TRADE DATA ANALYSIS

Dataset Information: Historical trade data from various Binance accounts over 90 days, containing:

Port_IDs: Unique identifiers for accounts.
Trade_History: Historical trades with details like timestamp, asset, side (BUY/SELL), price, and more.

Objective: Analyze the dataset to calculate financial metrics for each account, rank them, and provide a top 20 list.

Metrics to Calculate:
ROI (Return on Investment)
PnL (Profit and Loss)
Sharpe Ratio
MDD (Maximum Drawdown)
Win Rate
Win Positions
Total Positions

Steps to Complete the Task:

Data Exploration and Cleaning: Load and inspect the dataset, handle missing values.

Feature Engineering: Determine feature importance and create a scoring system with weighted scores.

Ranking Algorithm: Develop an algorithm to rank accounts based on calculated metrics.

Documentation: Provide a concise report on methodology, findings, and assumptions.

SOLUTION

I have used Jupyter Notebook with Python to solve this problem. The final python script on the Notebook is attached for your reference, and these are the details of my Data Analysis on the dataset provided:

Step 1: Data Exploration and Cleaning

1. Load the Dataset:
    - I have used pandas library to load the CSV file into a DataFrame.
    - Check for missing values and handle them appropriately (e.g., I have removed the null values and duplicate rows in order to clean data).
    - Inspected the first few rows (using head()) to understand the structure and types of data.
2. Convert Data Types:
    - Ensured timestamps are in datetime format.
    - Converted the "Trade History" JSON format into rows that we can analyse and use and manipulate using the ast package and json_normalise() functions.
3. Handle Duplicates & Anomalies:
    - Checked for duplicate entries and remove if necessary.
    - Also removed null values which were not recognised.

Step 2: Feature Engineering

1. Classify Positions:
    - Combined side (BUY/SELL) and positionSide to determine whether a trade opens or closes a position.
    - Example categories: long_open, long_close, short_open, short_close.
2. Calculate Key Metrics Per Account:
    - PnL (Profit and Loss):

PnL =∑realizedProfit

- ROI (Return on Investment):

  ROI = (PnL/InitialInvestment)×100

  Estimated initial investment by summing quantity of first trades.

- Win Rate:

  Win Rate = (Win Positions/Total Positions)×100

  A win position is when realizedProfit > 0.

- Sharpe Ratio:

  Sharpe Ratio = Mean (Daily Returns) / Std Dev (Daily Returns)

  Calculate daily returns based on closing trades.

- MDD (Maximum Drawdown):

  Track peak portfolio value and compute largest percentage drop.

Step 3: Ranking Algorithm

NOTE: In the first code, I had included Sharpe Ratio after ROI and PnL metrics, but after further calculation I saw that the Sharpe Ratio was 0.0 for all Port_IDs, which means that the fund's returns are exactly equal to the returns of a risk-free asset, indicating that the fund is not providing any additional return for the risk it takes on.
So, I skipped including the Sharpe Ratio in my Final Score Calculation and used the rest of the metrics:

Normalization Technique

- Min-Max Normalization
- Scales all metrics to 0-1 range
- Preserves relative performance differences

Scoring Formula

Final Score = (Normalized_ROI * 0.35 + Normalized_PnL * 0.3 + Normalized_Win_Rate * 0.25 + (1 - Normalized_MDD) * 0.1)

Weighting Rationale

- ROI: 35% (Most critical performance indicator)
- PnL: 30% (Absolute profit measurement)
- Win Rate: 25% (Consistency indicator)
- MDD: 10% (Risk management factor)

Key Ranking Principles

- Positive scores indicate better performance
- Centered around average performance
- Allows comparison across different trading accounts

Limitations and Challenges

1. Data Dependent Limitations
    - Performance heavily relies on quality and completeness of input data
    - Short trading periods might skew results
    - There were outliers and missing values all of whom had to be dealt with
2. Metric Calculation Constraints
    - Assumes linear relationship between metrics
    - Does not account for complex trading strategies
    - Static weights might not suit all trading styles
3. Risk Assessment
    - Simple MDD calculation might not capture nuanced risk
    - Does not incorporate market volatility
    - Ignores external market conditions
4. Computational Limitations

    - Requires complete trade history
    - Sensitive to outliers
    - Assumes consistent trading behavior
5. Performance Ranking Caveats
    - Rankings are relative within the dataset
    - Past performance does not guarantee future results
    - Does not predict future trading success

Potential Improvements

- Implement machine learning models
- Incorporate more sophisticated risk metrics
- Dynamic weight adjustment
- Time-series based analysis
- Advanced drawdown calculations

Explanation of the Ranking System

The ranking system is designed to evaluate trading strategies or portfolios based on five key performance metrics. Each metric is weighted differently based on its importance. The Final Score is a weighted sum of these metrics, where higher scores indicate better performance.

Breakdown of Metrics and Weights

1. Return on Investment (ROI) – 35% weight
    - Measures the percentage gain or loss relative to the initial investment.
    - Higher is better because a higher ROI means greater profitability.
    - Weight: 0.35 → This gets the highest weight since profitability is a key goal.
2. Sharpe Ratio – 0% weight
    - (Explained above)
3. Profit and Loss (PnL) – 30% weight
    - Represents the absolute profit or loss in monetary terms.

    ○ Higher is better because a larger PnL means more money earned.
    ○ Weight: 0.30 → Important but slightly less than ROI since absolute profit depends on account size.
4. Win Rate – 25% weight
    ○ Percentage of trades that were profitable.
    ○ Higher is better because more winning trades indicate consistency.
    ○ Weight: 0.25 → Less weight because a high win rate doesn't always mean high profitability.

5. Maximum Drawdown (MDD) – (-10%) weight

    ○ Measures the largest drop in capital from peak to trough.
    ○ Lower is better (negative impact), so we multiply by -0.1 to penalize higher drawdowns.
    ○ Weight: -0.1 → It has the least weight but still matters since large drawdowns indicate risk.

JUPYTER NOTEBOOK CODE

JUPYTER NOTEBOOK PYTHON SCRIPT:

```
"""
Jupyter Notebook: Binance Data Analysis
"""

import pandas as pd
import numpy as np
import ast

# Load dataset
file_path  =  "/Users/ashokkumarsinha/Downloads/TRADES.csv"   #  Update  with
actual path
df = pd.read_csv(file_path)

# Data Cleaning and Preprocessing
```

```
df['Trade_History'] = df['Trade_History'].astype(str)
df['Trade_History'] = df['Trade_History'].apply(lambda x: '[]' if x in ['nan',
'None', ''] else x)
df['Trade_History'] = df['Trade_History'].apply(ast.literal_eval)
df = df.explode('Trade_History')
df = df.dropna(subset=['Trade_History'])
df = df.join(pd.json_normalize(df['Trade_History']))
df = df.drop(columns=['Trade_History'])

# Convert timestamp and clean data
df['timestamp'] = pd.to_datetime(df['time'], unit='ms')
df.drop(columns=['time'], inplace=True)
df.dropna(inplace=True)

# Classify positions
df['position_type'] = df['side'] + '_' + df['positionSide']

# Calculate Metrics per Account
def calculate_account_metrics(group):
    metrics = {}

    # Profit and Loss
    metrics['PnL'] = group['realizedProfit'].sum()

    # ROI Calculation
    total_investment = group['quantity'].sum()
        metrics['ROI']  =  (metrics['PnL']  /  total_investment)  *  100  if
total_investment != 0 else 0

    # Win Rate
    win_trades = group[group['realizedProfit'] > 0]
    metrics['Win Rate'] = (len(win_trades) / len(group)) * 100


     # Win Positions
    metrics['Win Positions'] = len(win_trades)
    metrics['Total Positions'] = len(group)

    # Daily Returns
    group['daily_returns'] = group['price'].pct_change()

    # Maximum Drawdown
    def calculate_mdd(profits):
```

```
        cumulative_profits = profits.cumsum()
        max_so_far = cumulative_profits.cummax()
        drawdown = (cumulative_profits - max_so_far) / max_so_far
        return drawdown.min() if len(drawdown) > 0 else 0

    metrics['MDD'] = calculate_mdd(group['realizedProfit'])

    return pd.Series(metrics)

# Group by Port_IDs and calculate metrics
account_metrics =
df.groupby('Port_IDs').apply(calculate_account_metrics).reset_index()

# Normalize Metrics (Min-Max Scaling)
def min_max_normalize(series):
    min_val = series.min()
    max_val = series.max()
    return (series - min_val) / (max_val - min_val) if max_val != min_val else
series

metrics_to_normalize = ['ROI', 'PnL', 'Win Rate']
for metric in metrics_to_normalize:
                               account_metrics[f'Normalized_{metric}'] =
min_max_normalize(account_metrics[metric])

# Improved Final Score Calculation
account_metrics['Final Score'] = (
    account_metrics['Normalized_ROI'] * 0.35 +        # Increased weight
    account_metrics['Normalized_PnL'] * 0.3 +         # Increased weight
     account_metrics['Normalized_Win Rate'] * 0.25 +   # Increased weight
      (1 - min_max_normalize(account_metrics['MDD'])) * 0.1   # 10% weight
(inverse of MDD)
)

# Rank Accounts
df_ranked = account_metrics.sort_values(by='Final Score', ascending=False)
df_top_20 = df_ranked.head(20)

# Save Results
df_ranked.to_csv("/Users/ashokkumarsinha/Downloads/ranked_accounts.csv",
index=False)
```

```python
df_top_20.to_csv("/Users/ashokkumarsinha/Downloads/top_20_accounts.csv",
index=False)

# Print Results
print("Top 20 Accounts:")
print(df_top_20[['Port_IDs', 'Final Score', 'ROI', 'PnL', 'Win Rate']])
print("\nRanking Complete. Top 20 Accounts saved.")

# Verify Score Distribution
print("\nFinal Score Distribution:")
print(account_metrics['Final Score'].describe())
```

```
Top 20 Accounts:
                Port_IDs  Final Score       ROI            PnL  Win Rate
15   3887577207880438784     0.999645  3.845048  169088.642497     100.0
28   3932103299427844097     0.805628  3.845048   59735.195497     100.0
78   4000877324693233921     0.782811  3.848953   46674.638252     100.0
98   4021669203289716224     0.734294  3.464756   39020.070699     100.0
68   3993014919980212480     0.729217  3.848953   16467.436391     100.0
52   3966142151544441601     0.726062  3.845048   14889.298496     100.0
83   4008537296438699777     0.713896  3.841143    8232.373929     100.0
48   3956076827719377409     0.713240  3.464756   27153.902549     100.0
128  4033614723417828608     0.712710  3.841143    7564.237454     100.0
97   4021243448368889856     0.708749  3.848953    4931.227103     100.0
57   3977116548751698176     0.708637  3.841143    5268.219947     100.0
67   3991414786174551297     0.708144  3.845048    4790.608179     100.0
12   3879821005658659073     0.704502  3.848953    2537.207815     100.0
32   3939318616482048768     0.704312  3.848953    2430.167116     100.0
50   3960874214179953664     0.704185  3.841143    2758.818696     100.0
146  4040843843196854529     0.702178  3.845048    1427.761184     100.0
117  4030626027667524352     0.701865  3.848953    1051.148450     100.0
137  4037179073830813185     0.701414  3.848953     797.033399     100.0
82   4006366295148391425     0.701397  3.848953     787.423050     100.0
30   3936410995029308417     0.701136  3.841143    1040.342736     100.0

Ranking Complete. Top 20 Accounts saved.

Final Score Distribution:
count    56.000000
mean      0.671076
std       0.097029
min       0.406205
25%       0.666801
50%       0.673829
75%       0.704216
max       0.999645
Name: Final Score, dtype: float64
```

OBSERVATIONS

Based on the weighted calculations, our results differ when we try to find the top 20 accounts based on ranking. According to the industry and company needs, we tweak the ranking metrics in order to see which accounts are the best performing.

Sharpe Ratio depends on the realizedProfits and Daily Returns, and in this dataset it leads to NaN or negative similar results for all Ports. So we ignored the Sharpe Ratio (0.0) in this case.

The Data Cleaning and Manipulation steps required more effort since the data was redundant and stored in a column under individual JSON files. The errors as well as the rectifications are available on the Jupyter Notebook (TradeAnalysis.ipynb) file attached.

The highest final score is 0.99
The .csv files have all the required metrics, and rankings.

Here are the direct links to the files shared on the same folder as this:

CSV file containing calculated metrics: ranked_accounts.csv (https://drive.google.com/file/d/1WJ6v8I8ZQKt8Iq3UPfQp0dsi6wiWcDv-/view?usp=drive_link)

List of top 20 accounts based on ranking: top_20_accounts.csv (https://drive.google.com/file/d/1MaFaSRxecj7MIPFQZHrtNW_UjmfaZ8Ok/view?usp=drive_link)

Final Jupyter Notebook Script: Binance.ipynb (https://drive.google.com/file/d/11AwjeG03DHpaQGc4cumFZQj0t4DCbEoV/view?usp=drive_link)