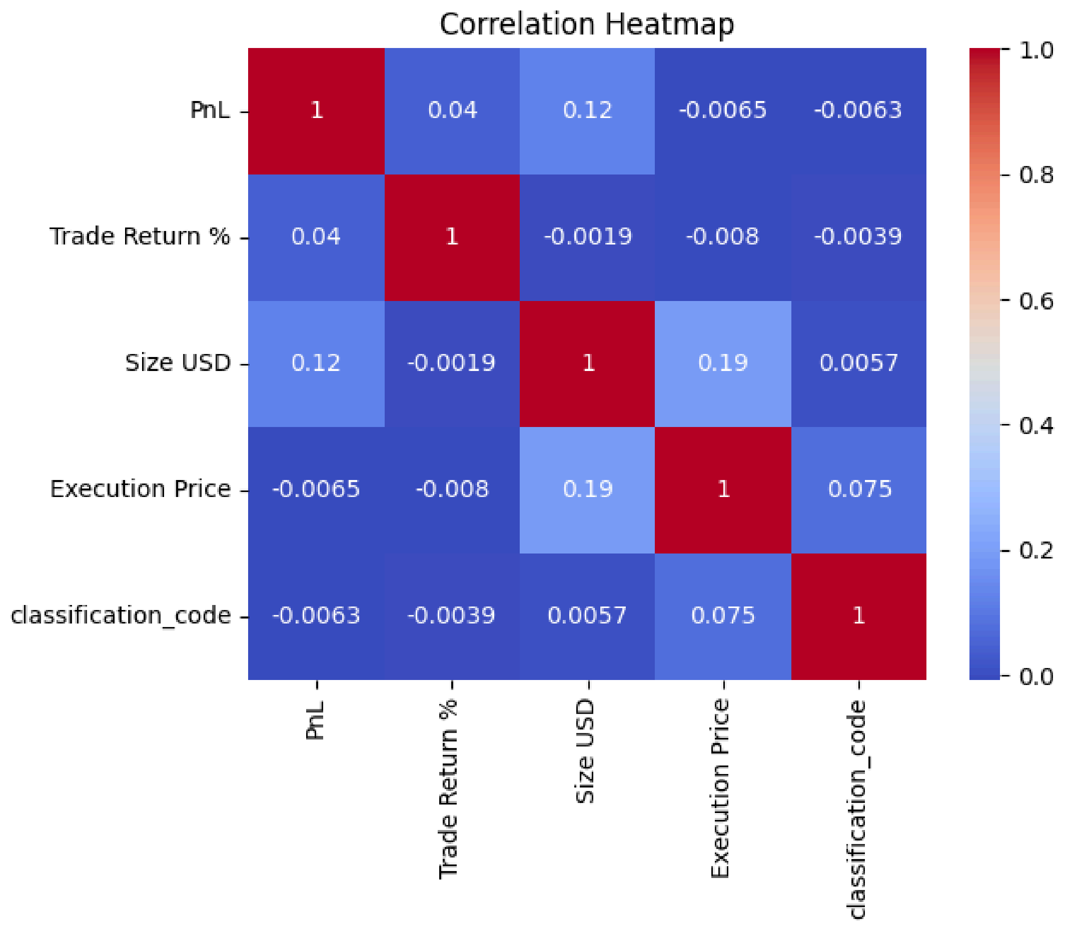
To quantify how various trading metrics align with the overall market sentiment, we computed both **Pearson** and **Spearman** correlation coefficients using the sentiment value.

- **Pearson Correlation :**
  It measures the strength and direction of a linear relationship between two continuous variables. Values close to:
  +1 imply a strong positive linear relationship,
  –1 indicate a strong negative linear relationship,
  0 suggests no linear correlation.

- **Spearman Correlation :**
  The Spearman rank correlation evaluates monotonic relationships (not necessarily linear), based on the ranked values of the data rather than their raw values. It is more robust to outliers and non-linear trends.

| Metric | Pearson | Spearman | Interpretation |
|---|---|---|---|
| **PnL** | –0.0063 | –0.0384 | Negligible negative correlation with sentiment. This indicates that **profit/loss outcomes are largely unaffected by the market sentiment** on a linear or ranked basis |
| **Trade Return %** | –0.0039 | –0.0450 | Similarly weak and slightly negative. Traders are **not consistently more profitable during Greed periods**; in fact, returns may marginally decrease, suggesting overconfident or riskier behavior. |
| **Size USD** | +0.0057 | –0.0150 | Very weak and inconsistent correlation with sentiment. **Trade sizes do not scale meaningfully with sentiment**, though slightly more volume is observed during Greed, especially in the Pearson sense. |
| **Execution Price** | +0.0752 | +0.0874 | The only mildly significant positive correlation. Higher market sentiment (Greed) tends to **coincide with higher execution prices**, possibly reflecting upward momentum in asset prices during euphoric periods. |
| **classification_code** | +1.0000 | +1.0000 | By design, this is perfectly correlated with sentiment as it is derived from the sentiment classification. |

- Following graph shows the correlation between numeric columns and classification code which are represented as 'Extreme Fear' : 0 , 'Extreme Greed' : 1, 'Fear' : 2 , 'Greed' : 3 , 'Neutral' : 4



These low correlations suggest that the features under consideration are largely independent of the sentiment variable, at least in a pairwise linear or rank-order sense. In other words, each feature captures a distinct dimension of trading behavior and is not simply a derivative of overall market sentiment.

This independence is advantageous in analytical modeling and feature selection, as it reduces the risk of multicollinearity - a condition where multiple features are highly correlated and could distort model interpretations or inflate variance.

Independence implies that while market sentiment may influence trade context, traders' profitability, position sizing, and price execution are also likely driven by strategy-specific, asset-specific, or risk management considerations—not just collective emotional states.

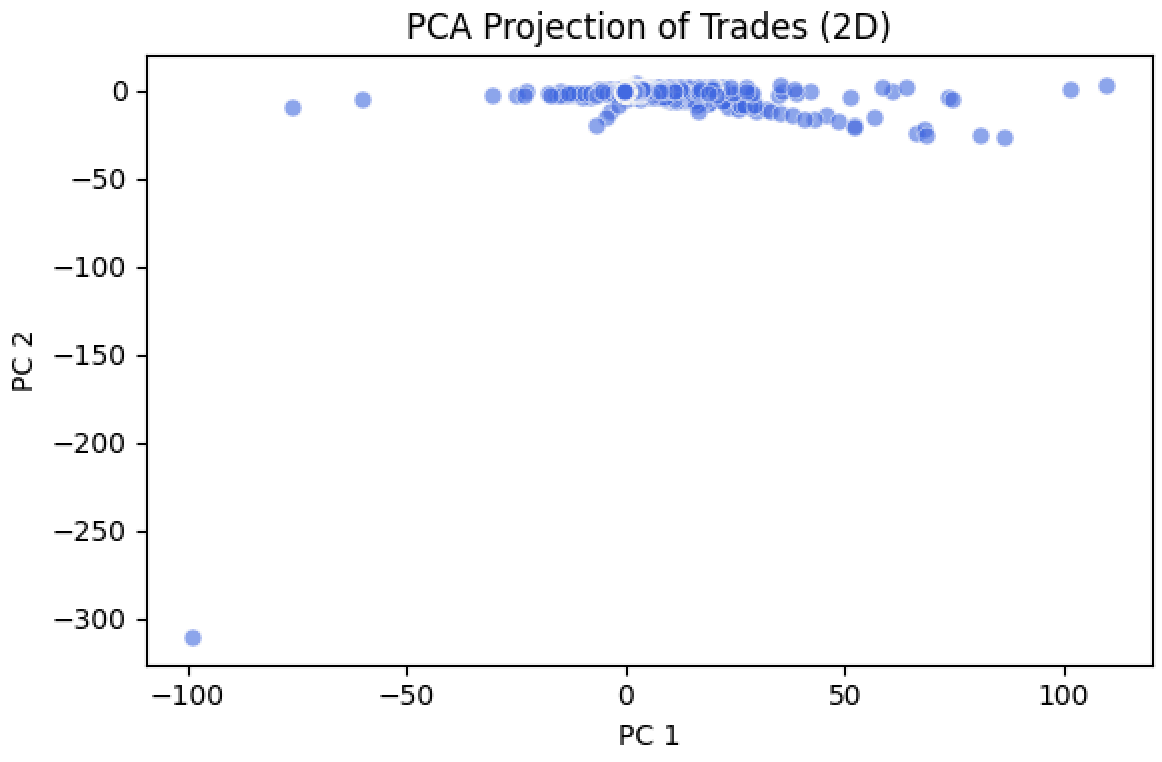
## Cluster Analysis : KMeans and PCA

Techniques like KMeans clustering and Principal Component Analysis (PCA) are sensitive to feature magnitudes because they rely on Euclidean distance or variance calculations. Without scaling, features with larger numeric ranges (like Volume Execution Price) would dominate the computation and distort the results.

So we scale the features using Standard Scaler which transforms the data such that it has Mean = 0 and Standard Deviation = 1.

- Principal Component Analysis

  To reduce the complexity of the trade dataset and enable better visualization and clustering, Principal Component Analysis (PCA) was applied to the standardized numerical features. PCA helps uncover the underlying structure of the data by projecting it into a new coordinate system defined by the directions of maximum variance.

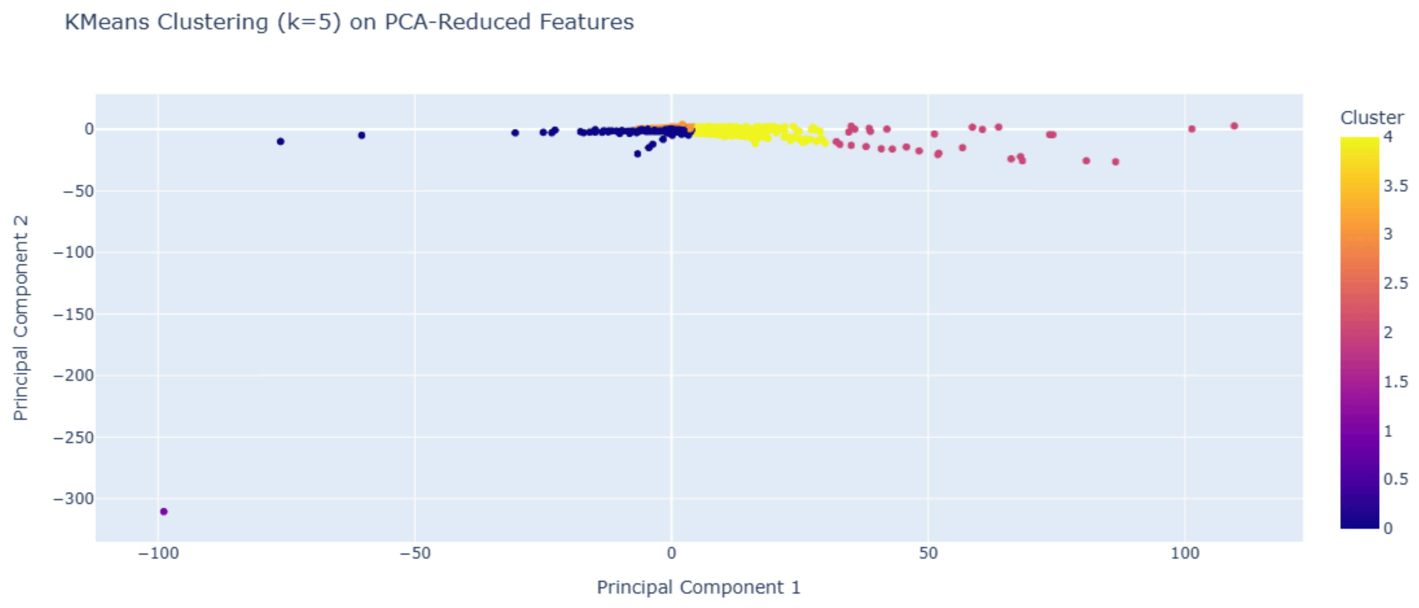


- This plot shows the projection of trade data onto the first two principal components.
- Most data points cluster tightly around the origin, indicating that the majority of variance is concentrated in a small region of the feature space
- PCA has successfully compressed the dataset into a form where variance can be analyzed visually. However, the data distribution is not uniform — it's dense in one region with sparse outliers.

## K-Means Clustering :

Clustering was done to segment trades into distinct behavioral groups based on key numerical features after dimensionality reduction via PCA.

The Elbow Method was applied by plotting the Sum of Squared Errors (SSE) against different values of k.



- KMeans clustering was applied with k=5, and the resulting clusters are color-coded.
- The clustering shows clearly separable regions along PC1, indicating that this component captures the most critical differentiators of trade behavior.
- KMeans has identified behaviorally distinct trade groups that weren't obvious in raw feature space.
- One or more clusters might correspond to traders influenced by Fear, others by Greed, or some executing high-risk strategies (e.g., large PnL variance).

## Modelling

To derive actionable intelligence from trade-level data, the analysis was extended beyond unsupervised exploration into a **supervised classification framework**, I converted the problem into a binary classification problem by adding another feature as ***Status.***

The Status feature has two values ‘ 1 ‘ represents Profit and ‘ 0 ‘ represents Loss for the data set that we already have the values are decided on the basis of PnL. If PnL is positive then Status is  1 else if negative then 0.

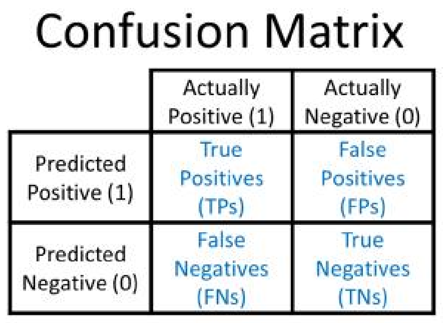
Next before training a model we need to encode all Categorical Features using One Hot Encoder.

### Problem : Imbalance Dataset

The dataset has 193,685 rows for profit against only 17,539 rows for loss which means only 9.05% data has loss as an output, so even if we make a **DUMB MODEL** that predicts profit at all times still the accuracy would be 91% so to make the smart model that understands the difference we will have to add some synthetic data to make the dataset balanced.

Balancing is done using **SMOTE** (Synthetic Minority Oversampling Technique), the new synthesized dataset has equal number of test cases for 1 and 0 as output values.

Different types of models were trained on the dataset that have given good results.
The results are judged by : Accuracy, Precision, Recall, F1-score, Confusion matrix and graphs like ROC (receiver operating characteristic) and PR (Precision recall)



Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

ROC curve plots True Positive Rate against False Positive Rate
PR curve plots Precision against Recall

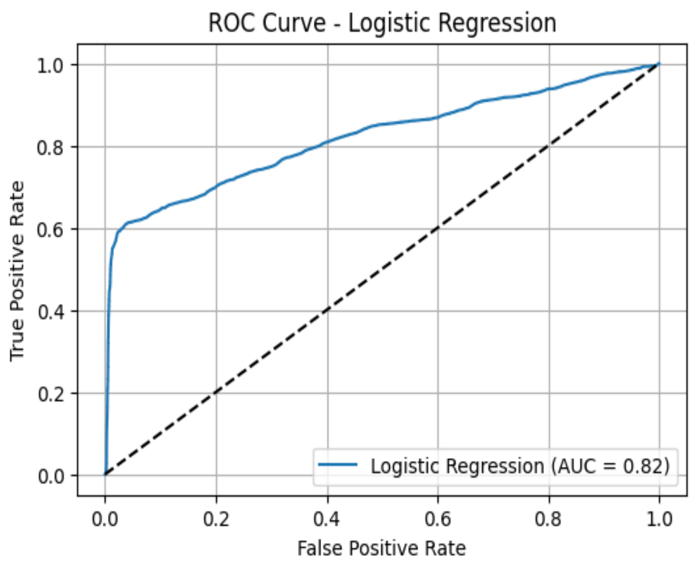
## Logistic Regression - Linear model

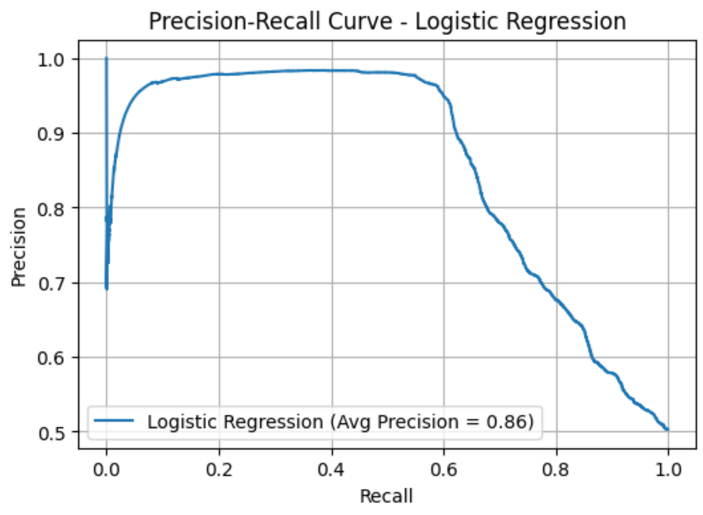
```
Accuracy :  0.7560342824689573
Classification Report :
               precision    recall  f1-score   support

         0.0       0.72      0.84      0.77     38488
         1.0       0.81      0.67      0.73     38986

    accuracy                           0.76     77474
   macro avg       0.76      0.76      0.75     77474
weighted avg       0.76      0.76      0.75     77474

Confusion Matrix :
 [[32459  6029]
 [12872 26114]]
```





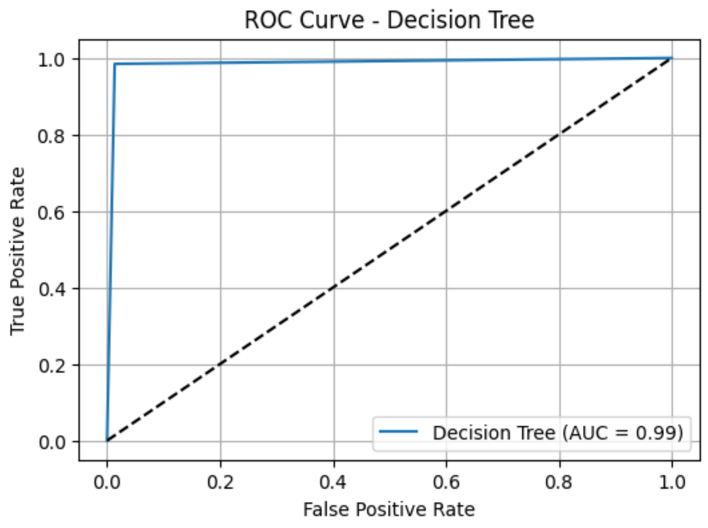
## Decision Tree Classifier - based on Binary Tree

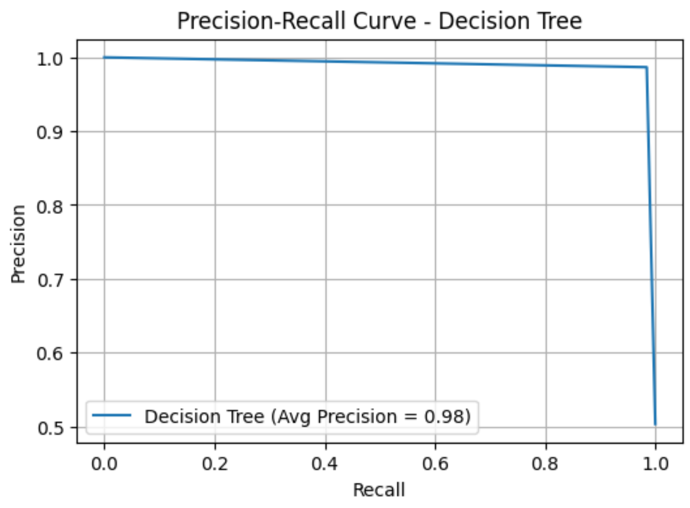
```
Training on Decision Tree
Results:
Accuracy :  0.9855306296305858
Classification Report :
               precision    recall  f1-score   support

         0.0       0.98      0.99      0.99     38488
         1.0       0.99      0.98      0.99     38986

    accuracy                           0.99     77474
   macro avg       0.99      0.99      0.99     77474
weighted avg       0.99      0.99      0.99     77474

Confusion Matrix :
 [[37970   518]
 [  603 38383]]
```





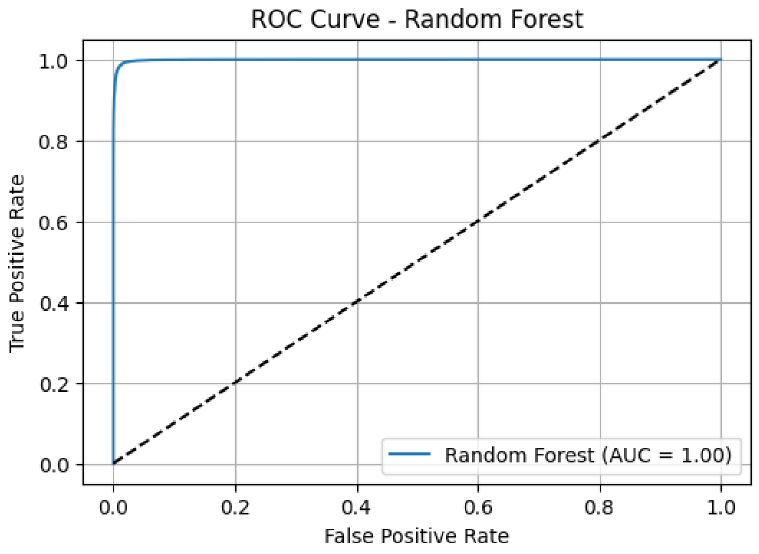
## Random Forest Classifier - based on Ensemble Learning

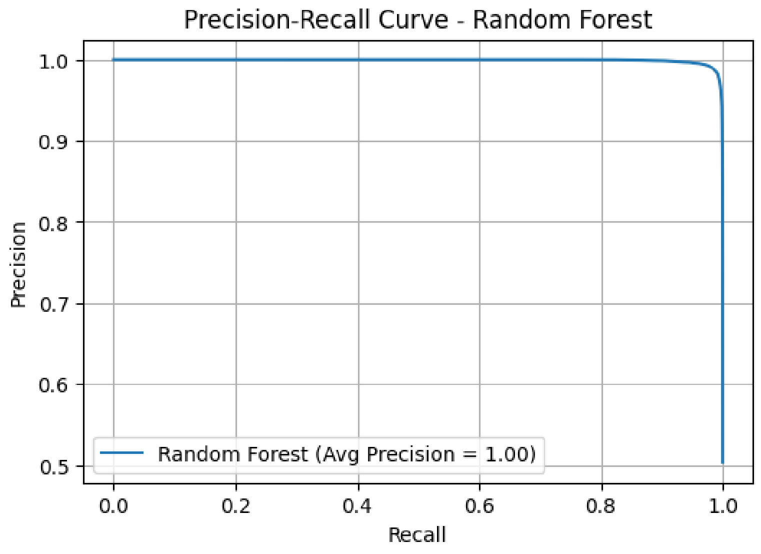
```
Accuracy :  0.9869117381315022
Classification Report :
               precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     38488
         1.0       0.99      0.99      0.99     38986

    accuracy                           0.99     77474
   macro avg       0.99      0.99      0.99     77474
weighted avg       0.99      0.99      0.99     77474

Confusion Matrix :
 [[37992   496]
 [  518 38468]]
```





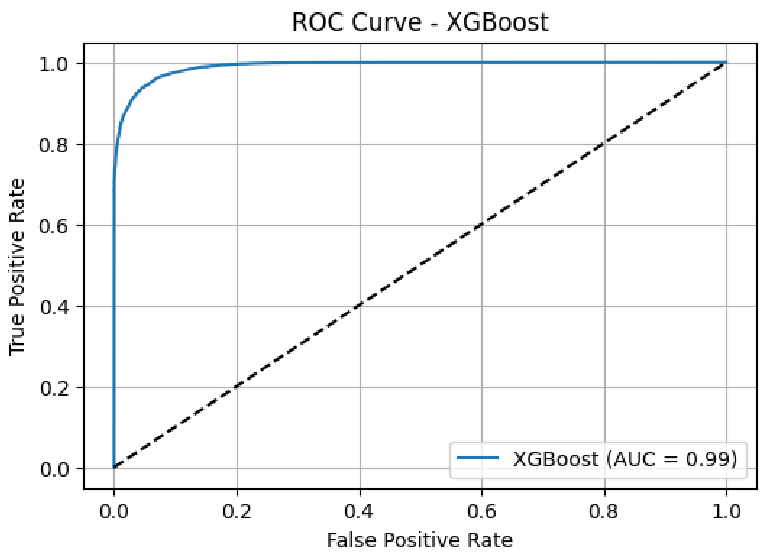
## XGBoost Classifier - Boosting algorithm

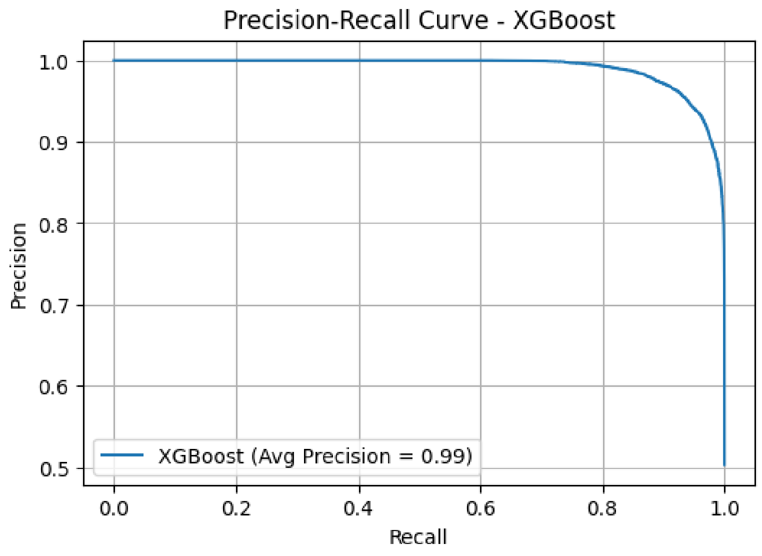
```
Accuracy :  0.9452332395384258
Classification Report :
               precision    recall  f1-score   support

         0.0       0.94      0.95      0.95     38488
         1.0       0.95      0.94      0.95     38986

    accuracy                           0.95     77474
   macro avg       0.95      0.95      0.95     77474
weighted avg       0.95      0.95      0.95     77474

Confusion Matrix :
 [[36537  1951]
 [ 2292 36694]]
```





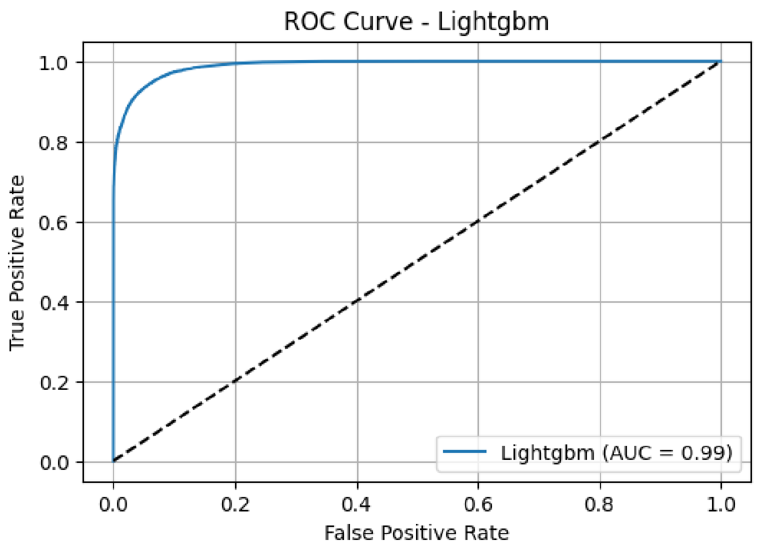
## Lightgbm Classifier - Boosting algorithm

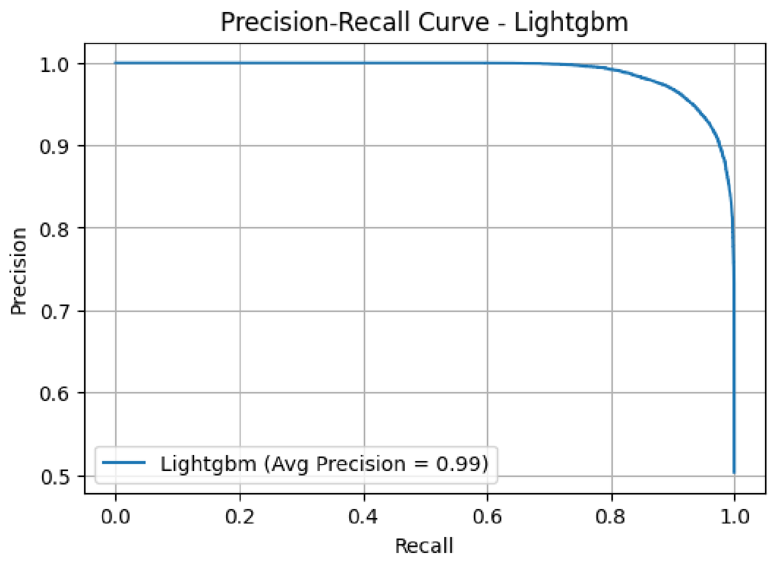
```
Accuracy :  0.9407672251335932
Classification Report :
               precision    recall  f1-score   support

         0.0       0.93      0.95      0.94     38488
         1.0       0.95      0.93      0.94     38986

    accuracy                           0.94     77474
   macro avg       0.94      0.94      0.94     77474
weighted avg       0.94      0.94      0.94     77474

Confusion Matrix :
 [[36609  1879]
 [ 2710 36276]]
```





## Evaluation of Models

- Logistic Regression has an accuracy of 75.6 % which is very less so it is rejected.

- Decision Tree has an accuracy score of 98.5 % which is very high for financial data, it is most likely overfitting on the training dataset.

- Random Forest has an accuracy score of 98.6 % which is very high for financial data, it is most likely overfitting on the training dataset.

- XGBoost has an accuracy of 94.5 % which appears realistic so this model can be used.

- LightGBM has an accuracy of 94 %, the model appears good but XGBoost is slightly better.

So we can choose XGBoost as our base model and tune it further to improve its performance. As of now XGBoost model has 1951 False Positive and 2292 False Negatives, this is an issue that can be resolved using hyper-parameter tuning.

**Bayesian Optimization** : builds a probabilistic model of the objective function and uses it to select the most promising hyperparameters to evaluate next.
This makes it :-

- **Efficient**: Finds optimal or near-optimal values in fewer iterations.
- **Informed**: Chooses future samples based on past evaluation results.
- **Powerful for expensive models**: Especially beneficial when model training is computationally expensive or time-consuming.

After Hyperparameter tuning the best parameters observed are :
**Best parameters**: 'n_estimators': 351, 'max_depth': 12, 'learning_rate': 0.28552046434530387, 'subsample': 0.4154485679833389, 'colsample_bytree': 0.9395086164541318

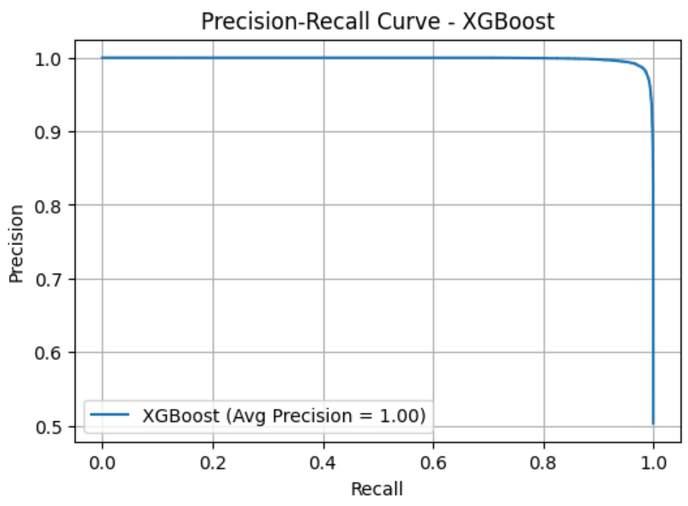
Evaluation Metrics of Hyperparameter Tuned XGBoost Classifier model:

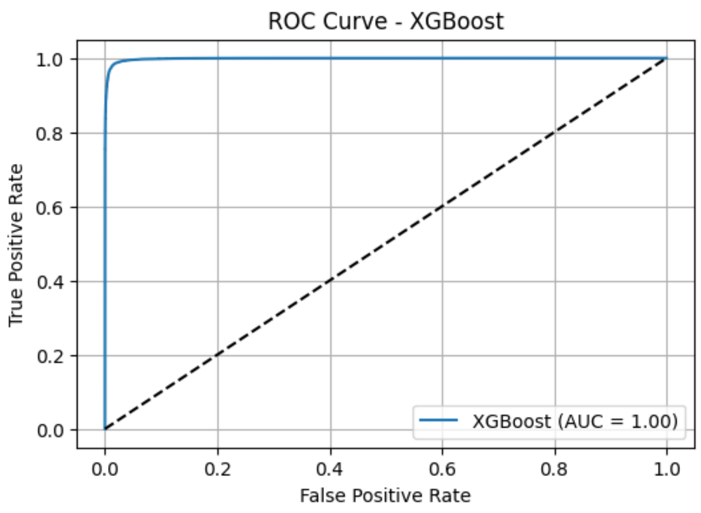
```
Accuracy :  0.9836203113302527
Classification Report :
               precision    recall  f1-score   support

         0.0       0.98      0.98      0.98     38488
         1.0       0.98      0.98      0.98     38986

    accuracy                           0.98     77474
   macro avg       0.98      0.98      0.98     77474
weighted avg       0.98      0.98      0.98     77474

Confusion Matrix :
 [[37889   599]
 [  670 38316]]
```





We can clearly observe that False Positives and False Negatives have been reduced significantly and a better model has been obtained by tuning it.

## Conclusion

In this study, we examined the intersection between **market sentiment** (as captured by the Fear and Greed Index) and **real-world trading behavior** (sourced from on-chain trading data), transforming raw transactional data into an insightful classification problem. Through rigorous preprocessing, exploratory analysis, correlation study, dimensionality reduction (PCA), clustering (KMeans), and model building enhanced by **Bayesian Optimization**, we were able to draw meaningful patterns from the data.

The trained model can also be used to form trading strategies that will help us decide if the trade can result in profit or loss , although for practical implementation of the strategy the model needs to be kept continuously training on new data from the market.

We have successfully trained a Machine Learning Model to understand the given dataset and provide output as Profit or Loss.