西南财经大学
SOUTHWESTERN UNIVERSITY OF FINANCE AND ECONOMICS

金融智能
期末课程作业

作业名称：机器学习方法对股票价格的预测分析

学生姓名：李楷

所在学院：经济信息工程学院

专　　业：金融学

学　　号：219020204210

成　　绩：

2020 年　5　月

目录

一、 研究背景及目的

随着金融科技的发展，已经有许多机器学习方法被个人投资者和机构投资者用于股票市场的预测和优化。本文使用了几种机器学习方法和一种用于经典时间序列的深度学习模型：LSTM，对中国平安（sh.601318）自发行上市到现在的股价走势进行预测和分析。本文旨在通过对比模型之间的预测效果，探讨机器学习模型于股票市场中的效果和可行性。

本文的研究结构大体为：

1. 数据收集以及预处理
2. 特征提取
3. 数据标准化
4. 模型使用
5. 结果分析
6. 模型比较

二、 数据处理

（一）数据获取

数据源来自于开源的 python 股票数据库 API 接口，网址为 http://baostock.com。主要涵盖几方面的数据：

1. 收盘价的各类价格，如开盘、收盘、最高、最低价格，因为时间序列明显是自相关的，这些信息在计量中也证实了并不是取其一即可。
2. 财务指标,基本面指标：如市盈率、市净率等。
3. 其他经济基本面指标：暂时只选取了银行拆借利率、股票沪深 300 指数表现。

在查询相关文献时，发现如上海银行间七天同业拆放利率 shibor7，由于其市场化程度最高，参与者最多，交易量最大（李亚鸽——上海银行间同业拆放利率与股票价格短期动态关系分析——基于 VAR 模型），因此许多论文会选取其最

股价波动性作计量分析，类似可以推广到其他许多指标。

本文借鉴这种思想，虽然没有直接更多参考文献，但使用了逐步回归方法筛选相关变量代替根据手工选取变量的办法，筛选变量并构造数据特征。数据列名如下所示：不包括日期一共有 33 列。例如，同业拆解利率就从周到年都有，但经过逐步回归后，自动就筛选保留了一周的同业拆解利率，而其他就被抛弃了。

['code', 'open', 'high', 'low', 'close', 'preclose', 'volume', 'amount',
'adjustflag', 'turn', 'tradestatus', 'pctChg', 'peTTM', 'pbMRQ',
'psTTM', 'pcfNcfTTM', 'isST', 'shiborON', 'shibor1W', 'shibor2W',
'shibor1M', 'shibor3M', 'shibor6M', 'shibor9M', 'shibor1Y',
'sh.000300open', 'sh.000300high', 'sh.000300low',
'sh.000300close', 'sh.000300preclose', 'sh.000300volume',
'sh.000300amount', 'sh.000300pctChg']

（二）数据变量说明

原始数据 http://baostock.com，其中本研究用到的第一类主要变量如下：

open	开盘价	
high	最高价	
low	最低价	
close	收盘价	
preclose	前收盘价	
volume	成交量 （累计 单位：股）	

amount	成交额（单位：人民币元）	
adjustflag	复权状态	(1：后复权， 2：前复权，3：不复权）
turn	换手率	[指定交易日的成交量(股)/指定交易日的股票的流通股总股数(股)]*100%
tradestatus	交易状态	(1：正常交易 0：停牌）
pctChg	涨跌幅（百分比）	日涨跌幅=[(指定交易日的收盘价-指定交易日前收盘价)/指定交易日前收盘价]*100%
peTTM	滚动市盈率	(指定交易日的股票收盘价/指定交易日的每股盈余TTM)=(指定交易日的股票收盘价*截至当日公司总股本)/归属母公司股东净利润 TTM
pbMRQ	市净率	(指定交易日的股票收盘价/指定交易日的每股净资产)=总市值/(最近披露的归属母公司股东的权益-其他权益工具)
psTTM	滚动市销率	(指定交易日的股票收盘价/指定交易日的每股销售额)=(指定交易日的股票收盘价*截至当日公司总股本)/营业总收入 TTM
pcfNcfTTM	滚动市现率	(指定交易日的股票收盘价/指定交易日的每股现金流 TTM)=(指定交易日的股票收盘价*截至当日公司总股本)/现金以及现金等价物净增加额 TTM

isST	是否 ST 股	1 是，0 否
shibor1W	1 周拆放利率	

第二第三类其他列名的含义由于篇幅，不再赘述。

（三）调用功能库说明

本文大致使用了以下的功能库：

```
import pandas as pd
import matplotlib.pyplot as plt
import patsy
import numpy as np
import seaborn as sns
import datetime
import patsy
import scipy.stats as scs
from pylab import mpl
%matplotlib inline
import tensorflow as tf
from matplotlib.pyplot import MultipleLocator
mpl.rcParams['font.sans-serif'] = ['SimHei']
from matplotlib.pyplot import MultipleLocator
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from tensorflow.keras.models import Sequential#from keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense
```

from tensorflow.keras.layers import LSTM

from tensorflow.keras.layers import Dropout

from sklearn.metrics import roc_auc_score,f1_score,roc_curve 功能库的类型大致囊括了从数据读取、处理、IO 模块，到计量统计功能、绘图，再到模型构建、训练，评价模型效果几大类。

（四）数据预处理

先是使用了 pd.concat（）合并第一、第二、第三类数据。

然后对空缺值填充并处理。由于周末和节假日等原因没有交易价格，需要 resample 并且用最后一个交易日的价格填充空缺值。

```
result3=pd.DataFrame(result3)

result3.index=result3.date

result3.index= pd.to_datetime(result3.index,format = '%Y-%m-%d')
result3= result3.resample('D').asfreq()


result3.pop('date')
result3=result3.ffill()
```

图 1

最终获得的初始数据集如下所示：

date	code	open	high	low	close	preclose	volume	amount	adjustflag	turn	...	shibor9M	shibor1Y	shopen	shhig
2007-03-01	sh.601318	50.0000	50.9700	45.8000	46.7900	33.8000	197763351	9500302911.0000	3	34.393626	...	2.932600	3.002800	2550.2600	2550.328
2007-03-02	sh.601318	47.0000	47.4400	45.8800	46.4100	46.7900	42504832	1981445189.0000	3	7.392145	...	2.932500	3.002800	2468.6720	2523.856
2007-03-03	sh.601318	47.0000	47.4400	45.8800	46.4100	46.7900	42504832	1981445189.0000	3	7.392145	...	2.932500	3.002800	2468.6720	2523.856
2007-03-04	sh.601318	47.0000	47.4400	45.8800	46.4100	46.7900	42504832	1981445189.0000	3	7.392145	...	2.932500	3.002800	2468.6720	2523.856
2007-03-05	sh.601318	45.9800	46.1700	43.5300	44.3800	46.4100	41919674	1880502808.0000	3	7.290378	...	2.932400	3.002800	2503.8160	2541.818
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2020-04-16	sh.601318	70.5000	70.7100	70.1600	70.3800	70.8500	28294260	1992746679.0000	3	0.261200	...	1.603000	1.694000	3777.7986	3807.356
2020-04-17	sh.601318	71.1000	73.3200	70.7900	72.4000	70.3800	90367948	6519197379.0000	3	0.834200	...	1.594000	1.683000	3831.9232	3863.457
2020-04-18	sh.601318	71.1000	73.3200	70.7900	72.4000	70.3800	90367948	6519197379.0000	3	0.834200	...	1.594000	1.683000	3831.9232	3863.457
2020-04-19	sh.601318	71.1000	73.3200	70.7900	72.4000	70.3800	90367948	6519197379.0000	3	0.834200	...	1.594000	1.683000	3831.9232	3863.457
2020-04-20	sh.601318	72.8900	73.8000	72.1000	73.5000	72.4000	65419398	4785645886.0000	3	0.603900	...	1.578000	1.674000	3747.8408	3853.805

图 2

标准化数据的部分在后续章节展示。

三、 数据描述性分析

（一）收盘价的价格走势

首先对中国平安的收盘价格画图。

图 3

具有波动性十分剧烈的时段，也具有振荡的时段，尤其是有 2017 年的转增股，并没有把这一信息放入在数据集中，预测任务具有挑战性。

（二）收益率分布

然后对中国平安的每日涨跌幅画出分布图，可以发现该股票的收益率变化具有尖峰肥尾的特征，其中极端值更多分布在右侧，与预期中左偏的巨额亏损不同。最大涨跌幅为+38%（在上市日）。

```
fig, ax= plt.subplots(figsize=(15,5))
ax.set_title('中国平安收益率分布图')
sns.distplot((data['pctChg']/100), kde=True,color='purple')
s=(data['pctChg']/100)
print("偏度:",s.skew())##
print("峰度:",s.kurt())##

```

```
偏度: 0.9517078203739542
峰度: 16.528443545937158
```

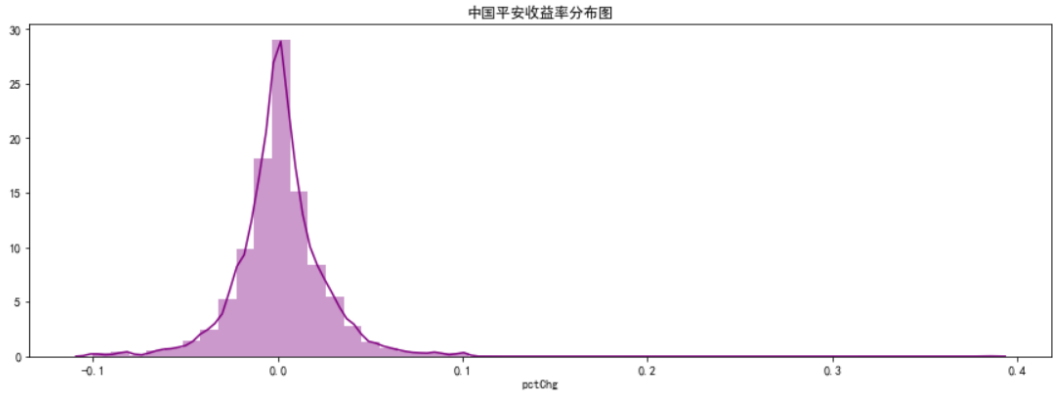


图 4

（三）交易状态分布

接着统计 tradestatus 是否停牌的状态，统计来说，该股票是上市状态比较稳定的股票，不会出现严重缺失数据的情况影响分析。

```
(data['tradestatus']).value_counts()
# 1)是否停牌
data2=data
data2['tradestatus'].replace(1, '正常交易',inplace = True) #替换
data2['tradestatus'].replace(0, '停牌',inplace = True)
#print(data2['tradestatus'])
#做图
fig, tradestatus = plt.subplots(figsize=(10,5)) #图大小设置


sns.set_palette('pastel') #设置图的格式及颜色为Pastel
sns.countplot(x=data2['tradestatus'], data=data2) #做频率图

tradestatus.set_title('[2] 交易状态统计') #改表头
tradestatus.set_ylabel('Count') #改纵坐标
tradestatus.set_xlabel('交易状态') #改横坐标

tradestatus.spines['right'].set_visible(False) # 去除右边的边框
tradestatus.spines['top'].set_visible(False)

#设置百分比
totals = []

for i in tradestatus.patches:
    totals.append(i.get_height())


total = sum(totals)

for i in tradestatus.patches:
    tradestatus.text(i.get_x() +.25,i.get_height()+20,
            str(round((i.get_height()/total)*100,2))+"%",fontsize = 12,
            color = "black")
plt.show()
```

图 5

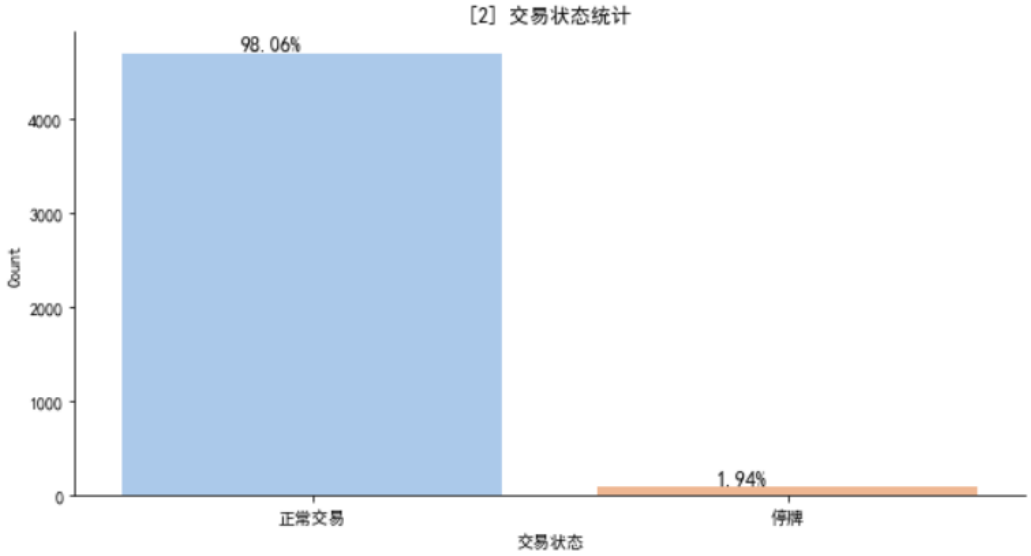


图 6

（四）相关信息分析

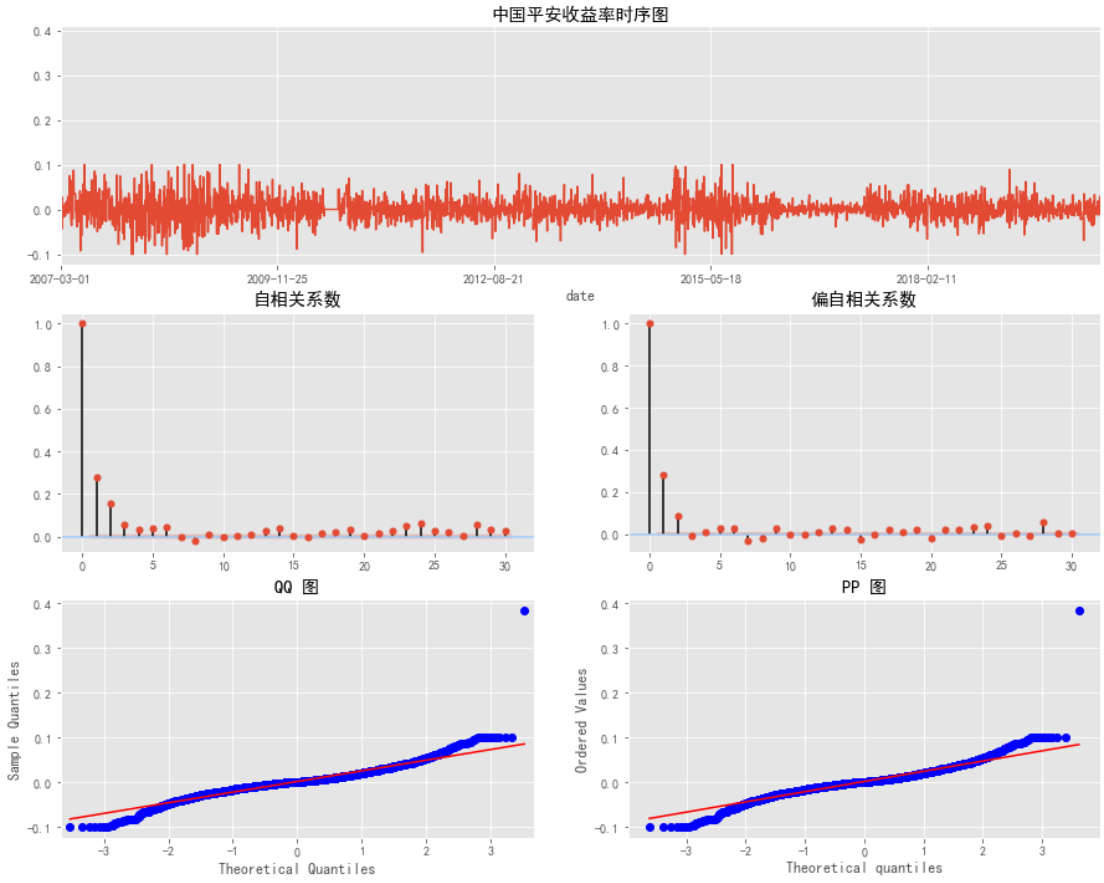


图 7

自相关性在几天内比较强，在长期比较弱；收益率的条件方差（Conditional Variance）随着时间而变化，即存在条件异方差的特征。

收益率序列的波动具有持续性，即存在波动集聚（Volatility Clustering）的现象。比如 2007-2008、2015-2016 具有较大的波动性。

QQ 图显示，收益率并不服从正态分布，极端值较多，具有厚尾的现象。

（五）划分数据集

然后划分测试集以及训练集，并做图示。时间序列划分特点是前面一段时间是训练集，其他为测试集，因此不能使用交叉划分的方法。

```
split_date = '2013-6-01'
data=data2
fig,ax1= plt.subplots(figsize=(20,8))
#fig,ax2 = plt.subplots(figsize=(9,4))
# ax1.set_xticks([datetime.date(i,12,1) for i in range(2007,2021)])
# ax1.set_xticklabels([datetime.date(i,12,1).strftime('%Y%m') for i in range(2007,2021)])

ax1.plot(data[data.index < split_date].index.astype(datetime.datetime),
         data[data.index < split_date]['close'],
         color='#CD423C', label='训练集')
ax1.plot(data[data.index >= split_date].index.astype(datetime.datetime),
         data[data.index >= split_date]['close'],
         color='#71C621', label='测试集')
```

图 8

图 9

从分布来说，尽量使得训练集和测试集都有选取到高低市值区间。

四、 构造模型特征

在计量回归之前，可以先对数据标准化，也可以不标准化，因为线性变化对计量的回归系数显著性无影响。通过逐步回归找出的与收盘价有显著关系的变量。而数据在深度学习、机器学习前必须标准化，并且必须剔除不相干、共线性的变量再重新标准化，否则在计量前的标准化的线性变换，筛选后没有重新标准化，会导致训练数据出问题，本人尝试过这种做法后发现会导致无法收敛，训练 loss 输出为 NAN，调整后就可以正常训练了。

尽管如此，本文还是采取了计量分析前标准化一次，机器学习前再对选取的变量重新标准化，一共标准化两次的做法。

标准化的方法如下所示，对训练集提取方差和标准差，然后运用到测试集中，然后拼接成 stand_data 标准化后的数据集。

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
ss = StandardScaler()

stand_data_train=training_set.iloc[:,1:]
stand_data_test=test_set.iloc[:,1:]

stand_ed_data_train = ss.fit_transform(stand_data_train)
stand_ed_data_test=ss.transform(stand_data_test)

origin_data_train = ss.inverse_transform(stand_ed_data_train)
origin_data_test = ss.inverse_transform(stand_ed_data_test)

print('data is ',stand_data_train)
print('data is ',stand_data_test)
print('trainset after standard ',stand_ed_data_train)
print('train after inverse ',origin_data_train)
print('testset after standard ',stand_ed_data_test)
print('test after inverse ',origin_data_test)
print('after standard mean and std is ',np.mean(stand_data_train), np.std(stand_data_train))

stand_ed_data_train=pd.DataFrame(stand_ed_data_train)
stand_ed_data_train.columns=data2.iloc[:,1:].columns

stand_ed_data_test=pd.DataFrame(stand_ed_data_test)
stand_ed_data_test.columns=data2.iloc[:,1:].columns

stand_data2=pd.concat([pd.DataFrame(stand_ed_data_train),pd.DataFrame(stand_ed_data_test)],ignore_index=True)

print(len(stand_ed_data_test))
print(len(stand_ed_data_train))
print(stand_data2)
```

图 10

数据标准化处理后的结果如下：

date	open	high	low	close	preclose	volume	amount	adjustflag	turn	tradestatus	...	shibor9M	shibor1Y	shopen	shhigl
2007-03-01	-0.093297	-0.091762	-0.261532	-0.250951	-0.874358	9.114494	9.754429	0.0	30.124502	0.202536	...	-0.627173	-0.690900	-0.619128	-0.65357
2007-03-02	-0.237398	-0.257288	-0.257585	-0.269220	-0.248118	0.856659	0.852377	0.0	5.819889	0.202536	...	-0.627257	-0.690900	-0.715629	-0.68450
2007-03-03	-0.237398	-0.257288	-0.257585	-0.269220	-0.248118	0.856659	0.852377	0.0	5.819889	0.202536	...	-0.627257	-0.690900	-0.715629	-0.68450
2007-03-04	-0.237398	-0.257288	-0.257585	-0.269220	-0.248118	0.856659	0.852377	0.0	5.819889	0.202536	...	-0.627257	-0.690900	-0.715629	-0.68450
2007-03-05	-0.286392	-0.316840	-0.373510	-0.366816	-0.266437	0.825536	0.732865	0.0	5.728286	0.202536	...	-0.627340	-0.690900	-0.674061	-0.66351
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2020-04-16	0.891392	0.833868	0.940135	0.883182	0.911801	0.100832	0.865758	0.0	-0.598828	0.202536	...	-1.738790	-1.831716	0.832783	0.81508
2020-04-17	0.920212	0.956254	0.971212	0.980297	0.889143	3.402386	6.224910	0.0	-0.083058	0.202536	...	-1.746314	-1.841304	0.896801	0.88062
2020-04-18	0.920212	0.956254	0.971212	0.980297	0.889143	3.402386	6.224910	0.0	-0.083058	0.202536	...	-1.746314	-1.841304	0.896801	0.88062
2020-04-19	0.920212	0.956254	0.971212	0.980297	0.889143	3.402386	6.224910	0.0	-0.083058	0.202536	...	-1.746314	-1.841304	0.896801	0.88062
2020-04-20	1.006192	0.978762	1.035834	1.033182	0.986526	2.075431	4.172448	0.0	-0.290356	0.202536	...	-1.759691	-1.849149	0.797350	0.86935

4800 rows × 32 columns

图 11

定义如下的逐步回归函数。

```
#定义向前逐步回归函数
def forward_select(data,target):
    variate=set(data.columns)  #将字段名转换成字典类型
    variate.remove(target)  #去掉因变量的字段名
    selected=[]
    current_score,best_new_score=float('inf'),float('inf')  #目前的分数和最好分数初始值都为无穷大（因为AIC越小越好）
    #循环筛选变量
    while variate:
        aic_with_variate=[]
        for candidate in variate:  #逐个遍历自变量
            formula="{}~{}".format(target,"+".join(selected+[candidate]))  #将自变量名连接起来
            aic=ols(formula=formula,data=data).fit().aic  #利用ols训练模型得出aic值
            aic_with_variate.append((aic,candidate))  #将第每一次的aic值放进空列表
        aic_with_variate.sort(reverse=True)  #降序排序aic值
        best_new_score,best_candidate=aic_with_variate.pop()  #最好的aic值等于删除列表的最后一个值，以及最好的自变量等于列表最后一个自
        if current_score>best_new_score:  #如果目前的aic值大于最好的aic值
            variate.remove(best_candidate)  #移除加进来的变量名，即第二次循环时，不考虑此自变量了
            selected.append(best_candidate)  #将此自变量作为加进模型中的自变量
            current_score=best_new_score  #最新的分数等于最好的分数
            print("aic is {},continuing!".format(current_score))  #输出最小的aic值
        else:
            print("for selection over!")
            break
    formula="{}~{}".format(target,"+".join(selected))  #最终的模型式子
    print("final formula is {}".format(formula))
    model=ols(formula=formula,data=data).fit()
    return(model)
forward_select(data=select_data,target="close")
```

图 12

当只有第一类数据的时候，逐步回归法选取的变量及构造的方程为：

final formula is

close~high+low+pctChg+preclose+open+turn+amount+psTTM+pbMRQ

在加入了第二类第三类数据后，逐步回归选取的变量有所变化，特别是增加了是否停牌 tradestatus 这一变量，接着进行实验。最终选取属性列如下：

```
aic is -17751.416453118363,continuing!
aic is -19855.577621280674,continuing!
aic is -23309.37828832586,continuing!
aic is -24144.784196409026,continuing!
aic is -24765.126195617144,continuing!
aic is -24874.36176698694,continuing!
aic is -24883.64382525115,continuing!
aic is -24917.549223338247,continuing!
aic is -24942.77372003306,continuing!
aic is -26456.73153193671,continuing!
aic is -26727.760596174267,continuing!
aic is -26954.41700968411,continuing!
aic is -27085.468769194013,continuing!
aic is -27128.395853802584,continuing!
aic is -27139.70106666755,continuing!
aic is -27143.662434511614,continuing!
aic is -27146.751462542787,continuing!
for selection over!
final formula is close~high+low+pctChg+preclose+open+shpctChg+shvolume+amount+shclose+shpreclose+shlow+shhigh+shopen+pbMRQ+volume+shamou
nt+tradestatus

<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x2b642cdc278>
```

图 13

最终表达式 final formula is

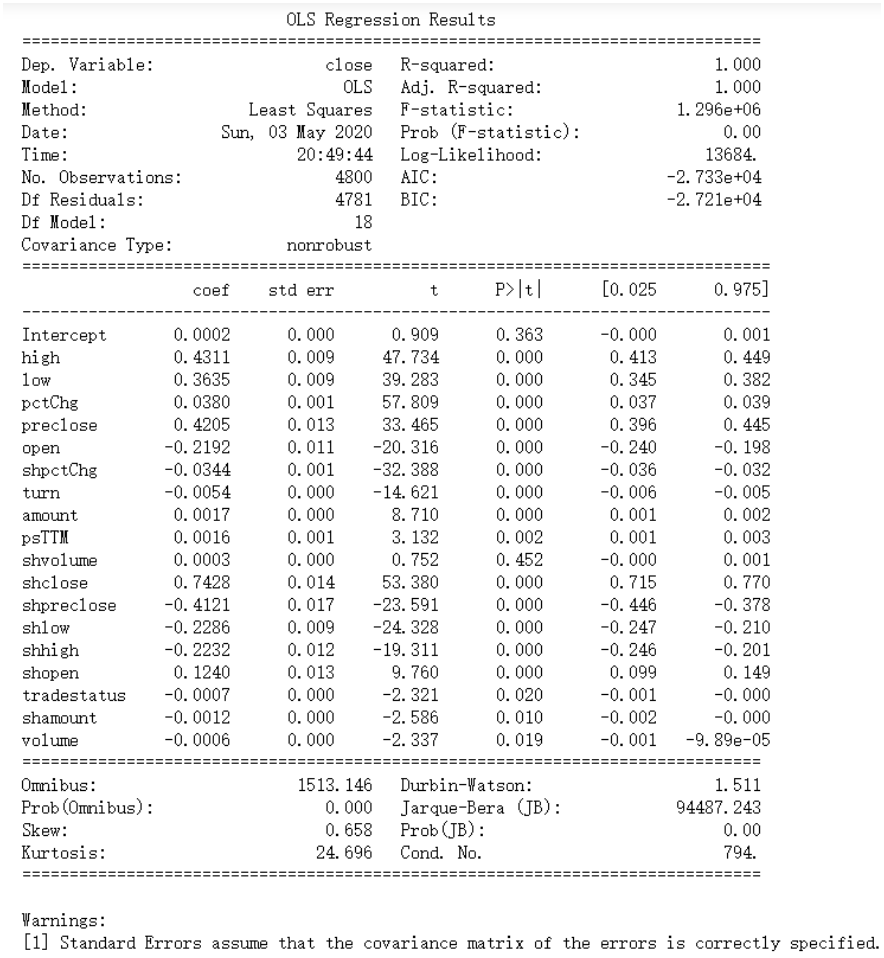
close~high+low+pctChg+preclose+open+shpctChg+turn+amount+psTTM+shvolu
me+shclose+shpreclose+shlow+shhigh+shopen+tradestatus+shamount+volume

其上即为模型的特征。

```
y, X = patsy.dmatrices('close~high+low+pctChg+preclose+open+shpctChg+shvolume+amount+shclose+shpreclose+shlow+shhigh+shopen+pbMRQ+vol
cltv = sm.OLS(y, X) #将设计矩阵传递给OLS以指定普通的最小二乘模型
res = cltv.fit()#估计模型并将结果存储在res中
print(res.summary())
```

图 14

平安股票收盘价对所选变量的 OLS 回归信息如下：



OLS Regression Results

Dep. Variable:	close	R-squared:	1.000
Model:	OLS	Adj. R-squared:	1.000
Method:	Least Squares	F-statistic:	1.296e+06
Date:	Sun, 03 May 2020	Prob (F-statistic):	0.00
Time:	20:49:44	Log-Likelihood:	13684.
No. Observations:	4800	AIC:	-2.733e+04
Df Residuals:	4781	BIC:	-2.721e+04
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0002	0.000	0.909	0.363	-0.000	0.001
high	0.4311	0.009	47.734	0.000	0.413	0.449
low	0.3635	0.009	39.283	0.000	0.345	0.382
pctChg	0.0380	0.001	57.809	0.000	0.037	0.039
preclose	0.4205	0.013	33.465	0.000	0.396	0.445
open	-0.2192	0.011	-20.316	0.000	-0.240	-0.198
shpctChg	-0.0344	0.001	-32.388	0.000	-0.036	-0.032
turn	-0.0054	0.000	-14.621	0.000	-0.006	-0.005
amount	0.0017	0.000	8.710	0.000	0.001	0.002
psTTM	0.0016	0.001	3.132	0.002	0.001	0.003
shvolume	0.0003	0.000	0.752	0.452	-0.000	0.001
shclose	0.7428	0.014	53.380	0.000	0.715	0.770
shpreclose	-0.4121	0.017	-23.591	0.000	-0.446	-0.378
shlow	-0.2286	0.009	-24.328	0.000	-0.247	-0.210
shhigh	-0.2232	0.012	-19.311	0.000	-0.246	-0.201
shopen	0.1240	0.013	9.760	0.000	0.099	0.149
tradestatus	-0.0007	0.000	-2.321	0.020	-0.001	-0.000
shamount	-0.0012	0.000	-2.586	0.010	-0.002	-0.000
volume	-0.0006	0.000	-2.337	0.019	-0.001	-9.89e-05

Omnibus:	1513.146	Durbin-Watson:	1.511
Prob(Omnibus):	0.000	Jarque-Bera (JB):	94487.243
Skew:	0.658	Prob(JB):	0.00
Kurtosis:	24.696	Cond. No.	794.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

图 15

注意到的是 shvolume 并没有显著，但是我们仍然继续保留它。总的来说，逐步向前回归法确定的变量没有比较大的问题。其中没有意义的赋权状态等恒为 3 的变量，以及上海同业拆解利率中只保留了 1 周的利率，说明如论文一致，相对其他时段的利率对于股价波动来说的确是一个更好的变量。

由于有些感兴趣的变量被去掉了，并且 MRQ 的显著性不是很高，于是将这些变量再做了一次回归。

```
y, X = patsy.dmatrices('close~ + peTTM+ pcfNcfTTM+ pbMRQ', data) #传递模型公式和相关数据以创建设计矩阵
cltv = sm.OLS(y, X) #将设计矩阵传递给OLS以指定普通的最小二乘模型
res = cltv.fit()#估计模型并将结果存储在res中
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  close   R-squared:                       0.304
Model:                            OLS   Adj. R-squared:                  0.304
Method:                 Least Squares   F-statistic:                     699.6
Date:                Sun, 03 May 2020   Prob (F-statistic):               0.00
Time:                        21:29:03   Log-Likelihood:                -5828.2
No. Observations:                4800   AIC:                         1.166e+04
Df Residuals:                    4796   BIC:                         1.169e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.3398      0.014     24.901      0.000       0.313       0.367
peTTM         -0.0398      0.018     -2.256      0.024      -0.074      -0.005
pcfNcfTTM     -0.0390      0.003    -14.032      0.000      -0.044      -0.034
pbMRQ          0.6366      0.015     43.604      0.000       0.608       0.665
==============================================================================
Omnibus:                      320.795   Durbin-Watson:                   0.007
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              409.180
Skew:                           0.621   Prob(JB):                     1.41e-89
Kurtosis:                       3.711   Cond. No.                         7.05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

图 16

可以发现，peTTM 的系数变了方向，推测是出现了多重共线性，因此逐步回归模型是在 PsTTM 与 PeTTM 中择其一，pbMRQ 在这是显著的。

总的来说从上表线性回归测试我们可以看出股票的收盘价格与对应的开盘价、最高、最低价价格数据，都显著相关，令人惊讶的是没有在向前逐步回归被去掉，说明时间序列强烈的自相关性。

五、 模型训练与预测分析

（一）LSTM 模型对股票价格走势的拟合

构造了特征后，因此接下来重要的工作就是划分数据集和测试集，构造预测价格的时间窗口，用这个时间窗口包含 t-n 至 t-1 的数据，用来预测 t 时刻的收盘价格 CLOSE，并且对比研究不同模型对不同时间时间窗口的刻画股市走势特征，预测收盘价格的能力优劣。

本文注意到有一种简单的做法是 train=data[:-1],test=[1:]，然后训练预测，这样的逐日进行回归，因为逐点预测训练的模型意义不是很大，预测曲线在较长时间上相当于蓝线的右移，类似天真预测，对于研究分析乃至指导投资的指导意义不是很大。

为了进一步探究模型在一个时期的判断能力，本文将过去 14 天的数据信息为窗口，逐步滑动，得到未来 7 天的价格预测信息，并对其取平均值，与过去 14 天的价格相比，判断是涨跌，然后根据现实情况计算出准确率。每滑动过一个交易日，按照“先进先出”的思想，就会用模型预测出来的价格值代替原窗口中序号最前向量的真实价格值。随着一个窗口内的不停的预测，使用预测价格的比例会不断升高，当预测的交易天数累计达到七天的时候，滑动窗口将使用下一周的测试集的价格数据重置窗口中累积七天的预测价格数据，然后再次启动滑动预测过程。

t 从测试集的第 14 行开始计算预测的第十五天的价格数据，给出抽象的数学模型如下：

$$\mathrm{YPredict}_t = \phi_0 + \phi_{t-1} * \mathrm{F}(\mathrm{Window}_{t-1}) + \cdots + \phi_{t-p} * \mathrm{F}(\mathrm{Window}_{t-p}) + \phi_{t-q} * \mathrm{YPredict}_{t-q} + \cdots + \phi_{t-14} * \mathrm{YPredict}_{t-14} + \epsilon$$

以第一个窗口为例：

$$\mathrm{YPredict}_0 = \mathrm{F}(\mathrm{Window}_{Y0}) = \phi 0 + \phi_1 * \mathrm{F}(\mathrm{Window}_1) + \cdots \phi_{14} * \mathrm{F}(\mathrm{Window}_{14}) + \epsilon$$

$$\epsilon \sim \mathrm{N}(\mu,\ \sigma^2)$$

显而易见的是，在每个窗口所预测的一周价格中，后预测的点将越来越多地使用到预测出来的价格信息，越来越少地使用到测试集中真实的价格信息，因此误差将被放大。

构造 LSTM 模型如下

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')] [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')]
```

```python
from tensorflow.keras.models import Sequential#from keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout  # from keras.layers  版本更新

def build_model(inputs, output_size, neurons, activ_func="linear",
                dropout=0.3, loss='mae', optimizer="adam"):#mae
    model = Sequential()

    model.add(LSTM(neurons, input_shape=(inputs.shape[1], inputs.shape[2]), return_sequences=True))
    ###
    model.add(Dropout(dropout))

#     ###
#     model.add(Dropout(dropout))
    model.add(Dense(units=output_size,activation='relu'))
    model.add(LSTM(128, input_shape=(inputs.shape[1], inputs.shape[2]),return_sequences=False))
   # model.add(LSTM(neurons))
    model.add(Dropout(dropout))
    model.add(Dense(units=output_size))
    model.add(Activation(activ_func))
#     adam = tf.keras.optimizers.Adam(decay=0.8)
    model.compile(loss=loss, optimizer=optimizer)
    return model
```

图 17

打印出 LSTM 模型的结构图：

```python
lstm_model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_11 (LSTM)               (None, 14, 256)           282624
_________________________________________________________________
dropout_22 (Dropout)         (None, 14, 256)           0
_________________________________________________________________
dense_22 (Dense)             (None, 14, 7)             1799
_________________________________________________________________
lstm_12 (LSTM)               (None, 128)               69632
_________________________________________________________________
dropout_23 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_23 (Dense)             (None, 7)                 903
_________________________________________________________________
activation_11 (Activation)   (None, 7)                 0
=================================================================
Total params: 354,958
Trainable params: 354,958
Non-trainable params: 0
_________________________________________________________________
```

图 18

训练模型，并进行预测：

```
std=np.sqrt( ss.var_[0])
mean=ss.mean_[0]
```

```
from matplotlib.pyplot import MultipleLocator

lstm_pred_prices= ((lstm_model.predict(LSTM_test_inputs)[:-pred_range][::pred_range]+1)*\
                   test_set2['close'].values[:-(window_len + pred_range)][::7].reshape(int(np.ceil((len(LSTM_test_inputs)-pred_range)/

row,col=lstm_pred_prices.shape
for i in range(row):
    for j in range(col):
        lstm_pred_prices[i][j]=(lstm_pred_prices[i][j]*std)+mean
```

图 19

训练的 MAE 随 EPOCHS 的情况如下：

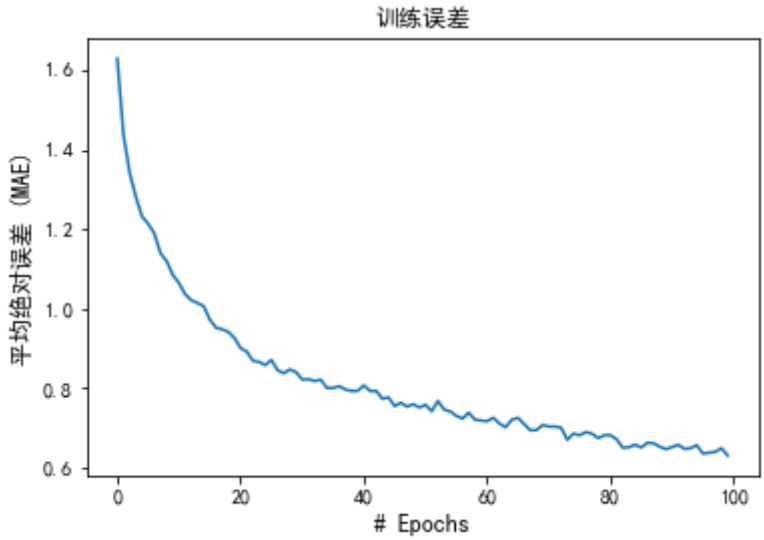


图 20

由于 LSTM 的初始权重是随机赋值的，因此为了进一步减少误差，应当使用 np.random.seed(rand_seed)以及

temp_model.save('lstm_model_randseed_%d.h5'%rand_seed)，指定 rang_seed 的区间为 0-20，保存 20 个已经训练过的 LSTM 模型并重复运行，取其预测结果均值。目的是减少随机赋初始权重带来的影响。

使用如下代码画出 LSTM 预测和实际走势图：

```
ax1.plot(stand_data2[(stand_data2.index)>= split_date].index[window_len:],
         test_set['close'][window_len:], label='实际值')
```

```
for (i,share_pred) in enumerate( lstm_pred_prices ):
    print(list(share_pred))
    print(i)
    # Only adding lines to the legend once
    if i==0:
        ax1.plot(stand_data2[(stand_data2.index)>= split_date].index[window_len:][i*pred_range:i*pred_range+pred_range],share_pred, c
    else:
        ax1.plot(stand_data2[(stand_data2.index)>= split_date].index[window_len:][i*pred_range:i*pred_range+pred_range],share_pred, c
```

图 21

使用循环单独画出间隔的 LSTM 输出窗口，分别查看不同时期的预测情况而不是将这些输出连成一条曲线。

图 22

变更一些超参数和模型结构后,数值和形态发生较大的变化：

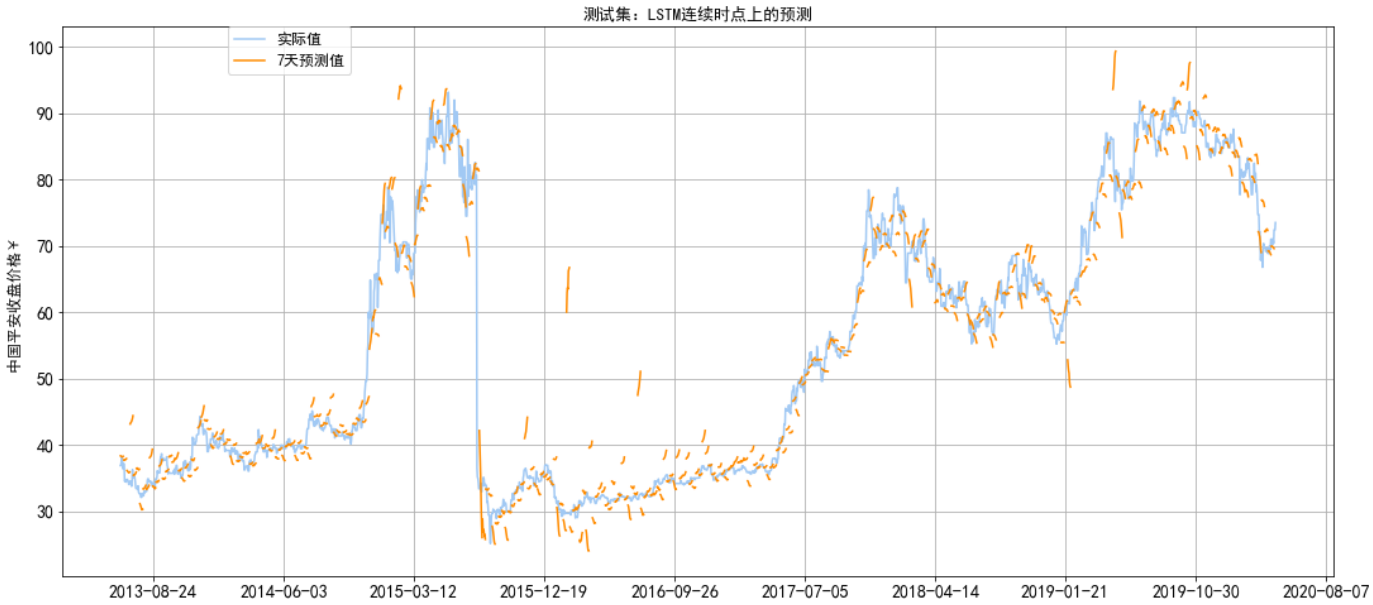


图 23

经过实验发现，不同的 LSTM 模型的结构、超参数，都会显著改变预测输出的结果，尤其是在绘制图片时，拟合曲线的效果变化比分类正确率的更大，能够直接决定离群点的数量，和偏离程度。比如图中系本文在调整参数尝试几次后，得到的较好结果，其绘制的图片中能发现有明显的离群点在 2013 年 8 月。比较神奇的是在 2015 年 6-8 月时期，由于转增配股的主要原因，股价剧烈下跌，其实输入的数据集是没有包括这方面的信息的，LSTM 输出的结果是在这一时段显示剧烈的振荡，其特点和之前高位时密集震荡的形态很像，只不过没有适应价格的变化，价格放在了半山腰上。

因此首先 LSTM 时序模型的预测效果十分依赖于超参数和模型结构的设置，其次十分依赖数据集能反映的信息状况，当发生数据集信息以外的事件时，往往不能及时反映。

接下构造一个将每个时间段内回归数值的问题转换成这一段的预测值与真实值互相比较的二分类问题，每次运行都会将回归的趋势线转换成趋势的二分类问题，思路是比较三类数据之间的关系：第一类是测试集的输入信息，第二类是测试集通过 LSTM 预测的输出信息，第三类是测试集的真实价格序列。下面以构造测试集 LSTM_test_outputs 的真实价格序列为例：

```
pred_range=7
window_len=14
LSTM_test_outputs_cal = []
test_set_select_cal=test_set2['close']#只要选取的列

for i in range(len(test_set_select_cal)-window_len):##用窗口长度数据预测下一个数据点#为了匹配2502个测试数据，不选择减去predrange
    temp_set = test_set_select_cal.values[(i+window_len):(i+window_len+pred_range)].copy()# temp_set = test_set[i:(i+window_len)].cop
    #temp_set = temp_set/temp_set[0] - 1#    不需要标准化了，这反而会造成误差
    #print(temp_set)
    LSTM_test_outputs_cal.append(temp_set)
#print(LSTM_test_outputs_cal)



LSTM_test_outputs_cal = [np.array(LSTM_test_outputs_cal) for LSTM_test_outputs_cal in LSTM_test_outputs_cal]
LSTM_test_outputs_cal = pd.DataFrame(LSTM_test_outputs_cal)#用dataframe提取解决奇怪的list嵌套：因为出现了空缺值
LSTM_test_outputs_cal=LSTM_test_outputs_cal.ffill()##因为没有range(len(test_set_select_cal)-window_len-predrange)
LSTM_test_outputs_cal=np.array(LSTM_test_outputs_cal)
print(LSTM_test_outputs_cal)

row1,col1=LSTM_test_outputs_cal.shape  ##这不能用dataframe
for i in range(0,row1):
    for j in range(0,col1):
        LSTM_test_outputs_cal[i][j]=(LSTM_test_outputs_cal[i][j]*std)+mean
        #print(LSTM_test_outputs_cal[i][j])
print(LSTM_test_outputs_cal)
```

图 24

三类数据形状信息如下：

```
:  1  print(LSTM_test_inputs_cal.shape,LSTM_test_outputs_cal.shape,lstm_pred_prices.shape)
(357, 14) (357, 7) (357, 7)
```

图 25

其中 LSTM_test_inputs_cal 和 LSTM_test_outputs_cal 都是取自测试集的逆标准化后的收盘价数据，因此在后续测试其他模型的准确率时不再重新命名，而是直接使用来产生分析结果，并无错误使用的问题。

然后定义如下的模型对趋势分类的正确率计算函数：

```
def calculateaccuracy(test_inputs_cal,test_outputs_cal,predictor_prices):
    flaglstm_count=[]
    for i in range(len(predictor_prices)):
        #print(i,j)
    #     print(test_inputs_cal[i])
    #     print(predictor_prices[i])
        a=np.mean(test_inputs_cal[i])
        b=np.mean(predictor_prices[i])
        #print(a,b)
        if b>=a:
            flaglstm=1
            flaglstm_count.append(flaglstm)
        else:
            flaglstm=0
            flaglstm_count.append(flaglstm)
    #print(flaglstm_count)
    flag_test_output_count=[]
    for i in range(len(test_outputs_cal)):
        #print(i,j)
    #     print(test_inputs_cal[i])
    #     print(predictor_prices[i])
        a=np.mean(test_inputs_cal[i])
        b=np.mean(test_outputs_cal[i])
        #print(a,b)
        if b>=a:
            flag_test=1
            flag_test_output_count.append(flag_test)
        else:
            flag_test=0
            flag_test_output_count.append(flag_test)
    #print(flag_test_output_count)
    if(len(flaglstm_count)==len(flag_test_output_count) ):
        # 计算分类正确率
        count_right=0
        for i in range(len(flaglstm_count)):
            if(flaglstm_count[i]==flag_test_output_count[i]):
                count_right+=1
        print("该方法趋势分类准确率为：",count_right/len(flaglstm_count))
    else:
        print("需要检查维度")
calculateaccuracy(LSTM_test_inputs_cal,LSTM_test_outputs_cal,lstm_pred_prices)
```

图 26

思想是将模型的数值通过求均值的方法将数值回归转换为二分类，比较数值的大小，若预测大于原输入段的价格，判断为涨，然后根据真实价格序列是否涨跌，将两者再对比，得到判断正确或者错误的结果，计算得出预测涨跌趋势的正确率。

LSTM 模型的结果是：准确率在 65.8%左右，F1 SCORE 在 0.68，AUC 为 0.655，并绘制了 ROC 曲线。

```
70 calculateaccuracy(LSTM_test_inputs_cal,LSTM_test_outputs_cal,lstm_pred_prices)
```

```
该方法趋势分类准确率为： 0.6554621848739496
AUC SCORE: 0.6552812858783008
F1 SCORE: 0.682170542635659
```

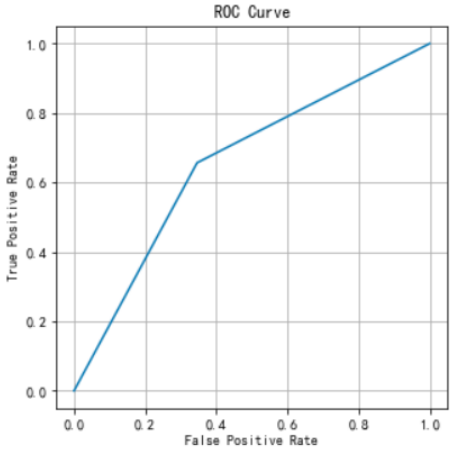


图 27

（二）BP 神经网络对股票价格走势的拟合

使用 Sklearn 自带的 MLPRegressor 模块测试神经网络的回归情况。与其他回归模型不同，可能是调参和变量等原因，可决系数的值看起来并不好。

```
from sklearn.neural_network import MLPRegressor #使用导入神经网络包
# for i in range(20,50):
np.random.seed(66)
target_pre_list=[]
mlp = MLPRegressor(hidden_layer_sizes=(160,14 ), activation='identity', solver='adam', alpha=0.00001, batch_size='auto', \
                   learning_rate='invscaling', learning_rate_init=0.00005, power_t=0.010, max_iter=3000, shuffle=False, \
                   random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.5, nesterovs_momentum=True, \
                   early_stopping=False, validation_fraction=0.15, beta_1=0.05, beta_2=0.01, epsilon=1e-09, n_iter_no_change=25, \
                   max_fun=30000) #简单起见，
mlp.fit(training_inputs_reshape, other_training_outputs)
mlp.score(training_inputs_reshape, other_training_outputs)
target_pre=mlp.predict(test_inputs_reshape)
# target_pre_list.append(target_pre)

#target_pre=np.mean(target_pre_list)
from sklearn.metrics import explained_variance_score,\
    mean_absolute_error,mean_squared_error,\
    median_absolute_error,r2_score
print('神经网络回归模型的平均绝对误差为：',mean_absolute_error(other_test_outputs,target_pre))
print('神经网络回归模型的均方误差为：',mean_squared_error(other_test_outputs,target_pre))
print('神经网络回归模型的中值绝对误差为：',median_absolute_error(other_test_outputs,target_pre))
print('神经网络回归模型的可解释方差值为：',explained_variance_score(other_test_outputs,target_pre))
print('神经网络回归模型的R^2值为：',r2_score(other_test_outputs,target_pre))
```

```
神经网络回归模型的平均绝对误差为： 0.9903553531683744
神经网络回归模型的均方误差为： 38.023573026954054
神经网络回归模型的中值绝对误差为： 0.1537290153506814
神经网络回归模型的可解释方差值为： -2.2164100039565717
神经网络回归模型的R^2值为： -2.2170292791587136
```

图 28

然后将预测值按之前的处理方法，逆标准化后，计算正确率和 F1 值、AUC 值，并绘制 ROC 曲线:

```
1 calculateaccuracy(LSTM_test_inputs_cal,LSTM_test_outputs_cal,mlp_test_outputs,check_axis=1)
```

```
该方法趋势分类准确率为： 0.5574229691876751
AUC SCORE: 0.5581706850363567
F1 SCORE: 0.5842105263157895
```

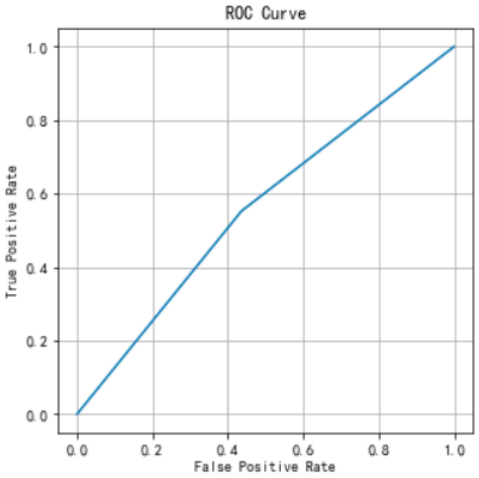


图 29

从 ROC 曲线中看得出来比抛硬币猜涨跌的情况要好一点，如果能排除参数的影响，BP 神经网络的确效果远远不如基于 LSTM 结构的深度学习模型。

（三）梯度提升回归树模型对股票价格走势的拟合

GBRT 是一种迭代的决策树算法，该算法由多棵决策树组成，目的是提升泛化能力，天然就可处理不同类型的数据（加入各种各样的 features），对空间外的异常点处理具有鲁棒性，预测效果较好。

测试梯度提升回归树模型 对价格走势 的拟合效果

```
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split

rng = np.random.RandomState(9)
GBR_house=GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False).fit(training_inputs_reshape, other_training_outputs)
print(GBR_house)

#评价
target_pre=GBR_house.predict(test_inputs_reshape)
from sklearn.metrics import explained_variance_score,\
    mean_absolute_error,mean_squared_error,\
    median_absolute_error,r2_score
print('梯度提升回归树模型的平均绝对误差为：',mean_absolute_error(other_test_outputs,target_pre))
print('梯度提升回归树模型的均方误差为：',mean_squared_error(other_test_outputs,target_pre))
print('梯度提升回归树模型的中值绝对误差为：',median_absolute_error(other_test_outputs,target_pre))
print('梯度提升回归树模型的可解释方差值为：',explained_variance_score(other_test_outputs,target_pre))
print('梯度提升回归树模型的R^2值为：',r2_score(other_test_outputs,target_pre))

```

图 30

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
梯度提升回归树模型的平均绝对误差为： 0.14743953340159713
梯度提升回归树模型的均方误差为： 2.1909451104363376
梯度提升回归树模型的中值绝对误差为： 0.026165770548288322
梯度提升回归树模型的可解释方差值为： 0.8150369740513416
梯度提升回归树模型的R^2值为： 0.8146325027291119
```

图 31

然然后将预测值按之前的处理方法，逆标准化后，计算正确率和 F1 值、AUC 值，并绘制 ROC 曲线:

```
1 calculateaccuracy(LSTM_test_inputs_cal,LSTM_test_outputs_cal,GBR_test_outputs,check_axis=1)
```

```
该方法趋势分类准确率为： 0.5378151260504201
AUC SCORE: 0.5357347876004592
F1 SCORE: 0.5736434108527133
```

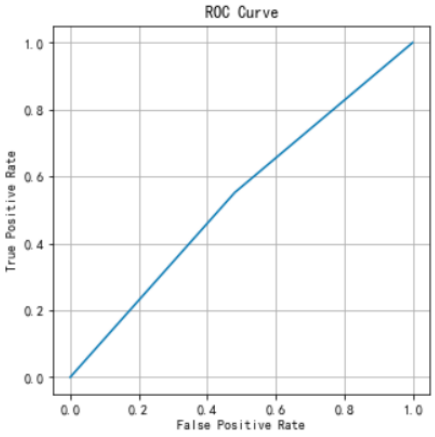


图 32

（四）支持向量回归对股票价格走势的拟合

```
from sklearn.svm import NuSVR
import numpy as np

clf = NuSVR(C=99,nu=1,kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, tol=0.001, cache_size=200, verbose=False, ma
clf.fit(training_inputs_reshape, other_training_outputs)
clf.score(training_inputs_reshape, other_training_outputs)

target_pre=clf.predict(test_inputs_reshape)




from sklearn.metrics import explained_variance_score,\
    mean_absolute_error,mean_squared_error,\
    median_absolute_error,r2_score
print('支持向量机回归模型的平均绝对误差为：',mean_absolute_error(other_test_outputs,target_pre))
print('支持向量机回归模型的均方误差为：',mean_squared_error(other_test_outputs,target_pre))
print('支持向量机回归模型的中值绝对误差为：',median_absolute_error(other_test_outputs,target_pre))
print('支持向量机回归模型的可解释方差值为：',explained_variance_score(other_test_outputs,target_pre))
print('支持向量机回归模型的R^2值为：',r2_score(other_test_outputs,target_pre))

```

```
支持向量机回归模型的平均绝对误差为： 0.2130167980329934
支持向量机回归模型的均方误差为： 2.4012534721541496
支持向量机回归模型的中值绝对误差为： 0.046612589124988335
支持向量机回归模型的可解释方差值为： 0.7968507117292182
支持向量机回归模型的R^2值为： 0.7968391155369483
```

图 33

然后将预测值按之前的处理方法，逆标准化后，计算正确率和 F1 值、AUC 值，并绘制 ROC 曲线:

```
calculateaccuracy(LSTM_test_inputs_cal,LSTM_test_outputs_cal,SVR_test_outputs,check_axis=1)
```

```
该方法趋势分类准确率为： 0.5546218487394958
AUC SCORE: 0.5506601607347876
F1 SCORE: 0.5954198473282443
```

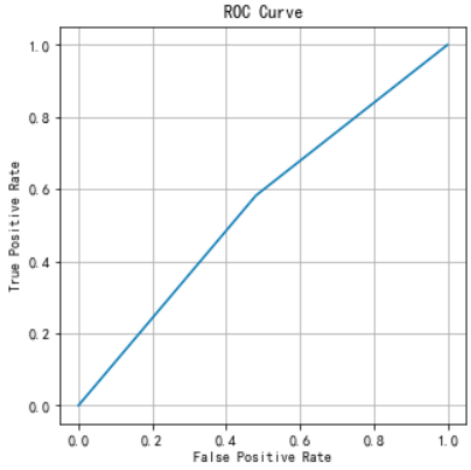


图 34

（五）鲁棒回归对股票价格走势的拟合

```
import numpy as np
from sklearn.linear_model import HuberRegressor, LinearRegression

rng = np.random.RandomState(0)

huber = HuberRegressor(epsilon=1.35, max_iter=100, alpha=0.0001, warm_start=False, fit_intercept=True, tol=1e-05).fit(training_input
huber.score(training_inputs_reshape, other_training_outputs)
target_pre=huber.predict(test_inputs_reshape)

linear = LinearRegression().fit(training_inputs_reshape, other_training_outputs)


print("Huber coefficients:", huber.coef_)

print("Linear Regression coefficients:", linear.coef_)


from sklearn.metrics import explained_variance_score,\
    mean_absolute_error,mean_squared_error,\
    median_absolute_error,r2_score
print('鲁棒回归模型的平均绝对误差为：',mean_absolute_error(other_test_outputs,target_pre))
print('鲁棒回归模型的均方误差为：',mean_squared_error(other_test_outputs,target_pre))
print('鲁棒回归模型的中值绝对误差为：',median_absolute_error(other_test_outputs,target_pre))
print('鲁棒回归模型的可解释方差值为：',explained_variance_score(other_test_outputs,target_pre))
print('鲁棒回归模型的R^2值为：',r2_score(other_test_outputs,target_pre))
```

```
鲁棒回归模型的平均绝对误差为： 0.3118595099082487
鲁棒回归模型的均方误差为： 4.042636101143394
鲁棒回归模型的中值绝对误差为： 0.064192028051102
鲁棒回归模型的可解释方差值为： 0.6579697279300425
鲁棒回归模型的R^2值为： 0.6579680007151569
```

图 35

然后将预测值按之前的处理方法，逆标准化后，计算正确率和 F1 值、AUC 值，并绘制 ROC 曲线:

```
In [253]: 1 calculateaccuracy(LSTM_test_inputs_cal,LSTM_test_outputs_cal,RR_test_outputs,check_axis=1)
```

```
该方法趋势分类准确率为： 0.5462184873949579
AUC SCORE: 0.5424799081515499
F1 SCORE: 0.586734693877551
```

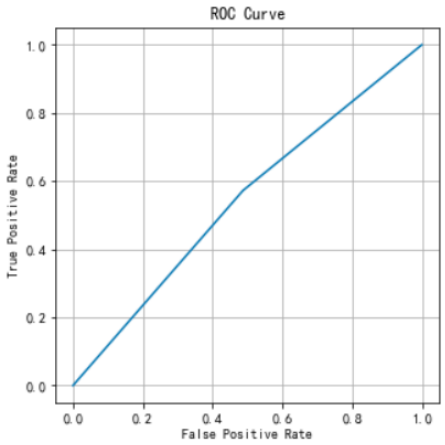


图 36

（六）模型分析及本文总结

对于回归模型来说，首先从 SVR、GBRT、RubustRegression 的结果中可以发现最终的准确率区别都不大，在 50%多的范围内，当然，一个比较重要的原因是本文没有更好地对这些模型的超参数进行优化，如果使用循环调节参数的方式，效果还能进一步提高。

其次本文发现回归模型的可决系数 R 平方值与分类准确率大多数情况下是正相关的，但是完全不是确定性的关系，在同一个模型内可决系数可能从 20%提高到 80%，最终分类准确率也只会提高 1%~4%甚至不提高的情况。而在不同模型间比较，比如 RR 和 GBRT，RR 的可决系数较低，但是正确率并不比 GBRT 低。

```
1 calculateaccuracy(LSTM_test_inputs_cal,LSTM_test_outputs_cal,LSTM_test_inputs_cal,check_axis=1)
```

```
该方法趋势分类准确率为： 0.5630252100840336
AUC SCORE: 0.5
F1 SCORE: 0.7204301075268817
```

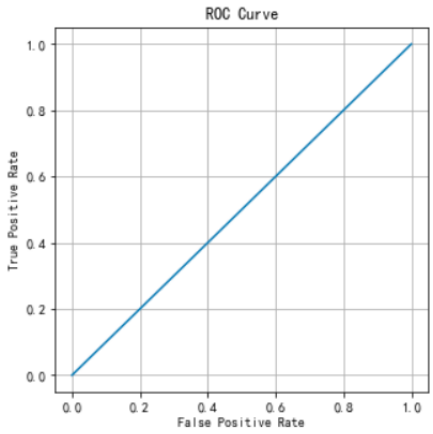


图 37

从上图中可以看出，是将训练集输入的数据集当做模型的预测情况，实际上如果根据测试集中的涨跌去推测未来的涨跌，也就是说视作同样的情况，其涨跌预测准确的概率应该是在 56%这样，这一数值数值只有 LSTM 模型能够超越，所有模型都没有达到 0.72 的 F1 分数，但是 AUC 的面积比例都超过了 0.5。

从结果上看，对比 LSTM 模型和其他机器学习回归模型，可以发现具有深层神经元结构的 LSTM 的预测效果要远好于回归模型，LSTM 的确是处理时间序列数据比较优良的模型。

本文尚有不足之处待以完善：在补充数据集上，第二类第三类还可以加入其他数据，比如经理人采购指数，M0、M1 货币量，GDP 增长率，公司季报的信息和公司分红转股的信息，尚来不及加入；在代码设计上，本文有一些程序来不及封装成函数以便重复调用，例如没有完善超参数的优化函数对各个模型做比较好的调参处理。

关于进一步增加本文指导性的展望：

1. 加入文本处理模型分析市场情绪变化和事件；
2. 加入相关公司的知识谱图的信息等进一步提高预测效果；
3. 构建量化选股构建组合模块，设计买入卖出模块和回测模型，对比沪深 300 买入并持有策略的收益率大小。