DACON
DATA TO VALUE

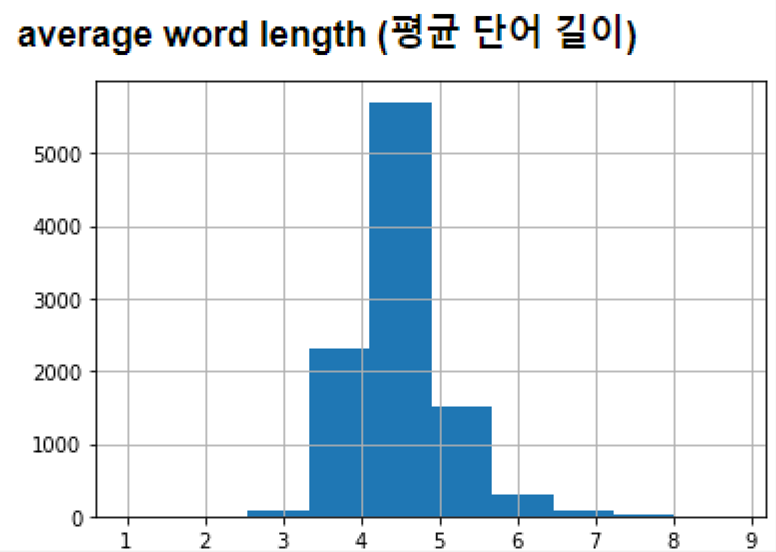
NLP
소설 작가 분류

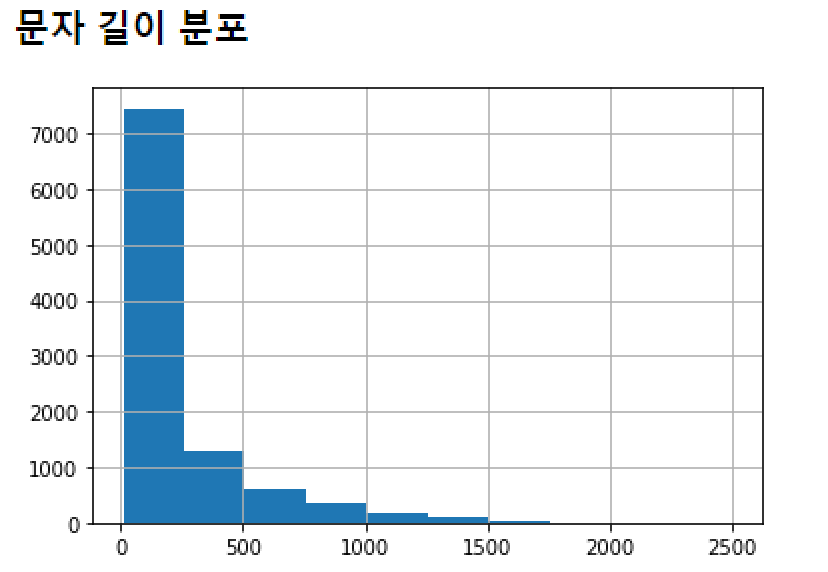
팀명 : Old Pine
응용통계학과 노승찬
응용통계학과 송준영

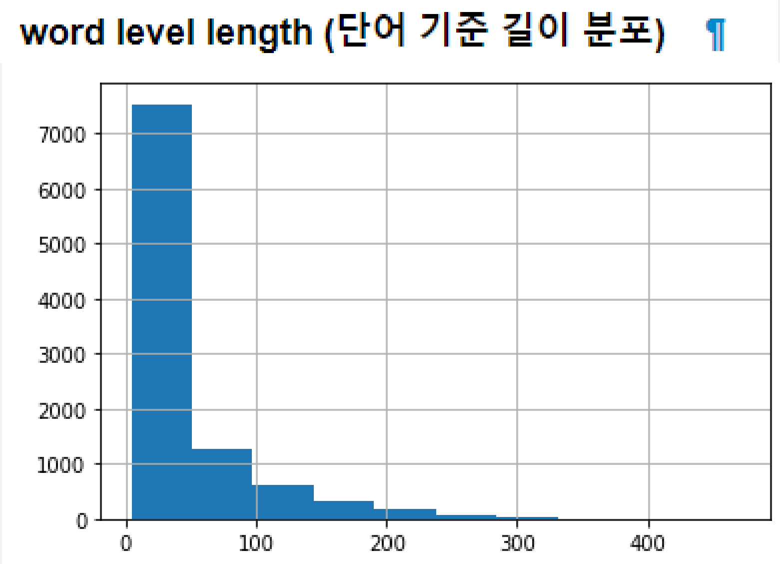
목차

EDA

텍스트의 분포에 대한 EDA







이러한 수치들이 모델링에 관여하면 **성능이 향상**될까?

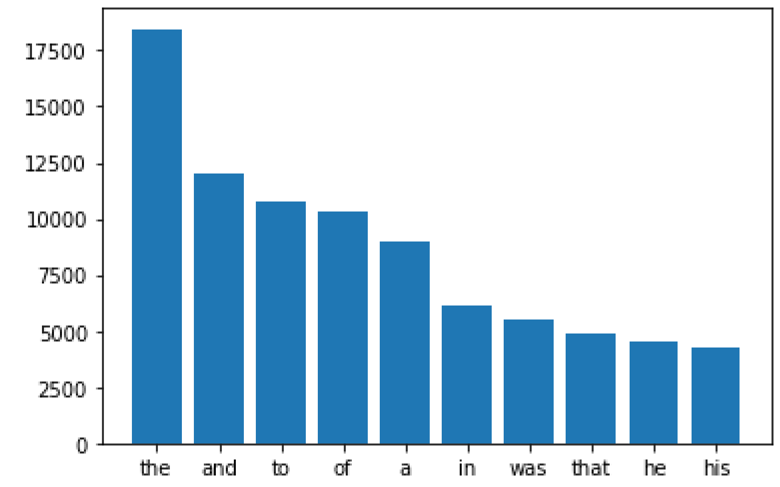
stopwords 분포

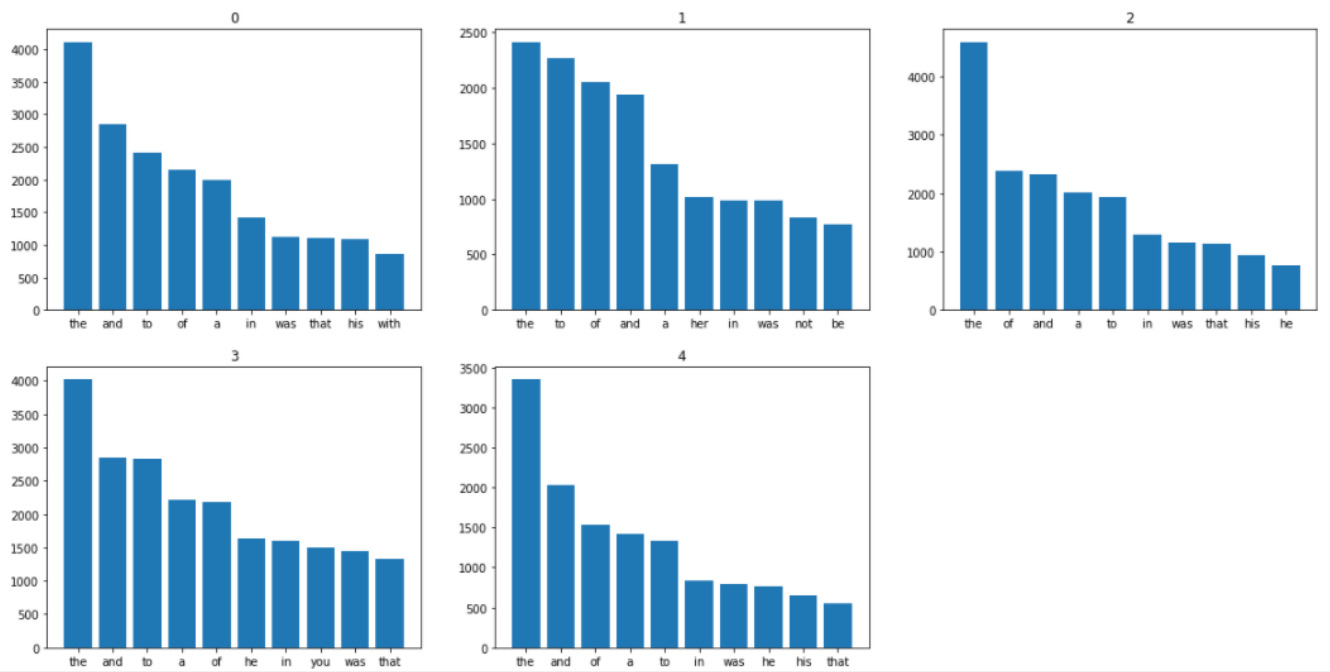
```python
def plot_top_stopwords_barchart(text):
    stop=set(stopwords.words('english'))

    new= text.str.split()
    new=new.values.tolist()
    corpus=[word for i in new for word in i]
    from collections import defaultdict
    dic=defaultdict(int)
    for word in corpus:
        if word in stop:
            dic[word]+=1

    top=sorted(dic.items(), key=lambda x:x[1],reverse=True)[:10]
    x,y=zip(*top)
    plt.bar(x,y)

plot_top_stopwords_barchart(train['text'])
```





작가별로 **자주 사용하는 STOPWORDS**가 있지 않을까?

작가별 **STOPWORDS의 빈도**로 작가를 파악하는 것에 도움이 되지 않을까?

MODELING

전처리

모델링

앙상블

A

불용어 제거 x
+
Embedding

>> Model 1 : RNN(LSTM + GRU)

>> Model 2 : CNN

B

추가 피처
+
TF-IDF
+
Embedding

>> Model 3 : Light GBM

>> Model 4 : XGBoost

Stacking(메타 러너 : Light GBM)

A. RNN & CNN

전처리

```python
#알파벳과 숫자를 제외한 나머지 제거
def alpha_num(text):
    return re.sub(r'[^A-Za-z0-9,! ]', '', text)

#불용어 제거
def remove_stopwords(text):
    final_text = []
    for i in text.split():
        if i.strip().lower() not in stopwords:
            final_text.append(i.strip())
    return " ".join(final_text)

stopwords = [
            #  "the","a", "about", "above", "after", "again", "against", "all", "am", "an", "and", "any", "are", "as",
            #  "at", "be", "because", "been", "before", "being", "below", "between", "both", "but", "by", "could",
            #  "did", "do", "does", "doing", "down", "during", "each", "few", "for", "from", "further", "had", "has",
            #  "have", "having", "he", "he'd", "he'll", "he's", "her", "here", "here's", "hers", "herself", "him", "himself",
            #  "his", "how", "how's", "i", "i'd", "i'll", "i'm", "i've", "if", "in", "into", "is", "it", "it's", "its", "itself",
            #  "let's", "me", "more", "most", "my", "myself", "nor", "of", "on", "once", "only", "or", "other", "ought", "our", "ours",
            #  "ourselves", "out", "over", "own", "same", "she", "she'd", "she'll", "she's", "should", "so", "some", "such", "than", "that",
            #  "that's", "the", "their", "theirs", "them", "themselves", "then", "there", "there's", "these", "they", "they'd", "they'll",
            #  "they're", "they've", "this", "those", "through", "to", "too", "under", "until", "up", "very", "was", "we", "we'd", "we'll",
            #  "we're", "we've", "were", "what", "what's", "when", "when's", "where", "where's", "which", "while", "who", "who's", "whom",
            #  "why", "why's", "with", "would", "you", "you'd", "you'll", "you're", "you've", "your", "yours", "yourself", "yourselves"
             ]
```

```python
#작가 분류이기 때문에 불용어를 제거 하지 않는 것이 성능향상에 도움됨

# train['text'] = train['text'].str.lower()
# test['text'] = test['text'].str.lower()
# train['text'] = train['text'].apply(alpha_num)
# .apply(remove_stopwords)
# test['text'] = test['text'].apply(alpha_num)
# .apply(remove_stopwords)
```

〉〉 Model 1 : RNN(LSTM + GRU)

설명

- RNN의 경우 어떠한 전처리도 하지 않음

- CNN은 특수문자만 제거

RNN

```
def get_model():
    #  with tf.device('/device:GPU:0'):
        model = Sequential([
            Embedding(vocab_size, embedding_dim, input_length=max_length),
            # Embedding(47256, 300, weights=[embedding_matrix], input_length=max_length, trainable=False),
            Bidirectional(GRU(16,activation='tanh', return_sequences=True)),
            # Dropout(0.1),
            Bidirectional(LSTM(16)),
            Dense(n_class, activation='softmax')
        ])
        model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=.005))
        return model
```

```
p_val = np.zeros((trn.shape[0], n_class))
p_tst = np.zeros((tst.shape[0], n_class))
for i, (i_trn, i_val) in enumerate(cv.split(trn, y), 1):
    print(f'training model for CV #{i}')
    clf = get_model()

    es = EarlyStopping(monitor='val_loss', min_delta=0.001, patience=2,
                       verbose=1, mode='min', baseline=None, restore_best_weights=True)

    # rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,
    #                         patience=2, min_lr=1e-6, mode='min', verbose=1)

    clf.fit(trn[i_trn],
            to_categorical(y[i_trn]),
            validation_data=(trn[i_val], to_categorical(y[i_val])),
            epochs=30,
            callbacks=[es
                       #  ,rlr
                       ]
            )
    p_val[i_val, :] = clf.predict(trn[i_val])
    p_tst += clf.predict(tst) / n_fold
```

〉〉 **Model 1 : RNN(LSTM + GRU)**

설명

- 임베딩층과 GRU층 LSTM층으로 구성
- Grid search로 노드의 개수 탐색
- Early Stopping 사용
- 마지막 output layer의 활성화 함수는 ‘soft max’

CNN

```python
def get_model():
    # with tf.device('/device:GPU:0'):
    model = Sequential([
        Embedding(vocab_size, embedding_dim, input_length=max_length),
        SpatialDropout1D(0.7),
        Conv1D(32, 8, padding="valid", activation="relu", strides=3),
        # Dropout(.3),
        Conv1D(32, 8, padding="valid", activation="relu", strides=3),
        GlobalMaxPooling1D(),
        # Dense(64, activation='relu'),
        Dense(n_class, activation='softmax')
    ])
    model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=.01))
    return model
```

```python
p_val = np.zeros((trn.shape[0], n_class))
p_tst = np.zeros((tst.shape[0], n_class))
for i, (i_trn, i_val) in enumerate(cv.split(trn, y), 1):
    print(f'training model for CV #{i}')
    clf = get_model()
    es = EarlyStopping(monitor='val_loss', min_delta=0.001, patience=5,
                       verbose=1, mode='min', baseline=None, restore_best_weights=True)
    rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,
                            patience=3, min_lr=1e-6, mode='min', verbose=1)
    clf.fit(trn[i_trn],
            to_categorical(y[i_trn]),
            validation_data=(trn[i_val], to_categorical(y[i_val])),
            epochs=100,
            batch_size=512,
            callbacks=[es
                    #  ,rlr
                    ]
            )
    p_val[i_val, :] = clf.predict(trn[i_val])
    p_tst += clf.predict(tst) / n_fold
```

》 Model 2 :CNN

설명

- 베이스 라인의 CNN 코드에서 SpatialDropout1D을 추가하여 과적합 방지 및 성능 향상 도모.

(노드를 드랍해버리는 것이 아닌 몇몇 컬럼 자체를 드랍)

B. LightGBM & XGBoost

변수 추가

```python
## Number of words in the text ##
train["num_words"] = train["text"].apply(lambda x: len(str(x).split()))
test["num_words"] = test["text"].apply(lambda x: len(str(x).split()))

## Number of unique words in the text ##
train["num_unique_words"] = train["text"].apply(lambda x: len(set(str(x).split())))
test["num_unique_words"] = test["text"].apply(lambda x: len(set(str(x).split())))

## Number of characters in the text ##
train["num_chars"] = train["text"].apply(lambda x: len(str(x)))
test["num_chars"] = test["text"].apply(lambda x: len(str(x)))

## Number of stopwords in the text ##
train["num_stopwords"] = train["text"].apply(lambda x: len([w for w in str(x).lower().split() if w in stop_words]))
test["num_stopwords"] = test["text"].apply(lambda x: len([w for w in str(x).lower().split() if w in stop_words]))

## Number of punctuations in the text ##
train["num_punctuations"] =train['text'].apply(lambda x: len([c for c in str(x) if c in string.punctuation]) )
test["num_punctuations"] =test['text'].apply(lambda x: len([c for c in str(x) if c in string.punctuation]) )

## Number of title case words in the text ##
train["num_words_upper"] = train["text"].apply(lambda x: len([w for w in str(x).split() if w.isupper()]))
test["num_words_upper"] = test["text"].apply(lambda x: len([w for w in str(x).split() if w.isupper()]))

## Number of title case words in the text ##
train["num_words_title"] = train["text"].apply(lambda x: len([w for w in str(x).split() if w.istitle()]))
test["num_words_title"] = test["text"].apply(lambda x: len([w for w in str(x).split() if w.istitle()]))

## Average length of the words in the text ##
train["mean_word_len"] = train["text"].apply(lambda x: np.mean([len(w) for w in str(x).split()]))
test["mean_word_len"] = test["text"].apply(lambda x: np.mean([len(w) for w in str(x).split()]))
```

설명

- 아래와 같은 변수 추가
  - 단어 개수
  - 불용어 개수
  - 글자 개수
  - 특수문자 개수
  - 대문자 개수
  - 단어 길이의 평균
- Kaggle 내에 있는 프로젝트 참고

Feature Engineering

```python
def runMNB(train_X, train_y, test_X, test_y, test_X2):
    model = naive_bayes.MultinomialNB()
    model.fit(train_X, train_y)
    pred_test_y = model.predict_proba(test_X)
    pred_test_y2 = model.predict_proba(test_X2)
    return pred_test_y, pred_test_y2, model
```

```python
### Fit transform the tfidf vectorizer ###
tfidf_vec = TfidfVectorizer(ngram_range=(1,5), analyzer='char')
full_tfidf = tfidf_vec.fit_transform(train_df['text'].values.tolist())
train_tfidf = tfidf_vec.transform(train_df['text'].values.tolist())
test_tfidf = tfidf_vec.transform(test_df['text'].values.tolist())

cv_scores = []
pred_full_test = 0
pred_train = np.zeros([train_df.shape[0], 5])
kf = model_selection.KFold(n_splits=5, shuffle=True, random_state= 42)
for dev_index, val_index in kf.split(train_X):
    dev_X, val_X = train_tfidf[dev_index], train_tfidf[val_index]
    dev_y, val_y = train_y[dev_index], train_y[val_index]
    pred_val_y, pred_test_y, model = runMNB(dev_X, dev_y, val_X, val_y, test_tfidf)
    pred_full_test = pred_full_test + pred_test_y
    pred_train[val_index,:] = pred_val_y
    cv_scores.append(metrics.log_loss(val_y, pred_val_y))
print("Mean cv score : ", np.mean(cv_scores))
pred_full_test = pred_full_test / 5.

# add the predictions as new features #
train_df["nb_tfidf_char_eap"] = pred_train[:,0]
train_df["nb_tfidf_char_hpl"] = pred_train[:,1]
train_df["nb_tfidf_char_mws"] = pred_train[:,2]
test_df["nb_tfidf_char_eap"] = pred_full_test[:,0]
test_df["nb_tfidf_char_hpl"] = pred_full_test[:,1]
test_df["nb_tfidf_char_mws"] = pred_full_test[:,2]
```

```python
n_comp = 20
svd_obj = TruncatedSVD(n_components=n_comp, algorithm='arpack')
svd_obj.fit(train_tfidf)
train_svd = pd.DataFrame(svd_obj.transform(train_tfidf))
test_svd = pd.DataFrame(svd_obj.transform(test_tfidf))

train_svd.columns = ['svd_char_'+str(i) for i in range(n_comp)]
test_svd.columns = ['svd_char_'+str(i) for i in range(n_comp)]
train_df = pd.concat([train_df, train_svd], axis=1)
test_df = pd.concat([test_df, test_svd], axis=1)
del full_tfidf, train_tfidf, test_tfidf, train_svd, test_svd
```

- 나이브 베이즈 모델의 예측값을 피처로 추가
- 정보를 압축하고 컴펙트하게 표현하기 위해 SVD 사용하여 차원축소
- TF-IDF에 svd 형상을 만들어 특징 세트에 추가
- Kaggle에서 소스 필사. 의미에 대한 학습 필요

Light GBM

```
[ ]
def runLGBM(train_X, train_y, test_X, test_y=None, test_X2=None, seed_val = 42, child=1):
    param = {}
    param['objective'] = 'multiclass'
    param['boosting_type'] = 'gbdt'
    param['subsample_freq'] = 5
    param['max_depth'] = 30
    param['num_leaves'] = 100
    param['num_class'] = 5
    param['colsample_bytree']=0.7
    param['subsample'] = 0.8
    param['min_data_in_leaf'] = 64
    param['metric'] = 'multi_logloss'
    param['subsample_for_bin'] = 23000
    param['min_child_weight'] = child
    param['learning_rate'] = 0.01
    param['seed'] = seed_val
    n_estimators = 20000

    # plst = param.items()
    lgbmtrain = lgbm.Dataset(train_X, label=train_y)

    if test_y is not None:
        lgbmtest = lgbm.Dataset(test_X, label=test_y)
        # watchlist = [ (lgbmtrain,'train'), (lgbmtest, 'test') ]
        model = lgbm.train(param, lgbmtrain, n_estimators,valid_sets= [lgbmtrain,lgbmtest], early_stopping_rounds=50, verbose_eval= 100)
    else:
        lgbmtest = lgbm.Dataset(test_X)
        model = lgbm.train(plst, lgbmtrain, num_rounds)

    pred_test_y = model.predict(test_X)
    if test_X2 is not None:
        pred_test_y2 = model.predict(test_X2)
    return pred_test_y, pred_test_y2, model
```

>> Model 3

XGBoost

```
def runXGB(train_X, train_y, test_X, test_y=None, test_X2=None, seed_val = 42, child=1, colsample=0.3):
    param = {}
    param['objective'] = 'multi:softprob'
    param['eta'] = 0.1
    param['max_depth'] = 10
    param['silent'] = 1
    param['booster'] = 'dart'
    param['num_class'] = 5
    param['eval_metric'] = "mlogloss"
    param['min_child_weight'] = child
    param['subsample'] = 0.8
    param['colsample_bytree'] = colsample
    param['seed'] = seed_val
    param['tree_method'] = 'gpu_hist'
    num_rounds = 20000

    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=50, verbose_eval= 100)
    else:
        xgtest = xgb.DMatrix(test_X)
        model = xgb.train(plst, xgtrain, num_rounds)

    pred_test_y = model.predict(xgtest, ntree_limit = model.best_ntree_limit)
    if test_X2 is not None:
        xgtest2 = xgb.DMatrix(test_X2)
        pred_test_y2 = model.predict(xgtest2, ntree_limit = model.best_ntree_limit)
    return pred_test_y, pred_test_y2, model
```

>> Model 5

스태킹(메타 러너 : Light GBM)

```
#  p_trn =np.zeros((stk_trn.shape[0], n_class))
p_val = np.zeros((stk_trn.shape[0], n_class))
p_tst = np.zeros((stk_tst.shape[0], n_class))
for i, (i_trn, i_val) in enumerate(cv.split(stk_trn, y), 1):
    print(f'training model for CV #{i}')
    clf = lgb.LGBMClassifier(objective='multiclass',
                             n_estimators=10000,
                             learning_rate=0.01,
                             boosting_type ='gbdt',
                              max_depth=5,
                               feature_fraction=0.4,
#                               min_child_weight=0.01,
                              num_leaves=30,
                             random_state=seed,
                             n_jobs=-1,
                             verbose=100)
    clf.fit(stk_trn[i_trn], y[i_trn],
            eval_set=[(stk_trn[i_val], y[i_val])],
            eval_metric='multi_logloss',early_stopping_rounds=100,
            verbose=100)
#     p_trn[i_trn, :] = clf.predict_proba(stk_trn[i_trn])
    p_val[i_val, :] = clf.predict_proba(stk_trn[i_val])
    p_tst += clf.predict_proba(stk_tst) / n_fold
print()
print('models:',model_names)
print(clf)
# print(f'train cv accuracy : {accuracy_score(y, np.argmax(p_trn, axis=1)) :.6f}')
print(f'valid cv accuracy : {log_loss(y, p_val) :.6f}')
```

설명

- Grid search를 통해 parameter tuning
- XGBoost, DNN, RandomForest를 시도하였으나 LihgtGBM의 경우가 성능 가장 좋음.
- 최종 스태킹 CV Logloss : 0.41…

전처리

모델링

앙상블

A

불용어 제거 x
+
Embedding

>> Model 1 : RNN(LSTM + GRU)

>> Model 2 : CNN

B

추가 피처
+
TF-IDF
+
Embedding

>> Model 3 : Light GBM

>> Model 4 : XGBoost

Stacking(메타 러너 : Light GBM)

결과 및 느낀 점

최종 스코어(private 기준)
log loss : 0.23471(27위)

- Pre-trained embedding을 사용하였으나 성능 하락.
- 오히려 Lemmatization, 불용어 제거 등의 전처리하지 않은 데이터셋의 성능이 더 우수.(RNN 기준)
- **어텐션에** 대한 시도를 못해본 것이 아쉬움.
- NLP관련 지식이 부족하여 이론적으로 접근하는 것이 아닌 다양한 방법을 모두 시도하여 시간적으로 많은 투자가 있었던 것이 아쉬움.
- NLP 분야에 입문하는 것에 큰 도움이 되었음.

참고 사이트

https://www.kaggle.com/sudalairajkumar/simple-feature-engg-notebook-spooky-author

https://jiho-ml.com/

참고 서적

핸즈온 머신러닝 : 사이킷런, 케라스, 텐서플로를 활용한 머신러닝, 딥러닝 완벽 실무

텐서플로와 머신러닝으로 시작하는 자연어 처리

한 학기간 좋은 수업과 강의 제공해주셔서

진심으로 감사드립니다.