**北京邮电大学计算机学院（国家示范性软件学院）**

**2023-2024 学年第 2 学期实验报告**

**课程名称： 人工智能基础**

**实验名称： 股票预测**

**实验完成人：**

**姓名：**\_\_ 胡翔宇 \_\_**学号：**\_\_2021211448

**姓名：**\_\_\_徐楷丁 \_\_**学号：**\_\_2021211434

**姓名：**\_\_\_杨超杰 \_\_**学号：**\_\_2021211432

**姓名：**\_\_\_马骐荣 \_\_**学号：**\_\_2021211840

**指导教师：**  **高慧**

**日 期： 2024年 6月 15日**

1. **实验目的**

实验的目的是通过完成一个回归实验，来熟悉人工智能领域中机器学习技术的应用，特别是回归分析方法，并通过实际操作加深对模型构建、训练和评估的理解。同时，通过预测股票走势的实验，探索不同机器学习框架的应用，提高编程能力和实验报告撰写技巧，确保模型预测的准确性，并培养团队合作能力。

1. **实验内容**

同学们自行完成回归实验，以此来熟悉人工智能中机器学习下方法的实现

实验形式——分组进行，组员自定

代码不限——来源、语言

框架不限——Tensorflow、Pytorch、Caffe

数据集——股票走势（会给）

均方根误差0.2

提交形式——Word和代码（打包），实验报告中附上所有组员姓名和学号，开发环境参数，代码框架和性能截图等必要解释

拓展：

尝试用一个网络预测多个维度并且准确率合格——加1-5分

尝试使用Transformer（encoder/decoder）并且合格——加1-5分

1. **实验环境**

硬件：GP66，RTX3070，I7-11800H，32GB ram

软件：VS code，python 3.12，pytorch，matplotlib，numpy，pandas，sklearn等

1. **实验过程描述**

**1.整体流程概述：**

实验流程图如下所示：

图示

描述已自动生成

下面将根据流程来进行解释：

**2. 数据预览**

在数据预览阶段，首先我们使用pandas读取csv：

文本, 表格

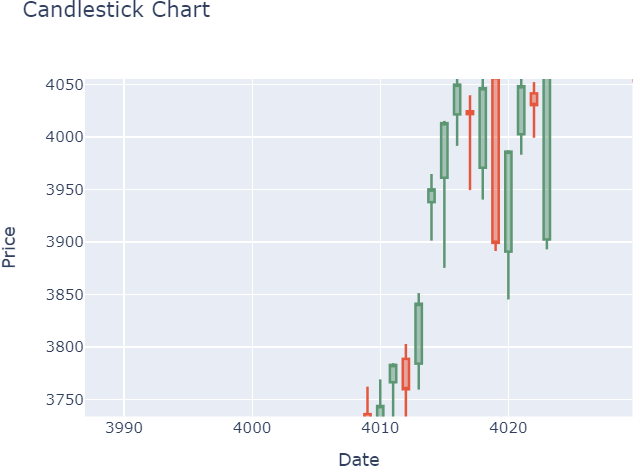
中度可信度描述已自动生成

我们调用plotly库，使用数据集中的open, high, low, close画出股票的蜡烛图:

图表, 折线图, 直方图

描述已自动生成

放大股票的蜡烛图，我们可以看到当天股票的各项走势柱状图：



使用matplotlib，可以画出数据集中所有的数据走势如下：

图表, 折线图

描述已自动生成

整体来看，股票成上升趋势，而中间部分经历了一个巨大的牛市和熊市。

然后我们使用MA来评估最佳的lookback值，移动平均线（Moving Average，简称MA）是一种在技术分析中常用的趋势指标，它通过计算一定时间周期内的平均价格来帮助投资者识别股票价格的趋势。移动平均线的作用主要包括：

平滑价格数据：移动平均线通过平均化价格，减少短期波动带来的噪音，使长期趋势更加明显。

识别趋势：当价格在移动平均线之上时，通常被解释为上升趋势；当价格在移动平均线之下时，则可能表示下降趋势。

交易信号：移动平均线的交叉可以作为买入或卖出的信号。例如，短期移动平均线上穿长期移动平均线，可能被视为买入信号；反之，则可能是卖出信号。

支撑和阻力：移动平均线有时也被视为价格的潜在支撑或阻力水平

图表, 折线图, 直方图

描述已自动生成

这里我们选取lookback等于10，在防止过拟合的同时，对数据的评估效果较好。

**3. 数据预处理**

首先查看数据集中是否有空值：



手机屏幕的截图

描述已自动生成

可以看到数据没有空值，可以进行下一步处理

然后将数据集归一化到范围[0,1]:

文本

描述已自动生成

按照时间序列**划分数据集**为训练集，测试集，比例为8：2，lookback=10，将data按lookback分组，data为长度为lookback的list，list共有data\_raw-lookback项。

其中X作为每10个数据集的前9个，最后1个为y，也就是预测的对象

文本

描述已自动生成

划分后的数据集格式如下所示：

文本

描述已自动生成

然后将各个数据转化为torch的tensor格式：

文本

描述已自动生成

定义输入特征为：



原本的数据集还有date日期和label第二天的最高价，由于数据已经按照日期排序好了，并且本来就是预测数据集第二天的数据，所以在预测多维输出的网络中，把这两个数据不作为输入，删除掉。

输出也是相同的特征值，这里用到了**一个网络预测多维**

**4. 使用LSTM建模并预测评估**

LSTM（Long Short-Term Memory）模型是一种特殊类型的循环神经网络RNN，它被设计用来解决传统RNN在处理长序列数据时遇到的长期依赖问题，以下是它的特别之处：

门控机制：LSTM引入了三个门控单元——遗忘门（forget gate）、输入门（input gate）和输出门（output gate），这些门控单元可以学习数据中哪些信息是重要的，哪些可以忽略。

遗忘门：遗忘门决定在新信息到来之前，哪些旧信息需要被遗忘。它通过选择性地丢弃或保留之前的记忆单元状态来更新记忆。

输入门：输入门有两个部分，一个是更新记忆单元状态的候选值（candidate values），另一个是决定这些候选值更新程度的sigmoid层。

输出门：输出门决定最终的输出，即当前的记忆单元状态和输入门的更新值如何结合，形成LSTM单元的最终输出。

定义LSTM模型如下：

屏幕上有字

描述已自动生成

设置训练的参数如下：

文本

描述已自动生成

然后使用loss函数为MSE，优化器为Adam优化器，初始lr为0.01

文本

描述已自动生成

最后用**测试集**进行评估，得到如下MSE与RMSE，满足实验要求：

文本

描述已自动生成



100 epoch



200 epoch

文本

描述已自动生成

600 epoch

文本

描述已自动生成

10000 epoch（过拟合）

600 epoch各维度的预测结果可视化如下：

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

可以看出对于大部分数据，LSTM的表现都比较好，比较贴合原始数据(除了原始数据出现剧烈震荡时)

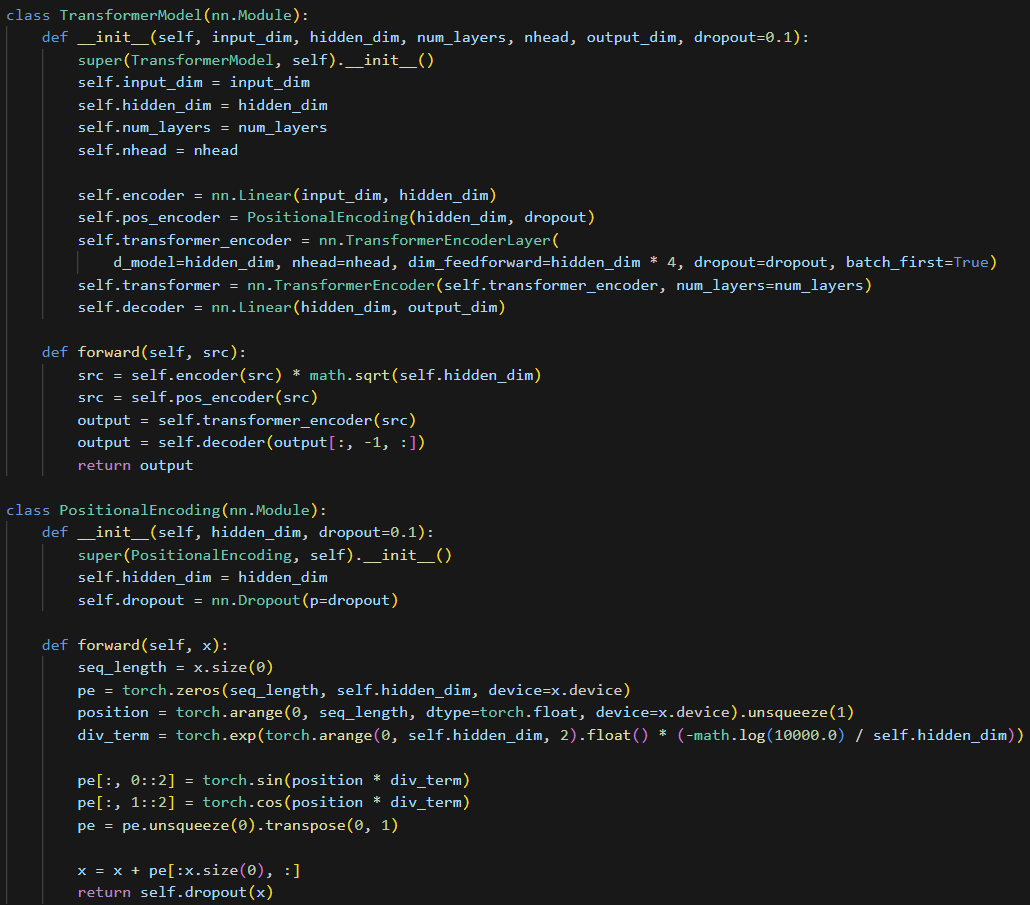
**5. 使用Transformer建模并预测评估**

Transformer是一种用于自然语言处理（NLP）和其他序列到序列（sequence-to-sequence）任务的深度学习模型架构，它在2017年由Vaswani等人首次提出。Transformer架构引入了自注意力机制（self-attention mechanism），这是一个关键的创新，使其在处理序列数据时表现出色

以下是Transformer的一些重要组成部分和特点：

* 自注意力机制（Self-Attention）：这是Transformer的核心概念之一，它使模型能够同时考虑输入序列中的所有位置，而不是像循环神经网络（RNN）或卷积神经网络（CNN）一样逐步处理。自注意力机制允许模型根据输入序列中的不同部分来赋予不同的注意权重，从而更好地捕捉语义关系。
* 多头注意力（Multi-Head Attention）：Transformer中的自注意力机制被扩展为多个注意力头，每个头可以学习不同的注意权重，以更好地捕捉不同类型的关系。多头注意力允许模型并行处理不同的信息子空间。
* 堆叠层（Stacked Layers）：Transformer通常由多个相同的编码器和解码器层堆叠而成。这些堆叠的层有助于模型学习复杂的特征表示和语义。
* 位置编码（Positional Encoding）：由于Transformer没有内置的序列位置信息，它需要额外的位置编码来表达输入序列中单词的位置顺序。
* 残差连接和层归一化（Residual Connections and Layer Normalization）：这些技术有助于减轻训练过程中的梯度消失和爆炸问题，使模型更容易训练。
* 编码器和解码器：Transformer通常包括一个编码器用于处理输入序列和一个解码器用于生成输出序列，这使其适用于序列到序列的任务，如机器翻译。

定义Transformer模型如下：



训练的过程与LSTM类似，同样是使用MSE作为loss函数，Adam作为优化器，lr设置为0.01，进行100个epoch训练，一个网络多维预测，最后在**测试集**对多维预测进行评估，得到其MSE与RMSE如下：

文本

描述已自动生成

文本

描述已自动生成

100 epoch

文本

描述已自动生成

600 epoch（过拟合）



10000 epoch（又没过拟合了？）

600 epoch各维度的预测结果可视化如下：

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

但是epoch=10000时各维数据的预测可视化如下：

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 折线图, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

对比可以看出，Transformer的预测结果震荡非常明显（但随着epoch变多震荡变小，说明需要更多epoch来训练模型），这与LSTM模型大不相同，并且根据RMSE作为基准评估，LSTM的效果更好。

**6. Train Loss下降对比**

图表

描述已自动生成

这里可以看出，LSTM模型的收敛速度更快，由此可以发现，不仅其预测MSE更小，收敛速度也相比Transformer更快。

**7. 改进**

尝试加入学习率下降机制：学习率下降（Learning Rate Decay）是深度学习训练过程中常用的一种技术，用于调整优化算法中的学习率。学习率是控制模型权重更新幅度的超参数，其下降机制的主要目的是：

* 避免震荡：在训练初期，较大的学习率有助于模型快速收敛，但随着训练的进行，过大的学习率可能导致损失函数值震荡，难以收敛到最小值。
* 精确逼近：随着训练的深入，逐渐减小学习率可以使模型更精细地逼近损失函数的最小值，提高模型的精确度。
* 防止过拟合：适当降低学习率可以减缓权重的更新速度，有助于模型在训练数据上不过分拟合。

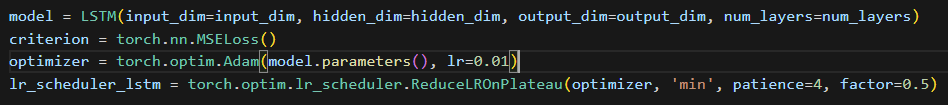
由于数据量并不大，所以在此我们使用patience的方式基于性能的衰减，如果连续超过patience次其valid\_loss达不到最佳loss，就降低其lr

为了加入valid集，我们需要重新划分训练测试验证集，按照8：1：1的比例划分：

文本

描述已自动生成

然后在训练LSTM模型中，加入学习率调整：



文本

描述已自动生成

这里我们设置连续4次达不到最小loss，lr减半。

最后评估的结果如下：

文本

描述已自动生成

相比不使用学习率下降相比，这里的效果反而下降了。

取多维预测数据中的一维度数据预测结果（这里取high）来进行分析：

图表, 折线图, 直方图

描述已自动生成

可以看出，这里导致正确率下降的原因可能是：原本valid集处是测试集，测试集在此处表现较好，但是划分为验证集后，这个区域不算如测试集的评估中了。

1. **实验结果**

在本次实验中，我完整了进行了一次机器学习网络的搭建和预测，并对结果进行了评估和改进，进行可视化分析。

其中，完成了**两个网络**的搭建并预测，并使用了**一个网络预测多维**，均**远低于**要求的最大RMSE（0.2），完成了拓展的要求。

两个模型的最佳表现对比如下：

文本

描述已自动生成 文本

描述已自动生成

LSTM Transformer

此外，尝试了学习率下降策略（虽然效果不如不加好？）

**实验心得：**

通过本次实验，我深入理解了深度学习在时间序列预测中的应用，特别是在股票价格预测方面的实践。以下是我在实验过程中的一些关键体会：

* 数据预处理的重要性：数据的清洗、归一化和特征选择对模型性能有显著影响。我学会了如何使用MinMaxScaler等工具来规范化数据，并通过可视化来探索数据特征。
* 模型选择与理解：我实践了LSTM和Transformer两种不同的模型。通过实验，我认识到了LSTM在捕捉时间序列中的长期依赖关系方面的优势，同时也体会到了Transformer模型在处理序列数据时的灵活性和效率。
* 超参数调整：实验中我尝试了不同的超参数设置，包括学习率、批次大小、层数等。我发现合适的超参数选择对模型的训练效果至关重要。
* 学习率下降策略：我实施了不同的学习率下降机制，观察到它在训练过程中对模型收敛速度和最终性能的影响。
* 模型评估：我学会了使用均方误差（MSE）和均方根误差（RMSE）等指标来评估模型的预测性能，并使用可视化工具来比较原始数据和预测结果。
* 软件工具的掌握：通过本次实验，我加深了对Python编程语言、深度学习库（如PyTorch）和数据可视化工具（如matplotlib和seaborn）的掌握。
* 问题解决能力：在实验过程中遇到的问题，如过拟合、梯度消失等，促使我学习了多种解决方案，并提高了我的问题解决能力。
* 理论与实践的结合：这次实验加深了我对深度学习理论的理解，并将这些理论应用于实际问题解决中。
* 未来方向：实验结束后，我对如何进一步改进模型、探索新的网络结构和优化技术有了更清晰的方向。

总结来说，这次实验不仅提升了我的技术技能，也加深了我对深度学习在金融领域应用的认识。我期待将这些知识和技能应用到更广泛的领域中。

1. **附件**

实验代码详见附件