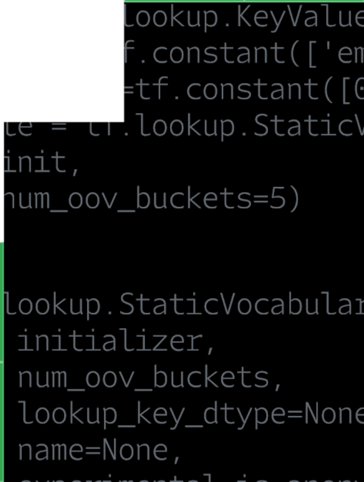
Google Developer Student Clubs

2주차 1조

팀원: (강용진), (조선빈), (조현진)



```
lookup.KeyValue
f.constant(['em
=tf.constant([0
te = tf.lookup.StaticV
init,
num_oov_buckets=5)


lookup.StaticVocabular
 initializer,
 num_oov_buckets,
 lookup_key_dtype=None
 name=None,
```

데이터 전처리, 모델 설계, 모델 훈련

데이터 전처리

결측치 처리

train.csv

: 결측치가 있는 행은 제거 (총 8693개 -> 6764개)

test.csv

: 평균값, 최빈값 등을 사용

데이터 전처리

Feature Engineering
: 새롭게 만들어낸 feature들

PassengerId

: 승객별 고유 ID.   0168_01, 0175_01, 0184_01

NoOfPassenger

: 같은 PassengerId를 가진 사람을 구분하는 번호.   0175_01, 0175_02, 0175_03

ExpenseInShip

: 배에서 소비한 지출. ‘RoomService’, 'FoodCourt’, 'ShoppingMall + 'Spa' + 'VRDeck’의 합으로 이루어짐

데이터 전처리

One-Hot Encoding

: 범주형(categorical) 데이터를 0과 1로

HomePlanet
Earth
Mars
Europa

→

hp1	hp2	hp3
0	1	0
0	0	1
1	0	0

데이터 전처리

MEstimateEncoder
: Simplified version of TargetEncoder

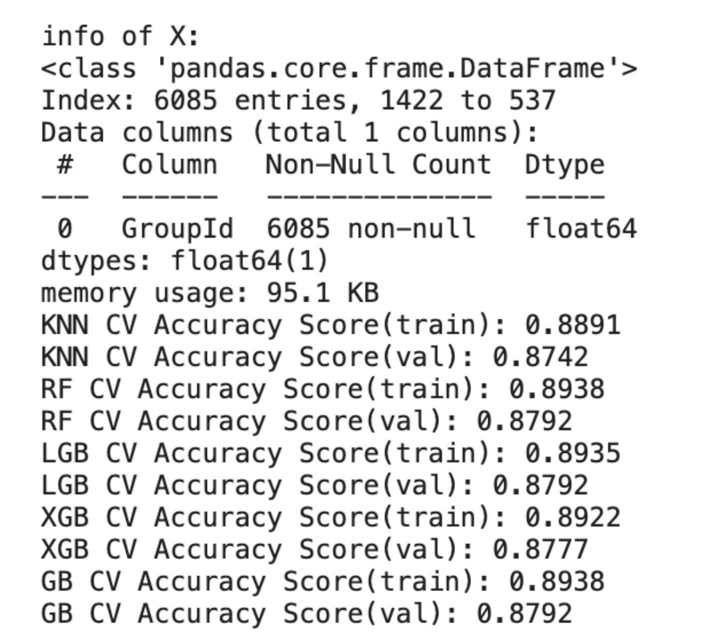
TargetEncoder
범주형(categorical) 데이터를 *결과값을 고려해서* 숫자로

HomePlanet	Transported
Earth	1
Earth	0
Earth	0
Mars	1
Mars	0

HomePlanet	Transported
0.33	1
0.33	0
0.33	0
0.5	1
0.5	0

데이터 전처리

Overfitting에 매우 취약



```
info of X:
<class 'pandas.core.frame.DataFrame'>
Index: 6085 entries, 1422 to 537
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   GroupId  6085 non-null   float64
dtypes: float64(1)
memory usage: 95.1 KB
KNN CV Accuracy Score(train): 0.8891
KNN CV Accuracy Score(val): 0.8742
RF CV Accuracy Score(train): 0.8938
RF CV Accuracy Score(val): 0.8792
LGB CV Accuracy Score(train): 0.8935
LGB CV Accuracy Score(val): 0.8792
XGB CV Accuracy Score(train): 0.8922
XGB CV Accuracy Score(val): 0.8777
GB CV Accuracy Score(train): 0.8938
GB CV Accuracy Score(val): 0.8792
```

모델 설계

사용한 모델

- KNN
- SVC
- Random Forest
- GradientBoosting
- LightGBM
- Xgboost
- VotingClassifier
- Neural Network

성능 평가

성능 평가

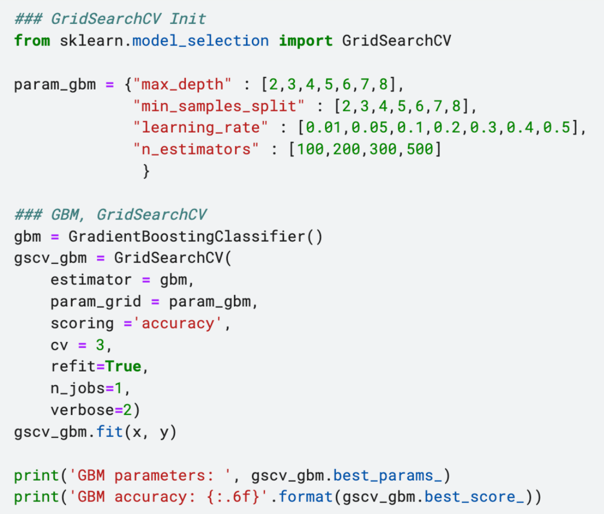
No.	preprocessing	model	Score
1	결측치 제거, 범주형 데이터 인코딩	GradientBoosting	0.80313
2	Expense 추가		0.80102
3	PassengerId를 분리해서 각각 추가		0.80406

최적화

최적화

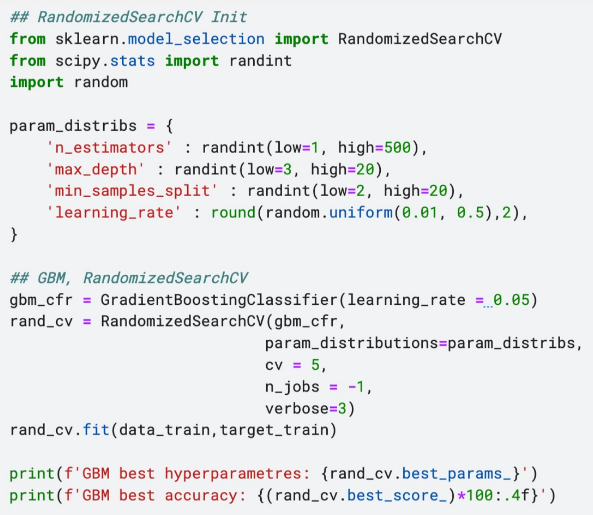
최적의 Hyperparameter 조합 찾기

GridSearchCV



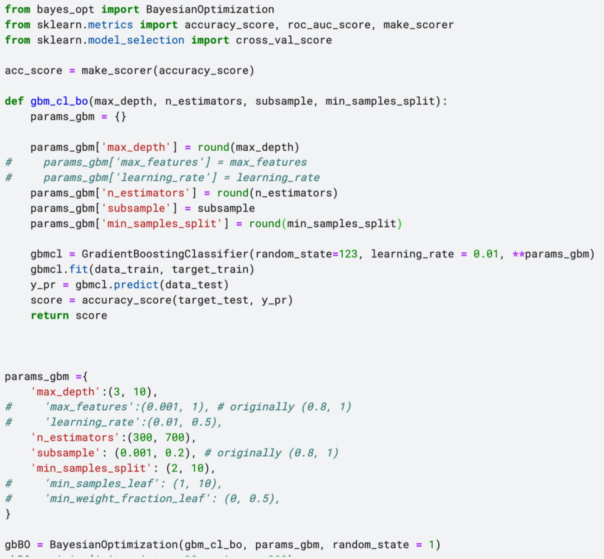
```
### GridSearchCV Init
from sklearn.model_selection import GridSearchCV

param_gbm = {"max_depth" : [2,3,4,5,6,7,8],
             "min_samples_split" : [2,3,4,5,6,7,8],
             "learning_rate" : [0.01,0.05,0.1,0.2,0.3,0.4,0.5],
             "n_estimators" : [100,200,300,500]
              }

### GBM, GridSearchCV
gbm = GradientBoostingClassifier()
gscv_gbm = GridSearchCV(
    estimator = gbm,
    param_grid = param_gbm,
    scoring ='accuracy',
    cv = 3,
    refit=True,
    n_jobs=1,
    verbose=2)
gscv_gbm.fit(x, y)

print('GBM parameters: ', gscv_gbm.best_params_)
print('GBM accuracy: {:.6f}'.format(gscv_gbm.best_score_))
```

RandomizedSearchCV



```
## RandomizedSearchCV Init
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
import random

param_distribs = {
    'n_estimators' : randint(low=1, high=500),
    'max_depth' : randint(low=3, high=20),
    'min_samples_split' : randint(low=2, high=20),
    'learning_rate' : round(random.uniform(0.01, 0.5),2),
}

## GBM, RandomizedSearchCV
gbm_cfr = GradientBoostingClassifier(learning_rate = 0.05)
rand_cv = RandomizedSearchCV(gbm_cfr,
                             param_distributions=param_distribs,
                             cv = 5,
                             n_jobs = -1,
                             verbose=3)
rand_cv.fit(data_train,target_train)

print(f'GBM best hyperparametres: {rand_cv.best_params_}')
print(f'GBM best accuracy: {(rand_cv.best_score_)*100:.4f}')
```

Bayesian Optimization



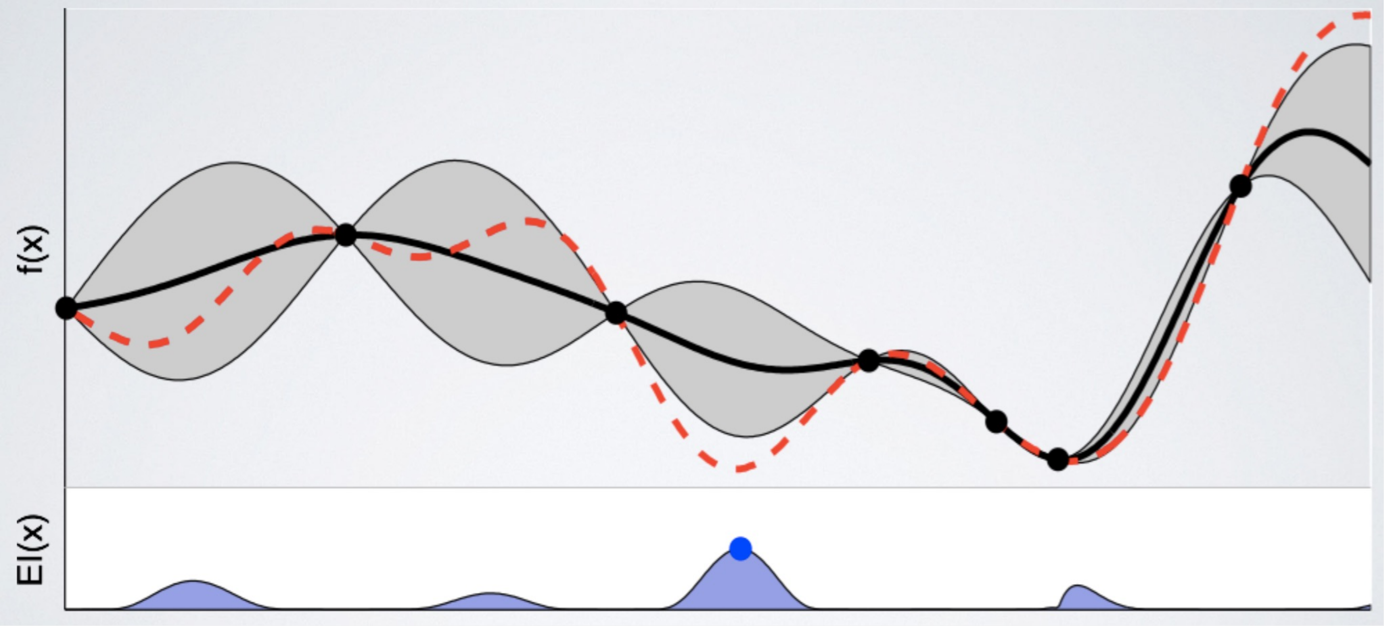
```
from bayes_opt import BayesianOptimization
from sklearn.metrics import accuracy_score, roc_auc_score, make_scorer
from sklearn.model_selection import cross_val_score

acc_score = make_scorer(accuracy_score)

def gbm_cl_bo(max_depth, n_estimators, subsample, min_samples_split):
    params_gbm = {}

    params_gbm['max_depth'] = round(max_depth)
#     params_gbm['max_features'] = max_features
#     params_gbm['learning_rate'] = learning_rate
    params_gbm['n_estimators'] = round(n_estimators)
    params_gbm['subsample'] = subsample
    params_gbm['min_samples_split'] = round(min_samples_split)

    gbmcl = GradientBoostingClassifier(random_state=123, learning_rate = 0.01, **params_gbm)
    gbmcl.fit(data_train, target_train)
    y_pr = gbmcl.predict(data_test)
    score = accuracy_score(target_test, y_pr)
    return score



params_gbm ={
    'max_depth':(3, 10),
#     'max_features':(0.001, 1), # originally (0.8, 1)
#     'learning_rate':(0.01, 0.5),
    'n_estimators':(300, 700),
    'subsample': (0.001, 0.2), # originally (0.8, 1)
    'min_samples_split': (2, 10),
#     'min_samples_leaf': (1, 10),
#     'min_weight_fraction_leaf': (0, 0.5),
}

gbBO = BayesianOptimization(gbm_cl_bo, params_gbm, random_state = 1)
```

최적화

Bayesian Optimization
: 미지의 함수(Black-box function)가 주어져도 최댓값/최솟값을 찾아간다

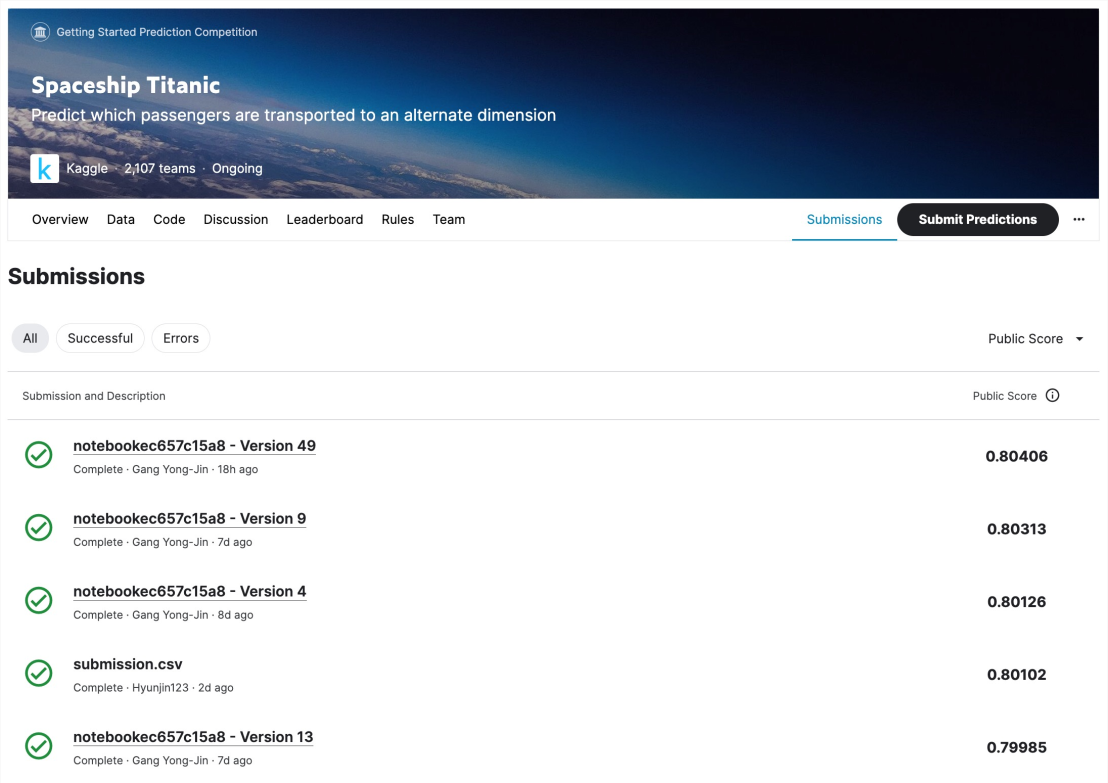
확률분포를 활용해 f(x)의 값을 추정하는 **Surrogate Model**

더 나은 값이 나올 x의 값을 조사하는 **Acquisition Function**



kaggle 제출

kaggle 제출



Spaceship Titanic

Predict which passengers are transported to an alternate dimension

Kaggle · 2,107 teams · Ongoing

Submissions

Submission and Description	Public Score
✓ notebookec657c15a8 - Version 49 Complete · Gang Yong-Jin · 18h ago	0.80406
✓ notebookec657c15a8 - Version 9 Complete · Gang Yong-Jin · 7d ago	0.80313
✓ notebookec657c15a8 - Version 4 Complete · Gang Yong-Jin · 8d ago	0.80126
✓ submission.csv Complete · Hyunjin123 · 2d ago	0.80102
✓ notebookec657c15a8 - Version 13 Complete · Gang Yong-Jin · 7d ago	0.79985