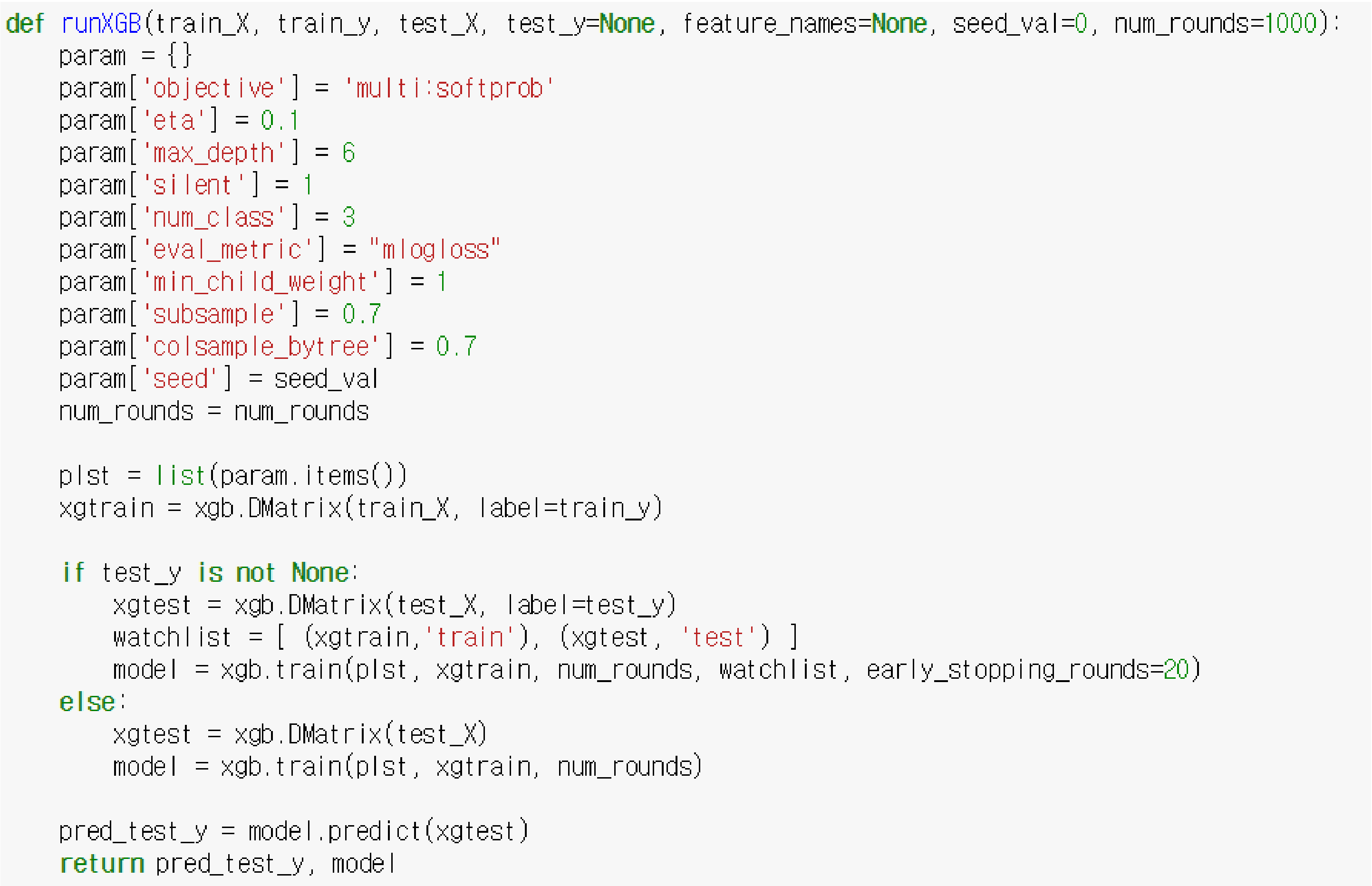
XGBoost with

Feature Engineering

서 예지
장 예훈
조 용걸

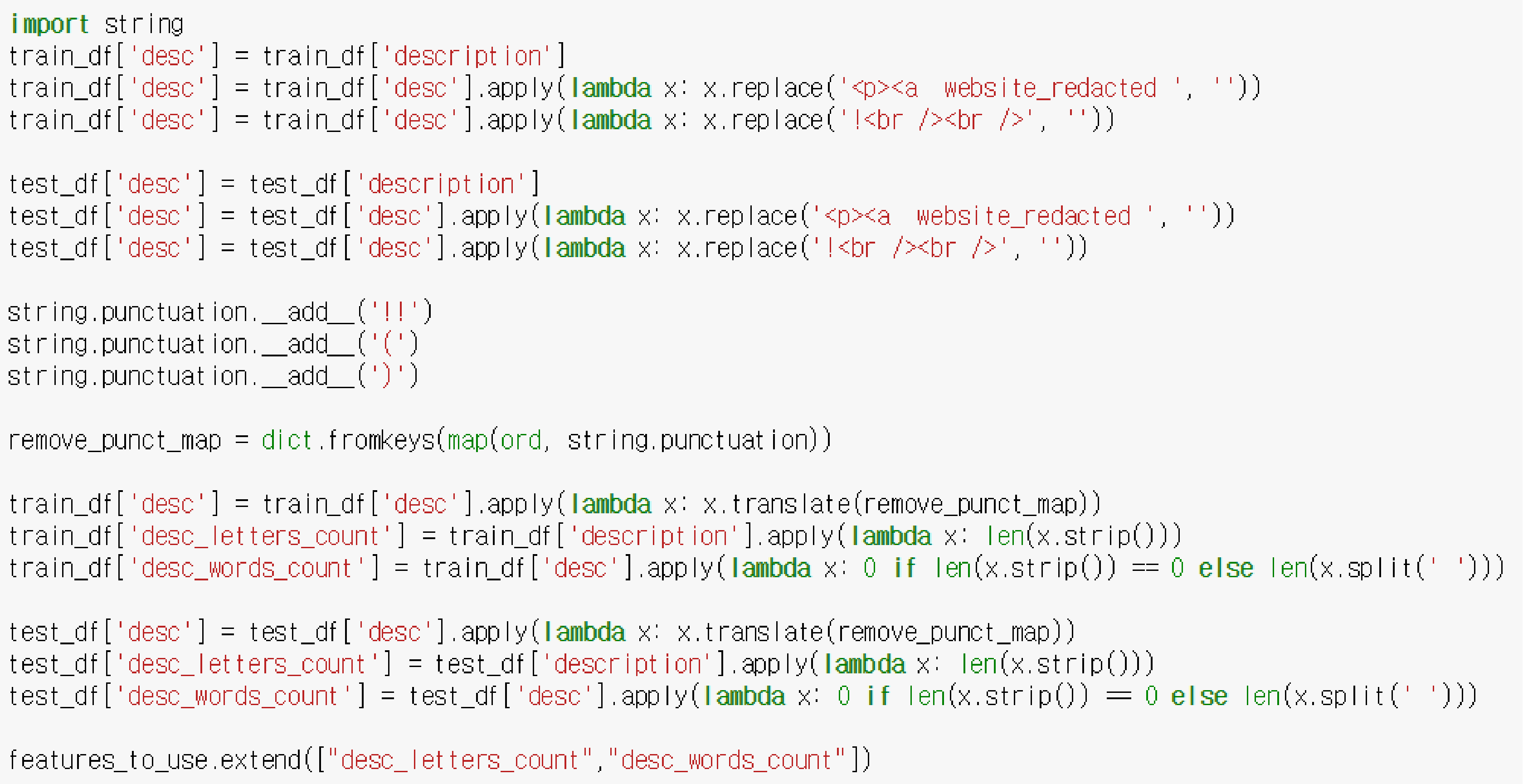
XGBoost



```
def runXGB(train_X, train_y, test_X, test_y=None, feature_names=None, seed_val=0, num_rounds=1000):
    param = {}
    param['objective'] = 'multi:softprob'
    param['eta'] = 0.1
    param['max_depth'] = 6
    param['silent'] = 1
    param['num_class'] = 3
    param['eval_metric'] = "mlogloss"
    param['min_child_weight'] = 1
    param['subsample'] = 0.7
    param['colsample_bytree'] = 0.7
    param['seed'] = seed_val
    num_rounds = num_rounds

    plst = list(param.items())
    xgtrain = xgb.DMatrix(train_X, label=train_y)

    if test_y is not None:
        xgtest = xgb.DMatrix(test_X, label=test_y)
        watchlist = [ (xgtrain,'train'), (xgtest, 'test') ]
        model = xgb.train(plst, xgtrain, num_rounds, watchlist, early_stopping_rounds=20)
    else:
        xgtest = xgb.DMatrix(test_X)
        model = xgb.train(plst, xgtrain, num_rounds)

    pred_test_y = model.predict(xgtest)
    return pred_test_y, model
```

> **XGBoost** is a recent implementation of Boosted Trees. It is a **machine learning algorithm** that yields great results on recent **Kaggle** competitions.

Feature Engineering

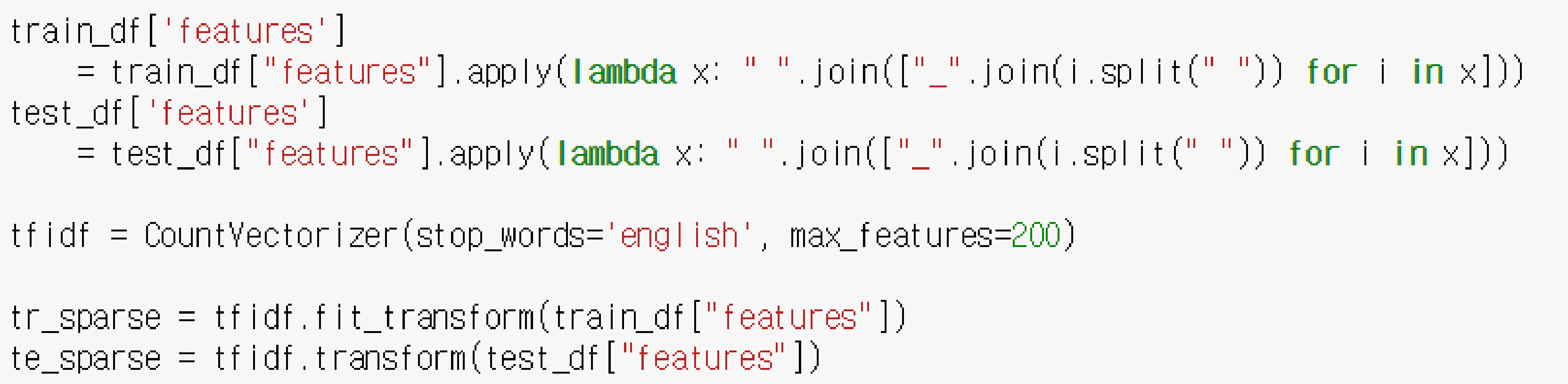


```
import string
train_df['desc'] = train_df['description']
train_df['desc'] = train_df['desc'].apply(lambda x: x.replace('<p><a  website_redacted ', ''))
train_df['desc'] = train_df['desc'].apply(lambda x: x.replace('!<br /><br />', ''))

test_df['desc'] = test_df['description']
test_df['desc'] = test_df['desc'].apply(lambda x: x.replace('<p><a  website_redacted ', ''))
test_df['desc'] = test_df['desc'].apply(lambda x: x.replace('!<br /><br />', ''))

string.punctuation.__add__('!!')
string.punctuation.__add__('(')
string.punctuation.__add__(')')

remove_punct_map = dict.fromkeys(map(ord, string.punctuation))

train_df['desc'] = train_df['desc'].apply(lambda x: x.translate(remove_punct_map))
train_df['desc_letters_count'] = train_df['description'].apply(lambda x: len(x.strip()))
train_df['desc_words_count'] = train_df['desc'].apply(lambda x: 0 if len(x.strip()) == 0 else len(x.split(' ')))

test_df['desc'] = test_df['desc'].apply(lambda x: x.translate(remove_punct_map))
test_df['desc_letters_count'] = test_df['description'].apply(lambda x: len(x.strip()))
test_df['desc_words_count'] = test_df['desc'].apply(lambda x: 0 if len(x.strip()) == 0 else len(x.split(' ')))

features_to_use.extend(["desc_letters_count","desc_words_count"])
```

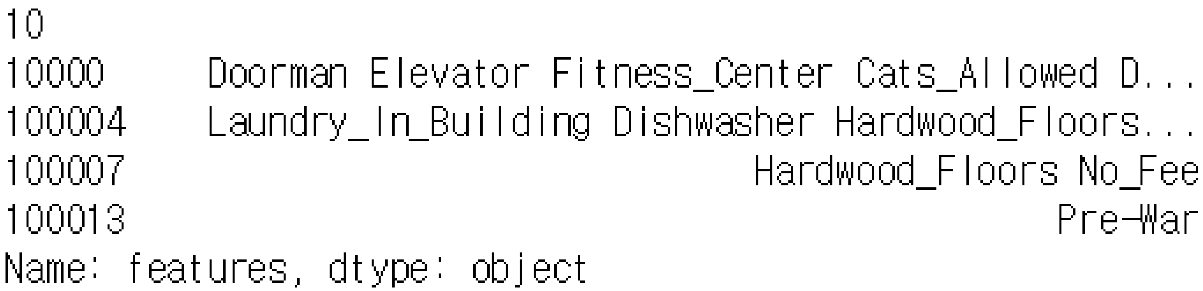
- Description에서
불필요한 단어, 문장 부호,
특수문자 등을 **제거**

- **단어 수를 Count**
이 때, 공백은 0 할당

Feature Engineering



```
train_df['features']
    = train_df["features"].apply(lambda x: " ".join(["_".join(i.split(" ")) for i in x]))
test_df['features']
    = test_df["features"].apply(lambda x: " ".join(["_".join(i.split(" ")) for i in x]))

tfidf = CountVectorizer(stop_words='english', max_features=200)

tr_sparse = tfidf.fit_transform(train_df["features"])
te_sparse = tfidf.transform(test_df["features"])
```

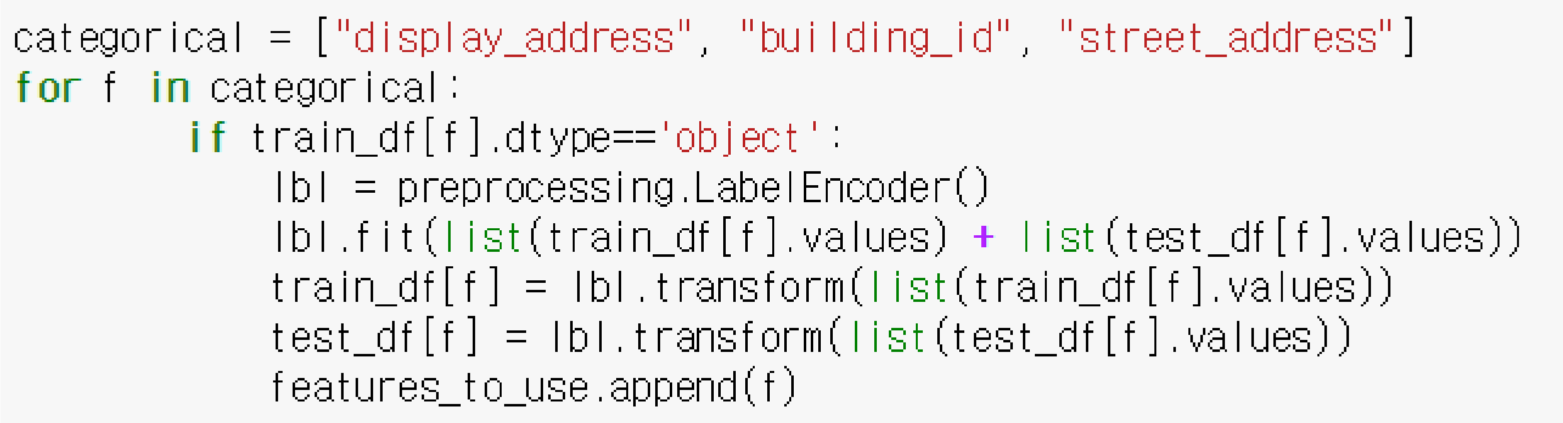


```
10
10000     Doorman Elevator Fitness_Center Cats_Allowed D...
100004    Laundry_In_Building Dishwasher Hardwood_Floors...
100007                             Hardwood_Floors No_Fee
100013                                            Pre-War
Name: features, dtype: object
```

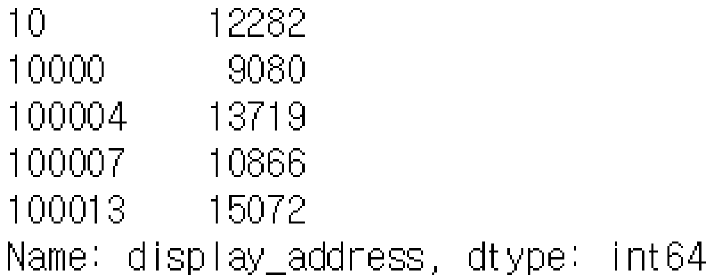
- [“features”]를 CountVectorize함

- 불필요한 200개의 단어 제거

Feature Engineering



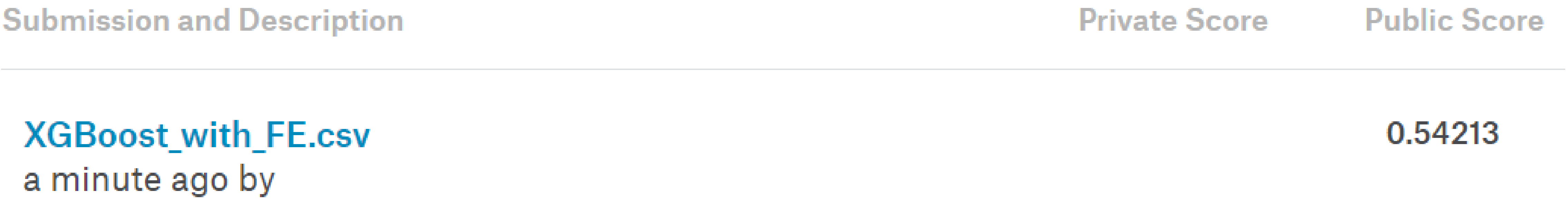
```
categorical = ["display_address", "building_id", "street_address"]
for f  in categorical:
        if train_df[f].dtype=='object':
            lbl = preprocessing.LabelEncoder()
            lbl.fit(list(train_df[f].values) + list(test_df[f].values))
            train_df[f] = lbl.transform(list(train_df[f].values))
            test_df[f] = lbl.transform(list(test_df[f].values))
            features_to_use.append(f)
```



```
10         12282
10000       9080
100004     13719
100007     10866
100013     15072
Name: display_address, dtype: int64
```

- 주소, 빌딩 id등의 feature들을 categorize함

Score



Submission and Description	Private Score	Public Score
XGBoost_with_FE.csv a minute ago by		0.54213

LogLoss : 0.52612686525190266

Score : 0.54213

THANK

YOU

☺