摘要

失业率作为一个关键的宏观经济指标，对于评估经济状况、制定政策和预测经济走势具有重要意义。本文研究对象是美国 1948 年 1 月到 2023 年 5 月的月度失业率数据，共计 905 条。建模前的检验结果表明数据具有平稳性，且不是白噪声数据。首先构建 ARIMA 模型，具体步骤包括基于 AIC 准则的模型定阶，模型参数估计，残差分析和结果预测。随后又搭建了隐马尔可夫模型 (HMM)，先知模型 (Prophet) 和长短记忆网络 (LSTM), 分别进行建模并给出未来 60 个月的预测结果。在模型表现方面，LSTM 模型的表现最佳，ARIMA 模型和 HMM 模型较优，Prophet 模型的表现则不太理想。四个模型给出的预测结果不尽相同，但均认为未来 60 个月内的失业率整体会呈现出增长趋势。

关键字: 失业率，ARIMA 模型，HMM 模型，Prophet 模型，LSTM 模型

Abstract

As a key macroeconomic indicator, the unemployment rate is of great significance for assessing economic conditions, formulating policies and forecasting economic trends. The research object of this paper is the monthly unemployment rate data from January 1948 to May 2023 in the United States, with a total of 905 records. The test results before modeling show that the data is stationary and not white noise data. Firstly, the ARIMA model is constructed, and the specific steps include model order determination based on the AIC criterion, model parameter estimation, residual analysis and result prediction. Then, the Hidden Markov Model (HMM), the Prophet Model (Prophet) and the Long-Short Memory Network (LSTM) were built, respectively, and the prediction results for the next 60 months were given. In terms of model performance, the LSTM model has the best performance, the ARIMA model and HMM model are better, and the Prophet model is not so ideal. The prediction results given by the four models are not the same, but they all indicate that the unemployment rate in the next 60 months will show an overall upward trend.

Keywords: Unemployment Rate, ARIMA Model, HMM Model, Prophet Model, LSTM Model

Contents

1 引言

时间序列分析在经济学和金融领域中扮演着重要的角色，帮助我们理解和预测经济变量的演变。失业率作为一个关键的宏观经济指标，对于评估经济状况、制定政策和预测经济走势具有重要意义。本文旨在通过对美国失业率时间序列的案例分析，探索失业率的月度变化模式、趋势以及未来走向。本文运用几种经典的时间序列模型，包括自回归移动平均模型（ARIMA）、隐马尔可夫模型（HMM）、先知模型（Prophet）和长短期记忆网络（LSTM），对失业率数据进行建模和预测。这项研究对于我们深入了解美国失业率的动态变化、提供有关未来失业率走势的预测，以及为决策者制定合适的政策和措施具有重要意义。

本文数据来源于数据科学竞赛平台Kaggle。共计 905 条数据，包含从 1948 年 1 月到 2023 年 5 月这期间的美国月度失业率信息。

2 预备知识

2.1 自回归移动平均模型 (ARIMA)

ARIMA 模型是一种常用于时间序列分析的统计模型。它结合了自回归 (AR)、差分 (I) 和移动平均 (MA) 过程，适用于具有一定程度的自相关和趋势的时间序列数据。ARIMA(p,d,q) 模型的公式表示如下：

$$(1-\phi_1 B-\phi_2 B^2-\ldots-\phi_p B^p)(1-B)^d X_t=(1+\theta_1 B+\theta_2 B^2+\ldots+\theta_q B^q)\omega_t$$

其中，X_t 是时间序列数据，在 t 时刻的值；B 是后移算子；p、d 和 q 分别代表自回归阶数、差分次数和移动平均阶数；$\phi_1,\phi_2,\ldots,\phi_p$ 和 $\theta_1,\theta_2,\ldots,\theta_q$ 是模型中的参数；ω_t 是误差项。

2.2 隐马尔可夫模型 (HMM)

隐马尔可夫模型是一种统计模型，用于建模具有隐藏状态的序列数据。HMM 假设隐藏状态和观测结果之间存在马尔可夫性质，通过观测结果推断隐藏状态。HMM 可以用以下公式表示：

$$P(O|\lambda)=\sum_{i=1}^{N}P(O,X=i|\lambda)=\sum_{i=1}^{N}P(O|X=i,\lambda)P(X=i|\lambda)$$

其中，$P(O|\lambda)$ 是给定模型参数 λ 下观测序列 O 的概率；$P(O,X=i|\lambda)$ 是给定模型参数 λ 下观测序列 O 和状态序列 X 的联合概率；$P(O|X=i,\lambda)$ 是给定模型参数 λ 下观测序列 O 在状态 i 下的概率；$P(X=i|\lambda)$ 是给定模型参数 λ 下状态 i 的概率。

2.3 先知模型 (Prophet)

Prophet 是一种用于时间序列预测的开源模型，由 Facebook 开发。它基于加法模型，将时间序列分解为趋势、季节性和假日效应，并使用非线性可加的趋势组件进行预测。Prophet 的公式表达如下：

$$y(t)=g(t)+s(t)+h(t)+\varepsilon_t$$

其中，$y(t)$ 是在时刻 t 的观测值；$g(t)$ 是趋势组件，描述数据的长期增长或下降趋势；$s(t)$ 是季节性组件，描述数据的周期性变化；$h(t)$ 是假日效应组件，考虑到假日对数据的影响；ε_t 是误差项。

2.4 长短记忆网络 (LSTM)

LSTM 是一种递归神经网络 (RNN) 的变种，用于处理和预测时间序列数据。相比于传统的 RNN，LSTM 具有较强的记忆能力，能够捕捉长期依赖关系。LSTM 使用门控机制来控制信息的流动和遗忘，其中包括输入门、遗忘门和输出门。LSTM 的公式如下：输入门（Input Gate）：

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

遗忘门（Forget Gate）：

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

细胞状态更新（Cell State Update）：

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

输出门（Output Gate）：

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

隐藏状态更新（Hidden State Update）：

$$h_t = o_t \cdot \tanh(c_t)$$

其中，x_t 是输入序列在时刻 t 的值；h_t 是隐藏状态（输出）；c_t 是细胞状态（记忆）；σ 是 sigmoid 函数；W 和 b 是模型参数。

3 探索性数据分析 (EDA)

3.1 时序图

观察序列可知 (Figure 1)，数据整体上具有平稳性，中心值在 6% 附近。值得注意的是，在 2020-2022 年期间，可能受新冠疫情的影响，失业率发生了一次急剧波动，达到最大值 15%，这也是美国近 80 年期间最高的失业率。

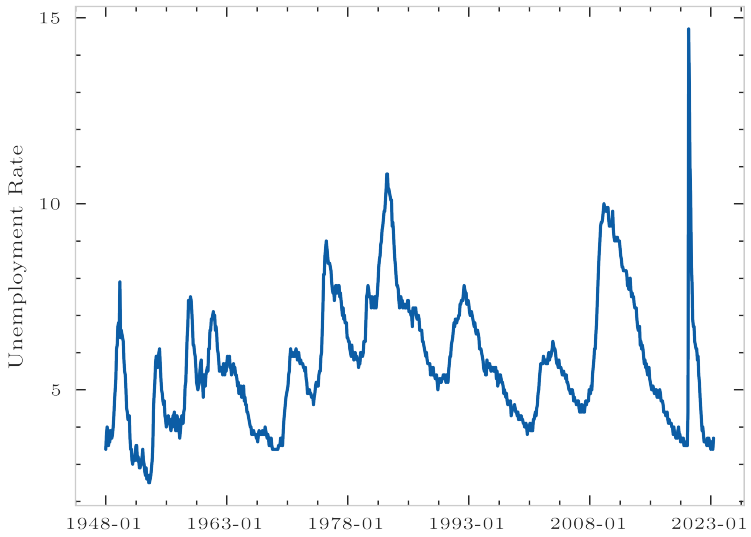


Figure 1: 美国失业率数据时序图 (1948-01 至 2023-05)

3.2 平稳性检验

时间序列中的平稳性是指统计特性在时间上保持稳定的性质。简而言之，平稳性意味着时间序列的统计特性（如均值、方差和自协方差等）不随时间变化而显著改变。接下来分别从可视化观察和统计性检验两个角度出发，对该序列的平稳性进行检验。

3.2.1 ACF 和 PACF

ACF (Autocorrelation Function) 衡量时间序列数据在不同滞后阶数上的自相关系数。PACF (Partial Autocorrelation Function) 衡量滞后阶数与当前观测值之间的直接相关程度。观察 Figure 2 和 Figure 3 发现，该序列很有可能是平稳序列。但仅从图形观察来判断序列是否平稳还不够严谨，因此需要借助统计量进行统计性检验。

3.2.2 ADF 单位根检验

单位根检验是一种用于检验时间序列数据是否具有单位根 (即非平稳性) 统计方法。其原理基于自回归模型的构建和参数估计。单位根检验的常用方法之一是增广迪基-弗勒检验 (Augmented Dickey-Fuller test，简称 ADF 检验)。ADF 检验的原假设是时间序列具有单位根，即时间序列是非平稳的。根据 Table 1 检验结果可知，p 值非常小，拒绝原假设，即美国失业率月度数据是个平稳的时间序列数据。

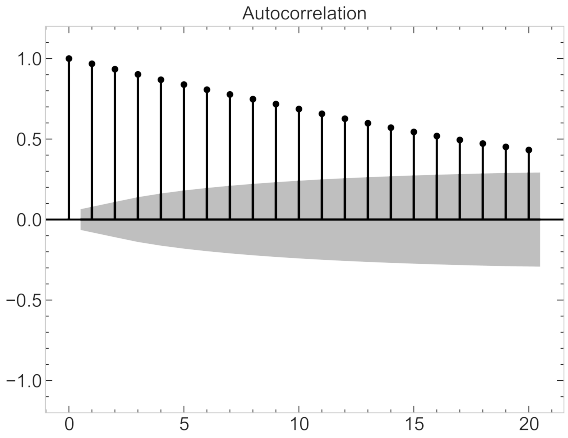


Figure 2: 自相关图

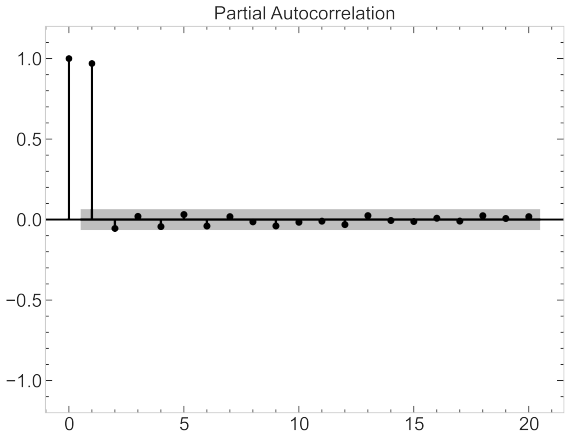


Figure 3: 偏自相关图

Table 1: ADF 单位根检验

ADF 统计量	p 值	1% 临界值	5% 临界值
-3.882	0.002	-3.438	-2.865

3.3 白噪声检验

白噪声检验是用于检验时间序列数据是否表现为白噪声的一种统计方法。白噪声是指具有无相关性和无自相关性的随机序列，其中每个观测值都是独立且具有相同的分布。Ljung-Box 检验是一种常用的白噪声检验方法之一，它基于时间序列数据的自相关函数（ACF）来评估数据的相关性。Ljung-Box 检验的原假设是数据是白噪声。对失业率数据进行 Ljung-Box 检验，得出检验统计量为 851.24，对应 p 值约为 10^{-180}, 即拒绝原假设，该序列不是白噪声。

4 模型构建

4.1 ARIMA 模型

4.1.1 基于 AIC 准则的模型定阶

在建立 ARIMA 模型时，首先要考虑 ARIMA 模型的阶数问题，这一步可以使用 AIC 准则（赤池信息准则）进行模型选择。AIC 是一种衡量模型拟合优良性和复杂性的准则，其目标是在给定数据集上找到既能很好地拟合数据又具有较低复杂度的模型。AIC 值越小表示模型的拟合优良性较好且具有较低的复杂性，因此更适合选择作为最佳模型。AIC 的数学表达式如下：

$$\mathrm{AIC} = 2k - 2\ln(L)$$

其中，k 表示模型的参数数量，L 表示模型的最大似然函数值。

具体步骤是事先给定三个参数 (模型的阶数 p,d,q) 的范围，本文设定自回归项阶数 p 的范围是 0 到 5，差分阶数 d 的范围是 0 到 2，移动平均项阶数 q 的范围是 0 到 5。给定大致范围后，通过枚举法生成不同的模型去拟合数据，不同的模型所求出的 AIC 结果不同，AIC 取最小值时所对应的阶数即为最优的 ARIMA 模型阶数。最终求得模型的最佳阶数为 (2,0,1)。

4.1.2 模型检验

模型检验主要分为两个部分，分别是模型的参数检验和残差分析。Table 2 给出了模型的各个参数的估计值，以及估计值对应的 p 值，不难看出，p 值均显著，故可认为给出的模型参数的估计值均有效。

Table 2: 模型参数估计

	coef	std err	z	P>|z|
const	5.5804	0.748	7.464	0.000
ar.L1	0.2317	0.098	2.365	0.018
ar.L2	0.7141	0.102	7.023	0.000
ma.L1	0.7972	0.096	8.346	0.000
sigma2	0.1707	0.001	142.344	0.000

那么可以得出模型的表达式为：

$$x_t = 5.584 + 0.2317x_{t-1} + 0.7141x_{t-2} + 0.7972\omega_{t-1} + \omega_t$$

接下来对模型的残差进行分析，对残差序列进行白噪声检验，发现 p 值为 0.9，较大，故可认为残差序列是个白噪声序列，再结合残差序列图和残差 Q-Q 图 (Figure 4,Figure 5)，不难发现，除了个别异常值，残差整体上符合零均值、等方差、不相关且服从正态分布的假设。

这一系列结果表明，模型是有效的，这将为下一步进行的结果预测提供可靠的保证。

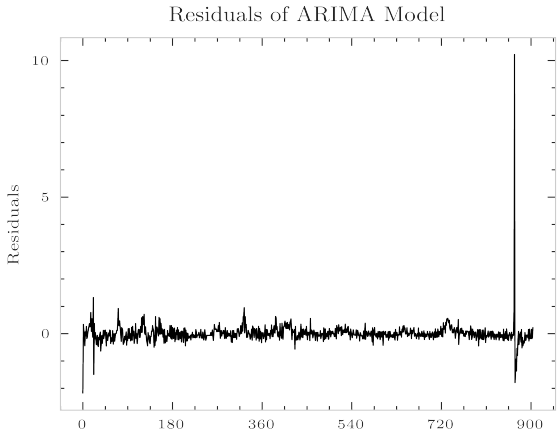


Figure 4: 残差序列图

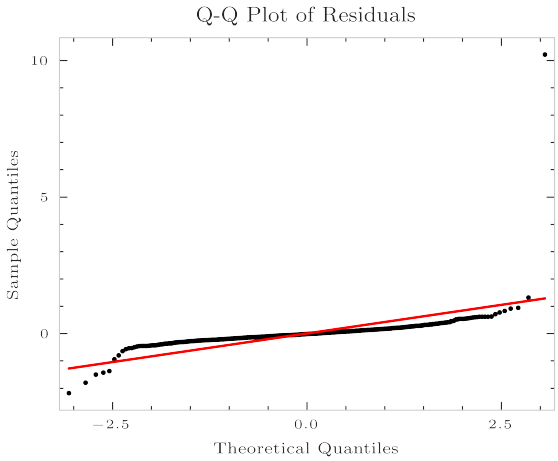


Figure 5: 残差 Q-Q 图

4.1.3 模型预测

根据训练出的模型，预测接下来 60 个月的月度失业率结果，预测结果以及模型拟合结果均可见于 Figure 6。ARIMA 模型预测结果显示，接下来 60 个月的失业率会呈现一个缓慢上升的趋势。

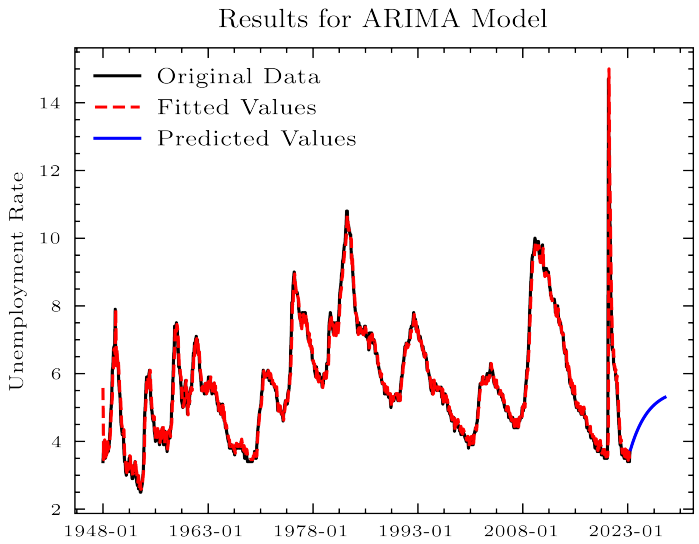


Figure 6: ARIMA 模型结果

4.2 HMM 模型

在进行隐马尔可夫模型求解的时候，本文假设观测状态服从一个多元正态分布，随后对数据进行拟合。拟合结果发现效果不佳，随后调整了模型中的超参数，超参数经过调整后确定为，隐藏状态数目为 10，最大迭代次数为 100。

随后模型给出接下来 60 个月的预测结果，结果显示，失业率整体上会持续保持增长，且增速较高，最大时可达 7.5%。见于 Figure 7。

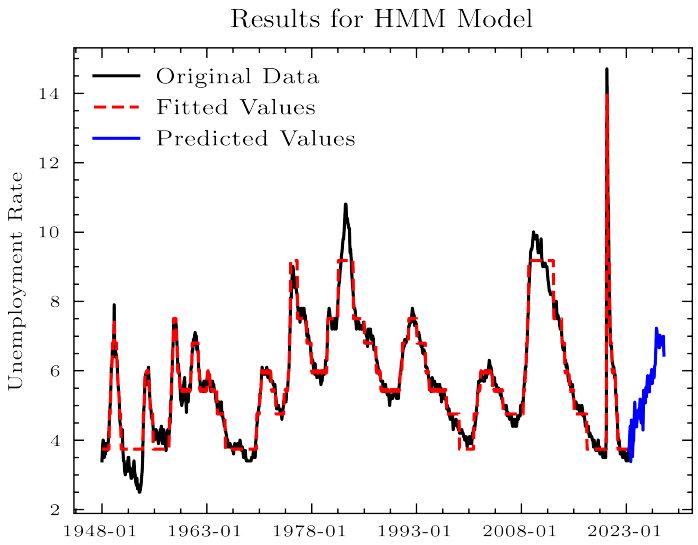


Figure 7: HMM 模型结果

4.3 Prophet 模型

首先，由于 Prophet 模型在求解的时候会将时间因素考虑进来，所以第一步将数据格式进行调整，共两列，一列原始数据，一列为标准日期格式 (如 2023-05)。在参数选择方面，设置函数以年规律进行拟合，且假设增长类型为线性形式。随后对数据进行拟合，拟合结果发现，该模型的灵敏度不高，仅能大体上反应失业率数据的一个变化趋势。

接下来给出未来 60 个月的预测结果，结果显示，未来的失业率会呈现周期波动，且波动范围越来越大，预测结果与前两个模型的结果大相径庭。见于 Figure 8。

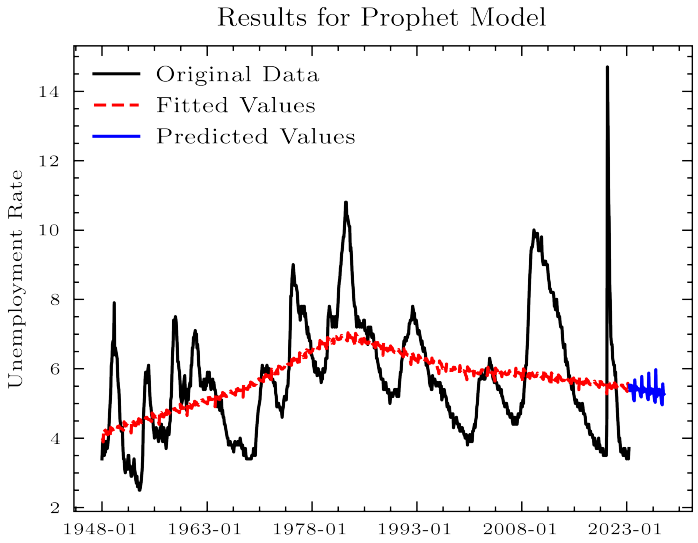
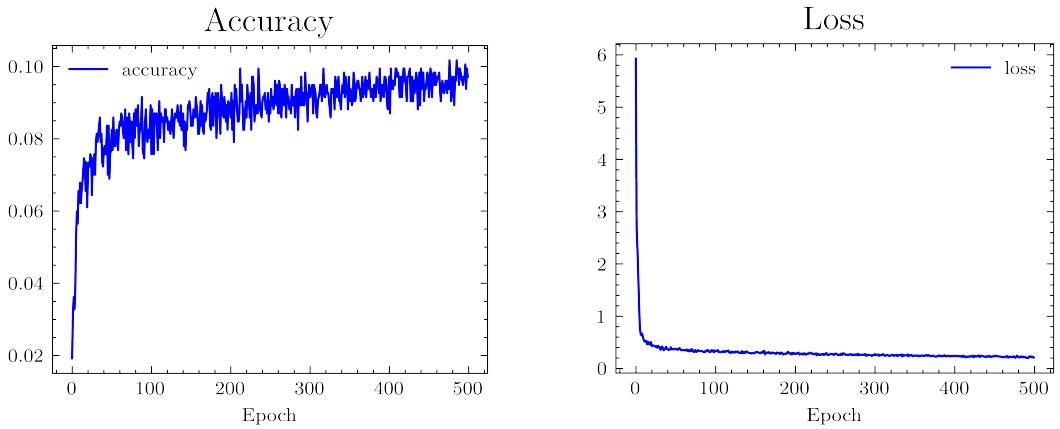


Figure 8: Prophet 模型结果

4.4 LSTM 模型

首先，搭建 LSTM 模型，考虑算力和运行时间，本文搭建的 LSTM 模型较为简易，共三层，第一层是 LSTM 层，有神经元 128 个，激活函数指定为 Sigmoid 函数。第二层是 Dropout 层，用于在训练过程中随机丢弃部分神经元，以减少过拟合的风险，参数 0.3 表示丢弃概率为 30%。第三层是连接层 (Dense 层)，用于输出预测结果，参数 1 表示输出的维度为 1。在模型训练超参数设置方面，指定损失函数为均方误差 (MSE)，优化器为 Adam 优化器，模型在训练过程中的评估指标为准确率 (accuracy)。

接下来是数据划分问题，本文设置序列长度为 20，也就是说，在拟合过程中认为当前月的失业率与刚刚过去 20 个月的失业率有关。最后将输入数据转换为 LSTM 所规定的三维张量形式，便可以进行模型的训练。观察训练过程可以看出，模型的准确率和损失分别在 400 次和 100 次后趋于稳定 (见于 Figure 9)，故本文将训练次数定为 500 次，累积运行时间约为 9 分钟。



(a) 准确度　　(b) 误差损失

Figure 9: LSTM 模型训练过程

模型拟合结果显示，拟合度较高。接下来进行预测，采用滚动预测的方式进行预测，结果显示，在未来 30 个月里，失业率会高速增长，随后增速减缓并达到最大值 6%，最后出现下降趋势。(见于 Figure 10)

5 模型比较

这一部分主要针对模型的表现和模型的结果这两个部分展开。

首先是模型的表现。MSE（Mean Squared Error）和 MAE（Mean Absolute Error）是常用的用于评价模型性能的指标。MSE 在评价模型时可以提供对平均误差的敏感度信息，而 MAE 则提供了对预测误差的一般性认知。MSE 是通过计算预测值与真实值之间差异的平方的平均值来衡量模型性能的指标。它的计算公式如下：

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

其中，y_i 是真实值，$\hat{y}_i$ 是模型的预测值，n 是样本数量。

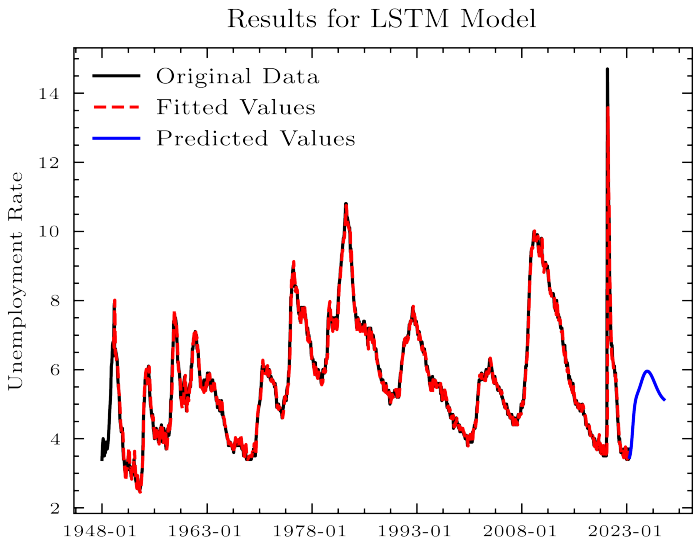


Figure 10: LSTM 模型结果

MADE 是通过计算预测值与真实值之间差异的绝对值的平均值来评价模型性能的指标。其计算公式如下：

$$MADE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

分别将不同模型的拟合数据与原始序列真实数据带入，分别求出不同的模型对应的 MSE 和 MADE，见于 Table 3。分析可知，从 MSE 这个评价指标看来，Prophet 模型的表现最差，HMM 模型的表现最佳；从 MADE 这个指标看来，仍是 Prophet 模型的表现最差，LSTM 模型的表现最佳。综合起来看，由于 HMM 模型的 MADE 较大，而 LSTM 模型的 MSE 与 HMM 模型的 MSE 相差较小，所以可大致得出结论，LSTM 模型的整体表现最佳。

Table 3: 模型的评价指标

模型	ARIMA	HMM	Prophet	LSTM
MSE	0.1757	0.1224	2.2197	0.1492
MADE	0.1753	0.2546	1.1600	0.1535

接下来就模型的结果进行比较。观察 Figure 11 的拟合数据可知，除了 Prophet 模型的拟合精度较差之外，其他模型的拟合效果均较优，从图形上看来，拟合数据对应的曲线基本上与原始重合。而对于预测值，四个模型给出的结果则差别很大，ARIMA 模型给出的预测值呈缓慢上升趋势，HMM 模型给出的预测值则以较快的速率波动式上升，Prophet 模型给出的结果则在 5.5% 附近上下波动，LSTM 模型则表明未来 60 个月的失业率会在上升一段时间后缓慢下降，见于 Figure 12。

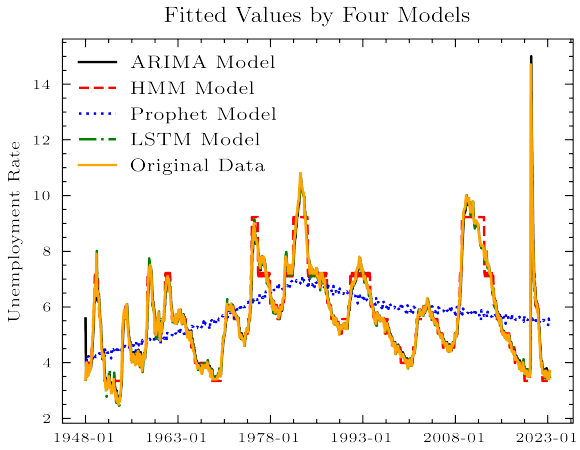


Figure 11: 不同模型的拟合值比较

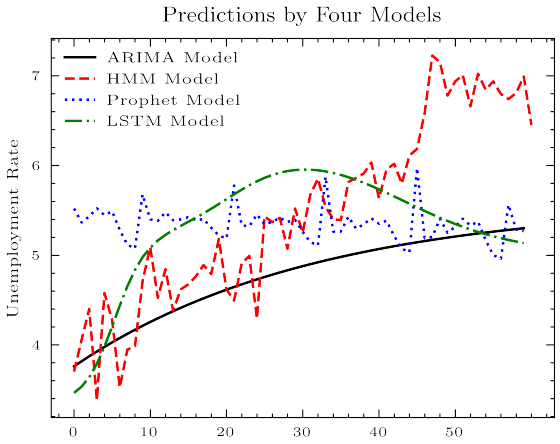


Figure 12: 不同模型的预测值比较

6 总结

- 预测结果

  四个模型给出的预测结果差别很大，但均表明未来的失业率不会下降，在未来近 10 个月的失业率的增速不会太快，整体看来，仍保持在 4% 附件。但在一年后，失业率的增速会加快，最大值可达 7%。整体看来，未来 60 个月的失业率均值约为 5

- 政策建议

  预测结果均表明接下来 60 个月的失业率会呈现上升趋势，且由 HMM 模型提供的预测结果表明最高可达 7%，所以应采取适当政策来应对失业率上升。政府可以鼓励创造就业机会的政策、提供培训和教育的计划、支持中小型企业，同时也应该考虑加大基础设施投资。

- 研究的局限性

  首先是模型的局限性，仅考虑了几个经典的模型，可能存在其他更优的模型来建模分析这组失业率数据。其次是模型参数的选取问题，很多超参数都是手动调参，可适当考虑贝叶斯优化、模拟退火等调参方法。最后是计算机算力问题，考虑到运行成本，搭建 LSTM 模型的结构比较简易，可能在训练的时候没有学习到数据的所有特征。

A 附录

A.1 Python 代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scienceplots
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm
import scipy.stats as stats
import sys
from itertools import product
from statsmodels.tsa.arima.model import ARIMA
from hmmlearn import hmm
from prophet import Prophet
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import Activation
# 定义模型评价指标函数
def MSE(a, b):
    c=a-b
    d=c*c
    sum =np.mean(d)
    return sum
def MADE(a,b):
    c=a-b
    d=abs(c)
    sum=np.mean(d)
    return sum
# 读取数据
data=pd.read_excel(r"C:\Users\LX\Desktop\时间序列\案例分析报告\
    ↪ usunemployment.xlsx")
data.head()
# 数据预处理, 格式调整
df=pd.melt(data,id_vars=['Year'],var_name='Month',value_name='Value')
```

```
df['Date']=pd.to_datetime(df['Year'].astype(str)+'-'+df['Month']).dt.
    ↪ strftime('%Y-%m')
df=df.drop(['Year', 'Month'],axis=1)
df=df.sort_values('Date')
df=df.iloc[:, ::-1]
df=df.dropna(axis=0)
print(df)
df.to_excel('C:\\Users\\LX\\Desktop\\时间序列\\案例分析报告\\dataset.
    ↪ xlsx', index=True)
newdf=pd.read_excel(r"C:\Users\LX\Desktop\时间序列\案例分析报告\
    ↪ dataset.xlsx")
newdf=newdf.iloc[:,1:3]
# 可视化
fig, ax = plt.subplots(figsize=(14,6))
sns.barplot(x='Date', y='Value', data=newdf, ax=ax)
plt.xticks(rotation=0)
plt.xticks(range(0, 905, 180))
plt.style.use('seaborn-whitegrid')
plt.show()
# 折线图
plt.style.use('science')
plt.plot(newdf['Date'],newdf['Value'])
plt.grid (False)
plt.ylabel('Unemployment Rate',fontsize=6)
plt.xticks(range(0, 905, 180),fontsize=5)
plt.yticks(fontsize=5)
plt.savefig('firstdata.png', transparent=True,dpi=1000)
plt.show()
plt.clf()

探索性数据分析 (EDA)
# 平稳性检验
# 单位根检验
result=adfuller(newdf['Value'])
print('ADF 统计量: ', result[0])
print('p 值: ', result[1])
print('临界值: ', result[4])
# 绘制ACF图
plt.style.use(['science','ieee','notebook'])
plot_acf(newdf['Value'], lags=20)
plt.ylim(-1.2,1.2)
```

```
plt.grid (False)
plt.savefig('ACF.png', transparent=True,dpi=1000)
plt.show()
# 绘制PACF图
plot_pacf(newdf['Value'], lags=20)
plt.ylim(-1.2,1.2)
plt.grid (False)
plt.savefig('PACF.png', transparent=True,dpi=1000)
plt.show()
# Ljung-Box检验
result_ljungbox=sm.stats.acorr_ljungbox(newdf['Value'],lags=10)  # 指
    ↪ 定滞后阶数
print('Ljung-Box检验结果: ')
print('统计量: ',result_ljungbox)
```

ARIMA

```
# 通过AIC对模型进行定阶
data=newdf['Value']
def find_best_arima_order(data):
    p_values=range(0, 5)
    d_values=range(0, 3)
    q_values=range(0, 5)
    best_aic=float('inf')
    best_order=None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p, d, q)
                try:
                    model = sm.tsa.ARIMA(data, order=order)
                    results = model.fit()
                    aic = results.aic
                    if aic<best_aic:
                        best_aic = aic
                        best_order = order
                except:
                    continue
    return best_order
best_order=find_best_arima_order(data)
print("最佳ARIMA阶数:",best_order)
# 模型训练
```

```
model=ARIMA(newdf['Value'],order=(2,0,1))
model_fit=model.fit()
# 得出拟合值
arima_fitted_values=model_fit.fittedvalues
# 模型评价
print("arimaMSE=",round(MSE(newdf['Value'],arima_fitted_values),4))
print("arimaMADE=",round(MADE(newdf['Value'],arima_fitted_values),4))
# 进行序列预测
forecast=model_fit.get_forecast(steps=60)
arima_predicted_values=forecast.predicted_mean
# 残差分析
print(model_fit.summary())
residuals = model_fit.resid
# 绘制残差序列图
plt.style.use(['science','ieee'])
plt.grid (False)
plt.plot(residuals,linewidth=0.5)
plt.title('Residuals of ARIMA Model',fontsize=8)
plt.ylabel('Residuals',fontsize=6)
plt.xticks(range(0, 905, 180),rotation=0,fontsize=5)
plt.yticks(fontsize=5)
plt.savefig('residual.png', transparent=True,dpi=1000)
plt.show()
plt.clf()
# 绘制残差的Q-Q图
plt.style.use(['science','ieee'])
sm.qqplot(residuals, line='s',linewidth=2.0,marker='o', markersize
    ↪ =0.8)
plt.grid (False)
plt.title('Q-Q Plot of Residuals',fontsize=8)
plt.yticks(fontsize=5)
plt.xticks(fontsize=5)
plt.ylabel('Sample Quantiles',fontsize=6)
plt.xlabel('Theoretical Quantiles',fontsize=6)
plt.savefig('residualQQ.png', transparent=True,dpi=1000)
plt.show()
plt.clf()
# 正态性检验
statistic, p_value = stats.shapiro(residuals)
print("Kolmogorov-Smirnov Test:")
print("Statistic:", statistic)
```

```
print("p-value:", p_value)
# 残差序列的白噪声检验
rrresult_ljungbox=sm.stats.acorr_ljungbox(residuals,lags=10)  # 指定滞
    ↪ 后阶数
print('Ljung-Box检验结果: ')
print('统计量: ',rrresult_ljungbox)
# 结果可视化
plt.style.use(['science','ieee'])
plt.grid (False)
plt.plot(newdf['Date'],newdf['Value'], label='Original Data')
plt.plot(arima_fitted_values, label='Fitted Values')
plt.plot(arima_predicted_values, label='Predicted Values',linestyle='-
    ↪ ')
plt.xticks(range(0, 905, 180),rotation=0,fontsize=5)
plt.yticks(fontsize=5)
plt.ylabel('Unemployment Rate',fontsize=6)
plt.legend(fontsize='small')
plt.title('Results for ARIMA Model',fontsize=8)
plt.savefig('ARIMA.png', transparent=True,dpi=1000)
plt.show()
```

HMM

```
# 数据准备
data=pd.read_excel(r"C:\Users\LX\Desktop\时间序列\案例分析报告\dataset
    ↪ .xlsx").iloc[:, 1:3]['Value']
train_data=data[:905]
train_data=np.array(train_data).reshape(-1, 1)
# 创建HMM模型对象
model=hmm.GaussianHMM(n_components=10,n_iter=100)
# 拟合模型
model.fit(train_data)
# 测未来的60个时间步
predicted_states, _=model.sample(n_samples=60)
predicted_values=predicted_states.flatten()
# 可视化
plt.style.use(['science','ieee'])
plt.grid (False)
plt.plot(newdf['Date'],newdf['Value'], label='Original Data')
hidden_states=model.predict(train_data)
hmm_fitted_values=model.means_[hidden_states].flatten()
plt.plot(hmm_fitted_values, label='Fitted Values')
```

```
start_value=train_data[-1]
predicted_values=model.sample(n_samples=60, random_state=0)[0].flatten
    ↪ ()
hmm_predicted_values=np.concatenate([start_value, predicted_values])
plt.plot(range(len(data),len(data)+len(hmm_predicted_values)),
    ↪ hmm_predicted_values, label='Predicted Values',linestyle='-')
plt.xticks(range(0, 905, 180),rotation=0,fontsize=5)
plt.yticks(fontsize=5)
plt.ylabel('Unemployment Rate',fontsize=6)
plt.legend(fontsize='small')
plt.title('Results for HMM Model',fontsize=8)
plt.savefig('HMM.png', transparent=True,dpi=1000)
plt.show()
plt.clf()
# 模型评价
print("hmmMSE=",round(MSE(newdf['Value'],hmm_fitted_values),4))
print("hmmMADE=",round(MADE(newdf['Value'],hmm_fitted_values),4))

Prophet
# 根据算法要求准备数据 (更改数据格式)
new_column_names={'Date': 'ds', 'Value': 'y'}
data_renamed=newdf.rename(columns=new_column_names)
# 训练模型
model=Prophet(yearly_seasonality=True,growth='linear')
model.fit(data_renamed)
# 要进行预测的时间格式
future=model.make_future_dataframe(periods=60,freq='M')
forecast=model.predict(future)
forecast_df=forecast[['ds','yhat','yhat_lower','yhat_upper']]
prophet_fitted_values=forecast_df.head(905)['yhat']
prophet_predcited_values=forecast_df['yhat'].tail(60)
# 模型评价
print("prophetMSE=",round(MSE(newdf['Value'],prophet_fitted_values),4)
    ↪ )
print("prophetMADE=",round(MADE(newdf['Value'],prophet_fitted_values)
    ↪ ,4))
# 模型结果可视化
plt.style.use(['science','ieee'])
plt.grid (False)
plt.plot(newdf['Date'],newdf['Value'], label='Original Data')
plt.plot(prophet_fitted_values,label='Fitted Values')
```

```
plt.plot(prophet_predcited_values,label='Predicted Values',linestyle='
    ↪ -')
plt.xticks(range(0, 905, 180),rotation=0,fontsize=5)
plt.yticks(fontsize=5)
plt.ylabel('Unemployment Rate',fontsize=6)
plt.legend(fontsize='small')
plt.title('Results for Prophet Model',fontsize=8)
plt.savefig('Prophet.png', transparent=True,dpi=1000)
plt.show()
plt.clf()
```

LSTM

```
# 定义函数来观察训练过程
def print_history1(history):
    # 绘制训练 & 验证的准确率值
    plt.style.use(['science','ieee'])
    plt.grid (False)
    plt.plot(history.history['accuracy'],linewidth=2.0,color='b')
    plt.title('Accuracy',fontsize=14)
    plt.xlabel('Epoch')
    plt.legend(['accuracy'])
    plt.savefig('modelaccuracy.png', transparent=True,dpi=1000)
    plt.show()
def print_history2(history):
    plt.style.use(['science','ieee'])
    plt.grid (False)
    plt.plot(history.history['loss'],linewidth=2.0,color='b')
    plt.title('Loss',fontsize=14)
    plt.xlabel('Epoch')
    plt.legend(['loss'])
    plt.savefig('modelloss.png', transparent=True,dpi=1000)
    plt.show()
# 定义用前20个历史数据预测目前
sequence_length=20
# 搭建一个简易的LSTM模型
lstmmodel=Sequential()
lstmmodel.add(LSTM(128,activation='sigmoid',input_shape=(
    ↪ sequence_length,1)))
lstmmodel.add(Dropout(0.3))
lstmmodel.add(Dense(1))
```

```
lstmmodel.compile(loss='mean_squared_error',optimizer='adam',metrics=[
    ↪ 'accuracy'])
data=pd.read_excel(r"C:\Users\LX\Desktop\时间序列\案例分析报告\dataset
    ↪ .xlsx").iloc[:,1:3]['Value']
train_data=data[:905]
# 定义辅助函数以生成训练集和标签
def create_sequences(data, seq_length):
    X = []
    y = []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)
# 设置序列长度和预测步数
sequence_length=20
sequence_length=int(sequence_length)
prediction_steps=60
prediction_steps=int(prediction_steps)
# 创建训练集和标签
X_train, y_train=create_sequences(train_data, sequence_length)
X_train=X_train.reshape(885,20,1)
# 开始训练模型
history=lstmmodel.fit(X_train, y_train, epochs=500, batch_size=16)
# 使用训练好的模型进行滚动预测
predicted_values=[]
last_sequence=X_train[-1]  # 使用训练集的最后一个序列作为起始点
for _ in range(prediction_steps):
    predicted_value=lstmmodel.predict(last_sequence.reshape(1,
        ↪ sequence_length, 1))
    predicted_values.append(predicted_value[0, 0])
    last_sequence=np.append(last_sequence[1:], predicted_value[0, 0])
# 借助前面定义的函数观察训练过程
print_history1(history)
print_history2(history)
lstm_predicted_values=predicted_values
lstm_fitted_values=lstmmodel.predict(X_train)#注意这里没有前20个时间的
    ↪ 拟合值
# 模型评价
print("lstmMSE=",round(MSE(np.array(newdf['Value'][20:]).reshape(-1,1)
    ↪ ,lstm_fitted_values),4))
```

```
print("lstmMADE=",round(MADE(np.array(newdf['Value'][20:]).reshape
    ↪ (-1,1),lstm_fitted_values),4))
# 可视化
plt.style.use(['science','ieee'])
plt.grid (False)
plt.plot(newdf['Date'],newdf['Value'], label='Original Data')
plt.plot(range(20, len(data)),lstmmodel.predict(X_train), label='
    ↪ Fitted Values')
plt.plot(range(len(data), len(data) + len(lstm_predicted_values)),
    ↪ lstm_predicted_values, label='Predicted Values',linestyle='-')
plt.xticks(range(0, 905, 180),rotation=0,fontsize=5)
plt.yticks(fontsize=5)
plt.ylabel('Unemployment Rate',fontsize=6)
plt.legend(fontsize='small')
plt.title('Results for LSTM Model',fontsize=8)
plt.savefig('LSTM.png', transparent=True,dpi=1000)
plt.show()
plt.clf()
```

结果比较

```
# 拟合结果比较
plt.style.use(['science','ieee'])
plt.grid (False)
plt.plot(newdf['Date'],newdf['Value'], label='Original Data',linewidth
    ↪ =1.0)
plt.plot(range(0,905),arima_fitted_values,label='ARIMA Model',
    ↪ linewidth=1.0)
plt.plot(range(0,905),hmm_fitted_values,label='HMM Model',linewidth
    ↪ =1.0)
plt.plot(range(0,905),prophet_fitted_values,label='Prophet Model',
    ↪ linewidth=1.0)
plt.plot(range(20,905),lstm_fitted_values,label='LSTM Model',linewidth
    ↪ =1.0,color='darkorange')
plt.xticks(range(0, 905, 180),rotation=0,fontsize=5)
plt.yticks(fontsize=5)
plt.ylabel('Unemployment Rate',fontsize=6)
plt.legend(fontsize='small')
plt.title('Fitted Values by Four Models',fontsize=8)
plt.savefig('Fourfittedvalues.png', transparent=True,dpi=1000)
plt.show()
plt.clf()
```

```
# 预测结果比较
plt.style.use(['science','ieee'])
plt.grid (False)
plt.plot(range(0,60),arima_predicted_values,label='ARIMA Model')
plt.plot(range(0,61),hmm_predicted_values,label='HMM Model')
plt.plot(range(0,60),prophet_predcited_values,label='Prophet Model')
plt.plot(range(0,60),lstm_predicted_values,label='LSTM Model')
plt.xticks(range(0, 60, 10),rotation=0,fontsize=5)
plt.yticks(fontsize=5)
plt.ylabel('Unemployment Rate',fontsize=6)
plt.legend(fontsize='x-small')
plt.title('Predictions by Four Models',fontsize=8)
plt.savefig('FourPredictions.png', transparent=True,dpi=1000)
plt.show()
plt.clf()
```

A.2 失业率数据

见于引言部分链接