악성 사이트 탐지 모델

부산경남
8반 안중찬

단변량 분석

단변량 분석

AIVLE

결측치 여부 조사

```
url_len 242
url_num_hyphens_dom 8
url_path_len 204
url_domain_len 68
url_hostname_len 67
url_num_dots 17
url_num_underscores 13
url_query_len 78
url_num_query_para 9
url_ip_present 2
url_entropy 2524
url_chinese_present 1
url_port 2
```

```
html_num_tags('iframe') 14
html_num_tags('script') 78
html_num_tags('embed') 3
html_num_tags('object') 7
html_num_tags('div') 302
html_num_tags('head') 4
html_num_tags('body') 4
html_num_tags('form') 14
html_num_tags('a') 300
html_num_tags('applet') 1
```

- 결측 값의 경우 url_path_len과 url_domain_len에 각각 한개씩 존재
- 이 때, 결측치를 제거하는 것이 좋긴 하지만, test 셋이 무엇인지 모르므로, 이에 대한 대비책이 필요하다.

열 각각의 원소의 개수 추출

```
url_len 242
url_num_hyphens_dom 8
url_path_len 204
url_domain_len 68
url_hostname_len 67
url_num_dots 17
url_num_underscores 13
url_query_len 78
url_num_query_para 9
url_ip_present 2
url_entropy 2524
url_chinese_present 1
url_port 2
```

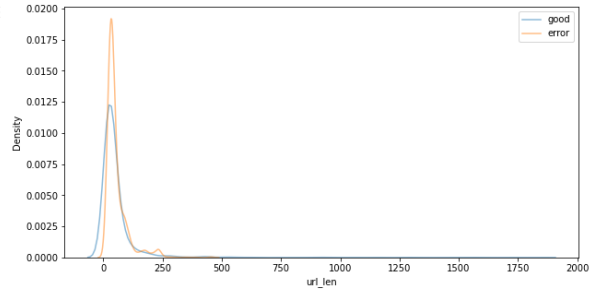
```
html_num_tags('iframe') 14
html_num_tags('script') 78
html_num_tags('embed') 3
html_num_tags('object') 7
html_num_tags('div') 302
html_num_tags('head') 4
html_num_tags('body') 4
html_num_tags('form') 14
html_num_tags('a') 300
html_num_tags('applet') 1
```

- Url_chinese_presen와 html_num_tags('applet')은 원소의 개수가 한 개임. 즉, 영향이 없는 변수임
- 이들을 제거해줄지 말지를 판별해야함

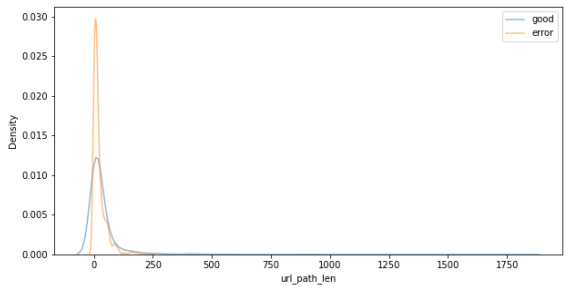
단변량 분석

정상과 불량에 대한 히스토그램 비교(length와 numbers)

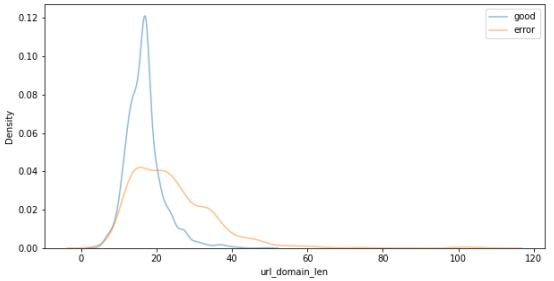
- url_len



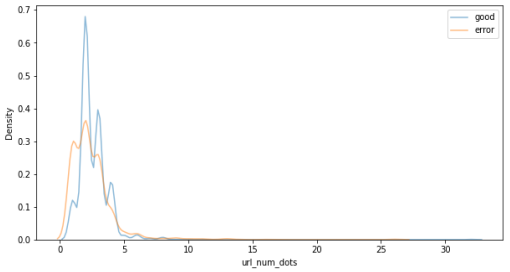
- url_path_len



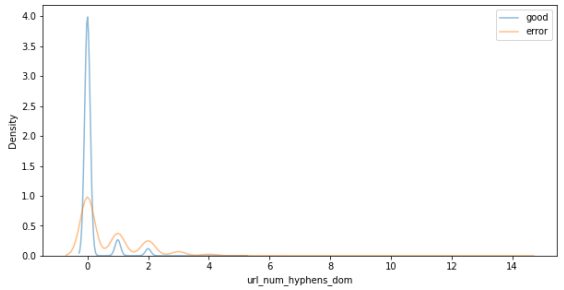
- url_domain_len



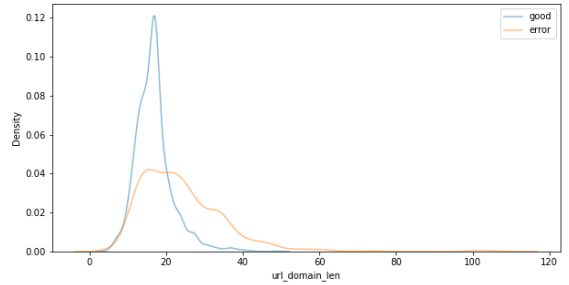
- url_num_dots



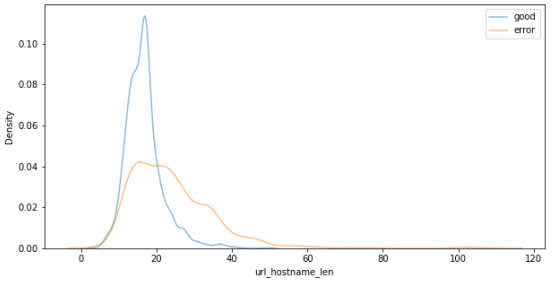
- url_num_hyphens_dom



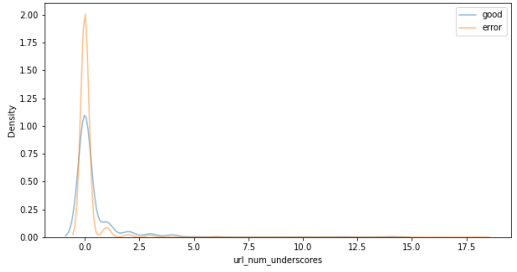
- url_domain_len



- url_hostname_len



- url_num_underscores



- url_len, url_path_len, url_num_underscores에서 최빈값들의 차이가 보인다. 이들은 모델에 적용할 수 있는 변수들임과 동시에, 분포가 비슷하여 넣을 것인지를 고려해봐야 한다.

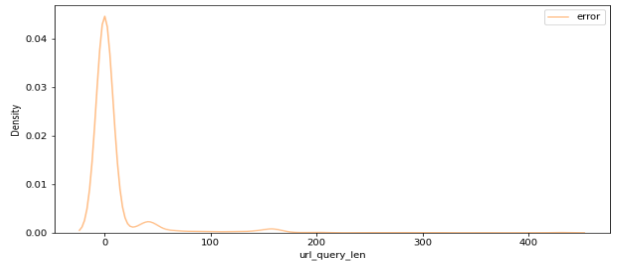
- Url_domain_len과 url_hostname_len은 분포가 불량과 정상이 매우 비슷하다는 것을 알 수 있다. 하지만, 불량과 정상의 분포가 확연히 차이나는 것 또한 볼 수 있다.

- 모든 분포가 최빈 값에서 차이가 많이 난다.

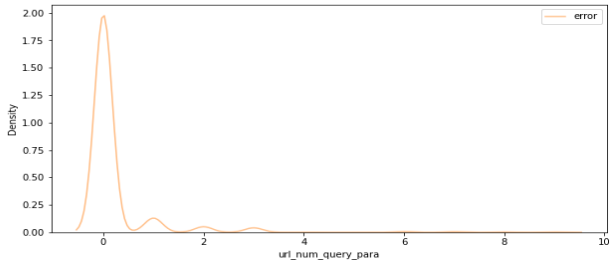
단변량 분석

정상과 불량에 대한 히스토그램 비교(present, port, ect…)

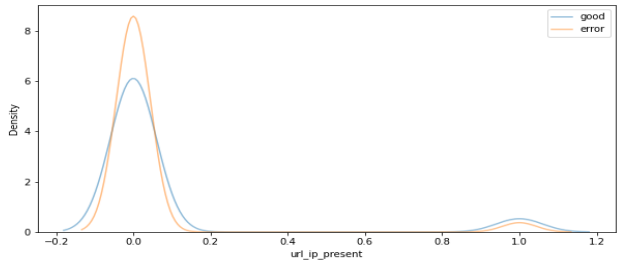
- url_query_len



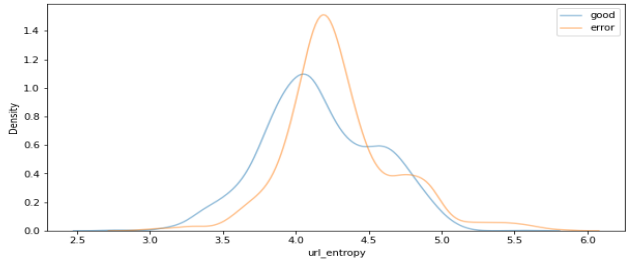
- url_num_query_para



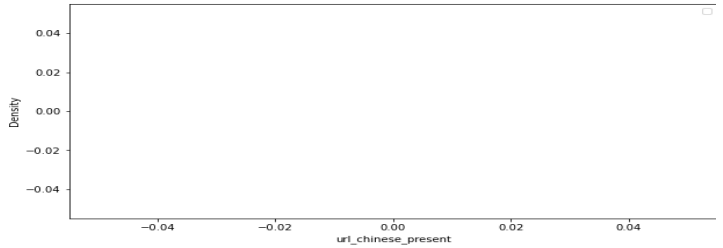
- url_ip_present



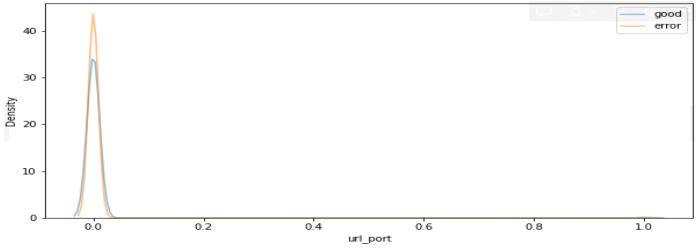
- url_entropy



- url_chinese_present



- url_port

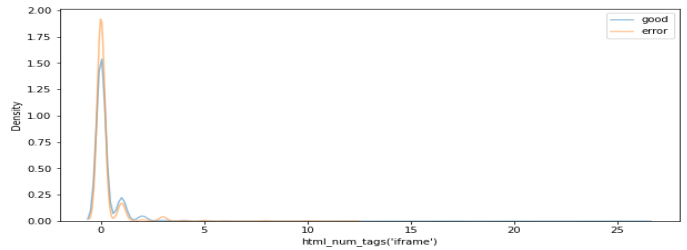


- url_query_len은 불량 데이터 셋에서만 분포가 존재한다.
- url_Chinese_present의 경우 불량과 정상에서 모두 분포가 존재하지 않으므로, 이를 학습시킬 수는 없다. Test 셋에서 분포가 존재하더라도 trian set에서 학습할 수 없으니 제거하는 방향이 올바르다.
- 나머지 분포에 대해서는 모두 비슷한 분포를 가지고 있다는 것을 안다.

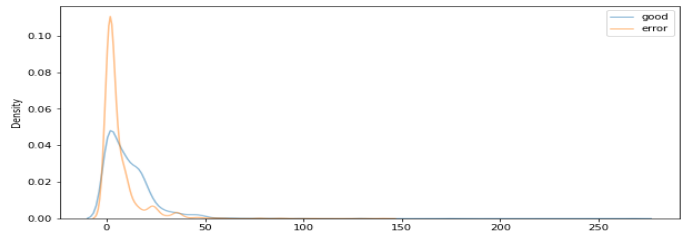
단변량 분석

정상과 불량에 대한 히스토그램 비교(Tag)

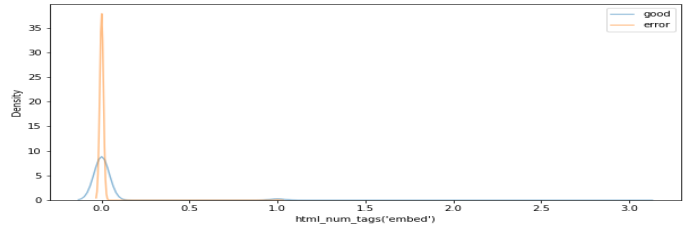
- html_num_tags('iframe')



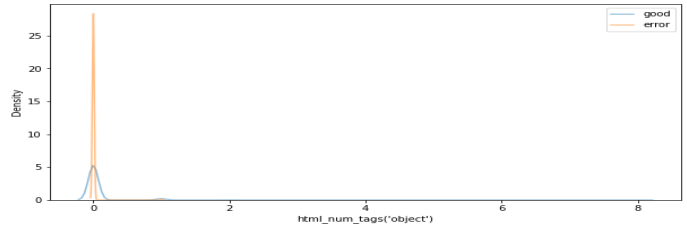
- html_num_tags('script')



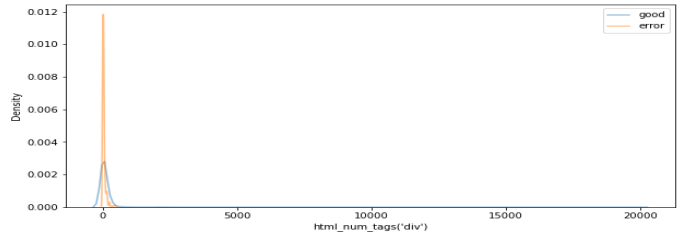
- html_num_tags('embed')



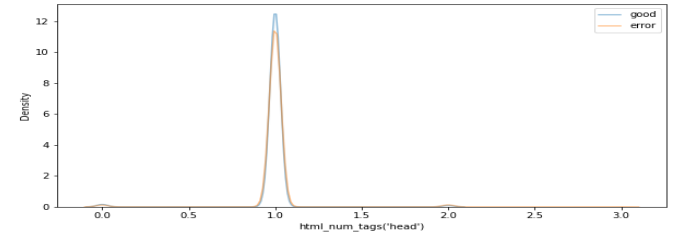
- html_num_tags('object')



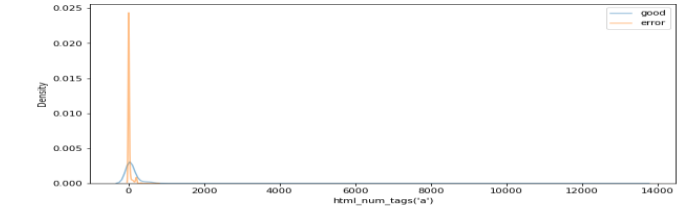
- html_num_tags('div')



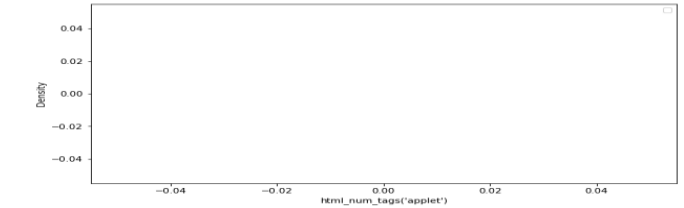
- html_num_tags('head')



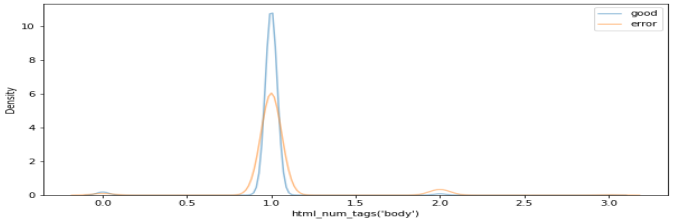
- html_num_tags('a')



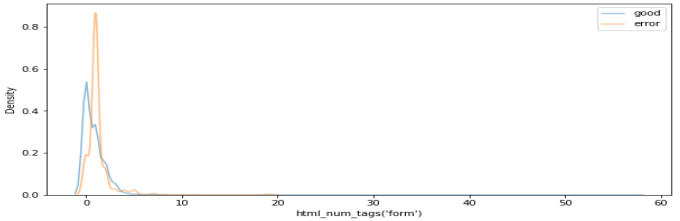
- html_num_tags('applet')



- html_num_tags('body')



- html_num_tags('form')



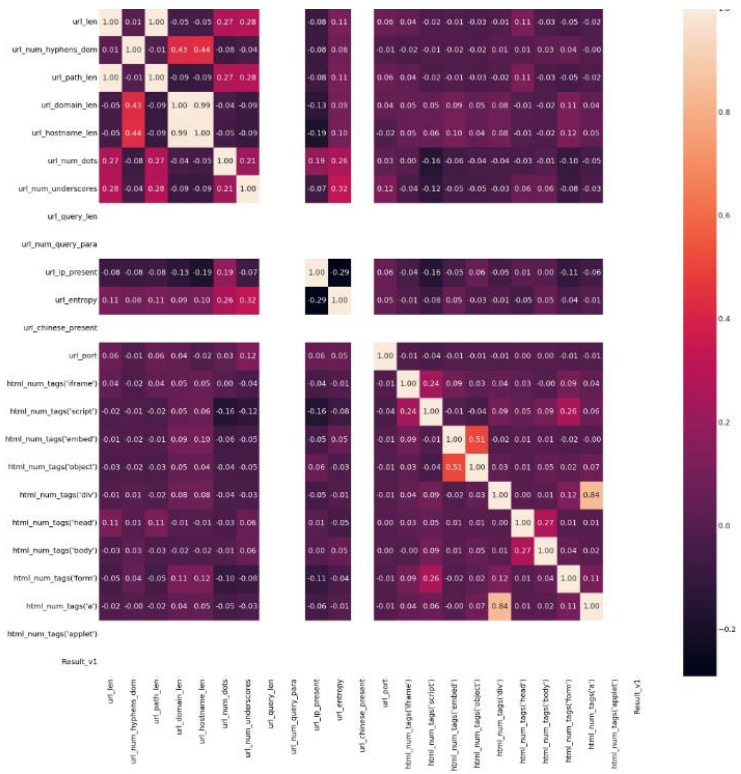
- Tag에 대한 변수들이다
- 변수들이 모두 비슷한 움직임을 보인다.
- 최빈값에서 많이 차이난다.
- Applet의 경우 분포가 존재하지 않는다.

이변량 분석

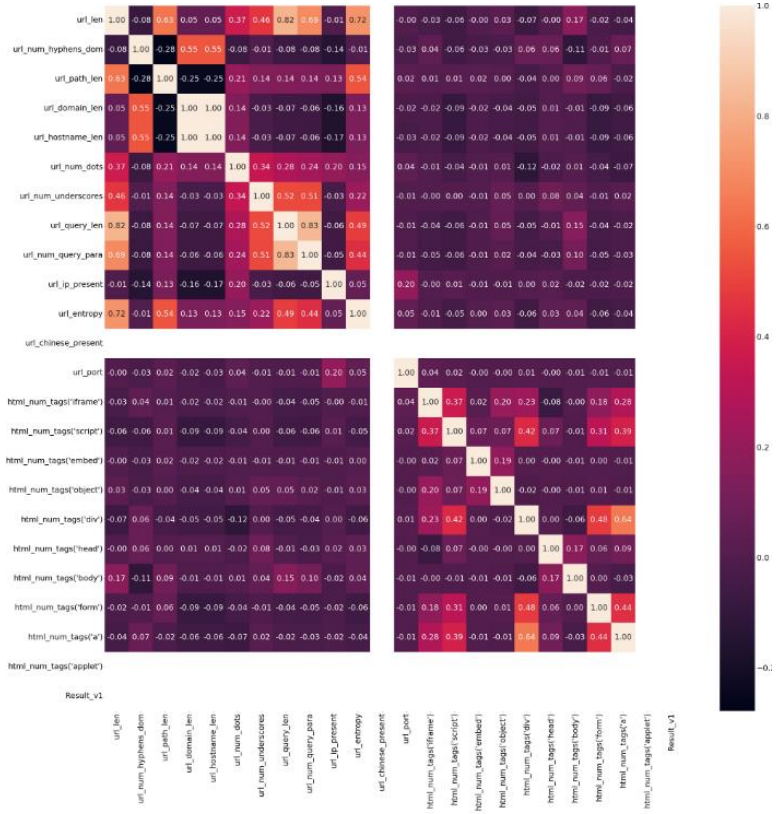
이변량 분석

정상과 불량간의 상관성의 차이가 존재하는가?

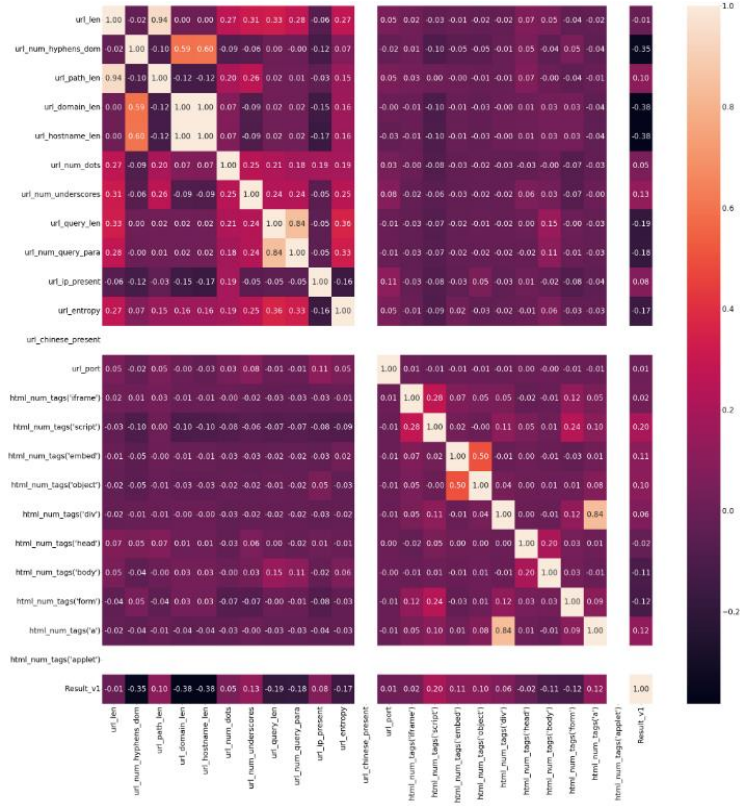
- 정상 데이터 상관분석



- 불량 데이터 상관분석



- 전체 데이터 상관분석



정상 데이터와 불량 데이터 간의 상관관계 분포가 다르므로, 모델이 쉽게 학습될 것이라는 것을 암시한다.

이변량 분석

AIVLE

Logistic Regression P value

- **로지스틱 회귀분석 결과**

url_len	0.932637
url_num_hyphens_dom	1.208474e-61
url_path_len	4.907666e-09
url_domain_len	3.815200e-19
url_hostname_len	1.028186e-19
url_num_dots	0.031656
url_num_underscores	4.394527e-13
url_query_len	0.999577
url_num_query_para	0.997772
url_ip_present	0.000002
url_entropy	0.915585
url_port	0.484254
html_num_tags('iframe')	0.692840
html_num_tags('script')	1.359310e-24
html_num_tags('embed')	3.335172e-07
html_num_tags('object')	4.489999e-09
html_num_tags('div')	4.049643e-12
html_num_tags('head')	0.544368
html_num_tags('body')	0.515296
html_num_tags('form')	2.7888460e-09
html_num_tags('a')	3.411547e-39

- **해석**

P value가 0.05이하이면 target에 영향을 많이 미치고,
아니면 영향을 미치지 않는다.

- **Tag 변수 제외 영향이 없는 변수**

url_len, url_query_len,
url_num_query_para, url_entropy, url_port,

- **Tag 변수 중 영향이 없는 변수**

Iframe, head, body

하지만, 이를 보고 단편적으로 생각하면 안된다.

Config To Select

Config To Select

AIVLE

Config to logistic

```
if config['url_len_off'] == False :
    all_columns.remove('url_len')

if config['url_entropy_off'] == False :
    all_columns.remove('url_entropy')

if config['url_port_off'] == False :
    all_columns.remove('url_port')

if config['iframe_off'] == False :
    all_columns.remove("html_num_tags('iframe')")

if config['head_off'] == False :
    all_columns.remove("html_num_tags('head')")

if config['body_off'] == False :
    all_columns.remove("html_num_tags('body')")
```

- Config['url_len_off']

url_len의 칼럼을 학습에 고려할 것인지 안할 것인지 결정
(True : 고려한다, False : 고려하지 않는다.)

- Config['url_entropy_off']

url_entropy의 칼럼을 학습에 고려할 것인지 안할 것인지 결정
(True : 고려한다, False : 고려하지 않는다.)

- Config['url_post_off']

url_post의 칼럼을 학습에 고려할 것인지 안할 것인지 결정
(True : 고려한다, False : 고려하지 않는다.)

- Config['iframe_off']

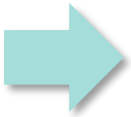
iframe의 칼럼을 학습에 고려할 것인지 안할 것인지 결정
(True : 고려한다, False : 고려하지 않는다.)

- Config['head_off']

head의 칼럼을 학습에 고려할 것인지 안할 것인지 결정
(True : 고려한다, False : 고려하지 않는다.)

- Config['body_off']

body의 칼럼을 학습에 고려할 것인지 안할 것인지 결정
(True : 고려한다, False : 고려하지 않는다.)



Logistic 분석에서 p value가 높게 나온 열들에 대한 취사 선택 여부 자동 결정

Config To Select

AIVLE

Config to One

```
# one_columns
one_columns = ['url_chinese_present',"html_num_tags('applet')"]

# one_distribute
one_distribute = ['url_query_len','url_num_query_para']
```

```
if config['one_column_on'] == False :
    for i in one_columns :
        all_columns.remove(i)
```

```
if config['one_distribute_on'] == False :
    for i in one_distribute :
        all_columns.remove(i)
```

- **설명**

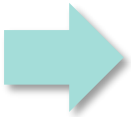
One_columns : 하나의 값만 가지는 칼럼. 의미가 없는 칼럼
One_distribute : error에서만 분포를 가지는 칼럼

- **Config['one_columns_on']**

하나의 값만 가지는 칼럼을 분석에 적용할 것인지 판단
(True : 적용한다, False : 버린다.)

- **Config['one_distribute_on']**

Error에서만 분포를 가지는 칼럼을 분석에 적용할 것인가
(True : 적용한다, False : 버린다.)



의미가 없어보이는 칼럼들을 자동으로 취사선택

Config To Select

Config to Scaler

```
if config['scaler'] == "MinMax" :
    scaler = MinMaxScaler()
    train_x = train.drop(columns = ["Result_v1"])
    column = train_x.columns
    train_y = train["Result_v1"]
    test_x = test.drop(columns = ["Result_v1"])
    test_y = test["Result_v1"]
    scaler = scaler.fit(train_x)
    train_x = scaler.transform(train_x)
    test_x = scaler.transform(test_x)
    train_x = pd.DataFrame(train_x)
    test_x = pd.DataFrame(test_x)
    train_x.columns = column
    test_x.columns = column
    train = pd.concat([train_x,train_y],axis = 1)
    test = pd.concat([test_x,test_y],axis = 1)
```

```
if config['scaler'] == "MaxAbs" :
    scaler = MaxAbsScaler()
    train_x = train.drop(columns = ["Result_v1"])
    column = train_x.columns
    train_y = train["Result_v1"]
    test_x = test.drop(columns = ["Result_v1"])
    test_y = test["Result_v1"]
    scaler = scaler.fit(train_x)
    train_x = scaler.transform(train_x)
    test_x = scaler.transform(test_x)
    train_x = pd.DataFrame(train_x)
    test_x = pd.DataFrame(test_x)
    train_x.columns = column
    test_x.columns = column
    train = pd.concat([train_x,train_y],axis = 1)
    test = pd.concat([test_x,test_y],axis = 1)
```

```
if config['scaler'] == "Standard" :
    scaler = StandardScaler()
    train_x = train.drop(columns = ["Result_v1"])
    column = train_x.columns
    train_y = train["Result_v1"]
    test_x = test.drop(columns = ["Result_v1"])
    test_y = test["Result_v1"]
    scaler = scaler.fit(train_x)
    train_x = scaler.transform(train_x)
    test_x = scaler.transform(test_x)
    train_x = pd.DataFrame(train_x)
    test_x = pd.DataFrame(test_x)
    train_x.columns = column
    test_x.columns = column
    train = pd.concat([train_x,train_y],axis = 1)
    test = pd.concat([test_x,test_y],axis = 1)
```

```
if config['scaler'] == "Robust" :
    scaler = RobustScaler()
    train_x = train.drop(columns = ["Result_v1"])
    column = train_x.columns
    train_y = train["Result_v1"]
    test_x = test.drop(columns = ["Result_v1"])
    test_y = test["Result_v1"]
    scaler = scaler.fit(train_x)
    train_x = scaler.transform(train_x)
    test_x = scaler.transform(test_x)
    train_x = pd.DataFrame(train_x)
    test_x = pd.DataFrame(test_x)
    train_x.columns = column
    test_x.columns = column
```

```
if config['scaler'] == "NONE" :
    train_x = train.drop(columns = ["Result_v1"])
    train_y = train["Result_v1"]
    test_x = test.drop(columns = ["Result_v1"])
    test_y = test["Result_v1"]
```

- **설명**

  MinMax, Standard, MaxAbs, Robust 스케일러중에서 무엇을 사용할 것인지 선택한다.

- **Config['scaler']**

  MinMax, Standard, MaxAbs, Robust,None 5개의 값을 선택할 수 있다.

하지만, 트리기반 모형에서는 쓸모가 없다.(스케일러가 의미 없다.)

Config To Select

AIVLE

Config to drop duplicate

```
if config['drop_duplicate'] == True :
    data.drop_duplicates(inplace = True)
    data.reset_index(drop = True, inplace = True)
```

- Config['drop_duplicate']

중복되는 값을 삭제 할 것인지를 판별한다.
(True : 삭제한다, False : 삭제하지 않는다.)

Config to MissForest

```
if config['missing_imput'] == True :
    X = data.drop(columns = ["Result_v1"])
    y = data["Result_v1"]
    imp = IterativeImputer(estimator = RandomForestRegressor(verbose=0, random_state = 42),
                           max_iter=10,
                           verbose = 0,
                           imputation_order= 'ascending',
                           random_state=0)
    X_imp=pd.DataFrame(imp.fit_transform(X))
    X_imp.columns = X.columns
    data = pd.concat([X_imp,y], axis = 1)
```

- 설명

결측치는 MissForest를 이용하여 채워주었다.
결측에 해당하는 칼럼을 제외한 나머지 칼럼을 가지고 결측값을 예측한다.
(정상과 불량간의 상관관계분포가 확연히 다르기 때문에 예측값의 편차가 클 것이므로 사용함)

- MissForest : RandomForest를 이용하여 결측값을 예측한 후, 결측치를 채움
  - Ascending : 결측값이 많은 순서대로 결측값을 채움

- Config['missing_imput']

MissForest를 사용할 것인지, 아니면 결측치를 삭제할 것인지 판단함.

대부분의 경우에, 중복치를 drop하지 않고, 결측치도 drop하지 않는 것이 성능이 높게 나옴

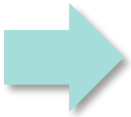
Config To Select

AIVLE

Iteration ALL CHOOSE for catboost

```
def iter_config(one_column,url_len,url_entropy,url_port,iframe,head,body,one_distribute,drop_duplicate,missing_imput,scaler) :
    config = dict()
    config['one_column_on'] = one_column
    config['url_len_off'] = url_len
    config['url_entropy_off'] = url_entropy
    config['url_port_off'] = url_port
    config['iframe_off'] = iframe
    config['head_off'] = head
    config['body_off'] = body
    config['one_distribute_on'] = one_distribute
    config['drop_duplicate'] = drop_duplicate
    config['missing_imput'] = missing_imput
    config['scaler'] = scaler

    train_x, train_y, test_x, test_y  = select_dataset(data,config,0.2)

    # catboost regressor
    model = CatBoostClassifier(random_state = 42, verbose = 0)
    model.fit(train_x, train_y)
    predict_y = model.predict(test_x)

    # f1 score
    f1 = metrics.f1_score(test_y, predict_y)

    return f1
```

모든 경우의 수를 생각하여 최고의 f1-score를 가려내자.



이 함수를 이용하여, 우리는 최고의 전처리 작업을 선택할 것이다.

Config To Select

AIVLE

Make iteration

```
one_column_select = [True,False]
url_len_select = [True,False]
url_entropy_select = [True,False]
url_port_select = [True,False]
iframe_select = [True,False]
head_select = [True,False]
body_select = [True,False]
one_distribute_select = [True,False]
drop_duplicate_select = [True,False]
missing_impute_select = [True,False]
scaler_select = ["NONE"]
```

```
iter = list(product(one_column_select,url_len_select,url_entropy_select,url_port_select,
iframe_select,head_select,body_select,one_distribute_select,
drop_duplicate_select,missing_impute_select,scaler_select))
```

```
result = []
for i in notebook.tqdm(iter) :
    time.sleep(0.01)
    output = iter_config(i[0],i[1],i[2],i[3],i[4],i[5],i[6],i[7],i[8],i[9],i[10])
    result.append(output)
    print(i,output)
```

```
,iter,f1
842,"(False, False, True, False, True, True, False, True, False, True, 'NONE')",0.9656084656084657
330,"(True, False, True, False, True, True, False, True, False, True, 'NONE')",0.9656084656084657
570,"(False, True, True, True, False, False, False, True, False, True, 'NONE')",0.9655172413793104
58,"(True, True, True, True, False, False, False, True, False, True, 'NONE')",0.9655172413793104
546,"(False, True, True, True, False, True, True, True, False, True, 'NONE')",0.9654255319148937
```

함수를 돌리면, 우리는 최고의 columns select와 imputation way, drop duplicate 여부를 판별가능하다.

이에 대한 결과로, 모델을 생성한다.

Config To Select

AIVLE

Make iteration

```
,iter,f1
842,"(False, False, True, False, True, True, False, True, False, True, 'NONE')",0.9656084656084657
```

- one_columns_select : 하나의 값만 가진 열은 제거하라.
- Url_len_select : url 길이에 해당하는 값을 제거하라.
- url_len_entropy_off : 엔트로피에 대한 값은 남겨둬라.
- Url_port_off : 포트에 해당하는 값은 버려라
- Iframe_select : iframe의 개수는 남겨둬라
- Head_select : head의 개수는 남겨둬라
- Body_select : body에 해당하는 값은 남겨둬라
- One_distribute_select : one_distribute에 해당하는 값은 남겨둬라
- Drop_duplicates : 중복되는 값은 남겨둬라
- Missing_imput : MissForest를 사용하라
- Scaler : 스케일러는 사용하지 않는다.

데이터 전처리가 완료되었다.

베이지안 옵티마이저

베이지안 옵티마이저

AIVLE

목적성 있는 파라미터 튜닝을 위하여

- 베이지안 옵티마이져는 어느 입력값을 받는 미지의 목적함수를 생성하여 해당 함수값을 최대로 만드는 최적해를 찾는 것을 목적으로 하는 파라미터 서치 기술
- 현재까지 조사된 입력값- 함수결과 값 점들을 바탕으로 미지의 목적 함수 형태에 대해 확률적으로 추정해나가면서 파리미터를 서칭
- 그리드 서치나 랜덤 서치처럼 어떠한 목적 없이 파라미터를 찾는 것이 아닌, 목적을 찾아가면서 파라미터를 적합시키기 때문에 정확한 파라미터 서칭 가능

Fitting Bayesian Optimization to CV5 CatBoost

```python
def fit_model(X,y,X_test,i,depth,bagging_temperature,learning_rate,subsample):
    cat_model = CatBoostClassifier(depth = int(depth),
                                   bagging_temperature=bagging_temperature,
                                   learning_rate=learning_rate,
                                   random_state=1339,#
                                   verbose=0,#
                                   subsample=subsample,
                                   eval_metric='F1'
                                   ).fit(X,y)
    cat_predict = cat_model.predict(X_test)
    return cat_predict
```

```python
def CAT_cv(depth,bagging_temperature,learning_rate,subsample):

    # KOLD CV
    kf=KFold(5,shuffle=True,random_state=1339)
    score = 0
    i=0
    for train_index,val_index in kf.split(X,y):
        i+=1
        X_train,X_val=X.iloc[train_index],X.iloc[val_index]
        y_train,y_val=y.iloc[train_index],y.iloc[val_index]
        cat_predict = fit_model(np.array(X_train),np.array(y_train),np.array(X_val),i,depth,bagging_temperature,learning_rate,subsample)
        score += metrics.accuracy_score(np.array(y_val),cat_predict)

    return score/5
```

베이지안 옵티마이저에 cross Validation을 사용하여 하이퍼 파라미터 서칭

베이지안 옵티마이저

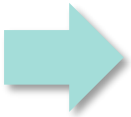
AIVLE

Result

```
pbounds = { 'depth': (1, 16),
            'bagging_temperature': (1, 10),
            'learning_rate': (0.01, 1.0),
            'subsample' : (0.01,1),
            }

# init_point = 5, n_iter = 12 --> 150 12*60 = 720분
bo = BayesianOptimization(f = CAT_cv, pbounds = pbounds, random_state = 1,verbose = 2)
bo.maximize(init_points = 5, n_iter = 10,acq = 'ei',xi = 0.01)
```

```
|   iter    |  target   | baggin... |   depth   | learni... | subsample |
-------------------------------------------------------------------------
| esc[0m 1      esc[0m | esc[0m 0.9667  esc[0m | esc[0m 4.753   esc[0m | esc[0m 11.8    esc[0m | esc[0m 0.01011 esc[0m | esc[0m 0.3093  esc[0m |
| esc[0m 2      esc[0m | esc[0m 0.9588  esc[0m | esc[0m 2.321   esc[0m | esc[0m 2.385   esc[0m | esc[0m 0.1944  esc[0m | esc[0m 0.3521  esc[0m |
| esc[0m 3      esc[0m | esc[0m 0.9662  esc[0m | esc[0m 4.571   esc[0m | esc[0m 9.082   esc[0m | esc[0m 0.425   esc[0m | esc[0m 0.6884  esc[0m |
| esc[0m 4      esc[0m | esc[0m 0.964   esc[0m | esc[0m 2.84    esc[0m | esc[0m 14.17   esc[0m | esc[0m 0.03711 esc[0m | esc[0m 0.6738  esc[0m |
| esc[95m 5      esc[0m | esc[95m 0.9686  esc[0m | esc[95m 4.756   esc[0m | esc[95m 9.38    esc[0m | esc[95m 0.149   esc[0m | esc[95m 0.2061  esc[0m |(BEST)
```

- 최고의 파라미터는 아래와 같다.
- Bagging_temperature =4.756
- Depth = 9
- Learning_rate = 0.149
- Subsample = 0.2061



하이퍼 파라미터 튜닝이 끝났다.

하지만,
이후에 어떤 경우의 수를 써도
성능이 좋아지지 않았다.

실험

실험

AIVLE

ALL COLUMNS USE

```
config = dict()
config['one_column_on'] =True
config['url_len_off'] =True
config['url_entropy_off'] =True
config['url_port_off'] =True
config['iframe_off'] =True
config['head_off'] =True
config['body_off'] =True
config['one_distribute_on'] =True
config['drop_duplicate'] =False
config['missing_imput'] =True
```

모든 칼럼을 사용하고 MISSFOREST로
결측을 보간한다.

베이지안 옵티마이징

iter	target	baggin...	depth	learni...	subsample
1	0.9686	4.753	11.8	0.01011	0.3093
2	0.9611	2.321	2.385	0.1944	0.3521
3	0.9632	4.571	9.082	0.425	0.6884
4	0.968	2.84	14.17	0.03711	0.6738
5	0.9658	4.756	9.38	0.149	0.2061
6	0.961	1.069	3.428	0.7123	0.7671
7	0.9686	5.675	14.44	0.4161	0.1487
8	0.9682	7.914	12.06	0.02387	0.1831

8번째 하이퍼 파라미터,
2순의 파라미터를 튜닝한다.

최종 스코어
0.95124

모든 칼럼에서 정보를 가지고 오는 경우가 가장 BEST임이 입증된다.

AIVLE

앙상블

```
cat1 = CatBoostClassifier(bagging_temperature =7.914, depth = 12, learning_rate = 0.02387, subsample = 0.1831,random_state=1339,verbose = 0,eval_metric='F1')
cat2 = CatBoostClassifier(bagging_temperature =5.675, depth = 14, learning_rate = 0.4161, subsample = 0.1487,random_state=1339,verbose = 0,eval_metric='F1')
```

```
from sklearn.ensemble import VotingClassifier
eclf = VotingClassifier(estimators=[
        ('cat1', cat1), ('cat2', cat2)],
        voting='soft')
```

최종 스코어
0.95247

BEST1모델과 BEST2모델을 앙상블시킨다.

앙상블 모형이 가장 뛰어나게 예측을 하였다.

감사합니다!