* 기본적인 모델들은 파이캐럿으로 돌릴 수 있어요 !
* 데이콘의 다른 코드들을 참고해서 해당 모델에 대해 간단하게 조사해오면 좋을 것 같습니다 !  
  + 원리 (간단하게)
  + 성능( 어떤 데이터에서 사용할 때 좋은지 ? )

### **모델 리스트**

* XGBoost Regressor  
    
   최적화된 그래디언트 부스팅 구현 가능한 파이썬 라이브러리  
  + 원리
    - 이전 모델이 과소적합한 샘플에 가중치 부여해 다음 모델에 예측 정확도 높이는 방식으로 모델 보완해가는 부스팅 기법 사용
  + 성능
    - 예측력이 좋아 주로 많이 사용함
    - 다른 알고리즘과 연계해 앙상블 학습 가능
    - 과적합 방지 가능
  + XGBoost는 훈련시간이 길어서 훈련 시간이 짧으면서 성능이 좋은 LightGBM을 더 많이 사용한다고 함
* LightGBM  
    
   그래디언트 부스팅 발전 → XGBoost → 속도 개선 → LightGBM  
  + 원리  
    - 리프 기준 분할 방식 사용
    - 트리의 균형 맞추지 않고 최대 손실값을 갖는 리프 노드를 지속적으로 분할하여 예측 오류 손실 최소화
  + **num\_leaves**: 클수록 정확도는 높아지지만 오버피팅 발생 가능  
      
     **min\_data\_in\_leaf**: 클수록 오버피팅 방지  
      
     **max\_depth**: 위 두개 파라미터와 결합하여 오버피팅 방지  
      
     **objective**: 사용하는 데이터셋의 타겟팅 값의 형태에 따라 조정 필요  
      
     **metric**: 성능 평가를 어떤 것으로 할 것인지 조정 필요
  + 성능  
    - 학습 시간이 짧음 ( 통상 XGBoost의 1.3~1.5배 )
    - 메모리 사용량이 상대적으로 적음
    - 대용량 데이터 처리 가능
    - 10,000건 이하 데이터셋에서는 오버피팅 발생 가능
* Gradient Boosting Regressor  
  + 원리
    - 여러 개의 약한 학습기를 순차적으로 학습, 예측하면서 잘못 예측한 데이터에 가중치 부여를 통해 오류를 개선해 나가면서 학습하는 방식
    - 가중치 업데이터를 경사 하강법을 이용
  + 하이퍼 파라미터
    - n\_estimator : 트리 개수 지정, 많을수록 성능 높아지지만 수행 시간 오래 걸림
    - max\_features, max\_depth, min\_samples\_leaf, min\_samples\_split
    - loss : 경사하강법에서 사용할 비용함수, 기본값 deviance
    - learning rate : 학습률, 0~1 사이의 값, 작을수록 예측 성능 높아지지만 수행시간 오래 걸림 (⇒ 너무 작게 설정하면 반복이 완료되어도 최소 오류값 찾지 못할 수 있음)
  + 성능
    - 일반적으로 랜덤 포레스트보다 예측 성능이 뛰어남
    - 수행시간 오래 걸리고, 하이퍼 파라미터 튜닝 노력이 필요
    - 대용량 데이터의 경우 학습에 매우 많은 시간이 필요하므로 수행 시간 문제 극복이 중요
    - 대안으로 XGBoost 사용
* Stacking  
  + 원리
    - 개별적인 여러 알고리즘을 서로 결합해 예측 결과 도출 + 메타 모델로 최종 학습 및 예측
    - 즉, 개별적인 기반 모델과 개별 기반 모델의 예측 데이터를 학습데이터로 하는 최종 메타 모델 필요
  + 성능
    - 여러 모델의 학습 결과로 메타 학습을 거치기 때문에 학습 데이터에 대한 우수한 성능
    - 주로 경진대회에서 마지막 성능 올리기 방법으로 쓰는 기법들 중 하나
    - 과적합을 개선하기 위해 CV 세트 기반 스태킹을 사용해야 함
* Random Forest Regressor  
    
   배깅(bagging)의 대표적인 알고리즘  
  + 원리
    - 다수의 의사결정나무 모델에 의한 예측을 종합하는 앙상블 방법
    - Boostrap 기법을 이용해 다수의 training data를 생성한 후(bagging) 생성된 training data로 decision tree 모델을 구축하고 예측을 종합한다.
  + 성능
    - decision tree에서의 과적합의 위험을 줄일 수 있다.
    - 데이터의 크기가 커도 계산량이 낮다.
    - Diversity와 Randomness를 확보한 알고리즘이기 때문에 성능이 좋음
* NGB Regressor  
    
   Natural Gradient Boost  
  + 원리
    - 기본적인 학습기는 결정트리이다.
    - weak learner로 예측한 결과를 활용해 확률 분포를 만들고 실제 target value를 토대로 성능 측정
  + 성능
    - 확률적인 예측을 해준다. 즉, 예측의 불확실성을 측정해줌

### **MAE**

* MAE(Mean Absolute Error, 평균 절대 오차)  
    
  + 개념 : 모든 절대오차의 평균
  + 특징
    1. MSE는 오차 제곱의 평균으로 오차가 커질수록 손실함수의 값이 급격하게 증가하지만, MAE는 절대 오차의 평균이므로 오차와 비례하게 손실이 증가
    2. Outlier에 robust(강건)한 특징을 가짐
    3. 모든 오차에 동일한 가중치를 부여함

### **회귀 성능 향상**

1. 과적합 방지  
   * 투입하는 feature 제한하기
   * 단계적 회귀(stepwise regression)  
       
      target에 대한 영향력이 약한 feature를 단계적으로 제외해나가며 가장 영향력이 큰 feature들만 분석에 사용하는 기법  
       
      영향력 << 정의하는 것에 따라 다름 주로 상관관계  
     + 상관관계 기준으로 낮은 feature들을 제거해가며 성능 확인
2. 과적합 방지 in XGBoost  
   * eta 값 낮추기 / 만약 eta값을 낮추게 된다면 num\_estimators는 반대로 높여줘야 합니다.
   * max\_depth 낮추기
   * min\_child\_weigh 높이기
   * gamma 값 높이기
3. 이상치 제거
4. 여러 개의 모델을 조합해 사용(앙상블)