# 数据获取

通过调用tushare库，该库是一个用于获取中国股票市场数据的Python库。

调用ts.pro\_bar()接口将数据存储在df中

再编写爬虫脚本，用于从Tushare获取基金数据，并对获取到的数据进行处理和存储。

|  |
| --- |
| for date in date\_list:  try:  pro = ts.pro\_api()  time.sleep(0.75)  df = pro.fund\_daily(ts\_code = '510050.SH',trade\_date = date,asset='FD')  if i == 0:  data = df  i+=1  else:  i+=1  data = pd.concat([data,df])  if i%100 == 0:  print('爬取第{}天数据，数据总长度为:{}'.format(date,len(data)))  except Exception as err:  print(err)  date\_list.insert(0,date)  time.sleep(30)  label = data\_sort['pre\_close']<=data\_sort['close']  data\_sort['label'] = label  data\_sort.head()  data\_sort.to\_csv('20\_year\_FD.csv',index = False) |



# 数据探索分析

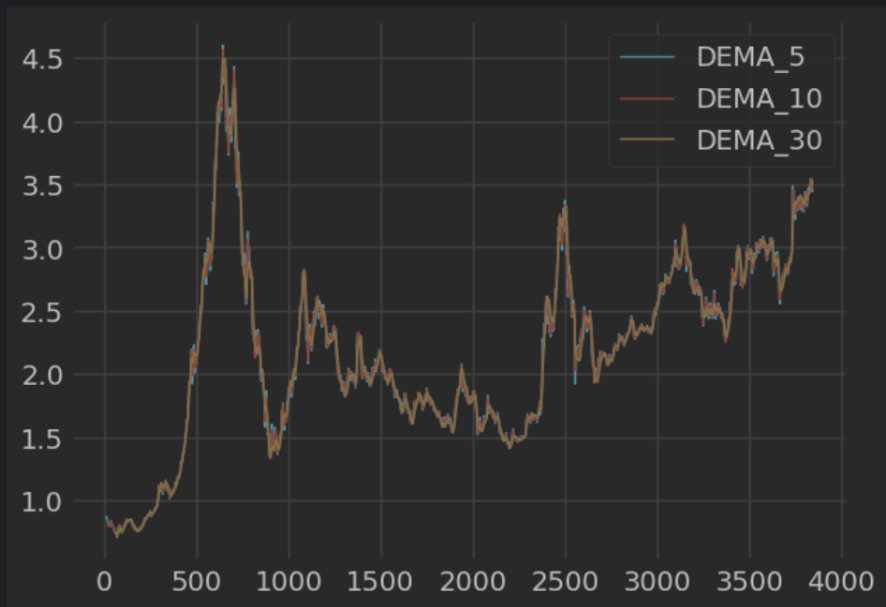
## 添加移动平均线指标

定义函数 add\_MA，这个函数接受两个参数，timeperiod 表示移动平均线的窗口大小，df 是一个DataFrame，表示需要添加移动平均线的数据集。函数内部会根据 timeperiod 和一系列预定义的移动平均线类型，计算对应的移动平均线值，并将其添加到DataFrame中。

代码通过调用 add\_MA 函数，分别为数据集 dataset 添加了窗口大小为5、10和30的移动平均线指标。

这里通过连续的三次调用，将不同窗口大小的移动平均线指标添加到了同一个数据集中。

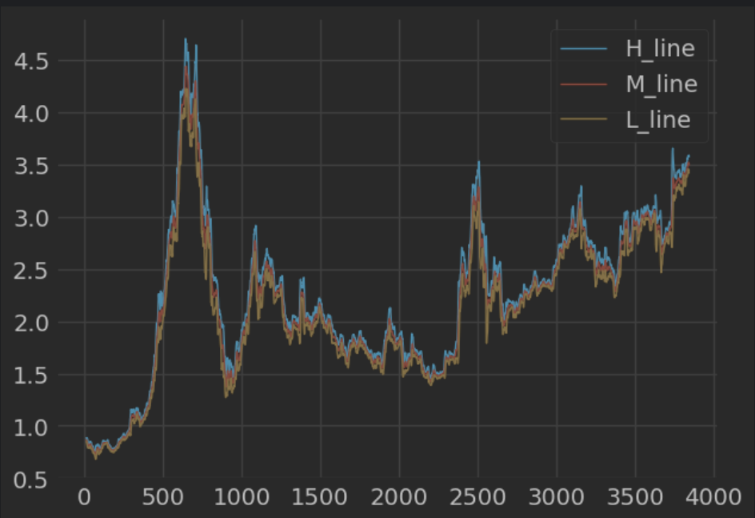
|  |
| --- |
| def add\_MA(timeperiod,df):  types=['SMA','EMA','WMA','DEMA','TEMA',  'TRIMA','KAMA','MAMA','T3']  types = [i+'\_'+str(timeperiod) for i in types]  for i in range(len(types)):  df[types[i]]=ta.MA(df.close.to\_numpy(),timeperiod=timeperiod,matype=i)  return df  dataset = add\_MA(5,dataset)  dataset = add\_MA(10,dataset)  dataset = add\_MA(30,dataset)  dataset.head() |



## 布林带计算

使用技术指标计算库TA-Lib计算布林带指标，并将计算得到的上轨、中轨和下轨线添加到给定的DataFrame中

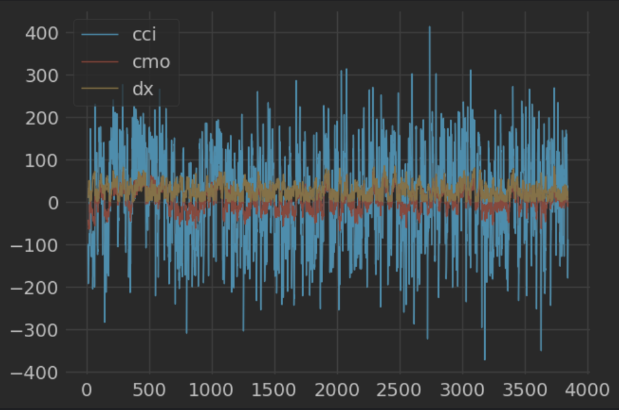
|  |
| --- |
| H\_line,M\_line,L\_line=ta.BBANDS(dataset.close.to\_numpy(), timeperiod=10, nbdevup=2, nbdevdn=2, matype=0)  dataset['H\_line']=H\_line  dataset['M\_line']=M\_line  dataset['L\_line']=L\_line |



## 加入动量指标

计算给定数据集的CCI、CMO、DX和MACD指标，并将计算结果添加到数据集中。

|  |
| --- |
| dataset['cci']=ta.CCI(dataset.high.to\_numpy(),dataset.low.to\_numpy(), dataset.close.to\_numpy(), timeperiod=14)  dataset['cmo'] = ta.CMO(dataset.close.to\_numpy(), timeperiod=14)  dataset['dx']=ta.DX(dataset.high.to\_numpy(),dataset.low.to\_numpy(), dataset.close.to\_numpy(), timeperiod=14)  macd,macdsignal,macdhist=ta.MACD(dataset.close.to\_numpy(),fastperiod=12, slowperiod=26, signalperiod=9)  dataset['macd'] = macd  dataset['macdsignal'] = macdsignal  dataset['macdhist'] = macdhist |



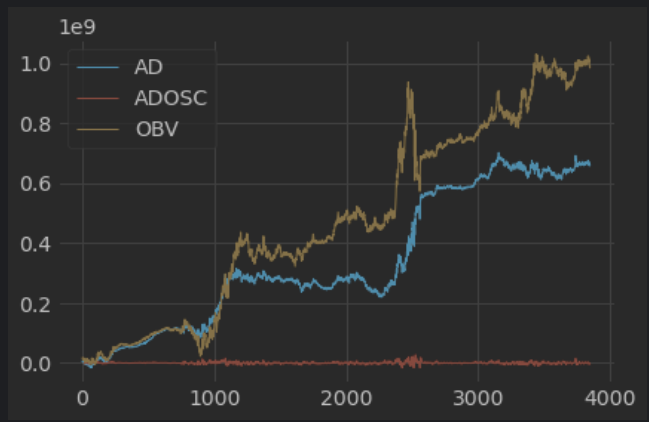
## 成交量指标

从 dataset DataFrame 中提取数据列（高价、低价、收盘价和交易量），并转换为Numpy数组。

使用TA-Lib库的技术指标计算函数（AD、ADOSC、OBV）来计算各个技术指标。将计算得到的技术指标结果添加到 dataset DataFrame 中的新列中

通过Pandas的 plot() 方法绘制技术指标的折线图，帮助用户直观地观察技术指标的变化趋势

|  |
| --- |
| dataset['AD'] = ta.AD(dataset.high.to\_numpy(),dataset.low.to\_numpy(),dataset.close.to\_numpy(),dataset.vol.to\_numpy())  dataset['ADOSC'] = ta.ADOSC(dataset.high.to\_numpy(),dataset.low.to\_numpy(),dataset.close.to\_numpy(),dataset.vol.to\_numpy(),fastperiod=3, slowperiod=10)  dataset['OBV'] = ta.OBV(dataset.close.to\_numpy(),dataset.vol.to\_numpy())  dataset[['AD','ADOSC','OBV']].plot(linewidth=1) |

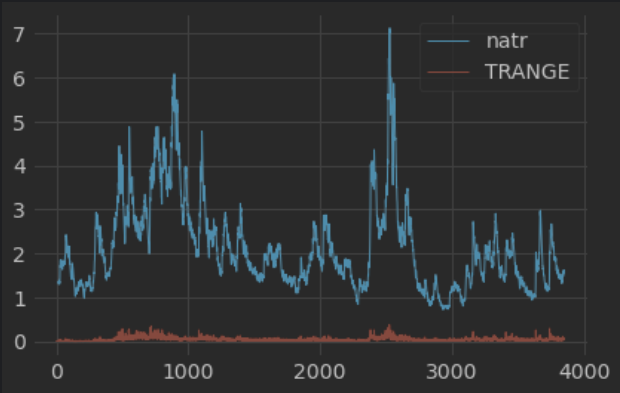


## 波特率指标

TA-Lib库的技术指标计算函数（如7NATR、ATR、TRANGE）来计算各个技术指标。

通过 .to\_numpy() 方法将高价、低价和收盘价数据转换为Numpy数组后传入函数。

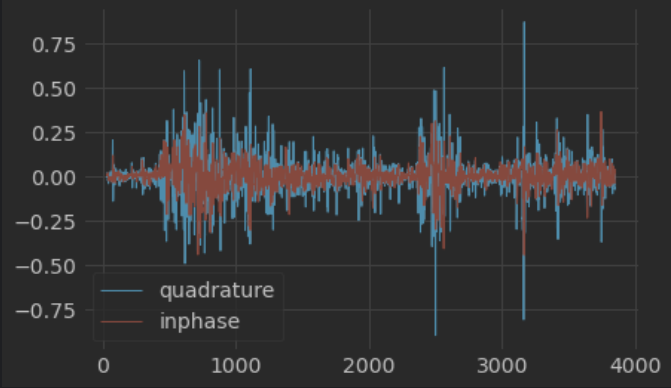
|  |
| --- |
| dataset['natr'] = ta.NATR(dataset.high.to\_numpy(),dataset.low.to\_numpy(),dataset.close.to\_numpy(),timeperiod=14)  dataset['atr'] = ta.ATR(dataset.high.to\_numpy(),dataset.low.to\_numpy(),dataset.close.to\_numpy(),timeperiod=14)  dataset['TRANGE'] = ta.TRANGE(dataset.high.to\_numpy(),dataset.low.to\_numpy(),dataset.close.to\_numpy()) |



## 周期性指标

将不同的希尔伯特变换函数应用于股价数据，以揭示其中潜在的市场趋势和周期表现，提供更多信息以进行后续的市场分析和策略制定。在理想状态下，这些希尔伯特变换函数能够提供关于趋势强度、价格变化的方向和速度、周期开始和结束的指标等信息，同时，这些信息能用于生成交易信号和制定投资决策。

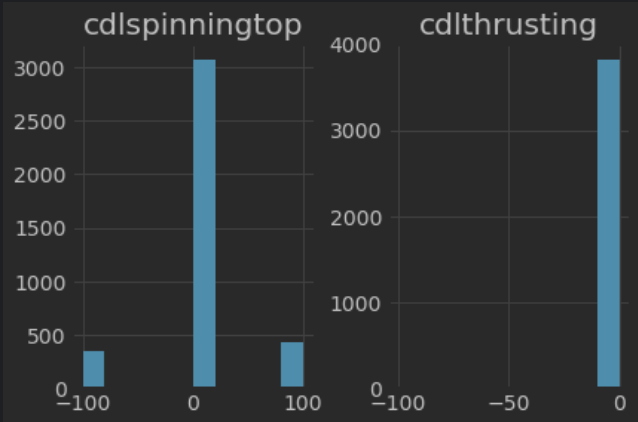
|  |
| --- |
| dataset['ht\_dcperiod'] = ta.HT\_DCPERIOD(dataset.close.to\_numpy())  dataset['ht\_dcphase'] = ta.HT\_DCPHASE(dataset.close.to\_numpy())  inphase,quadrature = ta.HT\_PHASOR(dataset.close.to\_numpy())  dataset['inphase'] = inphase  dataset['quadrature'] = quadrature  dataset['sine'],dataset['leadsine'] = ta.HT\_SINE(dataset.close.to\_numpy())  dataset['ht\_trendmode'] = ta.HT\_TRENDMODE(dataset.close.to\_numpy()) |



## 股市特定规律识别

使用TA-Lib库中的蜡烛图模式识别函数。每个CDL函数都代表一种蜡烛图模式。例如，CDL3BLACKCROWS 是三只乌鸦，CDLHAMMER 是锤形线，CDLDOJI 是十字线等。这些都是一种特殊的价格模式，预示着未来的价格走势。

这段代码的结果会在数据集中添加一系列新的列。每一列都会显示相应的蜡烛图模式是否在某一天出现，通常用正值表示上涨模式，负值表示下跌模式，零值表示没有模式出现。



# 预测收盘价

## 使用LSTM进行预测

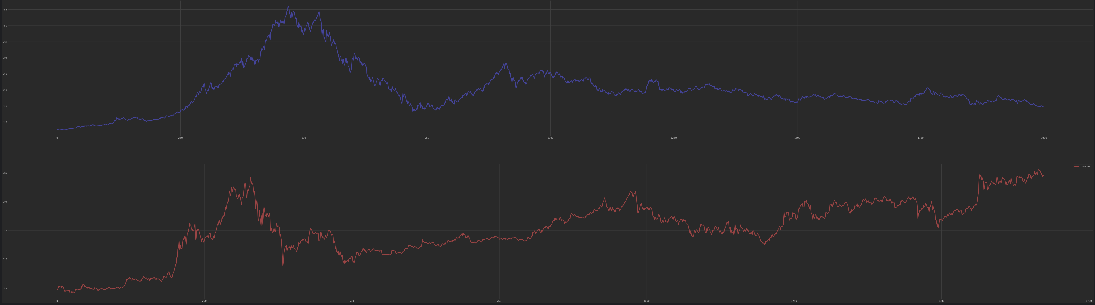
#### 1.1训练测试集划分

从原始数据集（存储在变量dataset中）删除了与目标任务无关的列（如 'label', 'ts\_code'和'trade\_date'），然后将结果存储在features\_close中。然后，目标标签（即 'close' 列，可能表示某种金融产品的收盘价）被提取并存储在label\_close中。

将features\_close和label\_close数据集根据训练样本的数量划分为训练集（training\_set和train\_label）和测试集（test\_set和test\_label）。前2000个样本用于训练，剩余的样本用于测试。

在完成数据集划分后，接下来的部分是对两个数据集进行可视化。通过matplotlib库实现，创建了两个子图。在第一个子图中，蓝色曲线代表训练集中的第一个特征随时间变化的趋势。在第二个子图中，红色曲线代表测试集中的第一个特征随时间变化的趋势。通过这样的方式，我们可以直观地理解训练集和测试集的特征分布差异。

|  |
| --- |
| features\_close = dataset.drop(labels = ['label','ts\_code','trade\_date'], axis = 1)  label\_close = dataset.close  train\_num = 2000  training\_set = features\_close[:train\_num].values.astype('float')  test\_set = features\_close[train\_num:].values.astype('float')  train\_label = label\_close[:train\_num].values.astype('float')  test\_label = label\_close[train\_num:].values.astype('float')  fig, (axes1, axes2) = plt.subplots(2, 1, figsize=(100, 30))  axes1.plot(training\_set[:,0], color='blue', label='Training set')  axes2.plot(test\_set[:,0], color='red', label='Test set') |



#### 1.2建立LSTM模型

训练数据进行归一化处理，使用的是MinMaxScaler，将数据放缩到0和1之间。然后，代码创建了输入数据的滑动窗口，窗口大小是5，使用连续5天的数据来预测第6天的股价。这样的处理是因为在处理时间序列问题时，我们总是假设未来N期的值可能与前M期有关。

之后是建立LSTM模型的部分。这个模型有四层LSTM层，每一层后面都跟着一个Dropout层用来防止过拟合。

|  |
| --- |
| # LSTM  regressor = Sequential()  # firt LSTM layer  regressor.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],X\_train.shape[2])))  regressor.add(Dropout(0.2))  # second LSTM layer  regressor.add(LSTM(units=50, return\_sequences=True))  regressor.add(Dropout(0.2))  # third LSTM layer  regressor.add(LSTM(units=50, return\_sequences=True))  regressor.add(Dropout(0.2))  # fourth LSTM layer  regressor.add(LSTM(units=50))  regressor.add(Dropout(0.2))  # output layer  regressor.add(Dense(units=1))  # compile the RNN  regressor.compile(optimizer='rmsprop', loss='mean\_squared\_error')  # fit to the training set  regressor.fit(X\_train, y\_train, epochs=60, batch\_size=32) |

