**제2회 통신망 안정성 확보를 위한 인공지능 해커톤**

**모델 설명서**

**<분야2: 유선 네트워크 장비의 신속한 조치를 위한 경보 유형 분류>**

**팀 명 : 신사임당**

**2023년 8월 24일**

1. **라이브러리 및 데이터 (Library & Data)**

* 데이터는 Q2\_train, Q2\_test 파일을 pd.read\_csv 로 불러옴
* alarmmsg\_original은 엑셀내부에서 전처리 진행하여 alarmmsg\_df 파일은 pd.read\_excel로 불러옴
* 기본 연산 위해 numpy, pandas 호출
* 메시지 벡터화 위해 from sklearn.feature\_extraction.text import TfidfVectorizer 호출
* 변수 스케일링 위해 from sklearn.preprocessing import MinMaxScaler 호출
* 메시지 벡터화 및 모델 학습을 순차적으로 하기 위해 from sklearn.pipeline import Pipeline 호출
* 메시지 벡터화와 이외의 변수들의 스케일링을 동시에 열 별로 하기 위해 from sklearn.compose import ColumnTransformer 호출
* 여러 개의 모델을 앙상블하여 학습하기 위해 from sklearn.ensemble import VotingClassifier
* 모델 학습위해 from sklearn.naive\_bayes import ComplementNB, from catboost import CatBoostClassifier 호출

import numpy as np

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import MinMaxScaler

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.ensemble import VotingClassifier

from sklearn.naive\_bayes import ComplementNB

from catboost import CatBoostClassifier

train\_df = pd.read\_csv("Q2\_train.csv")

test\_df = pd.read\_csv("Q2\_test.csv")

alarmmsg\_df = pd.read\_excel("alarmmsg\_edit.xlsx")

1. **데이터 전처리 (Data Cleansing & Pre-Processing)**

* **‘slotna’** 변수를 생성하여 slot이 결측치이면 1, 아니면 0을 할당. ticketno별로 max값 사용
* **‘alarmno’**에서 ‘alarmno’의 앞 6자리만 슬라이싱 한 후 ticketno별로 max값 사용
* alarmtime의 type을 datetime으로 바꿔주고 ticketno별로 max\_time과 min\_time을 구한 후, (max\_time – min\_time)으로 전표 별 지속시간인 **‘duration’** 변수 생성
* **‘level3’, ‘level4’, ‘level5’, ‘level7’** 변수를 생성하여 ticketno별로 alarmlevel 3, 4, 5, 7 각각의 개수를 할당
* ‘slot’, ‘port’는 train set과 test set의 분포가 달라 제거하기로 결정
* ‘sva’와 ‘root\_cause\_domain’은 데이터가 단일값이고 ‘unit’ 또한 거의 모든 값이 같은 값이라 제거하기로 결정
* ‘site’와 ‘sysname’은 두 번째 알파벳이 타입에 따른 특징이 있다고 판단하여 추출 및 범주화 후, 학습에 넣어봤지만 성능 저하가 일어나 제거하기로 결정
* alarmmsg\_original 전처리
  + 모든 문자 소문자화
  + 특수문자 및 조사 제거
  + 경보 메시지 설명 참고하여 약어 풀이
  + 같은 의미의 단어들 하나의 단어로 통일

    # slotna 변수 생성

    q2\_train['slotna'] = q2\_train['slot'].isna().astype(int)

    q2\_test['slotna'] = q2\_test['slot'].isna().astype(int)

    # alarmno 자릿수

    for i in range(len(q2\_train)):

        q2\_train['alarmno'][i] = int(str(q2\_train['alarmno'][i])[0:6])

    for i in range(len(q2\_test)):

        q2\_test['alarmno'][i] = int(str(q2\_test['alarmno'][i])[0:6])

    # 필요 없는 변수 drop

    q2\_train.drop(labels = ['slot', 'port', 'sva', 'root\_cause\_domain'], axis = 1, inplace = True)

    q2\_test.drop(labels = ['slot', 'port', 'sva', 'root\_cause\_domain'], axis = 1, inplace = True)

    # alarmtime 변수로 지속시간 변수 추가 및 time변수 삭제

    data = data\_total.groupby('ticketno')['alarmtime'].agg(\*\*{'min\_time':'min', 'max\_time':'max'}).reset\_index()

    data['duration'] = data['max\_time'] - data['min\_time']

    for i in range(len(data['duration'])):

      data['duration'][i] = data['duration'][i].seconds / 60

    # 전표별 level 당 개수 변수 생성

    level = data\_total.groupby(['ticketno','alarmlevel']).size().unstack(fill\_value = 0).reset\_index()

    data['level3'] = level[3]

    data['level4'] = level[4]

    data['level5'] = level[5]

    data['level7'] = level[7]

    data.drop(labels=['max\_time'], axis = 1, inplace = True)

    # 전표별 alarmmsg 이어붙이기

    msg = data\_total.groupby("ticketno")["alarmmsg\_original"].apply(list).reset\_index()

    # 전표별 데이터 합치기

    data = pd.merge(data, msg, how='left')

    data = pd.merge(data, data\_total.groupby('ticketno')['alarmno'].agg(\*\*{'alarmno':max}).reset\_index())

    data = pd.merge(data, data\_total.groupby('ticketno')['train'].agg(\*\*{'train':max}).reset\_index())

    data = pd.merge(data, data\_total.groupby('ticketno')['slotna'].agg(\*\*{'slotna': max}).reset\_index())

    data = pd.merge(data, data\_total.groupby('ticketno')['root\_cause\_type'].agg(\*\*{'type':max}).reset\_index())

    data['alarmmsg\_original'] = data['alarmmsg\_original'].apply(lambda msg\_list: ' '.join(msg\_list))

1. **탐색적 자료 분석 (Exploratory Data Analysis)**

* Tableau를 사용하여 데이터를 시각화 한 후 분석
* 경보 타입 별 개수의 분포를 보았을 때 Linkcut이 확연히 많아 데이터셋이 불균형함을 알 수 있음.
* train데이터와 test데이터의 열별 분포를 비교해 보았을 때 slot과 port 변수가 train, test에서 매우 다른 양상을 보였으며 alarmmsg\_original, unit 또한 매우 다른 것을 볼 수 있어 변수들을 학습에 사용할 때 과적합을 막기 위한 노력이 필요해 보임
* 하나의 ticketno가 여러 행에 걸쳐 나타나는 경우가 많음. 이는 ticketno 별로 경보가 지속되는 시간이 다르다는 것을 의미. 전표 별 지속시간 변수를 추가하는 것이 필요해 보임.
* 데이터의 결측치는 slot, port변수에 있는데 slot변수가 결측치일 경우 경보타입이 PowerFail인 경우가 대부분임. slot이 결측치일 경우 타입이 PowerFail일 확률이 높다고 해석할 수 있고 이를 판별할 수 있는 변수 생성 필요해 보임.
* 경보 타입 별로 alarmno의 분포를 보았을 때 LinkCut에서는 13자리 숫자만 나오고 PowerFail, UnitFail은 9자리, 13자리 숫자로 나온다는 것을 알 수 있었고 데이터를 자세히 보면 앞 6자리가 묶음으로 진행된다는 것을 알 수 있음. 이에 따라 alarmno의 앞 6자리가 분류에 도움을 줄 수 있을 것이라 판단
* alarmlevel의 값으로 3, 4, 5, 7 중 하나가 나오는데 LinkCut일 때는 4가 나오지 않음. 전표별로 각 level 개수가 어떻게 나오는지 파악이 필요해 보임

1. **변수 선택 및 모델 구축 (Feature Engineering & Initial Modeling)**

* 변수선택
* 최종 학습에 사용할 변수로 전표별 지속시간 **‘duration’**, 전표별 level 개수 **‘level3’, ‘level4’, ‘level5’, ‘level7’, ‘alarmmsg\_original’,** 기존 alarmno의 앞 6자리인 **‘alarmno’**, slot변수 결측치 여부 **‘slotna’** 선택.
* ‘alarmmsg\_original’ 변수를 학습에 사용하기 위해 **Tfidf vectorization** 방법을 사용.

Tfidf Vectorizer는 텍스트를 수치화 시켜 주는데 이 때 단어의 빈도수를 기반으로 하나 여러 문서에서 많이 등장하는 단어에는 패널티를 주는 방식

아래는 TfidfVectorizer의 파라미터 설명임

* + analyzer = ‘char’: analyzer는 학습의 단위를 단어(word)로 고려할지 글자(char) 각각으로 고려할지를 결정하는 파라미터인데 train데이터와 test데이터의 알람메시지가 많이 다르기 때문에 글자 각각으로 하는 것이 더 나을 것이라 판단하였고 테스트 셋에 대한 성능 또한 char일 때 더 잘 나와서 ‘char’로 선택
  + sublinear\_tf = True: sublinear\_tf는 TF(단어빈도수)의 스무딩 여부를 결정하는 파라미터이며 과적합을 막기 위해 True로 설정
  + max\_df = 0.8: 문서의 빈도가 지정된 임계값보다 엄격하게 높은 용어를 무시하는데 이 임계값을 설정하는 파라미터가 max\_df이며 문서의 80% 이상으로 나오는 단어는 분류에 도움을 주지 않을 것이라 판단하여 0.8로 설정
* 변수별로 스케일을 맞춰 주기 위해 ‘alarmno’, ‘duration’, ‘level3’, ‘level4’, ‘level5’, ‘level7’, ‘slotna’변수에 **MinMax scaler** 적용
* 모델 구축
* alarmmsg\_original 변수가 모델 학습에 핵심이라고 판단하여 텍스트 분류에 성능이 좋은 Multinomial Naive Bayes 모델을 떠올렸는데 데이터셋이 불균형함을 고려하여 imbalnced data를 잘 다루는 **Complement Naive Bayes**모델을 선택
* 또한 예측속도가 빠르고 과적합을 피하기에 좋으며 불균형한 데이터셋에 대해 성능이 좋은 **CatBoost Classifier**도 고려
* Complement Naive Bayes, CatboostClassifier, XGBoost, Support Vector Classifier, Randomforest 모델들을 각각 학습하여 테스트셋 성능을 비교해보았을 때 Catboostclassifier, Complement Naive Bayes가 가장 좋은 성능을 보였고 두 모델을 Voting Classifier를 통해 앙상블 해보았을 때 단일 모형보다 더 좋은 성능을 보임
* 따라서 **Voting Classifier을 이용한 CatBoost Classifier와 Complement Naive Bayes 앙상블 모델**을 구축

  # Create transformers

        message\_vectorizer = TfidfVectorizer(analyzer = 'char', sublinear\_tf = True, max\_df=0.8)

        additional\_scaler = MinMaxScaler()

        # Define column indices for the ColumnTransformer

        message\_col\_idx = 'alarmmsg\_original'

        additional\_cols = ['alarmno', 'duration', 'level3', 'level4', 'level5', 'level7', 'slotna']

        # Create the ColumnTransformer

        preprocessor = ColumnTransformer(

            transformers=[

                ('message', message\_vectorizer, message\_col\_idx),

                ('additional', additional\_scaler, additional\_cols)

            ],

            remainder='passthrough'

        )

        # Create individual classifiers

        classifier1 = ComplementNB(alpha=11)

        classifier2 = CatBoostClassifier(n\_estimators=80, random\_seed = 425)  # Adjust hyperparameters

        # Create the VotingClassifier

        voting\_classifier = VotingClassifier(

            estimators=[

                ('nb', classifier1),

                ('cb', classifier2)

            ],

            voting='soft',  # You can use 'hard' or 'soft' voting

        )

        # Create a pipeline with the ColumnTransformer and the classifier

        pipeline = Pipeline([

            ('preprocessor', preprocessor),

            ('classifier', voting\_classifier)

        ])

        x\_train = x\_train[['alarmmsg\_original'] + additional\_cols]

        # Train the pipeline

        self.model = pipeline.fit(x\_train, y\_train)

1. **모델 학습 및 검증 (Model Tuning & Evaluation)**

* 모델 학습 시에 train데이터에 대한 과적합을 최대한 피하기 위해 ComplementNB의 파라미터에서 큰 alpha 값을 고려하였고, CatBoostClassifier의 파라미터에서 n\_estimators를 작게 유지하였음.
  + alpha: additive smoothing parameter
  + n\_estimators: The maximum number of trees that can be built
* 테스트 셋에 대한 성능 검증 후 파라미터 값을 alpha = 11, n\_estimators = 80으로 고정
* voting\_classifier의 파라미터에서 voting은 soft로 설정하여 최종 아웃풋 결과의 확률값을 기반으로 모델들의 평균을 내어, 이 중 가장 확률값이 높은 아웃풋을 최종 아웃풋으로 설정
* 파라미터 최적화 후 테스트 셋에 대한 검증 결과 93.58% 정확도의 성능 도출

1. **결과 및 결언 (Conclusion & Discussion)**

* 알람메시지가 학습에 핵심으로 작용하여 메시지 전처리 및 벡터화가 전처리에서 가장 중요한 부분이었으며 모델 학습 시에는 train데이터와 test데이터 사이의 괴리로 인하여 train데이터에 대한 과적합을 막는 것이 핵심
* 따라서 메시지 전처리 시 Tfidfvectorization 방식과 데이터 셋 특성에 맞는 모델(Complement Naive Bayes, CatBoost Classifier)의 조합 및 파라미터 설정이 모델 성능에 큰 기여를 함.
* 변수 당 ticketno 별로 취합하는 방식을 더 다양한 방식으로 시도해 볼 필요가 있음.
* slot, port가 root\_cause\_domain에 따라 분포가 많이 달라서 학습에 이용하지 못했는데 domain에 따라 어떤 방식으로 나타나는지 알면 변수로 활용하여 모델 성능을 높일 수 있을 것으로 보임