**Project Documentation: Compound Risk Score Engine**

**Objective**

The goal of this project is to **build a risk scoring system** for wallet addresses interacting with the **Compound Protocol (V2/V3)**. By analyzing user behavior and transaction history, we derive a **quantitative risk score** ranging from **0 (low risk)** to **1000 (high risk)**.

**1. Data Collection**

We collect historical on-chain transaction data for 100 wallet addresses using the **Covalent API**. The data includes actions such as:

* supply (depositing tokens),
* borrow,
* repay,
* liquidation.

Each record contains the wallet address, transaction type, amount, token, and timestamp.

**2. Feature Engineering**

We derive key behavioral metrics from the raw transaction logs. These features reflect a user's financial activity and reliability in the protocol:

| **Feature** | **Description** | **Risk Influence** |
| --- | --- | --- |
| total\_borrows | Total amount borrowed across all tokens | ↑ Higher = Higher Risk |
| total\_supplies | Total tokens supplied (collateral) | ↓ Higher = Lower Risk |
| borrow\_to\_supply\_ratio | Ratio of borrow/supply | ↑ Higher = Higher Risk |
| num\_liquidations | Times a wallet was liquidated | ↑ Higher = Higher Risk |
| repay\_ratio | Ratio of repaid amount to total borrow | ↓ Higher = Lower Risk |
| transaction\_count | Total relevant DeFi actions | ↔ Neutral / Informative |

**Note**: All amounts are normalized to USD for consistency.

**3. Normalization Method**

We apply **Min-Max Normalization** on all numerical features to bring them into a **[0, 1] range**, using the formula:

Normalized Value=x−xmin​​/xmax-xmin

This ensures all features contribute proportionally to the score, preventing large values (like borrow amounts) from dominating the calculation.

**4. Risk Scoring Logic**

We compute the final risk score on a **0–1000 scale** by aggregating weighted feature contributions. The weights reflect how strongly each feature indicates risk:

risk\_score = (

0.3 \* normalized\_borrow\_to\_supply\_ratio +

0.25 \* normalized\_total\_borrows +

0.2 \* normalized\_num\_liquidations +

0.15 \* (1 - normalized\_repay\_ratio) + # higher repay = lower risk

0.1 \* normalized\_transaction\_count

)

risk\_score \*= 1000 # Scale to 0–1000

**Interpretation**:

* Score close to **0**: Safe user with strong collateral and repay behavior.
* Score close to **1000**: Risky user, frequently borrows, often liquidated, rarely repays.

**5. Analysis & Insights**

We visualize and analyze the distribution of scores using:

* **Histogram** – shows overall spread of scores.
* **Top 10 Highest Risk Wallets** – potential liquidation candidates.
* **Top 10 Lowest Risk Wallets** – reliable borrowers/suppliers.

**Output**

The final output is a **DataFrame** with the following columns:

* wallet\_address
* risk\_score

this can be exported to CSV/Excel file for further analysis.

**Use machine learning (e.g., clustering or classification) to validate risk groupings**.

To **validate and enhance the quality of our wallet risk scores**, we apply **unsupervised machine learning techniques**—specifically **clustering (KMeans)**—to detect natural groupings of wallets based on their on-chain behavior. This helps determine whether our scoring logic aligns with actual wallet behavior patterns.

**Why Use Clustering?**

* **Unsupervised Validation**: Since we don’t have labeled "risky" vs. "safe" wallet data, clustering allows us to group similar wallets and observe emergent patterns.
* **Data-Driven Grouping**: Clustering helps verify if wallets with similar feature behaviors fall into similar risk score ranges.
* **Anomaly Detection**: Clusters with high borrow and low repay/mint activity can act as early warning groups.
* **Interpretability**: Helps explain and potentially refine the scoring logic by analyzing the characteristics of each cluster.

**Implementation Overview**

We used **KMeans Clustering** from sklearn on the **normalized feature dataset**:

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

# Normalize features again using StandardScaler (better for KMeans)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(wallet\_features)

# Apply KMeans

kmeans = KMeans(n\_clusters=3, random\_state=42)

wallet\_features['cluster'] = kmeans.fit\_predict(X\_scaled)

**Cluster Analysis**

* **Cluster 0**: High borrow, low repay/mint → **High risk**
* **Cluster 1**: High mint and repay, low borrow → **Low risk**
* **Cluster 2**: Moderate in all features → **Medium risk**

We then compared the **risk scores** generated by our logic with the cluster assignments:

* **Alignment Found**: Wallets in Cluster 0 mostly had risk scores > 800.
* **Mismatches Identified**: Some wallets had low scores but were placed in risky clusters → indicates areas to refine scoring logic.

**Visualization**

A scatter plot of principal components (PCA) revealed distinct wallet behavior clusters:

python

CopyEdit

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

pca = PCA(n\_components=2)

components = pca.fit\_transform(X\_scaled)

plt.scatter(components[:, 0], components[:, 1], c=wallet\_features['cluster'], cmap='viridis')

plt.title("Wallet Clustering using PCA")

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.colorbar(label='Cluster ID')

plt.show()

**Conclusion**

Using **KMeans clustering**:

* **Improved confidence** in our scoring logic through unsupervised grouping validation.
* **Revealed edge cases** and potential scoring misalignments.
* **Suggested refinements** to feature weights or introduction of new features like time-based behaviors.

This enhancement adds **credibility** and **adaptability** to the wallet risk scoring system and lays the groundwork for supervised classification if labeled datasets become available in the future.

**Notes**

* Wallets with no transactions are given a default score of 0 or ignored.
* Scores are calculated only on available historical data and do not account for off-chain behavior.