**Assignment 1**

**CreditCard Fraud Detection Using Machine Learning Algorithms**

1. **What does each row in the dataset represent?**

Ans. Each row in this dataset represents a single credit card transaction. Think of it as one specific payment event that occurred, complete with all its associated details.

1. **Which columns are clearly understandable, and which ones look encrypted or coded? Can you guess what any of the V1, V2, ..., V28 columns might represent?**

Ans. The 'Time', 'Amount', and 'Class' columns are straightforward and easy to understand. However, the 'V1' through 'V28' columns appear to be encrypted or coded. This is likely done using a technique like Principal Component Analysis (PCA) to protect sensitive user information and privacy.

1. **What is the purpose of the Class column? What do the values 0 and 1 mean, and why is this important for a fraud detection project?**

Ans. The 'Class' column is our target variable; it tells us the outcome we want to predict. A value of '0' means the transaction is legitimate (not fraud), and '1' means it is fraudulent. This column is crucial because it provides the "answer key" that our machine learning model uses to learn what fraud looks like.

1. **Look at the values in the Amount and Time columns for the first 5 rows. What kind of patterns or differences do you notice? Do any amounts look unusually high or low?**

Ans. The 'Time' values generally show an increasing trend, representing the elapsed time between transactions. The 'Amount' values can vary significantly; some might be very small, while others could be quite large, indicating a wide range of transaction sizes.

1. **Why is it helpful to look at the first few rows of a dataset before doing any analysis or modeling? What kind of problems or surprises could this help you avoid later?**

Ans. Reviewing the first few rows provides an immediate snapshot of the data's structure, column names, and data types. This initial glance can reveal issues like unexpected data formats, missing values, or incorrect column interpretations, allowing for early correction and preventing later analytical errors.

1. **How many rows (entries) and columns (features) are there in the dataset?**

Ans. The dataset contains 284,807 rows (entries), each representing a transaction, and 31 columns (features), providing various details about these transactions.

1. **Do any columns have missing values? How do you know?**

Ans. Based on the initial analysis (df.isna().sum()), no columns have missing values. We know this because the "Non-Null Count" for every column matches the total number of entries in the dataset.

1. **What type of data (number or category) is stored in most of the columns?**

Ans. Most of the columns, particularly the 'V' features, 'Time', and 'Amount', store numerical data, specifically float64 (floating-point numbers) or int64 (integers).

1. **Which column do you think represents the final outcome—whether a transaction is fraudulent or not? Why?**

Ans. The 'Class' column represents the final outcome. Its values (0 for legitimate, 1 for fraudulent) clearly indicate the category we aim to predict, making it the ideal target variable for our classification model.

1. **Why do you think it is important to check this information before building a model?**

Ans. Checking this information is vital because it ensures the data is suitable for the chosen model and task. Understanding data types, missing values, and column roles prevents errors, guides preprocessing steps, and ensures the model learns from accurate and relevant information.

1. **How many columns in your dataset are of type float64? What does this type represent?**

Ans. Based on the df.info() output, there are 5 columns of type float64. This data type represents floating-point numbers, which are numbers with decimal points, suitable for continuous or precise measurements like transaction amounts or transformed features.

1. **What is the data type of the ‘Class’ column? Why do you think it is different from the**

**other columns?**

Ans. Initially, the 'Class' column is int64 (whole numbers: 0 or 1), but after renaming, it becomes an 'object' type (string: 'Not Fraud' or 'Fraud'). It's different because it represents categorical outcomes, not continuous measurements, serving as the discrete target variable for classification.

1. **Why is it important to know the data type of each column before building a machine**

**learning model?**

Ans. Knowing data types is crucial because machine learning models often require specific input formats (e.g., numerical). Mismatched types (like text in a numerical feature) can cause errors or lead to incorrect model interpretations, impacting performance and reliability.

1. **Why did we increase the number of rows shown in the output using the display setting?How did it help in understanding the dataset?**

Ans. Increasing the displayed rows allows for a more comprehensive visual inspection of the dataset's structure and content, especially when there are many columns. It helps in quickly spotting patterns, anomalies, or unexpected data formats that might not be visible in a truncated view.

1. **Did you find any column with a data type that surprised you? If yes, which one and why? If not, explain why the types make sense.**

Ans. No, the data types generally make sense. The 'V' columns being float64 is expected given they are likely results of PCA, and 'Time' and 'Amount' are naturally numerical. The 'Class' column's type aligns with its role as a categorical target.

1. **How many total records (rows) are present in the dataset? What does each row represent?**

Ans. There are 284,807 total records, or rows, in the dataset. Each one of these rows represents a single credit card transaction that occurred.

1. **How many columns (features) are included in the dataset? What kind of information do you think these columns might contain?**

Ans. The dataset includes 31 columns, which are also known as features. These columns contain various pieces of information about each transaction, such as the Time it occurred, the Amount of the transaction, and a Class label indicating if it was fraudulent or not. The remaining columns, labeled V1 through V28, contain coded or anonymized details about the transaction.

1. **Why is it important to know the number of rows and columns before starting data analysis?**

Ans. Knowing the dataset's dimensions (rows and columns) is fundamental for planning. It influences memory requirements, computational time for analysis and modeling, and helps assess the statistical power and complexity of the data for effective decision-making.

1. **If this dataset had only 100 rows and 5 columns, how would your approach to analysis change compared to working with 284,807 rows and 31 columns?**

Ans. With a smaller dataset (100 rows, 5 columns), analysis would be faster, and complex models might overfit more easily. For 284,807 rows and 31 columns, computational efficiency, dimensionality reduction, and handling class imbalance become more critical.

1. **How can the size of a dataset help us identify potential challenges in cleaning, visualization, or building models?**

Ans. Large datasets increase the likelihood of data quality issues (e.g., more missing values, outliers) and can slow down visualizations and model training. Their size often necessitates more robust preprocessing, feature engineering, and scalable modeling techniques.

1. **What are missing values, and why is it important to check for them before starting analysis?**

Ans. Missing values are simply absent data points in a dataset. It's crucial to check for them because they can lead to biased analyses, incorrect statistical calculations, and cause many machine learning models to fail or produce unreliable results.

1. **Based on your analysis, did your dataset contain any missing values? If yes, mention the columns and how many were missing.**

Ans. No, based on the df.isna().sum() output, the dataset did not contain any missing values across any of its columns. All counts were zero.

1. **What would you do if a column had a lot of missing values? Why?**

Ans. If a column had many missing values, I would first assess their proportion. If a large percentage is missing, I might consider removing the column entirely. Otherwise, I would explore imputation strategies (e.g., mean, median, mode, or more advanced methods) to fill them, depending on the data's nature.

1. **If there were missing values, how might they affect the result of a machine learning model?**

Ans. Missing values can significantly degrade model performance. Many models cannot process missing data, leading to errors. If imputed incorrectly, they can introduce bias or noise, causing the model to learn incomplete or misleading patterns, resulting in inaccurate predictions.

1. **Why is it important to have clean and complete data in real-world projects like fraud detection?**

Ans. Clean and complete data is paramount in fraud detection because missing or erroneous details can lead to missed fraud cases (costly) or false alarms (inconvenient for customers). Reliable data ensures the model makes accurate, trustworthy predictions, which is critical for financial security.

1. **What does the mean value of the Amount column tell you about the typical transaction size?**

Ans. The mean value of the 'Amount' column indicates the average transaction size in the dataset. For instance, if the mean is low, it suggests that most transactions are relatively small, which is common in credit card datasets.

1. **Which column in the dataset shows the highest variation, and what might that tell you?**

Ans. To determine the highest variation, one would look at the std (standard deviation) values in df.describe(). A column with a significantly higher standard deviation indicates that its values are widely spread out, suggesting a broad range of data points for that feature.

1. **From the min and max values of the Amount column, what can you say about the range of spending in the dataset?** T

Ans**.** he minimum and maximum values of the 'Amount' column define the entire spectrum of transaction sizes. This range can reveal if spending is concentrated within a narrow band or if there are extremely low or high transactions, potentially indicating outliers or specific transaction types.

1. **What does the Class column's summary (mean, min, max) tell you about how common fraud is in this dataset?**

Ans. The mean of the 'Class' column (which is 0.001727) directly indicates the proportion of fraudulent transactions. Since it's very close to zero, it tells us that fraud is extremely rare, highlighting a severe class imbalance in the dataset.

1. **Are there any columns that seem unusual or surprising to you when looking at their summary statistics? Explain why.**

Ans. The 'V' columns (V1-V28) might seem unusual as their means are extremely close to zero and standard deviations are around one. This is expected because they are likely the result of PCA, which centers and scales the data, but it can be surprising if one isn't familiar with such transformations.

1. **How many total transactions in the dataset are labeled as “Fraud” and “Not Fraud”?**

Ans. There are 284,315 transactions labeled as "Not Fraud" and only 492 transactions labeled as "Fraud" in the dataset.

1. **What does the distribution of fraud vs. non-fraud transactions tell you about the data?**

Ans. The distribution reveals a severe class imbalance: legitimate transactions vastly outnumber fraudulent ones. This means the model will see far more examples of "Not Fraud," making it challenging to learn the subtle patterns of the rare "Fraud" class effectively.

1. **Why is it important to know that fraud cases are very few in number compared to normal transactions?**

Ans. It's crucial because models trained on such imbalanced data tend to be biased towards the majority class ("Not Fraud"). They might achieve high overall accuracy by simply predicting "Not Fraud" most of the time, but fail to detect actual fraud, which is the primary goal.

1. **If a model predicted all transactions as “Not Fraud”, would it still get a high accuracy? Why might that be misleading?**

Ans. Yes, such a model would achieve an accuracy of approximately 99.8% (284,315 / 284,807). This is highly misleading because while it's "correct" most of the time, it would completely miss all 492 actual fraud cases, rendering it useless for fraud detection.

1. **How would you handle the challenge of working with a dataset where fraud is rare? Can you think of one possible strategy?**

Ans. One common strategy is to use resampling techniques, such as SMOTE (Synthetic Minority Over-sampling Technique), to create synthetic examples of the minority class (fraud). This helps balance the dataset, giving the model more fraud examples to learn from.

1. **Why do we split the data into training and testing sets?**

Ans. We split the data to evaluate our model's ability to generalize to unseen data. The training set teaches the model, while the testing set acts as a final, unbiased exam to see how well it performs on new, real-world transactions.

1. **What is the purpose of using the 'Class' column as the output (target)?**

Ans. The 'Class' column serves as the output (target) because it contains the labels we want our model to predict: whether a transaction is fraudulent or not. It's the "answer" the model learns to identify.

1. **Why do we keep the input features (like Time, V1–V28, Amount) separate from the output label?**

Ans**.** We separate them because the input features are the "clues" or independent variables that the model uses to make its predictions. The output label is the "answer" or dependent variable that the model is trying to learn and predict.

1. **What is the importance of setting the 'random\_state' while splitting the data?**

Ans. Setting random\_state ensures reproducibility. It means that every time you run the code, the data will be split into training and testing sets in exactly the same way, allowing for consistent results and easier debugging or comparison.

1. **Why is it important to use the 'stratify' option when splitting the data in this case?**

Ans. The 'stratify' option is critical for imbalanced datasets like this one. It ensures that the proportion of fraud and non-fraud cases is maintained in both the training and testing sets, preventing a scenario where the test set might have too few (or no) fraud cases, leading to unreliable evaluation.

1. **Why do we need to scale the features in our dataset before training the model?**

Ans. Scaling features is important because many machine learning algorithms are sensitive to the magnitude of input values. If features have vastly different scales (e.g., 'Amount' being large and 'V' features being small), the algorithm might disproportionately weigh features with larger values, leading to suboptimal performance.

1. **What would happen if we scaled the test data separately, without using the scaler trained on the training data?**

Ans. If we scaled the test data separately, it would introduce "data leakage." The test data would be scaled based on its own mean and standard deviation, which might be different from the training data's. This would create an inconsistency, making the model's performance on the test set an unrealistic reflection of its real-world capability.

1. **After scaling, what kind of values do we expect to see in our dataset?**

Ans. After StandardScaler is applied, we expect to see values that are centered around zero, with a standard deviation of one. This means most values will fall within a small range, typically between -3 and +3, making them more uniform for the model.

1. **Why is it important to apply the same scaling method to both training and testing data?**

Ans. Applying the same scaling method ensures consistency and fairness. The model learns patterns based on the scaled training data, so the test data must be transformed using the exact same parameters (mean and standard deviation) learned from the training set to ensure a valid evaluation.

1. **Can you think of a real-life example where using values of different scales might create confusion or unfair results?**

Ans. Imagine comparing student scores where one exam is graded out of 100 points and another out of 10 points. Without converting them to a common scale (e.g., percentages), directly comparing scores like "80" and "9" would be misleading and unfair.

1. **What is the purpose of using a Logistic Regression model in this fraud detection project?**

Ans. Logistic Regression is used here to classify transactions as either fraudulent or not fraudulent. It's a fundamental model that provides a probabilistic approach to binary classification, giving a baseline understanding of separability.

1. **Why is it important to split the dataset into training and testing parts?**

Ans. It's crucial to split the dataset into training and testing parts to fairly evaluate our model. If we trained and tested the model on the exact same data, it would simply memorize the answers and wouldn't be able to predict accurately on new, unseen transactions, giving us a misleadingly high performance score.

1. **What does it mean when your model has a high accuracy score? Is accuracy always the best way to evaluate a fraud detection model? Why or why not?**

Ans. A high accuracy score means the model correctly predicted a large percentage of all transactions. However, for fraud detection, accuracy is *not* always the best metric. Because fraud cases are extremely rare, a model could achieve high accuracy by simply predicting "Not Fraud" for almost everything, while still missing critical actual fraud cases.

1. **If your model predicted “Not Fraud” for all transactions, what would the accuracy look like? Would this be a good model? Why or why not?**

Ans. If the model predicted "Not Fraud" for all transactions, its accuracy would still be very high (around 99.8% in our dataset). However, this would be a terrible model for fraud detection because it would completely miss all actual fraudulent transactions, making it useless for its intended purpose.

1. **Suppose your model's accuracy is lower than expected. List two things you might try to improve the model’s performance.**

Ans. If accuracy is low, two strategies are: 1) **Feature Engineering:** creating more informative features from existing data, and 2) **Hyperparameter Tuning:** optimizing the model's internal settings to better fit the data.

1. **What kind of data did you use to train the Gaussian Naive Bayes model? Why is this step important before making predictions?**

Ans. We used the *scaled* training data (x\_train\_sc) to train the Gaussian Naive Bayes model. This step is crucial because the model learns the statistical properties (like mean and variance) of each feature for both fraud and non-fraud classes, which are then used to make predictions on new data.

1. **In your own words, how would you explain the job of the Naive Bayes model in this fraud detection project?**

Ans. The Naive Bayes model's job is to calculate the probability of a transaction being fraud or not fraud, based on the patterns it learned from past transactions. It assumes each piece of information (feature) independently contributes to that probability.

1. **What was the accuracy of the Gaussian Naive Bayes model on your test data? Do you think this is good enough for detecting fraud? Why or why not?**

Ans. The Gaussian Naive Bayes model achieved an accuracy of 97.73%. While seemingly high, this is likely *not* good enough for fraud detection because, due to imbalance, it might still miss a significant number of actual fraud cases (high False Negatives), which are critical to catch.

1. **What does the LDA model learn from the training data, and why is this learning useful in detecting fraud?**

Ans. The LDA model learns a linear combination of features that best separates the fraud and non-fraud classes. This learning is useful because it creates a clear decision boundary, helping to distinguish between the two transaction types in a linearly optimal way.

1. **What is the accuracy of your LDA model, and what does this number tell you about its performance?**

Ans. The LDA model achieved an accuracy of 99.95%. This number suggests a very high overall correctness rate, but as discussed, for fraud detection, its effectiveness hinges more on its ability to correctly identify the rare fraud cases.

1. **Was your dataset balanced between fraud and not-fraud cases? Why is this important when training a model?**

Ans. No, our dataset was highly imbalanced, with legitimate transactions vastly outnumbering fraudulent ones (e.g., 284,315 'Not Fraud' vs. 492 'Fraud'). This imbalance is critical because models trained on such data can become biased towards the majority class, leading them to perform poorly on the rare, but important, fraud cases.

1. **Imagine you are a bank manager. Based on the accuracy score of your model, would you feel confident using it to flag suspicious transactions? Why or why not?**

Ans. As a bank manager, I would *not* feel confident relying solely on accuracy. While 99.95% accuracy sounds great, the critical concern is the number of *missed frauds* (False Negatives). I'd need to see a very high recall for fraud before deployment.

1. **If you could improve the model, what is one thing you would try next time?**

Ans. One key improvement would be to apply resampling techniques like SMOTE to address the severe class imbalance, providing the model with more examples of fraudulent transactions to learn from effectively.

1. **What is a Random Forest model, and why do we use it for fraud detection?**

Ans. A Random Forest model is like a "team" of many individual decision trees. Each tree makes a prediction, and the forest combines their votes for a final decision. We use it for fraud detection because it's robust, handles complex patterns, and often provides better accuracy and generalization than a single tree.

1. **How does the accuracy of the Random Forest model compare to the Logistic Regression and LDA models you built earlier? Which one performed the best?**

Ans. The Random Forest model achieved 99.96% accuracy, which is slightly higher than Logistic Regression (99.90%) and comparable to LDA (99.95%). In terms of overall accuracy, Random Forest performed slightly better among these.

1. **What might be the advantage of using multiple models (like Random Forest) instead of just one decision tree?**

Ans. Using multiple models (ensemble learning) like Random Forest significantly reduces the risk of overfitting, improves overall accuracy, and enhances the model's ability to generalize to new, unseen data, making it more reliable than a single, potentially biased, decision tree.

1. **If your Random Forest model made wrong predictions, what could be the possible reasons for that?**

Ans. Wrong predictions could stem from the inherent class imbalance, where fraud patterns are too rare for even the ensemble to fully grasp. It could also be due to limitations in the features provided, or the model's hyperparameters not being optimally tuned for this specific problem.

1. **Do you think Random Forest is a good model for this project? Why or why not?**

Ans. Yes, Random Forest is generally a good choice for this project due to its robustness and ability to handle complex, high-dimensional data. While its high accuracy is promising, its effectiveness for fraud detection ultimately depends on its recall for the minority class.

1. **What is a Support Vector Machine (SVM) and how does it help in detecting fraud in this project?**

Ans. A Support Vector Machine (SVM) works by finding the optimal "boundary" (hyperplane) that best separates fraudulent from legitimate transactions in a high-dimensional space. It helps detect fraud by classifying new transactions based on which side of this boundary they fall.

1. **How did the SVM model perform on the test data? What was the accuracy score?**

Ans. The SVM model performed well on the test data, achieving an accuracy score of 99.94%.

1. **What does the KNN model do when it needs to classify a new transaction?**

Ans. When classifying a new transaction, the KNN model identifies its 'K' nearest (most similar) neighbors from the training data. It then assigns the new transaction the class (fraud or not fraud) that is most common among these 'K' neighbors.

1. **What value of ‘K’ (number of neighbors) did we use in our project, and how might changing this number affect the results?**

Ans. In your code, the KNeighborsClassifier() was initialized without a specific n\_neighbors parameter, meaning it defaults to K=5. Changing this value affects the model's bias-variance tradeoff: a smaller K can be more sensitive to noise (higher variance), while a larger K can smooth out patterns (higher bias).

1. **Which model achieved the highest accuracy in your results table?**

Ans. The Random Forest model achieved the highest overall accuracy in our results, at 99.96%.

1. **What could be the reason this model performed better than the others?**

Ans. Random Forest likely performed better because it's an ensemble model, meaning it combines predictions from many individual decision trees. This approach helps reduce overfitting and improves its ability to generalize well to new data, making it robust for complex datasets.

1. **Which model performed the worst in your results? Why do you think that happened?**

Ans. The Gaussian Naive Bayes model performed the worst, with an accuracy of 97.73%. This likely happened because it makes a strong assumption that all features are independent of each other, which is rarely true in real-world data, especially in complex scenarios like fraud detection.

1. **If you had to recommend one model to a company for fraud detection, which one would you choose and why?**

Ans. I would recommend further investigation into the Random Forest model. While it showed high accuracy, the primary reason is its strong performance across other critical metrics like recall for the 'Fraud' class, which is essential for catching actual fraud. Its robustness and ability to handle complex data also make it a strong candidate.

1. **What is one thing you learned from comparing all the models together?**

Ans. The most important lesson learned is that overall accuracy alone is a misleading metric for imbalanced datasets like fraud detection. It's crucial to look at a comprehensive set of metrics, especially recall and the confusion matrix, to truly understand a model's effectiveness in identifying the rare, but critical, fraud cases.

1. **Why this step is important:**

Ans. The confusion matrix is essential because it provides a granular view of model performance, going beyond simple accuracy. It reveals precisely how many correct and incorrect classifications were made for each class, which is vital for assessing a model's trustworthiness in real-world applications like fraud detection.

1. **What do the four numbers in the confusion matrix represent in your model’s results?**

Ans. The four numbers in the confusion matrix represent:

1. **True Negatives (TN):** Legitimate transactions correctly identified as legitimate.
2. **False Positives (FP):** Legitimate transactions incorrectly flagged as fraudulent (false alarms).
3. **False Negatives (FN):** Actual fraudulent transactions that the model *missed* and classified as legitimate.
4. **True Positives (TP):** Actual fraudulent transactions correctly identified as fraudulent.
5. **Did your model correctly identify most of the fraud cases? Why is this important in real life?**

Ans. Based on the example provided (e.g., 44 missed frauds out of 100 actual frauds), our model did *not* correctly identify most fraud cases. This is critically important in real life because each missed fraud (False Negative) can lead to significant financial losses for banks and customers, directly impacting trust and security.

1. **Were there any False Positives in your model? What could be the impact of these in a real business?**

Ans. Yes, our model did produce False Positives (e.g., 11 in the Logistic Regression example). In a real business, these false alarms can lead to customer inconvenience (e.g., legitimate card declines), increased operational costs due to unnecessary investigations, and potential damage to customer relationships.

1. **What steps can you take if your model is missing too many fraud cases (False Negatives)?**

Ans. If the model is missing too many fraud cases (high False Negatives), we can take several steps: employ resampling techniques (like SMOTE) to balance the data, explore different machine learning algorithms better suited for imbalanced classes, or adjust the model's classification threshold to make it more sensitive to detecting fraud.

1. **Based on your confusion matrix, would you say your model is ready to be used in a real bank? Why or why not?**

Ans. No, based on the confusion matrix, the model is *not* yet ready for use in a real bank. Despite high overall accuracy, the significant number of missed fraudulent transactions (False Negatives) indicates that it's not robust enough to prevent critical financial losses, which is paramount for a banking system.