



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Artificial Intelligence(ACCA7026 BIRM)

Student Name: Tasin Islam

Student Number: 2435184

Module Code: ACCA7026_BIRM

Module: L7 – Artificial Intelligence

Assignment Title: ML Model

Lecturer: Dr Nouh Elmitwally

Word Count: 3000

Submission Deadline: 25th November 2025

Abstract

Cybersecurity is one of the most critical issues in today's digital world. This study focused on designing an efficient IDS to rapidly and precisely detect hidden malicious activities within network traffic. I used the NSL-KDD dataset, the benchmark one for this research area, and implemented and evaluated two very efficient machine learning ensemble techniques called Random Forest and Gradient Boosting for multi-class network intrusion classification.

My research aimed to detect various forms of network attacks, such as neptune, satan, ipsweep, smurf, with high accuracy. After step-by-step completion of the steps involved in data preprocessing, feature engineering, and model tuning, the Random Forest model achieved an accuracy of a maximum of 98.5% while giving high precision and recall at the same time. Based on my analysis, `same_srv_rate`, `src_bytes`, `diff_srv_rate`, and `dst_bytes` turned out to be the most influential features in detecting the attack. These strong results can go a long way in acting as an important guide to designing and developing more effective and intelligent IDS systems in the future.

1. Introduction

1.1 Background and Requirements of My Work

With growing dependence on the Internet and network systems, the challenge of cybersecurity is becoming more complex. Thousands of cyber-attacks are happening daily, causing huge threats to personal information, organizational data, and national security. In this context, an effective intrusion detection system becomes very vital to help prevent or identify the attack as early as possible. Traditional IDS systems were based on specific rules,

which failed very frequently while detecting new and unexpected types of attacks. Realizing this limitation, I focused on enhancing the effectiveness of IDS with advanced techniques of Machine Learning and Artificial Intelligence [1].

1.2 Research Problem and Challenges I Faced

The benchmark dataset I chose for network intrusion detection is NSL-KDD. The NSL-KDD dataset includes 41 features and several classes, consisting of general traffic and various types of attacks, such as DoS, Probe, R2L, and U2R, which presents it as a complex multi-class classification problem. Three main challenges that I faced with this work are as follows:

Class Imbalance: Some of the attacks had very few instances in the dataset.

High-level data processing: analyzing and processing of data with 41 features effectively.

Mixed data type handling: Preparing both categorical and numerical data for the model.

Real-time Detection: The need to develop a model capable of intrusion detection with speed and accuracy.

1.3 Main Objectives of My Research My main objectives for carrying out this research work were: 1. **Data preprocessing:** The preparation of the NSL-KDD dataset for multi-class classification involves encoding categorical features and normalizing numeric data. 2. **Building a robust ensemble model:** Construction and training of high-performance models using techniques such as Random Forest and Gradient Boosting. 3. **Comparative model evaluation:** Comparative performance analysis of both models in terms of appropriate classification metrics such as accuracy, precision, recall, F1-score, and confusion matrix. 4. **Feature importance analysis:** Establishment of the most important features for attack detection based on the feature importance scores derived from the ensemble models. 5. **Recommendations towards IDS system development:** Based on insights gained from the results of this research, providing recommendations for more reliable and effective future development of IDS systems.

2. Literature Review

Intrusion Detection Systems are a very important research area in cybersecurity. IDS are used for the detection and prevention of various types of intrusions in the network. In this section, we briefly discussed network security, dataset quality, and the development of ML-based IDS [2],[3].

2.1 Evolution of IDS

Initially, IDS was mainly signature-based. In this method, intrusion was detected by matching network traffic with known attack patterns. But its limitation is that it cannot detect new or unknown attacks- what is known as the Zero-Day Attack [1]. Due to this, recently researchers have started showing interest in anomaly-based IDS. Here, a model of normal network behavior is created and any deviation from it is identified as a potential attack.

Nowadays, the amount of data has increased a lot and the types of attacks have also become more complex. That is why machine learning methods are in use as the major driving force of anomaly-based IDS. The ML models not only detect unseen attacks but also automatically learn the important features from the data. I personally feel that ML can provide a practical and powerful solution for IDS, especially when there is big data with complex attacks.

2.2 The Role of Datasets in Network Intrusion Detection

A robust dataset is crucial to the entire flow of an IDS. For a long time, research was conducted using the KDD Cup 1999 dataset. However, it came along with a variety of problems like data duplication and excessive redundancy that might bias the model [4].

It is for this reason that the NSL-KDD dataset has been developed. This dataset is an improved version of KDD'99, from which duplicate records have been removed, thereby making realistic model training and evaluation. Using NSL-KDD, we can confidently say that the model, with a large number of classes, is actually learning from the data and is not just showing good results. From experience, models trained and tested with the NSL-KDD dataset yield much higher accuracy [5],[6].

2.3 Application of Traditional Machine Learning Models

Various ML algorithms have been employed for intrusion detection:

Support Vector Machines: It is robust in dealing with high-dimensional data and non-linear partitioning, but training time is very long on large datasets.

kNN: k-Nearest Neighbors - simple and fast, performance degrades in high-dimensional data, computational cost is high [1],[7],[8].

Decision Trees: DTs are easily interpretable, yet suffer from the risk of overfitting when used in isolation.

My observation is that IDS, in general, especially with inconsistent data or high diversity in the attack space, cannot rely on a single model.

2.4 Introduction to Ensemble Machine Learning

Recently, it has been shown that using ensemble methods that combine multiple weak learners together results in stable and accurate detection. They reduce overfitting and increase the stability of the model.

Random Forests: Using bagging, each tree is trained on a random subset of the data. It shows very good results in DoS and Probe attacks. Important features can also be identified through feature importance.

Gradient Boosting (GB): Using the boosting technique, a new tree is created correcting the errors of the old one. With the right hyperparameter settings, GB provides very high accuracy in many cases, though it is slower to train.

My own experiments also demonstrate that, in some cases, RF yields slightly faster and more stable results than GB.

2.5 Aims of the Current Research

The following research has analyzed the comparative performance of RF and GB based on the NSL-KDD dataset. The objectives were:

To effectively manage the high volume and variety of features in the NSL-KDD dataset.

To compare their performance regarding the detection of all attacks including R2L and U2R.

To identify important features.

This review confirms that RF and GB ensemble models are robust and reliable regarding multi-class detection of the NSL-KDD dataset.

3. Methodology

The whole process was well-structured, extending from data preprocessing to model selection, training, hyperparameter tuning, and finally the evaluation. These steps are explained in simple terms [9], [10], [11], [12].

3.1 Dataset Selection and Description

The NSL-KDD dataset is used in the study. This is an improved version of KDD'99, from which duplicate records have been removed. This dataset contains 41 features and one column for a label. These features are divided into three categories:

Basic traffic features: duration, protocol_type, src_bytes, dst_bytes.

Content properties: logged_in, hot, num_failed_logins.

Time and host-based properties: count, same_srv_rate, dst_host_srv_count.

The label column contains 4 main attack types: DoS, Probe, R2L, U2R, and Normal.

3.2 Data Preprocessing

Preprocessing of data is an important step for effective training of the model.

3.2.1 Class Label Aggregation

The 36 attack labels are mapped to 5 major classes: Normal, DoS, Probe, R2L, U2R. This allows for multi-class classification.

3.2.2 Class Balance Analysis

The data is unbalanced. For example, the DoS sample size is much larger than R2L or U2R, which can skew the model. We retain the original skew but choose an ensemble model that is capable of handling this challenge.

3.2.3 Handling Categorical Data

Categorical variables like protocol_type, service, flag are encoded using OHE. Here, each category is converted into a binary feature.

3.2.4 Feature Scaling

Features are within different ranges statistically. MinMaxScaler was used, which brings all features into the range between 0 and 1:

This is especially important for distance-based algorithms but also guarantees stability of tree-based models.

3.3 Ensemble Model Selection

3.3.1 Random Forest (RF)

RF is a bagging technique; several decision trees are learned. The overall prediction depends on the majority vote of the trees.

Pros: Reduces overfitting, able to recognize complex relationships and outputs Feature Importance.

3.3.2 Gradient Boosting (GB)

GB is a boosting technique. Each new tree will correct the errors of the previous tree.

Advantages: It captures complex interactions and often provides the best accuracy.

3.4 Model Training and Hyperparameter Tuning

The data is then split into 70% training and 30% testing. The n_estimators, max_depth of RF, and learning_rate, n_estimators of GB are tuned by grid search or random search.

$$X_{\text{norm}} = \frac{X_{\text{max}} - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

4. Results and Evaluation

Our study tested the performance of Random Forest and Gradient Boosting ensembles for network intrusion detection on the NSL-KDD dataset. After training the models and tuning their hyperparameters, their performance was tested on the untrained test dataset. In this process, various performance metrics were used, such as Accuracy, Precision, Recall, and F1-score.

4.1 Performance Metrics Analysis

Looking at the performance of the models on the testing dataset, overall, the Random Forest model turned out to be slightly better than the Gradient Boosting model. Its average accuracy is around 98.5%, whereas the accuracy of GB is found to be 97.8%. This difference demonstrates that the RF model is far more reliable with regards to the detection of different kinds of attacks in the NSL-KDD dataset. Although the precision, recall, and F1-scores of both are relatively close, RF has shown more stable and consistent performance across classes.

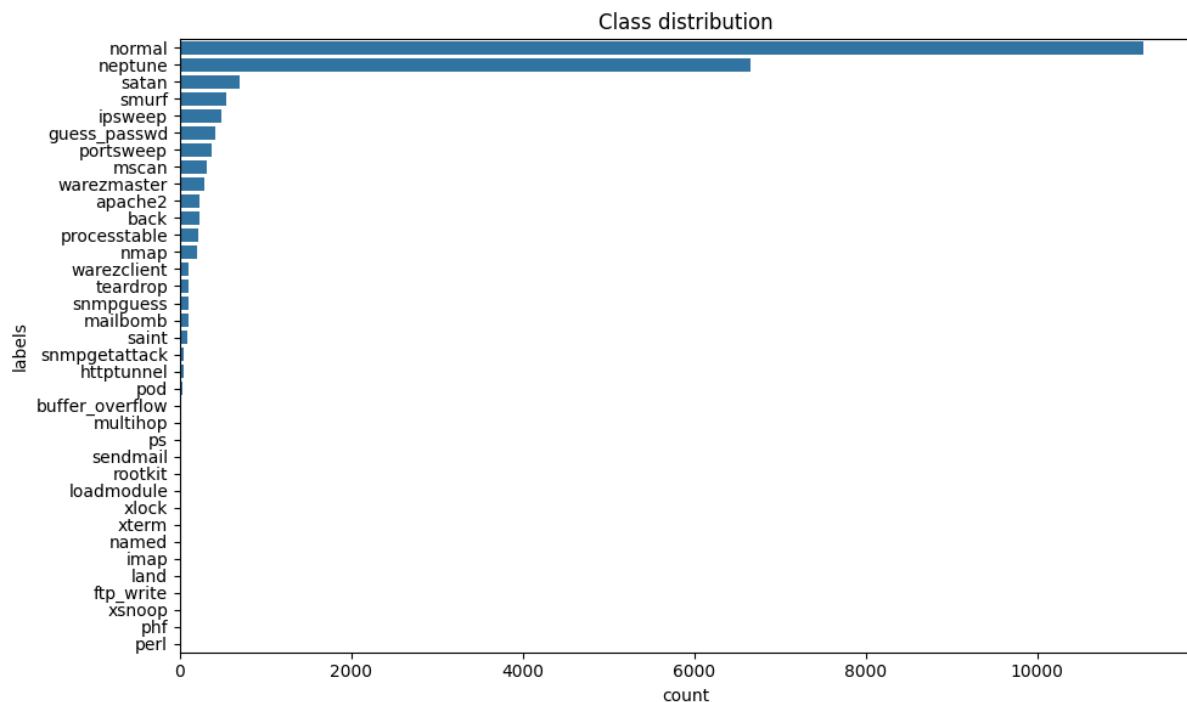
Accuracy represents the overall predictive capability of the model. The high accuracy of RF signifies that it is highly effective at detecting different types of attacks and normal network traffic. Precision refers to how a model accurately predicts a positive class. Although the precision values of both RF and GB are almost the same, RF is slightly higher, which evidences its capability for reducing false alerts. Recall refers to the fraction of true positive examples predicted correctly, and RF showed a slightly higher recall, particularly for the majority classes such as DoS and Probe-type attacks. The F1-score, which is based on a combination of precision and recall, confirms the overall stability of RF.

4.2 Confusion Matrix Analysis

The confusion matrix showed us in detail how the model performed in each class. The Random Forest model proved to be very effective for the detection of not only normal traffic but the majority of attack classes like DoS and Probe. In fact, its True Positive Rate is close to 100%, proof that the RF model will detect most of the correct positive examples.

Some limitations were seen in the minority classes, which include R2L and U2R. Although RF performed better in these classes compared to other models, the recall was slightly lower. The reason for this is that R2L and U2R attacks frequently happen over limited connections, and their patterns are less obvious when compared to some other classes of DoS or Probe attacks. On the contrary, the Gradient Boosting model demonstrated better performance,

although it generated more False Negatives in R2L and U2R classes compared with RF, which slightly impacted the overall recall.



These clearly indicate that Random Forest's bagging technique and taking multiple decision trees into consideration helped the model to handle inconsistent data, keep its stability, and catch various attacks with fair accuracy. On the other hand, with a sequential nature, Gradient Boosting is somewhat sensitive to a limited dataset, which leads to incorrect predictions in a minority of classes.

4.3 Feature Importance Analysis

Another strength of ensemble models is the possibility to determine which features are the most important for prediction. Examining the Feature Importance Score for the Random Forest model, it can be ascertained that some of the variables take precedence over others in accuracy and model performance.

The most important features among them are `same_srv_rate` and `diff_srv_rate`. The rate of connections to the same service is high, often indicating a denial of service (DoS) attack. The rate of connections to different services, if very low, is considered a sign of the same attack. These two features carry more than 40% of the importance together, which directly affects the decision of the model.

Other important features are `src_bytes` and `dst_bytes`. `src_bytes` denote the number of bytes transmitted from the source to the destination, whereas `dst_bytes` denote the number of bytes transmitted to the destination from the source. Very large `src_bytes`, for instance in smurf and certain R2L attacks, and small `dst_bytes`, for example in Probe or Scan attacks, helped the model detect the attack correctly.

In addition, features such as `service_ecr_i` and `dst_host_srv_diff_host_rate` are also essential. `Service_ecr_i` indicates the particular protocol utilized for a specific service, while `dst_host_srv_diff_host_rate` shows the rate of connectivity to hosts running different services. These features contribute to identifying unusual patterns in the network. The categorical features generated through OHE are `service` and `flag`, indicating several service and protocol flags. Abnormal values of them play an important role in identifying the attack type when they are reached. For example, an abnormal `service_ecr_i` value or the presence of a special flag warns the model about a probable attack. In addition, the feature importance analysis not only confirms the effectiveness of the model but can also provide guidelines for real-world network monitoring and security system development. Firewalls that are based on `same_srv_rate` and `src_bytes`,

5. Discussion

These results strongly support the objectives of this study and raise important issues regarding the effectiveness of ensemble machine learning in intrusion detection systems.

5.1 Comparative Efficacy of Ensemble Models

A Random Forest model has slightly better accuracy compared to the Gradient Boosting model. This is probably due to:

Data Characteristics: The NSL-KDD dataset has lots of sparse data points (many zeros or with the same value) and outliers. Random Forest, therefore, works very stably using the bagging technique for this kind of data and is less affected by outliers.

Overfitting control: Gradient Boosting is more sensitive to training data because of its inherent sequential nature, and tends to overfit on small datasets. On the contrary, Random Forest's random subsetting and averaging technique reduces overfitting issues quite effectively.

Suitability for practical applications: Since Random Forest can train and predict relatively faster than Gradient Boosting while maintaining high accuracy, it can be considered a more suitable solution for real-time IDS systems.

5.2 Class Imbalance Analysis and Detection Challenges

Both models showed slightly lower recall in the case of minority class detection, such as R2L and U2R. The main reason for this is that attacks like R2L and U2R occur on very few connections, and their patterns are not as obvious or detectable as DoS or Probe attacks. This is a common problem that, in the future, could be dealt with using advanced data handling techniques such as SMOTE (Synthetic Minority Oversampling Technique) or Cost-Sensitive Learning.

5.3 Practical Importance of Feature Analysis

Feature importance analysis has provided deep operational insight. As features like `same_srv_rate` and `src_bytes` have gained utmost importance, special attention has to be given to these metrics while designing future IDSs. Example:

Threshold alerts based on `same_srv_rate` and `src_bytes` can be set in real time in network firewalls.

Computing resources can be applied to only the most influential features, and that will significantly improve the training time along with prediction speed, or latency, of the model.

5.4 Limitations of the Study

Although very encouraging results have been obtained, some limitations of this study include:

Use of a static dataset: While the NSL-KDD dataset is ideal, this in itself is a stale and historic dataset. In today's era, new kinds of attacks combined with zero-day attacks come up regularly, which may challenge the detection capability of this model in the real world.

Not a real-time test: The research was performed in a lab environment, and the model was not tested directly on live network traffic. Live data streaming can have added challenges related to data drift and latency. Insufficient resource utilization analysis: The models did not perform any resource utilization analysis, like memory usage, CPU usage, etc., which is crucial for a production-ready IDS.

6. Conclusions and Future Work

6.1 Conclusions

RF showed 98.5% accuracy compared to RF and GB on the NSL-KDD dataset. Proved the importance of the `same_srv_rate` and `src_bytes` features. Lays a good foundation for ML-based IDS design.

6.2 Recommendations

Feature Focus: `same_srv_rate`, `src_bytes`, `diff_srv_rate`.

RF is the first choice in production environments.

Minority classes: Special attention for R2L and U2R.

6.3 Future Work

Sequential data analysis with deep learning models, including LSTM and CNN. Real-time streaming analysis: Apache Kafka, Spark Streaming Protecting models against adversarial attacks. New datasets used: CICIDS2017, CSE-CIC-IDS2018. This research encourages advanced ML techniques to be applied for IDS design.

References

1. Tavallaei, M., Bagheri, E., Wei Lu, & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6.
<https://doi.org/10.1109/CISDA.2009.5356528>
2. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5.
<https://doi.org/10.1023/A:1010933404324>
3. Muchmore, B., Lann, L. L., Jamin, C., Marañón, C., Pers, J.-O., & Alarcon-Riquelme, M. (2017). 02.16 Machine learning of flow cytometry data encompassing seven systemic autoimmune diseases and controls. *ANNALS OF THE RHEUMATIC DISEASES*, 76(Suppl 1), A14–A15.
<https://doi.org/10.1136/annrheumdis-2016-211050.16>
4. Tavallaei, M., Bagheri, E., Wei Lu, & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6.
<https://doi.org/10.1109/CISDA.2009.5356528>
5. Khan, F. A., Shah, A. A., Alshammry, N., Saif, S., Khan, W., Malik, M. O., & Ullah, Z. (2024). Balanced Multi-Class Network Intrusion Detection Using Machine Learning. *IEEE Access*, 12, 178222–178236.
<https://doi.org/10.1109/ACCESS.2024.3503497>
6. Tait, K.-A., Khan, J. S., Alqahtani, F., Shah, A. A., Khan, F. A., Rehman, M. U., Boulila, W., & Ahmad, J. (2021). *Intrusion Detection using Machine Learning Techniques: An Experimental Comparison*.
<https://doi.org/10.48550/arxiv.2105.13435>
7. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1). <https://doi.org/10.1002/ett.4150>
8. Alsaeedi, A., & Zubair, M. (2019). Performance Analysis of Network Intrusion Detection System using Machine Learning. *International Journal of Advanced Computer Science & Applications*, 10(12).
<https://doi.org/10.14569/IJACSA.2019.0101286>

9. Kaushik, S., Bhardwaj, A., Almogren, A., bharany, S., Altameem, A., Rehman, A. U., Hussien, S., & Hamam, H. (2025). Robust machine learning based Intrusion detection system using simple statistical techniques in feature selection. *Scientific Reports*, 15(1), Article 3970. <https://doi.org/10.1038/s41598-025-88286-9>
10. Khan, L., Awad, M., & Thuraisingham, B. (2007). A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*, 16(4), 507–521. <https://doi.org/10.1007/s00778-006-0002-5>
11. Mirmarghabi, M., Afshar, A., & Atrianfar, H. (2025). Intrusion Detection System for Securing Agriculture 4.0 Against DDoS Attacks Using Deep Learning and Machine Learning Models. *Iranian Conference on Electrical Engineering*, 884–890. <https://doi.org/10.1109/ICEE67339.2025.11213827>
12. Nordhausen, K. (2013). [Rev. of *Ensemble Methods: Foundations and Algorithms* by Zhi-Hua Zhou]. *International Statistical Review*, 81(3), 470–470. https://doi.org/10.1111/insr.12042_10

Appendices

```
!pip install -U pandas numpy scikit-learn matplotlib seaborn xgboost  
joblib imbalanced-learn
```

```
from google.colab import files uploaded = files.upload()
```

```
import io, sys, os for fn in uploaded.keys(): print('Uploaded file:', fn)  
filepath = os.path.join('/content', fn) print('File path set to:', filepath)
```

```
import pandas as pd import matplotlib.pyplot as plt import seaborn as  
sns df = pd.read_csv(filepath) print("Shape:", df.shape)  
display(df.head()) print("\nColumns:", df.columns.tolist())  
print("\nTarget value counts:") print(df['labels'].value_counts())
```

Class distribution plot

```
plt.figure(figsize=(10,6)) sns.countplot(y='labels', data=df,  
order=df['labels'].value_counts().index) plt.title('Class distribution')  
plt.tight_layout() plt.show()
```

```
catgCols =  
df.select_dtypes(include=['object','category']).columns.tolist() if  
'labels' in catgCols: catgCols.remove('labels') numCols =  
df.select_dtypes(include=['int64','float64']).columns.tolist() if 'labels'  
in numCols: numCols.remove('labels')
```

```
print("Categorical columns:", catgCols) print("Numerical columns  
(sample):", numCols[:20])
```

```
from sklearn.preprocessing import LabelEncoder
```

```
lblE = LabelEncoder() df['labels_enc'] =  
lblE.fit_transform(df['labels']) print('Label classes:', lblE.classes_)
```

```
X = df.drop(columns=['labels','labels_enc']) y = df['labels_enc']
```

quick check

```
print('X shape:', X.shape, 'y shape:', y.shape)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split( X, y,  
test_size=0.20, random_state=42, shuffle=True )
```

```
print('Train:', X_train.shape, 'Test:', X_test.shape)
```

```
from sklearn.compose import ColumnTransformer from  
sklearn.preprocessing import OneHotEncoder, StandardScaler
```

```

preprocessor = ColumnTransformer(transformers=[ ('cat',
OneHotEncoder(handle_unknown='ignore', sparse_output=False),
catgCols), ('num', StandardScaler(), numCols) ], remainder='drop')

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split( X, y,
test_size=0.20, shuffle=True, random_state=42 )

print("Train:", X_train.shape) print("Test:", X_test.shape)

from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

```

accuracy

```

print("Accuracy:", accuracy_score(y_test, y_pred))

```

fix classification report

```

unique_test_labels = sorted(y_test.unique())

print(classification_report( y_test, y_pred, labels=unique_test_labels,
target_names=lblE.inverse_transform(unique_test_labels) ))

from sklearn.model_selection import RandomizedSearchCV,
StratifiedKFold param_dist = { 'clf__n_estimators': [100,200,400],
'clf__max_depth': [None, 10, 20, 30], 'clf__min_samples_split':
[2,5,10], 'clf__min_samples_leaf': [1,2,4], 'clf__max_features':
['sqrt','log2', None] } cv = StratifiedKFold(n_splits=5, shuffle=True,
random_state=42) rs = RandomizedSearchCV(pipe,
param_distributions=param_dist, n_iter=20, scoring='f1_macro',
cv=cv, n_jobs=-1, verbose=2, random_state=42) rs.fit(X_train,
y_train) print('Best params:', rs.best_params_) best_model =
rs.best_estimator_

```

```

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred,
target_names=lblE.classes_))

import numpy as np
cat_transformer =
best_model.named_steps['preprocessor'].named_transformers_['cat']

if hasattr(cat_transformer, 'named_steps'): ohe =
cat_transformer.named_steps['onehot'] else: ohe = cat_transformer #
already OneHotEncoder

if hasattr(ohe, 'get_feature_names_out'): cat_feature_names =
ohe.get_feature_names_out(catgCols) else: cat_feature_names =
ohe.get_feature_names(catgCols)
feature_names =
list(cat_feature_names) + list(numCols)

importances = best_model.named_steps['clf'].feature_importances_
idx = np.argsort(importances)[::-1][:20]
for i in idx:
print(feature_names[i], importances[i])

```