**Assignment for Week 5**

**Artificial Neural Network (ANN) and Support Vector Machine (SVM)**

**Objective:**

Credit Card Fraud Detection Using Neural Networks

1. **Introduction**

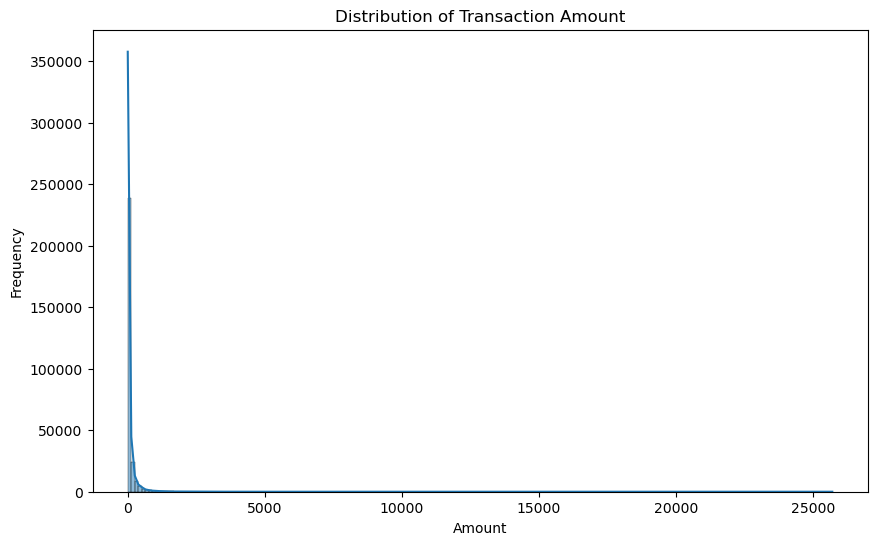
This work aims to develop a neural network model to detect fraudulent credit card transactions. Credit card fraud is a significant problem for financial institutions and consumers, resulting in billions of dollars in losses annually. Machine learning techniques like neural networks can help identify potentially fraudulent transactions in real time.

Given data has 31 attributes, where the first 30 attributes are input features and the output feature is class (A binary feature). The dataset contains numerical features V1-V28 and 'Time' and 'Amount' features. The target variable 'Class' indicates whether a transaction is fraudulent (1) or non fraudlent (0). Total No of data points: 284807, All the attributes are numerical features only

The dataset is highly imbalanced, with no fraud transactions of 284315 (99.83%) and Fraud transactions of 492 (0.172 %). Clearly data has very few minor class groups and thus the data is imbalanced.

1. **Exploratory Data Analysis**

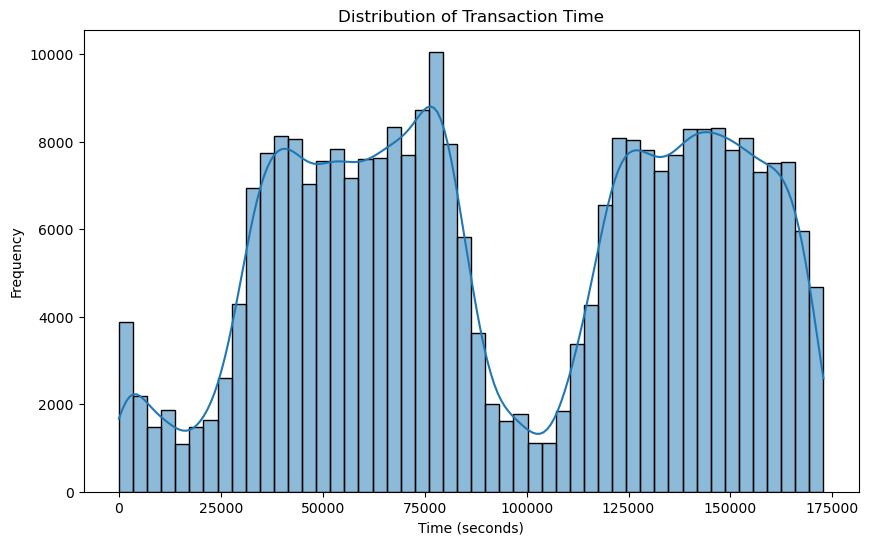
**Plot 1: Transaction Amount Distribution**

****

The histogram of transaction amounts shows:

* The majority of transactions are for relatively small amounts, with a high concentration below $100.
* There is a long tail extending to higher amounts, indicating some large transactions.
* The distribution is highly right-skewed, which is typical for financial transaction data.

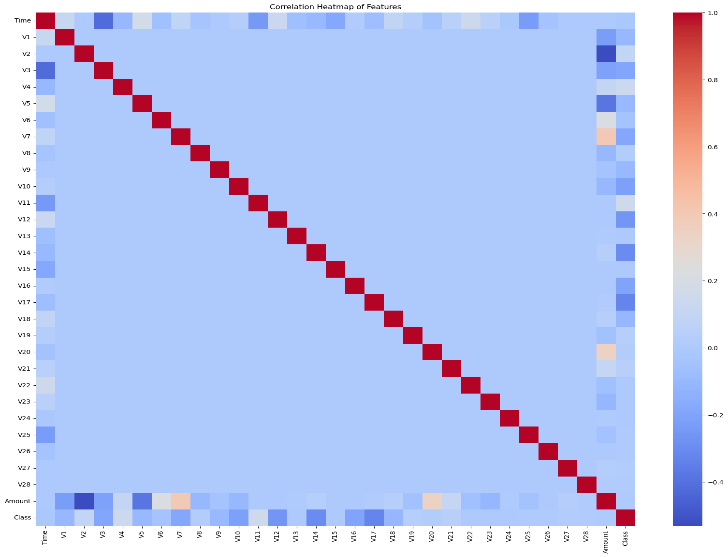
**Plot 2: Transaction Time Distribution**

****

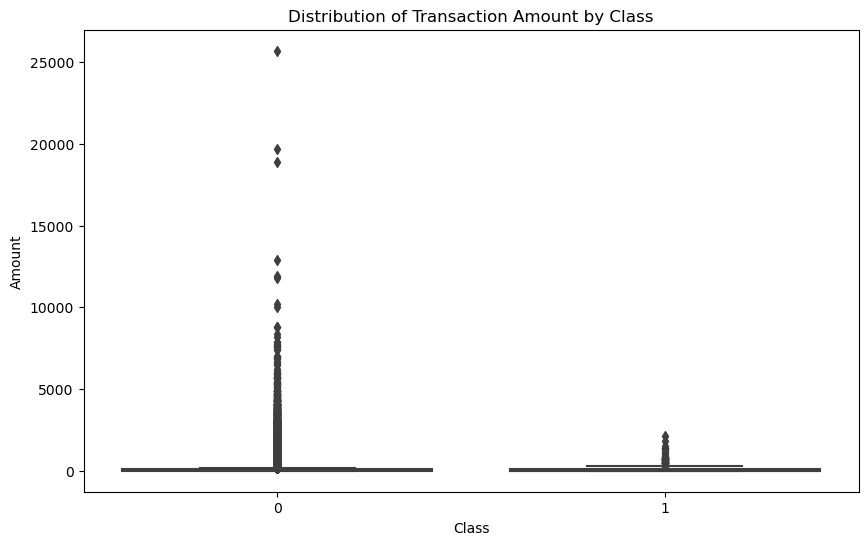
The plot of transaction times illustrates:

* Transactions occur throughout the entire time period, but with noticeable patterns.
* There are peaks and troughs, possibly corresponding to different times of day or week.
* The distribution is not uniform, suggesting temporal patterns in transaction frequency.

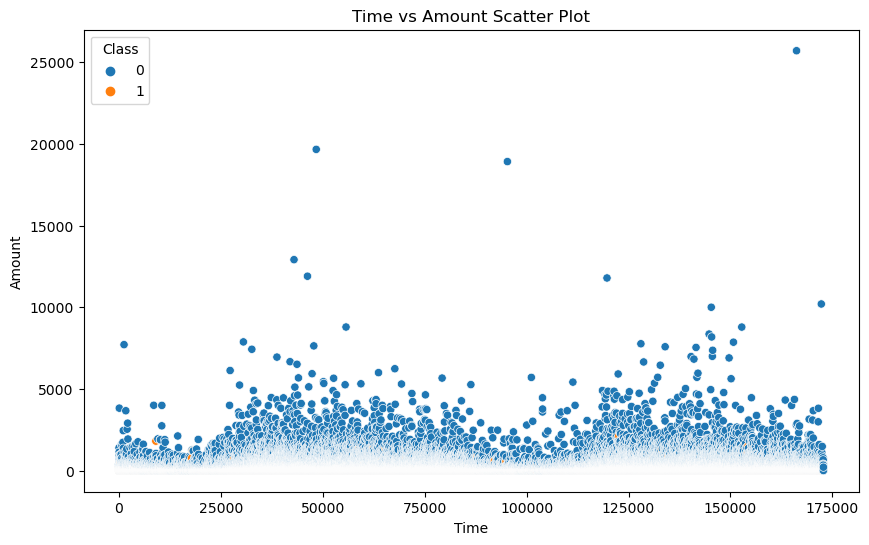
**Plot 3: Correlation Heat Map**



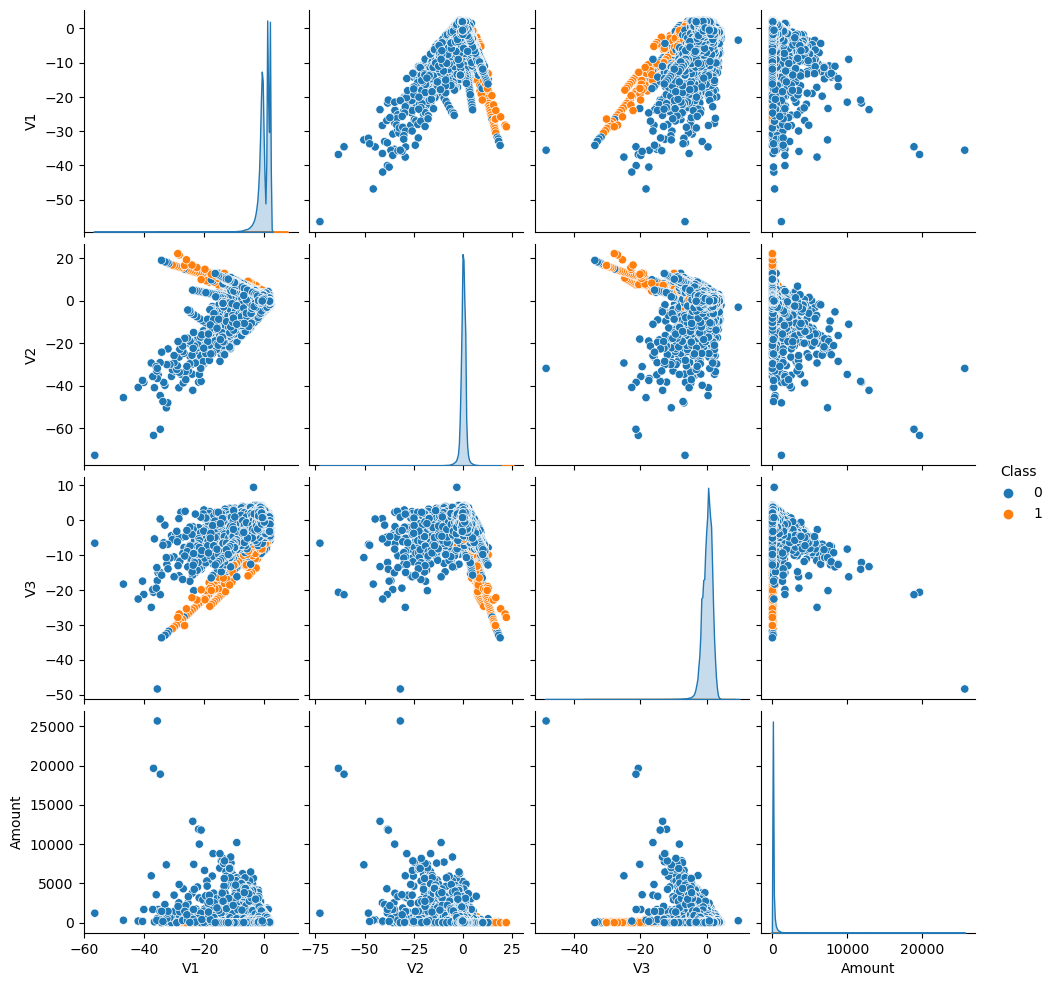
**Plot 4: Box plot for distribution of transaction amount by Class**



**Plot 5: Scatter plot of Time vs amount by Class**



**Plot 3: Pair plots of V1, V2, V3 and Amount by class**



**Interpretation of EDA plots:**

1. Transaction Amount Distribution: Shows the distribution of transaction amounts, helping identify any unusual patterns or outliers. Clearly the data is skewed alot on the right side and it has outliers
2. Transaction Time Distribution: Reveals patterns in transaction timing, which could be useful for detecting fraudulent behavior.
3. Correlation Heatmap: Highlights relationships between features, helping identify potential multicollinearity. We can see correlations mostly across time, amount and class most of the other attrbutes seem to be independent. The correlation among these is also small (max ~ 0.4)
4. Box Plot of Amount by Class: Compares transaction amounts between fraudulent and non-fraudulent transactions. non fraudlant is distributed over the very wide amounts where as most of the fraudlent transactions are less i.e., < 5000
5. Scatter Plot of Time vs Amount: Visualizes the relationship between transaction time and amount, colored by class. This plot is highlighting that we have very few fraudlant data and hence i cant segregate and identify any relation ship
6. Pair Plot: Visualizes relationships between selected features and the target variable. based on these plots we didnnot fine any simple or linear correlations of the features w.r.t class and thus need to model using complex models such as NN. (as we dont fine any simple linear relations ships)

**Neural Network Model**

* **Data Preprocessing**

Before training our neural network, we need to preprocess the data:

1. Split the data into features (X) and target (y)
2. Perform train-test split
3. Scale the features using StandardScaler

* **Model Architecture**

We'll use a simple feedforward neural network with the following architecture:

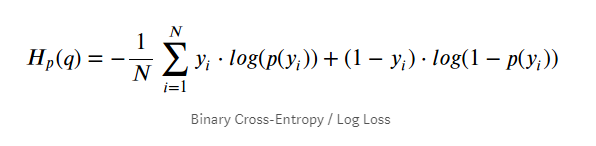
* Input layer: 30 neurons (matching our number of features)
* Hidden layer 1: 1 neurons, Sigmoid activation
* Output layer: 1 neuron, Sigmoid activation

We use binary cross-entropy as our loss function, which is appropriate for binary classification problems like fraud detection.

* **loss function for Binary classification**

Binary Cross-Entropy Loss Binary cross-entropy is specifically designed for binary classification tasks where the output is a probability between 0 and 1. It measures the performance of a classification model whose output is a probability value between 0 and 1.

The formula for binary cross-entropy loss is:



Where:

N is the number of samples

y\_i is the true label (0 or 1)

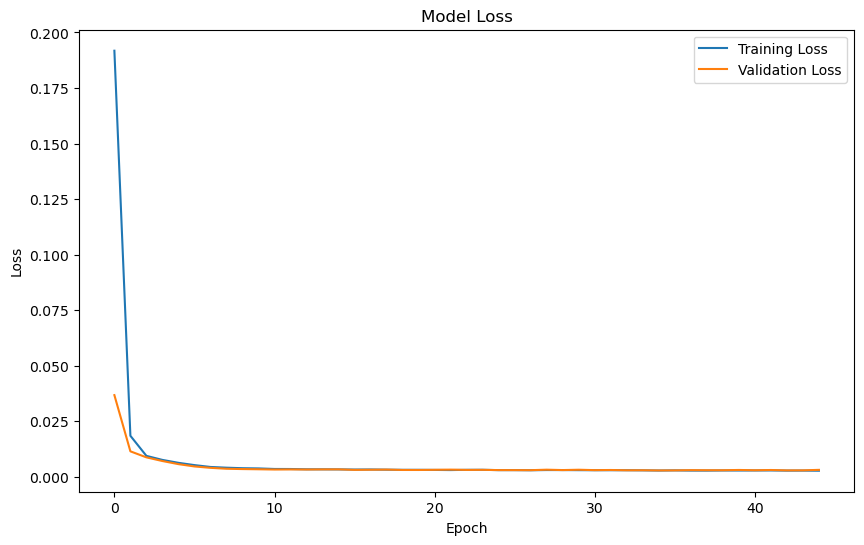
p\_i is the predicted probability of the positive class

This loss function penalizes confident and wrong predictions more than less confident ones. For example, predicting a probability of 0.9 for a negative example (actual label 0) would incur a high loss, while predicting 0.55 for the same example would incur a lower loss.

* **Model Training**

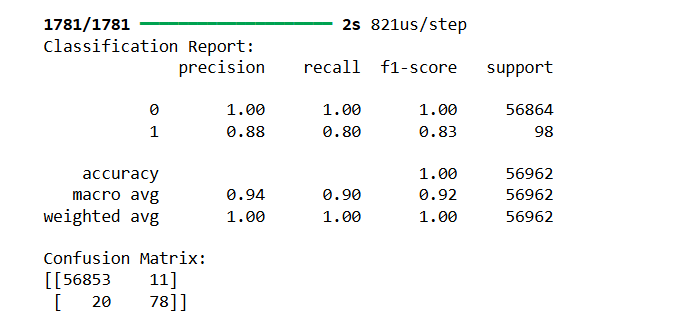
1. Training and Validation Loss

We'll train the model for 100 epochs with a batch size of 32, using 20% of the training data for validation.

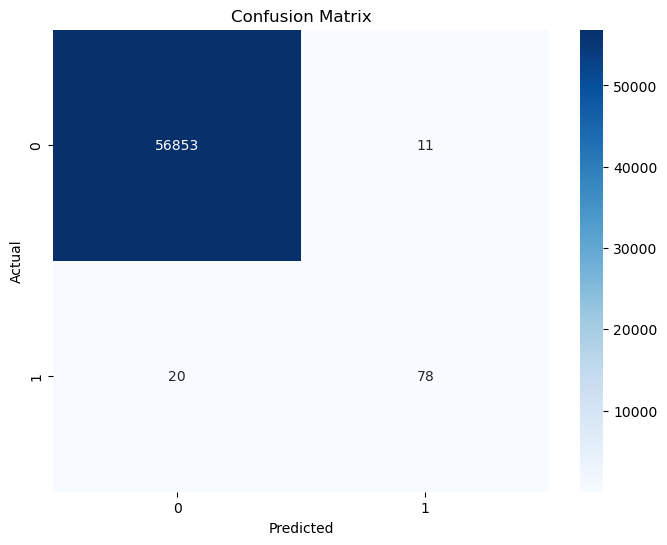


**Observations:**

* Loss Plot: The loss plot shows how the model's performance improves over time. A decreasing trend in both training and validation loss indicates the model is learning. If the validation loss starts increasing while training loss continues to decrease, it may indicate overfitting.
* Both training and validation loss decrease rapidly in the first few epochs, indicating quick initial learning.
* The validation loss plateaus after about 10-15 epochs, while training loss continues to decrease slightly.
* There's a small gap between training and validation loss, suggesting mild overfitting.
* **Model Performance:** Let's evaluate the model's performance on both the training and test



* Accuracy: Compare training and test accuracies. A significant difference might suggest overfitting. However, in fraud detection, accuracy alone can be misleading due to class imbalance.



* Confusion Matrix: For fraud detection:
* True Negatives (TN): Correctly identified non-fraudulent transactions
* False Positives (FP): Non-fraudulent transactions incorrectly flagged as fraudulent
* False Negatives (FN): Fraudulent transactions missed by the model (most critical in fraud detection)
* True Positives (TP): Correctly identified fraudulent transactions

**Additional Metrics:** Consider precision, recall, and F1-score, especially for the minority class (fraudulent transactions). Recall is particularly important in fraud detection to minimize missed fraudulent transactions.

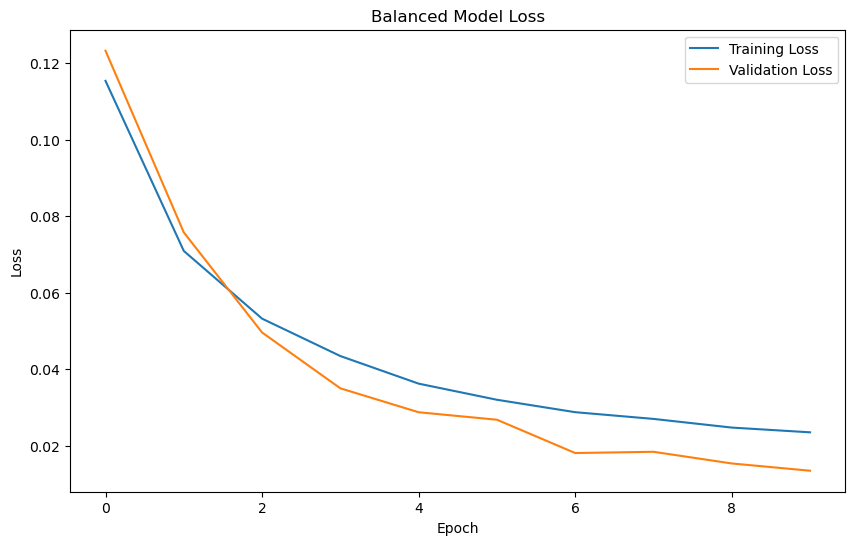
**Additional Activity:**

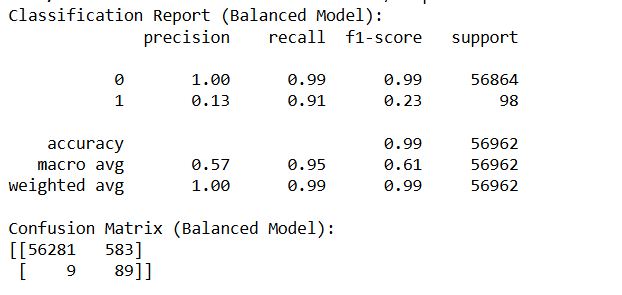
Handling Imbalanced data  
Even though I am increasing the number of nodes, there are still wrong predictions, i.e., the fraudulent calls are predicted as non-fraudulent (24 predictions).

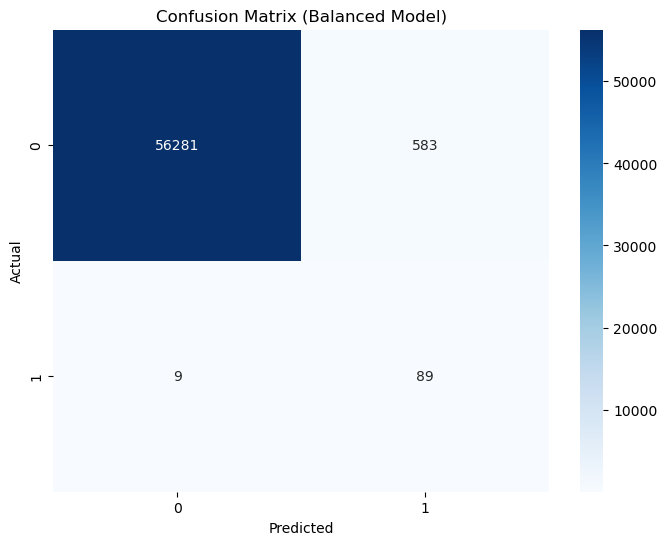
Our objective is to predict fraudulent calls. The major cause of these wrong predictions is imbalanced data. As we have >99% of non fraudlant data, the model is not balanced and gives wrong predictions for minor classes.

Thus, here we are using SMOTE (Synthetic Minority Over-sampling Technique):

Results based on SMOTE technique:







In credit card fraud, false negatives are generally considered more critical than false positives because a false negative means a fraudulent transaction is mistakenly allowed to go through, resulting in direct financial loss to the cardholder and the issuing bank, while a false positive only inconveniences a legitimate customer by temporarily blocking a transaction that is later verified as valid.

* **Conclusions**
* The neural network model performs excellently in detecting credit card fraud, with accuracy above 99.9% on both training and test sets.
* However, the high accuracy should be interpreted cautiously due to the highly imbalanced nature of the dataset. A model that always predicts "not fraud" would still achieve 99.828% accuracy.
* The model has very few false positives and false negatives, which is crucial in a real-world scenario to avoid inconveniencing customers while still catching fraudulent transactions.
* False Positives and Negatives: The confusion matrices show that the model has very few false positives (legitimate transactions classified as fraud) and false negatives (fraudulent transactions missed). This is crucial in a real-world scenario, as false positives can inconvenience customers, while false negatives can lead to financial losses.
* There is mild overfitting, but the use of early stopping helped mitigate this.
* While the model performs well on this dataset, deploying such a system in a real-world environment would require additional considerations like continuous updating, handling of real-time transactions and handling imbalanced data using suitable techniques.