해외 축구선수 이적료 예측

SAI A -ML 팀
(이명준, 최태정, 김주연)

목차

1. 공부 과정

ML

학습 교재

모임은 매주 수요일(데이콘 분석), 토요일(공부한 것) 각각 오후 10시 !!

Board view

일정

시작 전 1	진행 중 3	완료 8
제목 없음	제목 없음	OT 2022년 2월 18일
+ 새로 만들기	[스포츠] 해외 축구 선수 이적료 예측 미션 1	[문화] 영화 관객수 예측 모델 개발 2022년 3월 2일
	머신러닝 공부(chapter5)	머신러닝 공부(chapter 1) 2022년 3월 5일
	+ 새로 만들기	물류 | 회귀 | 교육 | 1백만원 2022년 3월 9일
		머신러닝 공부(chapter 2)
		금융 | 월간 16 | 신용카드 연체
		머신러닝 공부(chapter 3)
		머신러닝 공부(chapter 4)
		+ 새로 만들기

2. EDA

EDA란 탐색적 자료 분석 (Exploratory data analysis) 의 약자로 다음의 순서를 따른다.

1. 데이터 가져오기
2. 데이터 모양 확인
3. 데이터 타입 확인
4. 데이터 기초 분석
5. 데이터 클린징
6. 데이터 시각화

EDA를 통해 우리는 데이터의 기본 구조를 확인하면서 중요한 변수가 무엇인지, 이상치와 결측값은 무엇인지 등을 알아낼 수 있고 이를통해 기본 가정과 최적의 요인 , 적절한 모델 등을 알아낼 수 있다.

데이터 설명

id : 선수 고유의 아이디

name : 이름

age : 나이

continent : 선수들의 국적이 포함되어 있는 대륙입니다

contract_until : 선수의 계약기간이 언제까지인지 나타내어 줍니다

position : 선수가 선호하는 포지션입니다. ex) 공격수, 수비수 등

prefer_foot : 선수가 선호하는 발입니다. ex) 오른발

reputation : 선수가 유명한 정도입니다. ex) 높은 수치일 수록 유명한 선수

stat_overall : 선수의 현재 능력치 입니다.

stat_potential : 선수가 경험 및 노력을 통해 발전할 수 있는 정도입니다.

stat_skill_moves : 선수의 개인기 능력치 입니다.

value : FIFA가 선정한 선수의 이적 시장 가격 (단위 : 유로) 입니다 (타겟)

간단한 데이터 확인

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8932 entries, 0 to 8931
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               8932 non-null   int64  
 1   name             8932 non-null   object 
 2   age              8932 non-null   int64  
 3   continent        8932 non-null   object 
 4   contract_until   8932 non-null   object 
 5   position         8932 non-null   object 
 6   prefer_foot      8932 non-null   object 
 7   reputation       8932 non-null   float64
 8   stat_overall     8932 non-null   int64  
 9   stat_potential   8932 non-null   int64  
 10  stat_skill_moves  8932 non-null   float64
 11  value            8932 non-null   float64
dtypes: float64(3), int64(4), object(5)
memory usage: 837.5+ KB
```

```
train.describe()
```

	id	age	reputation	stat_overall	stat_potential	stat_skill_moves	value
count	8932.000000	8932.000000	8932.000000	8932.000000	8932.000000	8932.000000	8.932000e+03
mean	7966.775750	25.209136	1.130878	67.091133	71.997201	2.401702	2.778673e+06
std	4844.428521	4.635515	0.423792	6.854910	5.988147	0.776048	5.840982e+06
min	0.000000	16.000000	1.000000	47.000000	48.000000	1.000000	1.000000e+04
25%	3751.750000	21.000000	1.000000	63.000000	68.000000	2.000000	3.750000e+05
50%	7696.500000	25.000000	1.000000	67.000000	72.000000	2.000000	8.250000e+05
75%	12082.250000	28.000000	1.000000	72.000000	76.000000	3.000000	2.600000e+06
max	16948.000000	40.000000	5.000000	94.000000	94.000000	5.000000	1.105000e+08

문자형 데이터 처리

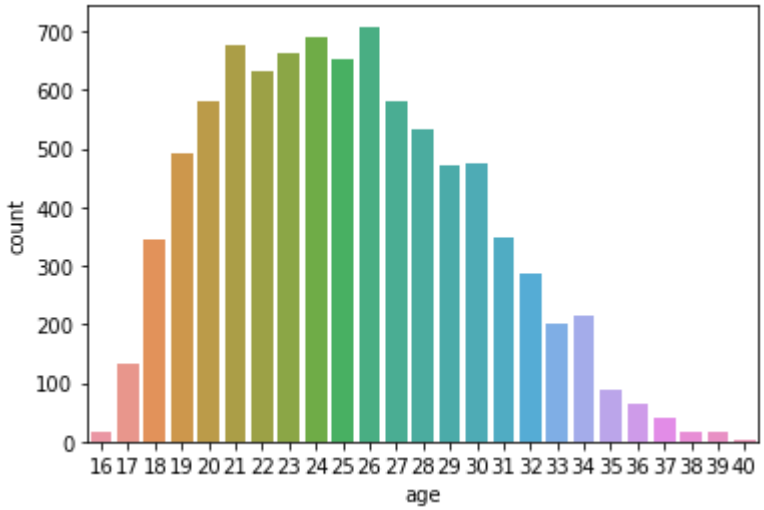
```
train = train.drop(['id','name'],axis=1)
test = test.drop(['id','name'],axis=1)
```

```
def to_numeric(x):
    if '2018' in x:
        x = 2018
    elif '2019' in x:
        x = 2019
    elif '2020' in x:
        x = 2020
    elif '2021' in x:
        x = 2021
    elif '2022' in x:
        x = 2022
    elif '2023' in x:
        x = 2023
    elif '2024' in x:
        x = 2024
    elif '2025' in x:
        x = 2025
    elif '2026' in x:
        x = 2026
    return x
```

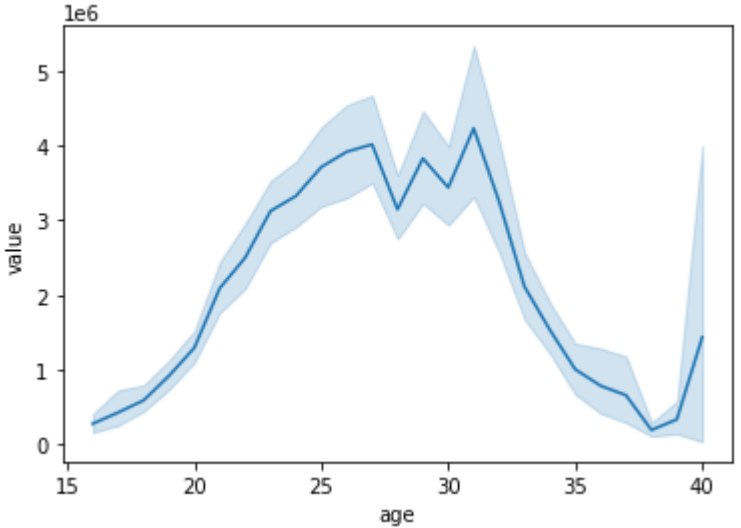
```
train['contract_until'] = train['contract_until'].apply(lambda x : to_numeric(x))
test['contract_until'] = test['contract_until'].apply(lambda x : to_numeric(x))
```

age(나이)

```
sns.countplot(train['age'])
```

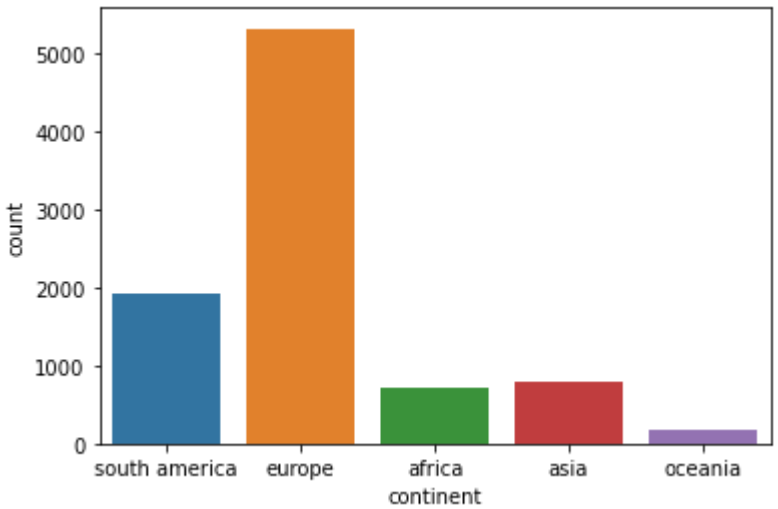



```
sns.lineplot(data=train, x='age',y='value')
```

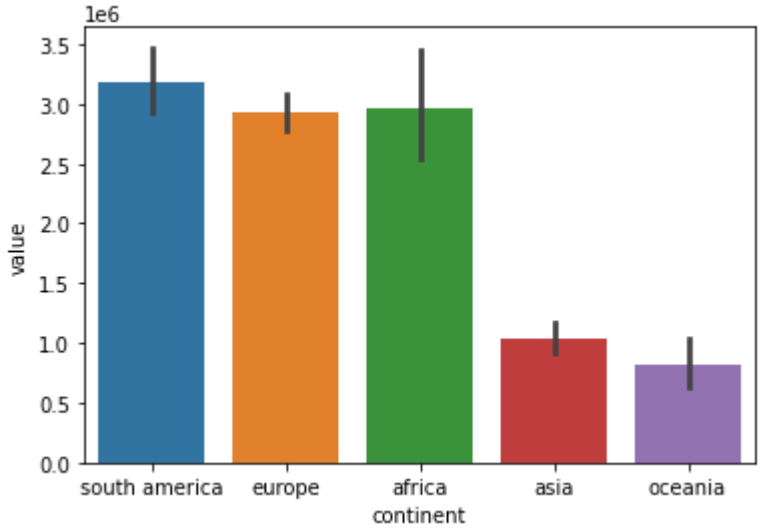


contitent (출신)

```
sns.countplot(train['continent'])
```

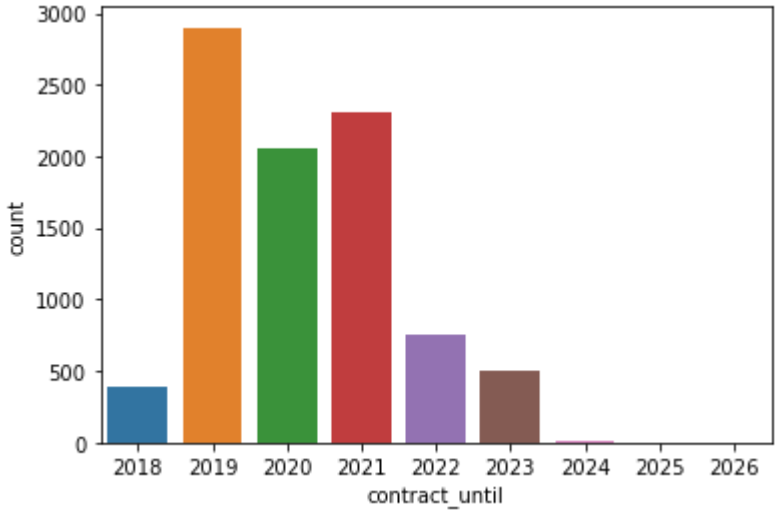



```
sns.barplot(data=train, x='continent',y='value')
```

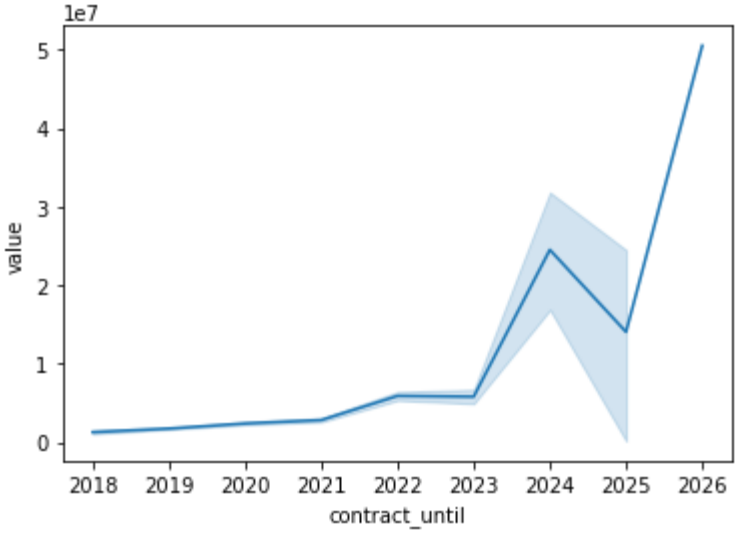


contract_until(계약기간)

```
sns.countplot(train['contract_until'])
```

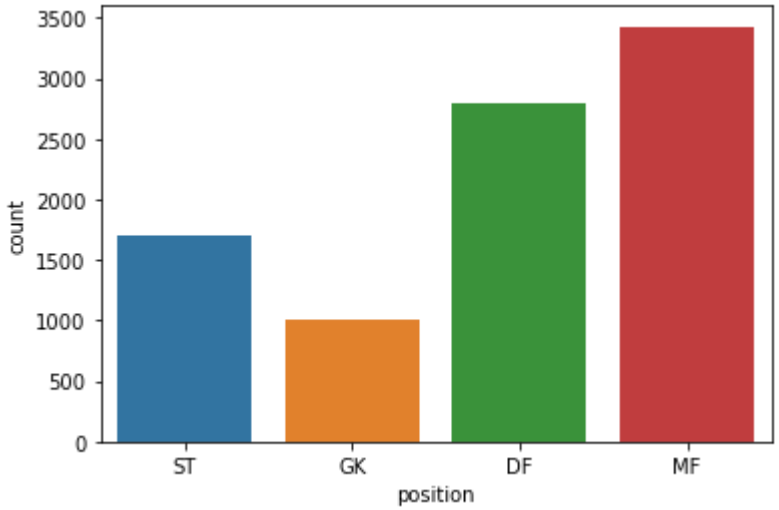



```
sns.lineplot(data=train, x='contract_until', y='value')
```

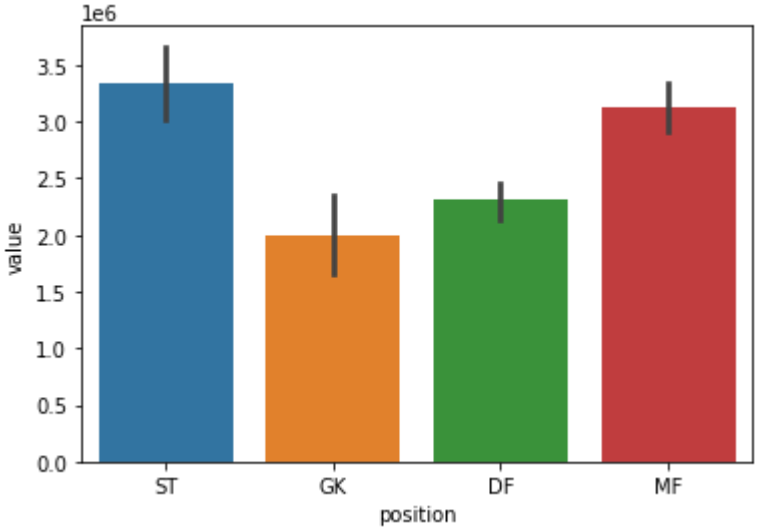


position

```
sns.countplot(train['position'])
```

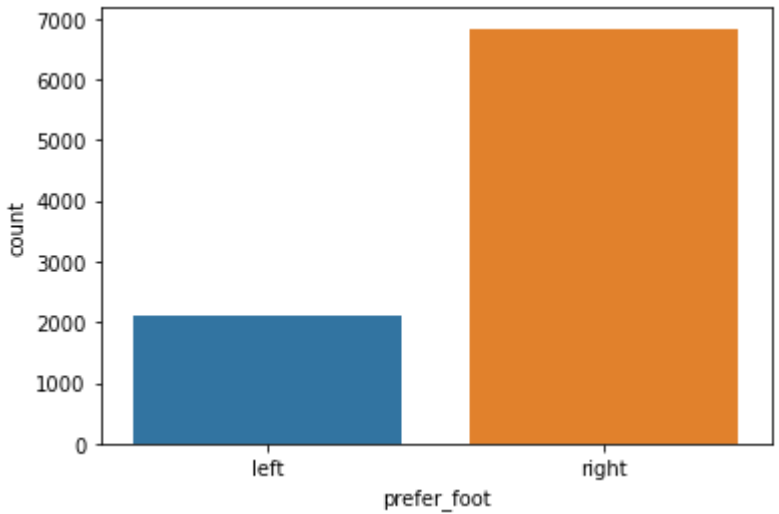



```
sns.barplot(data=train, x='position', y='value')
```

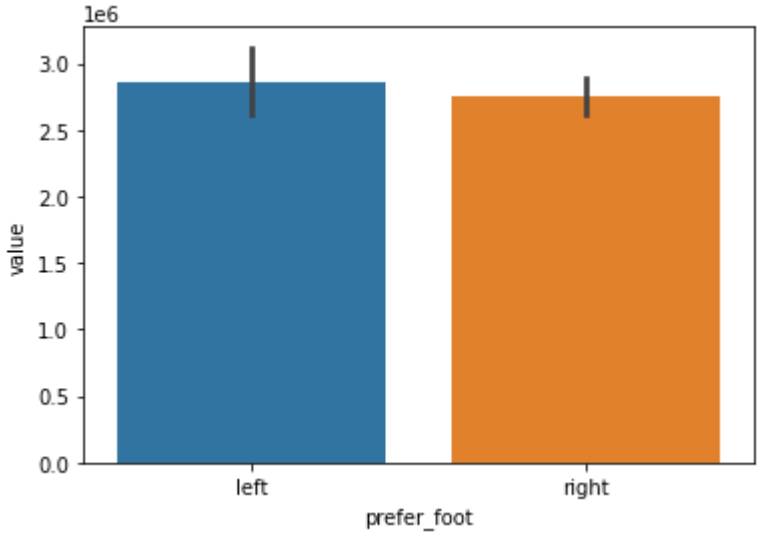


prefer_foot

```
sns.countplot(train['prefer_foot'])
```

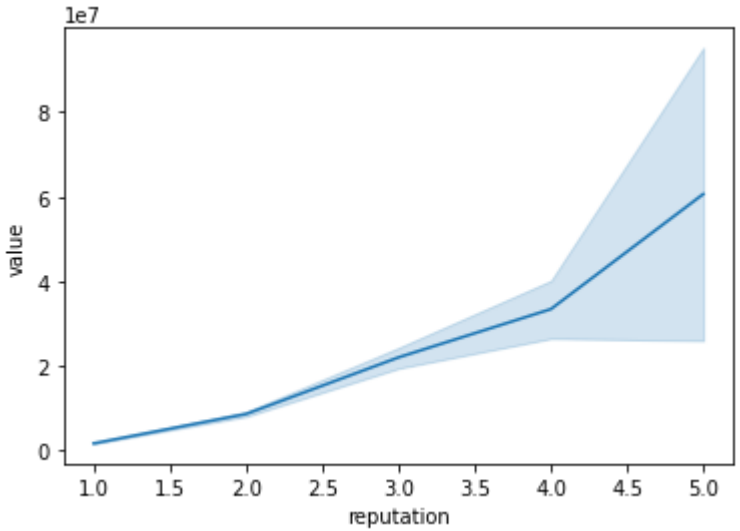



```
sns.barplot(data=train, x='prefer_foot', y='value')
```



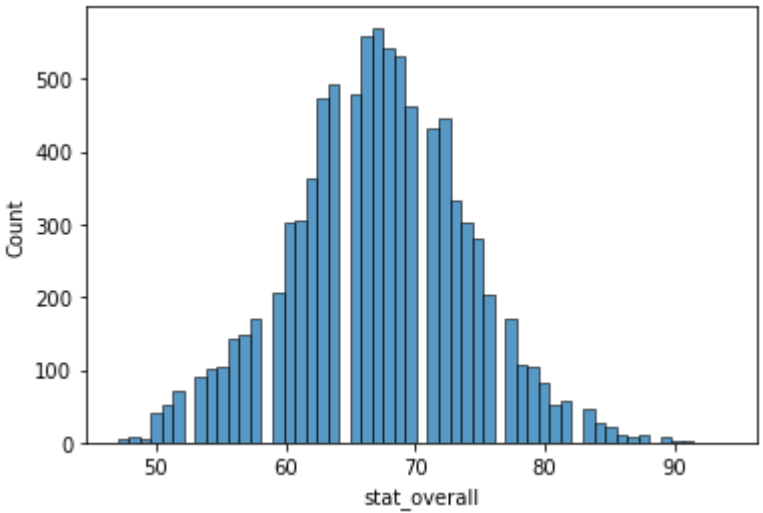
reputation(유명세)

```
sns.lineplot(data=train, x='reputation', y='value')
```

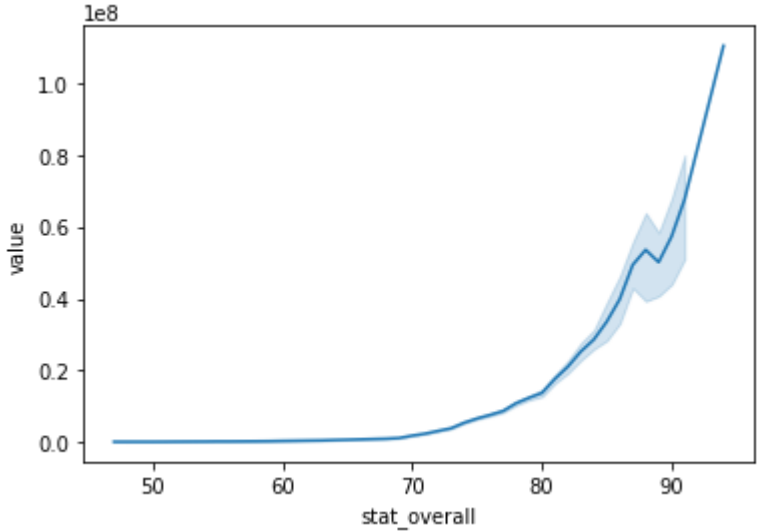


stat_overall(능력치)

```
sns.histplot(train['stat_overall'])
```

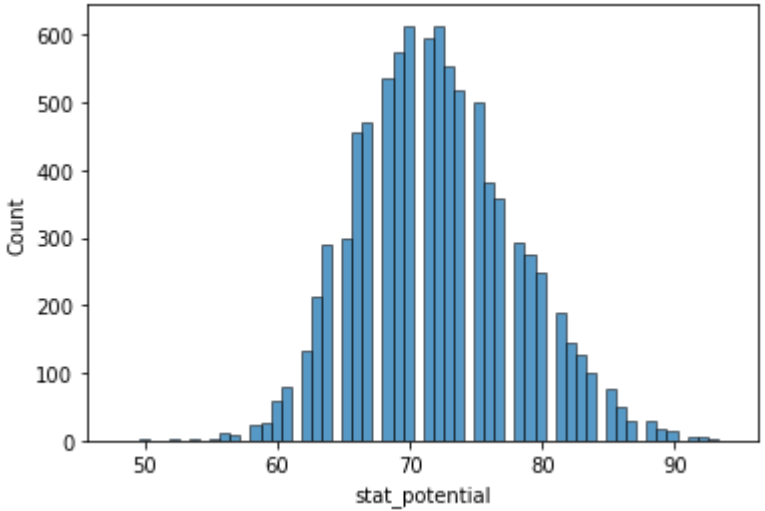



```
sns.lineplot(data=train,x='stat_overall',y='value')
```

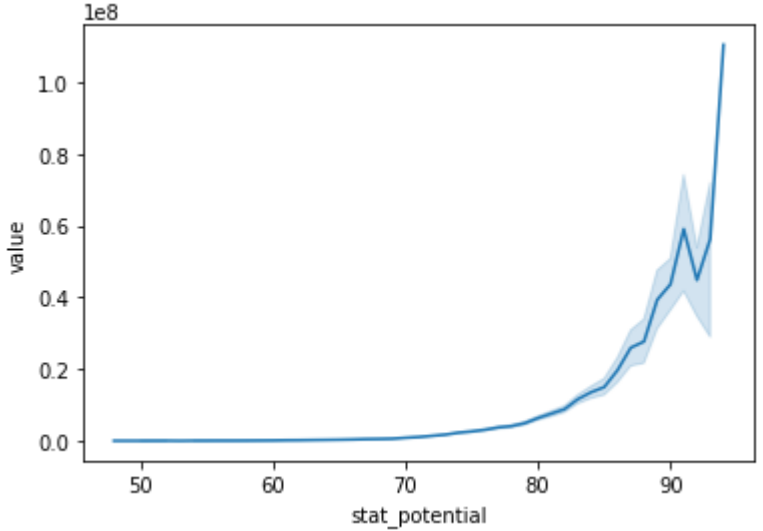


stat_potential(잠재력)

```
sns.histplot(train['stat_potential'])
```

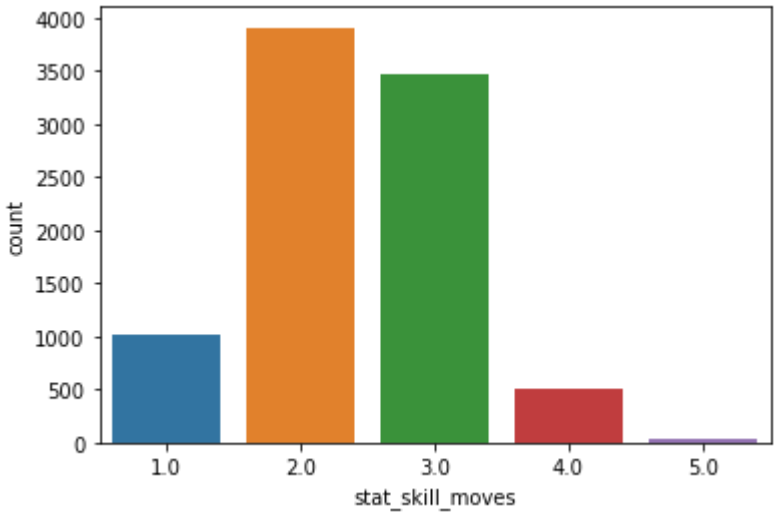



```
sns.lineplot(data=train,x='stat_potential',y='value')
```

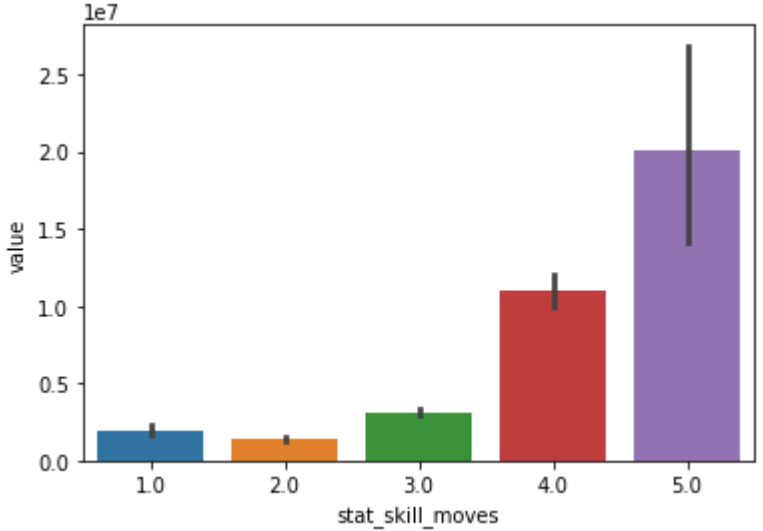


stat_skill_moves(개인기)

```
sns.countplot(train['stat_skill_moves'])
```




```
sns.barplot(data=train,x='stat_skill_moves',y='value')
```



3. 전처리

전처리란 raw data를 데이터 분석 목적과 방법에 맞는 형태로 처리하기 위하여 불필요한 정보를 분리 제거하고 가공하기 위한 예비적인 조작을 말합니다.

- **데이터 정제(cleansing)**: 결측 값들을 채워 넣고, 이상치를 식별 또는 제거하고, 잡음 섞인 데이터를 평활화하여 데이터의 불일치성을 교정

- **데이터 변환(transformation)**: 데이터 유형 변환 등 데이터 분석이 쉬운 형태로 변환하는 기술. 정규화, 집합화,요약, 계층 생성 등의 방법을 활용한다.

- **데이터 필터링(filtering)**: 오류 발견, 보정, 삭제 및 중복성 확인 등의 과정을 통해 데이터의 품질을 향상하는 기술

- **데이터 통합(integration)**: 데이터 분석이 용이하도록 유사 데이터 및 연계가 필요한 데이터들을 통합하는 기술

- **데이터 축소(reduction)**: 분석 시간을 단축할 수 있도록 데이터 분석에 활용되지 않는 항목 등을 제거하는 기술

인코딩

인코딩에는 두가지 방식이 있는데 1. 레이블 인코딩 2.원핫 인코딩 이다.

1. **레이블 인코딩**은 카테고리형 피처에 숫자를 부여하여 숫자형 카테고리로 만드는 것이다.
2. **원핫 인코딩**은 피처값의 유형에 따라 새로운 피처를 추가해 고유값에 해당하는 칼럼에만 1을 표시하고 나머지 칼럼에는 0을 표시하는 것이다.

원핫 인코딩

```
train = pd.get_dummies(train)
test= pd.get_dummies(test)
```

	age	contract_until	reputation	stat_overall	stat_potential	stat_skill_moves	value	continent_africa	continent_asia	continent_europe	continent_oceania	continent_south america
0	31	2021	5.0	94	94	4.0	110500000.0	0	0	0	0	1
1	27	2020	4.0	91	93	1.0	72000000.0	0	0	1	0	0
2	31	2021	5.0	91	91	3.0	80000000.0	0	0	0	0	1
3	32	2020	4.0	91	91	3.0	51000000.0	0	0	1	0	0
4	25	2021	3.0	90	93	1.0	68000000.0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
927	18	2019	1.0	48	63	3.0	60000.0	1	0	0	0	0
928	19	2020	1.0	47	59	2.0	40000.0	0	0	1	0	0
929	18	2021	1.0	47	64	2.0	50000.0	0	0	0	0	1
930	18	2021	1.0	47	65	1.0	50000.0	0	0	1	0	0
931	19	2020	1.0	47	63	2.0	60000.0	0	0	1	0	0

32 rows × 18 columns

교차 검증

교차 검증이란 데이터 편중을 막기 위해 단순히 데이터를 학습데이터와 테스트 데이터로 나누는것 대신 별도의 여러 세트로 구성된 학습 데이터세트와 테스트 데이터세트에서 학습과 평가를 수행하는 것

=> 본 고사를 치르기 전에 모의고사를 여러번 보는 것

K-FOLD 교차검증

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold,cross_val_score
import numpy as np
```

```
kf = KFold(n_splits=7, shuffle=True)

X= train.drop('value',axis=1)
Y= train['value']
```

```
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
model_xgb = XGBRegressor(n_estimators=500)
model_lgbm = LGBMRegressor(n_estimators=500)

valid_scores = []

for train_idx, valid_idx in kf.split(X,Y) :
    X_tr = X.iloc[train_idx]

    y_tr = Y.iloc[train_idx]

    X_val = X.iloc[valid_idx]

    y_val = Y.iloc[valid_idx]

    model_xgb.fit(X_tr, y_tr)

    valid_prediction = model_xgb.predict(X_val)

    score = RMSE(y_val, valid_prediction)

    valid_scores.append(score)
```

```
for train_idx, valid_idx in kf.split(X,Y) :
    X_tr = X.iloc[train_idx]

    y_tr = Y.iloc[train_idx]

    X_val = X.iloc[valid_idx]

    y_val = Y.iloc[valid_idx]

    model_lgbm.fit(X_tr, y_tr)

    valid_prediction = model_lgbm.predict(X_val)

    score = RMSE(y_val, valid_prediction)

    valid_scores.append(score)
```

모델링

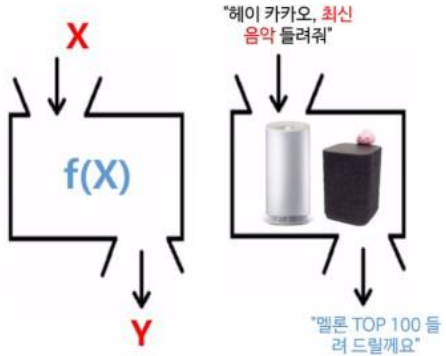
데이터를 기반으로 분석을 하기 위해서는 전,후의 인과관계 혹은 상관관계와 같이 input과 output의 관계를 찾아내야 합니다.

세상의 다양한 현상들은 X와 Y로 설명이 가능한데요.

'매출에 대한 예측', '고장에 대한 예측', '수요에 대한 예측' 등은 어떠한 원인인자에 의해서 영향을 받게 됩니다.

예를 들어, 내년도 수요를 예측한다고 하면 제품에 대한 선호도, 수요환경 등등 다양한 X 인자를 통해서 수요예측(Y)를 찾게 되는 것이죠.

머신러닝에서 의미하는 모델이란 결국 X와 Y의 관계를 찾아낸다고 볼 수 있습니다.



모델링-XGBoost

XGBoost는 트리 기반 학습에서 자주 사용하는 모델 기법중 하나로

GradientBoost의 단점인 느린 수행시간 및 과적합 문제를 해결한 모델

모델링 - LightGBM

LightGBM은 XGBoost와 함께 자주 사용하는 모델 기법으로

XGBoost에 비해 학습에 걸리는 시간이 매우 단축되었고 메모리 사용량도 상대적으로 적다

하이퍼 파라미터 튜닝

하이퍼 파라미터란 모델링할 때 사용자가 직접 세팅해주는 값으로

세부 세팅값이라고 이해하면 편하다.

모델의 하이퍼 파라미터 값을 변경해주면서 최적의 모델 파라미터 값을 찾는것이 중요하다.

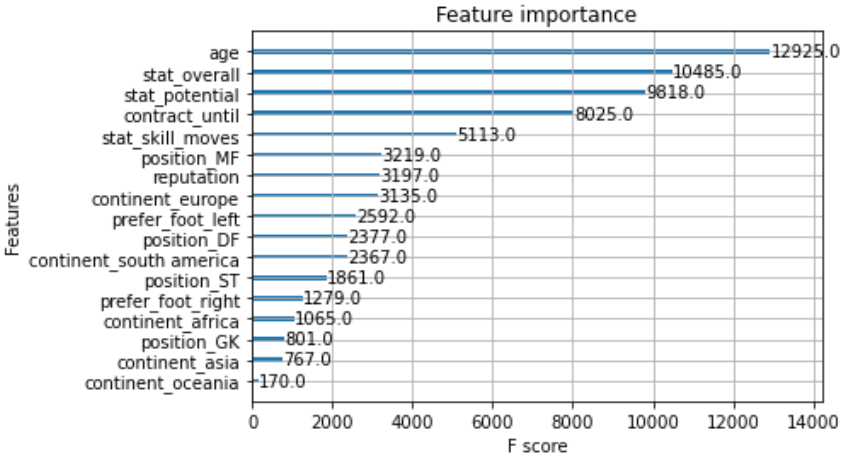
같은 모델이라도 파라미터 값에 따라 결과도 큰 차이를 보여준다.

하이퍼 파라미터 튜닝 방식 (optuna)

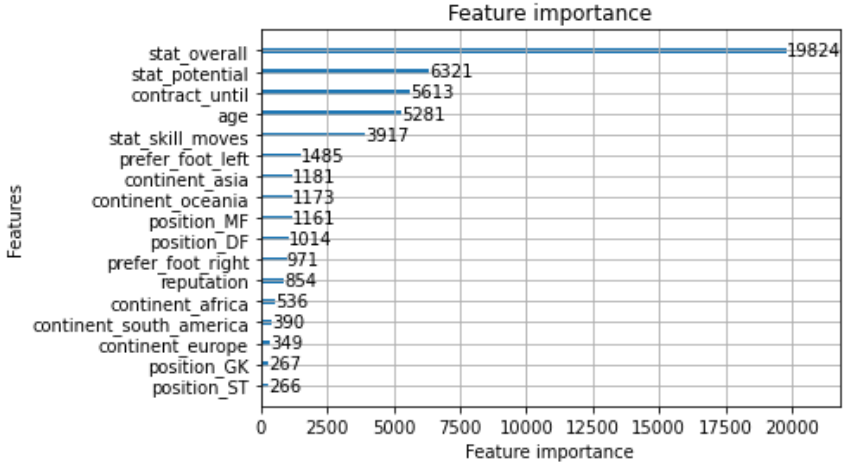
```python
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25)

def objective(trial:Trial):
    params = {
        'objective' : 'reg:squarederror',
        'eval_metric' : 'rmse',
        'learning_rate' : trial.suggest_float('learning_rate',0.01,0.2),
        'n_estimators' : trial.suggest_int('n_estimators',100,5000),
        'min_child_weight' : trial.suggest_int('min_child_weight',1,250),
        'min_split_loss' : trial.suggest_int('min_split_loss',1,5),
        'max_depth' : trial.suggest_int('max_depth',3,10),
        'subsample' : trial.suggest_float('subsample',0.5,1),
        'colsample_bytree' : trial.suggest_float('colsample_bytree',0.1,1),
        'reg_lambda' : trial.suggest_float('reg_lambda',0,1),
        'reg_alpha' : trial.suggest_float('reg_alpha',0,1)
    }

    model = XGBRegressor(**params)
    xgb_model = model.fit(x_train,y_train)
    xgb_pred = model.predict(x_test)
    score = RMSE(y_test, xgb_pred)

    return score
```

```python
from optuna.samplers import TPESampler
study = optuna.create_study(direction='minimize',sampler = TPESampler())
study.optimize(objective, n_trials=25)
print("Best Score:", study.best_value)
print("Best trial:", study.best_trial.params)
```

피처 중요도



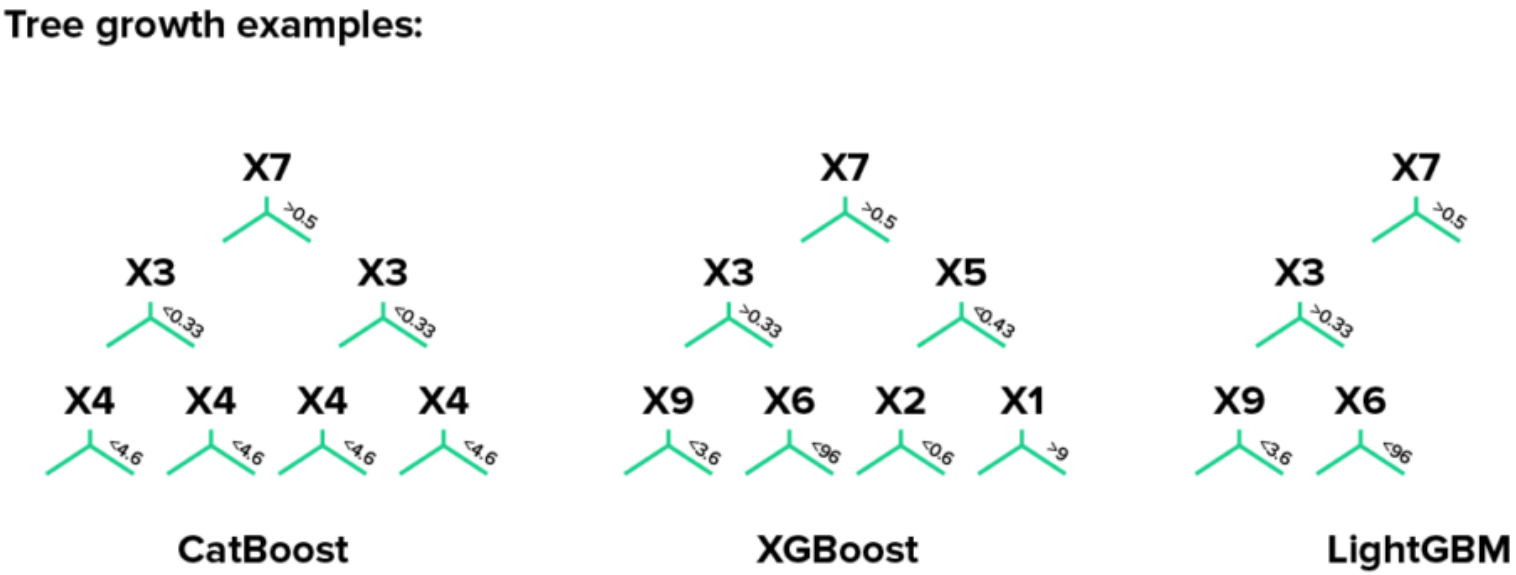

```
from xgboost import plot_importance
plot_importance(xgb)
```




```
from lightgbm import plot_importance
plot_importance(lgbm)
```

5.2 모델링 - catboost

회귀트리 비교



5.2 모델링 - catboost

time	datapoint	class label
12:00	x1	10
12:01	x2	12
12:02	x3	9
12:03	x4	4
12:04	x5	52
12:05	x6	22
12:06	x7	33
12:07	x8	34
12:08	x9	32
12:09	x10	12

cat ->categorical의 약자

-Boosting(부스팅) 기반의 모델

-대부분이 범주형변수로 이루어진 데이터셋에서 예측 성능이 우수하다.

-CatBoost는 범주형 변수를 encoding 작업을 하지 않고도 그대로 모델의 input으로 사용할 수 있다.

-하이퍼파라미터의 영향을 잘 안받는다고 한다.(좋음)

1. 먼저 `x1` 의 잔차만 계산하고, 이를 기반으로 모델을 만든다. 그리고 `x2` 의 잔차를 이 모델로 예측한다.
2. `x1, x2` 의 잔차를 가지고 모델을 만든다. 이를 기반으로 `x3, x4` 의 잔차를 모델로 예측한다.
3. `x1, x2, x3, x4` 를 가지고 모델을 만든다. 이를 기반으로 `x5, x6, z7, x8` 의 잔차를 모델로 예측한다.
4. `... 반복` => 과거의 데이터를 이용해 현재의 데이터를 인코딩 하는 원리

catboost - 코드

```
!pip install catboost
```

```
from catboost import CatBoostRegressor
```

```
import optuna
from optuna import Trial, visualization
from sklearn.metrics import mean_squared_error
def objective(trial):
  param = {
      "random_state":42,
      'learning_rate' : trial.suggest_loguniform('learning_rate', 0.01, 0.2),
      'bagging_temperature' :trial.suggest_loguniform('bagging_temperature', 0.01, 100.00),
      "n_estimators":trial.suggest_int("n_estimators", 1000, 5000),
      "max_depth":trial.suggest_int("max_depth", 4, 16),
      'random_strength' :trial.suggest_int('random_strength', 0, 100),
      "colsample_bylevel":trial.suggest_float("colsample_bylevel", 0.4, 1.0),
      "l2_leaf_reg":trial.suggest_float("l2_leaf_reg",1e-8,3e-5),
      "min_child_samples": trial.suggest_int("min_child_samples", 5, 100),
      "max_bin": trial.suggest_int("max_bin", 200, 500),
      'od_type': trial.suggest_categorical('od_type', ['IncToDec', 'Iter']),
  }
  model = CatBoostRegressor(**param)
  model.fit(x_train,y_train)
  cat_pred = model.predict(x_valid)
  rmse = np.sqrt(mean_squared_error(y_valid, cat_pred))
  return rmse
```

=> catBoostRegressor 불러옴.

=>catboostregressor의 최적의 파라미터값을 찾는 optuna 실행(오래걸

회귀이므로 평가산식 **rmse**사용

catboost - 코드

```
from optuna.samplers import MOTPESampler
study = optuna.create_study(study_name = 'cat_parameter_opt',direction = 'minimize',sampler = MOTPESampler())
study.optimize(objective, n_trials=3)
print("Best Score:",study.best_value)
print("Best trial",study.best_trial.params)
```

```
Best Score: 629592.2791827534
Best trial {'learning_rate': 0.11706728357416832, 'bagging_temperature': 20.245591506467544, 'n_estimators': 4197, 'max_depth': 4, 'random_strength': 26,
 'colsample_bylevel': 0.5347225991452065, 'l2_leaf_reg': 1.8148497510904374e-05, 'min_child_samples': 57, 'max_bin': 208, 'od_type': 'Iter'}
```

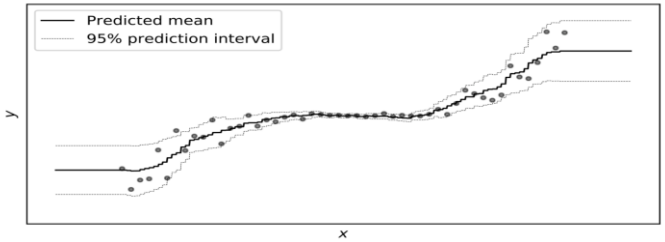
=>최고점수: 629592점, 최적의 파라미터값을 찾음 학습률은 0.1정도, 최대깊이는 4정도로 나옴.

```
cat = CatBoostRegressor(**study.best_params)
cat.fit(x_train,y_train)
pred=cat.predict(X_test)
```

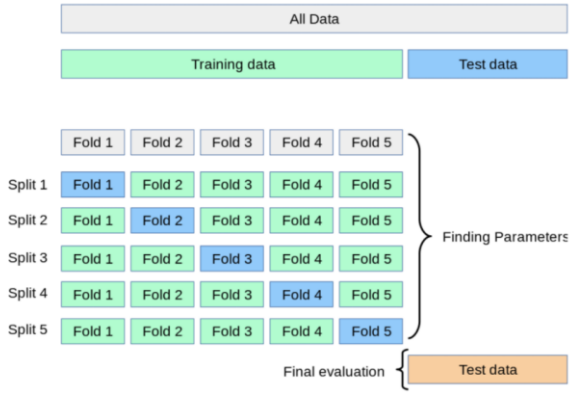
=>catboost모델에 학습시킨후 예측.

5.3 NGBoost(Natural Gradient Boost) (모델링)

- XGBoost와 LightGBM 이후 개발된 부스팅 계열 알고리즘
  - 확률적 예측을 하며, 예측 불확실성을 측정해줌
  - 아래 좌측 그림은 NGBoost로 학습한 확률 회귀 선으로 두꺼운 선은 예측 값의 평균이며, 가는 선은 95% 확률 예측 구간. 이와 같이 확률적 예측 가능
- K 폴드 교차 검증 사용(sklearn cross_val_score 함수)
  - K 개의 데이터 폴드 세트를 만들어 K번만큼 각 폴드 세트에 학습, 검증 평가를 반복적으로 수행
  - 아래 우측 그림에서 5개의 폴드된 데이터 세트를 학습과 검증 데이터를 변경하며 5번 평가 수행
  - 5개 평가 평균한 결과를 통해 예측 성능 평가

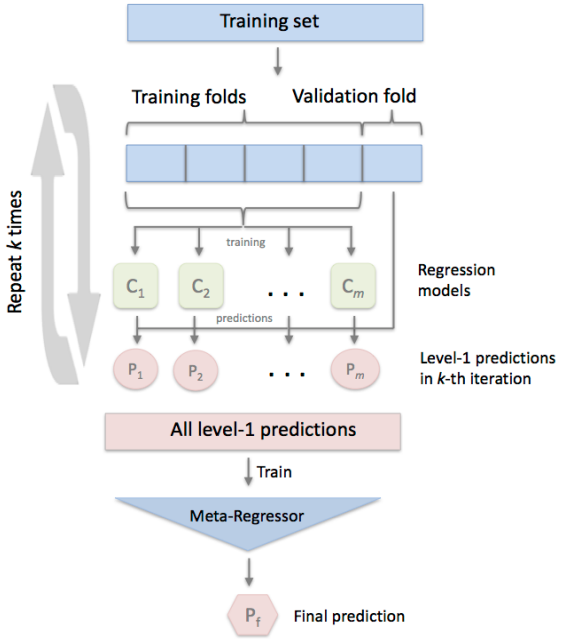



```
ngb = NGBRegressor(random_state = 2020)
cv_ngb = cross_val_score(ngb, X_train, y_train, cv = KFold(n_splits = 4, random_state = 1011, shuffle
= True), scoring = 'neg_mean_squared_error')
```



5.3 CV Stacking (모델링)

- CV 세트 기반 스태킹은 개별 모델들이 각각 교차 검증으로 메타 모델을 위한 학습용 스태킹 데이터 생성과, 테스트용 스태킹 데이터를 생성한 뒤 이를 기반으로 메타 모델이 학습과 예측을 수행(좌측 그림)
- StackingCVRegressor 사용, 하이퍼파라이터 튜닝을 거친 개별 회귀 모델들과, 메타 모델을 입력하여 실행 가능




```
stack = StackingCVRegressor(regressors = (gbm, lgbm), meta_regressor = xgb)
stack.fit(np.array(X), y)
    StackingCVRegressor(meta_regressor=XGBRegressor(base_score=0.5,
                                                    booster='gbtree',
                                                    colsample_bylevel=1,
                                                    colsample_bynode=1,
                                                    colsample_bytree=1.0, gamma=0.0,
                                                    gpu_id=-1,
                                                    importance_type='gain',
                                                    interaction_constraints='',
                                                    learning_rate=0.4,
```

6.앞으로의 계획

매주 수요일(데이콘 분석): 데이콘 연습가능한 문제들을 매주 지속적으로 공부할 예정입니다.

매주 토요일(책 내용 공부) : chapter5~매주 진행예정

감사합니다. (_ _)