Midterm

20223888 이선우

1. Data 설명

특성변수 (Features Variables) :

본 데이터셋 2007년 1월 2일 ~ 2020년 10월 16일 까지의 삼성전자의 일별 주가데이터를 통해 생성한 기술적 지표 변수 (22개의 특성 변수)

표적변수 (Target Variable) :

해당 시점부터 5 영업일 이후 시점까지의 종가의 상승여부 (+1 or 0) 로 이진변수

해당 데이터를 Training과 Test셋으로 나누어 준다.

```
data = pd.read_csv('./Data/RAWDATA.csv', index_col=0)

X_train = pd.concat([data[data.columns[:-1]][0:1700], data[data.columns[:-1]][2000:3000]])
y_train = pd.concat([data['Yc'][0:1700], data['Yc'][2000:3000]])
X_test = pd.concat([data[data.columns[:-1]][1700:2000], data[data.columns[:-1]][3000:]])
y_test = pd.concat([data['Yc'][1700:2000], data['Yc'][3000:]])
```

해당 표적변수는 Balance Data로 Undersampling이나, Oversampling 등의 기법을 적용하지 않음.

```
y_train.value_counts(), y_test.value_counts()
>>> (1.0    1418
     0.0    1282
     Name: Yc, dtype: int64,
     1.0    346
     0.0    298
     Name: Yc, dtype: int64)
```

활용 성능 지표 : Recall & Accuracy

동일 기간에 대해 일간 상승 하락 횟수가 비슷하나, 가격이 우상향한 삼성전자 주가의 특성을 참고해, Positive (상승)을 Postitive로 예측하는 것이 중요하다 판단. >> 'Recall' Score를 기준으로 Parameter 튜닝을 진행.

이후 Parameter를 적용한 모델을 평가시 일반화 성능을 평가하는 것이므로 Accuracy를 기준으로 최종 모델을 선택. Balance Data 이므로 F1 score 대신, Accuracy로 평가하는 것이 보다 일반화 성능이 좋을 것이라 판단.

2. 학습 활용 모델들

아래의 Classification 기법을 활용하며 Tuning을 통해 찾은 Best Parameter들로 Fitting 결과를 비교

1. **Logistic Regression (Default 임계값인 0.5를 활용해 Fitting을 진행한다.)**
2. **DecisionTreeClassifier**
3. **Bagging** (Weak 분류기가 Base Estimator(Stump Tree)가 되도록 Tuning 범위 설정)
4. **RandomForest**
5. **Adaboost** (Weak 분류기가 Base Estimator(Stump Tree)가 되도록 Tuning 범위 설정)
6. **Ensemble Voting**

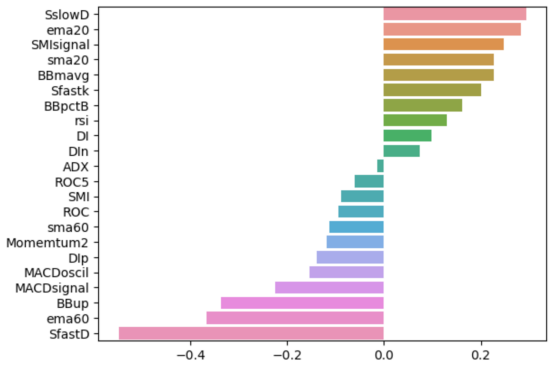
   1 ~ 5 까지의 Tuning된 모델을 베이스로 (2개 : models, 3개 : 10 models, 4개 : 5 models, 5개 : 1 model) 값을 비교한다.

튜닝시, random_state 설정을 주어 올바른 튜닝이 진행될 수 있도록 함

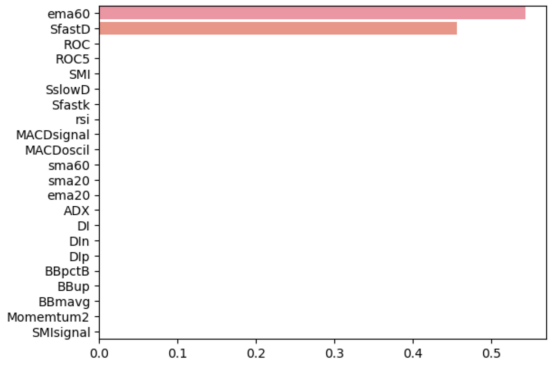
아래 표는 각 방법을 활용해 얻은 Best parameter로 튜닝한 결과이다.

Model	Tuning (Recall Base)	Tuning Result (Best Parameters)	Fitting Result
Logistic Regression	진행하지 않음 (Threshold : 0.5 활용)	'Threshold': 0.5	Accuracy : 0.52950
Decision Tree	'max_depth' : [1,2,3,4,5,6], 'min_samples_split' : [1,2,3,4,5,6,7,8,9,10], 'min_samples_leaf' : [50, 100, ~ , 1000] 'max_leaf_nodes' : [1, 2, ~ 20]	'max_depth': 2, 'max_leaf_nodes': 3, 'min_samples_leaf': 50, 'min_samples_split': 2	Accuracy : 0.53727
Bagging	'base_estimator__max_depth': [1, 2, 3], 'base_estimator__min_samples_split' : [1, 2], 'base_estimator__min_samples_leaf' : [10, 30, 50], 'base_estimator__max_leaf_nodes' : [1, 2], 'n_estimators' : [10, 11, 12, 13, ~ 30], 'max_samples' : [0.2, 0.4, 0.6, 0.8]	'base_estimator__max_depth': 1, 'base_estimator__max_leaf_nodes': 2, 'base_estimator__min_samples_leaf': 50, 'base_estimator__min_samples_split': 2, 'max_samples': 0.6, 'n_estimators': 23	Accuracy : 0.54037
Random Forest	'n_estimators': [50, 100, 150], 'max_features': [1, 2, 3, 4, 5], 'max_depth': [1, 2, 3, 4, 5], 'min_samples_split': [1, 2, 3, 4, 5], 'min_samples_leaf': [1, 2, 3, 4, 5],	'max_depth': 1, 'max_features': 1, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50	Accuracy : 0.53261
Adaboost	'base_estimator__max_depth': [1, 2, 3], 'base_estimator__min_samples_split' : [1, 2], 'base_estimator__min_samples_leaf' : [10, 30, 50], 'base_estimator__max_leaf_nodes' : [1, 2], 'n_estimators': [10,11,12, ~ 50], 'n_estimators': [50, 100, 150, ~ 300]	'base_estimator__max_depth': 1, 'base_estimator__max_leaf_nodes': 2, 'base_estimator__min_samples_leaf': 50, 'base_estimator__min_samples_split': 2, 'n_estimators': 12	Accuracy : 0.47671
Ensemble Voting	'combination' = [${}_5C_2$, ${}_5C_3$, ${}_5C_4$, ${}_5C_5$] 'voting' = ['soft', 'hard']	combination : ['lgr', 'bag'], voting = hard	**Accuracy : 0.54348**

Logistic Regression의 계수값들 (plot 1) / DecisionTree 의 변수 중요도 (plot 2)

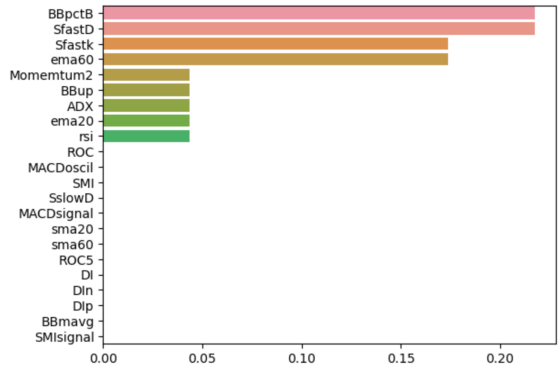


plot 1

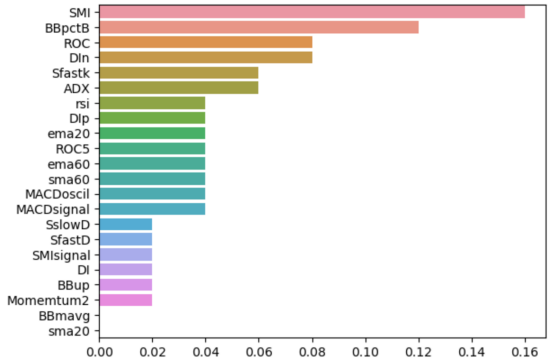


plot 2

Bagging의 변수 중요도 (plot 3) / Random Forest의 변수 중요도 (plot 4)

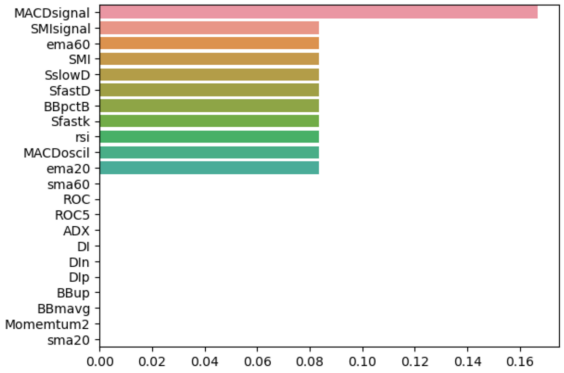


plot 3

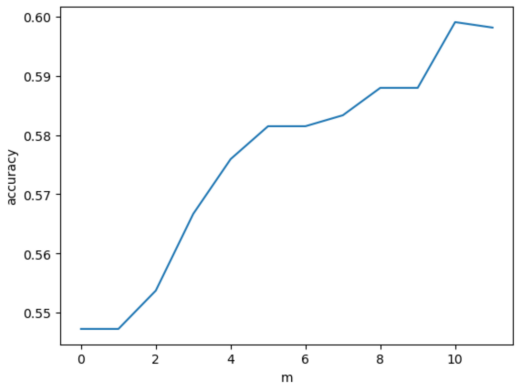


plot 4

Adaboost의 변수 중요도 (plot 5) / M개의 분류기에 따른 정확도 변화 (plot 6)



plot 5



plot 6

>> Adaboost에서 정확도가 과적합이 발생하고 있음을 확인할 수 있음 > 더 약한 학습기와의 결합이 좋아보임

3. 분석 결과

최종 활용 Model은 Accuracy가 높은 Ensemble Voting 기법으로 보임

위 모델은 **Logistic Regression** 과 **Bagging** 모델을 **Hard voting** 방식으로 합성한 것.

```
models = [
    ('lgr', LogisticRegression()),

    ('bag', BaggingClassifier(DecisionTreeClassifier(max_depth=1,\
                                                     max_leaf_nodes=2,\
                                                     min_samples_leaf=50,\
                                                     min_samples_split=2,\
                                                     random_state=1),\
                              max_samples=0.6, n_estimators=23, bootstrap=True, oob_score=True, random_state=3))]
vc = VotingClassifier(models, voting='hard', n_jobs=-1)
vc.fit(X_train_sc, y_train)
Accuracy_score(y_test, vc.predict(X_test_sc)))
```

>> Accuracy Score : 0.54348