주가예측 분류 모형 by 머신러닝

1. 주제선정 이유

지난 코로나 시기 카카오는 코로나로 인한 폭등장에서 우리나라 대표적 기술주로서 시가총액이 조 단위인 기업임에도 불구하고 1년 정도의 시간에도 불구하고 800%의 성장을 보여주었습니다. 하지만 2021년 6월 21일 최고가 이후 줄곧 하락세를 면치 못하고 현재 50000원 언저리로 주가가 떨어지며 많은 개인투자자들에게 손실을 안겨주었습니다. 급격한 상승과 하락이 동반된 카카오 주가의 변화는 매우 좋은 데이터라 생각하였고, 따라서 2021년 초부터 2022년 말까지의 데이터를 이용하여 주가 분류 모형을 만들기로 결정하였습니다.

2. 변수 설명

Date	Open	High	Low	Close	Volume	Change
2021-01-04	78680	79583	77375	79483	853740	0.016680
2021-01-05	78880	79381	77676	78881	748559	-0.007574
2021-01-06	79282	82291	79181	79383	1435715	0.006364
2021-01-07	80687	80987	78880	80788	775191	0.017699
2021-01-08	82896	87512	81189	87111	2466715	0.078267
...	...	...	...	...	...	...
2022-05-25	81000	82000	80300	81800	1365261	0.014888
2022-05-26	81900	83300	81000	81500	1338305	-0.003667
2022-05-27	82400	83000	81500	81700	1038755	0.002454
2022-05-30	82800	84000	82600	83900	2091429	0.026928
2022-05-31	84100	85000	82900	85000	2170810	0.013111

앞서 이야기한 시점의 카카오 데이터를 API를 이용해 KRX에서 불러왔습니다. 불러온 데이터에는 시가, 고가, 저가, 종가, 거래량, 증감율에 관한 데이터가 존재하고, 이로는 분석에 부족하다는 판단을 내려 여러 시계열 변수를 추가하였습니다.

1) RSI

$$SI \quad 100 - [\frac{100}{1 - RS}]$$

상대강도지수 RSI는 일정 기간 동안 주가 변동폭 사이에서 상승 압력과 하락 압력을 서로 비교함으로써 주가의 움직임 강도를 측정하여 백분율로 나타낸 지표를 뜻합니다. 추세 전환 시점을 예측하기 좋은 신뢰성 지표로 자주 사용됩니다.

2) MACD

$$\mathrm{MACD}_t = \mathrm{EMA}^{12}_{S,t} - \mathrm{EMA}^{26}_{S,t}$$

$\mathrm{EMA}^{d}_{S,t}$: t 시점에서 S의 d일간의 지수평활

장기 이동평균과 단기 이동평균을 비교하는 모멘텀 지수로, 26일 지수이동평균과 12일 지수이동평균의 차이로 정의됩니다 (Appel, 2005). Chong과 Ng (2008), Wang과 Kim (2018) 등의 논문에서 주식 가격 분석 및 예측 등에 사용되었습니다.

3) STOCH RSI

$$StochRSI = \frac{RSI - min(RSI)}{max(RSI) - min(RSI)}$$

Stoch RSI는 스토캐스틱 RSI라고 불리며 Tuhsard chande와 Stanley kroll에 의해 개발된 지표입니다. 일정기간의 RSI를 측정할 수 있는 오실레이터로 스토캐스틱 공식에 RSI를 대입하여 계산합니다. StochRSI는 가격의 2차 파생물(가격을 이용해 구한 RSI는 1차, RSI을 이용해 구한 StochRSI는 2차)이기 때문에 때때로 시장 가격과 일치하지 않을 수도 있습니다. 또한 RSI보다 훨씬 빨리 움직이는 지표입니다.

4) ADL

$$ADL(K) = \sum_{n=0}^{k-1} [\, UP(k-n) - DN(k-n) \,]$$

ADL (등락주선)은 일정 기준일 이후부터 전일의 종가에 비해 오른 종목 수에서 내린 종목 수를 뺀 것을 매일마다 누계해서 그것을 선으로 이어 작성한 것으로 주가의 선행지표로 분석하곤 합니다.

5) ATR

$$\mathrm{ATR} = \frac{\mathrm{ATR}_{1} * (N-1) + \mathrm{TR}}{N}$$

ATR(Average True Range)은 주가의 변동성을 측정하는데 사용되는 기술적 지표이다. 특정 기간 동안 실제 가격이 움직인 변동폭을 측정하여 평균화한 지표로, 현재 시장의 리스크의 정도를 나타낸다.

6) MFI

MFI (Money Flow Index)는 주식(혹은 증권)의 가격과 거래량을 사용하여 주식이 과잉 매도인지 과잉 매수인지를 식별하기 위한 기술적 지표입니다. 이는 증권의 가격의 모멘텀이 변경된다는 경고 신호로 사용되며, 이는 RSI와 동일하게 0에서 100까지의 값을 가질 수 있습니다. 상대적강도지수, RSI와는 달리, MFI는 증권의 가격과 거래량 데이터를 모두 통합하여 사용하기 때문에 기술적 분석가들은 MFI를 거래량이 가중된 RSI라고 부르기도 합니다.

7) ROC

ROC(Rate of change, Price Rate of change)는 과거 일정 시점 가격에 대비해 현재가격의 변화율을 백분율로 나타낸 것입니다. 5일 전 시점을 기준으로 ROC를 나타냈습니다.

8) OBV

$$\mathrm{OBV}_t = \begin{cases} 0, & \text{if } t = 0 \\ \mathrm{OBV}_{t-1} + \mathrm{Volume}_{t-1}, & \text{if } S_t > S_{t-1} \\ \mathrm{OBV}_{t-1} - \mathrm{Volume}_{t-1}, & \text{if } S_t < S_{t-1} \\ \mathrm{OBV}_{t-1}, & \text{if } S_t = S_{t-1} \end{cases}$$

Volume_t : t 시점의 거래량

거래량의 변동을 이용해 주식 가격을 추정하는 지표로, 누적 거래량으로 개별 주식의 매수와 매도 경향을 지수화한 것입니다. 상승할 때 거래량을 더하고 하락할 때 거래량을 빼는 방식을 이용합니다.

9) CCI

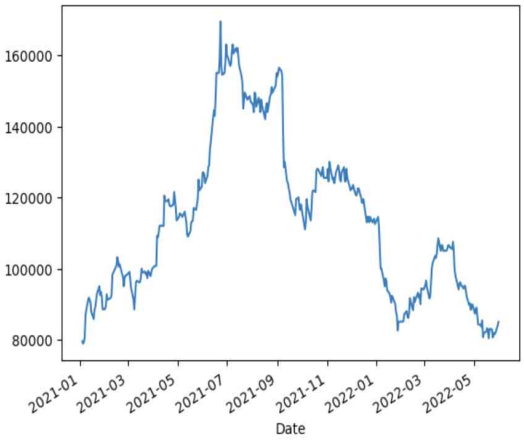
$$CI \quad M - \frac{m}{d} \times 0.15$$

추세채널지수 CCI는 추세의 시작과 끝을 포착하기 위해서 개발이 된 것으로 현재 주가와 이동평균과의 이격을 이용해서 개발한 보조지표입니다. 현재의 가격이 이동평균과 얼마나 떨어져 있는지를 나타냅니다.

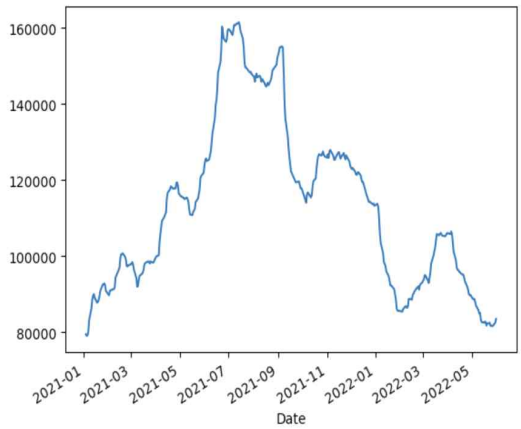
3. 데이터 전처리

1) 지수평활

데이터의 시가, 고가, 저가, 종가, 거래량, 증감률 변수에서 종가(Close) 변수의 경우 위의 데이터는 대략적이며 시계열에 대한 스파이크가 많이 포함되어 있음을 알 수 있습니다. 다음과 같이 종가 데이터는 매우 부드럽지 않으며 모형이 추세를 추출하기 어려울 수 있습니다. 이를 줄이기 위해 기술 지표를 계산하기 전에 데이터를 지수 평활하였습니다. 고점과 저점이 많으면 근사하기 어렵거나 기술적 지표를 계산할 때 추세를 추출하기 어려울 수 있기 때문입니다.



<지수 평활 전 Close 데이터>



<지수 평활 후 Close 데이터>

2) 거래량(Volume) 정규화 & EMA 변수 추가

볼륨은 가격 변동과 상관 관계가 있으므로 정규화했습니다. 또한, 지수 이동 평균 (EMA)을 추가하였는데 이는 가장 최근의 데이터 포인트에 더 큰 가중치와 중요성을 부여하는 이동 평균 (MA) 유형입니다. 지수 이동 평균은 지수 가중치 이동 평균 이라고도 합니다. 지수 가중 이동 평균은 해당 기간의 모든 관측치에 동일한 가중치를 적용하는 단순 이동 평균 (SMA) 보다 최근 가격 변동에 더 크게 반응합니다. EMA 는 시차의 부정적인 영향을 어느 정도 완화하는 역할을 하기에 추가하였습니다.

3) NASDAQ 증감 변수 추가

나스닥 변수의 일별 증감률을 계산하여 변수로 추가하였습니다.

4) Target 변수 생성

1거래일 후와 같이 너무 짧은 미래의 예측은 백색소음의 영향을 받기 쉽기 때문에 이 영

```
# 지수생성

def _get_indicator(data):

    data['ema50'] = data['Close'] / data['Close'].ewm(50).mean()
    data['ema20'] = data['Close'] / data['Close'].ewm(20).mean()
    data['ema15'] = data['Close'] / data['Close'].ewm(15).mean()
    data['ema5'] = data['Close'] / data['Close'].ewm(5).mean()

    data['normVol'] = data['volume'] / data['volume'].ewm(5).mean()
    del(data['volume'])
    return data


df = _get_indicator(df1)
print(df.columns)
```

```
Index(['Open', 'High', 'Low', 'Close', 'Change', 'RSI', 'MACD', 'STOCH', 'ADL',
       'ATR', 'MFI', 'ROC', 'OBV', 'CCI', 'ema50', 'ema20', 'ema15', 'ema5',
       'normVol', 'Nasdaq'],
      dtype='object')
```

향을 줄여 예측 가능성을 높이기 위해 5거래일 후인 t + 5 시점의 등락 예측을 목표로 하였습니다. 이에 맞춰 다음과 같이 Target 변수를 생성하였습니다.

```
def _produce_pred(data, num):

    prediction = (data.shift(-num)['Close'] > data['Close'])
    prediction = prediction.iloc[:-num]
    data['Target'] = prediction.astype(int)

    return data

df = _produce_pred(df, num = 5)
```

5) 결측치 제거

마지막으로 변수를 생성하며 시차의 차이에 의한 시작 시점의 데이터와 끝부분의 데이터에 발생한 결측치를 판다스의 dropna()를 이용하여 결측치를 제거하였습니다.

4. 모델 적합

앞서 만들어진 데이터를 Target 변수와 Feature 변수로 분리한 후, 4:1 비율로 Train 과 Test 데이터로 분리하였습니다. 이 때, 시계열 데이터라는 점을 고려하여 shuffle을 False 로 설정하였습니다. 이후 데이터를 모델에 적합시 최적의 하이퍼 파라미터를 찾기 위해 GridsearchCV 를 적용하였고, 시계열이라는 데이터 특성을 고려해 cv를 TimeSeriesSplit을 이용해 설정하였습니다. 예측 시차를 5로 설정하였기에 gap = 4 로 두어 훈련데이터 세트로 인한 과적합을 방지하였습니다.

모델은 RandomForest 와 KNN, XGBoost, LightGBM을 이용하였습니다. 그리고 마지막으로 보팅 방식의 앙상블 방법을 이용하여 모델을 학습 시켜봤습니다. 보팅의 경우 Test accuracy 가 높은 RandomForest 와 XGBoost, LightGBM 세 모델을 이용하였습니다. 우선 분석 결과는 다음과 같습니다.

```
X = df.drop(columns='Target')
y = df['Target']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffle = False)
```

```
tscv = TimeSeriesSplit(gap = 4, n_splits=5)
```

Model	Train accuracy	Best parameter	Test accuracy
RandomForest	0.77	'max_depth': 2	0.68
KNN	0.74	'n_neighbors': 18	0.48
XGBoost	0.93	'max_depth': 1	0.65
LightGBM	0.85	'max_depth': 1	0.76
Voting			0.61

아래와 같이 각 모델 별로 train 함수를 만들어 test accuracy 와 confusion matrix를 결과로 나올 수 있도록 하였습니다.

```
## 랜덤포레스트 모델

def _train_random_forest(X_train, y_train, X_test, y_test):


    # Create a new random forest classifier
    rf = RandomForestClassifier(random_state = 42)

    # Dictionary of all values we want to test for n_estimators
    params_rf = {'max_depth': np.arange(1, 20)}

    # Use gridsearch to test all values for n_estimators
    rf_gs = GridSearchCV(rf, params_rf, cv=tscv, scoring="accuracy")

    # Fit model to training data
    rf_gs.fit(X_train, y_train)

    # Save best model
    rf_best = rf_gs.best_estimator_

    # Check best n_estimators value
    print(rf_gs.best_params_)
    pred = rf_best.predict(X_test)
    print('예측 정확도: {0:.4f}'.format(accuracy_score(y_test, pred)))
    print(classification_report(y_test, pred))
    print(confusion_matrix(y_test, pred))

    return rf_best

rf_model = _train_random_forest(X_train, y_train, X_test, y_test)
```

```
# LightGBM 모델

def _train_lgb(X_train, y_train, X_test, y_test):

    lgb = LGBMClassifier(random_state = 42)
    params_lgb = {'max_depth':np.arange(1, 20)}
    lgb_gs = GridSearchCV(lgb, params_lgb, cv=tscv, scoring="accuracy")

    lgb_gs.fit(X_train, y_train)

    lgb_best = lgb_gs.best_estimator_

    print(lgb_gs.best_params_)
    print(lgb_gs.best_score_)
    pred = lgb_best.predict(X_test)
    print('예측 정확도: {0:.4f}'.format(accuracy_score(y_test, pred)))
    print(classification_report(y_test, pred))
    print(confusion_matrix(y_test, pred))
    return lgb_best


lgb_model = _train_lgb(X_train, y_train, X_test, y_test)
```

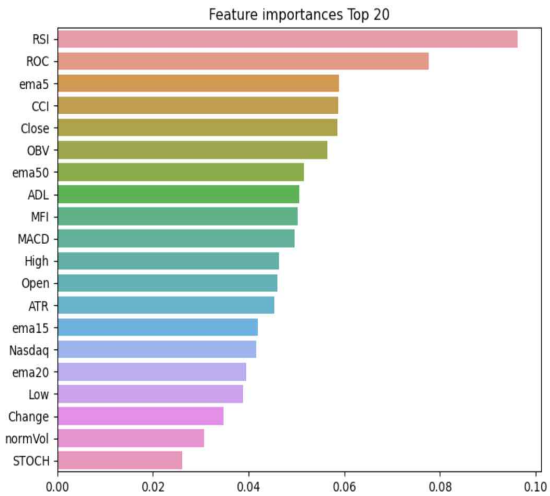
5. 결과 해석

RandomForest는 배깅 방식의 앙상블 기법답게 비교적 빠른 수행속도와 정확도 측면에서 좋은 성과를 낼 수 있다는 장점 그대로 여러 모형 중 test accuracy가 0.68로 준수한 성능을 보이고 있습니다. 다만, 트리 기반의 앙상블이므로 따로 하이퍼 파라미터를 튜닝하여도 크게 모델의 성능이 달라지진 않았습니다.

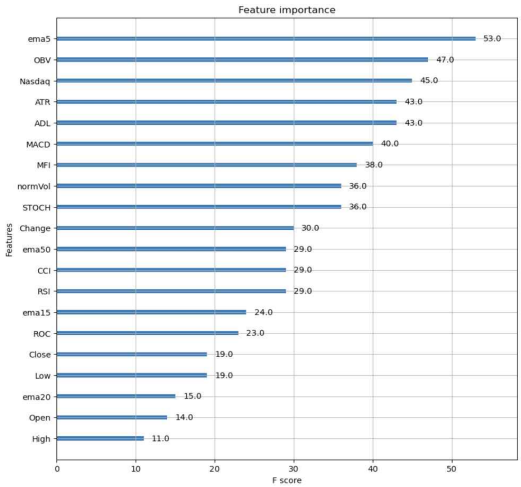
KNN의 경우에는 Train accuracy에 비해 예상 이상으로 낮은 Test accuracy를 보이고 있는데, 이는 Volume 변수의 정규화와 종가(Close) 변수의 지수평활로 인해 정보의 손실이 발생했기 때문으로 보입니다.

XGBoost는 CART 앙상블 모델을 사용하며, 분류에 강한 모델답게 test accuracy가 0.65로 준수한 성능을 보였습니다. 다만, Train accuracy 가 0.95로 매우 높으므로 과적합이 발생한 것으로 의심됩니다.

LightGBM의 경우 XGBoost의 단점을 보완한 모델답게 Test accuracy가 0.76으로 높은 성능을 보이고 있습니다. Train accuracy 도 0.85로 준수하므로 과적합은 발생하지 않은 것으로 보입니다.



<랜덤포레스트 변수 중요도>



<XGBoost 변수중요도>

마지막으로 배깅과 부스팅 방식의 두 앙상블 모델에서 각각 변수 중요도를 살펴봤습니다. 그림과 같이 피처 변수의 개수가 적어 쉽사리 판단할 수 없지만 모든 피처 변수 중 딱히 유의하지 않은 변수는 없는 것으로 보입니다.