PARK Yeseo
COSE362 Machine Learning

Spam Mail Detection Project

i. 실행 환경

Kaggle kernal 에 있는 notebook 을 이용하여, 프로그램을 실행하였습니다. Data 들을 direct 하게 input 으로 가지고 올 수 있고, 최종 결과를 output 으로 바로 내보낼 수 있었습니다. 언어는 파이썬을 사용하였습니다.

ii. 데이터에 대한 분석과 정제 방법

주어진 train.csv 와 test.csv 의 모양이 같다라고 설명되어있었지만, 사실상 파일을 열었을때, 두 파일의 column 의 수가 달랐습니다. train.shape 을 하였을때 (3620,3) 즉, 3620 개의 row 와 3 개의 column 으로 구성되어있음을 확인하였습니다. train.columns 을 하였을때 Index(['id', 'label', 'mail'], dtype='object') 이 나왔습니다. test.shape 을 하였을때 (1551,2) 즉. 1551 개의 row 와 2 개의 column 으로 구성되어있었고, test.columns 을 하였을때 Index(['id', 'mail'], dtype='object')이 나왔습니다. 이로 인하여, test 의 column 은 train 의 column 수 보다 1 개가 작았고, label 이 없다는 것을 확인하였습니다. 최종적으로 예측하여 구해야 하는 타깃이 label 이라는것도 알 수 있었습니다. 두 데이터들 모두 isnull()을 사용하여, 비어있는 cell 이 있는지 확인하였고, 모두 정상적으로 데이터가 있음을 확인하였습니다.

<정제방법>

학습을 하기 전에 train 과 test 의 mail column 의 내용을 정리하여 주었습니다. 특수기호, URL, 의미없는 공백, punctionaiton 을 삭제하고, stopwords 도 삭제를 하여 clean words 만 남도록 필터링을 하였습니다.

<Training>

'mail' 과 'label'이 둘다 있는 train.csv 를 이용하여 학습을 시켰습니다. Id 는 큰 영향을 안끼쳤고, X= X=train['mail'], y=train['label']으로 설정해 주었습니다. train_test_split 을 이용하여 train.csv 의 데이터중 90%는 train 으로, 10%는 test 로 어느 모델을 사용하였을때 가장 높은 accuracy 가 나오는지 확인을 가능하게 하였습니다.

iii. 본인이 구현한 모델에 대한 설명

각각 모델을 정하고 나서, 그 다음 단계인 classification 함수를 무조건 실행하게 됩니다. 그러기 위해서 먼저 classification 함수를 구현하였습니다. 앞에서 언급한 대로, train data 를 test_split 과 test_size=0.1 를 이용하여 90%는 학습을, 10%는 test 를 하도록 하였습니다. 그다음 pipeline 을 이용하여, 한번에 CountVectorizer()와 TfidfTransformer()가 실행 됩니다. CountVectorizer()는 모든단어들을 사전에 해당하는 id 와 같이 저장합니다. 그리고 단어가 몇번 나왔는지를 세어줍니다. TfidTransformer()는 the 나 and

와 같이 문장에 많이 등장하지만 의미가 별로 중요하지 않는 단어들의 가치를 낮춰줍니다.
pip_model.fit(x_train, y_train) 을 실행 함으로서 각 단어의 빈도수를 파악한 다음 들 필요한 단어들의 비중을 낮추어 줍니다. 그 다음에는 accuracy 를 뽑아내고, Classification_report 를 불러 presicion, recall, f1-score,support 값을 알수 있고, 마지막에 confusion_matrix 를 print 할수 있도록 하였습니다.

총 6 개의 다른 모델들을 사용하였습니다.

iv. 성능 평가와 결과분석

a. LogisticRegression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classification(model, X, y)
```

```
Accuracy: 99.17127071823204
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       259
           1       0.99      0.98      0.99       103

    accuracy                           0.99       362
   macro avg       0.99      0.99      0.99       362
weighted avg       0.99      0.99      0.99       362

[[258   1]
 [  2 101]]
```

Figure 1 Used LogisticRegression

이 함수는 spam 이다 아니다만 구분해서, binary classification 으로 됩니다. Logistic function 을 이용하여, 0.5 미만이면 0, 0.5 이상이면 1 로 보여줍니다. Accuracy 는 99.17 과 precision 과 F1-score 는 0.99 로 높은 값이 나왔음을 확인 할 수 있습니다.

b. MultinomialNB

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
classification(model, X, y)
```

```
Accuracy: 86.1878453038674
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       259
           1       1.00      0.51      0.68       103

    accuracy                           0.86       362
   macro avg       0.92      0.76      0.80       362
weighted avg       0.88      0.86      0.85       362

[[259   0]
 [ 50  53]]
```

Figure 2 Used MultinomialNB

Naïve Bayes 를 적용한 모델이며, 단어의 순서와 상관없이 단어의 출현 빈도를 이용하여, 정상메일과 spam 메일중 에서 어느곳에 더 많이 쓰이는지 단어를 분석하여 확률을 구합니다. 정상메일(0)의 precision 값은 0.84 로 다소 낮게 나왔습니다. 특히, Spam 일 경우(1) recall 과 f1-score 이 각각 0.51 과 0.68 로 나왔습니다. 또한 confusion matrix 표를 보면 정상메일로 예측을 하였지만, 실제로는 spam 인 경우가 50 으로 나왔습니다. 이들을 종합해 보면, spam 메일을 정상메일이라고 잘못 예측하는 경우가 높다는 것을 알 수 있었습니다.
종합적으로 6 개의 모델 중에서 가장 성능이 낮았습니다. 그러한 이유는 나이브 베이즈 모델 특성중 하나인 모든 특징이 동등하다고 여겨지는 가정이 자주 일어났기 때문입니다. 이 spam detection 의 실험에서는 앞에 언급한 is, an, the 와 같이 단어의 의미가 크게 없는 단어들도 다른 단어들과 동등하게 비중을 두어 사용이 됩니다. 그래서 정확도가 다른 모델들에 비해 낮게 나왔습니다.
하지만, 나이브 베이즈 모델은 데이터가 많은 모델에서도 빠르며, binary classification 을 사용하였기에 다른 모델들에 비해서 효율은 적어도 spam mail detection 을 할 수 있는 모델 입니다.

c. SVC

```
from sklearn.svm import SVC
model = SVC(C=3)
classification(model, X, y)
```

```
Accuracy: 99.4475138121547
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       259
           1       0.99      0.99      0.99       103

    accuracy                           0.99       362
   macro avg       0.99      0.99      0.99       362
weighted avg       0.99      0.99      0.99       362

[[258   1]
 [  1 102]]
```

Figure 3 Used SVC

하이퍼파라미터인 C 의 값을 100 으로 하여도 3 일때와 같은 accuracy 값이 나왔습니다. 이론적으로는, C 의 값이 작으면 일부 오차를 허용할 수 있는 서포트 백터를 선정합니다. 다른말로 C 의 값이 커지면, 오차를 허용되는 범위가 줄어듭니다. Confusion matrix 또한 False Negative 와 False positive 의 값이 1 임으로 예측값과 실제값이 차이가 적다는것을 알 수 있습니다. MLP Classifier 다음으로 SGDClasssifier 와 같이 성능이 가장 좋은 모델에 속합니다.

d. RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
classification(model, X, y)
```

```
Accuracy: 98.34254143646409
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       259
           1       0.99      0.95      0.97       103

    accuracy                           0.98       362
   macro avg       0.99      0.97      0.98       362
weighted avg       0.98      0.98      0.98       362

[[258   1]
 [  5  98]]
```

Figure 4 Used RandomForestClassifier

데이터를 무작위로 뽑아 모델을 만든 다음에 각각의 모델들의 예측 결과값의 평균으로 result 를 출력합니다. 각각 모델들은 전체 데이터에서 랜덤하게 서로 다르게 추출되고, 중복이 허용이 가능하여 일반화 성능을 높임과 동시 모델의 다양성도 있습니다. 하지만, confusion matrix 에서 실제값은 spam 이지만, 정상메일이라고 예측한 경우가 5 로, recall 의 spam 메일과 종합하면 일부 spam 메일이 잘못 예측된 경우가 있음을 보여줍니다. 데이터가 많은 세트에서도 잘 동작하지만 훈련과 예측이 상대적으로 느립니다. 특히, parameter 를 적용하지 않았는데도 불구하고 98.3%라는 높은 accuracy 값이 나왔습니다.

e. MLPClassifier

```
from sklearn.neural_network import MLPClassifier
model=MLPClassifier()
classification(model, X, y)
```

```
Accuracy: 99.72375690607734
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       259
           1       1.00      0.99      1.00       103

    accuracy                           1.00       362
   macro avg       1.00      1.00      1.00       362
weighted avg       1.00      1.00      1.00       362

[[259   0]
 [  1 102]]
```

Figure 5 Used MLPClassifier

Spam 일때의 recall 값을 제외하고 precision 와 f1-score 모두 1.00 이 나왔습니다. 또한 confusion matrix 에서 예측값과 실제값이 다른 경우가 1 과 0 으로 낮은 수가 나왔습니. Neural Network 를 기반으로 한 모델이며, 6 가지 모델중에서 가장 성능이 잘 나온 모델입니다. MLP classifier 에서는 최소 Hidden layer 가 1 개 이상 존재해서 기존에 있던 단층 perceptron 에 비해 분류 할 수 있는 방법이 많아져 accuracy 가

높아집니다. 그래서 앞에 나온 단층 퍼셉트론에 비해서 accuracy 가 높게 나왔습니다. 하지만 은닉층이 많아지면 정확도는 올라가지만 계산을 해야 하는 parameter 가 많아지게 됨으로 학습을 하는데 상대적으로 오래 걸립니다. 실제로 train.csv 를 돌릴 때에도 다른 모델들은 금방 값이 나왔지만 이 모델은 3 분넘게 걸렸습니다. 그러나, 한번 학습을 한 후에는 적용을 하는데 빠릅니다. 다른 장점으로는 데이터 수가 작아도 비슷한 accuracy 를 나타낼 수 있습니다.

f. SGDClassifier

```
from sklearn.linear_model import SGDClassifier
model=SGDClassifier()
classification(model, X, y)
```

```
Accuracy: 99.4475138121547
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       259
           1       0.99      0.99      0.99       103

    accuracy                           0.99       362
   macro avg       0.99      0.99      0.99       362
weighted avg       0.99      0.99      0.99       362

[[258   1]
 [  1 102]]
```

Figure 6 Used SGDClassifier i

```
from sklearn.linear_model import SGDClassifier
model=SGDClassifier(loss="hinge", penalty="l2", max_iter=100)
classification(model, X, y)
```

```
Accuracy: 99.72375690607734
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       259
           1       1.00      0.99      1.00       103

    accuracy                           1.00       362
   macro avg       1.00      1.00      1.00       362
weighted avg       1.00      1.00      1.00       362

[[259   0]
 [  1 102]]
```

Figure 7 Used SGDClassifier ii

제가 실험한 모델 중에서 MLP 다음으로 가장 성능이 좋은 모델이였습니다. Loss="hinge"로 설정하여 liner SVM 모델을 보여줍니다. 그럼에도 불구하고 Train.csv 의 데이터로 예측했을 때에서는 log 값을 사용한는 SVC 의 결과값과 같게 나왔습니다. SGDClasssifier 에 parameter 값들을 지정해 주었더니 99.4475 에서 99.723 으로 accuracy 가 상승하였습니다. 하지만, 일정 epoch 동안 성능이 향상되지 않아서 더 큰 iteration 을 하라는 에러메세지가 떴습니다. 그래서 max_iter 를 1000 과 5000 으로 바꾸어 주었지만 100 번 했을때의 결과랑 같게 나왔습니다. Test.csv 에는 데이터가 1000 개 뿐이라 정확도는 높았지만 다른것과 비교해서 큰 차이는 없었습니다. 하지만 학습하는 속도가 빠르고, 전체가 아닌 개별 데이터의 경사하강법을 이용하기에, 미분값이 크게 변함으로 local minumum 에서 빠질 확률을 줄여준다는

장점이 있습니다. 데이터의 수가 커지면 커질수록 대용량에서는 더욱 효과적일것입니다.

v. Overall & Accuracy 를 높인 방법

전체적인 모델에 대해서 test.csv 를 leaderboard 에 올려 확인하지는 않았지만, 대다수 모델이 train.csv 에 나왔던 accuracy 보다 낮게 측정 되었습니다. 예를 들면 SVC 모델은 train.csv 로 훈련 하였을 경우 99.4475 의 accuracy 를 보여주었지만 Kaggle leader board 에서는 0.95951 의 값이 나왔습니다. 또한 MLP classifier 에서도 train.csv 는 99.7238 이였지만, leader board 에서 test.csv 에 적용 시켰을때는 0.98312 로 줄어들었습니다.

Train data split 에서 test size 를 0.2, 0.1, 0.05 로 3 가지 경우에 대해서 해보았는데 0.1 일때정확도가 제일 높게 나왔습니다. 0.2 일때가 가장 이상적일꺼라고 생각했던 제 예상과 다르게 나왔습니다. 0.1 일때가 더 높았다는것으로 추론하면 0.2 인 경우에는 train 의 수가 부족하였고, 0.05 는 반대로 test 해야하는 데이터가 적었을것으로 추론 됩니다.

vi. Error Analysis

100%가 되지 않았던 이유에 여러가지가 있었던것 같습니다. 먼저, 언어가 영어가 아니면 인식이 잘 되지 않았던것 같습니다. 예를들면 직접적으로 test.csv 와 MLPclassifer 를 적용하여 얻은 predicted 의 label 을 비교하였을때, id 가 29 번인곳에서 “hastala vista”라는 스페인 어가 있었습니다. 또한 440 번 id 또한 메일 내용이 완전한 포르투칼언어로 되어있었습니다. 하지만, predicted value 에서는 이 두 id 를 spam 으로 인식하였습니다.

두번째는 오타나 암호화 문자처럼 쓰면 정상단어라고 인식을 할 수 없다는 문제점도 있었습니다. 예를들면 train 에 id 5 번에서는 pain 대신 pa!n 으로 써졌다는 것을 알 수 있습니다. 이 경우, predicted 값은 spam 으로 인식을 하였습니다. 또한 id 25 번째에서는 “taablets” 이라는 철자 오류가 있었습니다. 단순히 눈으로 보면 알 수 있는 단어여도 오타나, 암호화를 한다면 의미가 없는 단어여서 spam 으로 간주 될 수 있는 문제점이 생일 수도 있습니다.

Clean mail

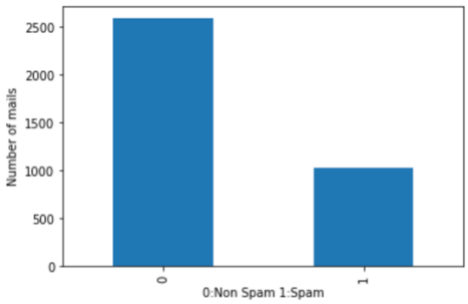
처음에, mail 의 내용을 정제할때는 숫자는 고려를 하지 않았습니다. 하지만 나중에 확인을 해 본 결과, 돈 단위 같이 숫자가 의미있는 숫자임에도 spam 으로 예측을 하여서, 추가적으로 filter 단계에서 number 를 삭제하였습니다. 적은 숫자였지만 모든 모듈에서 accuracy 가 약간 올랐습니다.

vii. 시각화

a. matplotlib

```
import matplotlib.pyplot as plt
train['label'].value_counts().plot(kind='bar')
plt.xlabel('0:Non Spam 1:Spam')
plt.ylabel('Number of mails')
print(train.label.value_counts())
```

```
0    2589
1    1031
Name: label, dtype: int64
```




```
import matplotlib.pyplot as plt
import seaborn as sns
submission['label'].value_counts().plot(kind='bar')
plt.xlabel('0:Non Spam 1:Spam')
plt.ylabel('Number of mails')
print(submission.label.value_counts())
```

```
0    1088
1     463
Name: label, dtype: int64
```

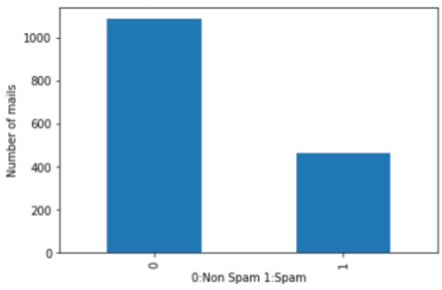


Figure 8 (Train.csv 의 데이터(좌) Submission.csv 의 데이터(우))

Train.csv 의 실제값과 test.csv 의 예측값을 각각 바그래프로 나탄내 주었습니다. Train.csv 데이터에서는 총 3620 개의 데이터중 정상메일이 2589(약 71.5%)와 Spam 메일 1031 개(28.5%)로 이루어져 있습니다. Test.csv 의 데이터로 예측한 submission.csv 파일에는 총 1551 개의 데이터중 정상메일이 1088 개(약 70%)와 spam 메일 403 개(약 30%)로 train.csv 의 비율과 비슷하게 예측되었습니다.

b. WordCloud

Figure 9 Test.csv 에서 정상메일

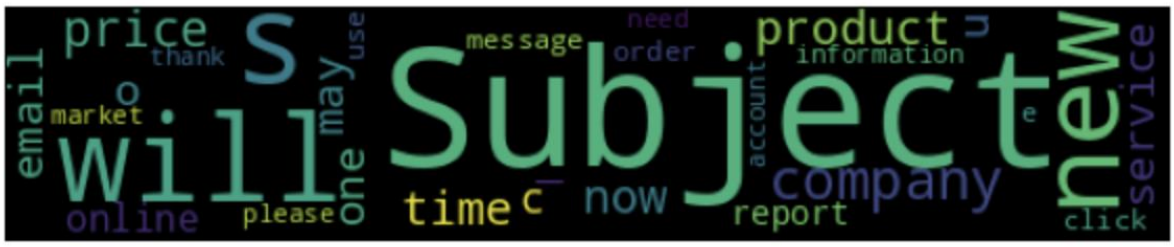


Figure 10 Test.csv 에서 Spam 메일

test.csv 데이터에서 WorldCloud 를 이용하여 정상메일과 spam 메일에서 어떤 단어수가 중요도나 높은 빈도로 나왔는지 직관적으로 알 수 있게 하였습니다. Top30 개의 단어를 출력하게 하였고, 글자의 크기가 클 수록 중요학고 빈도수가 많다는 의미입니다. 정상메일에서는 etc 라는 단어가 많이 보였습니다. Spam

메일에서는 subject 이라는 단어가 가장 크게 눈에 띄었고, 그다음으로 will, new 라는 단어가 많이 보였음을 알 수 있습니다.