# Week8

| | |
|---|---|
| Created By | DongGu Kim |
| Last Edited | @May 08, 2020 4:10 PM |
| Property | |
| Tags | |

# LGBM 모델링 결과 점검

# SVM 모델링 결과 점검

## 결론

## Random Seed 30번 추가한 모델링 코드

- Week7에 나온 대로 LGBM은 99.6% SVM은 99.7%의 정확도를 기록했습니다. 이 결과가 우연하게 좋게 나온 것인지 점검하기 위해서 Random Seed를 30번씩 반복해서 모델의 성능을 다시 점검했습니다.

---

## 1. LGBM 모델링 결과 점검

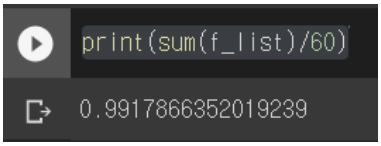
1. 소요시간 (Random Seed list 30번 * list size 3 * kfold 4 = 36개 모델링 소요시간)
    - 약 70분 * 30개 = 2100분
    - 코랩 가동 중단 때문에 '%%time'으로 나온 수치는 따로 없음
    - 대신 random seed size 3 * kfold4 = 12개 모델링 소요시간인 70분을 30으로 곱함

2. F-measure 비교

```
lgbm_results = os.listdir('lgbm_models/submissions/')[:30]

f_list = list()
for result in lgbm_results:
    submission = pd.read_csv('lgbm_models/submissions/'+result)
    result = classification_report(Y_test, submission['pred'], target_names=['normal','smishing'], output_dict=True)
    f_list.append(result['smishing']['f1-score'])
    f_list.append(result['normal']['f1-score'])

print(sum(f_list)/60)
```




## 2. SVM 모델링 결과 점검

1. 소요시간 (Random Seed list 30번 * list size 3 * kfold 4 = 36개 모델링 소요시간)

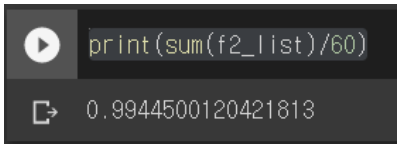
```
CPU times: user 32min 1s, sys: 20min 52s, total: 52min 53s
Wall time: 28min 27s
```

2. F-measure 비교

```
svm_results = os.listdir('svm_models/submissions/')[:30]

f2_list= list()
for result in svm_results:
    submission = pd.read_csv('svm_models/submissions/'+result)
    result = classification_report(Y_test, submission['pred'], target_names=['normal','smishing'], output_dict=True)
    f2_list.append(result['smishing']['f1-score'])
    f2_list.append(result['normal']['f1-score'])

print(sum(f2_list)/60)
```




## 3. 결론

- 정확도가 99%이상 나온 것이 우연인지..?
  - 같은 모델링을 30번 반복한 평균도 99%이상으로 기록되었으므로 이는 우연이라 보기 힘들다.

- 속도적 측면
  - 동일한 모델링을 SVM은 LGBM에 비해 30배 빠르게 진행했다.

- f-measure 측면
  - 미세하지만 (약 0.0025) SVM이 더 정확했다.

- 최종 결론
  - 속도, f-measure에서 더 우수한 SVM을 스미싱 여부를 가려주는 챗봇의 모델로 채택했다.

- 다음 계획
  - 원래 딥러닝 기반 LSTM 모델링을 하려했으나, LIME으로 넘어갈 예정이다.
  - LGBM, SVM이 빠른 속도로 우수한 성능을 내주고 있기 때문이다.
  - LIME을 우선적으로 수행하면서 LSTM 기반 모델링도 구현해볼 것이다.

## 4. Random Seed 30번 추가한 모델링 코드

1) LightGBM(LGBM)

```
%%time
# 4FOLD, 3SEED ENSEMBLE
# 총 12개의 모델을 평균내어 예측한다
# classification_repory
```

```
result_list = list()
result2_list = list()

# 30번의 랜덤시드 설정
for try_num in range(30):
    lucky_seed=[]
    for i in range(3):
        lucky_seed.append(random.randint(1,10000))

    # enumerate: 인덱스와 값을 둘다 반복시킬 때 사용
    for num,rs in tqdm(enumerate(lucky_seed)):

        kfold = KFold(n_splits=4, random_state = rs, shuffle = True)

        # numpy.zeros((row,col))
        # row*col size 영행렬 생성
        cv=np.zeros((X_train.shape[0],2))

        for n, (train_idx, validation_idx) in tqdm(enumerate(kfold.split(X_train))):
            print(train_idx, validation_idx)
            x_train, x_validation = X_train[train_idx], X_train[validation_idx]
            y_train, y_validation = Y_train.loc[train_idx], Y_train.loc[validation_idx]

            lgbm = LGBMClassifier(n_estimators=380,
                        learning_rate=0.035,
                        max_depth=7,
                        min_child_samples=50,
                        random_state=4321)

            lgbm.fit(x_train, y_train, eval_set=[(x_validation, y_validation)], early_stopping_rounds= 30, verbose=100)

            # 모델결과 저장 lib
            if not(os.path.isdir('lgbm_models/%s/'%(str(lucky_seed)))):
                os.makedirs(os.path.join('lgbm_models/%s/'%(str(lucky_seed))))
            joblib.dump(lgbm, 'lgbm_models/%s/%s_fold_model_%s.pkl'%(str(lucky_seed), n, rs))

            # numpy.zeros((row,col))로 만들어주었던 영행렬: cv
            # data object에 X_validation 예측 값을 넣어줌
            # CROSS-VALIDATION , EVALUATE CV
            cv[validation_idx,:] = lgbm.predict_proba(x_validation)


    # MODEL LOAD & TEST PREDICT
    # 12 MODELS 평균 사용
    models = os.listdir('lgbm_models/%s/'%(str(lucky_seed)))
    models_list = [x for x in models if x.endswith(".pkl")]

    # 모델결과가 잘 나왔는지 check
    # assert: 좌항과 우항의 값이 같으면 정상 작동, 다르면 오류 발생
    assert len(models_list) ==12


    temp_predictions = np.zeros((X_test.shape[0],2))

    # 12개 모델을 반복시켜서 결과산출 -> 12로 나눠서 평균값 계산
    for model in models_list:
        model = joblib.load('lgbm_models/%s/'%(str(lucky_seed))+model)
        predict_proba = model.predict_proba(X_test)
        temp_predictions += predict_proba/12


    submission = pd.DataFrame(data=np.zeros((X_test.shape[0],2)))
    submission.index = Y_test.index
    submission.index.name = 'id'
    submission+=temp_predictions

    submission = submission.sort_index()
    submission = submission.groupby('id').mean()

    submission['pred'] = 0
    for idx in submission.index:
        if (submission[0][idx] < submission[1][idx]):
            submission['pred'][idx]= 1

    submission.to_csv('lgbm_models/submissions/%s_submission.csv'%(str(lucky_seed)), index=True)

    result = classification_report(Y_test, submission['pred'], target_names=['normal','smishing'])
    print(result)
    result_list.append(result)
    print('')
    result2 = accuracy_score(Y_test, submission['pred'])
```

```
    print(result2)
    result2_list.append(result2)
```

2) Support Vector Machine (SVM)

```
%%time
# 4FOLD, 3SEED ENSEMBLE
# 총 12개의 모델을 평균내어 예측한다
# classification_repory
result_list = list()
result2_list = list()

# 30번의 랜덤시드 설정
for try_num in range(30):
    lucky_seed=[]
    for i in range(3):
        lucky_seed.append(random.randint(1,10000))

    # enumerate: 인덱스와 값을 둘다 반복시킬 때 사용
    for num,rs in tqdm(enumerate(lucky_seed)):

        kfold = KFold(n_splits=4, random_state = rs, shuffle = True)

        # numpy.zeros((row,col))
        # row*col size 영행렬 생성
        cv=np.zeros((X_train.shape[0],2))

        for n, (train_idx, validation_idx) in tqdm(enumerate(kfold.split(X_train))):
            x_train, x_validation = X_train[train_idx], X_train[validation_idx]
            y_train, y_validation = Y_train.loc[train_idx], Y_train.loc[validation_idx]

            model_svc = LinearSVC(class_weight='balanced',
                              random_state=4321)
            svm_model = CalibratedClassifierCV(model_svc)
            svm_model.fit(x_train, y_train)

            # 모델결과 저장 lib
            if not(os.path.isdir('svm_models/%s/'%(str(lucky_seed)))):
                os.makedirs(os.path.join('svm_models/%s/'%(str(lucky_seed))))
            joblib.dump(svm_model, 'svm_models/%s/%s_fold_model_%s.pkl'%(str(lucky_seed), n, rs))

            # numpy.zeros((row,col))로 만들어주었던 영행렬: cv
            # data object에 X_validation 예측 값을 넣어줌
            # CROSS-VALIDATION , EVALUATE CV
            cv[validation_idx,:] = svm_model.predict_proba(x_validation)


    # MODEL LOAD & TEST PREDICT
    # 12 MODELS 평균 사용
    models = os.listdir('svm_models/%s/'%(str(lucky_seed)))
    models_list = [x for x in models if x.endswith(".pkl")]

    # 모델결과가 잘 나왔는지 check
    # assert: 좌항과 우항의 값이 같으면 정상 작동, 다르면 오류 발생
    assert len(models_list) ==12


    temp_predictions = np.zeros((X_test.shape[0],2))

    # 12개 모델을 반복시켜서 결과산출 -> 12로 나눠서 평균값 계산
    for model in models_list:
        model = joblib.load('svm_models/%s/'%(str(lucky_seed))+model)
        predict_proba = svm_model.predict_proba(X_test)
        temp_predictions += predict_proba/12


    submission = pd.DataFrame(data=np.zeros((X_test.shape[0],2)))
    submission.index = Y_test.index
    submission.index.name = 'id'
    submission+=temp_predictions

    submission = submission.sort_index()
    submission = submission.groupby('id').mean()

    submission['pred'] = 0
    for idx in submission.index:
        if (submission[0][idx] < submission[1][idx]):
```

```
            submission['pred'][idx]= 1

    submission.to_csv('svm_models/submissions/%s_submission.csv'%(str(lucky_seed)), index=True)

    result = classification_report(Y_test, submission['pred'], target_names=['normal','smishing'])
    print(result)
    result_list.append(result)
    print('')
    result2 = accuracy_score(Y_test, submission['pred'])
    print(result2)
    result2_list.append(result2)
```