```
import os
import pandas as pd
import numpy as np
import requests
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import pickle


# Step 1: Download the NSL-KDD files

base_url = "https://raw.githubusercontent.com/defcom17/NSL_KDD/master/"
files = ["KDDTrain+.txt", "KDDTest+.txt"]
for file in files:
    if not os.path.exists(file):
        print(f"Downloading {file}...")
        r = requests.get(base_url + file)
        with open(file, "wb") as f:
            f.write(r.content)
```

```
Downloading KDDTrain+.txt...
Downloading KDDTest+.txt...
```

```
# Step 2: Load and Prepare Data

columns = [
    "duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","urgent",
    "hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted","num_root",
    "num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login",
    "count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
    "diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count","dst_host_same_srv_rate",
    "dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate","dst_host_serror_rate",
    "dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate","label","difficulty"
]


# Load data
train_df = pd.read_csv("KDDTrain+.txt", names=columns)
test_df = pd.read_csv("KDDTest+.txt", names=columns)


# Combine for preprocessing
df = pd.concat([train_df, test_df], ignore_index=True)


# Drop difficulty
df.drop(columns=["difficulty"], inplace=True)


# Map attack types to categories
attack_mapping = {
    'normal': 'normal',
    # DoS
    'back': 'DoS', 'land': 'DoS', 'neptune': 'DoS', 'pod': 'DoS', 'smurf': 'DoS', 'teardrop': 'DoS',
    'mailbomb': 'DoS', 'apache2': 'DoS', 'processtable': 'DoS', 'udpstorm': 'DoS',
    # Probe
    'satan': 'Probe', 'ipsweep': 'Probe', 'nmap': 'Probe', 'portsweep': 'Probe', 'mscan': 'Probe', 'saint': 'Probe',
    # R2L
    'guess_passwd': 'R2L', 'ftp_write': 'R2L', 'imap': 'R2L', 'phf': 'R2L', 'multihop': 'R2L', 'warezmaster': 'R2L',
    'warezclient': 'R2L', 'spy': 'R2L', 'xlock': 'R2L', 'xsnoop': 'R2L', 'snmpguess': 'R2L', 'snmpgetattack': 'R2L',
    'httptunnel': 'R2L', 'sendmail': 'R2L', 'named': 'R2L',
    # U2R
    'buffer_overflow': 'U2R', 'loadmodule': 'U2R', 'rootkit': 'U2R', 'perl': 'U2R', 'sqlattack': 'U2R',
    'xterm': 'U2R', 'ps': 'U2R'
}

df['label'] = df['label'].map(attack_mapping)

# Fill NaN values created by the mapping with a placeholder
# This prevents the ValueError in train_test_split caused by unmapped labels
df['label'].fillna('unmapped', inplace=True)
```

```
<ipython-input-9-3538165c5989>:22: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me
```

```
    df['label'].fillna('unmapped', inplace=True)


# Separate features (X) and target (y)
X = df.drop(columns=["label"])
y = df["label"]

# Perform one-hot encoding for categorical features
X = pd.get_dummies(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Train Model

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Step 4: Evaluate and Save

y_pred = model.predict(X_test)
print("\nClassification Report (Multi-Class):")
print(classification_report(y_test, y_pred))

# Save model
with open("nsl_kdd_model.pkl", "wb") as f:
    pickle.dump(model, f)

print("\nModel saved as nsl_kdd_model.pkl")

with open("feature_columns.pkl", "wb") as f:
    pickle.dump(X.columns.tolist(), f)

print("\nFeatures saved as feature_columns.pkl")
```

```
Classification Report (Multi-Class):
              precision    recall  f1-score   support

      normal       0.99      1.00      1.00     15450
    unmapped       1.00      0.99      1.00     14254

    accuracy                           1.00     29704
   macro avg       1.00      1.00      1.00     29704
weighted avg       1.00      1.00      1.00     29704


Model saved as nsl_kdd_model.pkl

Features saved as feature_columns.pkl
```

```
from google.colab import files
files.download('nsl_kdd_model.pkl')
```

```
from google.colab import files
files.download('feature_columns.pkl')
```

```
print ("numpy", np.__version__)
print ("pandas", pd.__version__)
import sklearn
print("scikit-learn", sklearn.__version__)
```

```
numpy 2.0.2
pandas 2.2.2
scikit-learn 1.6.1
```

Versions required to download the same in vs code myenv