통계분석실습

KOSPI 기업의 수익률 예측

IT 공학과 1812797 이정민
통계학과 1816793 장은조
통계학과 1816001 홍주리

Summary

Data

Y : 수익률(simple monthly return)

X : 73개의 설명변수 모두 사용

직전 5년치 데이터로 그 다음 1년을 예측

1년씩 time window rolling

Training					Test
2002	2003	2004	2005	2006	2007

Training					Test
2003	2004	2005	2006	2007	2008

.
.
.

Training					Test
2013	2014	2015	2016	2017	2018

Model

Chapter1. 회귀 모형

1.1 다중선형회귀

1.2 KNN regression

1.3 Linear Median Regression

1.4 Quantile Regression Forest

Chapter2. Ensemble(1)

2.1 Bagging

2.2 Random Forest

Chapter3. Ensemble(2)

3.1 Gradient Boosting Machine(GBM)

3.2 eXtrem Gradient Boosting (XGB)

Summary

Data

Y : 수익률(simple monthly return)

X : 73개의 설명변수 모두 사용

직전 5년치 데이터로 그 다음 1년을 예측

Rolling Window Scheme

Training					Test
2002	2003	2004	2005	2006	2007

Training					Test
2003	2004	2005	2006	2007	2008

.
.
.

Training					Test
2013	2014	2015	2016	2017	2018

Model

Chapter1. 회귀 모형

1.1 다중선형회귀

1.2 KNN regression

1.3 Linear Huber Regression

1.4 Quantile Regression Forest

Chapter2. Ensemble(1)

2.1 Bagging

2.2 Random Forest

Chapter3. Ensemble(2)

3.1 Gradient Boosting Machine(GBM)

3.2 eXtrem Gradient Boosting (XGB)

✓ 각 모델별 예측 수익률을 기준으로 기업을 10개 그룹 / 20개 그룹으로 나눈 후 SR과 소요시간 비교

✓ SR과 모델 소요 시간 2가지 요소를 고려하여 최종 모델 선정

Chapter1. 회귀 모형

1.1 Multiple Linear Regression

SR

✓ 10개의 그룹으로 나눈 경우

Avg	Std	SR
0.87	6.73	0.45

✓ 20개의 그룹으로 나눈 경우

Avg	Std	SR
1.37	8.28	0.57

기준값

소요 시간

✓ 12.81237 secs

10개의 그룹으로 나누었을 때 보다 20개의 그룹으로 나누었을 때 SR이 더 커짐을 확인 할 수 있었다.

다중선형회귀의 경우 모델 실행 시간이 약 13초로 **오래 걸리지 않는다**는 장점을 갖고 있다.

1.2 KNN Regression

KNN Regression

KNN Regression 의 경우 K값이 너무 작으면 over fitting 너무 크면 under fitting한 경향을 보이게 된다.

⇒ **적절한 K값**을 선택하는 것이 중요!

```
result1<-c()  #KNN K값 1부터 50까지 돌렸을때 SR누적한 저장소
for ( n in 1:50 ){
  (중간과정생략)
result<-data.frame(Avg=Avg_vw,Std=Std_vw,SR=SR_vw)

result1<-c(result1,result$SR)

}
```

수행 시간을 고려하여 K 값의 범위를 1부터 50까지로 설정한 뒤,

이 중 SR을 극대화 해주는 K값을 찾기 위해 for문을 활용하였다.

→ 약 5분 정도의 시간 소요

1.2 KNN Regression

SR

✓ 10개의 그룹으로 나눈 경우
k = 14

Avg	Std	SR
1.49	5.62	0.92

✓ 20개의 그룹으로 나눈 경우
k = 23

Avg	Std	SR
2.17	7.68	0.98

소요 시간

✓ 5.494469 secs

예측 수익률을 기준으로 20개 그룹으로 나누었을 때 SR 값이 10개 그룹으로 나누었을 때 보다 더 크다.

K값만 정해진다면 KNN 모델을 적합 시키는데 걸린 시간은 5초로 **오래 걸리지 않는다**는 장점을 갖고 있다.

1.3 Linear Median Regression

Linear Median Regression

SR

✓ 10개의 그룹으로 나눈 경우

Avg	Std	SR
0.92	7.95	0.4

✓ 20개의 그룹으로 나눈 경우

Avg	Std	SR
1.89	11.32	0.58

소요 시간

✓ 3.165914 mins

20개의 그룹으로 나누었을 때 더 큰 SR을 갖고 있으며,
10개의 그룹으로 나눈 경우 SR이 0.4로 기준점 0.45보다 작게 나와 의미가 없다.

1.4 Quantile Regression Forest

Quantile Regression Forest

SR

✓ 10개의 그룹으로 나눈 경우

Avg	Std	SR
1.14	7.07	0.56

✓ 20개의 그룹으로 나눈 경우

Avg	Std	SR
2.14	10.68	0.69

소요 시간

✓ 28.02792 mins

Quantile Regression은 Random forest 의 아이디어를 분위수 회귀에 적용한 모델로, ranger함수에서 'quantreg=True'로 설정하여 구현한다.
Randomforest의 아이디어를 가져왔기 때문에 **seed를 설정**해 주었다.

Chapter2. Ensemble(1)

2.1 Bagging

- 모델의 수행시간을 고려하여 **nbagging = 100**으로 설정하였다. 즉, 100번의 반복추출로bootstrap을 진행한다는 의미이다.
- **Coob=True**로 설정하여 out-of-bag(OOB)도 고려하도록 설정하여 정확도를 높이고자 했다. 여기서 OOB는 반복추출로 뽑히지 않은 데이터를 의미한다.

Bagging parameter

```
# S3 method for factor
ipredbagg(y, X=NULL, nbagg=25, control=
                  rpart.control(minsplit=2, cp=0, xval=0),
                  comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, …)
# S3 method for numeric
ipredbagg(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
                   comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, …)
# S3 method for Surv
ipredbagg(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
                comb=NULL, coob=FALSE, ns=dim(y)[1], keepX = TRUE, …)
# S3 method for data.frame
bagging(formula, data, subset, na.action=na.rpart, …)
```

```
bagging(ret~.-code-tp-yy-mm, data=dm1_train, nbagging=100,coob=TRUE)
```

2.1 Bagging

SR

✓ 10개의 그룹으로 나눈 경우

Avg	Std	SR
-0.01	4.14	-0.01

✓ 20개의 그룹으로 나눈 경우

Avg	Std	SR
0.06	5.76	0.04

소요 시간

8.660004 mins

10개 그룹으로 나눈 경우 SR 값이 음수가 나왔으므로 유의미하지 않은 결과이다.

20개 그룹으로 나눈 경우 SR 값이 양수가 나왔지만 0.04로 매우 작기때문에 유의미하지 않은 결과이다.

두 경우 모두 수행시간이 10분 이내로 회귀모형 보다는 오래 걸리는 편이다.

2.2 Random Forest

- Randomforest()함수보다 ranger() 함수가 **시간적인 측면**에서 **더 효율적**이기 때문에 Randomforest를 ranger() 함수를 이용해서 구현하였다.
- 다양한 parameter가 있지만 default 값만으로 충분한 결과를 낼 수 있다고 생각했기 때문에 따로 **설정하지 않았다.**

Ranger parameter

```
ranger(
  formula = NULL,
  data = NULL,
  num.trees = 500,
  mtry = NULL,
  importance = "none",
  write.forest = TRUE,
  probability = FALSE,
  min.node.size = NULL,
  max.depth = NULL,
  replace = TRUE,
  sample.fraction = ifelse(replace, 1, 0.632),
  case.weights = NULL,
  class.weights = NULL,
  splitrule = NULL,
  num.random.splits = 1,
  alpha = 0.5,
  minprop = 0.1,
  split.select.weights = NULL,
  always.split.variables = NULL,
  respect.unordered.factors = NULL,
  scale.permutation.importance = FALSE,
  local.importance = FALSE,
  regularization.factor = 1,
  regularization.usedepth = FALSE,
  keep.inbag = FALSE,
  inbag = NULL,
  holdout = FALSE,
  quantreg = FALSE,
  oob.error = TRUE,
  num.threads = NULL,
  save.memory = FALSE,
  verbose = TRUE,
  seed = NULL,
  dependent.variable.name = NULL,
  status.variable.name = NULL,
  classification = NULL,
  x = NULL,
  y = NULL,
  ...
)
```

2.2 Random Forest

SR

✓ 10개의 그룹으로 나눈 경우

Avg	Std	SR
1.04	6.63	0.54

✓ 20개의 그룹으로 나눈 경우

Avg	Std	SR
0.88	7.73	0.39

소요 시간

22.95309 mins

10개 그룹으로 나눈 경우 SR 값이 0.54로 목표했던 0.4보다 크므로 유의미한 결과이다.

두 경우 비교했을 때, random forest의 경우 10개 그룹으로 나눈 것이 더 효과적이라고 할 수 있다.

SR 대비 수행시간이 **20분을 초과**했으므로 모델의 효율을 따졌을 때 **좋다고 보기 어렵다.**

Chapter3. Ensemble(2)

3.1 Gradient Boosting Machine

- Gradient Boosting Algorithm (GBM)은 회귀분석 또는 분류 분석을 수행할 수 있는 **부스팅 계열**의 예측모형이다.
- 머신러닝 알고리즘 중에서도 **예측 성능이 높다**고 알려진 알고리즘이다.
- 여러 트리로 구성된 앙상블 모델로 결과를 예측할 때, **gradient descent**를 이용해 순차적으로 트리를 만들어가며 **이전 트리의 오차를 보완**하는 방식
- 이 때, **가중치를 부여**하는 방법으로 **Gradient descent(경사하강법)**을 사용한다.

GBM parameter

```
gbm(
  formula = formula(data),
  distribution = "bernoulli",
  data = list(),
  weights,
  var.monotone = NULL,
  n.trees = 100,
  interaction.depth = 1,
  n.minobsinnode = 10,
  shrinkage = 0.1,
  bag.fraction = 0.5,
  train.fraction = 1,
  cv.folds = 0,
  keep.data = TRUE,
  verbose = FALSE,
  class.stratify.cv = NULL,
  n.cores = NULL
)
```

파라미터명	설명
shrinkage	• 학습률 • 범위 : 0~1
distribution	• 훈련에 사용할 분포
n.trees	• tree의 수 • 크면 과적합될 수 있음
interaction.depth	• 각 tree의 최대 깊이 • 대체적으로 1이 잘 작동함
n.minobsinnode	• 터미널 노드의 최소 관측값 개수
cv.folds	• cross-validation fold 개수
train.fraction	• train과 valid 데이터셋의 비율 • 1이면 valid nan으로 나타남
...	

3.1 Gradient Boosting Machine

Parameter tunning

Training				
2002	2003	2004	2005	2006

n.trees <- c(3,5,7)

interaction.depth <- c(1,3,5)

shrinkage <- c(0.1, 0.05, 0.01)

총 27가지 파라미터 조합

```
for (j in 1:nrow(hyper_grid)) {
  set.seed(123)
  m_gbm_untuned <- gbm(
    formula = ret~.,
    distribution = "gaussian",
    data = dm1_train,
    var.monotone = NULL,
    n.trees = hyper_grid$n.trees[j],
    interaction.depth = hyper_grid$interaction.depth[j],
    shrinkage = hyper_grid$shrinkage[j],
    n.minobsinnode = 10,
    train.fraction = 0.7,
    cv.folds = 5,
    keep.data = TRUE,
    verbose = TRUE,
    n.cores = NULL
  )

  gbm_min_mse[j] <-  which.min(m_gbm_untuned$cv.error)

  cat(j, "\n")
}
```

MSE 가 가장 작은 조합을 PICK

3.1 Gradient Boosting Machine

SR

✓ 10개의 그룹으로 나눈 경우

Avg	Std	SR
2.72	10.17	0.93

✓ 20개의 그룹으로 나눈 경우

Avg	Std	SR
0.86	7.8	0.38

소요 시간

2.881789 mins

* 튜닝 5.752766 mins

10개 그룹으로 나눈 경우 SR 값이 0.93로 기준점 0.45보다 크므로 **유의미한 결과**이다.

두 경우 비교했을 때, GBM의 경우 **10개 그룹**으로 나눈 것이 더 효과적이라고 할 수 있다.

KNN Regression과 SR이 거의 비슷하지만, 시간은 KNN이 약 5초로 훨씬 빠르편이다.

3.2 eXtreme Gradient Boosting

- eXtreme Gradient Boosting (XGB)은 회귀분석 또는 분류 분석을 수행할 수 있는 **부스팅 계열**의 예측모형이다.
- GBM을 속도와 성능면에서 향상시킨 알고리즘이다.
- XGB은 GBM 모델들의 실행에 비해 **빠르다.** (병렬처리와 최적화를 장점으로 내세우는 gradient boosting 알고리즘)
- GBM보다 성능이 더 좋은 편이다. (**Regularization** 항이 추가되어 **과적합을 방지**해준다.)
- Early stopping(조기 종료) 기능이 있다.

XGB parameter

파라미터명	설명
booster	• 어떤 부스터 구조를 쓸지 결정 • 의사결정기반모형(gbtree), 선형모형(gblinear), dart
nthread	• XGBoost 실행하는데 사용되는 병렬 스레드 수
eta	• 학습률 • 높을수록 과적합 되기 쉽다. • 주로 0.01~0.3
nrounds	• 생성할 weak learner의 수
max_depth	• 트리의 최대 깊이 • 보토 3-10 사이 값이 적용된다. • 높을수록 복잡도 커져 과적합 되기 쉽다.
min_child_weight	• 관측치에 대한 가중치 합의 최고 • 높을수록 과적합 방지된다.
gamma	• 리프 노드의 추가분할을 결정할 최소손실 감소값 • 높을수록 과적합 방지
subsample	• weak learner가 학습에 사용하는 데이터 샘플링 비율 • 보통 0.5~1 사용 • 낮을수록 과적합 방지
colsample_bytree	• 각 tree별 사용된 feature의 퍼센테이지 • 보통 0.5~1 사용된다. • 낮을수록 과적합 방지
lambda	• 가중치에 대한 L2 Regularization 적용 값
Alpha	• 가중치에 대한 L1 Regularization
...	

3.2 eXtreme Gradient Boosting

Parameter tunning

Training				
2002	2003	2004	2005	2006

max_depth <- c(3,4,5)

eta <- c(0.01,0.05,0.1)

booster <- c('gbtree','gblinear','dart')

colsample_bytree <- c(0.5,0.7,1)

총 81가지 파라미터 조합

```
for (j in 1:nrow(hyper_grid)) {
  set.seed(123)
  m_gbm_untuned <- gbm(
    formula = ret~.,
    distribution = "gaussian",
    data = dm1_train,
    var.monotone = NULL,
    n.trees = hyper_grid$n.trees[j],
    interaction.depth = hyper_grid$interaction.depth[j],
    shrinkage = hyper_grid$shrinkage[j],
    n.minobsinnode = 10,
    train.fraction = 0.7,
    cv.folds = 5,
    keep.data = TRUE,
    verbose = TRUE,
    n.cores = NULL
  )

  gbm_min_mse[j] <-  which.min(m_gbm_untuned$cv.error)

  cat(j, "\n")
}
```

MSE 가 가장 작은 조합을 PICK

3.2 eXtreme Gradient Boosting

SR

✓ 10개의 그룹으로 나눈 경우

Avg	Std	SR
37.25	8.79	14.68

✓ 20개의 그룹으로 나눈 경우

Avg	Std	SR
51.58	13.45	13.28

소요 시간

1.176934 mins

* 튜닝 10.38929 mins

10,20개로 나눈 경우 모두 앞 선 모델들보다 **SR이 압도적으로 높다.**

GBM(2분)보다 소요 시간이 1분 가량 더 적게 걸린다.

GBM을 **성능과 속도면**에서 압도적으로 향상시킨 모델임을 한번 더 확인할 수 있다.

예측 모델 비교

종류	SR	수행시간
Linear Mean Regression	0.57	12.81 secs
KNN Regression	0.98	5.49 secs
Linear Median Regression	0.58	3.17 mins
Quantile Regression Forest	0.69	28.03 mins
Bagging	0.04	8.66 mins
Random Forest	0.54	22.95 mins
Gradieent Boosting	0.93	2.88 mins
eXtreme Gradient Boosting	14.68	1.17 mins

최종적으로 eXtreme Gradient Boosting (XGB)선택!