Kaggle 데이터를 이용한 머신러닝 예측

정보통신대학원 융합정보기술학과 인공지능전공

202476717

신영찬

개요

URL 정보와 해당 사이트의 스팸여부가 들어있는 데이터 셋을 활용하여 URL 을 활용하여 스팸인지 아닌지 여부를 확인하기에 적합한 모델을 비교하여 선정한 후 성능 개선을 통해 모델을 구축하고자 한다.

1. Kaggle 데이터셋

사용된 데이터셋: https://www.kaggle.com/datasets/shivamb/spam-url-prediction

1) 데이터 구성 (148,303 개)

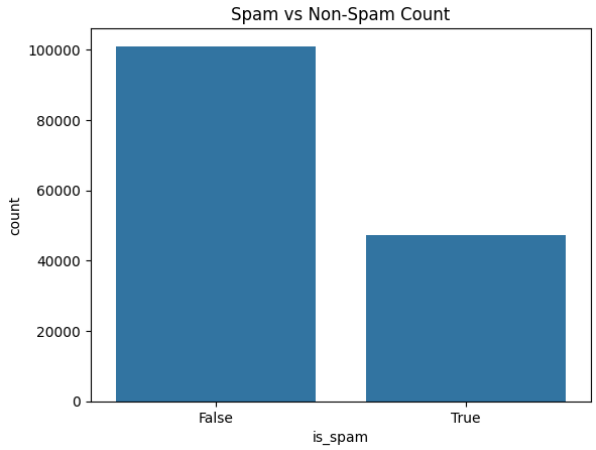
- URL
- Spam 여부

2) 데이터 전처리

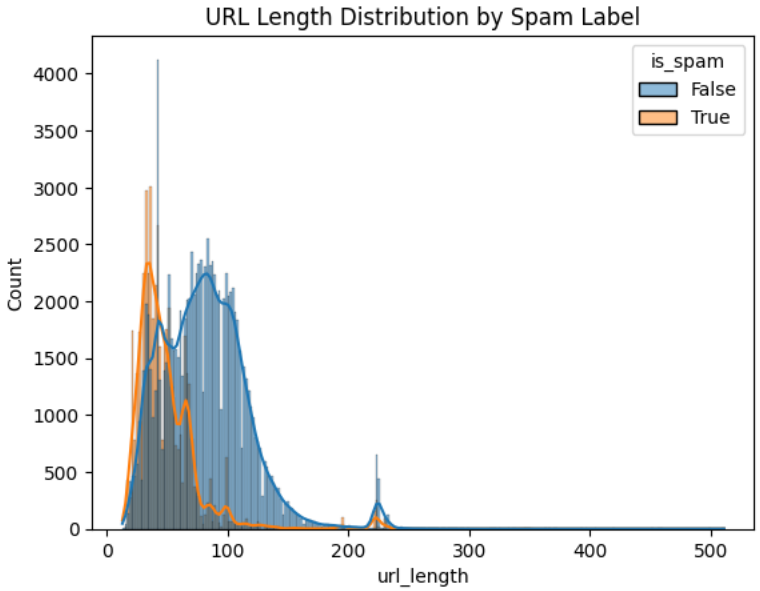
머신러닝 모델은 문자열(URL)을 직접 이해할 수 없으므로 URL 에서 수치형 특징 추출 및 전처리

- url_length (URL 길이)
- domain_length (도메인 길이)
- subdomain_count (서브 도메인 점 개수)
- path_depth (경로 깊이)
- digit_count (숫자 개수)
- special_count (특수문자 개수)
- is_https (HTTPS 여부)

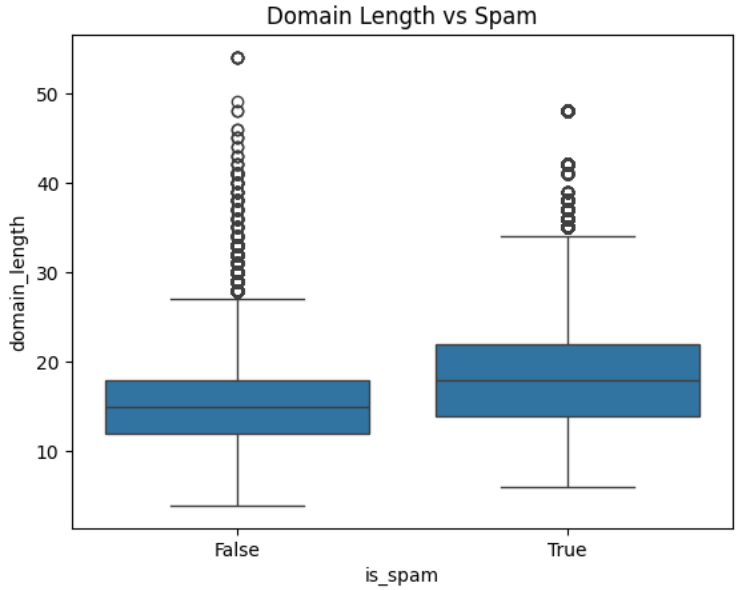
2. 시각화 분석



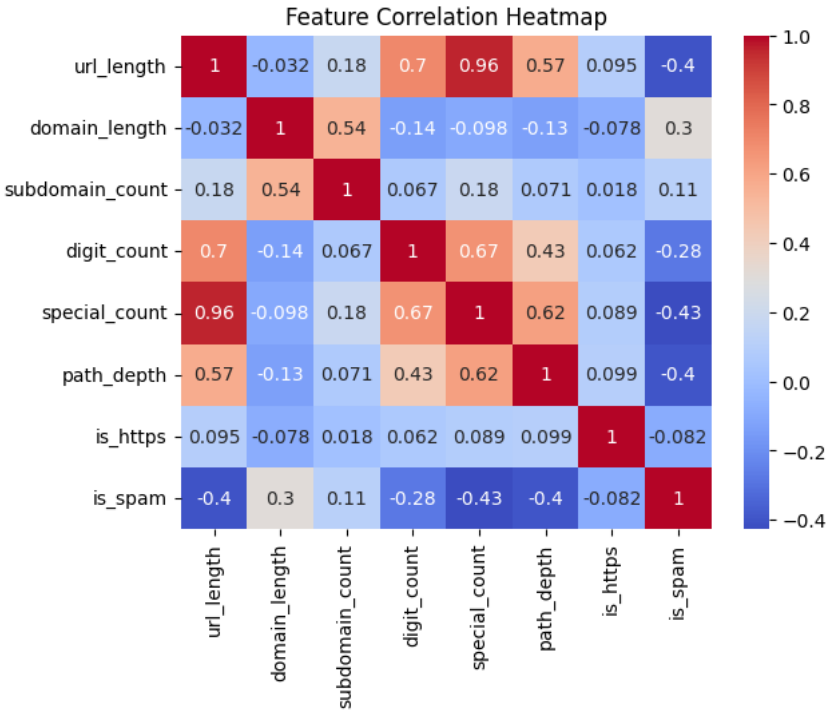
[스팸 여부 그래프]



[URL 길이에 따른 스팸 여부]



[도메인 길이와 스팸 비교]



[Heatmap 분석 결과]

Heatmap 분석 결과 url_length(URL 길이)와 digit_count(숫자 개수) 등의 변수가 유의미한 상관관계를 나타냈으며 스팸 URL 은 정상 URL 에 비해 길이가 길고 특수문자와 숫자가 더 많이 포함되는 경향이 있음을 확인함. HTTPS 를 사용하지 않는 비율이 스팸에서 더 높게 나타났다.

3. ML 모델 비교 및 선정

```python
models = {
    "RandomForest": RandomForestClassifier(
        n_estimators=300,
        max_depth=None,
        random_state=42
    ),

    "XGBoost": XGBClassifier(
        n_estimators=300,
        max_depth=6,
        learning_rate=0.05,
        subsample=0.8,
        colsample_bytree=0.8,
        eval_metric='logloss',
        random_state=42
    ),
    "LightGBM":LGBMClassifier(
        n_estimators=300,
        learning_rate=0.05,
        max_depth=-1,
        subsample=0.8,
        colsample_bytree=0.8
    )
}
```

```
====== 📊 모델 성능 비교 결과 (RF / XGB / LGBM) ======
          Model  Accuracy        F1  Precision    Recall
0  RandomForest  0.947271  0.918652   0.903982  0.933806
2      LightGBM  0.905061  0.849412   0.859245  0.839801
1       XGBoost  0.903645  0.846559   0.859854  0.833668
```

Random Forest, LightGBM, XGBoost 3 개의 모델을 선정하여 각각 성능을 비교 했다.

결과
- Random Forest: F1-Score 0.918
- LightGBM: F1-Score 0.849
- XGBoost: F1-Score 0.846

비교 결과 Random Forest 가 가장 우수함으로 해당 모델을 선정

4. 데이터셋 분리

전체 데이터를 학습용(Train) 80% 평가용(Test) 20%로 분리 하였다. 또한 stratify 옵션을 적용하여 학습 데이터와 평가 데이터 간의 스팸 클래스 비율을 일정하게 유지하여 데이터 편향을 방지하였다.

5. 학습 및 초기 평가

```
rf_model = models["RandomForest"]

rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)

rf_acc = accuracy_score(y_test, rf_preds)
rf_f1 = f1_score(y_test, rf_preds)
rf_precision = precision_score(y_test, rf_preds)
rf_recall = recall_score(y_test, rf_preds)

print("RF 성능:", rf_acc, rf_f1, rf_precision, rf_recall)
```

✓ 11.9s

```
RF 성능: 0.9472708270119011 0.9186518256527619 0.9039819838263896 0.933805646610976
```

선정된 Random Forest 의 기본 설정으로 학습을 진행한 후 평가 데이터를 통해 성능을 확인했다.

1) 초기 모델 성능

- Accuracy (정확도): 약 94.7%
- Precision (정밀도): 약 90.4%
- Recall (재현율): 약 93.4%
- F1-Score: 0.918

6. 최적화

```python
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier

# 기존 모델 정의 사용 (random_state=42 유지)
rf_model_base = RandomForestClassifier(random_state=42)

param_grid_rf_refined = {
    "n_estimators": [500, 700, 1000],
    "max_depth": [20, 30, 40, None],
    "min_samples_split": [2, 5],
    "min_samples_leaf": [1, 2, 3],
    "max_features": ["sqrt", 0.3, 0.5],
    "class_weight": ["balanced", "balanced_subsample"]
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
search_rf_refined = RandomizedSearchCV(
    rf_model_base,
    param_grid_rf_refined,
    n_iter=50,  # 탐색 횟수 증가
    scoring="f1",
    cv=cv,
    n_jobs=-1,
    random_state=42,
    verbose=1
)
```

모델의 일반화 성능을 높이고 과적합을 방지하기 위해 RandomizedSearchCV 를 사용하여 하이퍼 파라미터 최적화를 수행했다.

- 튜닝 파라미터: n_estimators(트리 개수), max_depth(트리 깊이), class_weight(클래스 가중치) 등

7. 최종 예측 및 분석결과

1) 최종 예측

```
최종 정확도 (Accuracy): 0.9392
최종 F1-Score: 0.9112
최종 정밀도 (Precision): 0.8531
최종 재현율 (Recall): 0.9777

[ 최종 분류 성능 리포트 ]
              precision    recall  f1-score   support

           0       0.99      0.92      0.95     20204
           1       0.85      0.98      0.91      9457

    accuracy                           0.94     29661
   macro avg       0.92      0.95      0.93     29661
weighted avg       0.95      0.94      0.94     29661


[ 혼동 행렬 (Confusion Matrix) ]
[[18612  1592]
 [  211  9246]]
```

2) 결과

- 높은 재현율(Recall 97.8%)

최적화 과정에서 class_weight='balanced'를 적용한 결과, 재현율이 매우 높게 나타났다. 실제 스팸 URL 을 스팸이라고 탐지해내는 능력이 탁월함을 의미하며 보안 모델의 특성상 스팸을 놓치지 않는 것이 중요하므로 매우 긍정적인 결과이다.

- 안정적인 성능

약 14 만 개의 데이터 중 94%에 달하는 정확도로 스팸을 걸러낼 수 있음을 확인하였다.

- 혼동 행렬 분석

전체 테스트 데이터 29,661 개 중 스팸을 정상으로 잘못 분류한 케이스는 211 건으로 성능 향상을 증명했다.

8. 결론

URL 문자열에서 유의미한 특징을 추출하는 전처리 과정이 모델 성능에 큰 영향을 미침을 확인 할 수 있었다. 또한, 모델 비교를 통해 Random Forest 가 해당 데이터셋에 가장 적합함을 입증하였으며 하이퍼 파라미터 튜닝을 통해 재현율을 97% 이상으로 끌어올려 보안 위협 탐지에 효과적인 모델을 구축 할 수 있었다.