In [1]:

```
from csv import field_size_limit
from os import closerange
from cmath import sqrt
import tushare as ts
import numpy as np
import pandas as pd
import talib
from pandas import DataFrame as DF
from sklearn import svm
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
import matplotlib.dates as mdates
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import preprocessing as pre
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from sklearn.model_selection import GridSearchCV
from matplotlib.font_manager import FontProperties
from sklearn.ensemble import RandomForestClassifier #Random Forest Classifier model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import plot_roc_curve,roc_curve,auc,roc_auc_score
from sklearn.metrics import classification_report
```

In [2]:
```
# 1.Stock basic data acquisition
#BYD's stock code is 002594.SZ, Data is collected and stored in a local csv file
ts.set_token('32d73911de77ba68d52192f9cc366878995fbffc3e631e411c97846d')
pro=ts.pro_api()
df=pro.daily(ts_code='002594.SZ',start_date='20160101',end_date='20230630')
#Take the transaction date, open, high,low,close,pre_close pct_chg ,
#trading volume and other data of BYD in the past six years
df=df.sort_values('trade_date')
df
df1=df.set_index('trade_date')
df1
df1.to_csv('002594.SZ_daily.csv')
pd.options.display.max_rows = 12
result=pd.read_csv('002594.SZ_daily.csv',index_col=0,parse_dates=True)
result
```

Out[2]:

	ts_code	open	high	low	close	pre_close	change	pct_chg	vol	amount
trade_date										
2016-01-04	002594.SZ	64.40	64.40	58.40	58.76	64.40	-5.64	-8.7600	114880.14	7.091776e+05
2016-01-05	002594.SZ	55.80	60.61	55.80	59.35	58.76	0.59	1.0000	166704.92	9.830371e+05
2016-01-06	002594.SZ	59.50	60.78	59.37	60.42	59.35	1.07	1.8000	98824.86	5.943399e+05
2016-01-07	002594.SZ	59.00	59.47	55.00	55.41	60.42	-5.01	-8.2900	41293.10	2.367176e+05
2016-01-08	002594.SZ	58.00	60.68	56.70	59.43	55.41	4.02	7.2600	203567.18	1.201831e+06
2016-01-11	002594.SZ	57.99	60.99	57.20	57.88	59.43	-1.55	-2.6100	200493.50	1.184371e+06
...	...	...	...	...	...	...	...	...	...	...
2023-06-21	002594.SZ	267.01	273.28	266.25	267.78	268.25	-0.47	-0.1752	107811.76	2.906861e+06
2023-06-26	002594.SZ	262.43	268.69	262.30	262.89	267.78	-4.89	-1.8261	101512.22	2.681453e+06
2023-06-27	002594.SZ	262.02	263.85	257.08	260.05	262.89	-2.84	-1.0803	99028.65	2.570405e+06
2023-06-28	002594.SZ	260.00	262.50	258.10	260.25	260.05	0.20	0.0769	57269.51	1.489543e+06
2023-06-29	002594.SZ	260.25	260.78	255.23	255.70	260.25	-4.55	-1.7483	83485.05	2.148587e+06
2023-06-30	002594.SZ	254.13	260.00	253.70	258.27	255.70	2.57	1.0051	75724.36	1.952425e+06

1821 rows × 10 columns

In [3]:

```
df=pro.daily(ts_code='002594.SZ',start_date='20160101',end_date='20230630')
df['log_return'] = np.log(df['close' ]/ df['pre_close'])
df['up'] = np.where(df.log_return >= 0.0025, 1, 0)
df=df.sort_values('trade_date')
df
df1=df.set_index('trade_date')
df1
df1.to_csv('002594.SZ_daily.csv')
pd.options.display.max_rows = 12
data=pd.read_csv('002594.SZ_daily.csv',index_col=0,parse_dates=True)
data
```

Out[3]:

trade_date	ts_code	open	high	low	close	pre_close	change	pct_chg	vol	amount	log_return	up
2016-01-04	002594.SZ	64.40	64.40	58.40	58.76	64.40	-5.64	-8.7600	114880.14	7.091776e+05	-0.091652	0
2016-01-05	002594.SZ	55.80	60.61	55.80	59.35	58.76	0.59	1.0000	166704.92	9.830371e+05	0.009991	1
2016-01-06	002594.SZ	59.50	60.78	59.37	60.42	59.35	1.07	1.8000	98824.86	5.943399e+05	0.017868	1
2016-01-07	002594.SZ	59.00	59.47	55.00	55.41	60.42	-5.01	-8.2900	41293.10	2.367176e+05	-0.086560	0
2016-01-08	002594.SZ	58.00	60.68	56.70	59.43	55.41	4.02	7.2600	203567.18	1.201831e+06	0.070039	1
2016-01-11	002594.SZ	57.99	60.99	57.20	57.88	59.43	-1.55	-2.6100	200493.50	1.184371e+06	-0.026427	0
...	...	...	...	...	...	...	...	...	...	...	...	...
2023-06-21	002594.SZ	267.01	273.28	266.25	267.78	268.25	-0.47	-0.1752	107811.76	2.906861e+06	-0.001754	0
2023-06-26	002594.SZ	262.43	268.69	262.30	262.89	267.78	-4.89	-1.8261	101512.22	2.681453e+06	-0.018430	0
2023-06-27	002594.SZ	262.02	263.85	257.08	260.05	262.89	-2.84	-1.0803	99028.65	2.570405e+06	-0.010862	0
2023-06-28	002594.SZ	260.00	262.50	258.10	260.25	260.05	0.20	0.0769	57269.51	1.489543e+06	0.000769	0
2023-06-29	002594.SZ	260.25	260.78	255.23	255.70	260.25	-4.55	-1.7483	83485.05	2.148587e+06	-0.017638	0
2023-06-30	002594.SZ	254.13	260.00	253.70	258.27	255.70	2.57	1.0051	75724.36	1.952425e+06	0.010001	1

1821 rows × 12 columns

In [4]:

```python
# 2.Simple derived variable data construction
df1['O-C'] = df1['open'] - df1['close']
df1['H-L'] = df1['high'] - df1['low']
df1['pre_close'] = df1['close'].shift(1)
df1['price_change'] = df1['close'] - df1['pre_close']
df1['p_change'] = (df1['close'] - df1['pre_close']) / df1['pre_close'] * 100
df1.head(5)
```

Out[4]:

trade_date	ts_code	open	high	low	close	pre_close	change	pct_chg	vol	amount	log_return	up	O-C	H-L	price_change
20160104	002594.SZ	64.4	64.40	58.40	58.76	NaN	-5.64	-8.76	114880.14	7.091776e+05	-0.091652	0	5.64	6.00	NaN
20160105	002594.SZ	55.8	60.61	55.80	59.35	58.76	0.59	1.00	166704.92	9.830371e+05	0.009991	1	-3.55	4.81	0.59
20160106	002594.SZ	59.5	60.78	59.37	60.42	59.35	1.07	1.80	98824.86	5.943399e+05	0.017868	1	-0.92	1.41	1.07
20160107	002594.SZ	59.0	59.47	55.00	55.41	60.42	-5.01	-8.29	41293.10	2.367176e+05	-0.086560	0	3.59	4.47	-5.01
20160108	002594.SZ	58.0	60.68	56.70	59.43	55.41	4.02	7.26	203567.18	1.201831e+06	0.070039	1	-1.43	3.98	4.02

In [5]:

```python
# 3.Moving average related data construction
df1['MA5'] = df1['close'].rolling(5).mean()
df1['MA10'] = df1['close'].rolling(10).mean()
df1.dropna(inplace=True)
df1.head()
```

Out[5]:

trade_date	ts_code	open	high	low	close	pre_close	change	pct_chg	vol	amount	log_return	up	O-C	H-L	price_change	p
20160115	002594.SZ	59.01	59.8	57.02	58.05	59.90	-1.85	-3.09	109584.00	639074.9572	-0.031372	0	0.96	2.78	-1.85	-
20160118	002594.SZ	57.01	58.6	56.52	57.68	58.05	-0.37	-0.64	105704.97	607753.5816	-0.006394	0	-0.67	2.08	-0.37	-
20160119	002594.SZ	57.50	59.1	56.98	58.87	57.68	1.19	2.06	137399.99	798925.2821	0.020421	1	-1.37	2.12	1.19	
20160120	002594.SZ	59.00	59.7	57.10	57.74	58.87	-1.13	-1.92	113692.67	663507.6674	-0.019381	0	1.26	2.60	-1.13	-
20160121	002594.SZ	57.10	58.3	56.00	56.02	57.74	-1.72	-2.98	115329.17	660993.1444	-0.030241	0	1.08	2.30	-1.72	-

In [6]:
```python
# 4.Construct derived variable data through the TA-Lib library
df1['RSI'] = talib.RSI(df1.close.values,timeperiod=14)
df1['MOM'] = talib.MOM(df1.close.values,timeperiod=5)
df1['EMA12'] = talib.EMA(df1.close.values,timeperiod=12) #12-day moving average
df1['EMA26'] = talib.EMA(df1.close.values,timeperiod=26) #26-day moving average
df1['MACD'],df1['MACDsignal'],df1['MACDhist'] = talib.MACD(df1.close.values,fastperiod=6,slowperiod=12,signalperiod=9)
df1.dropna(inplace=True)
df1.head()
```

Out[6]:

	ts_code	open	high	low	close	pre_close	change	pct_chg	vol	amount	...	p_change	MA5	MA10	RSI	M
trade_date																
20160226	002594.SZ	52.81	53.09	51.22	52.35	51.80	0.55	1.06	88412.62	462164.3787	...	1.061776	53.676	53.181	41.582740	-
20160229	002594.SZ	53.10	53.30	49.00	50.51	52.35	-1.84	-3.51	124177.34	630542.5892	...	-3.514804	52.876	53.232	37.314420	-
20160301	002594.SZ	50.54	51.88	50.08	51.54	50.51	1.03	2.04	80229.88	409803.5939	...	2.039200	52.244	53.178	40.967341	-
20160302	002594.SZ	51.09	54.21	51.09	53.90	51.54	2.36	4.58	118288.17	623608.1316	...	4.578968	52.020	53.187	48.388603	-
20160303	002594.SZ	54.00	54.92	53.55	53.58	53.90	-0.32	-0.59	97766.04	529712.8847	...	-0.593692	52.376	53.220	47.516333	

5 rows × 25 columns

In [7]:
```python
X = df1[['close','vol','O-C','MA5','MA10','H-L','RSI','MOM','EMA12','MACD','MACDsignal','MACDhist']]
y = np.where(df1.log_return >= 0.0025, 1, 0)
```

In [8]:
```python
# Dividing the overall data into training and testing sets,
# with training sets accounting for 90% and testing sets accounting for 10%.
X_length = X.shape[0]
split = int(X_length * 0.9)
X_train,X_test = X[:split],X[split:]
y_train,y_test = y[:split],y[split:]
```

In [9]:
```
# Model building
# Set model parameters: max_depth of the decision tree is set to 3, that is, each decision tree has only 3 layers at most. The numb
er of weak learners (i.e., decision tree model) n_estimators is set to 10, that is,
# there are 10 decision trees in the random forest.The minimum sample number of leaf nodes is set to 10, that is, if the sample num
ber of leaf nodes is less than 10, the splitting stops. The role of the random state
# parameter random_state is to make the results consistent.

model = RandomForestClassifier(max_depth=3,n_estimators=10,min_samples_leaf=10,random_state=120)
model.fit(X_train,y_train)
```

Out[9]:
```
RandomForestClassifier(max_depth=3, min_samples_leaf=10, n_estimators=10,
                       random_state=120)
```

In [10]:
```
# Model evaluation and use, the model is evaluated and used to predict the rise and fall of the stock price the next day
y_pred = model.predict(X_test)
a = pd.DataFrame()
a["prediction"] = list(y_pred)
a["actual"] = list(y_test)
a.head(10)
```

Out[10]:

	prediction	actual
0	0	0
1	0	0
2	1	1
3	1	1
4	0	0
5	1	1
6	0	0
7	0	1
8	0	0
9	0	0

In [11]:
```
# Model accuracy evaluation
accuracy = accuracy_score(y_pred,y_test)
accuracy
model.score(X_test,y_test)
```

Out[11]: 0.8435754189944135

In [12]:
```
# Analyze the characteristic importance of characteristic variables
importances = model.feature_importances_
a = pd.DataFrame()
a["features"] = X.columns
a["importance of features"] = importances
a=a.sort_values("importance of features",ascending=False)
a
# It is found that the feature variables such as O-C, vol,MACD_hist, RSI, H-L, MOM, MA10, EMA12, MACD, MA5 indicators have a great
# influence on the prediction accuracy of the rise and fall of the stock price in the next day
```

Out[12]:

	features	importance of features
2	O-C	0.739528
1	vol	0.064348
11	MACDhist	0.060924
7	MOM	0.040247
9	MACD	0.021891
5	H-L	0.020007
6	RSI	0.018103
3	MA5	0.013744
4	MA10	0.007941
10	MACDsignal	0.006796
0	close	0.004643
8	EMA12	0.001827

In [13]:

```
# Model parameter tuning
from sklearn.model_selection import GridSearchCV
parameters={'n_estimators':[5,10,20],'max_depth':[2,3,4,5,6],'min_samples_leaf':[5,10,20,30]}
new_model = RandomForestClassifier(random_state=120)
grid_search = GridSearchCV(new_model,parameters,cv=6,scoring='accuracy')
grid_search.fit(X_train,y_train)
grid_search.best_params_

# output
{'max_depth': 6, 'min_samples_leaf': 30, 'n_estimators': 10}
```

Out[13]:

```
{'max_depth': 6, 'min_samples_leaf': 30, 'n_estimators': 10}
```

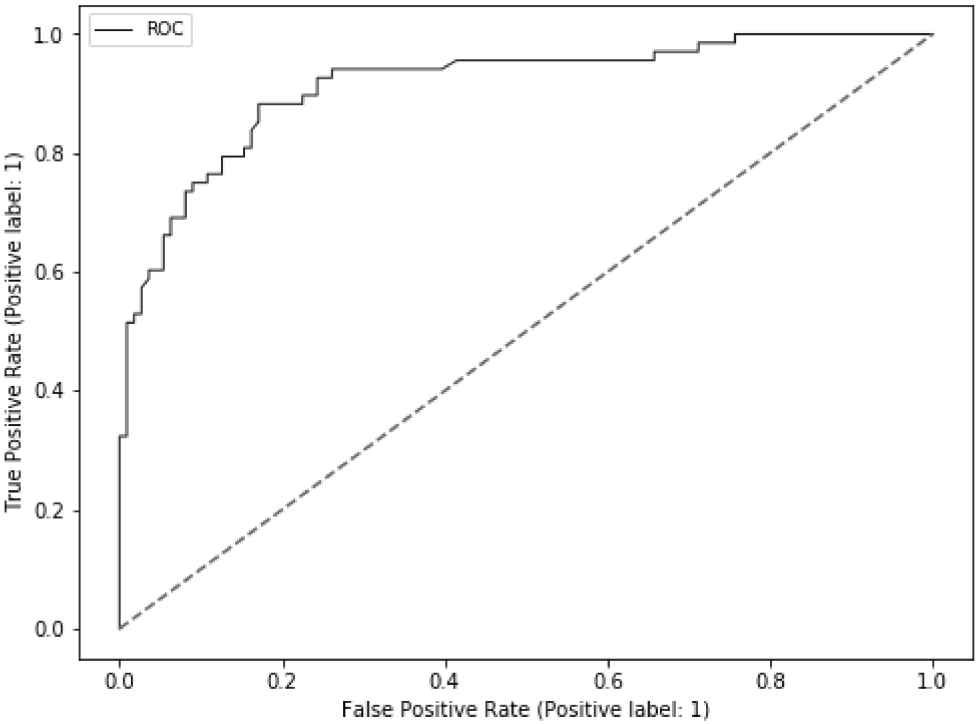
In [14]:
```
# The features variables we finally selected are: O-C, vol,MACD_hist, RSI, H-L, MOM, MA10, EMA12, MACD, MA5
# At the same time, adjust the relevant parameters of Random forest classifier

# 1 Feature variable selection
X = df1[['vol','O-C','H-L','MA5','MA10','EMA12','RSI','MOM','MACD','MACDhist']]
y = np.where(df1.log_return >= 0.0025, 1, 0)

# 2 Divide the training set and test set
X_length = X.shape[0]
split = int(X_length * 0.9)
X_train,X_test = X[:split],X[split:]
y_train,y_test = y[:split],y[split:]

# 3 Model reset
model = RandomForestClassifier(max_depth=6,n_estimators=10,min_samples_leaf=30,random_state=120)
model.fit(X_train,y_train)

# 4 Model evaluation and use (to predict the next day's stock price rise and fall)
y_pred = model.predict(X_test)
a = pd.DataFrame()
a["prediction"] = list(y_pred)
a["actual"] = list(y_test)
a.head(10)
```

Out[14]:

	prediction	actual
0	0	0
1	0	0
2	1	1
3	1	1
4	0	0
5	1	1
6	0	0
7	0	1
8	0	0
9	0	0

In [15]:

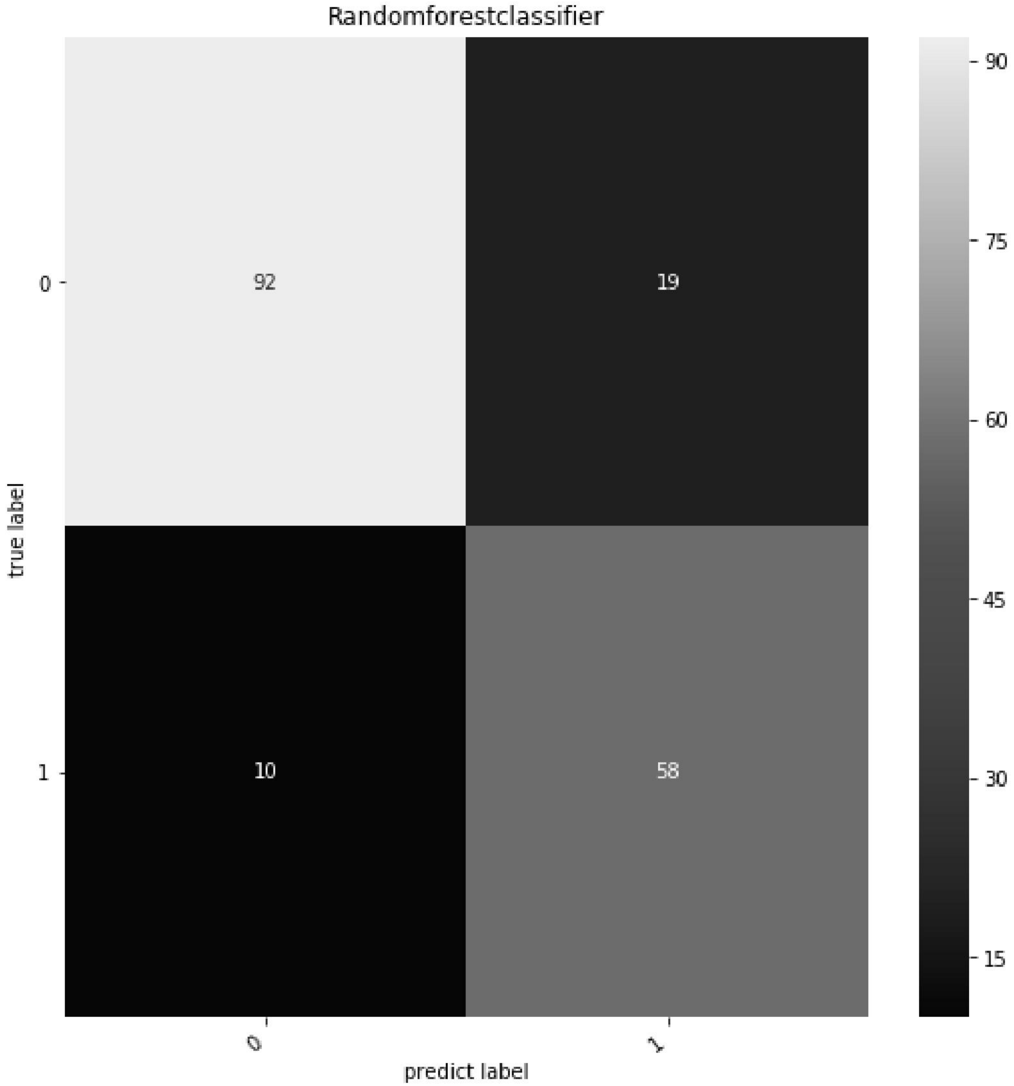
```python
# ROC of random forest classifier model
fig,ax = plt.subplots(figsize=(8,6))
roc = RocCurveDisplay.from_estimator(estimator=model, X=X_test, y=y_test, ax=ax, linewidth=1,label='ROC',color="b")
ax.legend(fontsize=9)
plt.plot([0,1], [0,1], linestyle='--')
plt.show()
```



In [16]:

```
#confusion matrix
figure = plt.subplots(figsize=(9,9))
plt.title("Randomforestclassifier")
cm=confusion_matrix(y_test, y_pred)
heatmap = sns.heatmap(cm, annot=True, fmt='d')
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right')
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=40, ha='right')
plt.ylabel("true label")
plt.xlabel("predict label")
```

Out[16]: Text(0.5,60,'predict label')



In [17]:

```
#classfication report
print(classification_report(y_test,y_pred,target_names=['0','1']))
```

```
              precision    recall  f1-score   support

           0       0.90      0.83      0.86       111
           1       0.75      0.85      0.80        68

    accuracy                           0.84       179
   macro avg       0.83      0.84      0.83       179
weighted avg       0.85      0.84      0.84       179
```