**BUAN 6341.001**

**Applied Machine Learning**

**PROJECT REPORT**

TRUSTGUARD ML

**Group 9:**

**Rachana Mahapatra**

**Venkata Prudhvi Marpina**

**Seelam Jyothi Praveen**

**Kalyan Teja Adapala**

**Venkata Aditya**

**Sucharitha Kakileti**

Enhancing Open Metaverse Finance with Machine Learninefinin4

APPLIED MACHINE LEARNING PROJECT REPORT

TrustGuard ML: Enhancing Open Metaverse Finance with Machine Learning for Secure and Transparent Transactions"

**INTRODUCTION: -**

The Open Metaverse, as it rapidly evolves, presents a vibrant arena for blockchain-based financial transactions, fueling the digital economy with a myriad of opportunities for trade, investment, and virtual asset exchange. This emerging network of virtual worlds relies heavily on blockchain technology to facilitate secure and transparent transactions. As this sector expands, it becomes a pivotal element in the next-generation digital economy, linking diverse virtual environments through a cohesive economic framework.

However, this sector is not without its challenges. The very nature of blockchain — decentralized and anonymous — can sometimes obscure the patterns of user behavior and open avenues for fraudulent activities. As financial interactions within the Open Metaverse grow both in complexity and volume, there is a pressing need to establish mechanisms that ensure these transactions are both transparent and trustworthy. Despite the inherent advantages of blockchain technology, such as immutability and traceability, these features alone are insufficient to fully mitigate the risks associated with such a dynamic and expansive digital environment.

Our project specifically targets these challenges by focusing on the blockchain financial transactions that underpin the economic interactions within the Open Metaverse. By harnessing advanced machine learning techniques, particularly in classification, we aim to develop systems that can not only detect and analyze patterns of normal versus anomalous transactional behaviors but also provide predictive insights that help in preempting fraudulent activities. This innovative approach leverages the data-driven power of machine learning to enhance security protocols and improve the overall reliability of blockchain transactions.

This proactive approach is essential for maintaining the integrity and reliability of financial dealings within the Open Metaverse, ultimately fostering a secure environment that can attract and sustain a wider user base. By ensuring a robust, transparent, and efficient transactional framework, our project aims to drive further growth and innovation in this virtual space, making it a cornerstone of the digital economy of the future. As the Open Metaverse continues to develop, our efforts to secure and optimize its financial transactions will play a crucial role in shaping its trajectory towards becoming a mainstream platform for global digital interaction.

**DATA OVERVIEW: -** The dataset includes 78,600 entries, each depicting a transaction within the metaverse. These entries are characterized by various attributes including the

**- Timestamp:** Date and time of the transaction.

**- Hour of Day:** Hour from the transaction timestamp.

**- Sending Address:** Sender's blockchain address.

**- Receiving Address:** Receiver's blockchain address.

**- Amount:** Simulated transaction amount.

**- Transaction Type:** Type of transaction (transfer, sale, etc.).

**- Location Region:** Simulated geographical region.

**- IP Prefix:** Simulated IP address prefix.

**- Login Frequency:** Login frequency by age group.

**- Session Duration:** Duration of user sessions in minutes.

**- Purchase Pattern**: Purchase behavior pattern**.**

**- Age Group:** User category based on activity history**.**

**- Risk Score:** Calculated risk score based on user and transaction data**.**

**- Anomaly:** Risk level (high, moderate, low).

Source: <https://www.kaggle.com/datasets/faizaniftikharjanjua/metaverse-financial-transactions-dataset/data>

**SOFTWARE & TOOLS: -**

**Programming Language:** Python

**Data Processing Libraries:** Pandas & NumPy

**Machine Learning Library:** Scikit-learn

**Visualization Tools:** Matplotlib & Seaborn

**Development Environments:** Jupyter Notebook

**OBJECTIVE: -**

This project tackles these challenges head-on by leveraging the power of machine learning, specifically classification techniques. We aim to develop a robust system that analyzes blockchain financial transactions within the Open Metaverse. This system will be equipped to:

* **Identify fraudulent transactions:** By classifying transaction patterns, the system will flag suspicious activities and anomalies, safeguarding the financial integrity of the Open Metaverse.
* **Assess transaction risk:** Through machine learning models, the system will categorize transactions based on their inherent risk profiles, enabling proactive measures and mitigating potential losses.
* **Analyze user behavior:** Classification algorithms will help us understand user behavior patterns within the Open Metaverse financial landscape. This knowledge can be used to improve user experience, personalize financial services, and enhance overall security.

**METHODOLOGY:**

Analysis

Model Building

Data

Model Evaluation

Model Training and Building

Feature Engineering

Data Preprocessing and Exploration

Conclusion

**DATA PREPROCESSING AND EDA:**

1. **Loading the file data**



1. **Checking the information, shape and columns of data set**A screenshot of a computer

   Description automatically generated
2. **Changing the timestamp column to date time format**

A screenshot of a computer

Description automatically generated

1. **Checking for duplicates and nulls and dropping them**

**A screenshot of a computer

Description automatically generated**

1. **Analysing ‘Risk-score’ vs ‘Age-group’**

**A diagram of a group of individuals

Description automatically generated with medium confidence**

1. **Analysing ‘Transaction type’ – ‘Risk Score’**

A diagram of a graph

Description automatically generated with medium confidence

1. **Analysis of variables**

A group of graphs and diagrams

Description automatically generated

1. **Analyze user behavior:**

In our endeavor to understand and enhance user interactions within the Open Metaverse's financial ecosystem, we've employed advanced machine learning techniques, notably K-means clustering, to analyze and interpret user behavior patterns. This approach is pivotal for personalizing financial services, optimizing user experience, and fortifying security measures. Here is a detailed explanation of the initial stages of our model development, focusing on data preprocessing and the selection of the optimal number of clusters for K-means analysis.

### Model Development: Part 1 - Data Preprocessing

1. **Data Preparation:**

We begin by importing essential Python libraries and modules such as Pandas for data manipulation, `StandardScaler` and `OneHotEncoder` from `sklearn.preprocessing` for feature scaling and encoding, and `ColumnTransformer` from `sklearn.compose` to apply these transformations appropriately to different data types.

2. **Feature Selection:**

We identify key numerical and categorical features that are likely to influence user behavior patterns:

- **Numerical Features:** These include 'amount', 'ip\_prefix', 'login\_frequency', 'session\_duration', and 'risk\_score'. These features provide quantitative insights into user activities and risk profiles.

- **Categorical Features:** These are 'transaction\_type', 'location\_region', and 'age\_group', which offer qualitative context about the transactions and demographic aspects of the users.

3. **Preprocessing Pipeline:**

A `ColumnTransformer` is created to apply different preprocessing strategies to the numerical and categorical data:

- **Numerical Data :** `StandardScaler` is used to normalize the numerical features, ensuring that our model isn't biased by the scale of the data.

- **Categorical Data :** `OneHotEncoder` is utilized to convert categorical variables into a form that could be provided to ML algorithms to do a better job in prediction.

This preprocessing pipeline is integral as it prepares our dataset for effective clustering, ensuring that each feature contributes equally to the analysis, thus preventing any dominance by a single feature due to its scale or variance.

### Determining the Optimal Number of Clusters

With the data appropriately preprocessed, we proceed to determine the optimal number of clusters for the K-means clustering algorithm:

1. **Elbow Method:**

- We compute the distortions — a measure of how far points within a cluster are from the centroid — for various numbers of clusters ranging from 1 to 10. This is achieved by initializing the KMeans algorithm with a range of `n\_clusters` values and fitting it to the transformed dataset.

- The distortions are calculated using the inertia of the KMeans model, which quantifies how closely related the points within a cluster are, hence serving as a measure of internal cluster coherence.

2. **Visual Analysis:**

- We plot these distortions against the number of clusters using Matplotlib to visualize the 'Elbow', a point after which the reductions in distortion diminish significantly. This point indicates the optimal number of clusters for our dataset.

- The plot helps in visually assessing the best value for `k` (number of clusters), where further increases in `k` result in diminishing returns on model improvement.

A graph with a line

Description automatically generated

In the above Elbow Method plot:

- **Distortion Decrease**: Starting from 1 cluster to 10 clusters, there is a sharp decline in distortion when moving from 1 to 2, 2 to 3, and then from 3 to 4 clusters. After 4 clusters, the rate of decline in distortion slows down significantly, suggesting diminishing returns in model improvement with additional clusters.

- **Choosing the Best `k`:** The plot shows a noticeable elbow at `k=4`, which suggests that increasing the number of clusters beyond 4 does not result in significantly better modeling of data. Therefore, `k=4` is chosen as the optimal number of clusters for our analysis.

Selecting `k=4` as the optimal number of clusters, we proceed with segmenting the Open Metaverse users into four distinct groups. This segmentation allows for more tailored analysis of user behaviors and more personalized approaches in enhancing transaction security and user engagement. By clustering users into groups based on their transactional patterns, we can identify common behaviors within each group, predict potential fraudulent activities more accurately, and improve the overall user experience by addressing the specific needs and risks associated with each cluster.

**Summary of Cluster Characteristics with `k=4`**

**Cluster 0:**

- Behavioral Traits: Engage infrequently with very low login frequencies and short session durations.

- Transaction Patterns: Mainly involved in purchases with a 'random' purchase pattern.

- Risk Profile: Exhibits the lowest risk levels, indicating cautious transaction behavior.

**Cluster 1:**

- Behavioral Traits: Highly active users with long session durations and frequent logins.

- Transaction Patterns: Primarily engaged in high-value sales.

- Risk Profile: Moderate risk scores with significant 'moderate\_risk' anomalies.

**Cluster 2:**

- Behavioral Traits: Regular users with moderate engagement.

- Transaction Patterns: Focused solely on purchases with consistent behavior.

- Risk Profile: Very low risk, making it the safest among all clusters.

**Cluster 3:**

- Behavioral Traits: Similar in engagement to Cluster 0 but involved in risky behaviors.

- Transaction Patterns: Engaged in 'phishing' and 'scam' activities.

- Risk Profile: Contains all high-risk anomalies, indicating potentially fraudulent behavior.

This refined analysis helps in deploying targeted strategies for user engagement, risk management, and security enhancements tailored to the distinct needs of each cluster, thus fostering a secure and engaging environment in the Open Metaverse. A screenshot of a graph

Description automatically generated

**Visual Insights and Strategic Implications from above heatmap in support of Cluster Analysis**

- Behavioral Patterns: The heatmap visually affirms the unique behavioral patterns and transactional preferences of each cluster, facilitating a deeper understanding of user dynamics within the Metaverse.

- Risk and Engagement Strategies: The detailed view provided by the heatmap allows for strategic planning in terms of risk management and engagement. For instance, enhanced security protocols can be designed for Clusters 1 and 3 due to their higher risk scores and intensive activities, while tailored marketing strategies could be developed for Clusters 0 and 2 to increase their platform engagement and transaction frequencies.

- Customized User Interaction: By analyzing these patterns, the Metaverse can offer more personalized user experiences, optimize user interaction based on cluster characteristics, and better manage the platform's overall risk profile.

**Cross-Tabulation Charts**

A screenshot of a graph

Description automatically generatedA table of numbers with red and blue squares

Description automatically generatedA red and blue squares with numbers

Description automatically generatedA red and blue numbers

Description automatically generated

The provided cross-tabulation charts for transaction types, location regions, anomalies, and purchase patterns offer a detailed view of how different attributes are distributed across the four clusters. Here’s a summary based on each image:

**Transaction Type Distribution**

- Phishing: Exclusively present in Cluster 3, indicating this cluster is heavily involved in fraudulent activities.

- Purchase: Predominantly in Cluster 2 (73.29%), showing that this cluster primarily focuses on purchasing activities.

- Sale: Strongly characterized by Cluster 1 (74.11%), suggesting that this cluster mainly deals with sales transactions.

- Scam: Similar to phishing, found only in Cluster 3, reinforcing the cluster's association with high-risk activities.

- \*\*Transfer\*\*: More evenly spread across Clusters 0, 1, and 2, with Cluster 1 showing slightly higher involvement (35.55%).

**Location Region Distribution**

- The distribution across regions is fairly even among Clusters 0, 1, and 2, suggesting a geographically diverse user base in these clusters. However, there are notable insights:

- Europe: Slightly more concentrated in Cluster 2.

- Cluster 3: Shows significantly less engagement across all regions, with notably low percentages indicating either a smaller size or less active participation in the Open Metaverse from this cluster.

**Anomaly Distribution**

- High Risk: Uniquely identifies Cluster 3, with 100% of its anomalies being high risk, highlighting its problematic nature.

- Low Risk: Most prominent in Cluster 2 (41.01%), aligning with its dominant purchasing activity, followed by Cluster 0.

- Moderate Risk: Almost exclusively in Cluster 1 (95.42%), suggesting a more complex transaction environment that may involve higher stakes or greater variability in transaction security.

**Purchase Pattern Distribution**

- Focused: Solely in Cluster 2 (100%), indicating a targeted approach to purchases.

- High Value : Confined to Cluster 1 (100%), aligning with its profile of handling more significant, high-stake transactions.

- Random: Predominantly in Cluster 0 (75.16%), suggesting a less predictable purchasing behavior which could involve a variety of different transaction types and purposes.

These distributions provide a comprehensive insight into the behavioral and operational dynamics of each cluster, defining their roles and risk profiles within the Open Metaverse. This understanding is crucial for tailoring specific strategies for engagement, risk management, and optimization of user interactions within the ecosystem.

1. **Identify fraudulent transactions:** By classifying transaction patterns, the system will flag suspicious activities and anomalies, safeguarding the financial integrity of the Open Metaverse.

**1. Data Overview and Preprocessing:**

* **Dataset Description**: The dataset consists of 78,600 entries, each representing a transaction in the metaverse with various attributes like transaction time, sender/receiver blockchain addresses, transaction amount, type, and others.
* **Initial Steps**:
  + **Data Loading**: DataFrame **df** is loaded with data from a CSV (**"metaverse\_transactions\_dataset.csv"**).
  + **Dropping Columns**: The columns 'ip\_prefix', 'timestamp', and 'risk\_score' are dropped. These columns are presumably removed because they might not add predictive value or could be redundant with other features in terms of the model's ability to predict anomalies.
* **Feature Encoding**:
  + **Label Encoding**: The 'anomaly' column, which contains categorical data indicating the risk level (high, moderate, low), is transformed into numerical labels using **LabelEncoder**. This step is crucial for modeling as logistic regression requires numerical input.

**2. Feature Preparation:**

* **Separating Features and Target**:
  + You isolate the target variable **y** as the 'anomaly' column.
  + The features **X** include all other columns except 'anomaly'.
* **One-Hot Encoding**:
  + Categorical features like 'transaction\_type', 'location\_region', 'purchase\_pattern', and 'age\_group' are one-hot encoded to transform them into a format that can be effectively used by the logistic regression model. One-hot encoding converts categorical data into binary (0 or 1) columns for each category, ensuring that the model treats these categories as separate entities without any ordinal relationship.

**3. Data Splitting:**

* The dataset is split into training and testing sets with 20% of the data reserved for testing. This split is essential for training the model on one subset of the data and then testing it on an independent subset to evaluate its performance.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**4. Model Training and Evaluation:**

* **Model Training**: A logistic regression model is initialized and trained on the processed training data. The model is configured to iterate up to 1000 times (**max\_iter =1000**) to converge on the optimal coefficients.
* **Prediction and Evaluation**:
  + The model makes predictions on the test set.
  + **Classification Report**: The output includes precision, recall, and F1-score for each class, providing insights into the model’s performance in identifying each category of anomalies.

A screenshot of a computer code

Description automatically generated

**Evaluation of Logistic Regression Model for Anomaly Detection in Metaverse Transactions**

**Model Performance Summary**

The logistic regression model was tasked with classifying metaverse transactions into three categories of risk: 'high\_risk', 'low\_risk', and 'moderate\_risk'. The performance of the model across these classes was evaluated using precision, recall, and F1-score metrics. Below is a summary of how the model performed for each risk category, along with overall accuracy and average metrics.

**Class Performance Details**

* **High Risk (Class 0):**
  + **Precision:** 100% - The model accurately identified all transactions it labeled as high risk.
  + **Recall:** 100% - The model successfully detected all high-risk transactions.
  + **F1-Score:** 100% - Indicates a perfect balance between precision and recall for high-risk transactions.
  + **Support:** 1,251 transactions were classified as high risk.
* **Low Risk (Class 1):**
  + **Precision:** 97% - High reliability in correctly predicting low-risk transactions.
  + **Recall:** 97% - Effectively identified 97% of all actual low-risk transactions.
  + **F1-Score:** 97% - Demonstrates a strong balance between precision and recall.
  + **Support:** 12,848 transactions were classified as low risk, making it the largest class.
* **Moderate Risk (Class 2):**
  + **Precision:** 79% - Less reliable, with about 21% of predictions being incorrect.
  + **Recall:** 80% - The model identified 80% of all true moderate-risk cases.
  + **F1-Score:** 79% - Indicates good performance, but highlights room for improvement.
  + **Support:** 1,621 transactions were classified as moderate risk.

**Overall Model Accuracy and Averages**

* **Accuracy:** The model achieved an overall accuracy of 96%, indicating high effectiveness across all predictions.
* **Macro Average:** Achieved 92% across precision, recall, and F1-score, reflecting good performance uniformly across all classes without considering class imbalance.
* **Weighted Average:** Also 96%, taking into account the prevalence of each class, confirming strong performance especially in predominant classes.

**Interpretation and Recommendations**

The model demonstrates excellent capability in identifying both high and low-risk transactions with nearly perfect metrics in these categories. However, the performance in identifying moderate-risk transactions, although good, suggests potential areas for improvement. Given the lower precision and recall in the moderate-risk category:

* **Data Enrichment:** Consider augmenting the dataset with more moderate-risk samples or additional features that could help distinguish this class more clearly.
* **Model Tuning:** Experiment with model parameters or try advanced algorithms that might capture the complexity of moderate-risk transactions better.
* **Feature Engineering:** Reassess the input features for potential enhancements or transformations to improve model sensitivity to moderate-risk characteristics.

**Conclusion**

The logistic regression model has proven to be highly effective for classifying metaverse transactions with respect to their risk levels, particularly for high and low-risk categories. With targeted improvements, particularly around the moderate-risk category, the model's utility and accuracy could be further enhanced.

**Supervised Learning Models Comparison for Fraudulent Transaction Identification:**

**1. Random Forest Classifier:**

**Model Description:**

A Random Forest Classifier is an ensemble learning method that builds multiple decision trees and aggregates their predictions to improve accuracy and generalization.

**Model Training:**

A basic Random Forest Classification model is initialized and trained on the processed training data.

**Random Forest:**

**A screenshot of a computer code

Description automatically generated**

It has returned an accuracy of 100%. As this model is without any hyperparameter tuning it is returning perfect accuracy and trying to overfit the data. To address this issue we performed a model with grid-search with hyperparameter tuning that performs hyperparameter tuning using GridSearchCV.

**GridSearchCV for Random Forest:**

A screenshot of a computer program

Description automatically generated

The GridSearchCV is used to search for the best combination of hyperparameters (n\_estimators and max\_depth) for the Random Forest model. The parameter grid specifies different values for n\_estimators (50, 100, 200) and max\_depth (3, 5, 7). The cv parameter is set to 5 for 5-fold cross-validation.

**Best Parameters and Best Score:**

After fitting the GridSearchCV object to the training data (X\_train, y\_train), the best parameters (best\_params) and best score (best\_score) are obtained. In this case, the best parameters are {'max\_depth': 7, 'n\_estimators': 200}

**Model Evaluation:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer code

Description automatically generated**

Predictions are made on the test data (X\_test), and the test accuracy is calculated using accuracy\_score. The test accuracy in this case is approximately 99.98%.

**Classification Report:**

A classification report is generated using classification\_report, providing precision, recall, F1-score, and support for each class (high\_risk, low\_risk, moderate\_risk) as well as macro and weighted averages.

transactions across different classes.

**Key components of the report:**

* **Precision:** This metric indicates the proportion of correctly predicted instances among all instances predicted as belonging to a particular class.
  + - Class 0 (high\_risk): Precision of 100% indicates that all instances predicted as high risk were indeed high risk.
    - Class 1 (low\_risk): Precision of 99% indicates that 99% of instances predicted as low risk were actually low risk.
    - Class 2 (moderate\_risk): Precision of 100% indicates that all instances predicted as moderate risk were indeed moderate risk.
* **Recall:** This metric indicates the proportion of correctly predicted instances among all instances that belong to a particular class.
  + - Class 0 (high\_risk): Recall of 100% indicates that all actual high-risk instances were correctly identified as high risk.
    - Class 1 (low\_risk): Recall of 100% indicates that all actual low-risk instances were correctly identified as low risk.
    - Class 2 (moderate\_risk): Recall of 95% indicates that 95% of actual moderate-risk instances were correctly identified as moderate risk.
* **F1-score:** This metric is the harmonic mean of precision and recall and provides a balance between the two. It is useful when classes are imbalanced. For example:
  + - Class 0 (high\_risk): F1-score of 100% indicates high precision and recall, implying a robust identification of high-risk transactions.
    - Class 1 (low\_risk): F1-score of 100% indicates high precision and recall, implying a robust identification of low-risk transactions.
    - Class 2 (moderate\_risk): F1-score of 98% indicates high precision and recall, with slightly lower recall compared to precision.

**Feature importance:**

A screenshot of a computer

Description automatically generated

Feature importance scores represent the contribution of each feature in the Random Forest model's decision-making process. Here's how you can interpret the feature's importance:

**High Importance Features:**

* Hour of Day: The hour of the day also has high importance, suggesting that certain times of the day may be more associated with fraudulent activities.
* Transaction Type: Different transaction types such as scams, sales, and transfers are also important features, indicating that the type of transaction influences the likelihood of fraud.
* Amount: The transaction amount is an essential predictor, with higher amounts potentially indicating higher risk.

**Medium Importance Features:**

* Session Duration: The duration of the session may provide additional context for identifying fraudulent behavior.
* Purchase Pattern and Login Frequency: These features capture the user's behavior patterns, which can be indicative of fraudulent activities.
* Age Group: Age groups such as veterans and new users may have varying levels of susceptibility to fraud.

**Low Importance Features:**

* Location Region: While location regions also contribute to the model's predictions, they have relatively lower importance compared to other features. This suggests that geographical location may have a lesser impact on fraud detection compared to other factors.

**Interpretation:**

Features with higher importance scores have a more significant influence on the model's predictions and should be closely examined for their contribution to identifying fraudulent transactions. Understanding the importance of each feature can help prioritize resources for fraud detection and prevention efforts. For example, features like transaction type and amount can be used to develop targeted strategies for monitoring and flagging suspicious transactions.

**2. Support Vector Machine Classifier Model:**

A screenshot of a computer program

Description automatically generated

A blue square with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

**SVM Model:**

**Accuracy:** The SVM model achieved an accuracy of approximately 83.21% on the test data. This indicates that the model correctly classified about 83.21% of the transactions as either fraudulent or non-fraudulent.

**Interpretation:** While the accuracy is decent, further analysis is needed to understand the model's performance across different classes and to identify areas for improvement.

**Linear SVM with Hyperparameter Tuning:**

**Best Parameters:** The best parameters obtained through grid search for the Linear SVM model were {'C': 0.001}, with a corresponding cross-validated accuracy score of approximately 95.45%.

**Accuracy:** After hyperparameter tuning, the Linear SVM model achieved an accuracy of approximately 95.87% on the test data.

* **Precision:** This metric indicates the proportion of correctly predicted instances among all instances predicted as belonging to a particular class.
  + - Class 0 (high\_risk): Precision of 100% indicates that all instances predicted as high risk were indeed high risk.
    - Class 1 (low\_risk): Precision of 97% indicates that 97% of instances predicted as low risk were actually low risk.
    - Class 2 (moderate\_risk): Precision of 84% indicates that only 84% of instances predicted as low risk were low risk.
    - .
* **Recall:** This metric indicates the proportion of correctly predicted instances among all instances that belong to a particular class.
  + - Class 0 (high\_risk): Recall of 100% indicates that all actual high-risk instances were correctly identified as high risk.
    - Class 1 (low\_risk): Recall of 98% indicates that most of the actual low-risk instances were correctly identified as low risk.
    - Class 2 (moderate\_risk): Recall of 74% indicates that only 74% of actual moderate-risk instances were correctly identified as moderate risk.
* **F1-score:** This metric is the harmonic mean of precision and recall and provides a balance between the two. It is useful when classes are imbalanced. For example:
  + - Class 0 (high\_risk): F1-score of 100% indicates high precision and recall, implying a robust identification of high-risk transactions.
    - Class 1 (low\_risk): F1-score of 100% indicates high precision and recall, implying a robust identification of low-risk transactions.

**Interpretation and Recommendations**

The model demonstrates excellent capability in identifying both high and low-risk transactions with nearly perfect metrics in these categories. However, the performance in identifying moderate-risk transactions, although good, suggests potential areas for improvement. Given the lower precision and recall in the moderate-risk category:

* **Data Enrichment:** Consider augmenting the dataset with more moderate-risk samples or additional features that could help distinguish this class more clearly.
* **Model Tuning:** Experiment with model parameters or try advanced algorithms that might capture the complexity of moderate-risk transactions better.
* **Feature Engineering:** Reassess the input features for potential enhancements or transformations to improve model sensitivity to moderate-risk characteristics.

**Conclusion:**

SVM models show promising results in identifying fraudulent transactions. The Linear SVM model, especially after hyperparameter tuning, demonstrates improved performance with higher accuracy and balanced metrics but it is still struggling to identify moderate-risk fraudulent transactions.

**3. Decision Tree Classifier Model:**

Decision trees are a non-parametric supervised learning method that recursively splits the dataset based on features to create a tree-like structure for classification.

**1. Initial Decision Tree Model:**

**A screenshot of a computer program

Description automatically generated**

Model Description: The initial Decision Tree Classifier achieved perfect accuracy on the test dataset.This may be sign of overfitting so hyperparameter tuning with grid search may be helpful to address this issue

**2. Hyperparameter Tuning with Grid Search:**

**A screenshot of a computer program

Description automatically generated**

**Model Description:**

Grid Search was employed to tune the hyperparameters of the Decision Tree Classifier. The best parameter configuration found was a maximum depth of 5. The final Decision Tree model trained with these parameters achieved perfect accuracy on the test dataset.

**Model Evaluation**:

The Decision Tree model achieved an accuracy of 100% on the test dataset, indicating that it can effectively classify transactions as fraudulent or legitimate. The best hyperparameter configuration obtained through GridSearchCV was a maximum depth of 5, which suggests that the model's complexity was appropriately tuned to avoid overfitting.

**Decision Tree and Feature Importance Analysis:**

A diagram of a company

Description automatically generated with medium confidence A screenshot of a computer screen

Description automatically generated

The feature importance analysis reveals that the most significant features for detecting fraudulent transactions are:

Transaction Type (e.g., purchase, sale, scam, transfer), Hour of the Day, Amount of Transaction, Age Group (e.g., new, veteran). Transaction types such as scams and sales contribute the most to the model's decision-making process, followed by features related to the hour of the day and transaction amount. These insights can guide further investigation into transaction patterns and help identify potential fraud indicators.

**Recommendations:**

Based on the evaluation results, Utilize the trained Decision Tree model with a maximum depth of 5 for real-time fraud detection in transaction data. Focus on monitoring transactions categorized as scams and sales, especially during specific hours of the day. Investigate transactions with unusual amounts or involving new or veteran users, as they may exhibit anomalous behavior indicative of fraud. In conclusion, the Decision Tree model offers high accuracy and interpretable decision-making, making it a valuable tool for detecting fraudulent transactions in the dataset.

**4. Voting Classifier:**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**