**【量化课堂】手把手教你如何应用机器学习**

**导语：**

在我们的数学课堂中，我们给大家简单介绍了几种机器学习方法的算法原理（SVM，朴素贝叶斯，随机森林等等），在每篇文章的最后，我们都放了一个非常小的例子来帮助大家使用这些算法。这一篇就给大家展示一个更贴近实际的例子，来帮助大家更好的使用这些算法。

本文中，为了降低提取的特征中的噪声，我们将常见技术分析指标进行离散化，作为特征值，如RSI，MACD，等等。然后应用SVM，朴素贝叶斯和随机森林算法来预测下一个交易日的涨跌情况。

为了和实际应用场景更接近，我们每天都会向训练数据中添加今天的数据，因为训练数据集发生了改变，我们每天都会训练一个新的模型。下面我们正式开工。

*本文由JoinQuant量化课堂推出 。难度标签为进阶上，理解深度标签：level-0*

**作者：yongpeng.r**  
**编辑：宏观经济算命师**

**阅读前需要了解：**  
文中涉及到SVM，随机森林，朴素贝叶斯算法，可以参照量化课堂中的 [支持向量机入门](https://www.joinquant.com/post/1510)，[随机森林入门](https://www.joinquant.com/post/1571)，[朴素贝叶斯入门](https://www.joinquant.com/post/1727)。

**训练预测代码介绍:**

本文是之前的支持向量机，随机森林，朴素贝叶斯等机器学习算法的应用篇，

**特征计算**  
我们先通过get\_price方法得到该支股票的前35个交易日的开高低收和交易量数据，然后利用talib包提供的特征计算API计算相应的特征。在这次实验中，我们选取了SMA，WMA，MOM，STCK，STCD，MACD，RSI，WILLR，CCI，MFI，OBV，ROC，CMO等技术指标作为训练模型的特征。计算SMA指标的代码如下：

trading\_days = get\_all\_trade\_days()

start\_day = trading\_days[index - 35]

end\_day = trading\_days[index]

stock\_data = get\_price(test\_stock, start\_date=start\_day, end\_date=end\_day, \

frequency='daily', fields=['close','high','low','volume'])

close\_prices = stock\_data['close'].values

sma\_data = talib.SMA(close\_prices)[-1]

**分类标签确定**  
如果下一个交易日的收盘价比当前交易日的收盘价高，label的值为True，否则为False。相关代码如下：

start\_day = trading\_days[index]

end\_day = trading\_days[index 1]

stock\_data = get\_price(test\_stock, start\_date=start\_day, end\_date=end\_day, \

frequency='daily', fields=['close','high','low','volume'])

close\_prices = stock\_data['close'].values

label = False

if close\_prices[-1] > close\_prices[-2]:

label = True

**特征离散化**  
通过talib库计算出来的特征都是连续值，由于股市中的噪声非常多，如果直接将连续值特征放到机器学习算法中训练，得到的结果很可能过拟合的（回测结果好，实盘结果差）。为了避免过拟合，本文中结合各个特征的含义，将连续值特征离散化为二值特征。比如对于SMA特征，离散化的方法就是和当日收盘价相比较，若SMA小于当前收盘价，离散之后的特征值是1，若SMA大于等于收盘价，离散之后的特征值是-1。SMA离散化的代码如下，x\_all是存放所有特征值的二维数组，x\_all中的第1列存放的就是SMA特征数据。

# SMA

if x*\_all[index][0] < x\_*all[index][-1]:

x\_all[index][0] = 1

else:

x\_all[index][0] = -1

**应用机器学习算法**  
在准备好数据之后，就需要利用机器学习算法来训练模型并且利用训练好的模型预测下一个交易日的涨跌情况了。下面是通过SVM算法预测的代码：

clf = svm.SVC()

clf.fit(x\_train, y\_train)

prediction = clf.predict(x\_test)