同济大学计算机系

数据挖掘期末大作业实验报告

学　　号　1854154

姓　　名　尹恒

专　　业　计算机科学与技术

授课老师　向阳

目录

一. 研究背景与意义

预测股票市场如何表现是最困难的事情之一。预测涉及很多因素-物理因素和生理因素，理性和非理性行为等。所有这些因素共同导致股票价格波动，很难以高精度预测。

随着中国经济建设的发展，越来越多的民众选择投资股市来实现自己资产的增值。入市股民也越来越关注如何从上千只股票中选择优质股，以及如何把握股票的涨跌规律这个难题。由于散户们获取关于上市公司信息的渠道有限，他们更倾向于使用历史数据的 K 线图以及各种各样的技术指标进行分析研判，并预测股票走势。而股民们使用较多的，包括了移动平均线 MA、随机指标 KDJ、趋势指标 MACD、相对强弱指标 RSI 以及布林带、换手率、量比、内外盘等技术指标。

数据挖掘领域的发展，越来越多的研究者希望借用计算机算法来对股票市场进行深度分析和挖掘，从而得到股票涨跌规律。使用这些方法的优势首先在于克服了传统指标回测方法无法对多种因素同时考虑的缺陷，使计算机可以像投资者一样对当前的行情进行综合研判。

二. 问题描述

2.1. 实现目标

- 对上证指数(SH000001)历史数据进行分析
- 数据挖掘得到有价值的特征因子
- 进行数据建模分析
- 预测上证指数明日的涨跌
- 对结果分析与评判

2.2. 实现情况

- 能对股票数据进行处理清理和分析
- 能对股票行情进行特征工程分析
- 完成对股票预测的多个模型建模并评价
- 完成多个优秀模型的融合模型
- 对 2020 年 12 月上证指数的预测准确率达到了 69%
- 对 2021 年 1 月前 7 天预测的准确率达到了 75%

三. 数据获取与清理

3.1. 下载数据

数据主要是在网易财经网站进行选择与下载。
进入网易财经上证指数板块，点击“历史交易数据”

下载完整的数据。

下载下来的数据格式如下：

	A	B	C	D	E	F	G	H	I	J	K	L
1	日期	股票代码	名称	收盘价	最高价	最低价	开盘价	前收盘	涨跌额	涨跌幅	成交量	成交金额
2	2021/1/8	'000001	上证指数	3570.108	3588.063	3544.891	3577.692	3576.205	-6.0964	-0.1705	3.46E+08	5.02E+11
3	2021/1/7	'000001	上证指数	3576.205	3576.205	3526.617	3552.909	3550.877	25.3279	0.7133	4.05E+08	5.46E+11
4	2021/1/6	'000001	上证指数	3550.877	3556.802	3513.126	3530.907	3528.677	22.2	0.6291	3.7E+08	5.22E+11
5	2021/1/5	'000001	上证指数	3528.677	3528.677	3484.715	3492.191	3502.958	25.7183	0.7342	4.08E+08	5.68E+11
6	2021/1/4	'000001	上证指数	3502.958	3511.655	3457.206	3474.679	3473.069	29.8891	0.8606	3.81E+08	5.23E+11
7	2020/12/31	'000001	上证指数	3473.069	3474.918	3419.727	3419.727	3414.453	58.6166	1.7167	3.36E+08	4.50E+11
8	2020/12/30	'000001	上证指数	3414.453	3414.454	3374.416	3375.009	3379.036	35.4165	1.0481	2.91E+08	3.78E+11
9	2020/12/29	'000001	上证指数	3379.036	3407.088	3376.088	3399.294	3397.285	-18.2492	-0.5372	3.12E+08	3.82E+11
10	2020/12/28	'000001	上证指数	3397.285	3412.519	3383.654	3396.359	3396.563	0.7228	0.0213	3.16E+08	3.98E+11
11	2020/12/25	'000001	上证指数	3396.563	3397.007	3348.345	3351.79	3363.113	33.4493	0.9946	2.95E+08	3.59E+11
12	2020/12/24	'000001	上证指数	3363.113	3394.075	3354.022	3382.193	3382.32	-19.2062	-0.5678	2.75E+08	3.63E+11
13	2020/12/23	'000001	上证指数	3382.32	3394.209	3360.202	3362.472	3356.782	25.5373	0.7608	3E+08	4.13E+11
14	2020/12/22	'000001	上证指数	3356.782	3415.754	3353.858	3410.968	3420.569	-63.7871	-1.8648	3.23E+08	4.32E+11
15	2020/12/21	'000001	上证指数	3420.569	3423.609	3381.116	3394.395	3394.896	25.6733	0.7562	2.84E+08	3.84E+11
16	2020/12/18	'000001	上证指数	3394.896	3413.813	3382.751	3400.486	3404.873	-9.9772	-0.293	2.81E+08	3.54E+11
17	2020/12/17	'000001	上证指数	3404.873	3406.155	3354.011	3367.277	3366.983	37.89	1.1253	2.75E+08	3.65E+11
18	2020/12/16	'000001	上证指数	3366.983	3378.663	3359.171	3371.263	3367.233	-0.2494	-0.0074	2.21E+08	3.09E+11
19	2020/12/15	'000001	上证指数	3367.233	3373.558	3348.419	3366.581	3369.12	-1.8875	-0.056	2.26E+08	3.15E+11
20	2020/12/14	'000001	上证指数	3369.12	3371.129	3338.626	3349.534	3347.191	21.9291	0.6551	2.4E+08	3.16E+11
21	2020/12/11	'000001	上证指数	3347.191	3383.178	3325.172	3381.008	3373.276	-26.0848	-0.7733	2.99E+08	3.80E+11
22	2020/12/10	'000001	上证指数	3373.276	3384.888	3357.753	3365.731	3371.964	1.3118	0.0389	2.47E+08	3.22E+11
23	2020/12/9	'000001	上证指数	3371.964	3422.536	3371.916	3416.078	3410.177	-38.2131	-1.1206	2.61E+08	3.39E+11
24	2020/12/8	'000001	上证指数	3410.177	3428.662	3403.026	3417.694	3416.604	6.4266	0.1881	2.27E+08	3.05E+11

该数据包含了从 1990 年 12 月 19 日到 2021 年 1 月 8 日的上证指数数据。具体指标有：

- 收盘价
- 最高价
- 最低价
- 开盘价
- 前收盘
- 涨跌额
- 涨跌幅
- 成交量
- 成交金额

3.2. 数据清理

1) 中文转英文

由于中文内容会在读取文件的时候产生一定问题，这里手动在 Excel 中将中文重命名为对于英文。

	A	B	C	D	E	F	G	H	I	J	K
1	date	symbol	close	high	low	open	pre_close	change	pct_chg	volume	amount
2	2021/1/8	SH000001	3570.108	3588.063	3544.891	3577.692	3576.205	-6.0964	-0.1705	3.46E+08	5.02E+11
3	2021/1/7	SH000001	3576.205	3576.205	3526.617	3552.909	3550.877	25.3279	0.7133	4.05E+08	5.46E+11
4	2021/1/6	SH000001	3550.877	3556.802	3513.126	3530.907	3528.677	22.2	0.6291	3.7E+08	5.22E+11
5	2021/1/5	SH000001	3528.677	3528.677	3484.715	3492.191	3502.958	25.7183	0.7342	4.08E+08	5.68E+11
6	2021/1/4	SH000001	3502.958	3511.655	3457.206	3474.679	3473.069	29.8891	0.8606	3.81E+08	5.23E+11

2) 检查数据

原始数据在 python 中读取为 df 格式如下：

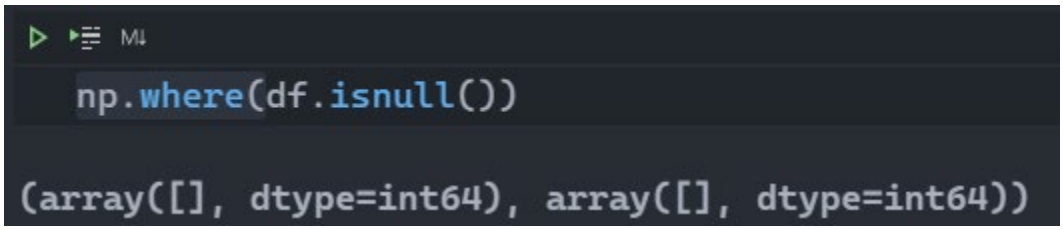
	date	symbol	close	high	low	open	pre_close	change	pct_chg	volume	amount
0	2021/1/8	SH000001	3570.1082	3588.0625	3544.8912	3577.6923	3576.2046	-6.0964	-0.1705	345557896	5.02E+11
1	2021/1/7	SH000001	3576.2046	3576.2046	3526.6174	3552.9087	3550.8767	25.3279	0.7133	405348226	5.46E+11
2	2021/1/6	SH000001	3550.8767	3556.8022	3513.1262	3530.9072	3528.6767	22.2	0.6291	370230926	5.22E+11
3	2021/1/5	SH000001	3528.6767	3528.6767	3484.7151	3492.1912	3502.9584	25.7183	0.7342	407995934	5.68E+11
4	2021/1/4	SH000001	3502.9584	3511.6554	3457.2061	3474.6793	3473.0693	29.8891	0.8606	380790800	5.23E+11
...	...	...	...	...	...	...	...	...	...	...	...
7343	1990/12/25	SH000001	120.2500	120.2500	114.5500	120.0900	114.55	5.7	4.976	15	6000
7344	1990/12/24	SH000001	114.5500	114.5500	109.1300	113.5700	109.13	5.42	4.9666	32	31000
7345	1990/12/21	SH000001	109.1300	109.1300	103.7300	109.0700	104.39	4.74	4.5407	28	16000
7346	1990/12/20	SH000001	104.3900	104.3900	99.9800	104.3000	99.98	4.41	4.4109	197	84000
7347	1990/12/19	SH000001	99.9800	99.9800	95.7900	96.0500	None	None	None	1260	494000

7348 rows × 11 columns

可以看到，数据格式比较正确，可以满足后续的工作。

3) 处理缺失数据

查看空值位置



并没有空值出现

将数值类型的列限制类型为 float，若出错则定为 NaN

```
# 数据类型限制为float， 无法转换则为NaN
float_cols = ['close', 'high', 'low', 'open', 'pre_close', 'change', 'pct_chg',
'volume', 'amount']
df[float_cols] = df[float_cols].apply(pd.to_numeric, errors = 'coerce')
df
```

	date	symbol	close	high	low	open	pre_close	change	pct_chg	volume	amount
0	2021/1/8	SH000001	3570.1082	3588.0625	3544.8912	3577.6923	3576.2046	-6.0964	-0.1705	345557896	5.020000e+11
1	2021/1/7	SH000001	3576.2046	3576.2046	3526.6174	3552.9087	3550.8767	25.3279	0.7133	405348226	5.460000e+11
2	2021/1/6	SH000001	3550.8767	3556.8022	3513.1262	3530.9072	3528.6767	22.2000	0.6291	370230926	5.220000e+11
3	2021/1/5	SH000001	3528.6767	3528.6767	3484.7151	3492.1912	3502.9584	25.7183	0.7342	407995934	5.680000e+11
4	2021/1/4	SH000001	3502.9584	3511.6554	3457.2061	3474.6793	3473.0693	29.8891	0.8606	380790800	5.230000e+11
...	...	...	...	...	...	...	...	...	...	...	...
7343	1990/12/25	SH000001	120.2500	120.2500	114.5500	120.0900	114.5500	5.7000	4.9760	15	6.000000e+03
7344	1990/12/24	SH000001	114.5500	114.5500	109.1300	113.5700	109.1300	5.4200	4.9666	32	3.100000e+04
7345	1990/12/21	SH000001	109.1300	109.1300	103.7300	109.0700	104.3900	4.7400	4.5407	28	1.600000e+04
7346	1990/12/20	SH000001	104.3900	104.3900	99.9800	104.3000	99.9800	4.4100	4.4109	197	8.400000e+04
7347	1990/12/19	SH000001	99.9800	99.9800	95.7900	96.0500	NaN	NaN	NaN	1260	4.940000e+05

7348 rows × 11 columns

丢弃掉含 NaN 的行

```
# 丢弃NaN行
df = df.dropna()
df
```

	date	symbol	close	high	low	open	pre_close	change	pct_chg	volume	amount
0	2021/1/8	SH000001	3570.1082	3588.0625	3544.8912	3577.6923	3576.2046	-6.0964	-0.1705	345557896	5.020000e+11
1	2021/1/7	SH000001	3576.2046	3576.2046	3526.6174	3552.9087	3550.8767	25.3279	0.7133	405348226	5.460000e+11
2	2021/1/6	SH000001	3550.8767	3556.8022	3513.1262	3530.9072	3528.6767	22.2000	0.6291	370230926	5.220000e+11
3	2021/1/5	SH000001	3528.6767	3528.6767	3484.7151	3492.1912	3502.9584	25.7183	0.7342	407995934	5.680000e+11
4	2021/1/4	SH000001	3502.9584	3511.6554	3457.2061	3474.6793	3473.0693	29.8891	0.8606	380790800	5.230000e+11
...	...	...	...	...	...	...	...	...	...	...	...
7342	1990/12/26	SH000001	125.2700	125.2700	120.2500	125.2700	120.2500	5.0200	4.1746	100	5.300000e+04
7343	1990/12/25	SH000001	120.2500	120.2500	114.5500	120.0900	114.5500	5.7000	4.9760	15	6.000000e+03
7344	1990/12/24	SH000001	114.5500	114.5500	109.1300	113.5700	109.1300	5.4200	4.9666	32	3.100000e+04
7345	1990/12/21	SH000001	109.1300	109.1300	103.7300	109.0700	104.3900	4.7400	4.5407	28	1.600000e+04
7346	1990/12/20	SH000001	104.3900	104.3900	99.9800	104.3000	99.9800	4.4100	4.4109	197	8.400000e+04

7337 rows × 11 columns

4) 检查重复数据

```
# 检查是否有重复时间行
np.where(df.duplicated(['date']))
```

```
(array([], dtype=int64),)
```

对 date 列进行重复判断，并未发现有重复数据，因此无需去重。

5) 保存清理完的数据

到此，数据基本清理完毕，进行保存。

```
[7]
# 保存处理好的数据
df.to_csv('./data/pre/SH000001.csv')
```

3.3. 数据转换为 qlib 的 bin 文件格式

qlib 量化分析库提供一个一种十分高效的数据存储格式，并且提供了脚本文件供用户将自己的数据集转换为 qlib 的 bin 文件格式。

转化过程中需要指定一些必要的参数如下：

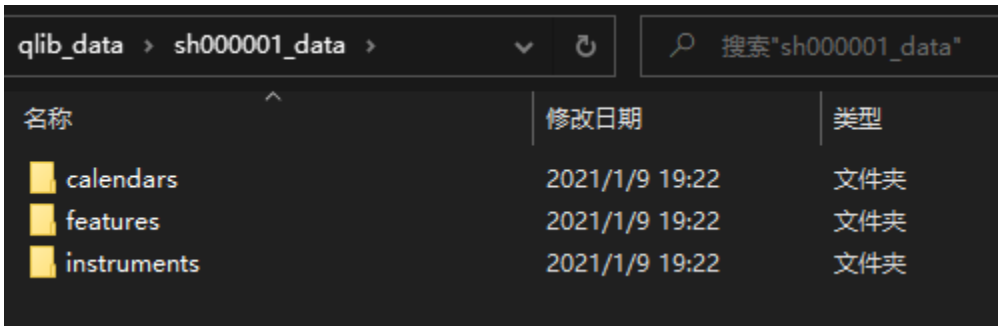
--date_field_name：日期域

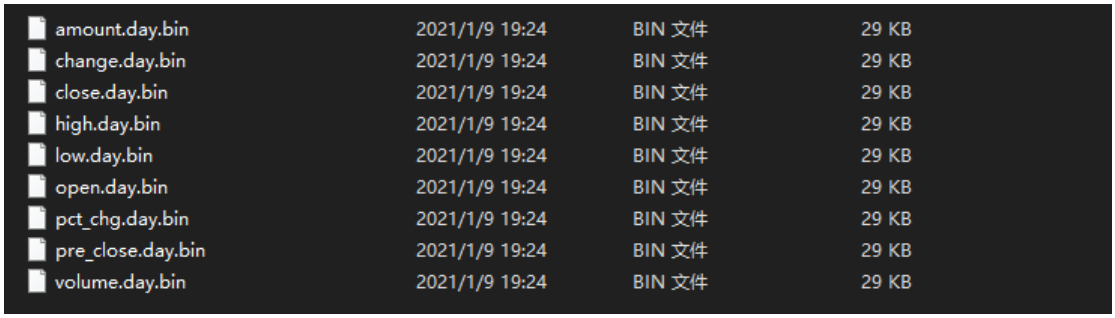
--symbol_field_name：股票代码域

执行如下命令，进行数据转换：

```
yin :: ~\source\repos\data-mining\qlib <main +1 ~0 -0 !> » python scripts/dump_bin.py dump_all --csv_path ./mytest/data/pre/SH000001.csv --qlib_dir ./mytest/data/qlib_data/sh000001_data --date_field_name date
```

得到 qlib 数据文件如下：





现在来查看一下 qlib 格式的数据：

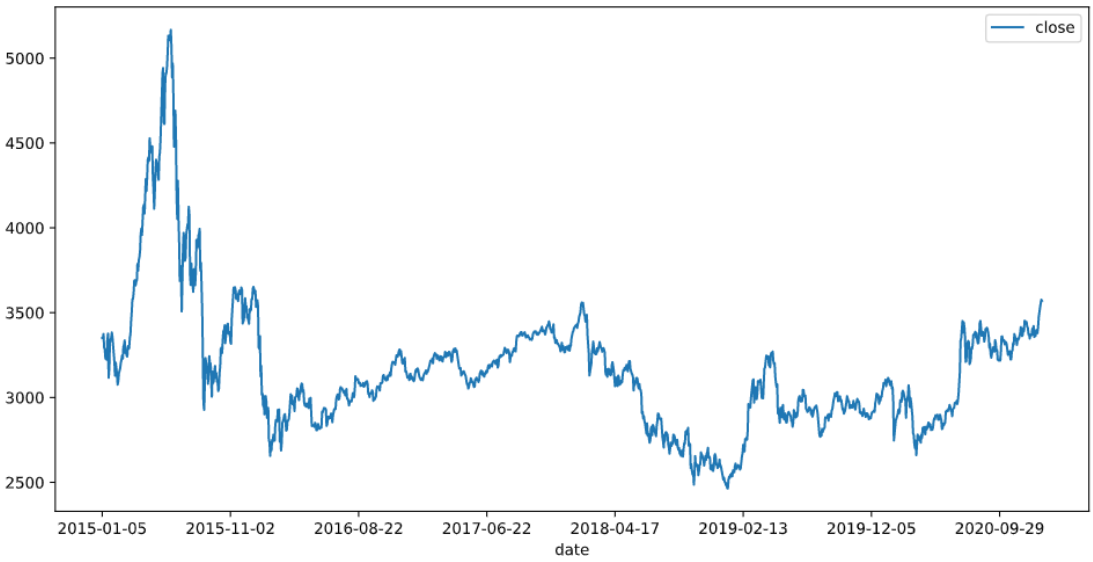
```
fields = ["$open", "$close", "$high", "$low", "$pre_close", "$change", "$pct_chg", "$volume", "$amount"]
df = D.features(D.instruments('all'), fields)
df.head()
```

instrument	datetime	$open	$close	$high	$low	$pre_close	$change	$pct_chg	$volume	$amount
SH000001	1990-12-20	104.300003	104.389999	104.389999	99.980003	99.980003	4.41	4.4109	197.0	84000.0
	1990-12-21	109.070000	109.129997	109.129997	103.730003	104.389999	4.74	4.5407	28.0	16000.0
	1990-12-24	113.570000	114.550003	114.550003	109.129997	109.129997	5.42	4.9666	32.0	31000.0
	1990-12-25	120.089996	120.250000	120.250000	114.550003	114.550003	5.70	4.9760	15.0	6000.0
	1990-12-26	125.269997	125.269997	125.269997	120.250000	120.250000	5.02	4.1746	100.0	53000.0

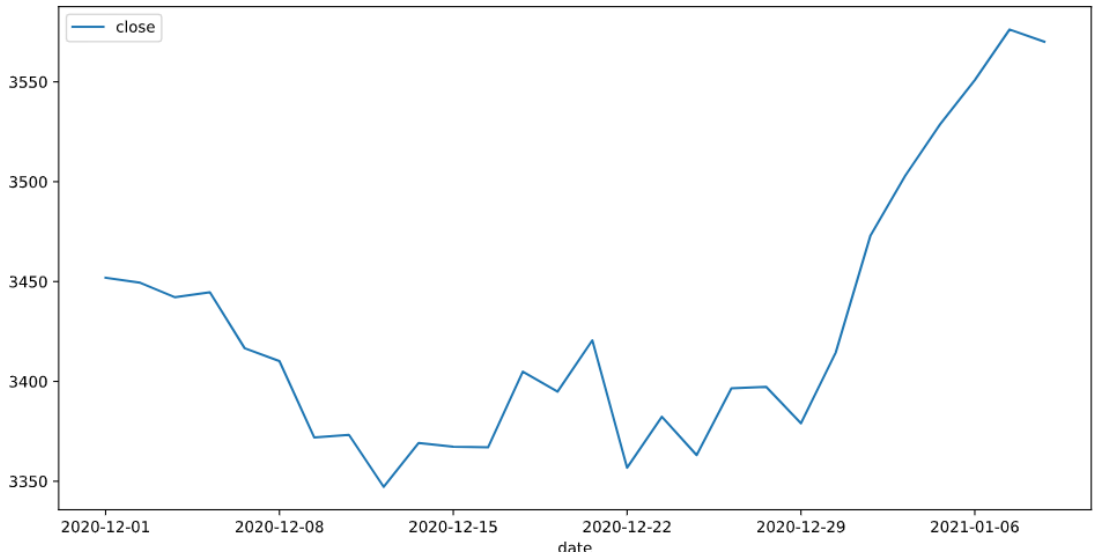
四. 行情特征工程

4.1. 收盘价变化趋势

15 年到最近的收盘价变化趋势



最近一个月的变化趋势



4.2. 以 qlib 的 Alpha158 数据集作为特征

构造 handler 数据处理器

qlib 自带了两个数据集，分别为 Alhpa158 和 Alpha360，开始进行尝试的时候，可以直接用 Alpha158 的 X 数据，label 转化为明日涨跌情况的二分类，1 代表涨，0 代表跌。虽然这是一种盲目的方法，但获取可以为开始寻找特征提供一点思路。

这里仿照 Alpha158 重写了一个自己的数据处理器 Alpha158up_down，它的 feature 与 Alpha158 一样，是 158 个由原始特征得到的特征因子。比如 ROC、MA、STD、BETA 等。

```
if use("ROC"):
    fields += ["Ref($close, %d)/$close" % d for d in windows]
    names += ["ROC%d" % d for d in windows]
if use("MA"):
    fields += ["Mean($close, %d)/$close" % d for d in windows]
    names += ["MA%d" % d for d in windows]
if use("STD"):
    fields += ["Std($close, %d)/$close" % d for d in windows]
    names += ["STD%d" % d for d in windows]
if use("BETA"):
    fields += ["Slope($close, %d)/$close" % d for d in windows]
    names += ["BETA%d" % d for d in windows]
if use("RSQR"):
    fields += ["Rsquare($close, %d)" % d for d in windows]
    names += ["RSQR%d" % d for d in windows]
if use("RESI"):
    fields += ["Resi($close, %d)/$close" % d for d in windows]
    names += ["RESI%d" % d for d in windows]
if use("MAX"):
    fields += ["Max($high, %d)/$close" % d for d in windows]
    names += ["MAX%d" % d for d in windows]
if use("LOW"):
    fields += ["Min($low, %d)/$close" % d for d in windows]
    names += ["MIN%d" % d for d in windows]
```

它的 label 定义为如下：

```python
def get_label_config(self):
    # return (["Ref($close, -2)/Ref($close, -1) - 1"], ["LABEL0"])
    return (["Ge(Ref($close, -1), $close)"], ["LABEL0"])
```

比较后一天的收盘价与今天的收盘价，得到一个二类值。

初步模型训练

模型参数如下：

```
"model": {
    "class": "LGBModel",
    "module_path": "qlib.contrib.model.gbdt",
    "kwargs": {
        "loss": "binary",
        "colsample_bytree": 0.8879,
        "learning_rate": 0.0421,
        "subsample": 0.8789,
        "lambda_l1": 205.6999,
        "lambda_l2": 580.9768,
        "max_depth": 8,
        "num_leaves": 210,
        "num_threads": 20,
    },
},
```

使用 lightGBM 的二分类模型，进行训练。

数据集划分如下：

```
"dataset": {
    "class": "DatasetH",
    "module_path": "qlib.data.dataset",
    "kwargs": {
        "handler": {
            "class": "Alpha158up_down",
            "module_path": "qlib.contrib.data.handler",
            "kwargs": data_handler_config,
        },
        "segments": {
            "train": ("2010-01-01", "2017-12-29"),
            "valid": ("2018-01-02", "2019-12-31"),
            "test": ("2020-01-02", "2020-12-31"),
        },
    },
},
```

训练集：2010-01-01——2017-12-29
验证集：2018-01-02——2019-12-31
测试集：2020-01-02——2020-12-31

训练结果如下：

```
# start exp to train model
with R.start(experiment_name="train_model"):
    R.log_params(**flatten_dict(task))
    # print(dataset.prepare('train'))
    model.fit(dataset)
    R.save_objects(trained_model=model)
    rid = R.get_recorder().id
```

```
[7184:MainThread](2021-01-09 20:09:52,841) INFO - qlib.workflow - [expm.py:245] - No tracking URI is provided. Use the default tracking URI.
[7184:MainThread](2021-01-09 20:09:52,854) INFO - qlib.workflow - [exp.py:181] - Experiment 1 starts running ...
[7184:MainThread](2021-01-09 20:09:53,162) INFO - qlib.workflow - [recorder.py:236] - Recorder 41d7e6bde90e47449f25084fab6f400b starts running under Experiment 1 ...
Training until validation scores don't improve for 50 rounds
[20]    train's binary_logloss: 0.693137        valid's binary_logloss: 0.70025
[40]    train's binary_logloss: 0.693137        valid's binary_logloss: 0.70025
Early stopping, best iteration is:
[1]     train's binary_logloss: 0.693137        valid's binary_logloss: 0.70025
```

预测准确率如下：

```
from sklearn.metrics import accuracy_score
accuracy_score(true_result, pred_result)
```

```
0.5390946502057613
```

该模型能达到 53%的准确率。

4.3. 使用 talib 库获取行情指标

talib 库提给出了股票分析中的各种指标，并且有一套函数库用来计算相应的指标。以下是 talib 提供的所有函数：

```
['HT_DCPERIOD', 'HT_DCPHASE', 'HT_PHASOR', 'HT_SINE', 'HT_TRENDMODE', 'ADD', 'DIV', 'MAX', 'MAXINDEX', 'MIN', 'MININD
EX', 'MINMAX', 'MINMAXINDEX', 'MULT', 'SUB', 'SUM', 'ACOS', 'ASIN', 'ATAN', 'CEIL', 'COS', 'COSH', 'EXP', 'FLOOR', 'L
N', 'LOG10', 'SIN', 'SINH', 'SQRT', 'TAN', 'TANH', 'ADX', 'ADXR', 'APO', 'AROON', 'AROONOSC', 'BOP', 'CCI', 'CMO', 'D
X', 'MACD', 'MACDEXT', 'MACDFIX', 'MFI', 'MINUS_DI', 'MINUS_DM', 'MOM', 'PLUS_DI', 'PLUS_DM', 'PPO', 'ROC', 'ROCP', '
ROCR', 'ROCR100', 'RSI', 'STOCH', 'STOCHF', 'STOCHRSI', 'TRIX', 'ULTOSC', 'WILLR', 'BBANDS', 'DEMA', 'EMA', 'HT_TREND
LINE', 'KAMA', 'MA', 'MAMA', 'MAVP', 'MIDPOINT', 'MIDPRICE', 'SAR', 'SAREXT', 'SMA', 'T3', 'TEMA', 'TRIMA', 'WMA', 'C
DL2CROWS', 'CDL3BLACKCROWS', 'CDL3INSIDE', 'CDL3LINESTRIKE', 'CDL3OUTSIDE', 'CDL3STARSINSOUTH', 'CDL3WHITESOLDIERS',
'CDLABANDONEDBABY', 'CDLADVANCEBLOCK', 'CDLBELTHOLD', 'CDLBREAKAWAY', 'CDLCLOSINGMARUBOZU', 'CDLCONCEALBABYSWALL', 'C
DLCOUNTERATTACK', 'CDLDARKCLOUDCOVER', 'CDLDOJI', 'CDLDOJISTAR', 'CDLDRAGONFLYDOJI', 'CDLENGULFING', 'CDLEVENINGDOJIS
TAR', 'CDLEVENINGSTAR', 'CDLGAPSIDESIDEWHITE', 'CDLGRAVESTONEDOJI', 'CDLHAMMER', 'CDLHANGINGMAN', 'CDLHARAMI', 'CDLHA
RAMICROSS', 'CDLHIGHWAVE', 'CDLHIKKAKE', 'CDLHIKKAKEMOD', 'CDLHOMINGPIGEON', 'CDLIDENTICAL3CROWS', 'CDLINNECK', 'CDLI
NVERTEDHAMMER', 'CDLKICKING', 'CDLKICKINGBYLENGTH', 'CDLLADDERBOTTOM', 'CDLLONGLEGGEDDOJI', 'CDLLONGLINE', 'CDLMARUBO
ZU', 'CDLMATCHINGLOW', 'CDLMATHOLD', 'CDLMORNINGDOJISTAR', 'CDLMORNINGSTAR', 'CDLONNECK', 'CDLPIERCING', 'CDLRICKSHAW
MAN', 'CDLRISEFALL3METHODS', 'CDLSEPARATINGLINES', 'CDLSHOOTINGSTAR', 'CDLSHORTLINE', 'CDLSPINNINGTOP', 'CDLSTALLEDPA
TTERN', 'CDLSTICKSANDWICH', 'CDLTAKURI', 'CDLTASUKIGAP', 'CDLTHRUSTING', 'CDLTRISTAR', 'CDLUNIQUE3RIVER', 'CDLUPSIDEG
AP2CROWS', 'CDLXSIDEGAP3METHODS', 'AVGPRICE', 'MEDPRICE', 'TYPPRICE', 'WCLPRICE', 'BETA', 'CORREL', 'LINEARREG', 'LIN
EARREG_ANGLE', 'LINEARREG_INTERCEPT', 'LINEARREG_SLOPE', 'STDDEV', 'TSF', 'VAR', 'ATR', 'NATR', 'TRANGE', 'AD', 'ADOS
C', 'OBV']
```

1) ADX 指标

ADX(Average Directional Movement Index)——平均趋向指数。
功能：判断盘整、振荡和单边趋势。
计算方法：
一、先决定股价趋势（Directional Movement，DM）是上涨或下跌：
“所谓 DM 值，今日股价波动幅度大于昨日股价波动幅部分的最大值，可能是创高价的部分或创低价的部分；如果今日股价波动幅度较前一日小，则 DM = 0。”
若股价高点持续走高，为上涨趋势，记作 +DM。
若为下跌趋势，记作 -DM。-DM 的负号（–）是表示反向趋势（下跌），并非数值为负数。
其他状况：DM = 0。
二、寻找股价的真实波幅（True Range，TR）：
所谓真实波幅（TR）是以最高价，最低价，及前一日收盘价三个价格做比较，求出当日股价波动的最大幅度。
三、趋势方向需经由一段时间来观察，研判上才有意义。一般以 14 天为指标的观察周期：先计算出 +DM、–DM 及 TR 的 14 日算术平均数，得到 +DM14、–DM14 及 TR14 三组数据作为起始值，再计算各自的移动平均值（EMA）。

```
+DI14 = +DM/TR14*100
-DI14 = +DM/TR14*100

DX = |(+DI14)-(-DI14)| / |(+DI14)+(-DI14)|

DX 运算结果取其绝对值，再将 DX 作移动平均，得到 ADX。
```

指标特点：

- ADX 无法告诉你趋势的发展方向。
- 如果趋势存在，ADX 可以衡量趋势的强度。不论上升趋势或下降趋势，ADX 看起来都

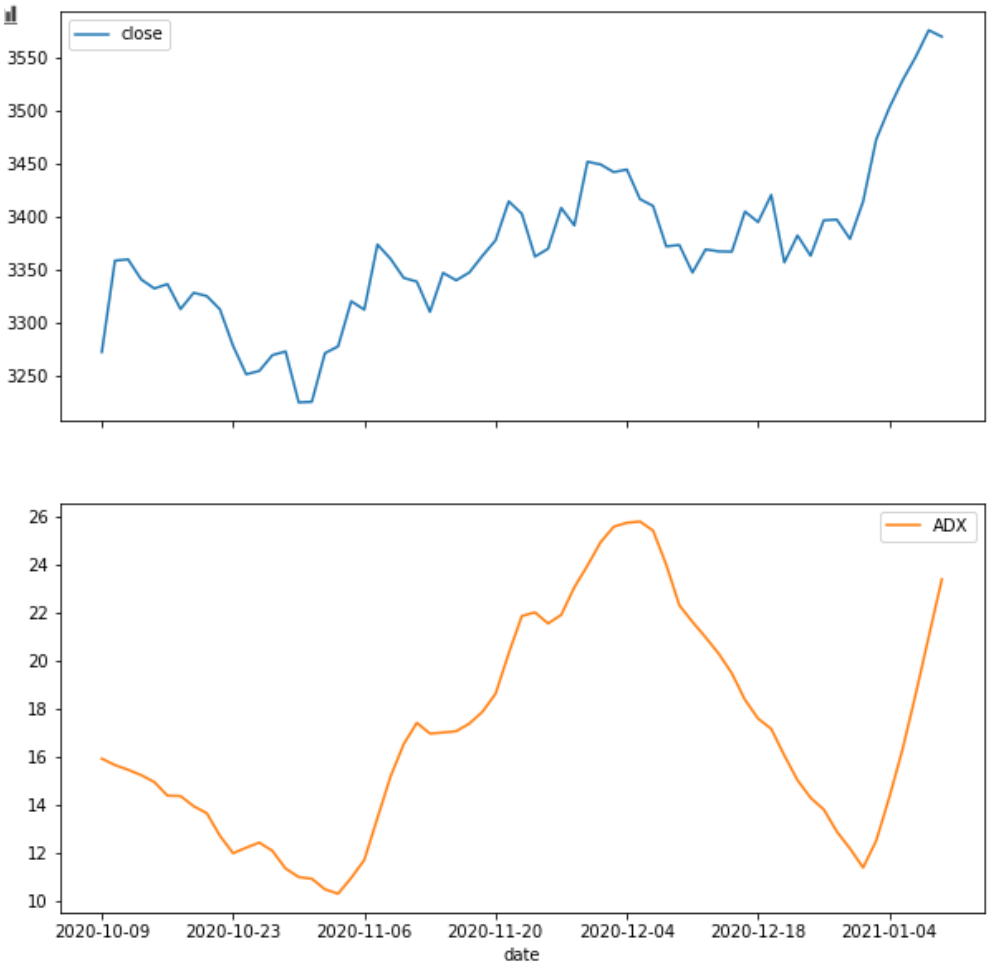
一样。

- ADX 的读数越大，趋势越明显。衡量趋势强度时，需要比较几天的 ADX 读数，观察 ADX 究竟是上升或下降。ADX 读数上升，代表趋势转强；如果 ADX 读数下降，意味着趋势转弱。
- 当 ADX 曲线向上攀升，趋势越来越强，应该会持续发展。如果 ADX 曲线下滑，代表趋势开始转弱，反转的可能性增加。
- 单就 ADX 本身来说，由于指标落后价格走势，所以算不上是很好的指标，不适合单就 ADX 进行操作。可是，如果与其他指标配合运用，ADX 可以确认市场是否存在趋势，并衡量趋势的强度。

得到 ADX 指标：

```
df['ADX'] = ta.ADX(df['high'], df['low'], df['close'], timeperiod=14)
df
```

进行可视化：



ADX 与收盘价走势对比图

从图中可以发现，ADX 指标一定程度上反映了股票的趋势，但是信息确实有一定延迟，不过依然可以作为一个特征来参考。

2) AROON 指标

该指标是通过计算自价格达到近期最高值和最低值以来所经过的期间数，阿隆指标帮助你预测价格趋势到趋势区域（或者反过来，从趋势区域到趋势）的变化。

Aroon(上升)=[(计算期天数-最高价后的天数)/计算期天数]*100

Aroon(下降)=[(计算期天数-最低价后的天数)/计算期天数]*100

```
df['AROONDOWN'], df['AROONUP'] = ta.AROON(df['high'], df['low'], timeperiod=14)
```

3) AROONOSC 指标

阿隆振荡指标。

```
df['AROONOSC'] = ta.AROONOSC(df['high'], df['low'], timeperiod=14)
```

4) MA 指标

MA(Moving average)——移动平均线。

移动平均线（MA）具有趋势的特性，它比较平稳，不像日 K 线会起起落落地震荡。越长期的移动平均线，越能表现稳定的特性。不轻易向上向下，必须等股价趋势的真正明朗。移动平均线说到底是一种趋势追踪工具，便于识别趋势已经终结或者反转，新的趋势是否正在形成。

（1）上升行情初期，短期移动平均线从下向上突破中长期移动平均线，形成的交叉叫黄金交叉。

预示股价将上涨：黄色的 5 日均线上穿紫色的 10 日均线形成的交叉；10 日均线上穿绿色的 30 日均线形成的交叉均为黄金交叉。

（2）当短期移动平均线向下跌破中长期移动平均线形成的交叉叫做死亡交叉。预示股价将下跌。黄色的 5 日均线下穿紫色的 10 日均线形成的交叉；10 日均线下穿绿色的 30 日均线形成的交叉均为死亡交叉。

（3）在上升行情进入稳定期，5 日、10 日、30 日移动平均线从上而下依次顺序排列，向右上方移动，称为多头排列。预示股价将大幅上涨。

（4）在下跌行情中，5 日、10 日、30 日移动平均线自下而上依次顺序排列，向右下方移动，称为空头排列，预示股价将大幅下跌。

（5）在上升行情中股价位于移动平均线之上，走多头排列的均线可视为多方的防线；当股价回档至移动平均线附近，各条移动平均线依次产生支撑力量，买盘入场推动股价再度上升，这就是移动平均线的助涨作用。

（6）在下跌行情中，股价在移动平均线的下方，呈空头排列的移动平均线可以视为空方的防线，当股价反弹到移动平均线附近时，便会遇到阻力，卖盘涌出，促使股价进一步下跌，这就是移动平均线的助跌作用。

（7）移动平均线由上升转为下降出现最高点，和由下降转为上升出现最低点时，是移动平均线的转折点。预示股价走势将发生反转。

本次实验特征构造中使用了 5 日、10 日、20 日、30 日、60 日的移动平均值。

```
df['MA5'] = ta.MA(df['close'], timeperiod=5, matype=0)
df['MA10'] = ta.MA(df['close'], timeperiod=10, matype=0)
df['MA20'] = ta.MA(df['close'], timeperiod=20, matype=0)
df['MA30'] = ta.MA(df['close'], timeperiod=30, matype=0)
df['MA60'] = ta.MA(df['close'], timeperiod=60, matype=0)
```

5) KD 指标

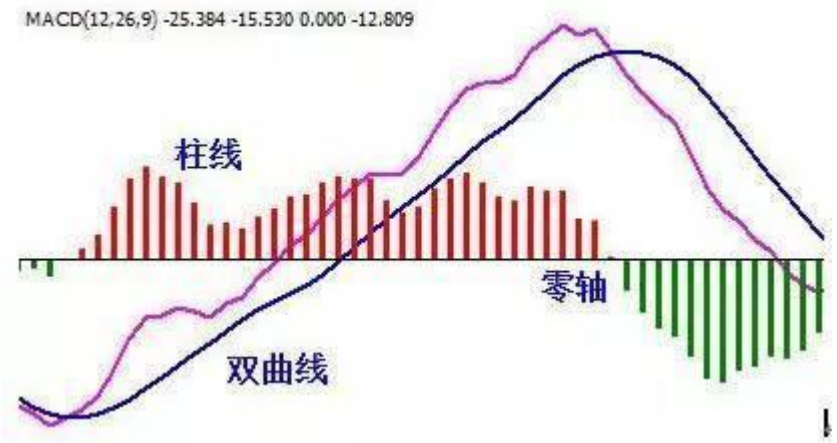
KDJ 指标又叫随机指标，是一种相当新颖、实用的技术分析指标，它起先用于期货市场的分析，后被广泛用于股市的中短期趋势分析，是期货和股票市场上最常用的技术分析工具。随机指标 KDJ 一般是用于股票分析的统计体系，根据统计学原理，通过一个特定的周期（常为 9 日、9 周等）内出现过的最高价、最低价及最后一个计算周期的收盘价及这三者之间的比例关系，来计算最后一个计算周期的未成熟随机值 RSV，然后根据平滑移动平均线的方法来计算 K 值、D 值与 J 值，并绘成曲线图来研判股票走势。

特征选取中使用了 KDJ 中的 K 和 D：

```
df['K'] , df['D']= ta.STOCHF(df['high'], df['low'], df['close'], fastk_period=5, fastd_period=3, fastd_matype=0)
```

6) MACD 指标

MACD 称为异同移动平均线，是从双指数移动平均线发展而来的，由快的指数移动平均线（EMA12）减去慢的指数移动平均线（EMA26）得到快线 DIF，再用 2×（快线 DIF-DIF 的 9 日加权移动均线 DEA）得到 MACD 柱。MACD 的意义和双移动平均线基本相同，即由快、慢均线的离散、聚合表征当前的多空状态和股价可能的发展变化趋势，但阅读起来更方便。MACD 的变化代表着市场趋势的变化，不同 K 线级别的 MACD 代表当前级别周期中的买卖趋势。



1. 当 DIF 和 DEA 均大于 0(即在图形上表示为它们处于零轴以上)并向上移动时，一般表示为股市处于多头行情中，可以买入或持股。

2. 当 DIF 和 DEA 均小于 0(即在图形上表示为它们处于零轴以下)并向下移动时，一般

表示为股市处于空头行情中，可以卖出股票或观望。

3. 当 DIF 和 DEA 均大于 0(即在图形上表示为它们处于零轴以上)但都向下移动时，一般表示为股票行情处于退潮阶段，股票将下跌，可以卖出股票和观望。

4. 当 DIF 和 DEA 均小于 0 时(即在图形上表示为它们处于零轴以下)但向上移动时，一般表示为行情即将启动，股票将上涨，可以买进股票或持股待涨。

MACD 指标也在本次构造的特征范围之内。

```
df['DIF'], df['DEM'], df['HISTOGRAM'] = ta.MACD(df['close'], fastperiod=12, slowperiod=26, signalperiod=9)
```

7) 布林线指标

布林线(BOLL)由约翰·布林先生创造，其利用统计原理，求出股价的标准差及其信赖区间，从而确定股价的波动范围及未来走势，利用波带显示股价的安全高低价位，因而也被称为布林带。其上下限范围不固定，随股价的滚动而变化。布林指标和麦克指标 MIKE 一样同属路径指标，股价波动在上限和下限的区间之内，这条带状区的宽窄，随着股价波动幅度的大小而变化，股价涨跌幅度加大时，带状区变宽，涨跌幅度狭小盘整时，带状区则变窄。

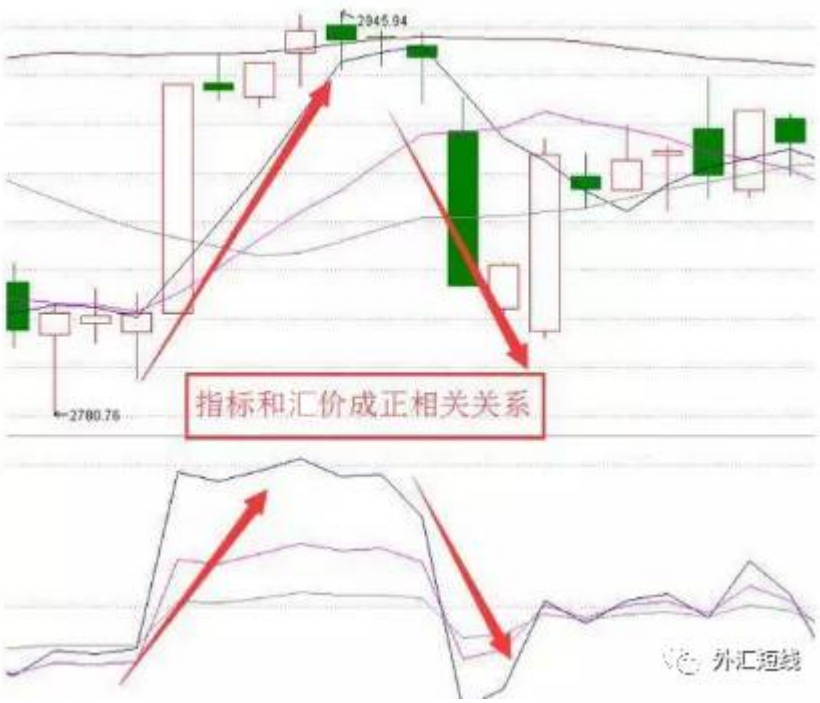
- 股价在中轨和上轨中间运行，处于多头市场，说明较为强势，同样，股价在下轨和中轨中间运行，处于空头市场，说明较为弱势；
- 股价由下向上穿越中轨时，为加仓信号，但可能会受到中轨的压力；
- 当股价由上向下中轨时，为减仓信号，同样可能会受到中轨的支撑；
- 股价穿越上轨时，卖出信号。同样当股价穿越下轨时，是买入信号；
- 当布林线的开口越来越小的时候，说明股价的波动在减小，多空双方力量趋于一致，此时股价将会选择突破方向；
- 当布林线的开口越来越大的时候，说明一方的力量在增强，股价在进行突破；
- 中轨既然是均线，就有趋势的指示作用，如果向上说明趋势向上，反之向下。

布林线指标获取如下：

```
# 布林线
df['UPPERBAND'], df['MIDDLEBAND'], df['LOWERBAND'] = ta.BBANDS(df['close'], timeperiod=5, nbdevup=2, nbdevdn=2, matype=0)
```

8) RSI 指标

RSI 指标最早是被用于期货交易中，后来人们发现用该指标来指导股票市场投资效果也十分不错，并对该指标的特点不断进行归纳和总结。



上图可以清晰的看清股价和 RSI 之间的变化关系，两者不同的是汇价涨跌没有限制，而 RSI 的数值只在 0—100 之间波动，根据经验，RSI 数值多数情况在 30—70 之间变动为正常，请注意：

当 RSI>80 时，属于超买状态，短期回调概率较高，建议卖出

当 RSI<20 时，属于超卖状态，短期反弹概率较高，建议买入

实验中用到了 6 日、12 日、24 日相对强弱指标 RSI。

```
df['RSI6'] = ta.RSI(df['close'], timeperiod=6)
df['RSI12'] = ta.RSI(df['close'], timeperiod=12)
df['RSI24'] = ta.RSI(df['close'], timeperiod=24)
```

9) EMA 指标

Exponential Moving Average，是一种趋向类指标，其构造原理是仍然对价格收盘价进行算术平均，并根据计算结果来进行分析，用于判断价格未来走势的变动趋势。

10) ATR 指标

ATR 是由 Wilder 在他的"NewConceptsinTechnicalTradingStrategies"一书中提出的。一般来说,一段时期内的价格波动可以定义为这段时期内的最高价与最低价之间的差值,但该定义忽略了另外一个事实:一个时间段到下一个时间段之间价格可能出现跳跃,从而使得一个时间段内的最低(高)价在前一个时间段的收盘价之上(下)。真实波动区间(TrueRange,记成 TR)的想法即由此而来,被定义为以下三个值的最大者:最高价减最低价、最高价减前收盘价、最低价减前收盘价。

ATR 指标有以下一些特性：

- 波动幅度的概念表示可以显示出交易者的期望和热情。
- 大幅的或增加中的波动幅度表示交易者在当天可能准备持续买进或卖出股票。
- 波动幅度的减少则表示交易者对股市没有太大的兴趣。

这里选用了 ATR 指标的归一化结果 NATR：

```
df['NATR'] = ta.NATR(df['high'], df['low'], df['close'], timeperiod=14)
```

后续实验会发现，该指标非常重要，**可以说是所有特征里最重要的。**

11) TRANGE 指标

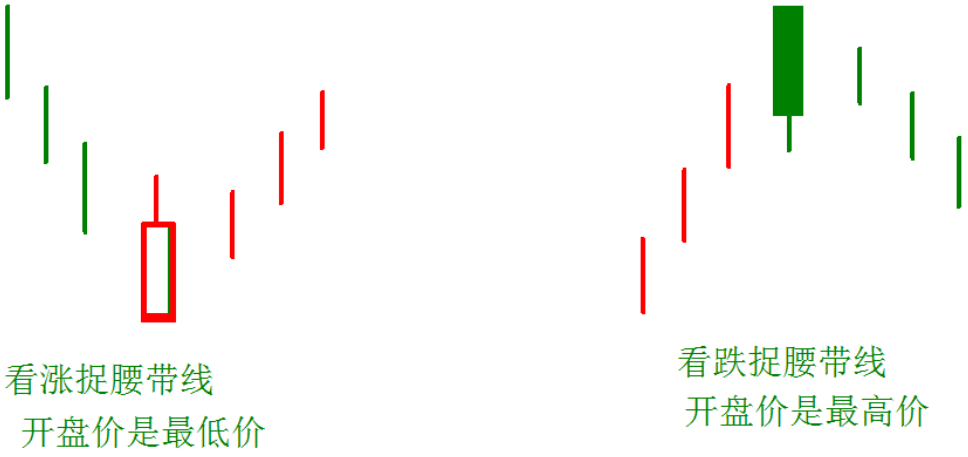
这也是用来反映波动率的一个指标，在实际测试中，它的重要性也十分大。

```
df['TRANGE'] = ta.TRANGE(df['high'], df['low'], df['close'])
```

4.4. 使用 talib 库获取一些模式

Belt-hold

捉腰带线
两日 K 线模式，下跌趋势中，第一日阴线，
第二日开盘价为最低价，阳线，收盘价接近最高价，预示价格上涨。




```
df['CDLBELTHOLD'] = ta.CDLBELTHOLD(df['open'], df['high'], df['low'], df['close'])
```

Closing Marubozu

收盘缺影线
一日 K 线模式，以阳线为例，最低价低于开盘价，收盘价等于最高价，
预示着趋势持续。

```
df['CDLCLOSINGMARUBOZU'] = ta.CDLCLOSINGMARUBOZU(df['open'], df['high'], df['low'], df['close'])
```

4.5. 特征降维

close	high	low	open	pre_close	change	pct_chg	volume	amount	...	LOWERBAND	RSI6	RSI12	RSI24	EMA	NATR	integer	TRANGE	CDLBELTHOLD
104.3900	104.3900	99.9800	104.3000	99.9800	4.4100	4.4109	197	8.400000e+04	...	NaN	NaN	NaN	NaN	NaN	NaN	0	NaN	0
109.1300	109.1300	103.7300	109.0700	104.3900	4.7400	4.5407	28	1.600000e+04	...	NaN	NaN	NaN	NaN	NaN	NaN	0	5.4000	0
114.5500	114.5500	109.1300	113.5700	109.1300	5.4200	4.9666	32	3.100000e+04	...	NaN	NaN	NaN	NaN	NaN	NaN	0	5.4200	0
120.2500	120.2500	114.5500	120.0900	114.5500	5.7000	4.9760	15	6.000000e+03	...	NaN	NaN	NaN	NaN	NaN	NaN	0	5.7000	0
125.2700	125.2700	120.2500	125.2700	120.2500	5.0200	4.1746	100	5.300000e+04	...	99.754864	NaN	NaN	NaN	NaN	NaN	0	5.0200	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3502.9584	3511.6554	3457.2061	3474.6793	3473.0693	29.8891	0.8606	380790800	5.230000e+11	...	3339.421255	77.932831	67.823903	61.407759	3423.254038	1.188239	0	54.4493	0
3528.6767	3528.6767	3484.7151	3492.1912	3502.9584	25.7183	0.7342	407995934	5.680000e+11	...	3348.781056	81.207047	70.496011	63.127443	3442.421795	1.184311	0	43.9616	0
3550.8767	3556.8022	3513.1262	3530.9072	3528.6767	22.2000	0.6291	370230926	5.220000e+11	...	3398.997066	83.710605	72.635944	64.550280	3462.140869	1.180700	0	43.6760	0
3576.2046	3576.2046	3526.6174	3552.9087	3550.8767	25.3279	0.7133	405348226	5.460000e+11	...	3454.365733	86.223278	74.901631	66.107278	3482.879729	1.187641	0	49.5872	0
3570.1082	3588.8625	3544.8912	3577.6923	3576.2046	-6.0964	-0.1705	345557896	5.020000e+11	...	3491.582228	82.545545	73.307835	65.385976	3498.739451	1.191067	0	43.1713	0

```
all_features = ['close', 'high', 'low', 'open', 'pre_close', 'change', 'pct_chg', 'volume', 'amount', 'label', 'ADX', 'ADXR',
'AROONDOWN', 'AROONUP', 'AROONOSC', 'ULTOSC', 'MA5', 'MA10', 'MA20', 'MA30', 'MA60', 'K', 'D', 'DIF', 'DEM', 'HISTOGRAM',
'UPPERBAND', 'MIDDLEBAND', 'LOWERBAND', 'RSI6', 'RSI12', 'RSI24', 'EMA', 'NATR', 'integer', 'TRANGE', 'CDLBELTHOLD',
'CDLCLOSINGMARUBOZU']
```

目前特征维数过高，需要进行降维，这一步在后续模型中通过随机森林算法得到特征重要度来进行筛选。

五. 模型算法设计

由于预测的问题比较特殊，因此使用多个模型分别训练，对训练预测结果进行分析，取最佳的几个模型进行模型融合，得到一个最终的模型。

5.1. 数据集划分

由于要对近期的数据进行预测，因此训练集最好不要用太久远的，因此选择与近期数据比较相邻的数据。(验证集只在 lightGBM 使用，最后并未采用该模型)

```
# 数据集划分
train_start_time = '2016-01-01'
train_end_time = '2020-11-30'
valid_start_time = '2021-01-01'
valid_end_time = '2021-01-07'
test_start_time = '2020-12-01'
test_end_time = '2020-12-31'
```

5.2. 多种模型训练

1) LightGBM

建模如下:

```
# lightGBM
params = {'num_leaves': 210, #结果对最终效果影响较大，越大值越好，太大会出现过拟合
          'min_data_in_leaf': 30,
          'objective': 'binary', #定义的目标函数
          'max_depth': 8,
          'learning_rate': 0.03,
          "min_sum_hessian_in_leaf": 6,
          "boosting": "gbdt",
          "feature_fraction": 0.9,  #提取的特征比率
          "bagging_freq": 1,
          "bagging_fraction": 0.8,
          "bagging_seed": 11,
          "lambda_l1": 0.1,             #l1 正则
        #   'lambda_l2': 0.001,     #l2 正则
          "verbosity": -1,
          "nthread": -1,                #线程数量，-1 表示全部线程，线程越多，运行的速度越快
          'metric': {'binary_logloss', 'auc'},  ##评价函数选择
          "random_state": 2019, #随机数种子，可以防止每次运行的结果不一致
          # 'device': 'gpu' ##如果安装的事gpu 版本的 lightgbm,可以加快运算
          }

dtrain = lgb.Dataset(x_train, label=y_train)
dvalid = lgb.Dataset(x_valid, label=y_valid)
num_boost_round = 1000
early_stopping_rounds = 100
verbose_eval = 20


lgbmModel = lgb.train(
```

```
            params,
            dtrain,
            num_boost_round=num_boost_round,
            valid_sets=[dtrain, dvalid],
            # valid_names=["train", "valid"],
            early_stopping_rounds=early_stopping_rounds,
            verbose_eval=verbose_eval,
        )
```

2) logistic

建模如下：

```
# logistic
from sklearn.linear_model import LogisticRegression
lrModel = LogisticRegression(penalty='l2')
lrModel.fit(x_train, y_train)
pred_lrModel = lrModel.predict(x_test)

model_report(y_test, pred_lrModel)
```

3) GBDT

建模如下：

```
# GBDT
from sklearn.ensemble import GradientBoostingClassifier
gbdtModel = GradientBoostingClassifier(n_estimators=200)
gbdtModel.fit(x_train, y_train)
pred_gbdtModel = gbdtModel.predict(x_test)

model_report(y_test, pred_gbdtModel, 'GBDT')
```

4) SVC

建模如下：

```
# SVC
from sklearn.svm import SVC
```

```
svmModel = SVC(kernel='rbf', probability=True)
svmModel.fit(x_train, y_train)
pred_svmModel = svmModel.predict(x_test)

model_report(y_test, pred_svmModel, 'SVC')
```

5) QDA

建模如下：

```
from sklearn.discriminant_analysis import QuadraticDiscriminan
tAnalysis
qdaModel = QuadraticDiscriminantAnalysis()
qdaModel.fit(x_train, y_train)
pred_qdaModel = qdaModel.predict(x_test)

model_report(y_test, pred_qdaModel, 'QuadraticDiscriminantAnal
ysis')
```

6) RandomForest

建模如下：

```
# RandomForest
from sklearn.ensemble import RandomForestClassifier
rfModel = RandomForestClassifier(n_estimators=10000,random_sta
te=11)
rfModel.fit(x_train, y_train)
pred_rfModel = rfModel.predict(x_test)

model_report(y_test, pred_rfModel, 'RandomForest')
```

六. 实验分析与模型效果评价

6.1. 不降维情况下对多种模型进行训练

1) LightGBM

预测结果:

```
模型lightGBM
指标报告:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.48      1.00      0.65        11

    accuracy                           0.48        23
   macro avg       0.24      0.50      0.32        23
weighted avg       0.23      0.48      0.31        23

准确率: 0.4782608695652174
```

2) logistic

预测结果:

```
模型
指标报告:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.48      1.00      0.65        11

    accuracy                           0.48        23
   macro avg       0.24      0.50      0.32        23
weighted avg       0.23      0.48      0.31        23

准确率: 0.4782608695652174
```

3) GBDT

预测结果:

```
模型GBDT
指标报告:
              precision    recall  f1-score   support

           0       0.57      0.33      0.42        12
           1       0.50      0.73      0.59        11

    accuracy                           0.52        23
   macro avg       0.54      0.53      0.51        23
weighted avg       0.54      0.52      0.50        23

准确率: 0.5217391304347826
```

4) SVC

预测结果:

```
模型SVC
指标报告:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.48      1.00      0.65        11

    accuracy                           0.48        23
   macro avg       0.24      0.50      0.32        23
weighted avg       0.23      0.48      0.31        23

准确率: 0.4782608695652174
```

5) QDA

```
模型QuadraticDiscriminantAnalysis
指标报告:
              precision    recall  f1-score   support

           0       0.69      0.75      0.72        12
           1       0.70      0.64      0.67        11

    accuracy                           0.70        23
   macro avg       0.70      0.69      0.69        23
weighted avg       0.70      0.70      0.69        23

准确率: 0.6956521739130435
```

6) RandomForest

预测结果：

```
模型RandomForest
指标报告：
              precision    recall  f1-score   support

           0       0.70      0.58      0.64        12
           1       0.62      0.73      0.67        11

    accuracy                           0.65        23
   macro avg       0.66      0.66      0.65        23
weighted avg       0.66      0.65      0.65        23

准确率: 0.6521739130434783
```

6.2. 不降维情况下多模型结果汇总分析

将以上结果进行汇总分析得到：

		PRECISION	RECALL	F1-SCORE	SUPPORT
LIGHTGBM	0	0.00	0.00	0.00	12
	1	0.48	1.00	0.65	11
	accuracy			**0.48**	23
	macro avg	0.24	0.50	0.32	23
	weight avg	0.23	0.48	0.31	23
LOGISTIC	0	0.00	0.00	0.00	12
	1	0.48	1.00	0.65	11
	accuracy			**0.48**	23
	macro avg	0.24	0.50	0.32	23
	weight avg	0.23	0.48	0.31	23
GBDT	0	0.57	0.33	0.42	12
	1	0.50	0.73	0.59	11
	accuracy			**0.52**	23
	macro avg	0.54	0.53	0.51	23
	weight avg	0.54	0.52	0.50	23
SVC	0	0.00	0.00	0.00	12
	1	0.48	1.00	0.65	11
	accuracy			**0.48**	23
	macro avg	0.24	0.50	0.32	23
	weight avg	0.23	0.48	0.31	23
QDA	0	0.69	0.75	0.72	12
	1	0.70	0.64	0.67	11

	accuracy			**0.70**	23
	macro avg	0.70	0.69	0.69	23
	weight avg	0.70	0.70	0.69	23
RANDOMFOREST	0	0.70	0.58	0.64	12
	1	0.62	0.73	0.67	11
	accuracy			**0.65**	23
	macro avg	0.66	0.66	0.65	23
	weight avg	0.66	0.65	0.65	23

容易发现，二次判别分析和随机森林算法有着相较于其他算法高出很多的准确率，分别达到了 0.70 和 0.65，GBDT 的准确率达到了 0.52。

考虑到特征维数过多对随机森林影响可能比较小，对其他算法影响比较大，所以这个结果也是情理之中的。

6.3. 随机森林算法

作为新兴起的、高度灵活的一种机器学习算法，随机森林（Random Forest，简称 RF）拥有广泛的应用前景，从市场营销到医疗保健保险，既可以用来做市场营销模拟的建模，统计客户来源，保留和流失，也可用来预测疾病的风险和病患者的易感性。最初，我是在参加校外竞赛时接触到随机森林算法的。最近几年的国内外大赛，包括 2013 年百度校园电影推荐系统大赛、2014 年阿里巴巴天池大数据竞赛以及 Kaggle 数据科学竞赛，参赛者对随机森林的使用占有相当高的比例。此外，据我的个人了解来看，一大部分成功进入答辩的队伍也都选择了 Random Forest 或者 GBDT 算法。所以可以看出，Random Forest 在准确率方面还是相当有优势的。

该算法有这样一些特点：

- 在当前所有算法中，具有极好的准确率/It is unexcelled in accuracy among current algorithms;
- 能够有效地运行在大数据集上/It runs efficiently on large data bases;
- 能够处理具有高维特征的输入样本，而且不需要降维/It can handle thousands of input variables without variable deletion;
- 能够评估各个特征在分类问题上的重要性/It gives estimates of what variables are important in the classification;
- 在生成过程中，能够获取到内部生成误差的一种无偏估计/It generates an internal unbiased estimate of the generalization error as the forest building progresses;
- 对于缺省值问题也能够获得很好得结果/It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing

6.4. 随机森林算法得到各特征重要度

随机森林算法在训练完成后，可以得到各个特征的重要度，我们可以对特征重要度进行排序查看，以选择重要的特征。

```
# 特征重要性
```

```
importances = rfModel.feature_importances_
features_importances = [[feature, importance] for feature, importance in zip(features, importances)]
features_importances_df = pd.DataFrame(features_importances, columns=['feature', 'importance'])
features_importances_df.sort_values(by='importance', ascending=False)
```

特征重要度排行如下:

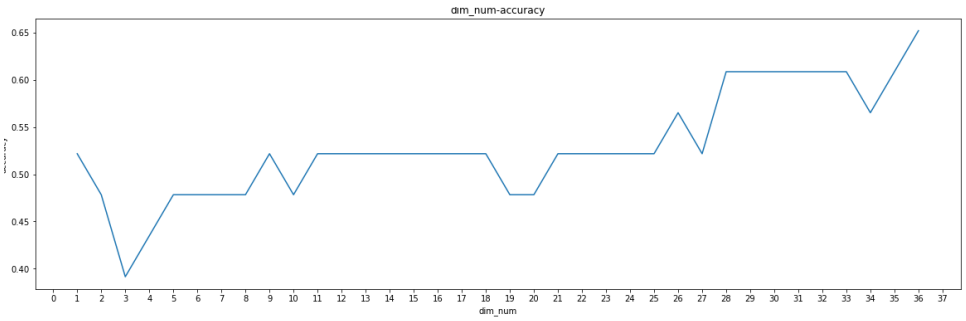
	feature	importance
21	D	0.041750
32	NATR	0.039659
14	ULTOSC	0.039347
34	TRANGE	0.039327
6	pct_chg	0.037863
20	K	0.037628
5	change	0.036927
7	volume	0.036587
9	ADX	0.034763
10	ADXR	0.034218
8	amount	0.033711
28	RSI6	0.033300
23	DEM	0.032857
24	HISTOGRAM	0.031755
29	RSI12	0.031016
30	RSI24	0.030884
22	DIF	0.030024
19	MA60	0.026982
18	MA30	0.025676
27	LOWERBAND	0.025033
17	MA20	0.024025
4	pre_close	0.023810
0	close	0.023449
3	open	0.023179
25	UPPERBAND	0.023103
2	low	0.022696
16	MA10	0.022645
1	high	0.022492
31	EMA	0.021556
26	MIDDLEBAND	0.021413
15	MA5	0.021400

13	AROONOSC	0.020805
12	AROONUP	0.018496
11	AROONDOWN	0.017766
36	CDLCLOSINGMARUBOZU	0.006191
35	CDLBELTHOLD	0.004438
33	CDL3OUTSIDE	0.003228

可以看到，我们总共有 36 维的特征，其中 D、NATR、ULTOSC 等的重要度十分高，而例如 CDLCLOSINGMARUBOZU、CDLBELTHOLD、CDL3OUTSIDE 等特征的重要度十分低。因此，我们可以对特征进行筛选，得到更加合适的特征集合。

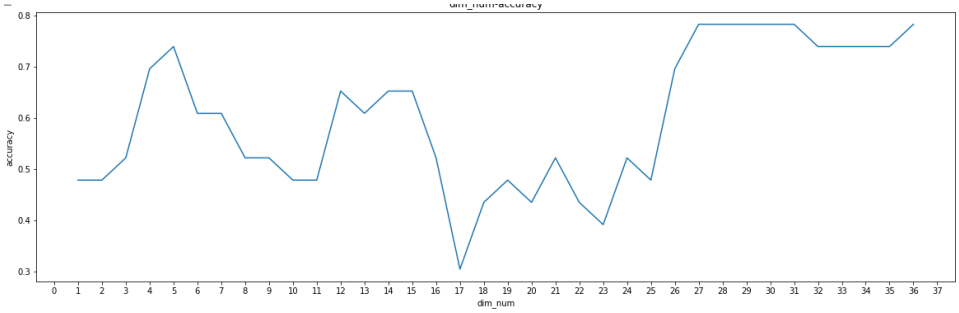
6.5. 依次减少不重要特征进行多次训练

设总特征数为 M，依次抛弃 N(N<M)个重要度最低的特征，进行多次训练，得到随机森林算法准确度变化如下：

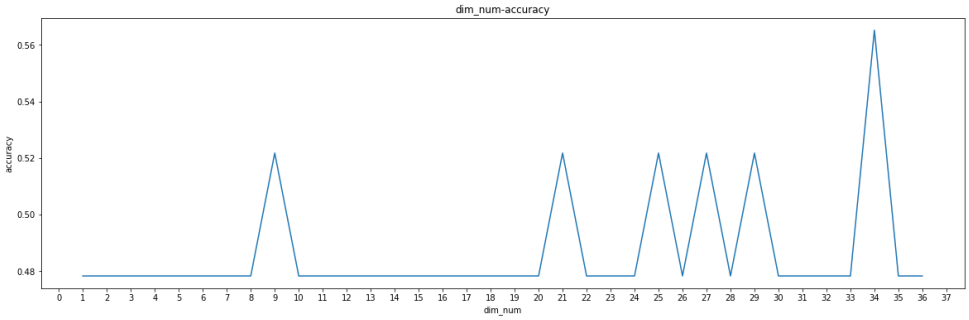


由图可见，对于随机森林算法而言，特征维度还是越大越好。

QDA:

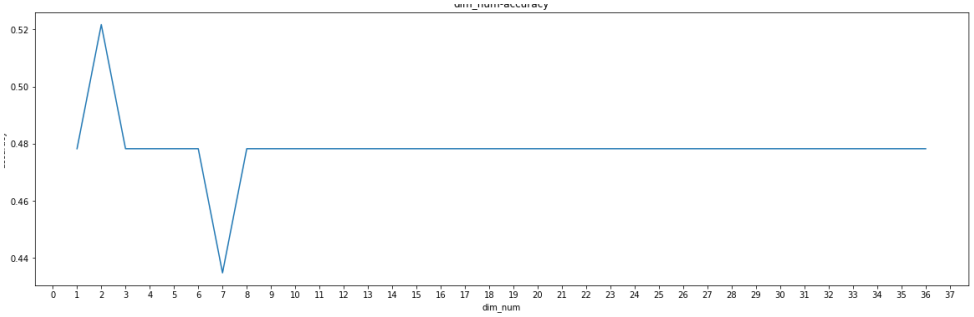


lightGBM:



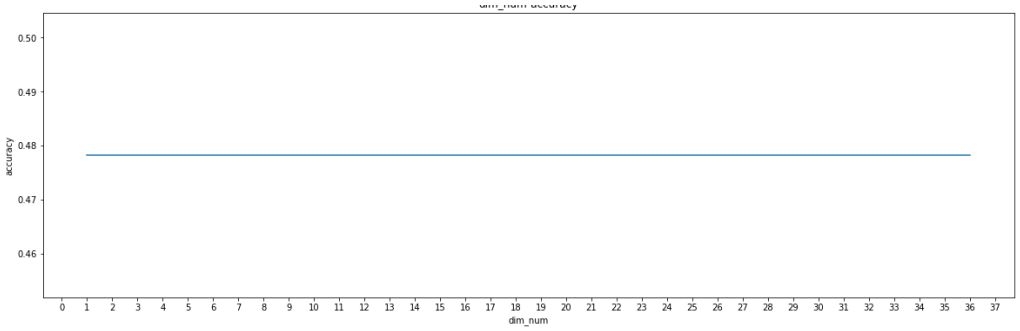
对于 lightGBM 算法，由图可以知道，该模型在特征维数为 34 维的情况下表现良好。

logistic：

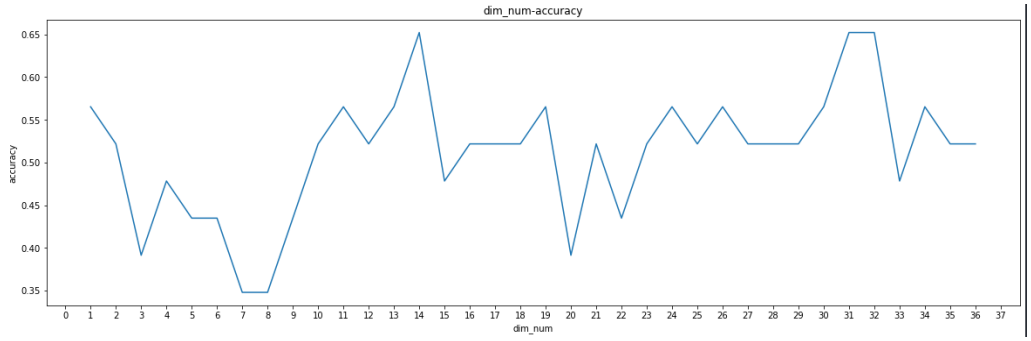


维数对该模型影响不大

SVC：



GDBT：



该模型在 31 维或 32 维的情况下预测结果较好。

6.6. 选择融合模型

由上面分析可以得到，准确度表现最好的 3 个模型依次是 QDA、RandomForest、GDBT，因此下面考虑对这三个模型进行融合。

1) 硬融合

```
from sklearn.ensemble import VotingClassifier
clf1 = QuadraticDiscriminantAnalysis()
clf2 = RandomForestClassifier(n_estimators=10000,random_state=
11)
clf3 = GradientBoostingClassifier(n_estimators=200)

eclf = VotingClassifier(estimators=[('QDA', clf1), ('RF', clf2
), ('GBDT', clf3)], voting='hard')
eclf.fit(x_train, y_train)
pred_eclf = eclf.predict(x_test)

model_report(y_test, pred_eclf, 'eclf')
```

```
模型eclf
指标报告:
              precision    recall  f1-score   support

           0       0.70      0.58      0.64        12
           1       0.62      0.73      0.67        11

    accuracy                           0.65        23
   macro avg       0.66      0.66      0.65        23
weighted avg       0.66      0.65      0.65        23

准确率: 0.6521739130434783
```

准确率达到 65%

2) 软融合

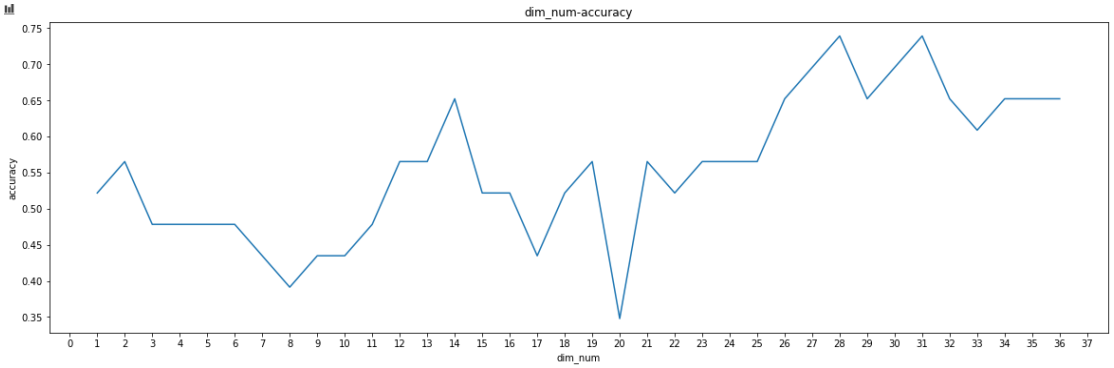
```
from sklearn.ensemble import VotingClassifier
clf1 = QuadraticDiscriminantAnalysis()
clf2 = RandomForestClassifier(n_estimators=10000,random_state=
11)
clf3 = GradientBoostingClassifier(n_estimators=200)

eclf = VotingClassifier(estimators=[('QDA', clf1), ('RF', clf2
), ('GBDT', clf3)], voting='soft')
```

```
eclf.fit(x_train, y_train)
pred_eclf = eclf.predict(x_test)

model_report(y_test, pred_eclf, 'soft')
```

```
指标报告:
              precision    recall  f1-score   support

           0       0.70      0.58      0.64        12
           1       0.62      0.73      0.67        11

    accuracy                           0.65        23
   macro avg       0.66      0.66      0.65        23
weighted avg       0.66      0.65      0.65        23

准确率: 0.6521739130434783
```

准确率还是 65%

6.7. 降维对融合模型的影响



图中可以看到， 8 号、17 号、20 号、29 号以及最后几个特征的引入让模型准确度显著下降，可以尝试将这几个特征删除，再进行训练。

```
模型eclf
指标报告:
              precision    recall  f1-score   support

           0       0.78      0.58      0.67        12
           1       0.64      0.82      0.72        11

    accuracy                           0.70        23
   macro avg       0.71      0.70      0.69        23
weighted avg       0.71      0.70      0.69        23

准确率: 0.6956521739130435
```

一个月预测的准确率可以达到 69%。

七. 预测结果分析

上面已经对 2020 年 12 月的股市进行了预测与分析，准确率能达到 69%，如下图所示：

```
模型eclf
指标报告:
              precision    recall  f1-score   support

           0       0.78      0.58      0.67        12
           1       0.64      0.82      0.72        11

    accuracy                           0.70        23
   macro avg       0.71      0.70      0.69        23
weighted avg       0.71      0.70      0.69        23

准确率: 0.6956521739130435
```

我们用该融合模型再对 2021 年的 1 月的前几天(1 号-7 号)进行预测：

```
    # 预测
    pred_eclf = eclf.predict(x_test)
    model_report(y_test, pred_eclf, 'eclf')
    print(y_test)
    print(pred_eclf)
```

```
模型eclf
指标报告:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.67      0.80         3

    accuracy                           0.75         4
   macro avg       0.75      0.83      0.73         4
weighted avg       0.88      0.75      0.77         4

准确率: 0.75
[1 1 1 0]
[1 1 0 0]
```

一共 4 天，其中 3 天预测正确。可见该融合模型确实具有一定的预测能力，更未来的预测效果，我们可以等一段时间见分晓。

值得一提的是，QDA 模型的对这几天的预测正确率为 100%。所以，可能 QDA 模型比融合模型可能还是要强。

```
模型QuadraticDiscriminantAnalysis
指标报告:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         3

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4

准确率: 1.0
[1 1 1 0]
[1 1 1 0]
```

八. 大作业代码下载网址

github 地址：https://github.com/Ethan-yh/stock-prediction
百度网盘地址：https://pan.baidu.com/s/1Og86zIfKHbA1rVH8Wqzw-Q
提取码：ewhs

九. 学习体会

9.1. 总体体会

本次试验进行了一个数据挖掘的实验，通过所学知识对实际问题进行分析和解决，整个过程虽然十分艰辛，但是却学习到了很多。

首先，股票预测这个任务在目前看来是基本没有人能有一个准确度非常高的模型的，尤其是对近期的预测(详情请看后文——随机漫步假说)。虽然在一开始就知道最后的实验结果准确率可能只有 50%左右，但是还是认认真真地将数据挖掘的各方面知识应用在问题解决上。

在尝试多个模型得到结果都比较一般的情况下，随机森林算法的尝试却让结果有了改观。随机森林算法也是在近些年的数据挖掘比赛中常用到的算法，它有着非常高的准确度，尤其是在分类问题上。

与线性判别分析类似，二次判别分析是另外一种线性判别分析算法，二者拥有类似的算法特征，区别仅在于：当不同分类样本的协方差矩阵相同时，使用线性判别分析；当不同分类样本的协方差矩阵不同时，则应该使用二次判别。

这次的股票预测，最成功的模型就是 QDA 模型和随机森林算法。尤其是 QDA，无论是 12 月的预测还是 1 月初的预测，结果都十分不错。

最后尝试用融合模型，使得模型更加具有鲁棒性，得到一个月的预测结果还算不错，并且一月的前几天预测正确率也还比较可观。

9.2. 对股票预测的想法

虽然这次实验对股票预测进行了完整的建模，但是还是想说，股票预测这个事情，尤其是短期预测，它依旧是一个十分困难的问题。预测股票的长远走向可能还有依可行，但是预测明日涨跌确实比较困难。

我这次所建立的模型也只在最近的数据上表现不错，但是未来，我相信该模型肯定不再适用，或者说要用更新的数据进行更新训练。

随机漫步理论指出，股票市场内有成千上万的精明人士，并非全部都是愚昧的人。每一个人都懂得分析，而且资料流入市场全部都是公开的，所有人都可以知道，并无甚么秘密可言。既然你也知，我也知，股票现在的价格就已经反映了供求关系。或者本身价值不会太远。所谓内在价值的衡量方法就是看每股资产值、市盈率、派息率等基本因素来决定。这些因素亦非什么大秘密。每一个人打开报章或杂志都可以找到这些资料。如果一只股票资产值十元，断不会在市场变到值一百元或者一元。市场不会有人出一百元买入这只股票或以一元沽出。现时股票的市价根本已经代表了千万醒目人士的看法，构成了一个合理价位。市价会围绕着内在价值而上下波动。这些波动却是随意而没有任何轨迹可寻。此一说法的真正涵义是，没有什么单方能够战胜股市，股价早就反映一切了，而且股价不会有系统地变动。天真的选股方法，如对着报纸的股票版丢掷飞镖，也照样可以选出战胜市场的投资组合。

由此可见，以目前认识看来，股票市场依旧是一个难以预测的市场。机器学习可以辅助我们去分析市场，进行策略制定与回测。但是要想十分准确地去预测股票，目前看来机器学习与数据挖掘技术还未发展到此地步。不过，技术和观念总是向前发展的，如今我们看似不可能的事情，随着科学的推进与发展，未来的人工智能很有可能在这一领域开拓出新的理论与应用，我们作为这个过程的参与者和建设者，将拭目以待!