

잡케어 추천 알고리즘 경진대회

CONTENT



- a. 목표
- b. 배경
- c. 데이터 가공
- d.모델링
- e. 결과

목표

개별

데이터 전처리 데이터 분석 과정 학습

여러가지 모델링을 통한 머신러닝 학습

단체

DACON 리더보드 100등 안에 들기

배경



잡케어 서비스에 적용 가능한 추천 알고리즘 개발

- 1. 구직자에게 이력서를 AI 기술로 직무역량을 분석하는 시스템
- 2. 구인구직 빅데이터 기반으로 커리어 관리 서비스인 잡케어 서비스를 구축
 - 3. 데이터를 기반으로 개인별 맞춤형 컨텐츠 추천 모델 제작

데이터 가공

11

데이터 추가

Train, Test 데이터셋 외에 속성 코드 D, H, L 추가

```
train_data = pd.read_csv(f'{DATA_PATH}train.csv')
test_data = pd.read_csv(f'{DATA_PATH}test.csv')

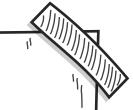
code_d = pd.read_csv(f'{DATA_PATH} 속성_D_코드.csv')
code_h = pd.read_csv(f'{DATA_PATH} 속성_H_코드.csv')
code_l = pd.read_csv(f'{DATA_PATH} 속성_L_코드.csv')

train_data.shape , test_data.shape

((501951, 35), (46404, 34))
```

```
#데이터 추가 (D, H, L)
def add_code(df, d_code, h_code, l_code):
    df = df.copv()
   # D Code
    df['person_prefer_d'] = df['person_prefer_d_1'].apply(lambda x: d_code[x]['속성 D 세분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_1'].apply(lambda x: d_code[x]['속성 D 소분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_1'].apply(lambda x: d_code[x]['속성 D 중분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_1'].apply(lambda x: d_code[x]['속성 D 대분류코드'])
   df['person_prefer_d'] = df['person_prefer_d_2'].apply(lambda x: d_code[x]['속성 D 세분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_2'].apply(lambda x: d_code[x]['속성 D 소분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_2'].apply(lambda x: d_code[x]['속성 D 중분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_2'].apply(lambda x: d_code[x]['속성 D 대분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_3'].apply(lambda x: d_code[x]['속성 D 세분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_3'].apply(lambda x: d_code[x]['속성 D 소분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_3'].apply(lambda x: d_code[x]['속성 D 중분류코드'])
    df['person_prefer_d'] = df['person_prefer_d_3'],apply(lambda x: d_code[x]['속성 D 대분류코드'])
    df['person_prefer_d'] = df['contents_attribute_d'].apply(lambda x: d_code[x]['속성 D 세분류코드'])
    df['person_prefer_d'] = df['contents_attribute_d'].apply(lambda x: d_code[x]['속성 D 소분류코드'])
    df['person_prefer_d'] = df['contents_attribute_d'].apply(lambda x: d_code[x]['속성 D 중분류코드'])
    df['person_prefer_d'] = df['contents_attribute_d'].apply(lambda x: d_code[x]['속성 D 대분류코드'])
   # H Code
    df['person_prefer_h'] = df['person_prefer_h_1'].apply(lambda x: h_code[x]['속성 H 대분류코드'])
    df['person_prefer_h'] = df['person_prefer_h_1'].apply(lambda x: h_code[x]['속성 H 중분류코드'])
    df['person_prefer_h'] = df['person_prefer_h_2'].apply(lambda x: h_code[x]['속성 H 대분류코드'])
    df['person_prefer_h'] = df['person_prefer_h_2'].apply(lambda x: h_code[x]['속성 H 중분류코드'])
   df['person_prefer_h'] = df['person_prefer_h_3'].apply(lambda x: h_code[x]['속성 H 대분류코드'])
    df['person prefer h'] = df['person prefer h 3'].apply(lambda x: h code[x]['속성 H 중분류코드'])
    df['person_prefer_h'] = df['contents_attribute_h'].apply(lambda x: h_code[x]['속성 H 대분류코드'])
    df['person_prefer_h'] = df['contents_attribute_h'].apply(lambda x: h_code[x]['속성 H 중분류코드'])
    df['contents_attribute_l'] = df['contents_attribute_l'].apply(lambda x: l_code[x]['속성 L 세분류코드'])
    df['contents_attribute_l'] = df['contents_attribute_l'].apply(lambda x: l_code[x]['속성 L 소분류코드'])
    df['contents_attribute_I'] = df['contents_attribute_I'].apply(lambda x: I_code[x]['속성 L 중분류코드'])
    df['contents_attribute_l'] = df['contents_attribute_l'].apply(lambda x: l_code[x]['속성 L 대분류코드'])
    return df
```

데이터 가공



데이터 제거

#학습에 필요없는 컬럼 리스트 cols_drop = ["id","person_prefer_f","person_prefer_g","contents_open_dt", "contents_rn","person_rn"]

0	x_train								
₽		d_l_match_yn	d_m_match_yn	d_s_match_yn	h_I_match_yn	h_m_match_yn	h_s_match_yn	person_attribute_a	person_attribute_a_1 p
	0	1	1	1	0	0	0	1	4
	1	0	0	0	1	1	0	1	3
	2	0	0	0	1	0	0	2	0
	3	0	0	0	1	0	0	2	0
	4	1	1	1	0	0	0	1	3
	501946	0	0	0	1	0	0	1	1
	501947	1	1	0	1	0	0	1	6
	501948	1	1	1	1	0	0	1	7
	501949	1	0	0	1	0	0	1	1
	501950	1	1	1	1	0	0	1	6

501951 rows × 73 columns

모델링

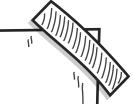
```
#의사결정트리
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
tree_preds1 = tree.predict(test)
submission = pd.read_csv('/content/drive/MyDrive/데이콘/직업 추천 2022/데이터/sample_submission.csv')
submission['target'] = tree_preds1
submission.to_csv('baseline_tree1.csv', index=False)
# 트리의 깊이 제한 max depth=n
tree = DecisionTreeClassifier(max random forest
tree.fit(X train, y train)
tree_preds2 = tree.predict(test)
                                [17] model = RandomForestClassifier(n_estimators = 300, max_depth=60, n_jobs = -1)
submission = pd.read_csv('/conten
                                      x = df_train.iloc[:,:-1]
submission['target'] = tree_preds
                                      y = df_train.iloc[:,-1]
```

model.fit(x,y)

submission.to_csv('baseline_tree2

RandomForestClassifier(max_depth=60, n_estimators=300, n_jobs=-1)





XGBoost

```
#634042

xgb_wrapper = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)

xgb_wrapper.fit(x_train, y_train)

w_preds = xgb_wrapper.predict(x_test)
```

catboost

모델링

```
#LGBM
from lightgbm import LGBMClassifier
lgbm = LGBMClassifier()
lgbm.fit(X_train,y_train)
preds - tree predict(test)
submis
            from sklearn.neighbors import KNeighborsClassifier
submis
            classifier = KNeighborsClassifier(n_neighbors=100)
submis
            classifier.fit(X_train, y_train)
LGBMCI
            preds = classifier.predict(test)
            submission = pd.read_csv('<u>/content/drive/MyDrive</u>/데이콘/직업 추천 2022/데이터/sample_submission.csv')
            submission['target'] = preds
            submission.to_csv('baseline_knn.csv', index=False)
           KNeighborsClassifier(n_neighbors=100)
```





▼ 학습 시작

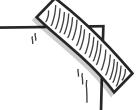
```
[ ] scores = []
    models = []
    models = []
    for tri, vai in cv.split(x_train):
        print("="*50)
        preds = []
        model = CatBoostClassifier(iterations=iterations,random_state=SEED,task_type="CPU",eval_metric="F1",cat_features=cat_features,one_hot_max_size=4)
        model.fit(x train.iloc[tri]. v train[tri].
                eval_set=[(x_train.iloc[vai], y_train[vai])],
                early_stopping_rounds=patience ,
                verbose = 100
                                                                                                                                       catboost
        models.append(model)
        scores.append(model.get_best_score()["validation"]["F1"])
        if is_holdout:
            break
```

▼ CV 결과 확인

```
[] print(scores)
print(np.mean(scores))

[0.6874037980301614, 0.6848302889157011, 0.6839807158620558, 0.6784028212274615, 0.6794157162050138]
0.6828066680480788
```

결론



잡케어 추천 알고리즘 경진대회에서 데이터 전처리 및 모델링의 과정을 통하여 성능을 향상하였으며 catboost, CV 5Fold, threshold 변경을 최종적으로 적용한 결과를 통해

- 1. 30일 동안 47번 제출하였으며
- 2. public점수 기준으로 하여 0.700483559을 얻었으며
 - 3. 리더보드에 최고 81위를 기록함.





감사합니다