案例三：文本分类

中文文本分类案例：邮件分类器

文件说明

（1）获取文件夹中每一封邮件中的所有词语，保存所有邮件中的单词到列表中，每个元素是一个子列表，其中存放一个邮件中的所有单词

（2）获取邮件的特征向量和标签，特征为前600个单词的每个单词在每个邮件中出现的频率

（3）创建模型（选择MultinomialNB和BernoulliNB任意一个），进行训练，并保存训练模型和前600个单词

（4）使用测试集测试：首先加载训练好的模型，并导入前600单词文件，读取测试文件，将测试文件进行特征向量转换后进行预测，输出预测的标签。

import os  
import jieba  
  
  
def load\_emails(folder\_path, end, start=0):  
 emails = []  
 for filename in [str(i) + '.txt' for i in range(start, end + 1)]:  
 with open(os.path.join(folder\_path, filename), 'r', encoding='utf-8') as file:  
 content = file.read()  
 words = jieba.lcut(content)  
 emails.append(words)  
 return emails  
  
  
folder\_path = 'email-text'  
emails = load\_emails(folder\_path, end=150)  
# print(len(emails))  
  
from sklearn.feature\_extraction.text import CountVectorizer  
  
  
def get\_top\_words(emails, top\_n=600):  
 vectorizer = CountVectorizer(max\_features=top\_n, token\_pattern=r'(?u)\b[^0-9\W\_]+\b')  
 all\_words = [' '.join(email) for email in emails]  
 vectorizer.fit(all\_words)  
 top\_words = vectorizer.get\_feature\_names\_out()  
 return top\_words, vectorizer  
  
  
top\_words, vectorizer = get\_top\_words(emails)  
# print(top\_words)  
  
def get\_feature\_vectors(emails, vectorizer):  
 all\_words = [' '.join(email) for email in emails]  
 # print(vectorizer.transform(all\_words).toarray())  
 feature\_vectors = vectorizer.transform(all\_words).toarray()  
 return feature\_vectors  
  
  
feature\_vectors = get\_feature\_vectors(emails, vectorizer)  
from numpy import array  
  
labels = array([1] \* 127 + [0] \* 24) #  
from sklearn.naive\_bayes import MultinomialNB  
import joblib  
  
  
# 创建并训练模型  
model = MultinomialNB()  
model.fit(feature\_vectors, labels)  
  
# 保存模型和前600个单词  
joblib.dump(model, 'email\_classifier\_model.pkl')  
joblib.dump(top\_words, 'top\_600\_words.pkl')  
# 加载模型和前600个单词  
model = joblib.load('email\_classifier\_model.pkl')  
top\_words = joblib.load('top\_600\_words.pkl')  
# vectorizer = CountVectorizer(vocabulary=top\_words, token\_pattern=r'(?u)\b\w+\b')  
vectorizer = CountVectorizer(vocabulary=top\_words, token\_pattern=r'(?u)\b[^0-9\W\_]+\b')  
  
test\_folder\_path = 'email-text'  
test\_emails = load\_emails(test\_folder\_path, start=152, end=155)  
test\_feature\_vectors = get\_feature\_vectors(test\_emails, vectorizer)  
  
# 进行预测  
predictions = model.predict(test\_feature\_vectors)  
test\_emails\_name=[f'{k}.txt' for k in range(152,156)]  
for email, prediction in zip(test\_emails\_name, predictions):  
 print(f"邮件: {' '.join(email)}")  
 print(f"预测结果: {prediction}")  
 print()



股票预测

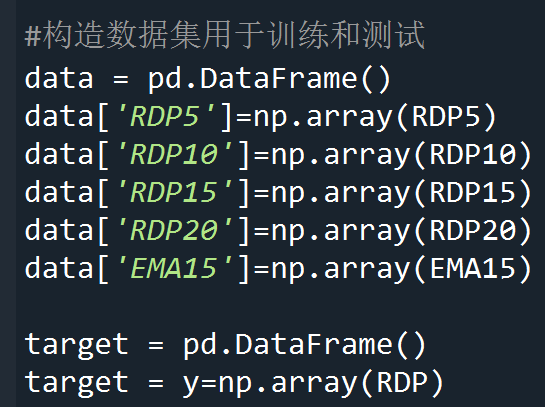
1、数据来源：中国平安.csv给出中国平安A股（601318）2017年以来的股票数据。读取ch15\_PinganStock.csv股票数据

2、本案例主要是对第5列的“收盘价”进行数据预处理。输入向量由4个基于5天预测时段的收益率（RDP5、RDP10、RDP15、RDP20）与转变后的收盘价（EMA15）5个分量组成。EMAn通过当天收盘价减去该天前n天的价格指数滑动平均值而获得。输出变量（RDP）是首先分别将当天与其后第5天的原始收盘价转换为各自前3天的指数滑动平均值，然后再根据转换的新值求收益率。

计算公式

|  |  |
| --- | --- |
| 变量名称 | 计算公式 |
| RDP5 |  |
| RDP10 |  |
| RDP15 |  |
| RDP20 |  |
| EMA15 |  |
| RDP |  |
| 注 | 表示第i天之前n天的指数滑动平均值，p(i)表示第i天的收盘价。。 |

最后构造的数据集为

c

划分训练集和测试集9：1比例

# 训练集324天（2017-11-07至2019-03-07），测试集37天（2019-03-08至2019-04-30）

选择一种回归模型进行训练，输出测试集的性能指标，并对预测结果和真实值进行对比可视化

import pandas as pd  
import numpy as np  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
  
# 读取数据  
file\_path = 'ch15\_PinganStock.csv'  
data = pd.read\_csv(file\_path,encoding='gbk')  
  
# 计算EMA15  
def calculate\_ema(prices, days):  
 return prices.ewm(span=days, adjust=False).mean()  
  
data['EMA15'] = calculate\_ema(data['收盘'], 15)  
  
# 计算RDP5, RDP10, RDP15, RDP20  
def calculate\_rdp(prices, days):  
 return prices.pct\_change(periods=days).shift(-days)  
  
data['RDP5'] = calculate\_rdp(data['收盘'], 5)  
data['RDP10'] = calculate\_rdp(data['收盘'], 10)  
data['RDP15'] = calculate\_rdp(data['收盘'], 15)  
data['RDP20'] = calculate\_rdp(data['收盘'], 20)  
  
# 计算目标变量RDP  
def calculate\_target(prices, days):  
 ema\_current = prices.ewm(span=3, adjust=False).mean()  
 ema\_future = prices.shift(-days).ewm(span=3, adjust=False).mean()  
 return (ema\_future - ema\_current) / ema\_current  
  
data['RDP'] = calculate\_target(data['收盘'], 5)  
# print(data.shape)  
# 去除缺失值  
data = data.dropna()  
# print(data.shape)  
# 构建输入特征矩阵和目标变量  
X = data[['RDP5', 'RDP10', 'RDP15', 'RDP20', 'EMA15']]  
y = data['RDP']  
# print(X)  
# 使用给定时间范围划分训练集和测试集  
# train\_end\_date = '2019-03-07'  
# test\_start\_date = '2019-03-08'  
#  
l=round(len(data)\*0.9)  
X\_train = X[:l]  
y\_train = y[:l]  
X\_test = X[l:]  
y\_test = y[l:]  
# print(X\_test.shape)  
# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1, shuffle=False,)  
# 训练模型  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# 预测  
y\_pred = model.predict(X\_test)  
  
# 评估模型  
mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f'Mean Squared Error: {mse}')  
print(f'R^2 Score: {r2}')  
  
# 对比预测结果和真实值可视化  
plt.figure(figsize=(14, 7))  
plt.plot(y\_test.values, label='True Values')  
plt.plot(y\_pred, label='Predicted Values')  
plt.legend()  
plt.show()

