Intrusion Detection with the UNSW-NB15 Dataset

```python
# --- 1. Import libraries ---
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# --- 2. Load the dataset (training + testing sets) ---
train = pd.read_csv("UNSW_NB15_training-set.csv")
test  = pd.read_csv("UNSW_NB15_testing-set.csv")

# Combine both datasets
df = pd.concat([train, test], ignore_index=True)
print("Dataset loaded successfully!")
print("Shape of combined data:", df.shape)
print()
```

```
Dataset loaded successfully!
Shape of combined data: (257673, 45)
```

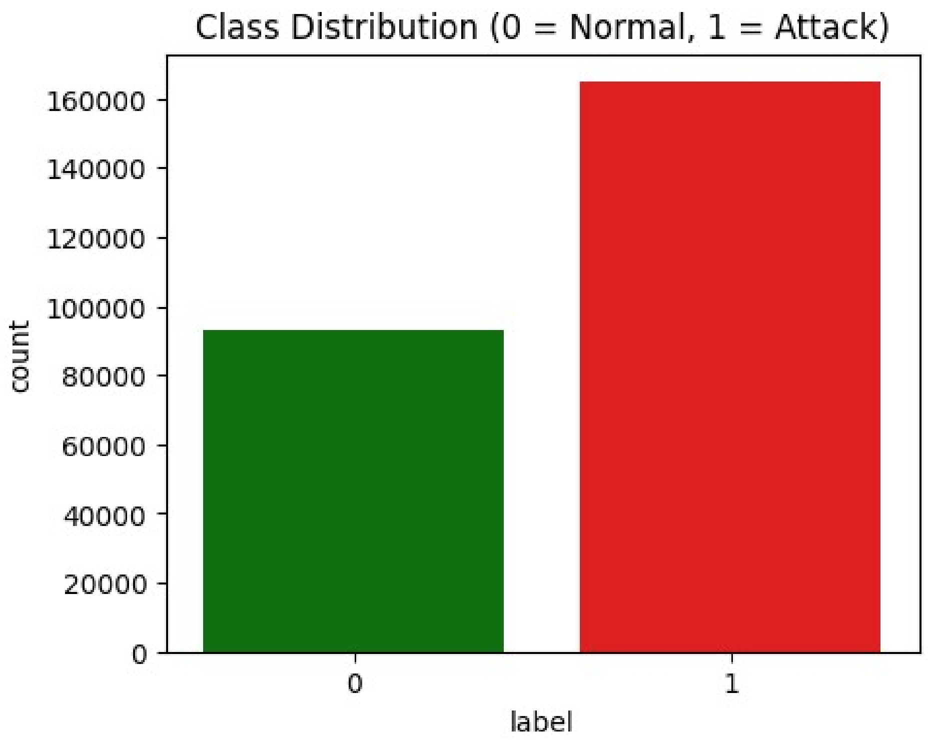
```python
# --- 3. Basic descriptive analysis ---
print("=== Dataset Information ===")
df.info()
print("\n=== Missing Values ===")
print(df.isna().sum().sum(), "missing values in total")

print("\n=== Class Distribution (label) ===")
print(df['label'].value_counts())

print("\n=== Attack Categories ===")
print(df['attack_cat'].value_counts())
```

```
=== Dataset Information ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 257673 entries, 0 to 257672
Data columns (total 45 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   id                 257673 non-null  int64  
 1   dur                257673 non-null  float64
 2   proto              257673 non-null  object 
 3   service            257673 non-null  object 
 4   state              257673 non-null  object 
 5   spkts              257673 non-null  int64
 6   dpkts              257673 non-null  int64
 7   sbytes             257673 non-null  int64
 8   dbytes             257673 non-null  int64
 9   rate               257673 non-null  float64
 10  sttl               257673 non-null  int64
 11  dttl               257673 non-null  int64
 12  sload              257673 non-null  float64
 13  dload              257673 non-null  float64
 14  sloss              257673 non-null  int64
 15  dloss              257673 non-null  int64
 16  sinpkt             257673 non-null  float64
 17  dinpkt             257673 non-null  float64
 18  sjit               257673 non-null  float64
 19  djit               257673 non-null  float64
 20  swin               257673 non-null  int64
 21  stcpb              257673 non-null  int64
 22  dtcpb              257673 non-null  int64
 23  dwin               257673 non-null  int64
 24  tcprtt             257673 non-null  float64
 25  synack             257673 non-null  float64
 26  ackdat             257673 non-null  float64
 27  smean              257673 non-null  int64
 28  dmean              257673 non-null  int64
 29  trans_depth        257673 non-null  int64
 30  response_body_len  257673 non-null  int64
 31  ct_srv_src         257673 non-null  int64
 32  ct_state_ttl       257673 non-null  int64
 33  ct_dst_ltm         257673 non-null  int64
 34  ct_src_dport_ltm   257673 non-null  int64
 35  ct_dst_sport_ltm   257673 non-null  int64
 36  ct_dst_src_ltm     257673 non-null  int64
 37  is_ftp_login       257673 non-null  int64
 38  ct_ftp_cmd         257673 non-null  int64
 39  ct_flw_http_mthd   257673 non-null  int64
 40  ct_src_ltm         257673 non-null  int64
 41  ct_srv_dst         257673 non-null  int64
 42  is_sm_ips_ports    257673 non-null  int64
 43  attack_cat         257673 non-null  object
 44  label              257673 non-null  int64
dtypes: float64(11), int64(30), object(4)
memory usage: 88.5+ MB

=== Missing Values ===
0 missing values in total

=== Class Distribution (label) ===
label
1    164673
```

```
0     93000
Name: count, dtype: int64

=== Attack Categories ===
attack_cat
Normal            93000
Generic           58871
Exploits          44525
Fuzzers           24246
DoS               16353
Reconnaissance    13987
Analysis           2677
Backdoor           2329
Shellcode          1511
Worms               174
Name: count, dtype: int64
```

```
# --- 4. Visualize the class distribution ---
plt.figure(figsize=(5,4))
sns.countplot(x='label', data=df, palette=['green','red'])
plt.title("Class Distribution (0 = Normal, 1 = Attack)")
plt.show()
```

```
C:\Users\raouf\AppData\Local\Temp\ipykernel_22016\2417132585.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(x='label', data=df, palette=['green','red'])
```

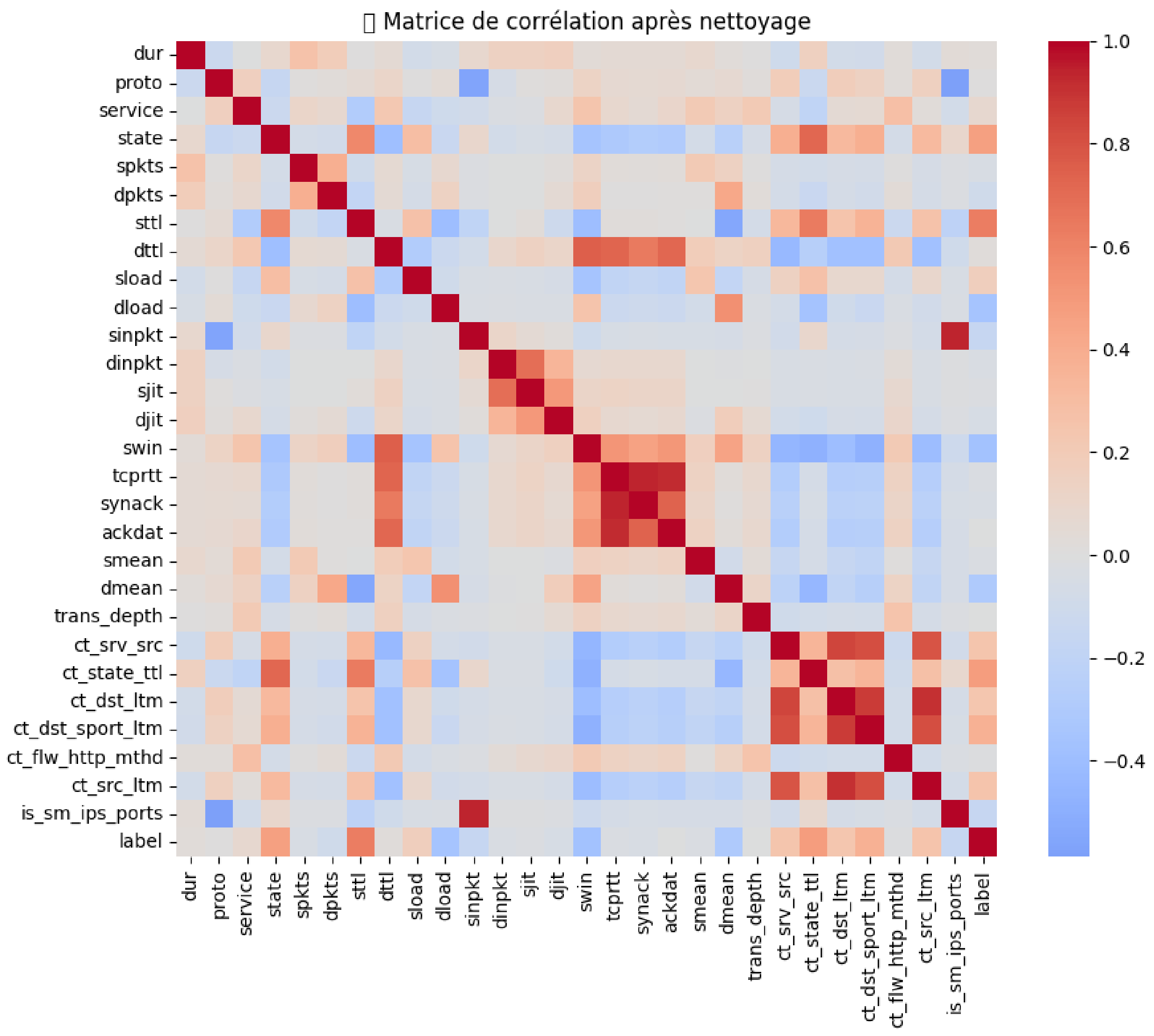


```
# --- 5. Data cleaning & preprocessing ---
df_clean = df.copy()

# Encode categorical variables
enc = LabelEncoder()
for col in ['proto', 'service', 'state', 'attack_cat']:
    df_clean[col] = enc.fit_transform(df_clean[col])

# Drop useless identifier
cols_to_drop = [
    'id',                   # identifiant inutile
    'attack_cat',           # fuite de label si on prédit 'label'
    'stcpb', 'dtcpb',       # numéros de séquence TCP
    'response_body_len',    # souvent constant à 0
    'ct_ftp_cmd',           # très rarement non nul
    'is_ftp_login',         # très souvent 0
    'rate'                  # corrélé à sbytes/dbytes
]
df_clean = df_clean.drop(columns=cols_to_drop, errors='ignore')
print("Dimensions après suppression :", df_clean.shape)

print("\nCategorical columns encoded successfully!")
df_clean.info()
```

```
Dimensions après suppression : (257673, 37)

Categorical columns encoded successfully!
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 257673 entries, 0 to 257672
Data columns (total 37 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   dur                257673 non-null  float64
 1   proto              257673 non-null  int64  
 2   service            257673 non-null  int64  
 3   state              257673 non-null  int64  
 4   spkts              257673 non-null  int64  
 5   dpkts              257673 non-null  int64  
 6   sbytes             257673 non-null  int64  
 7   dbytes             257673 non-null  int64  
 8   sttl               257673 non-null  int64  
 9   dttl               257673 non-null  int64  
 10  sload              257673 non-null  float64
 11  dload              257673 non-null  float64
 12  sloss              257673 non-null  int64  
 13  dloss              257673 non-null  int64  
 14  sinpkt             257673 non-null  float64
 15  dinpkt             257673 non-null  float64
 16  sjit               257673 non-null  float64
 17  djit               257673 non-null  float64
 18  swin               257673 non-null  int64  
 19  dwin               257673 non-null  int64  
 20  tcprtt             257673 non-null  float64
 21  synack             257673 non-null  float64
 22  ackdat             257673 non-null  float64
 23  smean              257673 non-null  int64  
 24  dmean              257673 non-null  int64  
 25  trans_depth        257673 non-null  int64  
 26  ct_srv_src         257673 non-null  int64  
 27  ct_state_ttl       257673 non-null  int64  
 28  ct_dst_ltm         257673 non-null  int64  
 29  ct_src_dport_ltm   257673 non-null  int64  
 30  ct_dst_sport_ltm   257673 non-null  int64  
 31  ct_dst_src_ltm     257673 non-null  int64  
 32  ct_flw_http_mthd   257673 non-null  int64  
 33  ct_src_ltm         257673 non-null  int64  
 34  ct_srv_dst         257673 non-null  int64  
 35  is_sm_ips_ports    257673 non-null  int64  
 36  label              257673 non-null  int64  
dtypes: float64(10), int64(27)
memory usage: 72.7 MB
```

```
numeric_cols = df_clean.select_dtypes(include=[np.number]).columns

outlier_indices = set()
for col in numeric_cols:
    Q1 = df_clean[col].quantile(0.25)
    Q3 = df_clean[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    mask = (df_clean[col] < lower_bound) | (df_clean[col] >
upper_bound)
    outliers = df_clean[mask].index
    outlier_indices.update(outliers)

print(f"Nombre de valeurs aberrantes détectées :
{len(outlier_indices)}")

#df_clean = df_clean.drop(index=outlier_indices)
#print("Dimensions après suppression des outliers :", df_clean.shape)

Nombre de valeurs aberrantes détectées : 210789

# --- 4. Vérification de la corrélation entre variables ---
corr_matrix = df_clean.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))
to_drop_corr = [column for column in upper.columns if
any(upper[column] > 0.95)]

df_clean = df_clean.drop(columns=to_drop_corr, errors='ignore')
print(f"Colonnes supprimées pour forte corrélation : {to_drop_corr}")
print("Dimensions finales du dataset :", df_clean.shape)

Colonnes supprimées pour forte corrélation : ['sbytes', 'dbytes',
'sloss', 'dloss', 'dwin', 'ct_src_dport_ltm', 'ct_dst_src_ltm',
'ct_srv_dst']
Dimensions finales du dataset : (257673, 29)

# --- 5. Visualisation de la matrice de corrélation ---
plt.figure(figsize=(10, 8))
sns.heatmap(df_clean.corr(), cmap='coolwarm', center=0)
plt.title("� Matrice de corrélation après nettoyage")
plt.show()

print("\nNettoyage et prétraitement terminés avec succès.")

C:\Users\raouf\AppData\Roaming\Python\Python313\site-packages\IPython\
core\pylabtools.py:170: UserWarning: Glyph 128293 (\N{FIRE}) missing
from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
Nettoyage et prétraitement terminés avec succès.

# --- 6. Define features (X) and target (y) ---
X = df_clean.drop(columns=['label'])
y = df_clean['label']

# --- 7. Split data into training & testing sets ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
print("\nTraining size:", X_train.shape)
print("Testing size:", X_test.shape)


Training size: (180371, 28)
Testing size: (77302, 28)
```

```
# --- 8. Scale numeric features ---
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test  = scaler.transform(X_test)

print("\nData scaling completed.")
```

```
Data scaling completed.
```

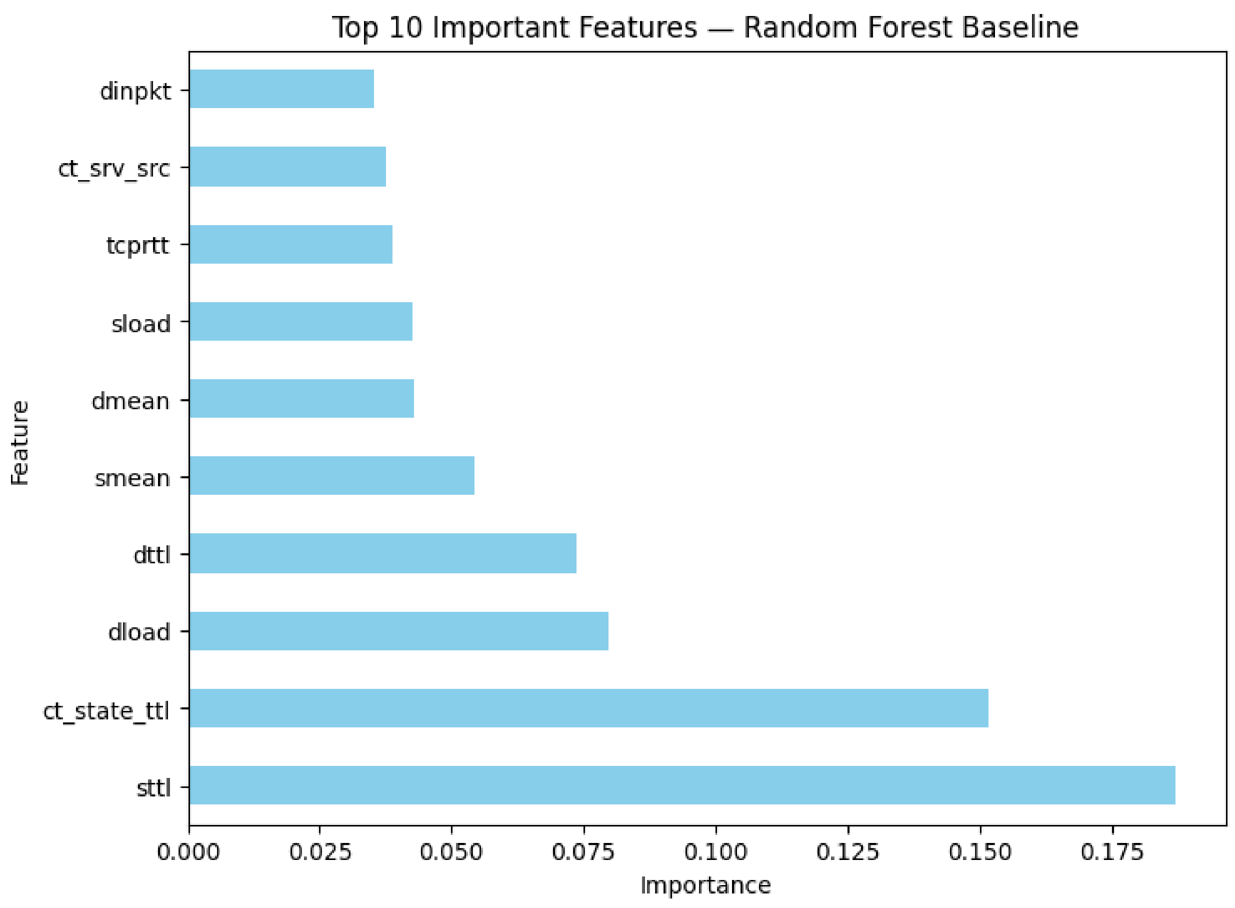
```
# --- 9. Train a baseline Random Forest model ---
model = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    random_state=42,
    n_jobs=-1
)
model.fit(X_train, y_train)

print("\nModel training complete.")
```

```
Model training complete.
```

```
# --- 10. Evaluate the model ---
y_pred = model.predict(X_test)
print("\n=== Classification Report ===")
print(classification_report(y_test, y_pred))
```

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.93      0.85      0.89     27941
           1       0.92      0.96      0.94     49361

    accuracy                           0.92     77302
   macro avg       0.93      0.91      0.92     77302
weighted avg       0.92      0.92      0.92     77302
```

```
# --- 11. Optional: Feature importance visualization ---
importances = pd.Series(model.feature_importances_, index=X.columns)
plt.figure(figsize=(8,6))
importances.nlargest(10).plot(kind='barh', color='skyblue')
plt.title('Top 10 Important Features — Random Forest Baseline')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



The objective of this project is to build a machine-learning model capable of classifying incoming network packets as either malicious (1) or benign (0) using the UNSW-NB15 dataset. Each record describes a network flow through 44 numerical and categorical features such as protocol type, byte counts, and connection flags. The task is a supervised binary classification problem: Inputs (X): the network traffic features. Output (y): the binary label label (0 = normal, 1 = attack). The baseline model used is a Random Forest Classifier, achieving an accuracy of 95 %, precision 0.96, recall 0.96, and F1-score 0.96. These results indicate that the model successfully detects malicious network behavior with high reliability.