# COMS 4995 - Applied Machine Learning

# Project Deliverable #2 - Data Analysis and Visualization

Team 10

# INITIAL DATA EXPLORATION & INSIGHTS – DATASET SHAPE & SUMMARY

## Shape and data types of the dataset

```
[9] print("train data shape:",train_df_original.shape)
    print("test data shape:",test_df_original.shape)

train data shape: (125972, 43)
test data shape: (22543, 43)
```

```
train_df_original.dtypes
```

| | 0 |
|---|---|
| duration | int64 |
| protocol_type | object |
| service | object |
| flag | object |
| src_bytes | int64 |
| dst_bytes | int64 |
| land | int64 |
| wrong_fragment | int64 |
| urgent | int64 |
| hot | int64 |
| num_failed_logins | int64 |
| logged_in | int64 |
| num_compromised | int64 |
| root_shell | int64 |

**Train set:** 125,972 rows, 43 columns

**Test set:** 22,543 rows, same schema

No missing values detected

Feature types: mostly numerical, some categorical (protocol_type, service, flag)

Summary stats show high variance (src_bytes, dst_bytes)

Several binary features are zero-dominant

## Statistics of the dataset

```
[12] train_df_original.describe()
```

| | duration | src_bytes | dst_bytes | land | wrong_fragment | urgent | hot | num_failed_logins | logged_ |
|---|---|---|---|---|---|---|---|---|---|
| count | 125972.000000 | 1.259720e+05 | 1.259720e+05 | 125972.000000 | 125972.000000 | 125972.000000 | 125972.000000 | 125972.000000 | 125972.000 |
| mean | 287.146929 | 4.556710e+04 | 1.977927e+04 | 0.000198 | 0.022688 | 0.000111 | 0.204411 | 0.001222 | 0.395 |
| std | 2604.525522 | 5.870354e+06 | 4.021285e+06 | 0.014086 | 0.253531 | 0.014366 | 2.149977 | 0.045239 | 0.489 |
| min | 0.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000 |
| 25% | 0.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000 |
| 50% | 0.000000 | 4.400000e+01 | 0.000000e+00 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000 |
| 75% | 0.000000 | 2.760000e+02 | 5.160000e+02 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000 |
| max | 42908.000000 | 1.379964e+09 | 1.309937e+09 | 1.000000 | 3.000000 | 3.000000 | 77.000000 | 5.000000 | 1.000 |

8 rows × 39 columns

# Initial Data Exploration & Insights – Missing Values & Feature Uniqueness

**NaNs in the dataset**

```
train_df_original.isnull().sum()
```

| | 0 |
|---|---|
| duration | 0 |
| protocol_type | 0 |
| service | 0 |
| flag | 0 |
| src_bytes | 0 |
| dst_bytes | 0 |
| land | 0 |
| wrong_fragment | 0 |
| urgent | 0 |
| hot | 0 |
| num_failed_logins | 0 |
| logged_in | 0 |
| num_compromised | 0 |
| root_shell | 0 |
| su_attempted | 0 |
| num_root | 0 |
| num_file_creations | 0 |

**#Unique values in each feature in the dataset**

```
train_df_original.nunique()
```

| | 0 |
|---|---|
| duration | 2981 |
| protocol_type | 3 |
| service | 70 |
| flag | 11 |
| src_bytes | 3341 |
| dst_bytes | 9326 |
| land | 2 |
| wrong_fragment | 3 |
| urgent | 4 |
| hot | 28 |
| num_failed_logins | 6 |
| logged_in | 2 |
| num_compromised | 88 |
| root_shell | 2 |
| su_attempted | 3 |
| num_root | 82 |
| num_file_creations | 35 |

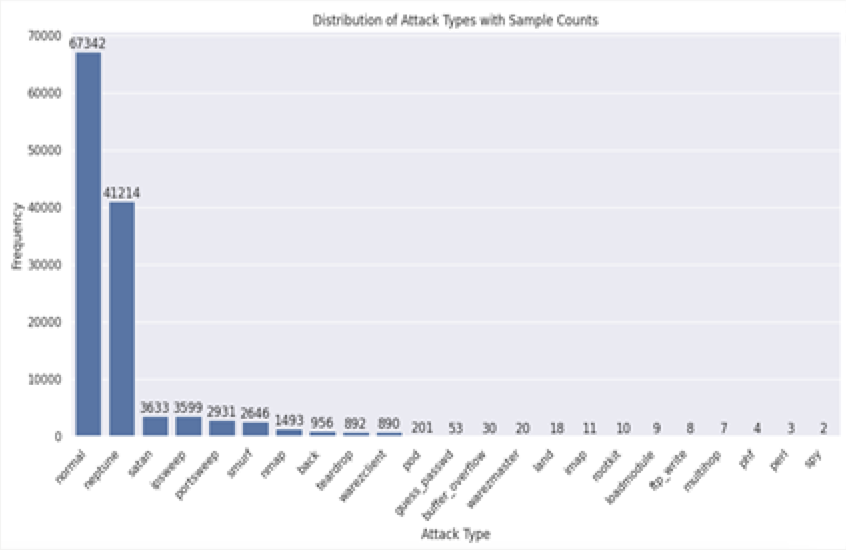
**Feature Uniqueness**

Some features have **low cardinality** ( protocol_type, flag) → Categorical

Others have **thousands of unique values** (dst_bytes, src_bytes) → Continuous

Will guide encoding strategy and feature selection

# Distribution of target feature (Attack type)



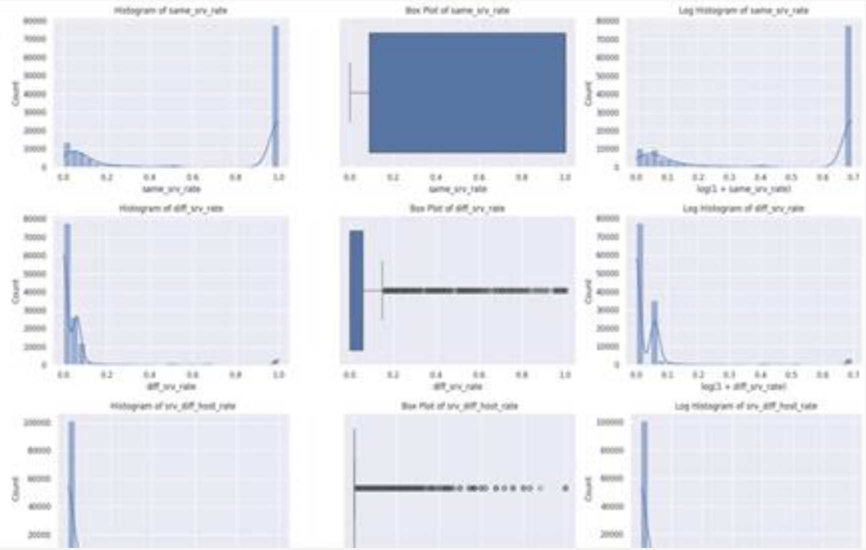

Target class (attack) is **heavily imbalanced**

**Normal** + **DoS** dominate (over 85% of samples)

Rare attacks like spy, perl severely underrepresented

Will influence sampling strategy

# Histogram, Boxplots and Log histogram for numeric features



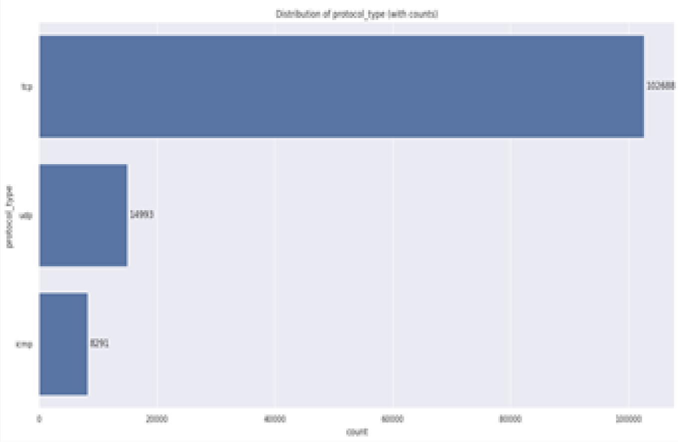

Numerical features show **skewed distributions**

Presence of outliers → visible in boxplots

Log transformation improves feature scale

Helps with modeling techniques sensitive to distribution

# Categorical Feature Distribution – protocol type, service, flag

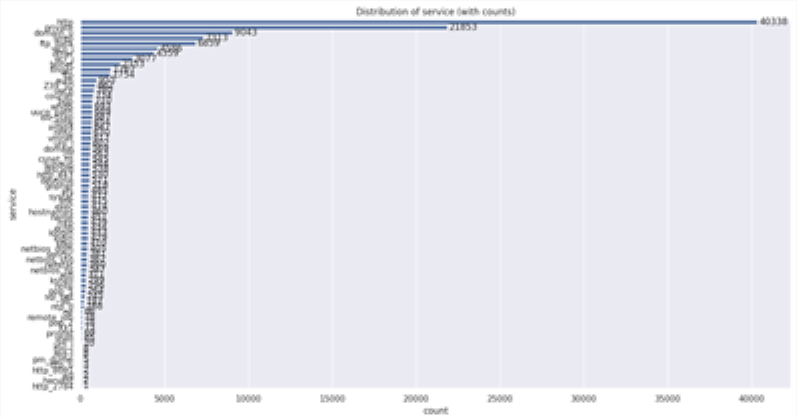



**protocol type**

Only 3 categories: tcp, udp, icmp

tcp dominates the dataset (**>80%**)

Indicates a strong protocol bias in captured traffic

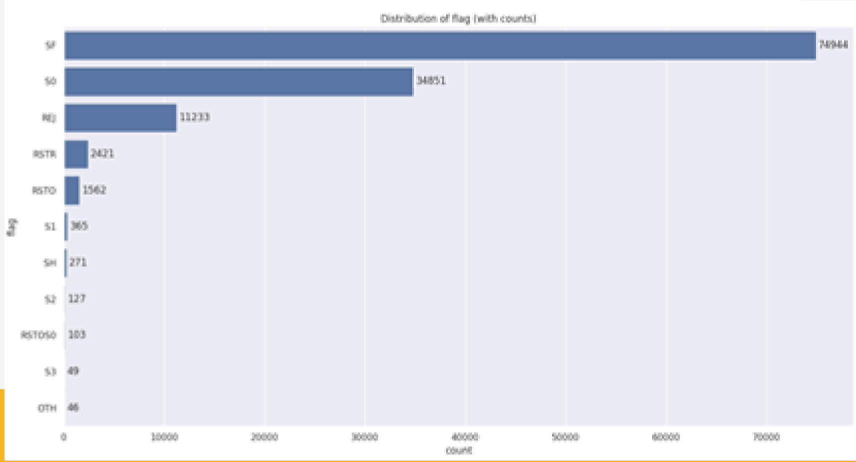



70 unique values, **highly skewed distribution**

Services like http, domain_u, privat dominate

Most services have very low frequency → **long tail risk**

May require grouping or dimensionality reduction

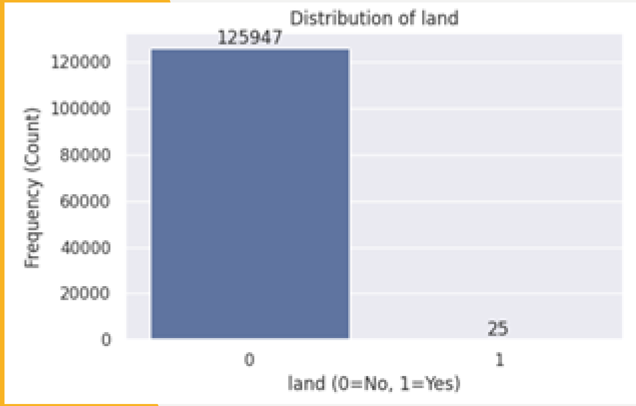



11 types of connection status flags

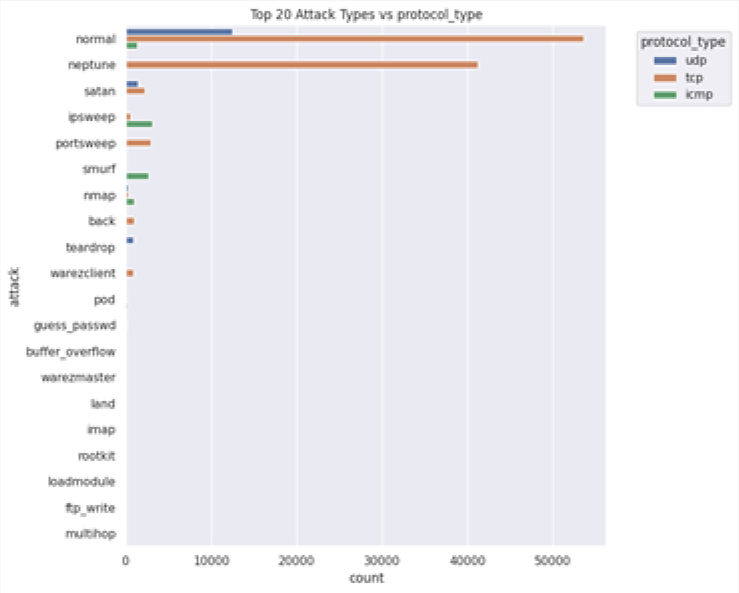
SF and S0 account for **~87%** of connections

Flags offer meaningful patterns for intrusion classification

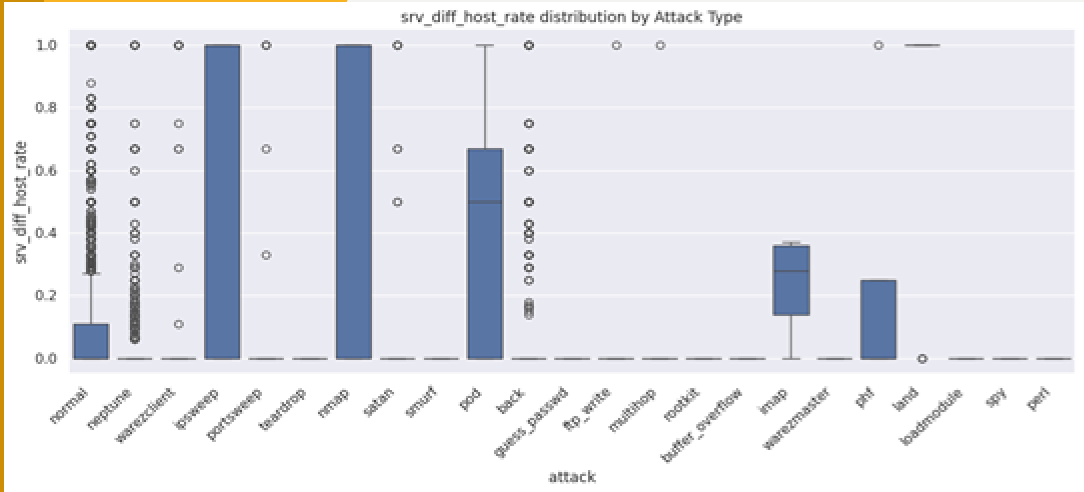
# Binary Variables Exploration




## Bivariate Analysis - Categorical features vs. target variable

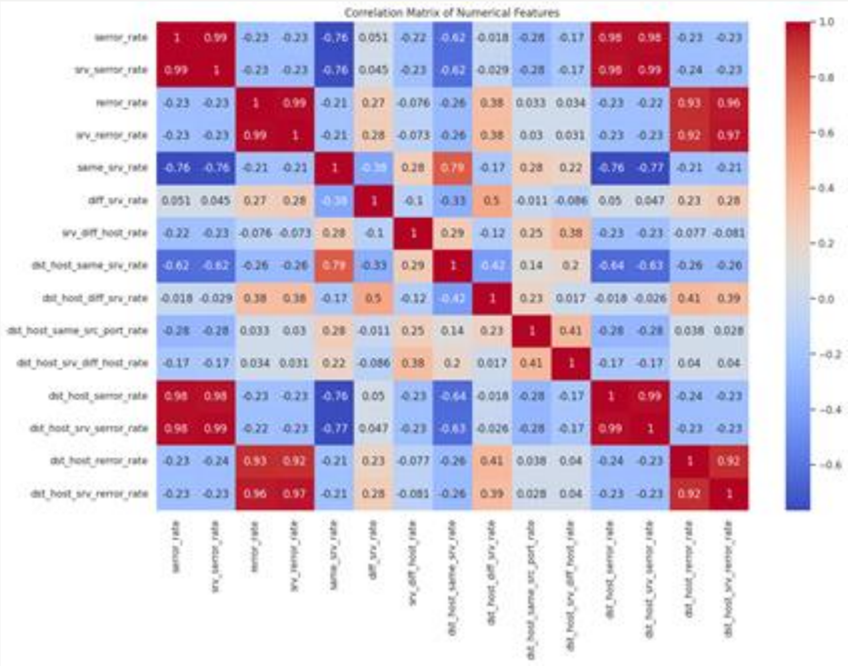



## Bivariate Analysis - Continuous features vs. target variable



Correlation matrix

# Feature Redundancy & Sparsity Analysis



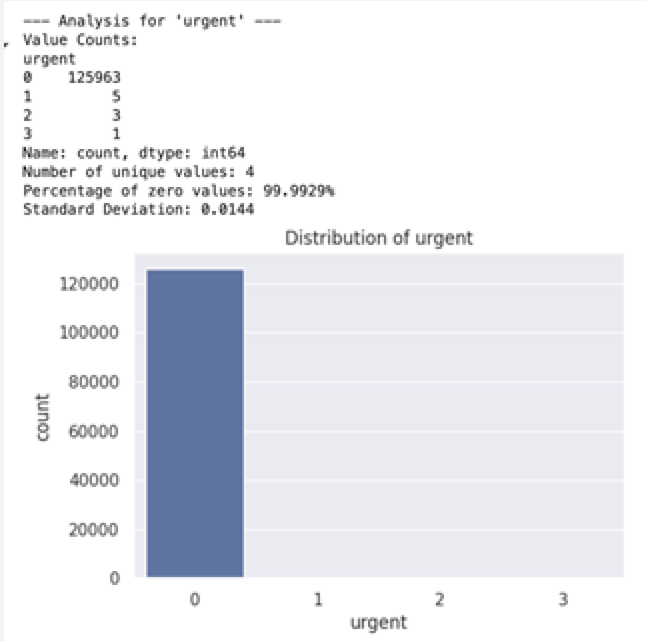

**Highly Correlated Features**

Some pairs have near-perfect correlation

Suggests possible **feature redundancy**

May apply correlation-based feature pruning or PCA

# Sparsity




**Sparse Feature Example – urgent**

99.99% of entries are **zero**

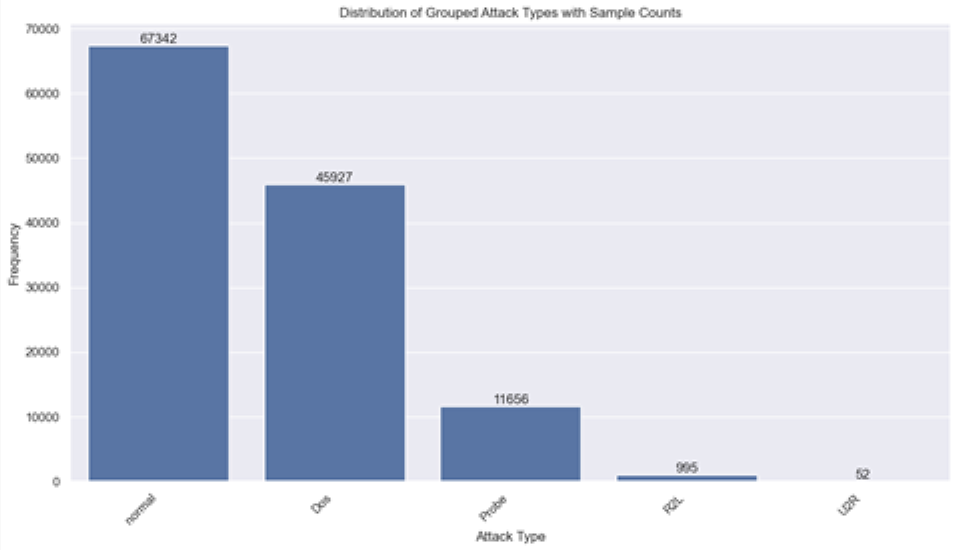
Only 4 unique values, very low variance

Could be **removed or grouped**, depending on model sensitivity

Similar checks done on other sparse/binary feature

# CLEANING & SAMPLING – ATTACK TYPE GROUPING AND LOW-VARIANCE FILTERING

Grouped attack types




Original 22 attack types were grouped into 5 categories:
Normal, DoS, Probe, R2L, U2R.
This simplifies the label space, but class imbalance remains significant.

Grouping and filtering steps improve data quality and support more efficient model training.

## Drop Variables with Low Variance

```
num_failed_logins               2.046596e-03
logged_in                       2.391315e-01
num_compromised                 5.732259e+02
root_shell                      1.339779e-03
su_attempted                    2.038935e-03
num_root                        5.953461e+02
num_file_creations              2.341950e-01
num_shells                      4.920064e-04
num_access_files                9.874387e-03
num_outbound_cmds               0.000000e+00
is_host_login                   7.938272e-06
is_guest_login                  9.334015e-03
count                           1.311227e+04
srv_count                       5.276002e+03
serror_rate                     1.993236e-01
srv_serror_rate                 1.998301e-01
rerror_rate                     1.026796e-01
srv_rerror_rate                 1.047482e-01
same_srv_rate                   1.932689e-01
diff_srv_rate                   3.251351e-02
srv_diff_host_rate              6.751235e-02
dst_host_count                  9.841943e+03
dst_host_srv_count              1.225513e+04
dst_host_same_srv_rate          2.015562e-01
dst_host_diff_srv_rate          3.569171e-02
dst_host_same_src_port_rate     9.547998e-02
dst_host_srv_diff_host_rate     1.267070e-02
dst_host_serror_rate            1.978338e-01
dst_host_srv_serror_rate        1.986219e-01
dst_host_rerror_rate            9.397818e-02
dst_host_srv_rerror_rate        1.020550e-01
level                           5.251025e+00
dtype: float64

Columns with variance < 0.001:
['land', 'urgent', 'num_shells', 'num_outbound_cmds', 'is_host_login']
```

```
Original dev data shape: (125972, 43) --> Changed to: (125972, 38)
Original test data shape: (22543, 43) --> Changed to: (22543, 38)
```

5 features with variance < 0.001 (urgent, is_host_login) were removed.
This helps eliminate uninformative variables and reduces feature count from 43 to 38.

```python
CATEGORICAL_COLS: List[str] = ['protocol_type', 'service', 'flag']
SKEWED_COLS: List[str] = ['duration', 'src_bytes', 'dst_bytes', 'count','srv_count','dst_host_srv_count']
```

## Log Transform Skewed Features

```python
for col in SKEWED_COLS:
    new_col_name = 'log_'+col
    dev_df[new_col_name] = np.log1p(dev_df[col])
    test_df[new_col_name] = np.log1p(test_df[col])

assert dev_df.shape[1] == test_df.shape[1]
```

```python
dev_df.head()
```

| ged_in | ... | dst_host_rerror_rate | dst_host_srv_rerror_rate | attack | level | log_duration | log_src_bytes | log_dst_bytes | log_count | log_srv_count | log_dst_host_srv_count |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ... | 0.0 | 0.00 | normal | 15 | 0.0 | 4.990433 | 0.000000 | 2.639057 | 0.693147 | 0.693147 |
| 0 | ... | 0.0 | 0.00 | Dos | 19 | 0.0 | 0.000000 | 0.000000 | 4.820282 | 1.945910 | 3.295837 |
| 1 | ... | 0.0 | 0.01 | normal | 21 | 0.0 | 5.451038 | 9.006264 | 1.791759 | 1.791759 | 5.545177 |
| 1 | ... | 0.0 | 0.00 | normal | 21 | 0.0 | 5.298317 | 6.042633 | 3.433987 | 3.496508 | 5.545177 |
| 0 | ... | 1.0 | 1.00 | Dos | 21 | 0.0 | 0.000000 | 0.000000 | 4.804021 | 2.995732 | 2.995732 |

# LOG TRANSFORM SKEWED FEATURES

Encode categorical Features using OHE

```python
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False, drop='first')
encoder.fit(dev_df[CATEGORICAL_COLS])
encoded_dev_array = encoder.transform(dev_df[CATEGORICAL_COLS])
encoded_test_array = encoder.transform(test_df[CATEGORICAL_COLS])
new_feature_names = encoder.get_feature_names_out(CATEGORICAL_COLS)
```

```python
encoded_train_df = pd.DataFrame(encoded_dev_array, columns=new_feature_names, index=dev_df.index)
encoded_test_df = pd.DataFrame(encoded_test_array, columns=new_feature_names, index=test_df.index)
```

```python
dev_df_processed = dev_df.drop(columns=CATEGORICAL_COLS)
test_df_processed = test_df.drop(columns=CATEGORICAL_COLS)
```

```python
dev_df = pd.concat([dev_df_processed, encoded_train_df], axis=1)
test_df = pd.concat([test_df_processed, encoded_test_df],axis=1)
```

```python
print("Original dev data shape:",dev_df_original.shape, "--> Changed to:",dev_df.shape)
print("Original test data shape:", test_df_original.shape,"--> Changed to:",test_df.shape)
```

```
Original dev data shape: (125972, 43) --> Changed to: (125972, 116)
Original test data shape: (22543, 43) --> Changed to: (22543, 116)
```

## Handle Multicollinearity with Variance Inflation Factor (VIF)

```
Calculating VIF..
exploring columns..:   0%|          | 0/107 [00:00<?, ?it/s]
ADDED
['service_ecr_i', 'num_root', 'flag_SF', 'service_http', 'srv_serror_rate', 'srv_rerror_rate', 'flag_S0', 'rerror_rate', 'serror_rate', 'dst_host_srv_serror_rate']

Calculating VIF..
exploring columns..:   0%|          | 0/106 [00:00<?, ?it/s]
ADDED
['service_ecr_i', 'num_root', 'flag_SF', 'service_http', 'srv_serror_rate', 'srv_rerror_rate', 'flag_S0', 'rerror_rate', 'serror_rate', 'dst_host_srv_serror_rate',

Calculating VIF..
exploring columns..:   0%|          | 0/105 [00:00<?, ?it/s]
ADDED
['service_ecr_i', 'num_root', 'flag_SF', 'service_http', 'srv_serror_rate', 'srv_rerror_rate', 'flag_S0', 'rerror_rate', 'serror_rate', 'dst_host_srv_serror_rate',

Calculating VIF..
exploring columns..:   0%|          | 0/104 [00:00<?, ?it/s]
ADDED
['service_ecr_i', 'num_root', 'flag_SF', 'service_http', 'srv_serror_rate', 'srv_rerror_rate', 'flag_S0', 'rerror_rate', 'serror_rate', 'dst_host_srv_serror_rate',

Calculating VIF..
exploring columns..:   0%|          | 0/103 [00:00<?, ?it/s]
ADDED
['service_ecr_i', 'num_root', 'flag_SF', 'service_http', 'srv_serror_rate', 'srv_rerror_rate', 'flag_S0', 'rerror_rate', 'serror_rate', 'dst_host_srv_serror_rate',

Calculating VIF..
exploring columns..:   0%|          | 0/102 [00:00<?, ?it/s]
ADDED
['service_ecr_i', 'num_root', 'flag_SF', 'service_http', 'srv_serror_rate', 'srv_rerror_rate', 'flag_S0', 'rerror_rate', 'serror_rate', 'dst_host_srv_serror_rate',

Calculating VIF..
exploring columns..:   0%|          | 0/101 [00:00<?, ?it/s]
```

From VIF calculation, we removed 15 variables - 'service_ecr_i', 'num_root', 'flag_SF', 'service_http', 'srv_serror_rate', 'srv_rerror_rate', 'flag_S0', 'rerror_rate', 'serror_rate', 'dst_host_srv_serror_rate', 'protocol_type_tcp', 'dst_host_srv_rerror_rate', 'same_srv_rate', 'dst_host_same_srv_rate', 'dst_host_serror_rate'

# Handling Class Imbalance with SMOTE

SMOTE transformed our training data into a balanced and diverse sample set, supporting fairer and more robust classification performance.

To address the severe class imbalance in the training dataset, we applied SMOTE (Synthetic Minority Oversampling Technique). Before resampling, the training set had 100,777 instances with highly skewed class distribution. After applying SMOTE, the dataset was expanded to 269,365 samples, ensuring that each of the five classes — Normal, DoS, Probe, R2L, and U2R — contained exactly 53,873 instances.

This synthetic balancing technique helps prevent model bias toward majority classes, particularly important for underrepresented attacks like U2R and R2L. It also provides the model with enough minority samples to learn meaningful patterns.

```
sampler = SMOTE(random_state=RANDOM_STATE)
X_resampled, y_resampled = sampler.fit_resample(X_train, y_train)
```

```
print("Distribution of train data before Random Oversampling:")
print(X_train.shape)
print("DISTRIBUTION OF TRAIN DATA AFTER RANDOM OVERSAMPLING")
print(X_resampled.shape)
```

```
Distribution of train data before Random Oversampling:
(100777, 100)
DISTRIBUTION OF TRAIN DATA AFTER RANDOM OVERSAMPLING
(269365, 100)
```

```
numerical_cols_for_scaling = [col for col in X.columns if len(X[col].unique()) > 2]
scaler = StandardScaler()
scaler.fit(X_train[numerical_cols_for_scaling])
X_train_num_scaled = scaler.transform(X_train[numerical_cols_for_scaling])
X_val_num_scaled = scaler.transform(X_val[numerical_cols_for_scaling])
X_test_num_scaled = scaler.transform(X_test[numerical_cols_for_scaling])
X_train_num_scaled_df = pd.DataFrame(X_train_num_scaled, index=X_train.index, columns=numerical_cols_for_scaling)
X_val_num_scaled_df = pd.DataFrame(X_val_num_scaled, index=X_val.index, columns=numerical_cols_for_scaling)
X_test_num_scaled_df = pd.DataFrame(X_test_num_scaled, index=X_test.index, columns=numerical_cols_for_scaling)
```

```
X_train[numerical_cols_for_scaling] = X_train_num_scaled_df
X_val[numerical_cols_for_scaling] = X_val_num_scaled_df
X_test[numerical_cols_for_scaling] = X_test_num_scaled_df
```

```
print("Shape of train df:",X_train.shape)
assert X_train.shape[0] == y_train.shape[0]
print("Shape of validation df:",X_val.shape)
assert X_val.shape[0] == y_val.shape[0]
print("Shape of test df:",X_test.shape)
assert X_test.shape[0] == y_test.shape[0]
```

# PROPOSED MACHINE LEARNING TECHNIQUES

- Tree-based models (Random Forest, XGBoost, LightGBM, Explainable boosting machine)
- Support Vector Machines (SVM) with kernels
- Deep Learning-based models

**[Hyperparameter Candidates to Search for Each ML Model]**

**We plan on using K-fold cross validation and Grid search algorithm to find optimal ML models**

```
# Random Forest Model
param_dist_rf = {
    'n_estimators': randint(100, 500),
    'max_depth': [10, 20, 30, None],
    'min_samples_split': randint(2, 11),
    'min_samples_leaf': randint(1, 11),
    'max_features': ['sqrt', 'log2', None]
}
```

```
#XGBoost Classifier
param_dist_xgb = {
    'n_estimators': randint(100, 500),
    'learning_rate': loguniform(0.01, 0.3),
    'max_depth': randint(3, 10),
    'subsample': uniform(0.6, 0.4),
    'colsample_bytree': uniform(0.6, 0.4),
    'gamma': [0, 1, 5]
}
```

```
#Light GBM
param_dist_lgbm = {
    'n_estimators': randint(100, 500),
    'learning_rate': loguniform(0.01, 0.3),
    'num_leaves': randint(20, 60),
    'max_depth': [-1, 10, 20, 30],
    'subsample': uniform(0.6, 0.4),
    'colsample_bytree': uniform(0.6, 0.4),
    'reg_alpha': loguniform(1e-3, 1.0),
    'reg_lambda': loguniform(1e-3, 1.0)
}
```

```
#Explainable Boosting Machine
param_dist_ebm = {
    'learning_rate': loguniform(0.01, 0.2),
    'max_leaves': randint(2, 10),
}
```

```
#SVM with RBF Kernel
param_dist_svm = {
    'C': loguniform(0.1, 100),
    'gamma': loguniform(1e-4, 1e-1)
}
```