1. Features Used for the ML Technique

- **Fwd Packet Length Mean**: Average length of packets in the forward direction of a network flow.
- **Avg Fwd Segment Size**: Average size of data segments in the forward direction of a network flow.
- **Init_Win_bytes_forward**: Initial window size (in bytes) advertised by the sender in the forward direction.
- **Init_Win_bytes_backward**: Initial window size (in bytes) advertised by the receiver in the backward direction.
- **min_seg_size_forward**: Minimum segment size observed in the forward direction of a network flow.

b. Were the features defined in the paper in detail? → Yes, all of them

c. Why this feature is relevant for the detection

These features are highly relevant for intrusion detection, especially for identifying Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks, as well as other anomalies:

- **Packet Lengths and Segment Sizes (`Fwd Packet Length Mean`, `Avg Fwd Segment Size`, `min_seg_size_forward`):** Anomalies in packet and segment sizes can indicate malicious activity. For example, a flood of abnormally sized packets is common in DoS/DDoS attacks aiming to overwhelm a target.
- **Initial Window Sizes (`Init_Win_bytes_forward`, `Init_Win_bytes_backward`):** The initial TCP window size reflects flow control mechanisms. Malicious actors might manipulate these values (e.g., abnormally small or large windows) to disrupt connections or optimize attack traffic, making them strong indicators of an attack.
- **Correlation Value with the target**: there is a correlation between each feature and the target, as we can see plotted in the heatmap(Go to notebook).

d. How are these features extracted?

These features were extracted **directly from the raw network flow data contained in the CSV files**, where each row represents a flow and each column corresponds to a calculated network statistic or metric (e.g., packet lengths, window sizes). Using various libraries to help us manipulate it for further use. Like Pandas, Numpy. And others.
(all mentioned in the notebook)

2. ML Technique Parameters Used

- **Random Forest Classifier:**
  - `n_estimators=50`: Number of trees in the forest(after computing accuracy for different values if n = 5, 10, 50, 100, 1000). `max_depth=5`: Maximum depth of

each tree(We'd like to keep the trees shallow to avoid overfitting; we only have 5 features).
    - `random_state=42`: For reproducibility.
- **K-Nearest Neighbors (KNN) Classifier:**
    - `n_neighbors=5`: Number of neighbors to consider(Mentioned in the paper).
- **Logistic Regression Classifier:**
    - `max_iter=10000`: Maximum iterations for convergence(A high `max_iter` guarantees the algorithm has ample opportunity to find a stable solution.).
    - `C=0.1`: Inverse of regularization strength ((smaller `C`) is preferred when there's a risk of the model learning too much from the training data's noise or when features are highly correlated.).
    - `random_state=42`: For reproducibility.
    - `solver='saga'`: Algorithm for optimization.

3. ML Technique Implementation Details

The implementation follows a standard machine learning pipeline:

1. **Data Loading and Preprocessing**: Multiple CSV files are loaded and concatenated. Column names are cleaned.
2. **Duplicate Removal**: Duplicate rows are dropped.
3. **Missing Value Handling**: Infinite values are replaced with `NaN`. Missing values in 'Flow Bytes/s' and 'Flow Packets/s' are imputed with their medians.
4. **Target Variable Transformation**: The `Label` column is mapped to broader `Attack Type` categories (e.g., 'DDoS', 'DoS', 'Brute Force', 'Bot', 'BENIGN'). This `Attack Type` is then numerically encoded into `Attack Number` using `LabelEncoder`.
5. **Data Balancing**: 'BENIGN' traffic is oversampled to match 'intrusions' to balance the classes. A subset of 30,000 samples from this balanced data (`bc_data`) is then used for model training. The `Attack Type` is converted to a binary target (0 for 'BENIGN', 1 for 'Intrusion').
6. **Feature and Target Separation**: Features (`X`) are separated from the binary target (`y`), dropping the original `Attack Type` and `Attack Number` from `X` to avoid leakage.
7. **Data Splitting**: The data is split into 80% training (`X_train`, `y_train`) and 20% testing (`X_test`, `y_test`) sets using `stratify=y` to maintain class distribution.
8. **Feature Scaling**: `StandardScaler` is fitted on `X_train` and used to transform both `X_train` and `X_test`.
9. **Dimensionality Reduction**: `IncrementalPCA` is fitted on the scaled `X_train` to reduce dimensionality by half (`n_components=size`) and then transforms both scaled training and testing data.

10. **Model Training and Prediction**: Random Forest, KNN, and Logistic Regression models are trained on the PCA-transformed training data and make predictions on the PCA-transformed testing data.
11. **Model Evaluation**: `accuracy_score`, `confusion_matrix`, and `classification_report` are printed for each model.

4. ML Technique Training and Testing Details

- **Training Data**: 80% of the balanced and sampled 30,000-instance dataset. This data is scaled using `StandardScaler` and dimensionality-reduced using `IncrementalPCA` (both fitted only on training data).
- **Testing Data**: 20% of the balanced and sampled 30,000-instance dataset. This data is transformed using the *fitted* `StandardScaler` and `IncrementalPCA` from the training phase.
- **Stratification**: `train_test_split` ensures class proportions are preserved in both sets.

5. Compare the results presented in the paper to the results you have achieved: (We'll look at accuracy only)

RF : We got ~97% an exact match (compared to the paper)!
KNN : We got a whopping 99% which is 2% higher than the paper's results.
Possible causes:we used smaller datasets where (Only CICIDS 2017), a different normalization method which is very effective in KNN. instead of Min-Max we went with StandardScaler.
Logistic regression: we got 95% which is 4% higher than the paper's results.
Possible Causes: Smaller dataset, larger max_iteration, different solver (we used 'saga')

6. Analyze the detection mistakes and try to explain the cause of these mistakes:

Looking at the Confusion matrices for the model's. We can clearly see that the chosen features though very relevant can be limited to capture the whole picture. Some information as little as it is was lost when performing PCA. as for each model this is a sign that it has hit it's limit. Especially linear regression, the relationship between the target and the features can't output better results than 95%.

7. Explain what can be done to improve the detection "accuracy":

For random forest choosing a wider feature set, will allow us to expand and enlarge the parameters without being scared of overfitting.
For KNN 99% accuracy can't be significantly enhanced
For logistic regression. We need to find the optimal feature set especially for two aspects that are the size and relevance. Finding this set that has the best relationship with our target will surely yield better accuracy results.