Trường Đại học Công nghệ Thông tin – ĐHQG Tp. HCM
Khoa Mạng máy tính và truyền thông

BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

Môn học: Cơ chế hoạt động của mã độc

Tên chủ đề: DroidRL: Feature selection for android malware detection with reinforcement learning

Mã nhóm: G01 Mã đề tài: S45

Lớp: NT230.O21.ANTN

1. THÔNG TIN THÀNH VIÊN NHÓM:

(Sinh viên liệt kê tất cả các thành viên trong nhóm)

STT	Họ và tên	MSSV	Email
1	Hà Thị Thu Hiền	21522056	21522056@gm.uit.edu.vn
2	Phạm Ngọc Thơ	21522641	21522641@gm.uit.edu.vn
3	Nguyễn Ngọc Nhung	21521248	21521248@gm.uit.edu.vn

2. TÓM TẮT NỘI DUNG THỰC HIỆN:[1]

A. Chủ đề nghiên cứu trong lĩnh vực Mã độc: *(chọn nội dung tương ứng bên dưới)*

☑ Phát hiện mã độc

☐ Đột biến mã độc

☐ Khác: ……

B. Liên kết lưu trữ mã nguồn của nhóm:
Mã nguồn của đề tài đồ án được lưu tại: https://github.com/Ha-Hien/NT230.O21.ANTN-Malware.git

(Lưu ý: GV phụ trách phải có quyền truy cập nội dung trong Link)

C. Tên bài báo tham khảo chính:

<ghi đúng định dạng tài liệu bài báo tham khảo của nhóm theo chuẩn IEEE>

Wu, Y., Li, M., Zeng, Q., Yang, T., Wang, J., Fang, Z., & Cheng, L. (2023). DroidRL: Feature selection for android malware detection with reinforcement learning. *Computers & Security*, *128*, 103126.

[1] Ghi nội dung tương ứng theo mô tả

D. Dịch tên Tiếng Việt cho bài báo:

<dịch tên tiếng Việt cho bài báo trên>

DroidRL: Lựa chọn đặc trưng để phát hiện phần mềm độc hại trên Android với tính năng học tăng cường.

E. Tóm tắt nội dung chính:

<mô tả nội dung tóm tắt của bài báo/chủ đề trong vòng 350 từ>

DroidRL là một framework được thiết kế để chọn lựa các đặc trưng hiệu quả nhằm phát hiện malware trên hệ điều hành Android bằng cách sử dụng học tăng cường. Bài báo tập trung vào việc áp dụng Double Deep Q-Network (DDQN) kết hợp với mạng nơ-ron hồi quy (RNN) để lựa chọn các đặc trưng theo trình tự. Với việc sử dụng phương pháp nhúng từ (word embedding) để biểu diễn đối tượng, DroidRL có thể tăng cường sự liên quan ngữ nghĩa giữa các đặc trưng. Các kết quả thực nghiệm cho thấy DroidRL có thể giảm đáng kể số lượng đặc trưng từ 1083 xuống còn 24 mà vẫn giữ độ chính xác cao (95.6%) khi sử dụng bộ phân loại Random Forest.

F. Tóm tắt các kỹ thuật chính được mô tả sử dụng trong bài báo:

<mô tả các kỹ thuật chính được dùng trong nghiên cứu của bài báo>

(liệt kê tóm tắt vai trò của các kỹ thuật đó – kèm hình ảnh minh họa về hệ thống/kỹ thuật/phương pháp)

VD: Bài báo sử dụng 4 kỹ thuật cho 4 giai đoạn chính trong phương pháp đề xuất, bao gồm:

- Kỹ thuật 01: DroidRL triển khai thuật toán DDQN - thuật toán học tăng cường Reinforcement learning (Double Deep Q - Network) ⇒ thu được tập hợp con đặc trưng tối ưu ⇒ phân loại malware hiệu quả.
- Kỹ thuật 02: Mạng thần kinh tái phát (RNN - Recurrent NN) được sử dụng làm mạng quyết định của DDQN ⇒ cung cấp khả năng chọn tuần tự các đặc trưng.
- Kỹ thuật 03: Word embedding được áp dụng để biểu diễn đối tượng ⇒ nâng cao khả năng làm việc của framework nhằm tìm ra mức độ liên quan về mặt ngữ nghĩa của các đối tượng.
- Kỹ thuật 04: Sử dụng Random Forest làm lớp phân loại ⇒ đánh giá hiệu suất.

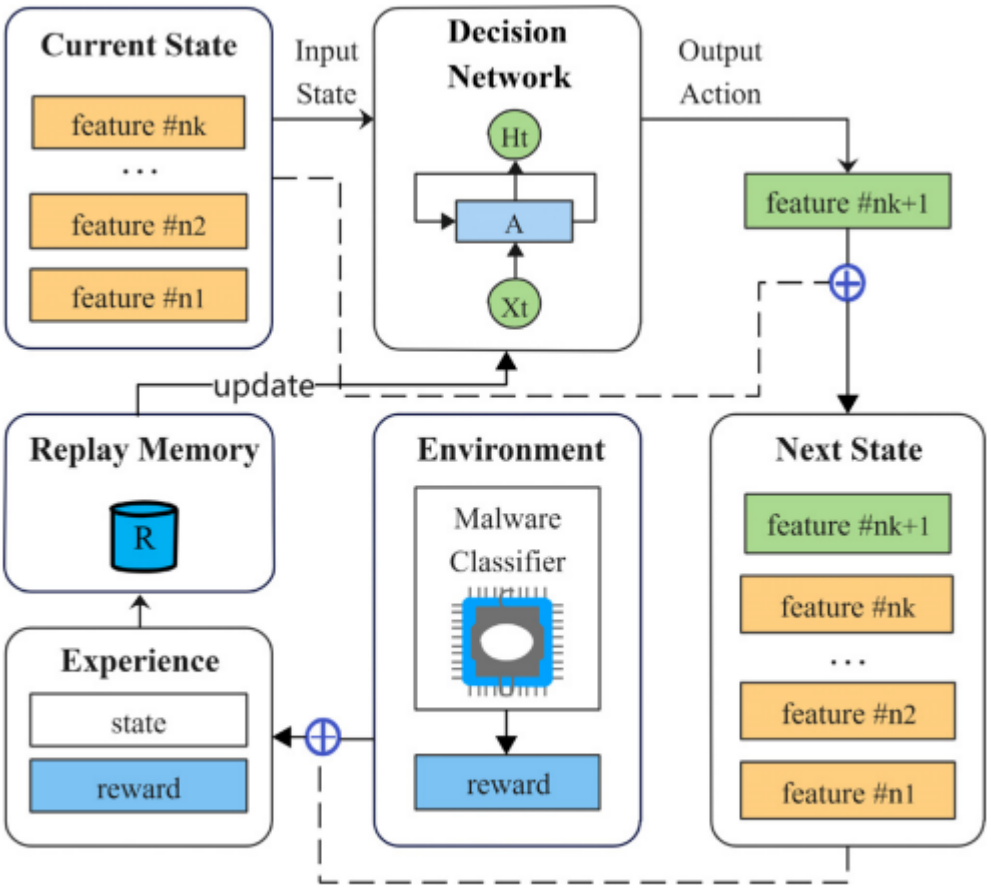

Fig. 1. DroidRL for feature selection in Android malware detection.

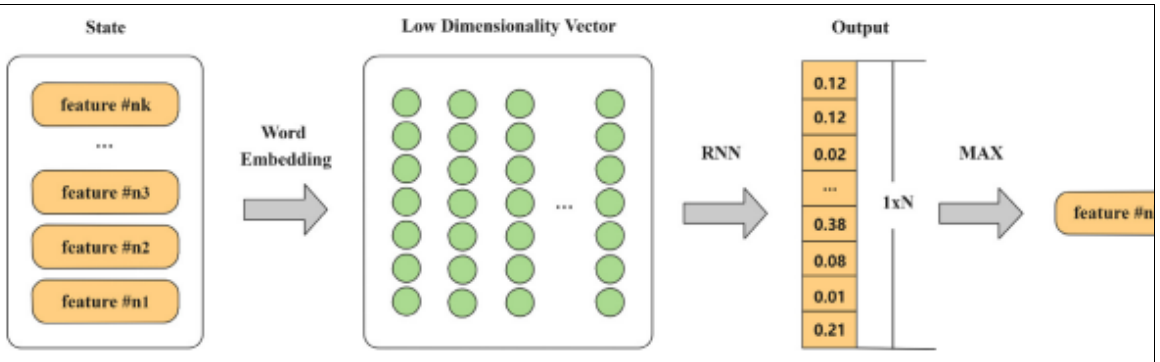

Fig. 2. DroidRL's Decision Network. The decision network takes in the agent's current state and predicts a new feature as action.

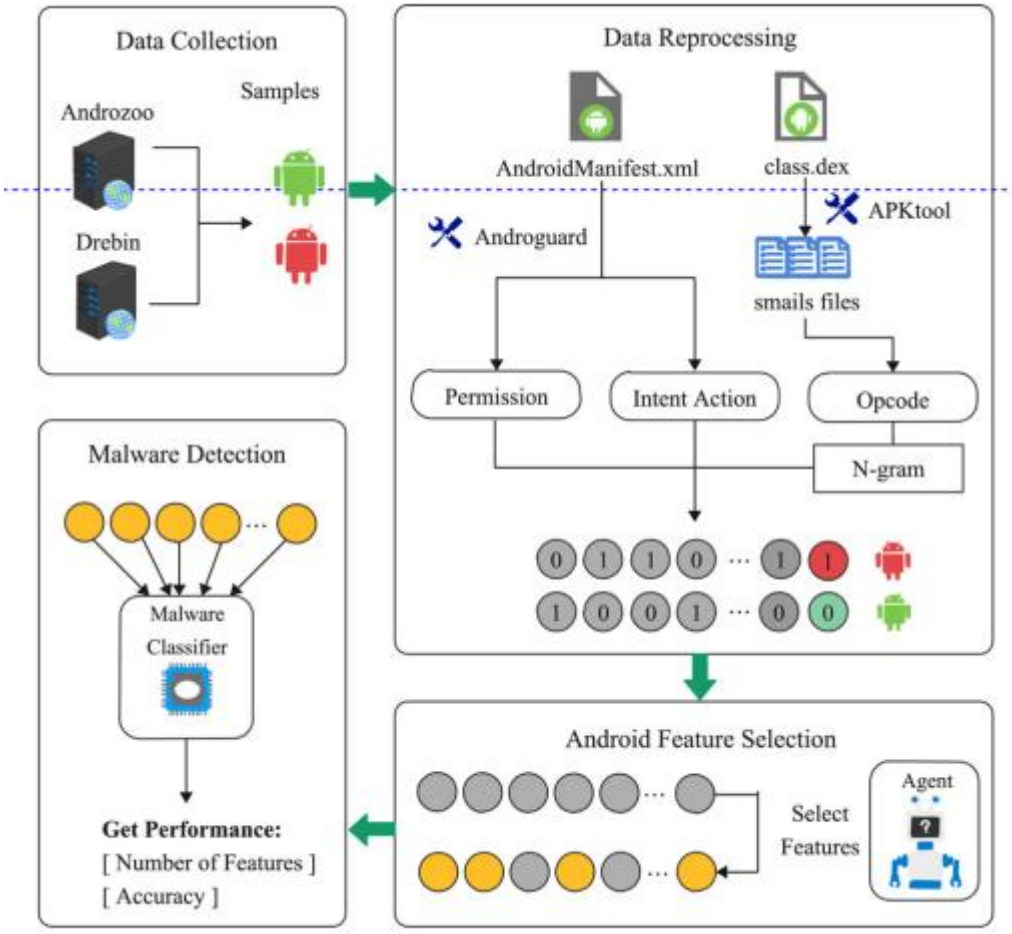

Fig. 3. Overview of DroidRL.

G. Môi trường thực nghiệm của bài báo:

<mô tả môi trường thực nghiệm của nhóm tác giả, bao gồm cấu hình máy tính (phần cứng, phần mềm), các chương trình hỗ trợ để hiện thực phương pháp, ngôn ngữ lập trình được sử dụng, các chương trình phần mềm dùng để kiểm tra khả năng/tính năng của phương pháp>

- Cấu hình máy tính: CPU: Intel Core i7
- RAM: 16GB
- GPU: NVIDIA GTX 1080
- Các công cụ hỗ trợ sẵn có:
  - TensorFlow (dùng cho việc triển khai mạng nơ-ron)
  - Scikit-learn (dùng cho việc đánh giá mô hình)
- Ngôn ngữ lập trình: Python
- Đối tượng nghiên cứu (chương trình phần mềm dùng để kiểm tra tính khả thi của phương pháp/tập dữ liệu – nếu có): Tập dữ liệu của DroidRL chứa **5000 benign sample** từ **Andro Zoo** và **5560 malware** từ **Drebin** để huấn luyện và kiểm tra mô hình.
- Tiêu chí đánh giá tính hiệu quả của phương pháp:
  - Số lượng đặc trưng được chọn
  - Độ chính xác của việc phát hiện malware (accuracy)
  - Thời gian chạy

H. Kết quả thực nghiệm của bài báo:

<mô tả ngắn gọn kết quả thực nghiệm của bài báo, tự nhận xét về khả năng, ưu và nhược điểm của phương pháp được đề cập trong bài báo>

DroidRL được đề xuất áp dụng thuật toán DDQN cho giai đoạn lựa chọn đặc trưng để chọn tập hợp con đặc trưng tối ưu nhằm phát hiện malware trên Android. Đặc biệt, mạng giống RNN được ứng dụng làm mạng quyết định trong DDQN nhờ khả năng xử lý các chuỗi có độ dài thay đổi. Với mục đích tìm ra mối tương quan giữa các đặc trưng, DroidRL sử dụng đặc trưng nhúng từ (word embedding) để thể hiện một cách chính xác các đặc trưng. Trong giai đoạn đào tạo, chính sách đào tạo được sử dụng để tăng không gian tìm kiếm đặc trưng của DroidRL. Các thử nghiệm trên Drebin và Androzoo chứng minh rằng framework DroidRL cho thấy hiệu suất tốt hơn so với các mô hình trích xuất đặc trưng tĩnh truyền thống, cải thiện rõ rệt hiệu suất phát hiện trên nhiều bộ phân loại khác nhau. DroidRL đã chứng minh khả năng lựa chọn đặc trưng hiệu quả, giảm từ **1083 xuống còn 24 đặc trưng**, đồng thời duy trì độ chính xác phát hiện malware ở mức **95.6%. Phương pháp này không chỉ tiết kiệm thời gian tính toán mà còn tăng cường hiệu suất của bộ phân loại**. **DroidRL được chứng minh là có hiệu quả trong các nhiệm vụ lựa chọn đặc trưng và hy vọng nó sẽ đóng vai trò là một yếu tố để xây dựng một trình phát hiện malware mạnh mẽ trong tương lai.**

I. Công việc/tính năng/kỹ thuật mà nhóm thực hiện lập trình và triển khai cho demo:

<liệt kê các công việc mà nhóm thực hiện cho đề tài dựa trên phân tích phương pháp/hệ thống được sử dụng trong bài báo đã tham khảo>

<liệt kê các công việc đã thực hiện+ tóm tắt kết quả của công việc này trong việc thực hiện phương pháp/demo>

- Thu thập bộ dữ liệu bao gồm **1514 mẫu lành tính và 1054 mẫu độc hại** từ Index of /CICDataset (CICAndMall2017 và CICAndMall2020), nhóm em chỉ lấy một số cái có thể tải được và phù hợp với tài nguyên hiện có.
- Thực thiện trích xuất đặc trưng thô từ các tệp APK, kết quả cho ra tệp JSON.

```
: !python3 droidlysis/droidlysis3.py --input test/benign --output ./test/output --config droidlysis/conf/general.conf

Processing file: test/benign/air.beingfashiondesigner.apk ...
Launching unzip process on test/benign/air.beingfashiondesigner.apk with output dir=./test/output/air.beingfashiondesi
gner.apk-7ce60a541c7d73c30181e4840b58dc348d2aa6baddcdba0e124e91dd1ed2f1f4/unzipped
caution: filename not matched:  classes2.dex
============= Report ============
Sanitized basename  : air.beingfashiondesigner.apk
SHA256              : 7ce60a541c7d73c30181e4840b58dc348d2aa6baddcdba0e124e91dd1ed2f1f4
File size           : 19673487 bytes
Is small            : False
Nb of classes       : 7830
Nb of dirs          : 264

Certificate properties
algo                : None
serialno            : None
country             : 500
owner               : None
timestamp           : (2016, 6, 6, 11, 21, 6)
year                : 2016

Manifest properties
activities          : ["'.AppEntry'", "'com.chartboost.sdk.CBImpressionActivity'", "'com.google.android.gms.ads.AdActi
vity'"]
main_activity       : .AppEntry
package_name        : air.beingfashiondesigner
permissions         : ['INTERNET', 'READ_PHONE_STATE', 'ACCESS_NETWORK_STATE', 'DISABLE_KEYGUARD', 'WAKE_LOCK', 'ACCES
S_FINE_LOCATION', 'ACCESS_COARSE_LOCATION', 'WRITE_EXTERNAL_STORAGE']
swf                 : True
```

- Viết mã trích xuất đặc trưng từ tệp JSON đã có theo chuẩn nhất định.

```
def get_static_feature(data):
    values = dict()
    for x in data.keys():
        if x == "sanitized_basename":
            continue
        if isinstance(data[x], dict):
            for key in data[x].keys():
                if isinstance(data[x][key], list):
                    #print(x, key)
                    if key == "permissions":
                        for perm in data[x][key]:
                            values[f'permissions_{perm.split(".")[-1]}'] = 1

                    values[f'{x}_{key}_len']  = len(data[x][key])
                    continue
                if key in ['main_activity', 'package_name', 'app_name']:
                    continue
                try:
                    values[f'{x}_{key}'] = int(data[x][key])
                except:
                    values[f'{x}_{key}'] = -1
        else:
            values[x] = int(data[x])
    return values
```

- Viết mã lấy dữ liệu từ các mẫu lành tính và các mẫu độc hại và thực hiện gán nhãn cho các mẫu (lành tính là 0, độc hại là 1).

```
import os, json
import pandas as pd

def get_dataset(folder, lable):
    dataset = dict()
    for filename in os.listdir(folder):
        dataset[filename] = get_static_feature(json.load(open(f'{folder}/{filename}')))
    df=pd.DataFrame(dataset).T
    df.fillna(value=0, inplace=True)
    df['class'] = lable
    return df
```

```
df_ben = get_dataset("malware/malware_json", 1)
df_mal = get_dataset("benign/benign_json", 0)
```

```
df_ben.shape
```

```
(1514, 1256)
```

```
df_mal.shape
```

```
(1054, 1383)
```

- Chuyển các dữ liệu đã có thành dataframe và ghép lại thành một bộ dự liệu để huấn luyện và kiểm tra.

```
# Kết hợp hai DataFrame
df_combined = pd.concat([df_ben, df_mal], ignore_index=True)
df_combined.fillna(value=0, inplace=True)

# Trộn ngẫu nhiên các hàng
HinT = df_combined.sample(frac=1).reset_index(drop=True)

HinT.head(5)
```

	file_nb_classes	file_nb_dir	file_size	file_small	filetype	file_innerzips	manifest_properties_activities_len	manifest_properties_lib
0	41.0	6.0	1612197.0	0.0	1.0	0.0	8.0	
1	0.0	0.0	836367.0	0.0	1.0	0.0	11.0	
2	465.0	30.0	561174.0	0.0	1.0	0.0	17.0	
3	178.0	29.0	172599.0	0.0	1.0	0.0	2.0	
4	4691.0	138.0	7885466.0	0.0	1.0	0.0	9.0	

5 rows × 1436 columns

- Tiến hành huấn luyện và đánh giá trên 4 mô hình là **RandomForestClassifier, LGBMClassifier, ExtraTreesClassifier, DecisionTreeClassifier** cho toàn bộ tất cả các feature trích xuất được (1435 đặc trưng). **Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time**.

```
import time
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from lightgbm import LGBMClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_s
```

```
# Initialize models
rf_model = RandomForestClassifier(random_state=42)
lgbm_model = LGBMClassifier(random_state=42)
et_model = ExtraTreesClassifier(random_state=42)
dt_model = DecisionTreeClassifier(random_state=42)
```

```
def evaluate_model(model, X_train, y_train, X_test, y_test):
    start_time = time.time()
    model.fit(X_train, y_train)
    end_time = time.time()

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])

    return accuracy, precision, recall, f1, roc_auc, end_time - start_time
```

```
# Create a dictionary to store results
results = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1': [], 'ROC-AUC': [], 'Time'
```

```
# Loop through each model
for model, model_name in zip([rf_model, lgbm_model, et_model, dt_model],
                             ['Random Forest', 'LightGBM', 'Extra Trees', 'Decision Tree']):

    # Evaluate model performance and time
    accuracy, precision, recall, f1, roc_auc, run_time = evaluate_model(model, X_train, y_train, X_test, y_te

    # Store results in the dictionary
    results['Model'].append(model_name)
    results['Accuracy'].append(round(accuracy, 4))
    results['Precision'].append(round(precision, 4))
    results['Recall'].append(round(recall, 4))
    results['F1'].append(round(f1, 4))
    results['ROC-AUC'].append(round(roc_auc, 4))
    results['Time'].append(round(run_time, 4))
```

```
# Create a performance evaluation/comparison table
results_df = pd.DataFrame(results)
print(results_df)
```

	Model	Accuracy	Precision	Recall	F1	ROC-AUC	Time
0	Random Forest	0.9922	0.9934	0.9934	0.9934	0.9998	0.2075
1	LightGBM	0.9981	1.0000	0.9967	0.9983	1.0000	0.2264
2	Extra Trees	0.9961	0.9967	0.9967	0.9967	0.9999	0.2986
3	Decision Tree	0.9883	0.9934	0.9868	0.9901	0.9887	0.0296

- Thực hiện phương pháp lựa chọn đặc trưng (lựa chọn được **323/1435** đặc trưng)

```
def evaluate_one_feature(feature, index='', metric=roc_auc_score):
    rootnode = DecisionTreeClassifier(max_depth=1, criterion='gini')
    rootnode.fit(X_train[feature].array.reshape(-1,1), y_train)
    preds = rootnode.predict(X_test[feature].array.reshape(-1,1))
    preds_tr = rootnode.predict(X_train[feature].array.reshape(-1,1))
    met = round(metric(y_test, preds), 4)
    if met > 0.5:
        return [feature, met, rootnode, preds, preds_tr]
    else:
        return [feature, met, None, [], []]
```

```
from joblib import Parallel, delayed
import multiprocessing

# Sử dụng joblib để chạy hàm evaluate_one_feature song song trên các đặc trưng
def parallel(f, items, n_workers=None, threadpool=False, progress=True):
    if n_workers is None:
        n_workers = multiprocessing.cpu_count()

    if threadpool:
        from multiprocessing.pool import ThreadPool as Pool
    else:
        from multiprocessing import Pool

    with Pool(n_workers) as pool:
        results = list(pool.imap(f, items))

    return results

# "conts` là danh sách các đặc trưng muốn đánh giá
results = parallel(f=evaluate_one_feature,
                   items=conts, n_workers=multiprocessing.cpu_count(), threadpool=False, progress=True)
```

```
    useful_features = result_df.loc[result_df['roc_auc_score'] > 0.5]
    print(f"{len(useful_features)} / {len(conts)} features have direct separating power (linear
```

```
323 / 1435 features have direct separating power (linear)
```

- Tiến hành huấn luyện và đánh giá lại 4 mô hình đã thiết lập dựa trên bộ dữ liệu đã được tiến hành phương pháp chọn đặc trưng. **Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time**.

Create a new data ¶

```
# Create new dataset with selected features
selected_features = useful_features['feature'].tolist()
X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]
```

```
# Optionally print the results
import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)
```

```
           Model  Accuracy  Precision  Recall      F1  ROC-AUC    Time
0  Random Forest    0.9202     1.0000  0.8647  0.9274   0.9999  0.1777
1       LightGBM    0.9942     1.0000  0.9901  0.9950   1.0000  0.1028
2    Extra Trees    0.9319     1.0000  0.8845  0.9387   1.0000  0.1363
3  Decision Tree    0.9864     0.9901  0.9868  0.9884   0.9863  0.0187
```

```
# Lưu tên các đặc trưng sử dụng trong quá trình huấn luyện
feature_names = X_train_selected.columns.tolist()
joblib.dump(feature_names, 'feature_names.joblib')
```

```
['feature_names.joblib']
```

```
len(feature_names)
```

```
323
```

- Kết quả cho thấy, sau khi thực hiện phương pháp lựa chọn đặc trưng, thì thời gian đã giảm đi, độ chính xác tuy giảm nhưng mức độ tuỳ thuộc vào thuật toán sử dụng để phân loại. Như mô hình LGBM cho thấy tất cả các chỉ số đánh giá đều không chênh lệnh nhiều so với lúc chưa giảm số lượng đặc trưng.

J. Các khó khăn, thách thức hiện tại khi thực hiện:

<mô tả khó khăn của nhóm (nếu có), ...>

- Bài báo không hề cung cấp bộ dữ liệu cũng như là các mã code liên quan, nên nhóm em dù hiểu bài báo nhưng để thực nghiệm lại thì nhóm em chưa đủ kiến thức.

- Khó khăn của nhóm em ban đầu là ở việc thu thập bộ dữ liệu, máy chủ nơi lưu trữ có đôi lúc bị sập, cũng như là các tài nguyên trên đây có dung lượng khá lớn. Nhóm em có thử nhiều cách tải, nhưng bị lỗi 403 Forbidden nhiều lần. Cố gắng lắm và cuối cùng tải được tầm hơn 2500 mẫu.
- Phương pháp học tăng cường để lựa chọn đặc trưng còn quá mới với nhóm em, và cũng có quá nhiều kỹ thuật nên nhóm em quyết định chọn một phương pháp lựa chọn đặc trưng khác.

3. TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:

95%

4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

STT	Công việc	Phân công nhiệm vụ
1	Tải và tìm hiểu bài báo	Hà Thị Thu Hiền, Phạm Ngọc Thơ, Nguyễn Ngọc Nhung
2	Thu thập bộ dữ liệu	Hà Thị Thu Hiền
3	Tìm kiếm cách trích xuất đặc trưng từ APK	Hà Thị Thu Hiền, Phạm Ngọc Thơ, Nguyễn Ngọc Nhung
4	Trích xuất đặc trưng, hình thành bộ dữ liệu hoàn chỉnh, tiền xử lý.	Hà Thị Thu Hiền
5	Chọn mô hình và thực hiện kịch bản demo, slide, thuyết trình.	Hà Thị Thu Hiền, Phạm Ngọc Thơ, Nguyễn Ngọc Nhung

BÁO CÁO TỔNG KẾT CHI TIẾT

Phần bên dưới của báo cáo này là tài liệu báo cáo tổng kết - chi tiết của nhóm thực hiện cho đề tài này.

Qui định: Mô tả các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, số liệu thống kê trong bảng biểu, có giải thích)

A. Phương pháp thực hiện

<Trình bày kiến trúc, thành phần của hệ thống trong bài báo>

1. DroidRL framework

Đối với framework DroidRL, nhiệm vụ chính là đào tạo tác nhân học cách chọn tuần tự các tập hợp con đặc trưng Android hợp lệ bằng cách tương tác với môi trường và sử dụng kiến thức đã học của nó. Phần này mô tả cách khung DroidRL đạt được mục tiêu của nó.

1.1. Overview of DroidRL

Hình 1 thể hiện sơ đồ nguyên lý của DroidRL. Phần cốt lõi của framework DroidRL được xây dựng bởi mạng quyết định (**decision network**) dựa trên DDQN (**Double Deep Q Network**). Trong mỗi bước, tác nhân tự trị thực hiện một cách độc lập một hành động do mạng quyết định **quyết định để chọn một đặc trưng vào trạng thái được quan sát** từ môi trường bằng cách sử dụng kiến thức trước đó. **Để đánh giá chất lượng của tập hợp con đặc trưng và sự khác biệt của các đặc trưng riêng lẻ đã chọn**, **reward** của hành động được tạo ra từ trình phân loại phần mềm độc hại **được xác định bằng độ chính xác của phân loại phần mềm** độc hại sử dụng các đặc trưng được chọn làm đầu vào. Hơn nữa, trạng thái của tác nhân, hành động được chọn và reward của thời điểm này được lưu trong bộ nhớ phát lại **để huấn luyện mạng quyết định**. Chính sách thăm dò-khai thác (**exploration-exploitation**) được tăng cường để **giải quyết vấn đề chi phí tính toán do các tập hợp đặc trưng không thể cạn kiệt.**

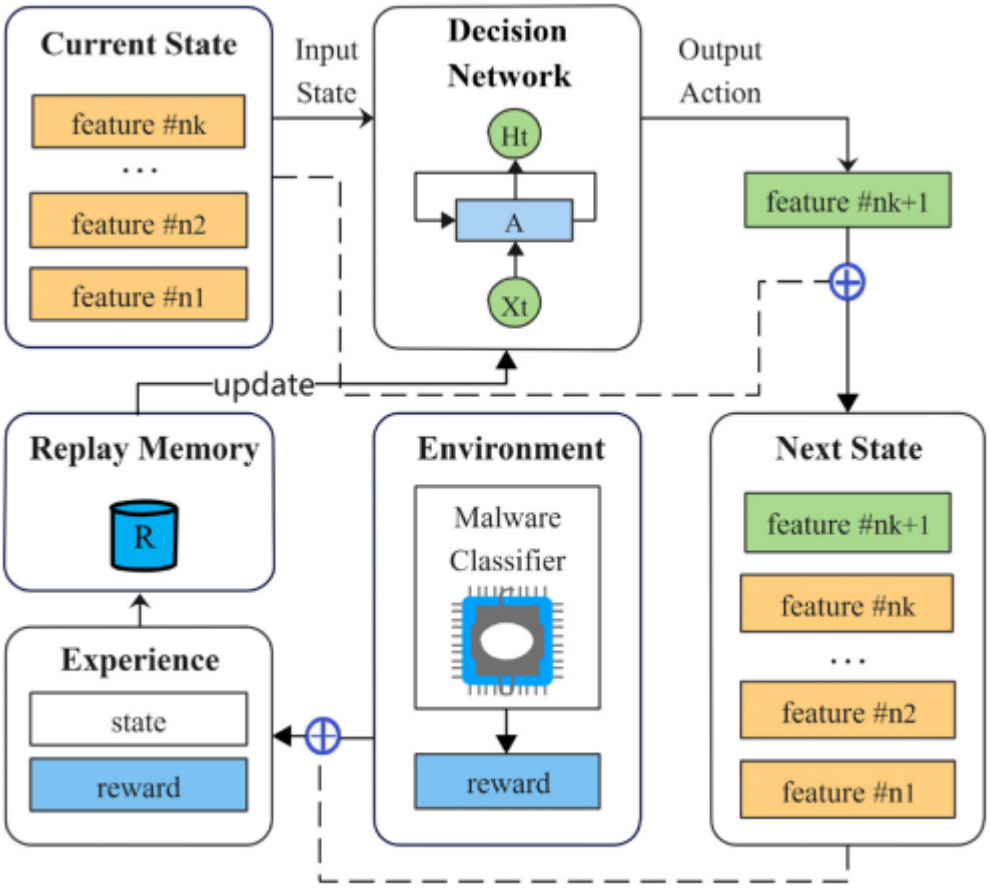

Fig. 1. DroidRL for feature selection in Android malware detection.

1.2 Key components in reinforcement learning

(1) Môi trường (Environment): Môi trường là nơi để tác nhân khám phá và nhận phản hồi.

Trong framework của tác giả, **môi trường chứa tất cả các đặc trưng ứng viên** và **chịu trách nhiệm đưa trạng thái hiện tại của tác nhân vào bộ phân loại phần mềm độc hại sau mỗi hành động được thực thi. Độ chính xác của việc phân loại được trả lại cho tác nhân dưới dạng reward. Tổng số đặc trưng nhất định** và **độ dài của các tập hợp đặc trưng hợp lệ cần được chọn** được xác định trong **môi trường. Khi tác nhân đã chọn đủ đặc trưng theo độ dài đã khai báo của tập hợp con đặc trưng hợp lệ, môi trường sẽ hướng dẫn tác nhân kết thúc vòng khám phá (exploration) này, trả lại reward cuối cùng và tự thiết lập lại.**

(2) Hành động (Action): Hành động là bước quan trọng mà tác nhân trong học tăng cường cần thực hiện từ không gian hành động dựa trên kinh nghiệm (experience) và trạng thái hiện tại (current state) của nó.

Trong framework DroidRL, **không gian hành động chứa các đặc trưng trong bộ đặc trưng thô được trích xuất thông qua các tệp APK đã dịch ngược. Trạng thái của tác nhân mô tả các đặc trưng hiện được chọn là kết quả của một loạt hành động.**

Nhiệm vụ chính của tác nhân là tìm ra tập hợp con đặc trưng tối ưu có khả năng phân biệt cao giữa phần mềm độc hại và phần mềm Android lành tính.

Đối với mỗi hành động trong framework DroidRL, **một đặc trưng không được chọn sẽ được thêm vào trạng thái**. **Thuật toán ε-Greedy** được sử dụng để thực hiện **sự cân bằng giữa việc thăm dò và khai thác**. Mỗi hành động được khám phá với **xác suất là ε**, trong khi **hành động có giá trị Q lớn nhất** được khai thác **với xác suất là 1 - ε**. Nhằm mục đích cho phép tác nhân trong giai đoạn đào tạo khám phá nhiều hơn ở giai đoạn đầu và khai thác nhiều hơn **bằng cách sử dụng trải nghiệm hiện có ở giai đoạn sau**, **một số cải tiến đã được thực hiện đối với thuật toán ε-Greedy như được hiển thị trong biểu thức. (1),** trong đó **episode là vòng huấn luyện hiện tại**, **E là tổng số vòng huấn luyện** và **P là tham số xác suất nằm trong khoảng từ 0 đến 1**. Thông tin chi tiết được đưa ra trong Phần 4.

$$\varepsilon = 1 - \frac{episode}{E} \times p \qquad (1)$$

(3) Reward: Reward là phản hồi do sự tương tác giữa tác nhân và môi trường thông qua hành động.

Trong bài viết này, **reward được xác định bởi độ chính xác của trình phân loại phần mềm độc hại trên Android, với các đặc trưng được chọn ở trạng thái hiện tại của tác nhân làm đầu vào**. **Tác nhân vào trạng thái s sau khi thực hiện hành động lựa chọn đặc trưng a**. Sau đó, **môi trường học tăng cường trả về reward tương ứng** từ bộ phân loại phần mềm độc hại **để đánh giá hàm giá trị hành động Q**

Với mục tiêu đạt được giá trị Q cao nhất bằng hành động, do đó **tác nhân có thể tìm thấy sự kết hợp đặc trưng hợp lệ dành cho độ chính xác cao nhất.** Trong phương trình. (2), **s và a lần lượt thể hiện trạng thái hiện tại và hành động được thực hiện ở bước hiện tại**. **r là reward nhận được**, **s' đại diện cho trạng thái tiếp theo mà tác nhân đạt được và a' đề cập đến hành động có thể đạt được giá trị Q cao nhất ở trạng thái tiếp theo**. Vì a' cũng có thể được tính bằng hàm Q nên **công thức gốc tương đương với biểu thức. (3),** trong đó **θ1 và θ2 lần lượt biểu thị các tham số của hai mạng trong DDQN.**

$$Q_{(s,a)} = r + \gamma Q(s', a') \qquad (2)$$

$$Q_{(s,a,\theta_1)} = r + \gamma Q(s', \arg\max_a Q(s', a, \theta_2), \theta_1) \qquad (3)$$

1.3 Decision network

Trong DroidRL, **mạng quyết định là bộ não của tác nhân**. Khi việc khai thác được thực hiện, **tác nhân sẽ đặt trạng thái hiện tại được biểu thị bằng vectơ vào mạng quyết**

định của DroidRL, sau đó mạng quyết định sẽ trả về hướng dẫn hành động tiếp theo cho tác nhân

Điều đáng nói là **độ dài trạng thái của tác nhân đang tiếp tục tăng lên**. Nó làm cho **đầu vào** của mạng quyết định có **độ dài không cố định**. Mạng quyết định của DroidRL **cần được thiết kế đặc biệt** vì **đầu vào** của **mạng thần kinh** thường **có hình dạng không đổi**. **RNN là một loại mạng lưới thần kinh được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên. Do độ dài không cố định của các ngôn ngữ tự nhiên, các mạng giống RNN được thiết kế để chấp nhận đầu vào có độ dài không xác định.** Vì lý do này, **mạng quyết định của DroidRL áp dụng RNN và các biến thể của nó.**

DroidRL cũng áp dụng một số thủ thuật có thể giúp ích cho hiệu quả đào tạo và khả năng lựa chọn đặc trưng.

(1) Nhúng từ (**word embedding**): **Đầu vào của mạng quyết định** của tác giả là **một chuỗi các đặc trưng được chọn.** Thay vì được trình bày dưới dạng one-hot vector, **việc nhúng từ được áp dụng để xử lý đầu vào**. **Nếu one-hot vector được sử dụng để biểu diễn chuỗi các đặc trưng** thì **toàn bộ ma trận đầu vào sẽ lớn và thưa thớt** và sẽ **dẫn đến một lượng tính toán và lưu trữ khổng lồ**. Ngoài ra, **không có thông tin ngữ nghĩa khi các đặc trưng được biểu thị bằng one-hot vector**, điều này **không có lợi cho mạng quyết định tìm ra mối tương quan giữa các đặc trưng. Việc áp dụng đặc trưng nhúng từ vào mạng quyết định của DroidRL có thể cải thiện khuôn khổ ở hai khía cạnh sau:**

1. **Nén một one-hot vector** thành **một vectơ dày đặc hơn** (**dense one**). **Word embedding** làm **giảm đáng kể kích thước đầu vào và cải thiện tốc độ đào tạo của mô hình.**

2. **Các vectơ đặc trưng được nén có nhiều ngữ nghĩa hơn.** Nhiệm vụ chính của DroidRL là chọn một tập hợp con các đặc trưng tối ưu. **Sau khi đặc trưng nhúng từ được thêm vào mạng quyết định, DroidRL có thể phân cụm các đặc trưng trong không gian nhiều chiều theo ngữ nghĩa của chúng.** Sau đó, nó **có thể tìm thấy tốt hơn các đặc trưng có thể kết hợp với các đặc trưng được chọn hiện tại**.

(2) **Thứ tự đặc trưng (features ordering)**: **Có sự cân nhắc đặc biệt** đối với việc **áp dụng các phương pháp xử lý ngôn ngữ tự nhiên** cho **việc lựa chọn đặc trưng DroidRL. Ngôn ngữ tự nhiên có tính chất tuần tự, nghĩa là việc đặt lại hai từ trong một câu có thể khiến câu trở nên khó hiểu và vô nghĩa.** Tuy nhiên, **trong việc lựa chọn đặc trưng**, việc thay thế bất kỳ hai đặc trưng được chọn nào trong chuỗi đặc trưng sẽ không có bất kỳ ảnh hưởng nào đến việc đưa ra quyết định của mạng quyết định. Trạng thái đầu vào [1,2,3] và trạng thái đầu vào [1,3,2] giống nhau về ý nghĩa vì chúng chứa các đặc trưng giống nhau và sẽ tạo ra cùng một đầu ra trong mạng quyết định. **Trao đổi vị trí của bất kỳ hai đặc**

trưng nào ở trạng thái đầu vào sẽ không ảnh hưởng đến kết quả. Đặc điểm lựa chọn đặc trưng này khác với đặc điểm của ngôn ngữ tự nhiên.

Nếu thuộc tính cụ thể này không được giải quyết, nó có thể có tác động tiêu cực đến việc học mạng quyết định giống như RNN.

Một thủ thuật được áp dụng để giải quyết vấn đề này. Trước khi các đặc trưng được đưa vào mạng quyết định, chúng được sắp xếp theo chỉ mục. Bằng cách này, **cùng một bộ đặc trưng có thể được đảm bảo chỉ tạo ra một đầu vào bất kể thứ tự lựa chọn đặc trưng**.

Sau khi áp dụng các thủ thuật trên, **cấu trúc mạng quyết định cuối cùng được hiển thị trong Hình 2, tác nhân đưa trạng thái của nó, một chuỗi đại diện cho các đặc trưng đã chọn, vào mạng quyết định. Lớp đầu tiên của mạng quyết định là lớp nhúng. Các đặc trưng được biểu thị bằng one-hot vector đi qua lớp nhúng và trở thành dense vector. Các vectơ này sau đó được đưa vào mạng giống RNN và cuối cùng đi vào lớp fully connected và lớp softmax.**

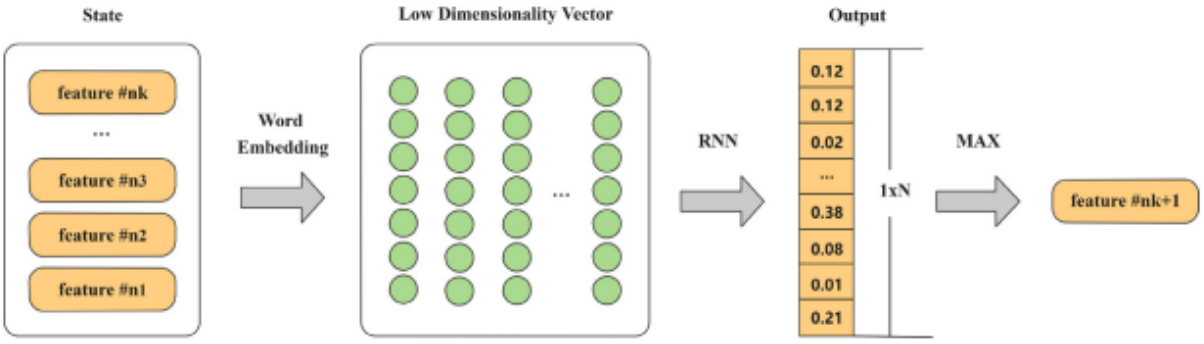

Fig. 2. DroidRL's Decision Network. The decision network takes in the agent's current state and predicts a new feature as action.

2. Training phase

Quá trình đào tạo DroidRL được trình bày chi tiết trong phần này. Thuật toán huấn luyện và thuật toán đánh giá được minh họa trong **Thuật toán 1**.

Algorithm 1: Training and evaluation process

Data: length of feature dictionary N, number of features to be selected F, total training episode E, replay memory warm-up episodes M, initial replay memory rmp, initial network weights θ_1, θ_2, initial probability of exploration ε

Result: Optimal feature subset and its accuracy

```
initialization;
for m ← 1 to M do
    sample some random states and feed into rmp;
end
for episode ← 1 to E do
    state ← an empty arrayfor f ← 1 to F do
        previous state ← state;with probability ε random
        choose a feature and put it into the state;
        or input the state to Decision network and get an N
        dimension vector, find the index of the max value in
        the vector and put it into the state;
        calculate reward of the state;
        put reward, previous state, state into rpm;
    end
    if episode mod Network_Learn_Frequency == 0 then
        sampling some samples from rpm and update θ1;
    end
    if episode mod Sync_Frequency == 0 then
        sampling some samples from rpm and learning;
        Synchronize θ1, θ2;
    end
    update ε by equation (1);
end
start evaluating;
optimal state ← an empty array;
for f ← 1 to F do
    input the current to decision network and get an N
    dimension vector, find the index of the max value in the
    vector and put it into the optimal state;
end
recalculate final_reward of the optimal state;
return optimal state, final_reward;
```

Để khắc phục các vấn đề về dữ liệu tương quan và phân phối dữ liệu huấn luyện không cố định, bộ nhớ phát lại (**relay memory**) được áp dụng trong DroidRL. **Trước khi quá trình đào tạo bắt đầu, DroidRL nhận được một số lấy mẫu ban đầu bằng cách chạy các tập khởi động và đưa chúng vào bộ nhớ phát lại**. Khi bắt đầu mỗi giai đoạn đào tạo, **trạng thái của tác nhân sẽ bị xóa và sau đó tác nhân bắt đầu chọn các đặc trưng**. **Mỗi tập kết thúc sau khi đã chọn đủ số lượng đặc trưng**. Trong các giai đoạn huấn luyện, **một chiến lược cân bằng giữa thăm dò và khai thác sẽ được áp dụng.** Như được mô tả **trong phương trình. (1), tác nhân có xác suất thăm dò rất cao ngay từ đầu nhưng khả năng khai thác càng cao khi số episode tăng lên.**

Trong trường hợp thăm dò, tác nhân chọn ngẫu nhiên một đặc trưng (không ở trạng thái của nó) trong một hành động. Việc thăm dò cho phép tác nhân thử nhiều cách kết hợp đặc trưng và không gian lựa chọn khả thi hơn.

Khi chính sách khai thác được thực thi, tác nhân sẽ sử dụng kinh nghiệm trước đó để chọn đặc trưng tối ưu. Thay vì chọn ngẫu nhiên một đặc trưng, tác nhân đưa trạng thái hiện tại của nó vào mạng quyết định và nhận một vectơ có cùng độ dài với từ điển đặc trưng biểu thị độ tin cậy của từng đặc trưng. Tác nhân lấy đặc trưng có độ tin cậy cao nhất làm hành động. Nếu **đặc trưng có độ tin cậy cao nhất đã được chọn thì tác nhân sẽ chọn đặc trưng có độ tin cậy cao thứ hai**, v.v. Như đã đề cập trước đó, **sau mỗi lần một đặc trưng mới được thêm vào trạng thái, trạng thái sẽ được sắp xếp lại để đảm bảo tính nhất quán của cách biểu diễn bộ đặc trưng**. Sau mỗi hành động được thực hiện, **các đặc trưng trong trạng thái sẽ được sử dụng để phân loại phần mềm độc hại và lành tính. Độ chính xác của việc phân loại khi reward kết hợp với trạng thái trước đó và trạng thái hiện tại sau khi thực hiện hành động sẽ được đưa vào bộ nhớ phát lại.** Tác nhân **tiếp tục khai thác, khám phá (exploit-explore) cho đến khi chọn đủ đặc trưng.**

Sau khi đào tạo tất cả các giai đoạn đào tạo, tác nhân cuối cùng sẽ chạy một giai đoạn đánh giá. Trong epispode này, **mỗi bước của tác nhân sẽ sử dụng những gì đã học được trong giai đoạn đào tạo. Đầu ra của episode này là tập hợp con đặc trưng tối ưu cuối cùng.**

3. Experiment setup

Phần này cung cấp thông tin về môi trường phần cứng để đào tạo DroidRL, tập dữ liệu của tác giả và cài đặt siêu tham số.

3.1. Training environment

Tác giả đã thực hiện tất cả các thử nghiệm trên **server với GPU Tesla V100 đơn và CPU có hai lõi. GPU được sử dụng để tăng tốc quá trình đào tạo mạng quyết định trong DroidRL, trong khi việc đào tạo và dự đoán các bộ phân loại trong DroidRL chỉ sử dụng CPU**. Sau quá trình đào tạo, bộ phân loại của DroidRL có thể được trích xuất riêng biệt để thử nghiệm hoặc triển khai trên bất kỳ phần cứng nào có khả năng chạy thuật toán học máy.

3.2. Dataset

Tập dữ liệu của DroidRL chứa **5000 benign sample** từ **Andro Zoo** và **5560 malware** từ **Drebin** để huấn luyện và kiểm tra mô hình.

Cả hai nguồn dữ liệu đều được sử dụng rộng rãi trong nghiên cứu những năm gần đây tập trung vào phát hiện phần mềm độc hại trên Android, giúp việc thực hiện các thử nghiệm so sánh với các phương pháp lựa chọn đặc trưng khác trở nên dễ dàng hơn. **AndroZoo (Allix et al., 2016b) cập nhật bộ sưu tập 16.000 nghìn APK khác nhau từ nhiều nguồn bao gồm cả Google Play, với mỗi ứng dụng được các sản phẩm AntiVirus khác nhau phân tích để gắn nhãn malware. Các mẫu malware trong nghiên cứu này chủ yếu được chọn từ Drebin (Arp et al., 2014b), một bộ dữ liệu được sử dụng phổ biến có chứa 5560 ứng dụng từ 179 nhóm malware khác nhau.**

Phân tích tĩnh được áp dụng trong công việc này, **trích xuất các quyền, hành động có ý định và mã hoạt động dưới dạng các đặc trưng thô từ các mẫu Android được APKtool và Androguard dịch ngược để củng cố thêm lựa chọn đặc trưng** dựa trên học tập.

Trong tổng số 457 quyền và 126 hành động có mục đích thường được coi là có liên quan cao đến hành vi độc hại của các ứng dụng Android, được chọn trong bài viết này để xây dựng bộ đặc trưng ban đầu. Các quyền cho biết ứng dụng cần truy cập dữ liệu nhạy cảm nào của người dùng (ví dụ: **danh bạ và SMS**), điều này rất cần thiết trong việc phát hiện malware trên Android. **Hành động có ý định là các đối tượng trừu tượng chứa thông tin về hoạt động sẽ được thực hiện cho một thành phần ứng dụng.**

Sau khi phân tách **class.dex để tạo ra các tệp smalis**, mã byte Dalvik (ví dụ: gọi-trực tiếp invoke-direct) thu được thông qua việc quét trường phương thức trong các tệp smalis với biểu thức chính quy. **Opcode có được bằng cách ánh xạ mã byte Dalvik thành một chuỗi các chữ cái như được mô tả trong Bảng 2.**

Table 2
Dalvik Instruction Transformation Table.

Letter	Dalvik Instruction
M	move, move/from16, move/16, move-wide, move-wide/from16, move-result, move-wide/16, move-object, move-object/from16, move-object/16 ...
R	return-void, return, return-wide, return-object
G	goto, goto/16, goto/32
I	if-eq, if-ne, if-lt, if-ge, if-gt, if-le, if-eqz, if-nez, if-ltz, if-gez, if-gtz, if-lez
T	aget, aget-wide, aget-object, aget-boolean, aget-byte, aget-char, aget-short, iget, iget-wide, iget-object, iget-boolean, iget-byte, iget-char ...
P	aput, aput-wide, aput-object, aput-boolean, aput-byte, aput-char, aput-short, iput, iput-wide, iput-object, iput-boolean, iput-byte, iput-char...
V	invoke-virtual, invoke-super, invoke-direct, invoke-static, invoke-interface, invoke-virtual/range, invoke-super/range, invoke-direct/range...

Các đặc trưng của Opcode được phân đoạn theo **N-gram để thu được trình tự chuyển đổi (tranformation sequence)**. Phương pháp **giảm kích thước** được sử dụng **để giải quyết vấn đề về số chiều lớn của vectơ đặc trưng do sự gia tăng số lượng N-gram với giá trị N**. Thứ nhất, tập N-gram gồm các mẫu độc hại là được trích xuất bằng quy trình trích xuất N-gram được đề xuất trong (ZHANG Zong-mei, 2019). **k N-gram tần số cao hàng đầu được chọn**, biểu diễn dưới dạng **Set = {x1, x2, x3, ⋯ , xk}.** Sau đó, **vectơ đặc trưng nhị phân k chiều feature = [m1, m2, m3, ⋯ , mk]** được xây dựng cho mẫu dựa trên tập đặc điểm này, trong đó mi là ”1” chỉ ra rằng tập N-gram của mẫu chứa phần tử xi trong tập đặc trưng.

Từ cuộc thảo luận ở trên, toàn bộ quá trình phát hiện malware Android bằng DroidRL để lựa chọn đặc trưng được mô tả trong Hình 3. Các mẫu ứng dụng Android được thu thập từ bộ dữ liệu Drebin và Andro Zoo, **được giải mã bằng các công cụ kỹ thuật đảo ngược của Android để trích xuất các quyền, hành động có mục đích và N-gram là các đặc trưng ban đầu**. Sau đó, lựa chọn đặc trưng DroidRL được áp dụng để chọn các tập hợp con đặc trưng hợp lệ từ các đặc trưng ban đầu. **Tập hợp con đặc trưng hợp lệ được lưu và sử dụng để xác thực hiệu suất của trình phân loại malware**, sử dụng số lượng đặc trưng và độ chính xác **làm chỉ mục đánh giá**. **Xác thực chéo 10 lần** được sử dụng trong thử nghiệm để đánh giá các mô hình và overfitting. Các

đặc trưng được chọn của DroidRL sẽ được sử dụng để huấn luyện bộ phân loại cuối cùng nhằm phát hiện malware.

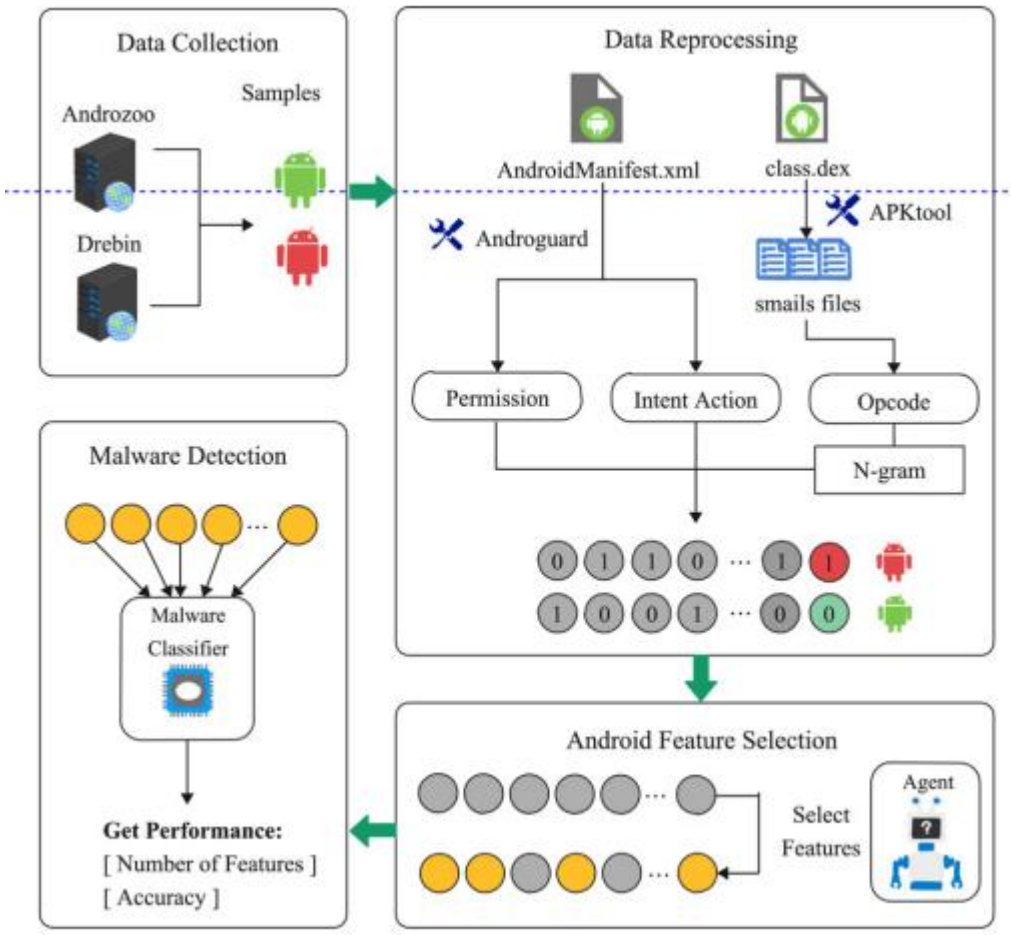

Fig. 3. Overview of DroidRL.

<Trình bày kiến trúc, thành phần đã thực hiện (nội dung mà nhóm đã thực hiện)>

- Thu thập bộ dữ liệu bao gồm **1514 mẫu lành tính và 1054 mẫu độc hại** từ Index of /CICDataset (CICAndMall2017 và CICAndMall2020), nhóm em chỉ lấy một số cái có thể tải được và phù hợp với tài nguyên hiện có.
- Thực thiện trích xuất đặc trưng thô từ các tệp APK, kết quả cho ra tệp JSON.

```
: !python3 droidlysis/droidlysis3.py --input test/benign --output ./test/output --config droidlysis/conf/general.conf

Processing file: test/benign/air.beingfashiondesigner.apk ...
Launching unzip process on test/benign/air.beingfashiondesigner.apk with output dir=./test/output/air.beingfashiondesi
gner.apk-7ce60a541c7d73c30181e4840b58dc348d2aa6baddcdba0e124e91dd1ed2f1f4/unzipped
caution: filename not matched:  classes2.dex
============ Report ============
Sanitized basename  : air.beingfashiondesigner.apk
SHA256              : 7ce60a541c7d73c30181e4840b58dc348d2aa6baddcdba0e124e91dd1ed2f1f4
File size           : 19673487 bytes
Is small            : False
Nb of classes       : 7830
Nb of dirs          : 264

Certificate properties
algo                : None
serialno            : None
country             : 500
owner               : None
timestamp           : (2016, 6, 6, 11, 21, 6)
year                : 2016

Manifest properties
activities          : ["'.AppEntry'", "'com.chartboost.sdk.CBImpressionActivity'", "'com.google.android.gms.ads.AdActi
vity'"]
main_activity       : .AppEntry
package_name        : air.beingfashiondesigner
permissions         : ['INTERNET', 'READ_PHONE_STATE', 'ACCESS_NETWORK_STATE', 'DISABLE_KEYGUARD', 'WAKE_LOCK', 'ACCES
S_FINE_LOCATION', 'ACCESS_COARSE_LOCATION', 'WRITE_EXTERNAL_STORAGE']
swf                 : True
```

- Viết mã trích xuất đặc trưng từ tệp JSON đã có theo chuẩn nhất định.

```python
def get_static_feature(data):
    values = dict()
    for x in data.keys():
        if x == "sanitized_basename":
            continue
        if isinstance(data[x], dict):
            for key in data[x].keys():
                if isinstance(data[x][key], list):
                    #print(x, key)
                    if key == "permissions":
                        for perm in data[x][key]:
                            values[f'permissions_{perm.split(".")[-1]}'] = 1

                    values[f'{x}_{key}_len']  = len(data[x][key])
                    continue
                if key in ['main_activity', 'package_name', 'app_name']:
                    continue
                try:
                    values[f'{x}_{key}'] = int(data[x][key])
                except:
                    values[f'{x}_{key}'] = -1
        else:
            values[x] = int(data[x])
    return values
```

- Viết mã lấy dữ liệu từ các mẫu lành tính và các mẫu độc hại và thực hiện gán nhãn cho các mẫu (lành tính là 0, độc hại là 1).

```
import os, json
import pandas as pd

def get_dataset(folder, lable):
    dataset = dict()
    for filename in os.listdir(folder):
        dataset[filename] = get_static_feature(json.load(open(f'{folder}/{filename}')))
    df=pd.DataFrame(dataset).T
    df.fillna(value=0, inplace=True)
    df['class'] = lable
    return df
```

```
df_ben = get_dataset("malware/malware_json", 1)
df_mal = get_dataset("benign/benign_json", 0)
```

```
df_ben.shape
```

```
(1514, 1256)
```

```
df_mal.shape
```

```
(1054, 1383)
```

- Chuyển các dữ liệu đã có thành dataframe và ghép lại thành một bộ dự liệu để huấn luyện và kiểm tra.

```
# Kết hợp hai DataFrame
df_combined = pd.concat([df_ben, df_mal], ignore_index=True)
df_combined.fillna(value=0, inplace=True)

# Trộn ngẫu nhiên các hàng
HinT = df_combined.sample(frac=1).reset_index(drop=True)

HinT.head(5)
```

	file_nb_classes	file_nb_dir	file_size	file_small	filetype	file_innerzips	manifest_properties_activities_len	manifest_properties_lib
0	41.0	6.0	1612197.0	0.0	1.0	0.0	8.0	
1	0.0	0.0	836367.0	0.0	1.0	0.0	11.0	
2	465.0	30.0	561174.0	0.0	1.0	0.0	17.0	
3	178.0	29.0	172599.0	0.0	1.0	0.0	2.0	
4	4691.0	138.0	7885466.0	0.0	1.0	0.0	9.0	

5 rows × 1436 columns

- Tiến hành huấn luyện và đánh giá trên 4 mô hình là **RandomForestClassifier, LGBMClassifier, ExtraTreesClassifier, DecisionTreeClassifier** cho toàn bộ tất cả các feature trích xuất được (1435 đặc trưng). **Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time**.

```
import time
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from lightgbm import LGBMClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
```

```
# Initialize models
rf_model = RandomForestClassifier(random_state=42)
lgbm_model = LGBMClassifier(random_state=42)
et_model = ExtraTreesClassifier(random_state=42)
dt_model = DecisionTreeClassifier(random_state=42)
```

```
def evaluate_model(model, X_train, y_train, X_test, y_test):
    start_time = time.time()
    model.fit(X_train, y_train)
    end_time = time.time()

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])

    return accuracy, precision, recall, f1, roc_auc, end_time - start_time
```

```
# Create a dictionary to store results
results = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1': [], 'ROC-AUC': [], 'Time': []}
```

```
# Loop through each model
for model, model_name in zip([rf_model, lgbm_model, et_model, dt_model],
                             ['Random Forest', 'LightGBM', 'Extra Trees', 'Decision Tree']):

    # Evaluate model performance and time
    accuracy, precision, recall, f1, roc_auc, run_time = evaluate_model(model, X_train, y_train, X_test, y_test)

    # Store results in the dictionary
    results['Model'].append(model_name)
    results['Accuracy'].append(round(accuracy, 4))
    results['Precision'].append(round(precision, 4))
    results['Recall'].append(round(recall, 4))
    results['F1'].append(round(f1, 4))
    results['ROC-AUC'].append(round(roc_auc, 4))
    results['Time'].append(round(run_time, 4))
```

```
# Create a performance evaluation/comparison table
results_df = pd.DataFrame(results)
print(results_df)
```

	Model	Accuracy	Precision	Recall	F1	ROC-AUC	Time
0	Random Forest	0.9922	0.9934	0.9934	0.9934	0.9998	0.2075
1	LightGBM	0.9981	1.0000	0.9967	0.9983	1.0000	0.2264
2	Extra Trees	0.9961	0.9967	0.9967	0.9967	0.9999	0.2986
3	Decision Tree	0.9883	0.9934	0.9868	0.9901	0.9887	0.0296

- Thực hiện phương pháp lựa chọn đặc trưng (lựa chọn được **323/1435** đặc trưng)

```
def evaluate_one_feature(feature, index='', metric=roc_auc_score):
    rootnode = DecisionTreeClassifier(max_depth=1, criterion='gini')
    rootnode.fit(X_train[feature].array.reshape(-1,1), y_train)
    preds = rootnode.predict(X_test[feature].array.reshape(-1,1))
    preds_tr = rootnode.predict(X_train[feature].array.reshape(-1,1))
    met = round(metric(y_test, preds), 4)
    if met > 0.5:
        return [feature, met, rootnode, preds, preds_tr]
    else:
        return [feature, met, None, [], []]
```

```
from joblib import Parallel, delayed
import multiprocessing

# Sử dụng joblib để chạy hàm evaluate_one_feature song song trên các đặc trưng
def parallel(f, items, n_workers=None, threadpool=False, progress=True):
    if n_workers is None:
        n_workers = multiprocessing.cpu_count()

    if threadpool:
        from multiprocessing.pool import ThreadPool as Pool
    else:
        from multiprocessing import Pool

    with Pool(n_workers) as pool:
        results = list(pool.imap(f, items))

    return results

# "conts` là danh sách các đặc trưng muốn đánh giá
results = parallel(f=evaluate_one_feature,
                   items=conts, n_workers=multiprocessing.cpu_count(), threadpool=False, progress=True)
```

```
    useful_features = result_df.loc[result_df['roc_auc_score'] > 0.5]
    print(f"{len(useful_features)} / {len(conts)} features have direct separating power (linear)")
```

```
323 / 1435 features have direct separating power (linear)
```

- Tiến hành huấn luyện và đánh giá lại 4 mô hình đã thiết lập dựa trên bộ dữ liệu đã được tiến hành phương pháp chọn đặc trưng. **Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time**.

Create a new data ¶

```
# Create new dataset with selected features
selected_features = useful_features['feature'].tolist()
X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]
```

```
# Optionally print the results
import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)
```

```
           Model  Accuracy  Precision  Recall      F1  ROC-AUC    Time
0  Random Forest    0.9202     1.0000  0.8647  0.9274   0.9999  0.1777
1       LightGBM    0.9942     1.0000  0.9901  0.9950   1.0000  0.1028
2    Extra Trees    0.9319     1.0000  0.8845  0.9387   1.0000  0.1363
3  Decision Tree    0.9864     0.9901  0.9868  0.9884   0.9863  0.0187
```

```
# Lưu tên các đặc trưng sử dụng trong quá trình huấn luyện
feature_names = X_train_selected.columns.tolist()
joblib.dump(feature_names, 'feature_names.joblib')
```

```
['feature_names.joblib']
```

```
len(feature_names)
```

```
323
```

- Kết quả cho thấy, sau khi thực hiện phương pháp lựa chọn đặc trưng, thì thời gian đã giảm đi, độ chính xác tuy giảm nhưng mức độ tuỳ thuộc vào thuật toán sử dụng để phân loại. Như mô hình LGBM cho thấy tất cả các chỉ số đánh giá đều không chênh lệnh nhiều so với lúc chưa giảm số lượng đặc trưng.

B. Chi tiết cài đặt, hiện thực

<cách cài đặt, lập trình trên máy tính, cấu hình máy tính sử dụng, chuẩn bị dữ liệu, v.v>

- Cài đặt tool để phân tích tệp APK.
- Tải các dữ liệu về phần mềm lành tính và độc hại.
- Môi trường thực hiện:

```
(base) jupyter-iec-hahien@bmtt-server:~/Hien$ nvidia-smi
Wed Jun 12 02:21:26 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.67                 Driver Version: 550.67         CUDA Version: 12.4     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4080 ...    Off |   00000000:01:00.0 Off |                  N/A |
| 47%   61C    P2            223W /  320W |   14636MiB /  16376MiB |     90%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|    0   N/A  N/A      7985      C   python                                    10632MiB |
|    0   N/A  N/A     67686      C   /opt/tljh/user/bin/python                  3994MiB |
+---------------------------------------------------------------------------------------+
```

C. Kết quả thực nghiệm

<mô tả hình ảnh về thực nghiệm, bảng biểu số liệu thống kê từ thực nghiệm, nhận xét về kết quả thu được.>

- Tiến hành huấn luyện và đánh giá trên 4 mô hình là **RandomForestClassifier, LGBMClassifier, ExtraTreesClassifier, DecisionTreeClassifier** cho toàn bộ tất cả các feature trích xuất được (1435 đặc trưng). **Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time**.

```python
import time
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from lightgbm import LGBMClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
```

```python
# Initialize models
rf_model = RandomForestClassifier(random_state=42)
lgbm_model = LGBMClassifier(random_state=42)
et_model = ExtraTreesClassifier(random_state=42)
dt_model = DecisionTreeClassifier(random_state=42)
```

```python
def evaluate_model(model, X_train, y_train, X_test, y_test):
    start_time = time.time()
    model.fit(X_train, y_train)
    end_time = time.time()

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])

    return accuracy, precision, recall, f1, roc_auc, end_time - start_time
```

```python
# Create a dictionary to store results
results = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1': [], 'ROC-AUC': [], 'Time': []}
```

```python
# Loop through each model
for model, model_name in zip([rf_model, lgbm_model, et_model, dt_model],
                             ['Random Forest', 'LightGBM', 'Extra Trees', 'Decision Tree']):

    # Evaluate model performance and time
    accuracy, precision, recall, f1, roc_auc, run_time = evaluate_model(model, X_train, y_train, X_test, y_test)

    # Store results in the dictionary
    results['Model'].append(model_name)
    results['Accuracy'].append(round(accuracy, 4))
    results['Precision'].append(round(precision, 4))
    results['Recall'].append(round(recall, 4))
    results['F1'].append(round(f1, 4))
    results['ROC-AUC'].append(round(roc_auc, 4))
    results['Time'].append(round(run_time, 4))
```

```python
# Create a performance evaluation/comparison table
results_df = pd.DataFrame(results)
print(results_df)
```

```
           Model  Accuracy  Precision  Recall      F1  ROC-AUC    Time
0  Random Forest    0.9922     0.9934  0.9934  0.9934   0.9998  0.2075
1       LightGBM    0.9981     1.0000  0.9967  0.9983   1.0000  0.2264
2    Extra Trees    0.9961     0.9967  0.9967  0.9967   0.9999  0.2986
3  Decision Tree    0.9883     0.9934  0.9868  0.9901   0.9887  0.0296
```

- Thực hiện phương pháp lựa chọn đặc trưng (lựa chọn được **323/1435** đặc trưng)

```python
def evaluate_one_feature(feature, index='', metric=roc_auc_score):
    rootnode = DecisionTreeClassifier(max_depth=1, criterion='gini')
    rootnode.fit(X_train[feature].array.reshape(-1,1), y_train)
    preds = rootnode.predict(X_test[feature].array.reshape(-1,1))
    preds_tr = rootnode.predict(X_train[feature].array.reshape(-1,1))
    met = round(metric(y_test, preds), 4)
    if met > 0.5:
        return [feature, met, rootnode, preds, preds_tr]
    else:
        return [feature, met, None, [], []]
```

```
from joblib import Parallel, delayed
import multiprocessing

# Sử dụng joblib để chạy hàm evaluate_one_feature song song trên các đặc trưng
def parallel(f, items, n_workers=None, threadpool=False, progress=True):
    if n_workers is None:
        n_workers = multiprocessing.cpu_count()

    if threadpool:
        from multiprocessing.pool import ThreadPool as Pool
    else:
        from multiprocessing import Pool

    with Pool(n_workers) as pool:
        results = list(pool.imap(f, items))

    return results

# "conts` là danh sách các đặc trưng muốn đánh giá
results = parallel(f=evaluate_one_feature,
                   items=conts, n_workers=multiprocessing.cpu_count(), threadpool=False, progress=True)
```

```
useful_features = result_df.loc[result_df['roc_auc_score'] > 0.5]
print(f"{len(useful_features)} / {len(conts)} features have direct separating power (linear)")
```

```
323 / 1435 features have direct separating power (linear)
```

- Tiến hành huấn luyện và đánh giá lại 4 mô hình đã thiết lập dựa trên bộ dữ liệu đã được tiến hành phương pháp chọn đặc trưng. **Đánh giá về Accuracy, Precision, Recall, F1, ROC-AUC, Time**.

Create a new data ¶

```
# Create new dataset with selected features
selected_features = useful_features['feature'].tolist()
X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]
```

```
# Optionally print the results
import pandas as pd
results_df = pd.DataFrame(results)
print(results_df)
```

```
           Model  Accuracy  Precision  Recall      F1  ROC-AUC    Time
0  Random Forest    0.9202     1.0000  0.8647  0.9274   0.9999  0.1777
1       LightGBM    0.9942     1.0000  0.9901  0.9950   1.0000  0.1028
2    Extra Trees    0.9319     1.0000  0.8845  0.9387   1.0000  0.1363
3  Decision Tree    0.9864     0.9901  0.9868  0.9884   0.9863  0.0187
```

```
# Lưu tên các đặc trưng sử dụng trong quá trình huấn luyện
feature_names = X_train_selected.columns.tolist()
joblib.dump(feature_names, 'feature_names.joblib')
```

```
['feature_names.joblib']
```

```
len(feature_names)
```

```
323
```

⇨ Kết quả cho thấy, sau khi thực hiện phương pháp lựa chọn đặc trưng, thì thời gian đã giảm đi, độ chính xác tuy giảm nhưng mức độ tuỳ thuộc vào thuật toán sử dụng để phân loại. Như mô hình LGBM cho thấy tất cả các chỉ số đánh giá đều không chênh lệnh nhiều so với lúc chưa giảm số lượng đặc trưng.

D. Hướng phát triển

<Nêu hướng phát triển tiềm năng của đề tài này trong tương lai. Nhận xét về tính ứng dụng của đề tài>.

- Thu thập bộ dữ liệu nhiều mẫu hơn nữa để đánh giá.
- Sử dụng các phương pháp lựa chọn đặc trưng tối ưu hơn để cải thiện hiệu suất mô hình cũng như là giảm chi phí tính toán và thời gian hơn.
- Sử dụng mô hình deep learning thay vì traditional learning để phân loại, so sánh xem có vượt trội hơn không.

Sinh viên báo cáo các nội dung mà nhóm đã thực hiện, có thể là 1 phần hoặc toàn bộ nội dung của bài báo. Nếu nội dung thực hiện có khác biệt với bài báo (như cấu hình, tập dữ liệu, kết quả,...), sinh viên cần chỉ rõ thêm khác biệt đó và nguyên nhân.

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- Đặt tên theo định dạng: [Mã lớp]-Project_Final_NhomX_Madetai. (trong đó X và Madetai là mã số thứ tự nhóm và Mã đề tài trong danh sách đăng ký nhóm đồ án).

  Ví dụ: [NT521.N11.ANTT]-Project_Final_Nhom03_CK01.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT