# **2021년(제2회) NH투자증권 빅데이터경진대회 모델개발설명서**

**팀명 : 161819**

**팀원 : 김민식, 오홍석, 유광열**

**A: 분석 배경 및 과정**

주식 보유기간을 예측함에 있어 중요한 사항은 예측하고자 하는 사람들에 대한 특징파악, 그 사람들이 구매한 주식에 대한 특징파악이 중요하다고 판단하였다. 또한 주식 보유기간에 관해 다양한 논문들을 읽어보고 기상요인, 사회적인 이슈등 많은 요인들이 주식 보유기간에 영향을 미친다는 것을 알 수 있었지만 모델에 적용시킬때 보유한 주식을 판매할 당시의 상황, 기상요인들을 어떻게 merge 할지에 대한 어려움으로 외부데이터 사용은 하지 않았다.

최대한 다양하고 독립적인 피쳐를 만들기 위해 구성원 각각이 개별적으로 서로 다른 피쳐를 생성하였다. 이후 생성한 여러개의 피쳐 중 가장 성능이 좋은 조합을 찾기 위해 전진선택법을 사용하였다. 피쳐의 데이콘에서 제공한 Baseline 코드에 각각의 피쳐들을 추가하며 성능을 확인해보았다. 이후 세가지 피쳐들 중 두가지 피쳐를 최종적으로 채택하고 해당 피쳐들에 대한 PCA, feature-selection등을 진행하여 모델 개발에 사용하였다. 피쳐에 대한 전처리로 Min-Max Scaling 등을 진행하였지만 성능향상은 없었다. feature-selection하는 과정에서도 다양한 모델들과 SelectPercentile을 활용하여 성능을 비교한 후 가장 좋은 성능을 낸 ExtraTreeRegressor모델로 피쳐를 선택하였다.

모델링을 하는 단계에서는 다양한 시도를 하였다. 첫번째로 모델이 서로 다른 특징을 가지고 있는 종속변수에 대하여 예측을 진행하면 과소적합을 하는 것이라고 판단하여 K-modes, K-prototype등의 방법을 사용하여 군집화를 진행하여 개별적인 모델링을 통해 submission을 도출해냈지만 해당 방법 또한 큰 성능 향상을 가져오지 못하였다. 두 번째로 dnn 모델과 lgbm catboost에 대해서 각각 seed값을 변경하는 방법을 통해 다양한 예측 값(csv)들을 생성하였고 생성된 값들에 대해서 평균(조화평균 혹은 기하 평균)을 통해서 앙상블을 진행하는 방법 또한 진행하였으나 이 또한 성능이 향상되지 않았다. 세번째로는 베이지안 옵티마이저를 활용해서 seed 별로 각각의 모델들에 대한 최적화된 파라미터로 각각 예측 값을 만들어서(csv) 대해서 평균으로 앙상블 진행한 방법도 성능 향상은 없었다.

최종적으로 ‘KNeighborsRegressor’ , ‘ExtraTreesRegressor‘, ‘GradientBoostingRegressor’, ‘XGBRegressor‘, ‘LGBMRegressor‘, ‘CatBoostRegressor‘ 단일모델을 K-fold 방법(K=5)으로 Fold별 RMSE를 계산하고 단일모델들로 가능한 모든 조합에 대하여 앙상블을 진행하고 그중 가장 성능이 좋은 단일모델들을 앙상블 한 것으로 최종결과물을 생성하였다.

**A’: 채택된 변수에 대한 설명**

1. Feature Generation을 위해 사용된 Data

* cus\_info
* iem\_info
* stk\_bnc\_hist
* stk\_hld\_train

2. Base Feature

* 주최 측에서 제공한 BaseLine에서 생성된 Feature를 Base Feature로 사용함.
* 사용된 Base Feature
  + hist\_d: 기존 hold\_d에 0.6을 곱한 거에서 0.5로 바꿈.
  + sex\_dit\_cd
  + cus\_age\_stn\_cd
  + ivs\_icn\_cd
  + cus\_aet\_stn\_cd
  + mrz\_pdt\_tp\_sgm\_cd
  + lsg\_sgm\_cd
  + tco\_cus\_grd\_cd
  + tot\_ive\_te\_sgm\_cd
  + mrz\_btp\_dit\_cd
  + btp\_cfc\_cd
  + mkt\_pr\_tal\_scl\_tp\_cd
  + stk\_dit\_cd
  + bnc\_qty
  + tot\_aet\_amt
  + stk\_par\_pr
  + stk\_p

3. Feature Generation

* byn\_dt\_last:
  + 매수일자의 해당 년도 마지막 일자에서 매수 일자를 빼서 byn\_dt\_last를 만듬
  + byn\_dt\_last가 Hist\_d보다 작을 경우 byn\_dt\_last에서 Hist\_d를 만듬
  + byn\_dt\_last가 Hist\_d보다 큰 경우 Hist\_d를 그대로 사용
* hist/tot\_(h,l,m):
  + 총 투자기간이 구간을 high, low, midian 값으로 만듬, 총 투자기간\_(h,l,m)
  + hist\_d / 총\_투자기간(h,l,m)을 나누어 해당 Feature를 만듬
* byn\_last\_(h,l,m):
  + byn\_dt\_last / 총\_투자기간(h,l,m)을 나누어 해당 Feature를 만듬
* bnc\_qty\_s:
  + 계좌,종목코드 별 bnc\_qty sum한 Feature
* tot\_aet\_amt\_s:
  + 계좌,종목코드 별 tot\_aet\_amt sum한 Feature
* tot\_aet/aet\_stn\_(h,lm):
  + 자산구간을 high, low, midian 값으로 만듬, 총자산\_(h,l,m)
  + tot\_aet\_amt/총자산\_(h,l.m)
* profit/tot\_aet:
  + 서비스등급을 가지고 수익기여도를 만듬
  + 수익기여도/tot\_aet\_amt
* profit/tot\_bnc\_qty:
  + train, test 나눠서 계좌별 전체 거래량을 구함
  + 수익기여도\*잔고수량/계좌별전체거래량로 해당 Feature를 만듬
* bnc\_qty/tot\_bnc\_qty:
  + 잔고수량/계좌별전체거래량로 해당 Feature를 만듬
* aet\_amt/tot\_aet:
  + 잔고금액/계좌별전체잔고금액로 해당 Feature를 만듬
* 계죄, 종목 별로 stk\_hld\_train,test 데이터를 shift하여 반복문을 사용하여 연산함
* 종목별 피처
* 매수 Feature
  + 계좌별\_평균\_일매수\_수량
  + 계좌별\_최고\_일매수\_수량
  + 계좌별\_매수\_최소\_주당금액
  + 계좌별\_매수\_최대\_주당금액
  + 계좌별\_매수\_평균\_주당금액
* 계좌별 이전 주식 보유기간 Feature
  + 계좌별\_평균보유기간
  + 게좌별\_최대보유기간
* 계죄별 매수,매도 합친 Feature
  + 계좌별\_평균\_일매수매도\_수량
  + 계좌별\_최고\_일매수매도\_수량
  + 해당일시\_주당금액
  + 계좌별\_매수매도\_최소\_주당금액
  + 계좌별\_매수매도\_최대\_주당금액
  + 계좌별\_매수매도\_평균\_주당금액
* 매수 Feature
  + 종목별\_평균\_일매수\_수량
  + 종목별\_최고\_일매수\_수량
  + 종목별\_매수\_최소\_주당금액
  + 종목별\_매수\_최대\_주당금액
  + 종모별\_매수\_평균\_주당금액
* 종목별 이전 주식 보유기간 Feature
  + 종목별\_최대보유기간
  + 종목별\_평균보유기간
* 종목별 매수,매도 합친 Feature
  + 종목별\_평균\_일매수매도\_수량
  + 종목별\_최고\_일매수매도\_수량
  + 종목별\_매수매도\_최소\_주당금액
  + 종목별\_매수매도\_최대\_주당금액
  + 종모별\_매수매도\_평균\_주당금액

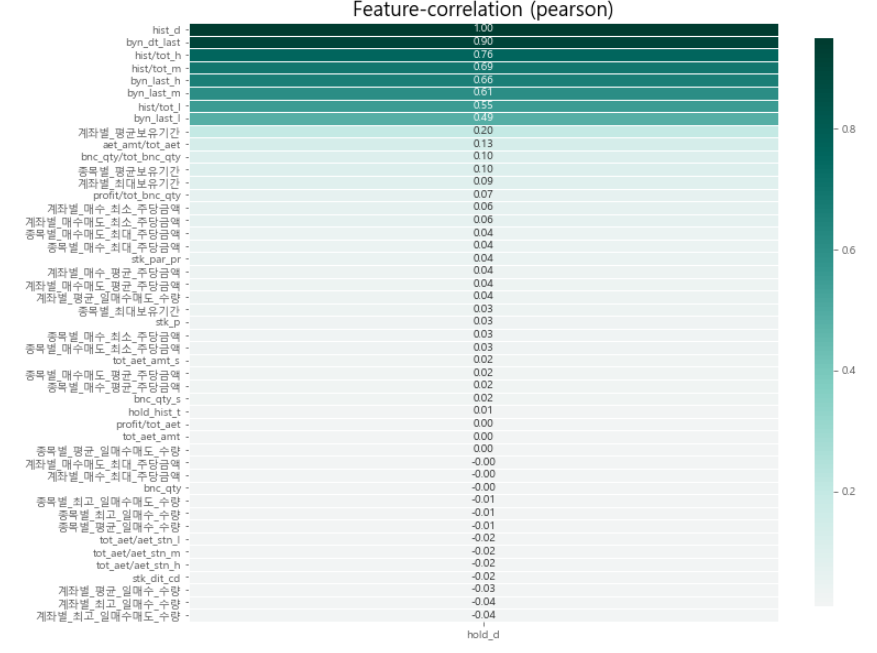
4. Feature Merge

* Base Feature + Feature Generation
* 총 58개의 Feature 사용

5. Feature Split

* Category Feature
  + sex\_dit\_cd
  + cus\_age\_stn\_cd
  + ivs\_icn\_cd
  + cus\_aet\_stn\_cd
  + mrz\_pdt\_tp\_sgm\_cd
  + lsg\_sgm\_cd
  + tco\_cus\_grd\_cd
  + tot\_ivs\_te\_sgm\_cd
  + mrz\_btp\_dit\_cd
  + btp\_cfc\_cd
  + mkt\_pr\_tal\_scl\_tp\_cd
* Continuos Feature
  + Category Feature를 제외한 나머지 Feature

6. Target Correlation



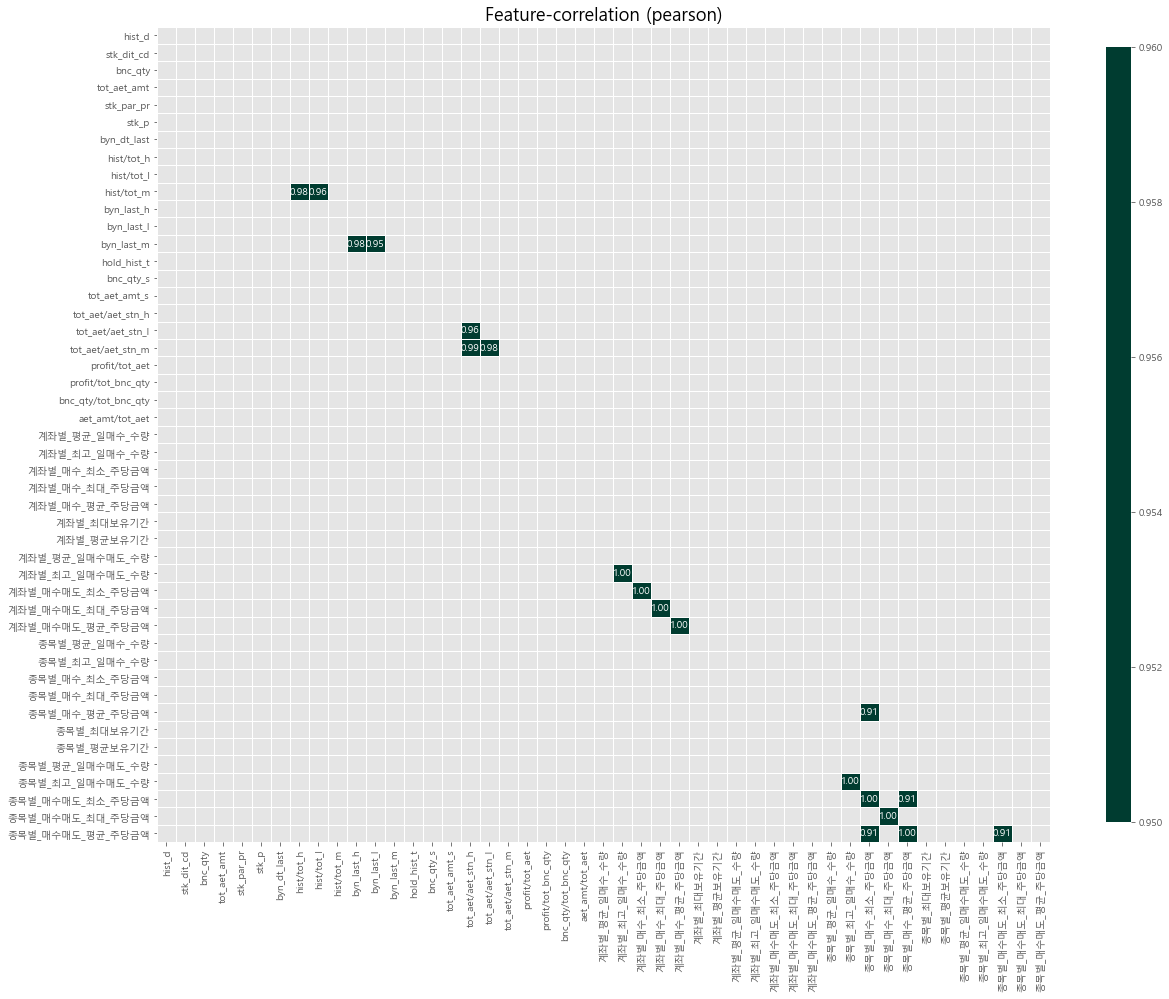
* Hold\_d와 Feature간에 상관계수를 확인해봄.

7. Statistical verification



* 해당 Feature와 target간의 상관관계에 대한 P-value를 확인하여 Feature의 상관관계의 통계적 정당성을 확인함.

8. Feature Correlation



* Feature간의 상관관계를 확인함
* 상관관계가 0.9이상인 Feature를 제거함
* 제거될 Feature를 쌍을 보고 target과 상관관계가 더 낮은 Feature를 제거함.
* 제거된 Feature
  + hist/tot\_m
  + byn\_last\_m
  + tot\_aet/aet\_stn\_l
  + 계좌별\_최고\_일매수\_수량
  + 계좌별\_매수매도\_최소\_주당금액
  + 계좌별\_매수매도\_평균\_주당금액
  + 계좌별\_매수\_최대\_주당금액
  + 종목별\_매수\_평균\_주당금액
  + 종목별\_최고\_일매수매도\_수량
  + 종목별\_매수매도\_최소\_주당금액
  + 종목별\_매수매도\_평균\_주당금액
  + 종목별\_매수매도\_최대\_주당금액

9. Category Feature One-Hot-Encoding

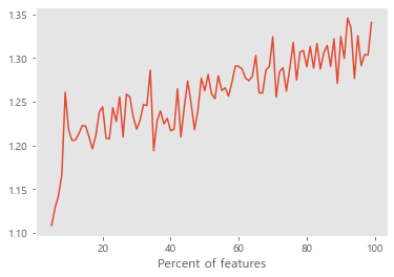
* Category Feature One-Hot-Encoding을 한 결과 76개 생성

10. Category Feature One-Hot-Encoding PCA

* PCA를 진행하여 분산의 설명량이 95% 이상이 되는 차원의 수를 구함
* PCA를 진행한 결과 37개의 pca로 줄어듬

11. Feature Selection

* ExtraTreeRegressor모델과 SelectPercentile를 활용하여 Feature를 selection함



* 해당 결과 best보다 38%일때 더 좋은 성능을 보여 Feature의 38%로만 사용함
* 최종 사용된 Feature
  + 'hist\_d',
  + 'stk\_dit\_cd',
  + 'tot\_aet\_amt',
  + 'stk\_par\_pr',
  + 'stk\_p',
  + 'byn\_dt\_last',
  + 'hist/tot\_h',
  + 'hist/tot\_l',
  + 'byn\_last\_h',
  + 'byn\_last\_l',
  + 'bnc\_qty\_s',
  + 'tot\_aet\_amt\_s',
  + 'profit/tot\_bnc\_qty',
  + 'bnc\_qty/tot\_bnc\_qty',
  + 'aet\_amt/tot\_aet',
  + '계좌별\_평균\_일매수\_수량',
  + '계좌별\_매수\_최소\_주당금액',
  + '계좌별\_매수\_평균\_주당금액',
  + '계좌별\_최대보유기간',
  + '계좌별\_평균보유기간',
  + '계좌별\_최고\_일매수매도\_수량',
  + '종목별\_매수\_최소\_주당금액',
  + '종목별\_매수\_최대\_주당금액',
  + '종목별\_평균보유기간',
  + 'pca\_1',
  + 'pca\_2',
  + 'pca\_5'

**B : 분석 범주**

* Feature 中 계좌(고객)별 종목별 각각 주식의 매수와 매도에 대한 통계치(평균, 최소, 최대)를 구하기 위해 hist를 사용할 때, 주당금액과 수량이 음수인 값은 제거하고 통계치를 구하였다.

(이유: 수량과 주당금액이 마이너스인 값들이 지나치게 크고, shift했을 경우 값을 왜곡시킬 수 있기 때문에 거하였다.)

* 총 투자 한 기간(baseline Feature) 중에 해당 주식을 얼마나 보유 기간의 비율(Feature 명: hist/tot\_(h,l,m))에 만들기 위한 총 투자 기간의 범주의 값을 기간 별 최소, 중간, 최대 일수를 나눠서 비율을 계산하였다. 범주는 다음 표와 같다.

(구간 값 > 하나의 정수값으로 대체)

공식: hist\_d / 해당 고객의 총 투자 기간(low, middle, high 값)

|  | low(최소 보유) | middle(중간 보유) | high(최대 보유) |
| --- | --- | --- | --- |
| 01: 6개월 미만 | 30일 | 90일 | 182일 |
| 02: 6개월 이상 ~ 1년 미만 | 185일 | 279일(1.5년) | 364일(1년) |
| 03: 1년 이상 ~ 3년 미만 | 365일(1년) | 730일(2년) | 1,094일(3년) |
| 04: 3년 이상 ~ 5년 미만 | 1,094일(3년) | 1,460일(4년) | 1,824일(5년) |
| 05: 5년 이상 ~ 10년 미만 | 1,824일(5년) | 2,555일(7년) | 3,649일(10년) |
| 06/ 10년 이상 | 3,649일(10년) | 5,475일(15년) | 7,300일(20년) |

* 고객 별 총 자산 구간 (baseline Feature) 대비 해당 주식에 거래한 금액의 비율(Feature명: 총자산\_h(I/m) ) 만들기 위해서 고객의 자산 구간 범주의 값을 구간의 최소, 중간 최대 값을 나눠서 계산하였다. 범주는 다음표와 같다.

(구간 값 > 하나의 정수값으로 대체)

공식: 잔고금액/ 해당 고객의 총자산 구간(low, middle, high 값)

|  | low(최소 금액) | middle(중간 금액) | high(최대 금액) |
| --- | --- | --- | --- |
| 1: 0원 ~ 1천만원 미만 | 백만원 | 5백만원 | 1천만원 |
| 2: 1천만원 ~ 3천만원 미만 | 천만원 | 2천만원 | 3천만원 |
| 3: 3천만원 ~ 5천만원 미만 | 3천만원 | 4천만원 | 5천만원 |
| 4: 5천만원 ~ 1억원 미만 | 5천만원 | 7천 5백만원 | 1억원 |
| 5: 1억원 이상 ~ 3억원 미만 | 1억원 | 2억원 | 3억원 |
| 6: 3억원 이상 | 3억원 | 4억원 | 5억원 |

* 고객 별 서비스 등급 (baseline Feature)을 활용한 파생 변수(거래 금액 대비 수익 기여도) 만들기 위해서 고객의 서비스 등급의 수익 기여도를 가지고 왔으며 범주값을 기준 값으로 대체함 (구간 값 > 하나의 정수값으로 대체)

|  | 수익 기여도 |
| --- | --- |
| 01: 탑클래스(5백만원 이상) | 5백만원 |
| 02: 골드(3백만원 이상) | 3백만원 |
| 03: 로얄(백만원 이상) | 백만원 |
| 04: 그린(오십만원 이상) | 오십만원 |
| 05: 블루(십만원 이상) | 십만원 |
| 9: 등급 미 정의 / 00(미정의) | 십만원 |

**C: 모델링**

1. 모델 예측값 제한

* 2021년 데이터 기간(예측)은 2021년 1월 1일 ~ 7월 31일로 제한되어 있기 때문에 Test 데이터 Hold\_d(예측해야할 대상)이 최대값은 146일이다. 모델이 예측한 보유기간 (Hold\_d)이 과거 보유일(hist\_d)에 146을 더한값보다 크면 값을 제한해준다.

(최대값: hist\_d +146)

>> 공식: Hold\_d(보유기간) - hist\_d(~ 20년 12월 31일까지 보유기간) <= 146

* 2020년 12월 30일까지는 보유 후 2021년 데이터 기간 이후의 보유을 가정하고 있기 때문에 Test 데이터 Hold\_d(예측해야할 대상)이 hist\_d보다 작은 값이 나올수가 없다. 모델이 예측값이 Hold\_d가 hist\_d보다 낮을 경우 hist\_d +1으로 제한해준다.

(최소값 : hist\_d + 1)

.>> 공식: Holdd\_d(보유기간) - hist\_d ( ~ 12월 31일까지 보유기간) >= 1

1. 모델 진행 과정

* 총 6개의 모델을 선정해서 학습을 진행

(‘KNeighborsRegressor’ , ‘ExtraTreesRegressor‘, ‘GradientBoostingRegressor’, ‘XGBRegressor‘, ‘LGBMRegressor‘, ‘CatBoostRegressor‘ )

* train 데이터 내에서 K-fold cross-validation(k-fold = 5)을 통해서 train, validation을 나눠서 학습을 진행해서 rmse(성능 지표)를 봄 / 이 때 모델 성능이 제일 안 좋은 Knn은 제외하고 앙상블 진행

| 모델 | K-FOLD validation 평균 Rmse |
| --- | --- |
| KNeighborsRegressor | 56.198 |
| ExtraTreesRegressor | 0.475 |
| GradientBoostingRegressor | 0.564 |
| XGBRegressor | 0.441 |
| LGBMRegressor | 0.892 |
| CatBoostRegressor | 0.962 |

* 각각의 단일 모델의 성능 파악한 후 모델별 가능한 조합별 앙상블을 진행한 후 가장 성능이 좋은 조합의 모델로 앙상블을 진행 결과 제출

[2개의 조합]

| 모델 조합 | K-FOLD validation 평균 Rmse |
| --- | --- |
| ExtraTreesRegressor●GradientBoostingRegressor | 0.476 |
| ExtraTreesRegressor●XGBRegressor | 0.441 |
| ExtraTreesRegressor●LGBMRegressor | 0.624 |
| ExtraTreesRegressor●CatBoostRegressor | 0.655 |
| GradientBoostingRegressor●XGBRegressor: | 0.470 |
| GradientBoostingRegressor●LGBMRegressor | 0.617 |
| GradientBoostingRegressor●CatBoostRegressor | 0.644 |
| XGBRegressor●LGBMRegressor | 0.588 |
| XGBRegressor●CatBoostRegressor | 0.615 |
| LGBMRegress:or●CatBoostRegressor | 0.883 |

[3개의 조합]

| 모델 조합 | K-FOLD validation 평균 Rmse |
| --- | --- |
| ExtraTreesRegressor●GradientBoostingRegressor  ●XGBRegressor | 0.451 |
| ExtraTreesRegressor●GradientBoostingRegressor  ●LGBMRegresso | 0.542 |
| ExtraTreesRegressor●GradientBoostingRegressor  ●CatBoostRegressor | 0.559 |
| ExtraTreesRegressor●XGBRegressor  ●LGBMRegressor | 0.528 |
| ExtraTreesRegressor●XGBRegressor  ●CatBoostRegressor | 0.544 |
| ExtraTreesRegressor●LGBMRegressor  ●CatBoostRegressor | 0.702 |
| GradientBoostingRegressor●XGBRegressor  ●LGBMRegressor | 0.525 |
| GradientBoostingRegressor●XGBRegressor  ●CatBoostRegressor | 0.539 |
| GradientBoostingRegressor●LGBMRegressor  ●CatBoostRegressor | 0.687 |
| XGBRegressor●LGBMRegressor●  CatBoostRegressor | 0.674 |

[4개의 조합]

| 모델 조합 | K-FOLD validation 평균 Rmse |
| --- | --- |
| ExtraTreesRegressor●GradientBoostingRegressor  ●XGBRegressor●LGBMRegressor | 0.497 |
| ExtraTreesRegressor●GradientBoostingRegressor  ●XGBRegressor●LGBMRegressor | 0.507 |
| ExtraTreesRegressor●GradientBoostingRegressor  ●LGBMRegressor●CatBoostRegressor | 0.609 |
| ExtraTreesRegressor●XGBRegressor  ●LGBMRegressor●CatBoostRegressor | 0.601 |
| GradientBoostingRegressor●XGBRegressor  ●LGBMRegressor●CatBoostRegressor | 0.591 |

5개의 조합(전체) : 0.553

이 중에서 가장 성능이 제일 잘 나온 ExtraTreesRegressor●XGBRegressor의 조합으로 앙상블을 진행한 뒤 결과를 예측해서 제출