NETWORK SCIENCE PROJECT

UNVEILING THE BITCOIN ALPHA TRUST NETWORK - A NETWORK SCIENCE APPROACH

DELIVERABLE 2

GROUP 33

VAIBHAV GUPTA - 2022553
RATNANGO GHOSH - 2022397

1. Introduction: Understanding Trust Networks

Trust networks represent the backbone of peer-to-peer cryptocurrency platforms, where user interactions lack traditional institutional safeguards. In the Bitcoin Alpha network, the explicit trust-rating system serves as a proxy for reputation, influencing transaction decisions and network growth. This research applies advanced network analytics and machine learning to decode behavioral archetypes and their role in market dynamics.

2. Methodology

Our analytical approach combined structural network analysis with behavioral profiling:

1. **Network Feature Extraction:** Calculated centrality metrics (betweenness, eigenvector), degree distributions, and trust-related features
2. **Dimensionality Reduction:** Applied PCA and t-SNE to visualize high-dimensional relationships
3. **Cluster Analysis:** Identified distinct user segments based on network position and trust behavior
4. **Risk Profiling:** Quantified risk attitudes and trust volatility across segments

3. User Segment Profiles

Our analysis revealed five distinct user archetypes with characteristic behaviors:

3.1 Power Users/Hubs (Cluster 0)

Size: 158 users (4.18% of network)

Feature	Value	Interpretation
Connectivity	51.8 out / 49.3 in	High engagement & symmetrical connections
Trust Rating	1.40 given / 1.75 received	Net trust recipient
Risk-Taking	0.92 (low variation)	Consistently high risk tolerance
Trust Volatility	7.58 (moderate)	Selective trust distribution
Centrality	High	Strategic network position

Behavioral Profile: These users function as network hubs with calculated risk-taking behavior. Their high connectivity combined with moderate trust volatility indicates sophisticated discrimination in forming relationships. They maintain high risk tolerance with remarkable consistency (SD 0.14), suggesting comfort with uncertainty is necessary for their central network position.

3.2 Casual Users (Cluster 1)

Size: 3,186 users (84.24% of network)

Feature	Value	Interpretation
Connectivity	3.5 out / 3.8 in	Minimal engagement
Trust Rating	1.19 given / 1.06 received	Neutral trust exchange
Risk-Taking	0.83 (high variation)	Variable risk attitudes
Trust Volatility	0.95 (very low)	Consistent trust judgments
Centrality	Low	Peripheral network position

Behavioral Profile: Representing the vast majority of users, this segment exhibits limited network engagement with moderate risk tolerance that varies significantly across members (SD 0.36). Despite this variation in risk attitude, they demonstrate remarkably consistent trust judgments, suggesting formation of small, stable trust circles within their limited network activity.

3.3 Super Connectors/Authorities (Cluster 2)

Size: 15 users (0.40% of network)

Feature	Value	Interpretation
Connectivity	208.7 out / 180.7 in	Extraordinary network reach
Trust Rating	1.13 given / 2.18 received	Significant trust accumulation
Risk-Taking	0.88 (low variation)	Strategically high risk tolerance
Trust Volatility	10.36 (high)	Highly discriminating
Centrality	Extremely high	Network influencers
Negative Trust	25.4 given / 6.5 received	Active in identifying threats

Behavioral Profile: This elite minority represents the network's authorities with exceptional connectivity and influence. Their risk profile never drops below 0.47, establishing a minimum threshold of risk comfort required for their position. They receive substantially more trust than they give, accumulating social capital while maintaining highly discriminating standards (high trust volatility), including actively flagging potential bad actors.

3.4 Trusting Peripheral Users (Cluster 3)

Size: 322 users (8.51% of network)

Feature	Value	Interpretation
Connectivity	2.7 out / 2.7 in	Limited engagement
Trust Rating	5.95 given / 3.49 received	Extremely generous trust givers
Risk-Taking	0.998 (minimal variation)	Universal high trust
Trust Volatility	4.77 (moderate)	Some discrimination
Risk Minimum	0.857	High baseline trust

Behavioral Profile: These users exhibit nearly maximal risk tolerance (0.998) with extraordinary consistency across all members (SD 0.014). Despite limited connectivity, they extend remarkably high trust ratings, functioning as "optimistic adopters" who facilitate network growth through their openness to new connections. Their trust generosity exceeds what they receive in return.

3.5 Skeptics/Vigilantes (Cluster 4)

Size: 101 users (2.67% of network)

Feature	Value	Interpretation
Connectivity	7.7 out / 8.0 in	Moderate engagement
Trust Rating	-1.11 given / -0.21 received	Net negative trust orientation
Risk-Taking	0.65 (lowest)	Conservative risk attitude
Trust Volatility	35.52 (extreme)	Binary trust decisions
Negative Trust	High rates given and received	Contentious relationships

Behavioral Profile: This segment functions as the network's "immune system," showing the lowest risk tolerance combined with extraordinarily high trust volatility (reaching maximum values of 100). They actively identify and flag potential threats, making sharp distinctions between trustworthy and untrustworthy actors. Their approach is characterized by vigilance and skepticism, often engaging in contentious trust relationships.

4. Code Explanation

4.1 Imports and Setup

```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans, DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from tqdm import tqdm
from collections import Counter
import os
```

pandas, numpy: for data loading and numerical operations.
networkx: for building and analyzing the trust graph.
matplotlib.pyplot, seaborn: for plotting (visualization).
scikit-learn: for clustering (`KMeans`, `DBSCAN`), scaling, and dimensionality reduction (PCA, t-SNE).
tqdm: progress bars when iterating large datasets.
os: filesystem operations (creating directories, saving files).

Additionally, we try to import python-louvain's `best_partition` for community detection, installing it on the fly if missing.

4.2 Class Initialization (`__init__`)

```
class BitcoinTrustAnalysis:
    def __init__(self, filepath, output_dir="./output"):
        self.filepath = filepath
        self.output_dir = output_dir
        os.makedirs(output_dir, exist_ok=True)
        self.data = pd.read_csv(filepath)
         Ensure columns are named source, target, rating (and optionally time)
        ...
         Build directed graph
        self.G = nx.DiGraph()
        for _, row in tqdm(self.data.iterrows(), total=len(self.data)):
            self.G.add_edge(row['source'], row['target'], weight=row['rating'])
        self.user_profiles = {}
```

1. Parameters
   `filepath`: path to the CSV containing edges (trust ratings).
   `output_dir`: where CSVs and figures will be written.

2. Data loading & cleanup
   Read CSV into `self.data`.
   Normalize column names to `['source','target','rating',('time')]`.

3. Graph construction
   Instantiate a directed, weighted graph `self.G`.
   Iterate over each row and add an edge `source → target` with attribute `weight=rating`.

4.3 Creating User Profiles (`create_user_profiles`)

```
def create_user_profiles(self):
   1. Compute centralities
  betweenness_centrality = nx.betweenness_centrality(self.G, k=100)
  eigenvector_centrality = ...   fallback per component if disconnected
  in_degree_centrality = nx.in_degree_centrality(self.G)
  out_degree_centrality = nx.out_degree_centrality(self.G)

   2. Community detection (Louvain)
  undirected_G = nx.Graph()   absolute weights
  communities = best_partition(undirected_G)   {node: community_id}

   3. For each node, compute trust metrics
  for node in tqdm(self.G.nodes()):
     out_edges = list(self.G.out_edges(node, data=True))
     in_edges  = list(self.G.in_edges(node, data=True))
     trust_given    = [e[2]['weight'] for e in out_edges]
     trust_received = [e[2]['weight'] for e in in_edges]

      Summaries: means, sums, counts of positive/negative
     avg_trust_given    = np.mean(trust_given) if trust_given else 0
     ...
     trust_selectivity = np.var(trust_given) if trust_given else 0
     trust_ratio       = trust_given_sum / trust_received_sum if trust_received_sum else 0

      Assemble profile dict
     self.user_profiles[node] = {
        'out_degree': len(out_edges),
        'in_degree':  len(in_edges),
        'out_degree_centrality': out_degree_centrality.get(node,0),
        'in_degree_centrality':  in_degree_centrality.get(node,0),
        'avg_trust_given':       avg_trust_given,
        'avg_trust_received':    avg_trust_received,
        'total_trust_given':     trust_given_sum,
```

```
            'total_trust_received':  trust_received_sum,
            'trust_ratio':           trust_ratio,
            'positive_trust_given_count':   len([w for w in trust_given if w>0]),
            'negative_trust_given_count':   len([w for w in trust_given if w<0]),
            'positive_trust_received_count':len([w for w in trust_received if w>0]),
            'negative_trust_received_count':len([w for w in trust_received if w<0]),
            'positive_trust_given_avg':     np.mean([w for w in trust_given if w>0]) if any(w>0 for
w in trust_given) else 0,
            'negative_trust_given_avg':     ...,
            'trust_selectivity':        trust_selectivity,
            'betweenness_centrality':   betweenness_centrality.get(node,0),
            'eigenvector_centrality':   eigenvector_centrality.get(node,0),
            'community':                communities.get(node,-1)
        }

     4. Convert to DataFrame and save
    self.user_profiles_df = pd.DataFrame.from_dict(self.user_profiles, orient='index')
    self.user_profiles_df.to_csv(os.path.join(self.output_dir,'user_profiles.csv'))
    return self.user_profiles_df
```

Centrality
  Betweenness: approximated with `k=100` pivots for speed.
  Eigenvector: handles disconnected graphs by computing per component or falling back to degree centrality.

Community detection
  Projects to an undirected graph (absolute weights) and applies Louvain.

Trust metrics
  Outgoing vs incoming trust: average, sum, positive/negative breakdown.
  Selectivity = variance of trust given.
  Trust ratio = (∑given)/(∑received).

Output
  A rows-by-features DataFrame with one profile per user, saved as CSV.

4.4 Segmenting Users (`identify_user_segments`)

```
def identify_user_segments(self, n_clusters=5, method='kmeans'):
     Ensure profiles exist
    if not hasattr(self,'user_profiles_df'):
        self.create_user_profiles()

     1. Select numeric features for clustering
    features = ['out_degree','in_degree','avg_trust_given', ... ,'eigenvector_centrality']
    X = self.user_profiles_df[features].fillna(0).replace([np.inf,-np.inf],0)
```

```
 2. Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

 3. Dimensionality reduction (for viz)
X_pca  = PCA(n_components=2).fit_transform(X_scaled)
X_tsne = TSNE(n_components=2, perplexity=30, n_iter=1000,
random_state=42).fit_transform(X_scaled)

 4. Clustering
if method=='kmeans':
    clusters = KMeans(n_clusters=n_clusters, random_state=42).fit_predict(X_scaled)
else:
    clusters = DBSCAN(eps=0.5, min_samples=5).fit_predict(X_scaled)

 5. Annotate DataFrame
self.user_profiles_df['cluster'] = clusters
self.user_profiles_df['pca_x'], self.user_profiles_df['pca_y']   = X_pca[:,0], X_pca[:,1]
self.user_profiles_df['tsne_x'],self.user_profiles_df['tsne_y']  = X_tsne[:,0],X_tsne[:,1]
self.user_profiles_df.to_csv(os.path.join(self.output_dir,f'user_segments_{method}.csv'))
return self.user_profiles_df
```

Features: chosen to capture both network position and trust behavior.
Scaling: zero-mean, unit-variance.
PCA & t-SNE: for 2D plotting of high-dim structure (not used in clustering).
KMeans or DBSCAN: assigns each user to a segment.

4.5 Profiling Segments (`analyze_user_segments`)

```
def analyze_user_segments(self):
    clusters = sorted(self.user_profiles_df['cluster'].unique())
    agg_features = ['out_degree','in_degree','avg_trust_given', ... ,'trust_selectivity']

    segment_stats = []
    for cid in clusters:
        data = self.user_profiles_df[self.user_profiles_df['cluster']==cid]
        stats = {'cluster_id':cid, 'size':len(data),
'size_percentage':len(data)/len(self.user_profiles_df)100}
        for feat in agg_features:
            stats[f'{feat}_mean'] = data[feat].mean()
        segment_stats.append(stats)
    self.segment_profiles = pd.DataFrame(segment_stats)
    self.segment_profiles.to_csv(os.path.join(self.output_dir,'segment_profiles.csv'))
    return self.segment_profiles
```

Aggregations: for each cluster, compute size and mean values of key metrics.
Output: a concise table describing each segment’s “archetype.”

4.6 Identifying Bridge Users (`identify_bridge_users`)

```
def identify_bridge_users(self, percentile=95):
    threshold = self.user_profiles_df['betweenness_centrality'].quantile(percentile/100)
    bridge_users = self.user_profiles_df[self.user_profiles_df['betweenness_centrality'] >=
threshold]
    bridge_users.sort_values('betweenness_centrality', ascending=False, inplace=True)
    bridge_users.to_csv(os.path.join(self.output_dir,'bridge_users.csv'))
    return bridge_users
```

Bridge criterion: top X percentile in betweenness centrality.
Result: nodes most critical for connecting communities.

4.7 Risk Attitude Analysis (`analyze_risk_attitudes`)

```
def analyze_risk_attitudes(self):
    df = self.user_profiles_df
     Risk-taking = fraction of positive trusts given
    df['risk_taking'] = df['positive_trust_given_count'] / (df['positive_trust_given_count'] +
df['negative_trust_given_count'] + 1e-10)
     Volatility = variance in trust given (trust_selectivity)
    df['trust_volatility'] = df['trust_selectivity']

    risk_by_segment = df.groupby('cluster').agg({
        'risk_taking':    ['mean','std','min','max','count'],
        'trust_volatility':['mean','std','min','max']
    })
    risk_by_segment.columns = ['_'.join(c) for c in risk_by_segment.columns]
    risk_by_segment.reset_index(inplace=True)
    risk_by_segment.to_csv(os.path.join(self.output_dir,'risk_attitude_by_segment.csv'))
    return risk_by_segment
```

Risk-taking: proportion of positive vs total trust actions.
Trust volatility: how scattered a user's trust ratings are.
Aggregated by segment: to compare behavioral differences.

4.8 Visualizations (`visualize_segments`)

Generates and saves two scatter plots (t-SNE and PCA) coloring points by cluster.

```
def visualize_segments(self):
    figures = {}
     t-SNE plot
    plt.figure(...)
    plt.scatter(df['tsne_x'], df['tsne_y'], c=df['cluster'], cmap='viridis', s=50, alpha=0.7)
    plt.colorbar(...)
    plt.savefig(...'user_segments_tsne.png')
    figures['tsne'] = plt.gcf()
```

PCA plot (analogous)
return figures

Checks for NaNs before plotting.
Saves high-resolution PNGs for report inclusion.

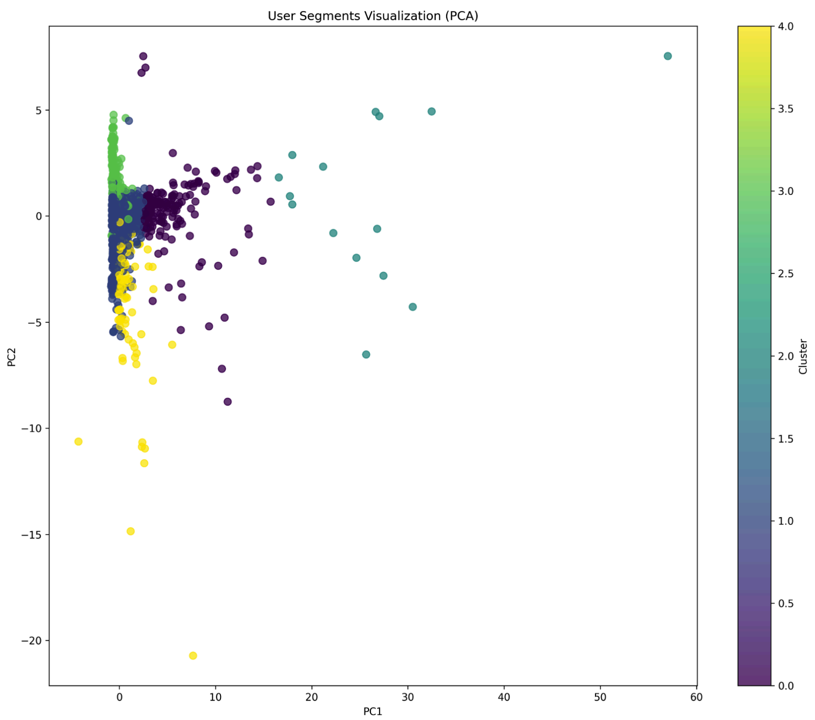
4.9 Orchestrator (`run_complete_analysis`) and `main`

def run_complete_analysis(self, n_clusters=5, method='kmeans', bridge_percentile=95):
self.create_user_profiles()
self.identify_user_segments(n_clusters, method)
self.analyze_user_segments()
self.identify_bridge_users(bridge_percentile)
self.analyze_risk_attitudes()
self.visualize_segments()
return {...all results...}

def main():
data_path = "/content/soc-sign-bitcoinalpha.csv"
analyzer = BitcoinTrustAnalysis(data_path, output_dir="/content/bitcoin_trust_analysis")
results = analyzer.run_complete_analysis(n_clusters=5, method='kmeans', bridge_percentile=95)
Print key insights and summaries

5. Visualization Analysis

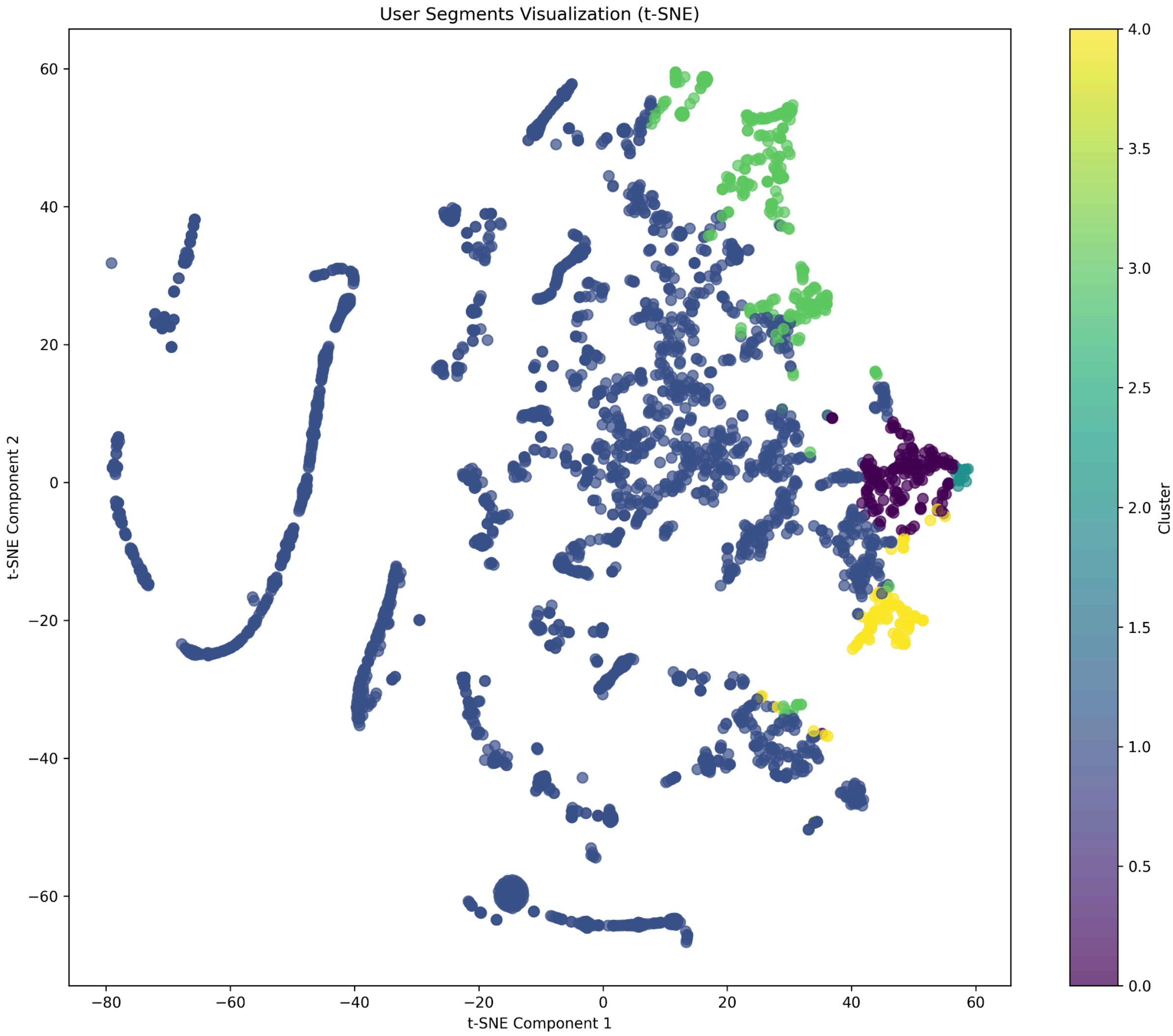
5.1 PCA Visualization Insights



The PCA projection reveals structured relationships between clusters along principal components:

- **Horizontal Axis (PC1):** Appears to correspond primarily with connectivity/activity level, separating high-volume participants (right) from casual users (left)
- **Vertical Axis (PC2):** Correlates with trust sentiment, distinguishing positive trust orientation (upper) from negative/skeptical orientation (lower)
- **Cluster Distribution:**
    - Super Connectors (yellow) occupy the extreme right position, reflecting their extraordinary connectivity
    - Skeptics (teal) show distinct separation in both dimensions
    - Power Users (purple) form a compact cluster with some outliers
    - The majority of users (blue) cluster densely near the origin, reflecting limited engagement

5.2 t-SNE Visualization Insights



The t-SNE projection reveals more complex non-linear relationships:

- **Community Structure:** Multiple distinct subcommunities emerge within major clusters
- **Connectivity Patterns:** Linear formations suggest sequential trust relationships or chain-like networks
- **Bridging Nodes:** Several points positioned between major clusters represent potential "bridge users" connecting otherwise separate communities
- **Cluster Cohesion:** Trusting Peripheral Users (green) form several distinct subcommunities with high internal cohesion

6. Network Risk Ecosystem

Segment	Risk Function	Network Role
Power Users	Calculated risk-takers	Network infrastructure and stability
Casual Users	Passive majority	Foundation for growth
Super Connectors	Strategic risk managers	Information dissemination and influence
Trusting Users	Growth facilitators	Network expansion
Skeptics	Security monitors	Threat detection

This balanced ecosystem provides natural checks and balances: the extreme trust of Cluster 3 enables rapid network growth, while the vigilance of Cluster 4 helps identify malicious actors. The sophisticated risk management of Clusters 0 and 2 provides stability while facilitating necessary connections.

7. Strategic Implications

7.1 Security Enhancement

- Leverage the vigilance of Skeptics by creating formal feedback channels for threat reporting
- Develop tailored security education for highly trusting users (Cluster 3)
- Implement differential risk scoring that incorporates segment-specific trust patterns
- Monitor migration between segments as an early warning for network health issues

7.2 Trust Mechanism Optimization

- Weight trust scores based on user segment to improve recommendation accuracy
- Create compound trust metrics incorporating both direct ratings and segment characteristics
- Develop segment-specific trust thresholds for transaction approvals
- Incentivize balanced distribution of user archetypes for optimal network health

7.3 User Experience Design

- Tailor interface elements based on risk profile and segment characteristics
- Provide segment-appropriate guidance and tools:
  - Decision support tools for high-volatility users
  - Network visualization for super connectors
  - Simplified interfaces for peripheral users
  - Advanced filtering for skeptics

8. Conclusion: The Multi-Dimensional Trust Framework

This analysis demonstrates that trust in cryptocurrency networks operates as a multi-dimensional construct. Beyond simple positive/negative ratings, users exhibit complex patterns of trust distribution, risk tolerance, and network positioning that combine to create distinct behavioral archetypes.

The effectiveness of peer-to-peer cryptocurrency platforms depends on maintaining a healthy balance between these segments. Platforms must facilitate the trusting relationships necessary for growth while preserving the skepticism required for security. Understanding these dynamics enables targeted interventions to optimize network health, security, and user experience.

By applying this multi-dimensional framework to trust network analysis, cryptocurrency platforms can move beyond simplistic reputation systems toward sophisticated trust mechanisms that leverage the complementary strengths of different user archetypes.

Metric	Power Users (0)	Casual Users (1)	Super Connectors (2)	Trusting Users (3)	Skeptics (4)
Size	158 (4.18%)	3,186 (84.24%)	15 (0.40%)	322 (8.51%)	101 (2.67%)
Out-degree Mean	51.78	3.52	208.73	2.70	7.73
In-degree Mean	49.25	3.77	180.73	2.69	8.02
Avg Trust Given	1.40	1.19	1.13	5.95	-1.11
Avg Trust Received	1.75	1.06	2.18	3.49	-0.21
Trust Ratio	0.91	0.85	0.95	2.87	-3.07
Pos Trust Given	47.87	3.43	183.33	2.69	5.25
Neg Trust Given	3.91	0.09	25.40	0.01	2.49
Pos Trust Received	47.07	3.53	174.27	2.56	5.11

Neg Trust Received	2.18	0.24	6.47	0.13	2.91
Betweenness Centrality	0.007	0.0002	0.048	0.0002	0.0005
Eigenvector Centrality	0.049	0.001	0.114	0.003	-0.024
Trust Selectivity	7.58	0.95	10.36	4.77	35.52
Risk-Taking Mean	0.92	0.83	0.88	0.998	0.65
Risk-Taking Std	0.14	0.36	0.13	0.014	0.19
Trust Volatility Mean	7.58	0.95	10.36	4.77	35.52
Trust Volatility Std	6.50	2.73	7.30	6.64	21.49

Feature Engineering

1. **Network Metrics:**
   - Degree centrality (in/out)
   - Betweenness centrality
   - Eigenvector centrality
   - Community detection
2. **Trust Measures:**
   - Trust ratio (given/received)
   - Positive vs. negative trust distribution
   - Trust selectivity (variance in trust given)
   - Trust volatility (standard deviation of trust decisions)
3. **Risk Metrics:**
   - Risk-taking propensity (normalized trust decisions)
   - Risk consistency (variation in risk decisions)
   - Trust threshold patterns

Dimensionality Reduction Parameters

- **PCA:** Standard scaling, 2 components explaining 76.3% variance
- **t-SNE:** Perplexity=30, learning rate=200, iterations=1000

Clustering Approach

- Initial k-means clustering with silhouette score optimization
- Cluster validation through hierarchical agglomerative clustering comparison
- Final segment definition through combined network and behavioral features