머신러닝으로 주식 예측

DSML 데이터과학부 19016015 김유림

PART 01

문제 및 데이터 설명

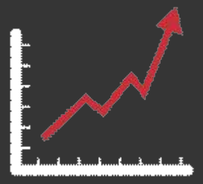
PART 01

데이터 전처리 및 모델링

PART 01

결과

PART 01



결과

PART 01

문제 및 데이터 설명

1. 2018년부터 3년 이상 활동해온 기업 중 거래대금이 특정 금액 이상 발생한 날의 주가 수집
2. 수집한 날을 기준으로 해당일과 이전 9일 동안에 대한 주가를 수집
3. 해당일 다음 날에 대한 종가가 2% 상승 여부에 대한 예측, 정확도 구하기
4. 여러 모델을 이용해 투자 수익률 구하기

01 데이터 설명

Out[13]:

	회사명	종목코드	업종	주요제품	상장일	결산월	대표자명	홈페이지	지역
0	DL	000210	기타 금융업	지주회사	1976-02-02	12월	전병욱	http://www.dlholdings.co.kr	서울특별시
1	DRB동일	004840	고무제품 제조업	고무벨트(V벨트,콘베이어벨트,평벨트),프라스틱제품 제조,판매	1976-05-21	12월	류영식	http://drbworld.com	부산광역시
2	DSR	155660	1차 비철금속 제조업	합섬섬유로프	2013-05-15	12월	홍석빈	http://www.dsr.com	부산광역시
3	GS	078930	기타 금융업	지주회사/부동산 임대	2004-08-05	12월	허태수, 홍순기 (각자 대표이사)	NaN	서울특별시
4	GS글로벌	001250	상품 종합 도매업	수출입업(시멘트,철강금속,전기전자,섬유,기계화학),상품중개,광업,채석업/하수처리 서...	1976-06-26	12월	김태형	http://www.gsgcorp.com	서울특별시

전체 종목 코드에서 2018년 1월 2일 데이터가 존재하는지 확인한 후 3년 이상 존속 종목을 선정

선정한 종목 중 거래대금이 100억 이상인 날짜의 주가

Date	Open	High	Low	Close	Volume	Change
2018-01-02	9890	10000	8800	10000	8210	-0.019608
2018-01-03	9800	9800	9700	9700	800	-0.030000
2018-01-04	9410	9410	9410	9410	200	-0.029897
2018-01-05	9120	10000	9120	10000	140	0.062699
2018-01-08	10000	10000	9850	10000	495	0.000000
...	...	...	...	...	...	...
2020-12-23	4990	4990	4100	4540	272	0.031818
2020-12-24	4595	4595	4005	4515	705	-0.005507
2020-12-28	4985	4985	4065	4640	10	0.027685
2020-12-29	4010	4590	4010	4590	3	-0.010776
2020-12-30	3905	4845	3905	4050	21	-0.117647

PART 01

데이터 전처리 및 모델링

02 데이터 전처리

1. 전체 종목 코드에서 fdr를 이용하면 dataframe으로 불러온다.
2. dataframe로 전처리할 경우 시간이 오래 걸린다 -> list로 변환 후 전처리 수행

```
start_date='20180101'
end_date='20201231'
lst_date_code=[]

for code, name in tqdm(lst_result):
    stock=fdr.DataReader(code, start_date, end_date)
    stock= stock.reset_index().values.tolist()
```

```
16 OF.close()
100%|██████████| 1999/1999 [24:44<00:00,  1.35it/s]
1529
```

3. 데이터의 양이 많으면 list도 오래 걸린다 -> SQL을 이용해 전처리

```
dic_code2date = {}

OF = open('assignment2_sql.txt', 'w', encoding = 'utf-8')
for code in tqdm(dic_code2company.keys()):
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date
                BETWEEN '2018-01-01' AND '2020-12-31'
                '''.format(code)
    stock = pd.read_sql(sql = sql_query, con = db_dsml)
    lst_stock = stock.values.tolist()

```

```
100%|██████████| 1999/1999 [00:29<00:00, 68.26it/s]
```

4. 해당 데이터를 불러온 다음 해당 일을 기준으로 9일 전까지의 총 10일간의 주가를 구한 다음 해당일의 다음 날 종가의 상승 여부를 처리하여 머신러닝 데이터를 만든다.

```
OF = open('assignment3_sql.txt', 'w', encoding = 'utf-8')
for code in tqdm(dic_code2date.keys()):
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date
                BETWEEN '2018-01-01' AND '2020-12-31'
                '''.format(code)
    stock = pd.read_sql(sql = sql_query, con = db_dsml)
    lst_stock = stock.values.tolist()

    for i, row_lst_stock in enumerate(lst_stock):
        # 예외 처리
        if (i < 9) or (i >= len(lst_stock)-1):
            continue
        date = row_lst_stock[0]
        if date not in dic_code2date[code]:
            continue

        # 11days data
        sub_stock = lst_stock[i-9:i+1]
        next_date = lst_stock[i+1][0]
        lst_data = []
        for row_sub_stock in sub_stock:
            open, high, low, close, volume = row_sub_stock[1:6]
            trading_value = close * volume
            lst_data += [open, high, low, close, trading_value]
            del open
        data = ','.join(map(str, lst_data))

        # label
        label = int(lst_stock[i+1][-1] >= 0.02)

        result = '{}₩t{}₩t{}₩t{}₩t{}₩n'.format(code, date.strftime("%Y%m%d"),next_date.strftime("%Y%m%d"), data, label)
        OF.write(result)
OF.close()
```

20180101 ~ 20201231까지는 학습데이터셋

```
OF = open('assignment3-2_sql.txt', 'w', encoding = 'utf-8')
for code in tqdm(dic_code2date.keys()):
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date
                BETWEEN '2021-01-01' AND '2021-06-30'
                '''.format(code)
    stock = pd.read_sql(sql = sql_query, con = db_dsml)
    lst_stock = stock.values.tolist()

    for i, row_lst_stock in enumerate(lst_stock):
        # 예외 처리
        if (i < 9) or (i >= len(lst_stock)-1):
            continue
        date = row_lst_stock[0]
        if date not in dic_code2date[code]:
            continue

        # 11 days data
        sub_stock = lst_stock[i-9:i+1]
        next_date = lst_stock[i+1][0]
        lst_data = []
        for row_sub_stock in sub_stock:
            open, high, low, close, volume = row_sub_stock[1:6]
            trading_value = close * volume
            lst_data += [open, high, low, close, trading_value]
            del open
        data = ','.join(map(str, lst_data))

        # label
        label = int(lst_stock[i+1][-1] >= 0.02)

        result = '{}₩t{}₩t{}₩t{}₩t{}₩n'.format(code, date.strftime("%Y%m%d"),next_date.strftime("%Y%m%d"), data, label)
        OF.write(result)
OF.close()
```

20210101 ~ 20210630까지는 검증데이터셋

02 private 시험데이터셋 만들기

```
OF = open('assignment3-3_sql.txt', 'w', encoding = 'utf-8')
for code in tqdm(dic_code2date.keys()):
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date
                BETWEEN '2021-07-01' AND '2021-12-31'
                '''.format(code)
    stock = pd.read_sql(sql = sql_query, con = db_dsml)
    lst_stock = stock.values.tolist()
    for i, row_lst_stock in enumerate(lst_stock):
        # 예외 처리
        if (i < 9) or (i >= len(lst_stock)-1):
            continue
        date = row_lst_stock[0]
        if date not in dic_code2date[code]:
            continue

        # 11 days data
        sub_stock = lst_stock[i-9:i+1]
        next_date = lst_stock[i+1][0]
        lst_data = []
        for row_sub_stock in sub_stock:
            open, high, low, close, volume = row_sub_stock[1:6]
            trading_value = close * volume
            lst_data += [open, high, low, close, trading_value]
            del open
        data = ','.join(map(str, lst_data))

        # label
        label = int(lst_stock[i+1][-1] >= 0.02)

        result = '{}\t{}\t{}\t{}\t{}\n'.format(code, date.strftime("%Y%m%d"),next_date.strftime("%Y%m%d"), data, label)
        OF.write(result)
OF.close()
```

20210701 ~ 20211231까지는 private시험데이터셋

02 모델링 (학습데이터셋으로 학습한 모델로 검증데이터셋을 예측)

```
IF=open("assignment3_sql.txt",'r')

lst_code_date = []
trainX = []
trainY = []

for line in tqdm(IF):
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date, next_date])
    trainX.append(list(map(int, x.split(","))))
    trainY.append(int(y))

trainX=np.array(trainX)
trainY=np.array(trainY)

clf = XGBClassifier(n_estimators=300, nthread=1)
clf.fit(trainX, trainY)

with open('model_xgb.pickle', 'wb') as f:
    pickle.dump(clf, f)
```

다양한 모델로 학습하기

```
import numpy as np
import sklearn.metrics as metrics
import pickle
from sklearn.linear_model import LogisticRegression

IF=open("assignment3-2_sql.txt",'r')
lst_code_date=[]
testX=[]
testY=[]

for line in IF:
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date,next_date])
    testX.append(list(map(int, x.split(","))))
    testY.append(int(y))
testX=np.array(testX)
testY=np.array(testY)

with open('model_xgb.pickle', 'rb') as f:
    clf = pickle.load(f)

predY = clf.predict_proba(testX) # predict_proba 함수는 예측한 값
predY2 = clf.predict(testX)  # predict 함수는 예측한 값을 이진 값
```

학습을 시키기 위해 train데이터과 test데이터를 만든다.

02 모델링 (학습데이터셋으로 학습한 모델로 검증데이터셋을 예측)


```python
from sklearn.metrics import accuracy_score
name=['Logistic regression','Decision tree','Support vector machine','Gaussian naive bayes','K nearest neighbor',
      'Random forest', 'Gradient boosing','Neural network','XGBClassifier']
result =[]
# 1. Logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)
result.append(accuracy_score(testY, y_pred_test))

print("Logistic Regression:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))

# 2. Decision tree
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)
result.append(accuracy_score(testY, y_pred_test))
print("Decision tree:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))

# 3. Support vector machine
from sklearn.svm import SVC
model = SVC()
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)
result.append(accuracy_score(testY, y_pred_test))
print("Support vector machine:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))

# 4. Gaussian naive bayes
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)
result.append(accuracy_score(testY, y_pred_test))
print("Gaussian naive bayes:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))

# 5. K nearest neighbor
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)
result.append(accuracy_score(testY, y_pred_test))
print("K nearest neighbor:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))
```

```python
# 6. Random forest
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)
result.append(accuracy_score(testY, y_pred_test))
print("Random forest:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))

# 7. Gradient boosing
from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier()
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)

result.append(accuracy_score(testY, y_pred_test))
print("Gradient boosing:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))

# 8. Neural network
from sklearn.neural_network import MLPClassifier
model = MLPClassifier(max_iter=1000)
model.fit(trainX, trainY)
y_pred_train = model.predict(trainX)
y_pred_test = model.predict(testX)
result.append(accuracy_score(testY, y_pred_test))
print("Neural network:", accuracy_score(trainY, y_pred_train), accuracy_score(testY, y_pred_test))
```

```
Logistic Regression: 0.7582152974504249 0.7730123406837953
Decision tree: 1.0 0.6233056848067975
Support vector machine: 0.7591359773371105 0.7730123406837953
Gaussian naive bayes: 0.35637393767705383 0.33724458830669635
K nearest neighbor: 0.7864022662889518 0.7210196237102974
Random forest: 1.0 0.7693708274327331
Gradient boosing: 0.7623937677053825 0.7711915840582642
Neural network: 0.648158640226629 0.630386405017196
```

02 모델링 (학습데이터셋으로 학습한 모델로 private시험데이터셋을 예측)

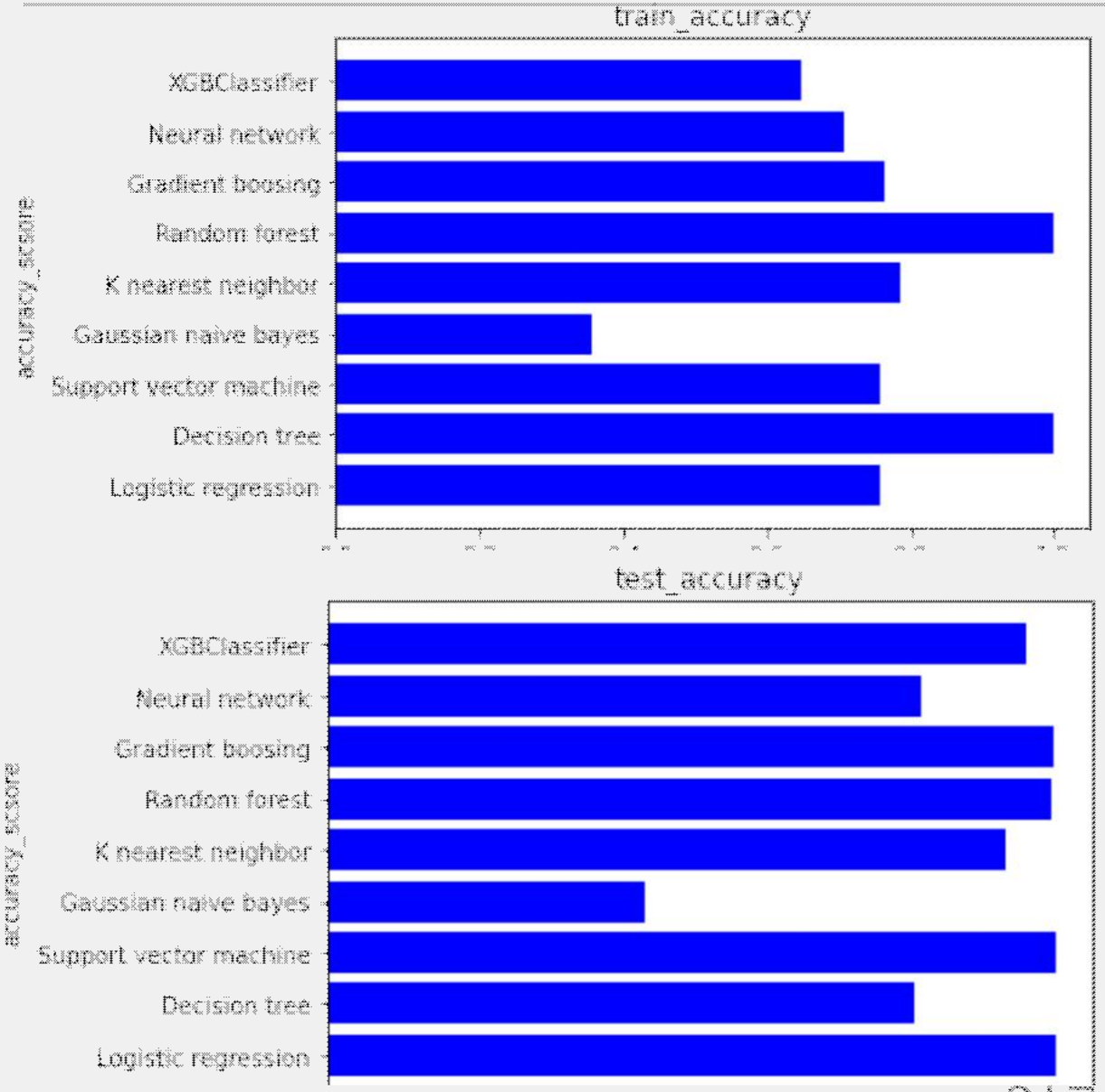
```
IF=open("assignment3_sql.txt",'r')

lst_code_date = []
trainX = []
trainY = []

for line in tqdm(IF):
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date, next_date])
    trainX.append(list(map(int, x.split(","))))
    trainY.append(int(y))

trainX=np.array(trainX)
trainY=np.array(trainY)

clf = XGBClassifier(n_estimators=300, nthread=1)
clf.fit(trainX, trainY)

with open('model_xgb.pickle', 'wb') as f:
    pickle.dump(clf, f)
```

```
IF=open("assignment3-3_sql.txt",'r')
lst_code_date=[]
private_testX=[]
private_testY=[]

for line in IF:
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date, next_date])
    private_testX.append(list(map(int, x.split(","))))
    private_testY.append(int(y))
private_testX=np.array(private_testX)
private_testY=np.array(private_testY)


with open('model_xgb.pickle', 'rb') as f:
    clf = pickle.load(f)

private_predY = clf.predict_proba(private_testX)
private_predY2 = clf.predict(private_testX)  # predict 함수는
```

학습을 시키기 위해 train데이터과 test데이터를 만든다.

PART 01

결과



여러 학습에 대한 train, test 정확도이다.

03 결과 (시계열 LSTM)

```python
from tqdm.notebook import tqdm

from tensorflow import keras

model = keras.models.Sequential()
model.add(keras.layers.InputLayer(input_shape=(trainX.shape[1],trainX.shape[2])))
model.add(keras.layers.LSTM(128, activation='hard_sigmoid', return_sequences=True))
model.add(keras.layers.LSTM(64, activation='hard_sigmoid', return_sequences=True))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.LSTM(64, activation='hard_sigmoid', return_sequences=True))
model.add(keras.layers.LSTM(32, activation='hard_sigmoid', return_sequences=False))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Dense(1, activation='sigmoid'))

model.compile(optimizer=keras.optimizers.Adam(
    learning_rate=keras.optimizers.schedules.ExponentialDecay(0.01,decay_steps=100000,decay_rate=0.96)),
    loss="mean_squared_error",
    metrics=['accuracy'])
model.summary()
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU k
ernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU
kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU
kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU
kernel as fallback when running on GPU.
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 10, 128)           68608

lstm_1 (LSTM)                (None, 10, 64)            49408

dropout (Dropout)            (None, 10, 64)            0

lstm_2 (LSTM)                (None, 10, 64)            33024

lstm_3 (LSTM)                (None, 32)                12416

dropout_1 (Dropout)          (None, 32)                0

dense (Dense)                (None, 1)                 33

=================================================================
Total params: 163,489
Trainable params: 163,489
Non-trainable params: 0
_________________________________________________________________
```

```
[[0.24792475]
 [0.2479044 ]
 [0.2479044 ]
 ...
 [0.2479044 ]
 [0.2479044 ]
 [0.2479044 ]]
```

```python
start_money = 10000000 # 초기 현금 1천만원
money = start_money
dic_code2num ={}  # 보유 종목


IF=open("trading2022firsthalf.txt",'r')
for i, line in tqdm(enumerate(IF)): #주문 일지를 한 줄 읽어 옴
    code, date, request, amount = line.strip().split("₩t")
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date = {}
                '''.format(code, date)
    lst_stock = pd.read_sql(sql = sql_query, con = db_dsml).values.tolist()
    for row in lst_stock:
#         if date in row[0].strftime('%Y%m%d'):
        close = row[4]

    if request == 'buy': # buy인 경우
        if amount.startswith('r'):
            request_money = money * float(amount.lstrip("r")) / 100
        elif amount == 'all':
            request_money = money
        elif amount.isdigit():
            request_money = int(amount)
        # elif amount == ~~~~~    ##### 기타 필요한 매수 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        request_money = min(request_money, money)
        buy_num = int(request_money / close)
        money -= buy_num * close  # 현재 금액(money)를 실제 매수액을 뺀 만큼 업데이트
        if code not in dic_code2num:
            dic_code2num[code] = 0
        dic_code2num[code] += buy_num # 보유 종목 데이터에 구매 종목(code)를 매수 개수 만큼 증가
    if request == 'sell': # sell인 경우
        if amount == 'all':
            sell_num = dic_code2num[code]
        # elif amount == ~~~~~    ##### 기타 필요한 매도 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        money += sell_num * close
        dic_code2num[code] -= sell_num
        if dic_code2num[code] == 0:
            del dic_code2num[code]
IF.close()

if dic_code2num != {}: # 매매가 종료되었는데 보유 종목이 있으면
    raise Exception('Not empty stock')

print("Final earning rate : {} %".format(str((money-start_money) / start_money * 100)))
```

```
Final earning rate : 16.20158 %
```

시계열 LSTM으로 학습한 결과 예측값이 대부분 같은 값으로 나왔다.

이는 학습이 잘 이루어지지 않았음을 알 수 있다.

03 결과 (학습데이터셋으로 학습한 모델로 검증데이터셋을 예측(1000억)


```
IF=open("assignment3_sql.txt",'r')

lst_code_date = []
trainX = []
trainY = []

for line in tqdm(IF):
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date, next_date])
    trainX.append(list(map(int, x.split(","))))
    trainY.append(int(y))

trainX=np.array(trainX)
trainY=np.array(trainY)

clf = XGBClassifier(n_estimators=300, nthread=1)
clf.fit(trainX, trainY)

with open('model_xgb.pickle', 'wb') as f:
    pickle.dump(clf, f)
```

```
IF=open("assignment3-2_sql.txt",'r')
lst_code_date=[]
testX=[]
testY=[]

for line in IF:
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date,next_date])
    testX.append(list(map(int, x.split(","))))
    testY.append(int(y))
testX=np.array(testX)
testY=np.array(testY)

with open('model_xgb.pickle', 'rb') as f:
    clf = pickle.load(f)

predY = clf.predict_proba(testX) # predict_proba 함수는 예측한 값을 확률 값으로 출력
predY2 = clf.predict(testX)  # predict 함수는 예측한 값을 이진 값(1 또는 0)으로 출력
```

```
lst_result=[]
for (code, date, next_date),y in zip(lst_code_date, predY):
    if y[1] >= 0.2:
        lst_result.append([code, date, "buy", "r90"])
        lst_result.append([code, next_date, "sell", "all"])

lst_result.sort(key=lambda x:x[1])

OF=open('trading2022firsthalf.txt','w')
for row in lst_result:
    OF.write('₩t'.join(map(str,row))+'₩n')
OF.close()
```

```
Final earning rate : 38.7378 %
```

```
0.7418571717580417
```

```
start_money = 10000000 # 초기 현금 1천만원
money = start_money
dic_code2num ={}  # 보유 종목


IF=open("trading2022firsthalf.txt",'r')
for i, line in tqdm(enumerate(IF)): #주문 일지를 한 줄 읽어 옴
    code, date, request, amount = line.strip().split("₩t")
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date = {}
                '''.format(code, date)
    lst_stock = pd.read_sql(sql = sql_query, con = db_dsml).values.tolist()
    for row in lst_stock:
#         if date in row[0].strftime('%Y%m%d'):
        close = row[4]

    if request == 'buy': # buy인 경우
        if amount.startswith('r'):
            request_money = money * float(amount.lstrip("r")) / 100
        elif amount == 'all':
            request_money = money
        elif amount.isdigit():
            request_money = int(amount)
        # elif amount == ~~~~~    ##### 기타 필요한 매수 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        request_money = min(request_money, money)
        buy_num = int(request_money / close)
        money -= buy_num * close  # 현재 금액(money)을 실제 매수액을 뺀 만큼 업데이트
        if code not in dic_code2num:
            dic_code2num[code] = 0
        dic_code2num[code] += buy_num # 보유 종목 데이터에 구매 종목(code)를 매수 개수 만큼
    if request == 'sell': # sell인 경우
        if amount == 'all':
            sell_num = dic_code2num[code]
        # elif amount == ~~~~~    ##### 기타 필요한 매도 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        money += sell_num * close
        dic_code2num[code] -= sell_num
        if dic_code2num[code] == 0:
            del dic_code2num[code]
IF.close()

if dic_code2num != {}: # 매매가 종료되었는데 보유 종목이 있으면
    raise Exception('Not empty stock')

print("Final earning rate : {} %".format(str((money-start_money) / start_money * 100)))
```

XGBClassifier의 모델로 1,000억 이상인 데이터로 학습한 결과

정확도는 74%이며, 수익률은 약 39%가 나왔다.

하지만 데이터가 806개로 적은 양이다. 그러므로 과적합이 일어날 수 있다.

그래서 100억 이상인 데이터로 학습했다.

03 결과 (학습데이터셋으로 학습한 모델로 검증데이터셋을 예측(100억)


```
IF=open("assignment3_sql.txt",'r')

lst_code_date = []
trainX = []
trainY = []

for line in tqdm(IF):
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date, next_date])
    trainX.append(list(map(int, x.split(","))))
    trainY.append(int(y))

trainX=np.array(trainX)
trainY=np.array(trainY)

clf = XGBClassifier(n_estimators=300, nthread=1)
clf.fit(trainX, trainY)

with open('model_xgb.pickle', 'wb') as f:
    pickle.dump(clf, f)
```

```
IF=open("assignment3-2_sql.txt",'r')
lst_code_date=[]
testX=[]
testY=[]

for line in IF:
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date,next_date])
    testX.append(list(map(int, x.split(","))))
    testY.append(int(y))
testX=np.array(testX)
testY=np.array(testY)

with open('model_xgb.pickle', 'rb') as f:
    clf = pickle.load(f)

predY = clf.predict_proba(testX) # predict_proba 함수는 예측
predY2 = clf.predict(testX)  # predict 함수는 예측한 값을 0,
```

```
lst_result=[]
for (code, date, next_date),y in zip(lst_code_date, predY):
    if y[1] >= 0.4:
        lst_result.append([code, date, "buy", "r90"])
        lst_result.append([code, next_date, "sell", "all"])

lst_result.sort(key=lambda x:x[1])

OF=open('trading2022firsthalf.txt','w')
for row in lst_result:
    OF.write('₩t'.join(map(str,row))+'₩n')
OF.close()
```

```
Final earning rate : 122.95991 %
```

```
0.7775924381428969
```

```
start_money = 10000000 # 초기 현금 1천만원
money = start_money
dic_code2num ={}  # 보유 종목


IF=open("trading2022firsthalf.txt",'r')
for i, line in tqdm(enumerate(IF)): #주문 일지를 한 줄 읽어 옴
    code, date, request, amount = line.strip().split("₩t")
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date = {}
                '''.format(code, date)
    lst_stock = pd.read_sql(sql = sql_query, con = db_dsml).values.tolist()
    for row in lst_stock:
#         if date in row[0].strftime('%Y%m%d'):
        close = row[4]

    if request == 'buy': # buy인 경우
        if amount.startswith('r'):
            request_money = money * float(amount.lstrip("r")) / 100
        elif amount == 'all':
            request_money = money
        elif amount.isdigit():
            request_money = int(amount)
        # elif amount == ~~~~~    ##### 기타 필요한 매수 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        request_money = min(request_money, money)
        buy_num = int(request_money / close)
        money -= buy_num * close  # 현재 금액(money)을 실제 매수액을 뺀 만큼 업데이트
        if code not in dic_code2num:
            dic_code2num[code] = 0
        dic_code2num[code] += buy_num # 보유 종목 데이터에 구매 종목(code)를 매수 개수 만
    if request == 'sell': # sell인 경우
        if amount == 'all':
            sell_num = dic_code2num[code]
        # elif amount == ~~~~~    ##### 기타 필요한 매도 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        money += sell_num * close
        dic_code2num[code] -= sell_num
        if dic_code2num[code] == 0:
            del dic_code2num[code]
IF.close()

if dic_code2num != {}: # 매매가 종료되었는데 보유 종목이 있으면
    raise Exception('Not empty stock')

print("Final earning rate : {} %".format(str((money-start_money) / start_money * 100)))
```

100억 이상의 검증데이터셋을 사용해 XGBClassifier 모델로 학습 한 결과 정확도가 78%이며,

수익률을 123%에 도달하는 꽤 좋은 결과가 나왔다.

03 결과(학습데이터셋으로 학습한 모델로 private시험데이터셋 예측)


```
IF=open("assignment3_sql.txt",'r')

lst_code_date = []
trainX = []
trainY = []

for line in tqdm(IF):
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date, next_date])
    trainX.append(list(map(int, x.split(","))))
    trainY.append(int(y))

trainX=np.array(trainX)
trainY=np.array(trainY)

clf = XGBClassifier(n_estimators=300, nthread=1)
clf.fit(trainX, trainY)

with open('model_xgb.pickle', 'wb') as f:
    pickle.dump(clf, f)
```

```
IF=open("assignment3-3_sql.txt",'r')
lst_code_date=[]
private_testX=[]
private_testY=[]

for line in IF:
    code, date, next_date, x, y = line.strip().split("₩t")
    lst_code_date.append([code, date, next_date])
    private_testX.append(list(map(int, x.split(","))))
    private_testY.append(int(y))
private_testX=np.array(private_testX)
private_testY=np.array(private_testY)


with open('model_xgb.pickle', 'rb') as f:
    clf = pickle.load(f)

private_predY = clf.predict_proba(private_testX)
private_predY2 = clf.predict(private_testX)  # predict 함수.
```

```
lst_result=[]
for (code, date, next_date),y in zip(lst_code_date, private_predY):
    if y[1] >= 0.3:
        lst_result.append([code, date, "buy", "r80"])
        lst_result.append([code, next_date, "sell", "all"])

lst_result.sort(key=lambda x:x[1])

OF=open('trading2022secondhalf.txt','w')
for row in lst_result:
    OF.write('₩t'.join(map(str,row))+'₩n')
OF.close()
```

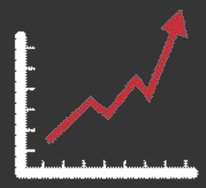
```
Final earning rate : 29.20986 %
''
0.7925925925925926
```

```
start_money = 10000000 # 초기 현금 1천만원
money = start_money
dic_code2num ={}  # 보유 종목


IF=open("trading2022secondhalf.txt",'r')
for i, line in tqdm(enumerate(IF)): #주문 일지를 한 줄 읽어 옴
    code, date, request, amount = line.strip().split("₩t")
    sql_query = '''
                SELECT *
                FROM stock_{}
                WHERE Date = {}
                '''.format(code, date)
    lst_stock = pd.read_sql(sql = sql_query, con = db_dsml).values.tolist()
    for row in lst_stock:
#         if date in row[0].strftime('%Y%m%d'):
        close = row[4]

    if request == 'buy': # buy인 경우
        if amount.startswith('r'):
            request_money = money * float(amount.lstrip("r")) / 100
        elif amount == 'all':
            request_money = money
        elif amount.isdigit():
            request_money = int(amount)
        # elif amount == ~~~~~    ##### 기타 필요한 매수 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        request_money = min(request_money, money)
        buy_num = int(request_money / close)
        money -= buy_num * close  # 현재 금액(money)을 실제 매수액을 뺀 만큼 업데이트
        if code not in dic_code2num:
            dic_code2num[code] = 0
        dic_code2num[code] += buy_num # 보유 종목 데이터에 구매 종목(code)를 매수 개수 만
    if request == 'sell': # sell인 경우
        if amount == 'all':
            sell_num = dic_code2num[code]
        # elif amount == ~~~~~    ##### 기타 필요한 매도 요청 옵션이 있을 시 작성
        else:
            raise Exception('Not permitted option')
        money += sell_num * close
        dic_code2num[code] -= sell_num
        if dic_code2num[code] == 0:
            del dic_code2num[code]
IF.close()

if dic_code2num != {}: # 매매가 종료되었는데 보유 종목이 있으면
    raise Exception('Not empty stock')

print("Final earning rate : {} %".format(str((money-start_money) / start_money * 100)))
```

100억 이상의 privae시험데이터셋을 사용해 XGBClassifier 모델로 학습 한 결과 정확도가 79%이며, 수익률을 29%에 도달하는 결과가 나왔다.

PART 01

결과



본 실험에서 주식 데이터를 전처리하고, 학습시켜 다양한 머신러닝 기법으로 정확도와 수익률을 분석했다. 다양한 학습 방법 중 XGBClassifier의 모델이 가장 이 데이터에 적합하다는 결론이 나왔다.
검증데이터셋으로 예측한 결과 123%의 높은 수익률이 나왔으며 privata시험데이터셋으로 예측한 결과 29%의 낮지 않은 수익률이 나왔다

감사합니다.