Assignment #2 Malware Classification

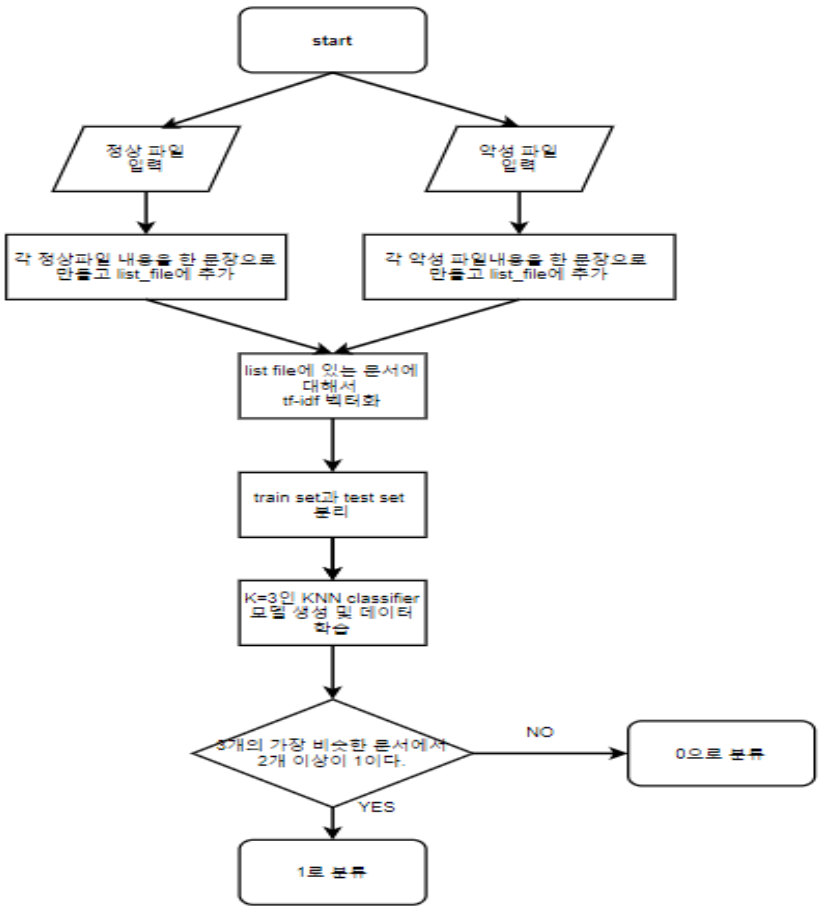
컴퓨터 소프트웨어 학부 2018008059 김은수

모델링 소개

- Classification 기법 사용
  - KNN classifier 사용
    - 입력벡터로 tf-idf를 사용

모델링 요약: KNN classifier를 사용해서 test 문서에 대해 가장 가까운 training data K개를 뽑고 그 data의 label에 따라서 test 문서의 label을 지정해줄 것이다. 문서의 가까운 정도를 알아내기 위해 tf-idf를 사용할 것이고 이를 위해서 sklearn의 TfidfVectorizer 모듈을 사용한다. KNN classification을 위해서 sklearn의 KNeighborsClassifier 모듈을 사용한다.

순서도:



프로그램 및 모델링 설명

- **파일 입력**

Malware와 정상파일에서 사용되는 API의 차이를 이용해서 malware와 정상파일을 분류할 것이다. 이를 위해 각 문서마다 단어에 대한 Tf-idf값을 갖는 벡터를 만들고 사용할 것이다. 따라서 scikit-learn의 *"TfidfVectorizer 모듈"*을 사용할 것이다.

KNN classification 모델의 입력 값으로 Tf-idf 값을 갖는 벡터를 사용할 것이다. Tf-idf는 여러 문서로 이루어진 문서군이 있을 때 어떤 단어가 특정 문서 내에서 얼마나 중요한 것인지를 나타내는 통계적 수치이다. Malware와 정상파일에서 사용되는 API의 차이를 이용해서 malware와 정상파일을 분류할 것이므로 Tf-idf가 적절하다 생각하여 사용하였다. 또한, 단지 단어의 빈도수를 기반으로 하지 않고, malware와 정상파일을 분류하기에 더 적절한 의미 있는 단어에 높은 수치를 부여하기 위해 tf-idf를 사용했다.

Tf-idf 값을 갖는 벡터를 만들기 위해서 scikit-learn의 TfidVectorizer를 사용할 것이다. 이를 위해서는 입력 값이 텍스트로 이뤄진 데이터 형태여야 한다. 따라서 각 파일(문서)에 있는 API를 모두 하나의 문장으로 만들었고 각 API는 띄어쓰기로 구분했다.

```
# get dataset; 0 is normality, 1 is malicious code
files0 = glob.glob("0/*.txt")
files1 = glob.glob("1/*.txt")
list_file = []
number0 = 0
number1 = 0

for file in files0:
    with open(file, 'r') as f:
        line = f.read().replace('\n',' ')
        number0 +=1
        list_file.append(line)

for file in files1:
    with open(file,'r') as f:
        line = f.read().replace('\n',' ')
        number1 +=1
        list_file.append(line)
```

즉, 코드는 다음과 같다. 0에 들어있는 txt 파일을 files0 리스트에 모두 담는다. 마찬가지로 1에 있는 txt 파일을 files1 리스트에 모두 담는다. 다음으로 files0에 있는 모든 file에 대해서 각 파일의 전체 내용을 각 한 문장으로 만들어서 list_file에 담는다. (한 파일 당 한 문장이 만들어진다.) 한 문장으로 만드는 방법은 다음과 같다. 각 file에 대해 줄바꿈을 space로 바꾸면서 read하고 그 내용을 line에 할당한다. 그리고 한 파일을 읽을 때마다 number 변수를 1씩 증가시켜준다. Number0은 0파일, 즉 정상파일의 개수를 나타내고 number1은 1파일, 즉 악성파일의 개수를 나타낸다. 그 다음에 파일마다 만들어진 문장을 list_file array에 넣어준다. 악성코드와 정상파일을 분

리하지 않은 이유는 그래야 전체 API에 대한 각 문서의 유의미한 tf-idf 벡터를 구할 수 있기 때문이다. 따라서 list_file에는 전체 문서가 들어가 있고 한 인덱스마다 한 파일이 들어가 있다.

- **Tf- idf 벡터화**

다음으로 tfidf_vectorizer = TfidfVectorizer()를 통해 Tf-idf 객체를 선언한다. 다음으로 tfidf_vectorizer.fit_transform(list_file)을 통해 list_file 내 단어를 학습시킨다. 이는 다음과 같이 진행된다. list_file에 있는 문서들 별로 단어의 등장 빈도를 구한다. 그리고 이에 대해서 특정 단어가 나타나는 문서 수를 역수 변환해준 뒤 곱해준다. 이를 통해서 각 문서마다 tf-idf 벡터가 나오고 이 벡터의 요소는 단어 별 tf-idf 값이다. 즉, tfidf_vectorizer.fit_transform(list_file)을 통해서 데이터에 대해 tf-idf 벡터화를 진행할 수 있다. 이렇게 각 문서에 단어에 대한 tf-idf vector를 구하고 이를 X라는 변수에 저장한다.

```
tfidf_vectorizer = TfidfVectorizer()
X = tfidf_vectorizer.fit_transform(list_file)
```

또한 label에 대한 list를 만들어줘야 한다. 0에 들어가 있는 파일의 개수를 number0에 저장하였고, 1에 들어가 있는 파일 개수를 number1에 저장했으므로 Y라는 변수를 다음과 같이 해주어서 각 문서에 대한 label을 설정해줬다.

```
Y = [0 for x in range(number0)]
Y.extend([1 for x in range(number1)])
```

즉, number0만큼 0을 Y 리스트에 넣어주고 그 다음에 number1만큼 1을 넣어준다. List_file에 0에 해당하는 파일이 먼저 들어가고 그 다음에 1에 해당하는 파일이 들어가기 때문에 label을 이런 식으로 해주었다.

- **Train set, Test set 분리**

실제로 생성한 분류 모델의 성능을 test해봐야하기 때문에 다음과 같이 학습과 검증 데이터 셋을 분리한다. 모든 데이터에는 분류된 label이 있기 때문에 검증 데이터를 예측 모델에 넣어서 실제 그 레이블을 잘 맞추는 지 보는 게 목적이다. *"Train_test_split함수"*를 이용해서 data, target(label)을 넣어주고 test_size는 0.2로 정해준다. 또한 stratify를 Y로 설정해서 0과 1 class 비율을 train set과 검증 데이터 셋에 유지해준다.

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2, stratify=Y)
```

- **모델 생성 및 데이터 학습**

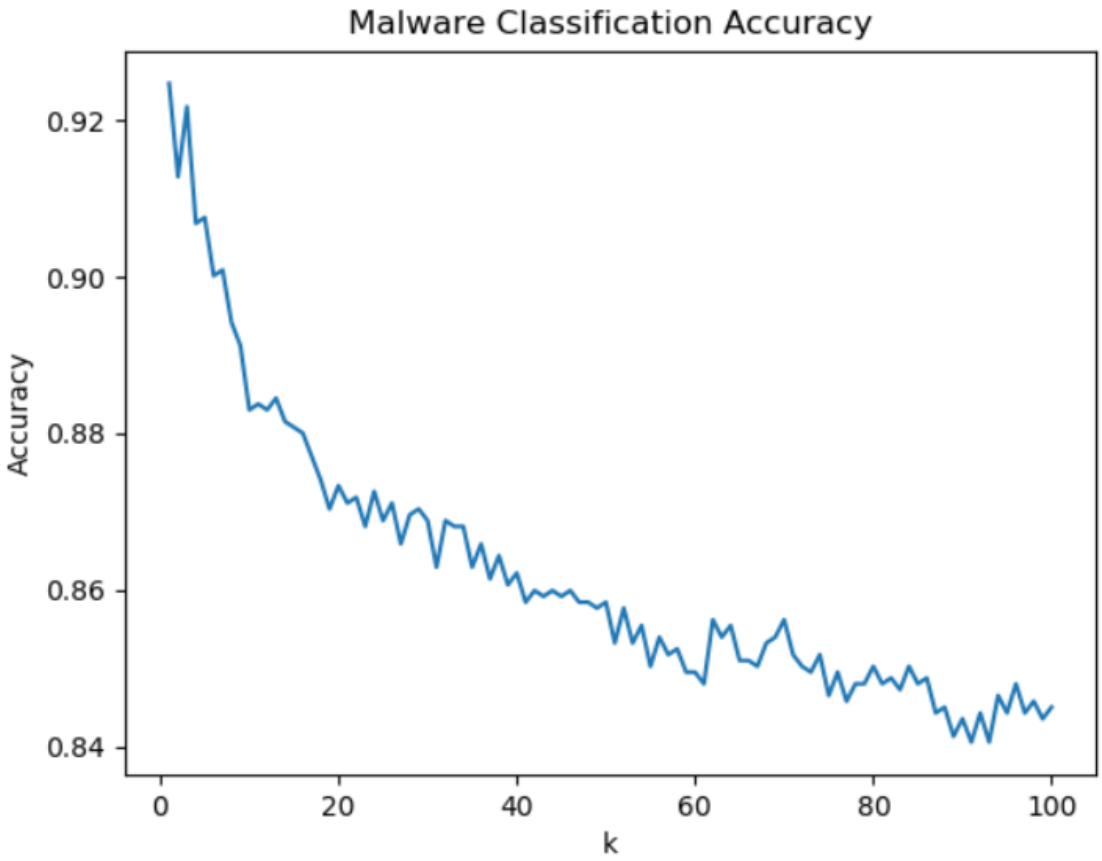
학습데이터를 가지고 모델을 생성한다. 모델은 KNN classifier를 사용한다. KNN은 test 문서가 들어왔을 때 이 test 문서와 가장 가까이에 있는 K개의 training data 중 과반수의 label로 test문서를 labeling해준다. 우선은 K를 3으로 지정해주었다.

```
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train, Y_train)
```

K=3일 때 정확도

```
C:\Users\김은수\PycharmProjects\security2\venv\Scripts\python.exe
 C:/Users/김은수/PycharmProjects/security2/assginment2.py
Accuracy: 0.9217585692995529
```

K=3으로 지정했는데 이 값이 적절한지 알아보기 위해 k값을 1부터 100까지 바꿔가면서 모델 정확도를 구했고 시각화를 해보았다. Matplotlib.pyplot 모듈을 이용하여 구해보니 다음과 같은 그래프가 나왔다.



K가 커질수록 비슷한 문서들로 판단하는 게 아니라 정확도가 떨어짐을 확인할 수 있었다. 따라서 k = 3으로 가정한 것이 적절하다는 판단으로, k=3으로 분류 모델을 생성해주었다.

또한 이 분류 모델로 해당 파일이 악성코드인지 아니면 정상 파일인지 확인할 수 있다.

KNeighborsClassifier(n_neighbor = 3)으로 생성해준 객체의 함수 predict를 이용해서 구할 수 있다.

```
classifier = KNeighborsClassifier(n_neighbors=3)
```

위와 같이 KNeighborsClassifier 객체 classifier를 선언해주었다면 그 객체를 이용해 구하면 된다.

```
test_predicted_0 = int(classifier.predict(X[0]))
print("Is this file malicious code? ")
if test_predicted_0 == 0:
    print("This is normality")
else:
    print("This is malware")
```

객체의 predict함수에 판단하고 싶은 파일의 벡터화 된 값을 넣어준다. 이 예시에서는 X[0]을 넣었다.

```
X = tfidf_vectorizer.fit_transform(list_file)
```

이와 같이 X는 list_file을 tf-idf 벡터화를 수행한 값을 저장한 변수다. 그리고 list_file에는 0폴더에 있는 txt 파일 뒤 1폴더에 있는 txt 파일이 들어갔으므로 X[0]은 0, 즉 정상파일일 것이다. 따라서 classifier.predict(X[0])을 실행한 뒤 나온 값은 만든 KNN 모델의 정확도가 0.9 정도이므로 아마 0일 것이다. 실제로 실행해보면 다음과 같다.

```
Is this file malicious code?
This is normality
```

즉, 정상파일이라고 판단함을 알 수 있다.

따라서 판단하고 싶은 파일을 벡터화 진행한 뒤 모델의 predict 함수 인자로 넣어주면 된다. 제출 코드에서는 해당 부분을 주석 처리하였다. 따라서 실제 코드를 실행시키면 정확도만 출력이 될 것이다. 판단하고 싶은 파일이 있으면 주석을 제거해 사용하면 된다.

사용한 도구 및 모듈

- Glob

  : 0과 1 폴더에 있는 txt 파일을 부르기 위해서 사용

- Matplotlib.pyplot

  : KNN classifier에서 K 값을 선택하기 위해서 K 에 따른 정확도를 시각적으로 표현하기 위해 사용.

- Train_test_split

  : 학습하는 set과 검증하는 set을 분리하기 위해 사용

- TfidfVectorizer

  : 문서 별 단어에 대한 tf-idf값을 구하기 위해 사용

- KNeighborsClassifier

  : 문서를 악성코드인지 정상 파일인지 KNN classification 방법을 이용해 분리하기 위해 사용