**南 京 师 范 大 学**

**《人工智能导论》**

**大作业报告**



**报告内容： 基于机器学习进行的股票价格预测**

**学 院： 计算机与电子信息学院/人工智能学院**

**专 业： 计算机科学与技术**

**小组成员： 问候，乔优，王沛，杨勇**

**指导老师： 宋歌**

计算机与电子信息学院/人工智能学院

二O二四年 6 月 17 日

目录

[[一]](#_Toc128387244)

数据处理 [3](#_Toc128387245)

[[二]](#_Toc128387247)

[模型构建 4](#_Toc128387248)

[[三]](#_Toc128387249)

[模型训练与优化 5](#_Toc128387250)

[[四]](#_Toc128387252)

[模型评估与预测结果分析 6](#_Toc128387253)

[[五]](#_Toc128387254)

[结果展示与分析 7](#_Toc128387257)

[[六]](#_Toc128387254)

[结论 9](#_Toc128387257)

[[七]](#_Toc128387254)

[参考文献 11](#_Toc128387257)

**引言：**本报告基于股票代码“600519”（贵州茅台股票）的历史开盘价数据，使用深度学习模型（LSTM）进行股票价格预测。通过对训练集和测试集的数据预处理、模型构建和训练，评估了模型的性能并进行了预测，实验环境：anaconda jupyter。

一. 数据预处理

贵州茅台股票数据来源于“600519.csv”文件，包含每日开盘价等信息。数据预处理阶段的步骤如下：

* **数据加载与选择特征：** 使用Pandas库加载CSV文件，并提取开盘价作为预测的目标特征。pd.read\_csv("600519.csv") 命令将CSV文件加载到一个名为df的pandas DataFrame中。而df = df[['tradeDate', 'openPrice']] 从原始数据集df中选择了'tradeDate'（可能是日期或时间戳）和'openPrice'两列。
* **划分训练集和测试集：** 将数据集分为训练集（前3000个数据点）和测试集（剩余数据点），以便进行模型训练和验证。3000个数据点对于许多机器学习模型而言，提供了一个相对充分的训练样本数量。
* **数据归一化处理：** 使用MinMaxScaler对开盘价数据进行归一化处理，将数据缩放到0到1的范围内，有利于模型更快速地收敛和提高预测准确性。使用与训练集相同的缩放参数（从训练集中计算得出）对测试集(test\_set)进行缩放，确保在两个数据集之间的缩放保持一致。

import pandas as pd  
from sklearn.preprocessing import MinMaxScaler  
  
df = pd.read\_csv("600519.csv")   
df = df[['tradeDate', 'openPrice']]  
  
training\_set = df.iloc[0:3000, 1:2].values # 训练集  
test\_set = df.iloc[3001:, 1:2].values # 测试集  
  
sc = MinMaxScaler(feature\_range=(0, 1))  
training\_set\_scaled = sc.fit\_transform(training\_set)  
test\_set\_scaled = sc.transform(test\_set)

二. 模型构建

为了有效地捕捉时间序列数据中的长期依赖关系，我们选择了长短期记忆网络（LSTM）作为预测模型的核心组件。模型的结构如下所示：

**模型层次设计：** 模型采用Sequential顺序模型，依次堆叠了两层LSTM层和Dropout层，最后接一个全连接层作为输出层。

第一层是LSTM层，包含80个单元，并设置为返回完整的序列，即return\_sequences=True。这意味着该层会返回每个时间步长的输出，而不是仅返回最后一个时间步长的输出。

第二层是一个Dropout层，Dropout是一种正则化技术，有助于防止过拟合，它随机地将输入单元的一部分设置为0，一般设置为0.2表示20%的输入单元被随机失活。

第三层是另一个LSTM层，包含100个单元，不返回完整的序列，因为默认情况下return\_sequences为False，所以该层只返回最后一个时间步长的输出。

最后是一个Dense层，输出一个值，用于回归问题中的预测。

**Dropout层的作用：** Dropout层在模型训练过程中以一定的概率随机丢弃部分神经元的输出，从而减少神经元之间的相互依赖关系，有助于防止过拟合。通过随机失活一部分神经元，模型更能够泛化到未见过的数据，并且更不容易受到训练集中的噪声影响。

model=tf.keras.Sequential([  
 LSTM(80, return\_sequences=True),  
 Dropout(0.2),  
 LSTM(100),  
 Dropout(0.2),  
 Dense(1)  
])

模型使用Adam优化器，并以均方误差作为损失函数进行编译。

三. 模型训练与优化

模型训练是整个预测流程中的关键步骤，我们采用了以下策略来优化模型的训练效果：

**模型编译与优化器选择：**

·使用model.compile方法对模型进行编译配置，以便进行训练前的准备工作。

·优化器选择:选择了Adam优化器(tf.keras.optimizers.Adam)，并设置学习率为0.001 (optimizer=tf.keras.optimizers.Adam(0.001))。

**模型检查点设置：** 使用ModelCheckpoint回调函数，保存在验证集上损失最低的模型权重，避免训练过程中模型性能的退化。检查是否已经存在指定路径的模型权重文件，如果存在，则加载已有的权重。

**训练过程监控与可视化：** 在每个epoch结束时，监控并绘制训练集和验证集的损失曲线，以便及时调整模型参数和监测训练进度。

model.compile(optimizer=tf.keras.optimizers.Adam(0.001),  
 loss='mean\_squared\_error')  
  
checkpoint\_save\_path = './checkpoint/stock.weights.h5'  
  
# 如果已经存在模型权重文件，则加载模型  
if os.path.exists(checkpoint\_save\_path + '.index'):  
 print('-------------load the model------------------')  
 model.load\_weights(checkpoint\_save\_path)  
  
cp\_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_save\_path,  
 save\_weights\_only=True,  
 save\_best\_only=True,  
 monitor='val\_loss')  
  
history = model.fit(x\_train, y\_train, batch\_size=64, epochs=50,  
 validation\_data=(x\_test, y\_test),  
 validation\_freq=1, callbacks=[cp\_callback])

通过50个epoch的训练，每次epoch结束时保存验证集上损失最低的模型权重。训练过程中监控并绘制了训练和验证损失曲线，以便对模型训练效果进行可视化和分析。

四．模型评估与预测结果分析

训练完成后，我们对模型进行了评估并进行了股票价格的预测。首先，通过函数predicted\_stock\_price = model.predict(x\_test)，使用训练好的模型对测试集 x\_test 进行预测，得到预测的股价。

然后对预测的股价数据进行反向缩放，将其转换回原始数据的规模。再对测试集中实际的股价数据进行反向缩放，同样将其转换回原始数据的规模。

接下来，通过三个方法函数来计算均方误差 (mse), 均方根误差 (rmse) 和平均绝对误差 (mae)。

mean\_squared\_error(predicted\_stock\_price, real\_stock\_price): 计算预测股价与真实股价之间的均方误差，衡量预测值与真实值的平方差的均值。

math.sqrt(mse): 将均方误差转换为均方根误差，以便更直观地理解误差量级。

mean\_absolute\_error(predicted\_stock\_price, real\_stock\_price): 计算预测股价与真实股价之间的平均绝对误差，用于衡量预测值与真实值之间的平均绝对差异。

predicted\_stock\_price = model.predict(x\_test)  
predicted\_stock\_price = sc.inverse\_transform(predicted\_stock\_price)  
real\_stock\_price = sc.inverse\_transform(test\_set\_scaled[60:])  
  
mse = mean\_squared\_error(predicted\_stock\_price, real\_stock\_price)  
rmse = math.sqrt(mse)  
mae = mean\_absolute\_error(predicted\_stock\_price, real\_stock\_price)

接着，绘制了真实股票价格与预测股票价格的对比图，展示了模型在测试集上的预测效果。

plt.plot(real\_stock\_price, color='red', label='Real Stock Price')  
plt.plot(predicted\_stock\_price, color='blue', label='Predicted Stock Price')  
plt.title('Stock Price Prediction')  
plt.xlabel('Time')  
plt.ylabel('Stock Price')  
plt.legend()  
plt.show()

这段代码用于将模型预测的股价与真实股价进行对比可视化，通过线图清晰地展示了两者之间的差异和趋势。

通过观察图表，可以直观地评估模型的预测能力。如果预测线（蓝色）与真实股价线（红色）高度吻合，说明模型的预测效果较好；如果有较大偏差，则需要进一步分析和改进模型。

图表标题和轴标签提供了对图表内容的描述，使读者能够理解图表所呈现的信息及其含义。

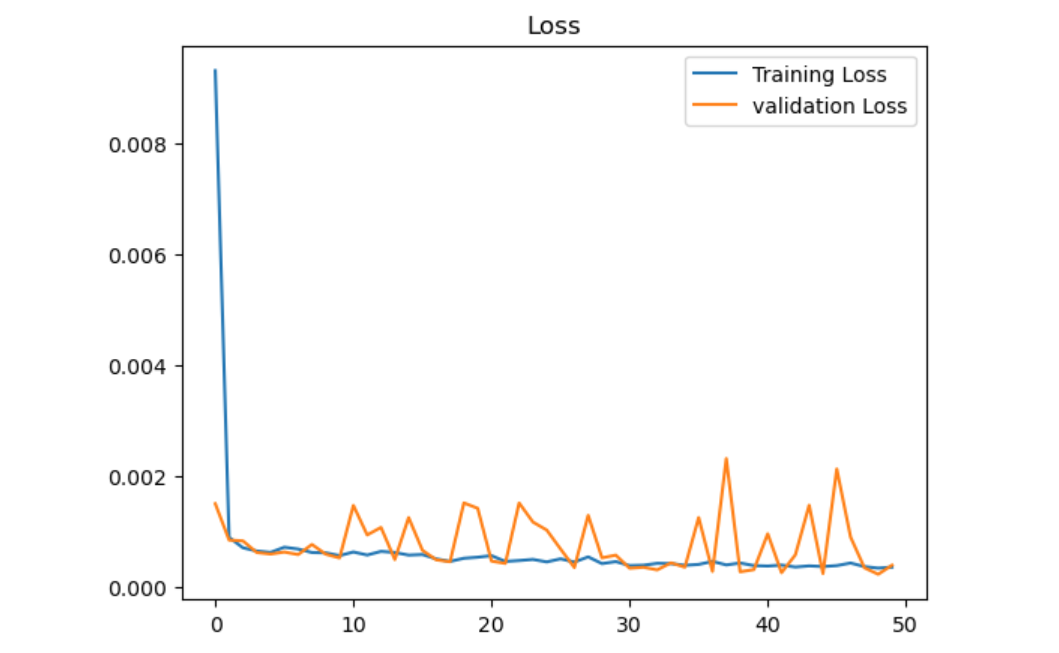
通过可视化方法进行结果分析是评估时间序列预测模型常用的手段之一，能够直观地向利益相关者展示模型的预测准确性和趋势捕捉能力

五. 结果展示与分析

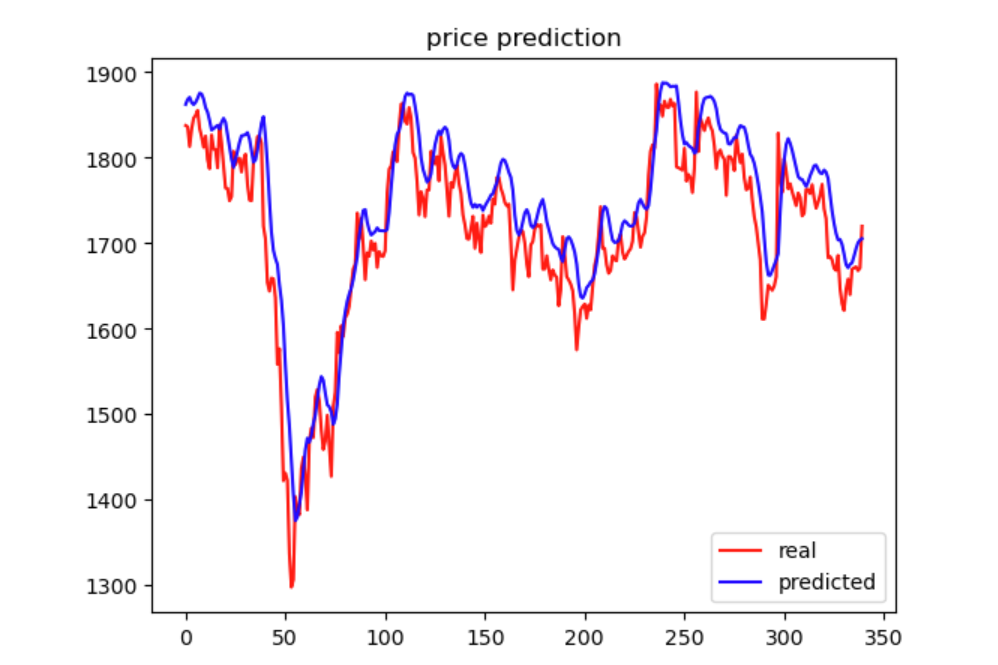
**·功能分析：**

代码首先加载了股票数据，并将其分为训练集和测试集。使用MinMaxScaler对训练集和测试集进行了数据缩放。构建了一个包含两个LSTM层和一个全连接层的序列模型，用于股价预测，并编译了模型使用Adam优化器和均方误差损失函数。使用ModelCheckpoint回调函数来保存验证集上损失最低的模型权重。

训练模型使用了训练集和验证集数据，并在每个epoch后验证模型的性能。在模型训练完成后，通过预测测试集数据得到了预测股价，然后反向缩放得到原始数据的预测结果。最后，通过均方误差、均方根误差和平均绝对误差来评估预测结果的准确性。



图一



图二

其中图一显示了预测的趋势与实际趋势的误差，图二显示了预测的股票趋势，并与实际的股票趋势做了对比。

图表显示了真实股价（红色线）和预测股价（蓝色线）随时间的变化。预测股价线与真实股价线非常接近，且两者的趋势相似，则说明模型在该测试数据集上具有良好的预测能力。

趋势图直观地展示模型的预测效果。评估指标（如均方误差、均方根误差和平均绝对误差）提供了定量的模型性能指标，帮助确定模型的预测精度和偏差水平。

六. 结论

本实验旨在利用深度学习模型（具体为包含LSTM层的序列模型）对股票价格进行预测，并通过多种评估指标对模型的预测性能进行全面分析。以下是对实验结果的总结和结论：

**实验步骤回顾：**

**·数据预处理：**

使用Pandas库加载股票数据，并将其分为训练集和测试集。对训练集和测试集进行MinMaxScaler缩放，将数据转换到[0, 1]的范围内，以便于模型训练和预测。

**·模型构建与训练：**

设计了一个包含两个LSTM层和一个全连接层的序列模型，用于时间序列的股票价格预测。使用Adam优化器和均方误差损失函数编译模型。利用ModelCheckpoint回调函数保存在验证集上损失最低的模型权重。在训练过程中，使用了训练集和验证集数据，并在每个epoch后评估模型的性能。

**·模型评估：**

使用测试集数据进行股票价格的预测。将预测结果反向缩放到原始数据的范围。计算均方误差（MSE）、均方根误差（RMSE）和平均绝对误差（MAE），作为评估模型预测准确性的指标。

**·结果可视化：**

绘制了真实股票价格和模型预测股票价格随时间变化的线图。通过图表直观地比较了模型预测结果与真实数据的一致性和趋势。

**模型改进建议：**

虽然当前模型表现良好，但仍需考虑以下改进方向：

**·模型复杂度:**

考虑增加模型的复杂度或深度，例如增加更多的LSTM层或神经元数量，以提升模型的表达能力和学习能力。

**·数据增强:**

考虑引入更多的数据增强技术，如滑动窗口、时间序列特征工程等，以提升模型对复杂市场变化的适应能力。

**·超参数调整:**

进行更系统和深入的超参数调整，例如学习率、批量大小、Dropout率等，以进一步提高模型的训练效果和泛化能力。

**·特征选择:**

考虑引入更多相关的市场因素和宏观经济数据作为输入特征，以增强模型的预测能力和稳定性。

**·模型评估:**

进一步扩展模型评估指标的范围，如交叉验证、时间序列交叉验证等，以更全面地评估模型在不同市场环境下的稳定性和预测准确性。

**实验应用性：**

本实验展示了利用深度学习模型进行股票价格预测的全面流程，并成功应用于实际数据集。这种方法不仅仅适用于股票价格预测，还可以扩展到其他时间序列预测问题，如销售预测、天气预测等。

**·泛化能力:**

深度学习模型在数据量充足的情况下通常表现出色，能够处理复杂的非线性关系，适用于多种市场环境下的股票价格预测。

**·实时性:**

可以通过适当的数据更新策略实现实时预测，帮助投资者和决策者快速响应市场变化。

**·可解释性:**

结合可视化和评估指标，能够直观地展示预测结果，有助于决策者理解市场趋势并制定有效的投资策略。

**·应用场景:** 不仅限于金融领域，还可以应用于供应链管理、需求预测和资源优化等领域，提升决策过程的精确性和效率。

七. 参考文献

TensorFlow官方文档: https://www.tensorflow.org/api\_docs

Keras官方文档: https://keras.io/api/

贵州茅台公司历史股价数据

组长：19220109问候（代码实现，数据收集）

组员：19220117杨勇（报告讲解）

19220303王沛（大作业报告编写）

19220207乔优（ppt制作）