Dacon 잡케어 추천 알고리즘 경진대회

1. 배경

잡케어는 일자리를 탐색하는 구직자에게 구직자의 이력서를 인공지능 기술로 직무역량을 자동 분석하여 훈련, 자격, 일자리 상담에 활용할 수 있도록 지원하는 시스템이다.

한국고용정보원은 구인구직 빅데이터 기반으로 커리어 관리 서비스인 잡케어 서비스를 구축하고 있다.

본 대회를 통해 데이터를 기반으로 개인별 맞춤형 컨텐츠 추천 모델을 만들고자 한다.

2. 목적

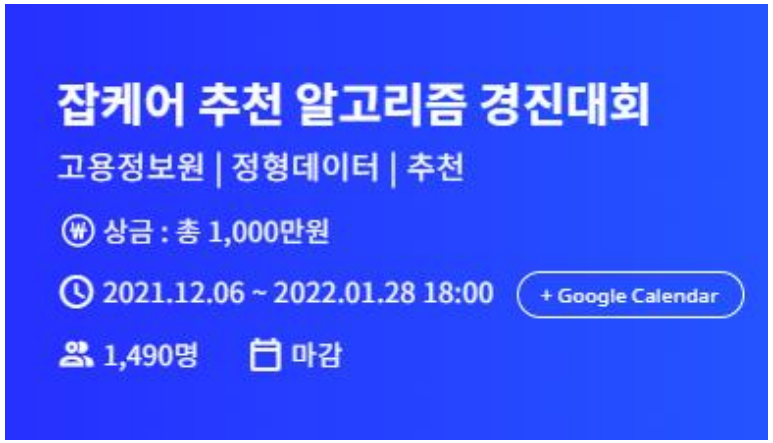
잡케어 서비스에 적용 가능한 추천 알고리즘 개발

3. 주최 / 주관

- 주최 : 한국고용정보원
- 주관 : 데이콘

4. 참가 대상

일반인, 학생 등 누구나



주최 : 한국고용정보원, 주관 : Dacon

Contents

01 경진대회 소개

1. 배경

잡케어는 일자리를 탐색하는 구직자에게 구직자의 이력서를 인공지능 기술로 직무역량을 자동 분석하여 훈련, 자격, 일자리 상담에 활용할 수 있도록 지원하는 시스템이다.

한국고용정보원은 구인구직 빅데이터 기반으로 커리어 관리 서비스인 잡케어 서비스를 구축하고 있다.

본 대회를 통해 데이터를 기반으로 개인별 맞춤형 컨텐츠 추천 모델을 만들고자 한다.

2. 목적

잡케어 서비스에 적용 가능한 추천 알고리즘 개발

3. 주최 / 주관

- 주최 : 한국고용정보원
- 주관 : 데이콘

4. 참가 대상

일반인, 학생 등 누구나

01 경진대회 소개

1. 규칙

- 제출 횟수 및 최대 팀원
  - a. 1일 최대 제출 횟수 : 3회
  - b. 팀 최대 인원 : 5명
- 리더보드
  - a. 평가 산식 : F1-score
  - b. Public Score : 전체 테스트 데이터 중 33%
  - c. Private Score : 전체 테스트 데이터 중 67%
- Test Dataset은 추론과정에서만 사용 가능
- 코드 평가(1차 평가)
  - a. 다음 조건을 만족하며 제출한 코드로 Private score 복원된 상위 4팀에게 상금 수여
  - b. 대회 종료 후 평가 희망자는 코드 공유 게시판에 코드 게시 및 코드와 설명자료를 dacon@dacon.io로 제출
    - i. 코드에 './data' 데이터 입/출력 경로 포함
    - ii. 코드 파일 확장자 : .R, .rmd, .py, .ipynb
  - c. 코드 제출 유의사항
    - i. 코드 인코딩 : UTF-8
    - ii. 개발환경(OS) 및 라이브러리 버전 기재
    - iii. 전체 실행 프로세스 및 코드 실행방법을 readme파일로 정리해서 제출
    - iv. 실행 방법대로 실행 시 모든 코드가 오류 없이 실행되어야 합니다.
- 설명자료 제출 유의사항
  - a. 데이터 전처리/모델링에 대한 설명
  - b. (사용했을 시) 추가 데이터셋 혹은 Pre-trained 모델 명시
- 코드공유 게시판 게시
  - a. 제목 양식 : 팀 이름, Private 순위와 점수, 모델 이름. 예) 데이콘팀, Private 1위/0.82, ResNet
  - b. 내용 : 전처리, 학습, 후처리, 추론 일련의 과정을 담은 코드를 게시
  - c. ipynb형식의 게시가 어려울 경우 github링크와 함께 코드 주요 부분과 마크다운을 작성하여 게시
- 코드 검증 환경
  - a. NVIDIA GeForce RTX 3090 / Ubuntu 18.04.6 LTS (64bit)
  - b. Tesla V100-PCIE-32GB / Ubuntu 16.04.6 LTS (64bit)
  - c. Colab GPU / Linux-5.4.104+-x86_64-with-Ubuntu-18.04-bionic
- 발표 평가(2차 평가)
  - a. 1차평가를 통해 상위 6팀 대상으로 진행
  - b. 2차 평가 대상자는 발표 영상을 10분 내외로 녹화하여 제출
  - c. 2차 평가 항목

✓ **Task : Classification (Binary)**

한국고용정보원 잡케어 서비스 사용자의 데이터를 바탕으로 특정 구직관련 컨텐츠를 열람했는지 예측하는 것이 목적

✓ **Metric : F1 Score**

Recall과 Prescision의 조화평균

2 * Recall * Precision / (Recall + Prescion)

02 아키텍쳐 개요 – Preprocessing & Feature Engineering

✓ **Date 컬럼**

요일, 주말, office hour 등 feature 추출
sin, cos 주기성 feature 추출 (year-normalized)

✓ **Categorical 컬럼**

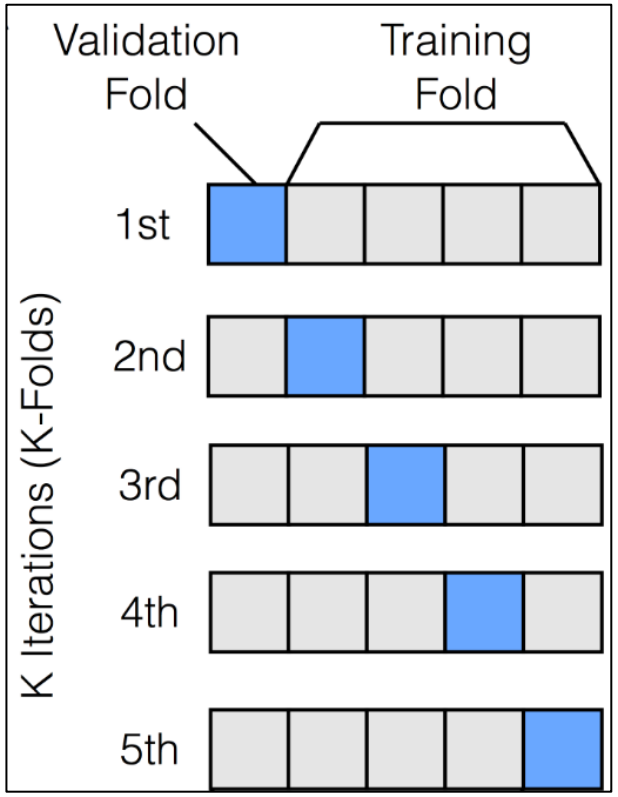
3개 이상의 클래스가 있을 경우 one-hot encoding

```
# Feature Engineering
# 1. 일자 관련
# train set
full_x["quarter"] = full_x["contents_open_dt"].dt.quarter
full_x["month"] = full_x["contents_open_dt"].dt.month
full_x["season"] = full_x["contents_open_dt"].apply(getSeason)
full_x["week_of_month"] = full_x["contents_open_dt"].apply(week_of_month)
# monday 0, sunday 6
full_x["weekday"] = full_x["contents_open_dt"].dt.weekday
full_x["hour"] = full_x["contents_open_dt"].dt.hour
full_x["office_hour"] = [1 if 18 >= i >= 9 else 0 for i in full_x["contents_open_dt"].dt.hour]

# sin, cos 주기성 변환 파생변수 컬럼추가
time_zero = datetime(1970, 1, 1, 0, 0, 0)
day_to_sec = 24 * 60 * 60
year_to_sec = 365.2425 * day_to_sec
full_x["freq_sin_year"] = np.sin(((full_x["contents_open_dt"] - time_zero).dt.total_seconds() / year_to_sec) * 2 * np.pi)
full_x["freq_cos_year"] = np.cos(((full_x["contents_open_dt"] - time_zero).dt.total_seconds() / year_to_sec) * 2 * np.pi)
full_x["freq_sin_day"] = np.sin(((full_x["contents_open_dt"] - time_zero).dt.total_seconds() / day_to_sec) * 2 * np.pi)
full_x["freq_cos_day"] = np.cos(((full_x["contents_open_dt"] - time_zero).dt.total_seconds() / day_to_sec) * 2 * np.pi)
```

02 아키텍쳐 개요 – XGBoost

✓ 5-Folds CV Model Ensemble



✓ Hyper-parameter tuning with Optuna

```python
# ===== modeling =====
# <10-folds inference>

# fold spliter setting
kfolds_spliter = StratifiedKFold(10, random_state=42, shuffle=True)
categoIdx = findIdx(full_x.columns, cat_vars)
cut_off = 0.5

# result container setting
val_prob = np.zeros((full_x.shape[0], 2))
test_prob = np.zeros((test_x.shape[0], 2))
val_perf = []

# optuna function
def optuna_objective_function(trial: Trial, train_x, train_y, val_x, val_y, model_name, ntrees=None, eta=None):
    tuning_params = {
        "max_depth": trial.suggest_int("max_depth", 4, 8, step=1),
        "subsample": trial.suggest_float("subsample", 0.5, 0.95, step=0.05),
        "colsample_bynode": trial.suggest_float("colsample_bynode", 0.5, 0.95, step=0.05),
        "reg_lambda": trial.suggest_float("reg_lambda", 0.01, 10.0, step=0.01),
        "scale_pos_weight": trial.suggest_float("scale_pos_weight", 1.0, 3.0, step=0.01)
    }
    model = xgb.XGBClassifier(booster="gbtree", objective="binary:logistic",
                              tree_method="gpu_hist", gpu_id=0, sampling_method="gradient_based",
                              n_estimators=ntrees,
                              learning_rate=eta,
                              random_state=fold, verbosity=0, use_label_encoder=False,
                              **tuning_params)

    model.fit(train_x, train_y, eval_metric=xgb_f1_score, verbose=False,
              eval_set=[(val_x, val_y)], early_stopping_rounds=int(ntrees * 0.2))

    prob = model.predict_proba(val_x)
    pred = [1 if i > cut_off else 0 for i in prob[:, 1]]
    opt_score = metrics.f1_score(val_y, pred)

    trial.report(opt_score, step=trial.number)
    if trial.should_prune():
        raise optuna.exceptions.TrialPruned()

    return opt_score
```

03 결과 정리

- ✓ 다양한 모델을 시도하였고 그 중 최고 스코어인 XGBoost를 최종 모델로 선정
  타 모델과 ensemble을 시도 하지 못한 것이 아쉬운 점
- ✓ 단일 모델로 상위 32% 수준에 랭크하며 대회를 마무리

628256	submission_LightGBM_GOSS_Try5.csv edit	2022-01-01 18:11:20	0.6910780298 -	☐
625629	submission_XGBoost_Gbtree_Try3.csv edit	2021-12-23 20:16:28	0.6925325801 -	☑
625627	submission_LightGBM_GOSS_Try4.csv edit	2021-12-23 20:15:34	0.6911621569 -	☐
623895	submission_XGBoost_Gbtree_Try2.csv edit	2021-12-21 20:11:58	0.6919947056 -	☐
622830	submission_CatBoost_GBM_Try1.csv edit	2021-12-19 19:47:36	0.6877318048 -	☐
622292	submission_LightGBM_GOSS_Try3.csv edit	2021-12-18 13:44:07	0.6884925483 -	☐

감사합니다

04 Appendix - 구현코드

```python
# optuna function
def optuna_objective_function(trial: Trial, train_x, train_y, val_x, val_y, model_name, ntrees=None, eta=None):
    tuning_params = {
        "max_depth": trial.suggest_int("max_depth", 4, 8, step=1),
        "subsample": trial.suggest_float("subsample", 0.5, 0.95, step=0.05),
        "colsample_bynode": trial.suggest_float("colsample_bynode", 0.5, 0.95, step=0.05),
        "reg_lambda": trial.suggest_float("reg_lambda", 0.01, 10.0, step=0.01),
        "scale_pos_weight": trial.suggest_float("scale_pos_weight", 1.0, 3.0, step=0.01)
    }
    model = xgb.XGBClassifier(booster="gbtree", objective="binary:logistic",
                              tree_method="gpu_hist", gpu_id=0, sampling_method="gradient_based",
                              n_estimators=ntrees,
                              learning_rate=eta,
                              random_state=fold, verbosity=0, use_label_encoder=False,
                              **tuning_params)

    model.fit(train_x, train_y, eval_metric=xgb_f1_score, verbose=False,
              eval_set=[(val_x, val_y)], early_stopping_rounds=int(ntrees * 0.2))

    prob = model.predict_proba(val_x)
    pred = [1 if i > cut_off else 0 for i in prob[:, 1]]
    opt_score = metrics.f1_score(val_y, pred)

    trial.report(opt_score, step=trial.number)
    if trial.should_prune():
        raise optuna.exceptions.TrialPruned()

    return opt_score
```