题目分析：

开发模型-> 目的：使用给出的**每日价格流**来确定交易者每天**是否**应该购买、持有或出售其投资组合中的资产。

给出：定价数据文件LBMA-GOLD.csv和BCHAIN-MKPRU.csv

解题思路：

①数据合并，把比特币和黄金的交易数据合并起来。
②训练模型，进行时序预测，实现当天数据预测第二天数据的时序模型。
③生成每日的预测数据。
④构建目标规划模型，最大化每日交易获利的策略。
⑤启发式算法求解。
⑥迭代五年交易期。
⑦灵敏度分析。

模型一：Prophet模型（2200688.pdf）

1. **Prophet 模型简介：**
   - Prophet 是由 Facebook 开源的一种数据预测工具，基于 Python 和 R 语言。
   - 该模型设计基于时间序列分解和机器学习拟合，能够处理时间序列中的异常值和缺失值。
   - Prophet 可以几乎完全自动地预测时间序列的未来趋势，并且使用了 pyStan 这个开源工具，在拟合模型时可以在短时间内获得预测结果。
2. **Prophet 模型组成：**
   - Prophet 是一个加法模型，包括四个组件：趋势项（trend term）、周期项（period term）、假日项（holiday term）和误差项（error term）。
   - 模型公式：$p(t) = a(t) + b(t) + c(t) + \epsilon$。
3. **趋势项（a(t)）：**
   - 提供了两种趋势项的函数：基于 logistic 回归和基于分段线性函数。
   - Logistic 回归模型公式：$a(t) = H(t) / (1 + exp(-(k + z(t)\theta) * (t - (m + z(t)T/\gamma))$。
4. **周期项（b(t)）：**
   - 采用傅里叶级数模型表示，使用正弦和余弦函数来建模时间序列的周期性。
   - 周期项公式：$b(t) = x(t)\delta$，其中 $x(t)$ 是傅里叶级数，δ 是系数向量。
5. **假日项（c(t)）：**
   - 假日项模型的构建，考虑了每个假日对模型的影响。
   - 假日项公式：$c(t) = m(t)k$，其中 $m(t)$ 表示是否是假日，k 是假日的系数向量。
6. **Change-point 选择：**
   - 使用 Laplace 分布来引入变点（changepoint）的概念。
   - Laplace 分布公式：$f(x|\theta, t) = exp(-|x - \theta|/t)/2t$。
   - 模型中的 Change-point 选择涉及三个重要指标：changepoint-range、n-changepoint、changepoint-prior-scale。这些参数影响变点的位置、数量以及增长率的分布。

总体来说，Prophet 模型通过这些组件和参数，能够更灵活地适应各种时间序列数据，同时兼顾了趋势、周期性和特殊事件（如假日）的影响。

4.1.2 Prophet

Prophet [1] is a data prediction tool based on python and R language open source by Facebook. As shown in Fig. 3. The algorithm is designed based on time series decomposition and machine learning fitting, and can handle not only the case of some outliers in the time series, but also the case of some missing values. Prophet can predict the future trend of the time series almost fully automatically, due to the use of pyStan, an open source tool, in the fitting model, also makes prophet can get the prediction results in a very short time.

Prophet is an additive model that consists of four components, namely the trend term, the period term, the holiday term and the error term.

$$p(t) = a(t) + b(t) + c(t) + \epsilon \tag{1}$$

where a(t) denotes the trend term, which is the trend of the time series over the non-period. b(t) denotes the period term, c(t) denotes the holiday term, which indicates the presence of a holiday on that day, and ϵ denotes the error term.

There are two important functions for the a(t) trend term, one based on the logistic regression function and the other based on the segmented linear function.

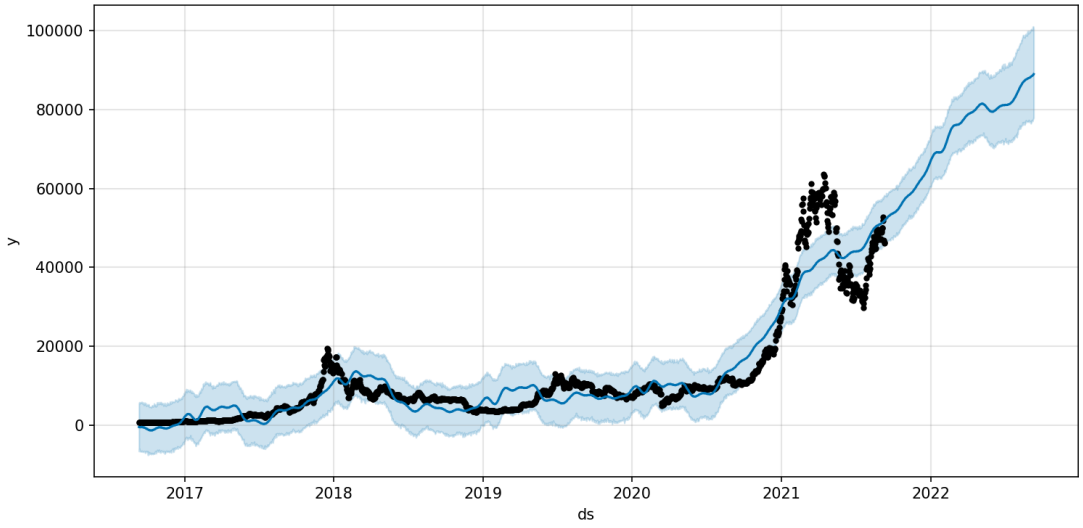
The logistic regression-based growth model is as follows:

$$a(t) = \frac{H(t)}{1 + exp(-(k + z(t)'\theta) * (t - (m + z(t)^T\gamma)))} \tag{2}$$

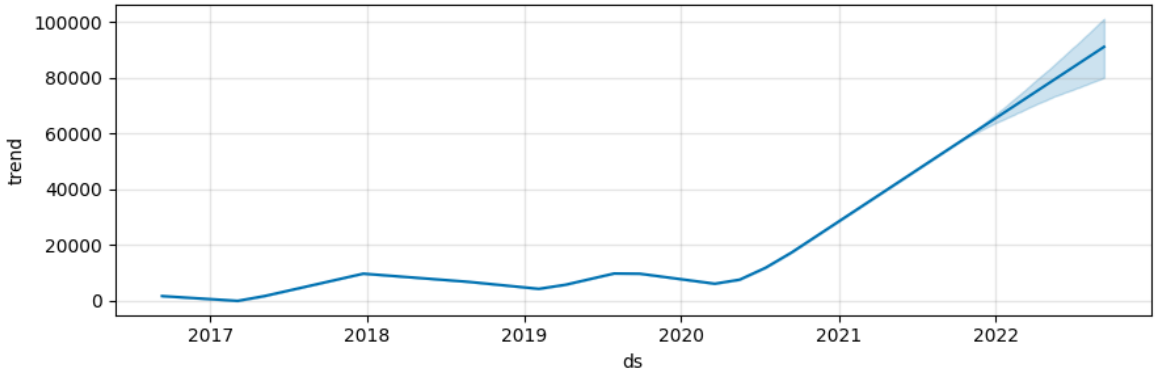
$$z(t) = (z_1(t), ..., z_s(t))^T \tag{3}$$

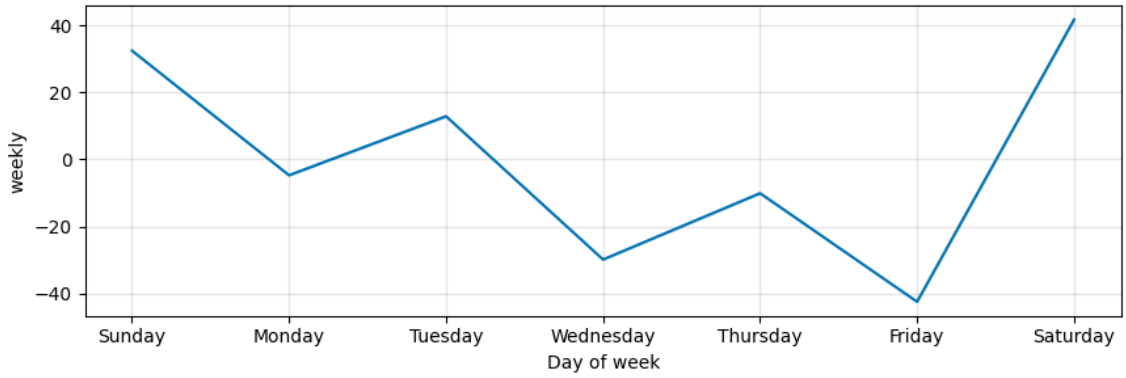
可以得到的：

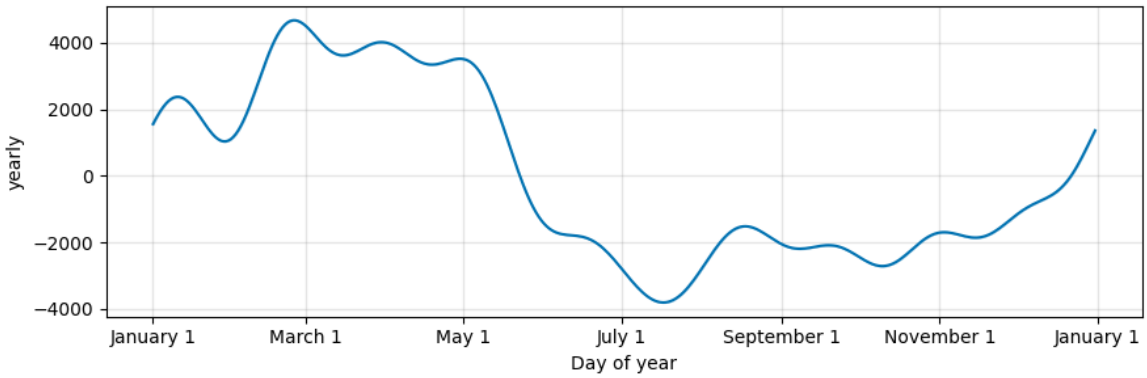
①模型拟合图：



②周期性、趋势图。。。







③评估指标：

```
Mean Squared Error (MSE): 471906594.433821
Root Mean Squared Error (RMSE): 21723.411206203804
Mean Absolute Error (MAE): 15198.773824319629
Mean Absolute Percentage Error (MAPE): 238.25979528317228%
```

文中采用**滑动分段**评估使效果更好

模型代码：

```
import pandas as pd
from prophet import Prophet
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import plotly.graph_objects as go
from prophet.plot import plot_plotly, plot_components_plotly

# 读取CSV文件
file_path = 'C:/Users/92579/Documents/GitHub/Mathematical-Modeling/学习记录/2022美赛/C题/2022_Problem_C_DATA/BCHAIN-MKPRU.csv'
df = pd.read_csv(file_path)

# 转换日期格式
```

```
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%y')  # 使用适当的日期格式
df.rename(columns={'Date': 'ds', 'Value': 'y'}, inplace=True)

# 创建并拟合Prophet模型
m = Prophet()
m.fit(df)

# 生成未来时间点
future = m.make_future_dataframe(periods=365)

# 进行预测
forecast = m.predict(future)

# 绘制预测结果图表
fig1 = m.plot(forecast)
fig2 = m.plot_components(forecast)

# 使用Plotly进行交互式可视化
fig3 = plot_plotly(m, forecast)
fig4 = plot_components_plotly(m, forecast)

# 实际值
y_true = df['y'].values

# 预测值
y_pred = forecast['yhat'].values[-len(y_true):]

# 计算均方误差（MSE）
mse = mean_squared_error(y_true, y_pred)

# 计算均方根误差（RMSE）
rmse = np.sqrt(mse)

# 计算平均绝对误差（MAE）
mae = mean_absolute_error(y_true, y_pred)

# 计算平均绝对百分比误差（MAPE）
mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100

# 打印评估指标
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Absolute Percentage Error (MAPE): {mape}%')

# 直接显示图表
fig1.show()
input("Press Enter to continue...")

fig2.show()
input("Press Enter to continue...")

fig3.show()
input("Press Enter to continue...")

fig4.show()
input("Press Enter to continue...")
```

该文后续还结合了XGBoost模型对模型进行优化处理

模型二：ARIMA模型（2203120.pdf）

价格预测模型

提高预测精度，特别是时间序列预测的准确性，是许多领域决策者经常面临的重要但常常困难的任务。自回归积分移动平均（ARIMA）模型是时间序列预测中最流行的线性模型之一，过去十年来已广泛应用于构建更准确的混合模型[2]。ARIMA模型结合了三种基本方法：

- **自回归（AR）：** 描述当前值与历史值之间的关系，并使用变量的历史时间数据来预测自身。AR模型的阶数被记录为p值。其公式如下：
- [y_t = \alpha_0 + \alpha_1 y*{t-1} + *alpha_2 y*{t-2} + \ldots + \alpha_p y*{t-p} + *varepsilon_t]
  其中，α是系数，ε_t表示误差。
- **积分（I）：** 当时间序列变得平稳时，需要进行差分，差分的阶数被记录为d值。通常，一阶差分就足够了。
- **移动平均（MA）：** 移动平均模型侧重于自回归模型中误差项的累积。MA模型的阶数被记录为q值。其公式如下：
  [y_t = c + \varepsilon_t + \theta_1 \varepsilon*{t-1} + *theta_2 *varepsilon*{t-2} + \ldots + \theta_q \varepsilon*{t-q}]
  *其中，c表示常数项，*εt是方差为白噪声过程，θ是系数。

该模型被称为ARIMA(p, d, q)，如图3所示，其中p是自回归项，q是移动平均项，D是差分次数。我们将按照下面列出的步骤构建和解决我们的模型。

步骤1：稳定化
为了使用ARIMA模型，时间序列必须是平稳的。使用增广迪基-富勒（Augmented Dickey-Fuller）单位根检验来测试平稳性。如果ADF测试得到的P值小于0.05，则是一个稳定的时间序列。如果不稳定，可以通过差分方法将非平稳过程转化为平稳过程。从表2可以看出，经过一阶差分后，时间序列变得稳定。

表2：差分前后黄金和比特币时间序列的P值

	差分前P值	**差分后P值**
黄金	0.6734	0.0043
比特币	0.9921	0.0000

步骤2：选择p和q
使用自相关函数（ACF）和偏自相关函数（PACF）确定p和q的值，确认方法如表3所示。

表3：p值和q值的确认方法

模型	**ACF**	**PACF**
AR(p)	截尾至0	p阶后截尾
MA(q)	q阶后截尾	截尾至0
ARMA(p, q)	q阶后截尾	p阶后截尾至0

注：截尾意味着落在置信区间内（95%的点符合规则）。

图4（a）显示了使用2级ACF和1级PACF对黄金数据进行截断，图4（b）显示了使用2级ACF和2级PACF对比特币数据进行截断。从中我们可以分别确定p值和q值。

步骤3：白噪声测试

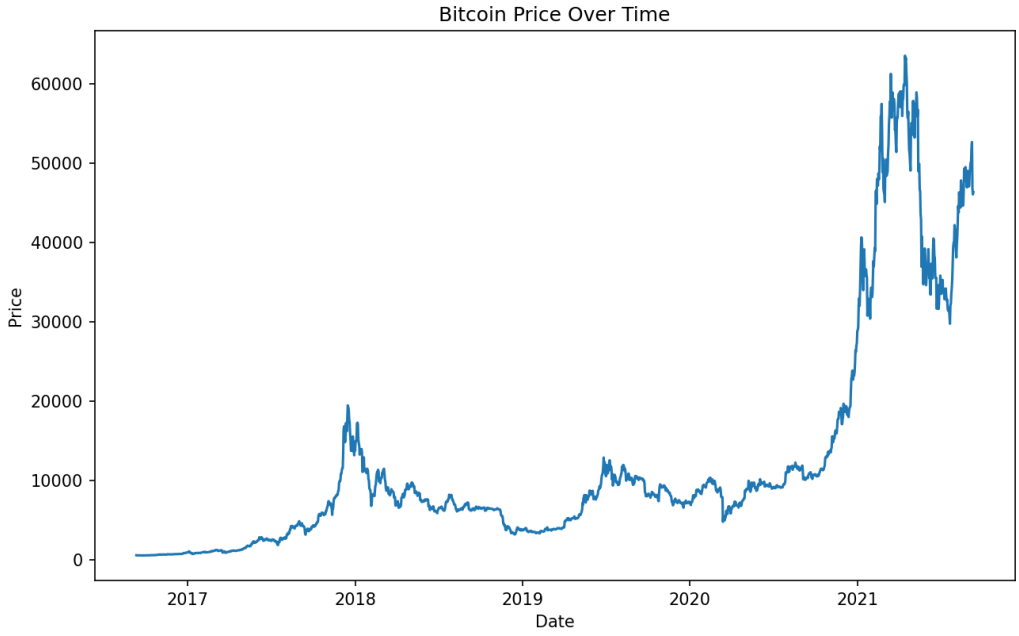
如果得到白噪声，意味着时间序列中的有用信息已经被提取出来，剩下的都是随机扰动，不能被预测和利用。如果残差序列通过了白噪声测试，建模可以终止，因为没有信息可继续提取。

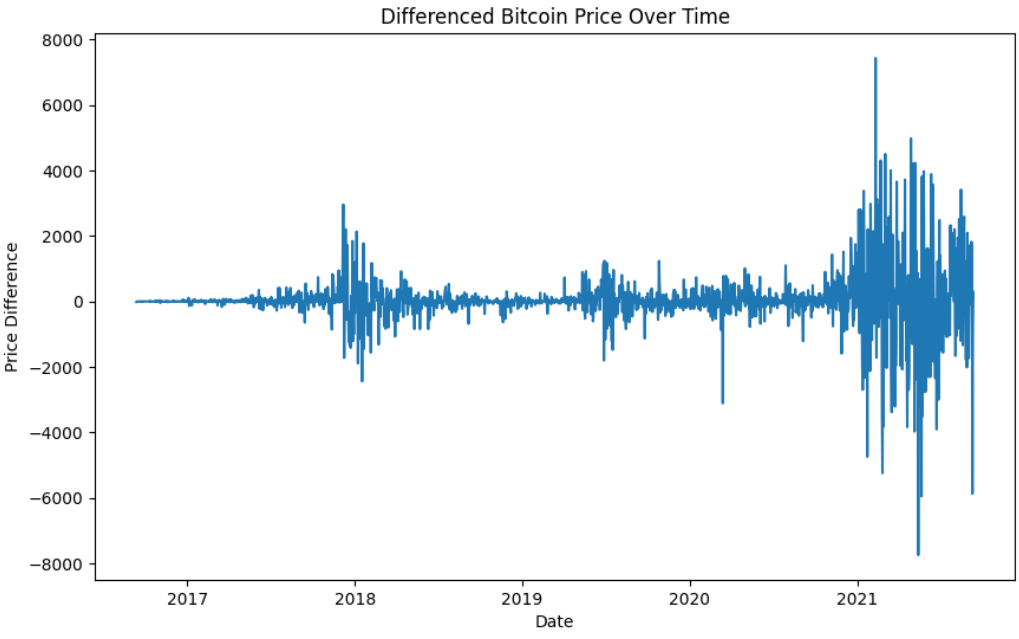
黄金的ARIMA模型为ARIMA(2,1,1)，比特币为ARIMA(2,1,2)。该模型的白噪声测试输出的p值非常接近0，说明它遵循均值为0的正态分布，即是一个白噪声。

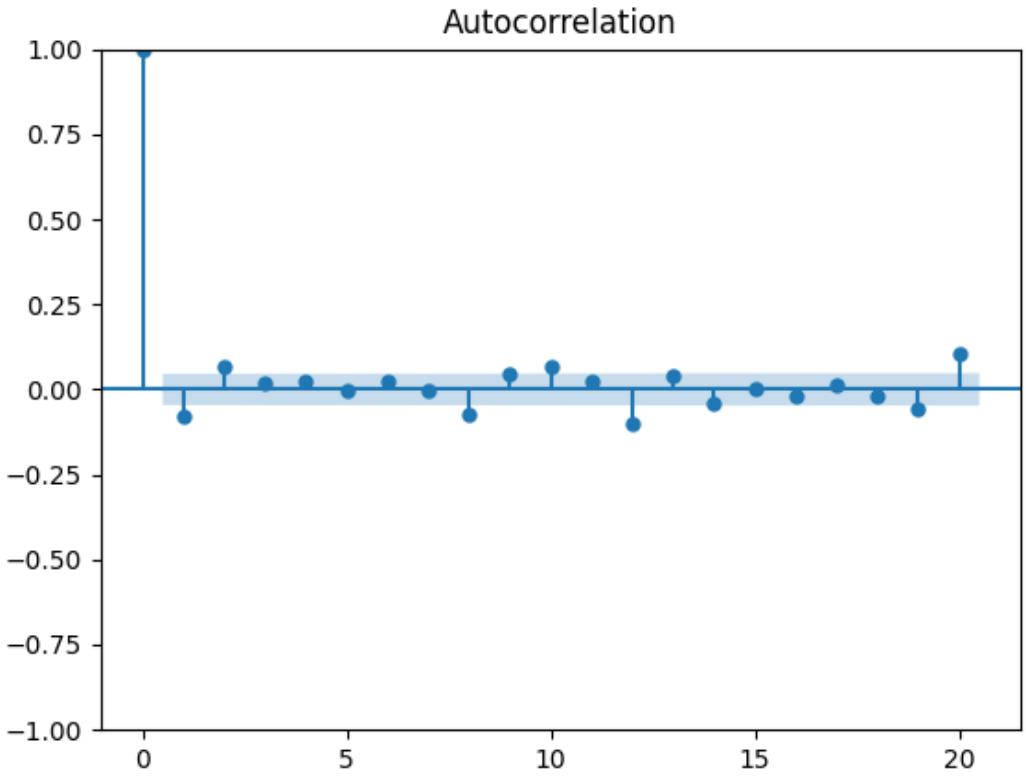
步骤4：预测

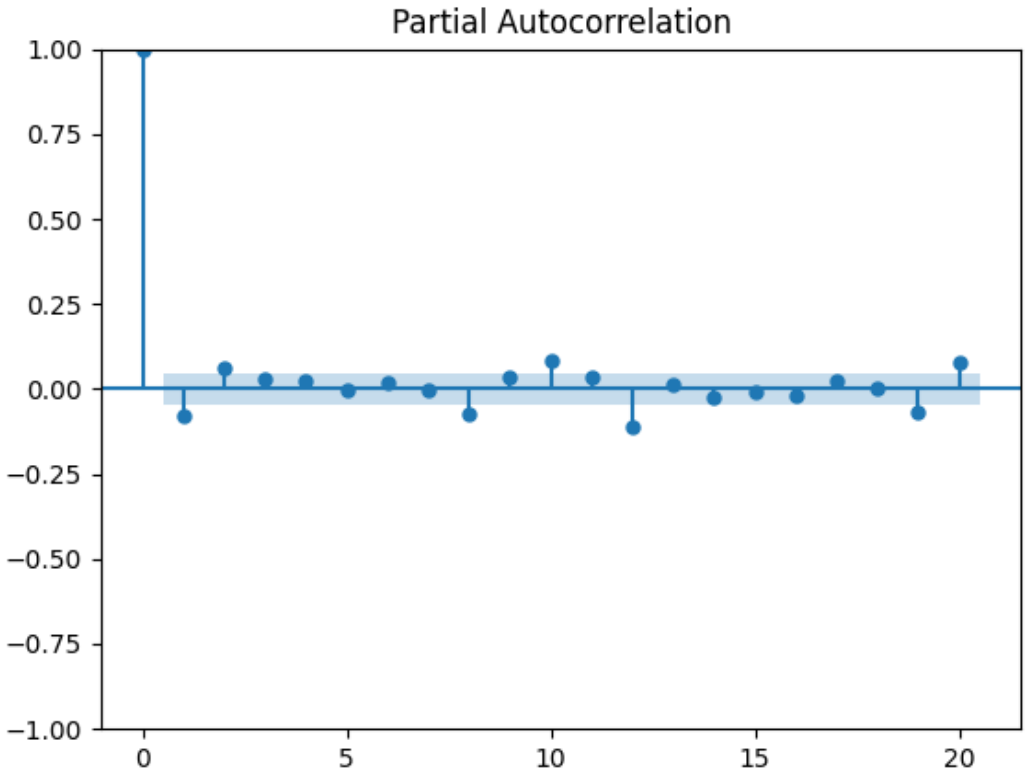
为了及时准确地预测未来市场，我们将连续30天的数据作为训练集，接下来3天的数据作为测试集。训练集的移动步长为3。一个周期的预测结果如图5所示。黄金的相关系数R^2为0.8592，比特币为0.9304。该模型的预测结果相对成功，可用于实际应用。

可以得到的：









模型代码：

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

```python
from statsmodels.tsa.stattools import adfuller

# 读取CSV文件
file_path = "C:/Users/92579/Documents/GitHub/Mathematical-Modeling/学习记录/2022美赛/C题/2022_Problem_C_DATA/BCHAIN-MKPRU.csv"
df = pd.read_csv(file_path, parse_dates=['Date'], index_col='Date')

# 数据预处理
df = df.dropna()  # 删除缺失值
df = df.resample('D').mean()  # 将数据按天重采样，取均值

# 可视化数据
plt.figure(figsize=(10, 6))
plt.plot(df['Value'])
plt.title('Bitcoin Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()

# 进行稳定性检验（Augmented Dickey-Fuller Test）
result_adf = adfuller(df['Value'])
print(f'ADF Statistic: {result_adf[0]}')
print(f'p-value: {result_adf[1]}')

# 如果时间序列不稳定，进行差分操作
if result_adf[1] > 0.05:
    df['Value'] = df['Value'].diff()
    df = df.dropna()

# 再次进行可视化
plt.figure(figsize=(10, 6))
plt.plot(df['Value'])
plt.title('Differenced Bitcoin Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price Difference')
plt.show()

# 根据ACF和PACF图确定p和q的值
plot_acf(df['Value'], lags=20)
plt.show()
plot_pacf(df['Value'], lags=20)
plt.show()

# 根据ACF和PACF图确定p和q的值
p = 2  # 根据图示确定
d = 1  # 一阶差分
q = 1  # 根据图示确定

# 拟合ARIMA模型
model = ARIMA(df['Value'], order=(p, d, q))
result = model.fit()

# 打印模型的参数
print(result.summary())

# 进行白噪声检验
residuals = result.resid
fig, ax = plt.subplots(1, 2, figsize=(12, 4))
```

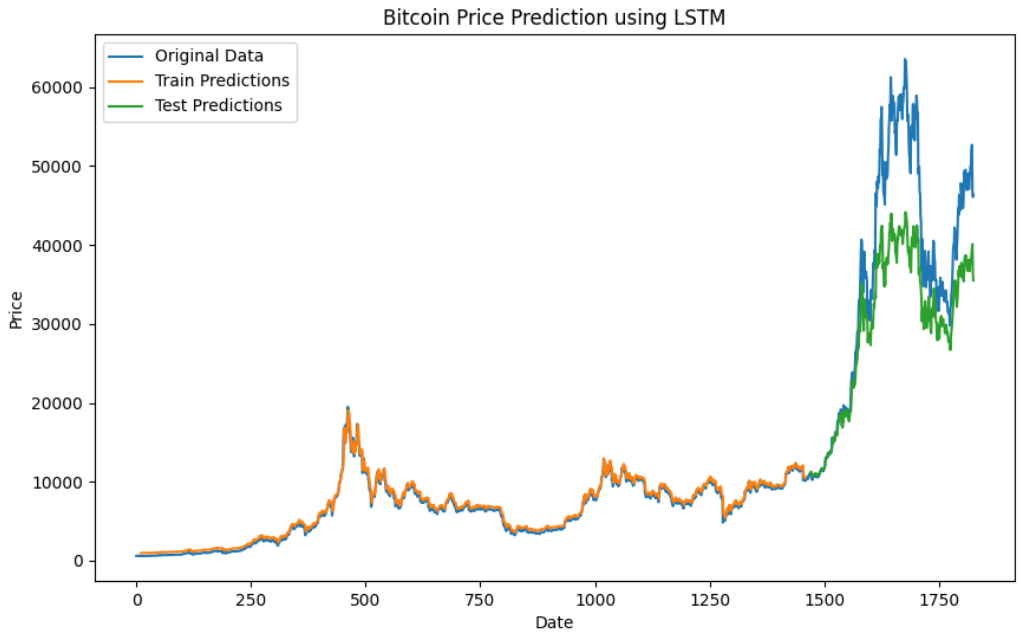
```
plot_acf(residuals, ax=ax[0])
plot_pacf(residuals, ax=ax[1])
plt.show()

# 进行模型评估和预测
future_days = 3  # 设置预测未来几天的数据
forecast = result.forecast(steps=future_days)

# 打印预测结果
print("Forecasted Values:")
print(forecast)

# 可视化预测结果
plt.figure(figsize=(10, 6))
plt.plot(df['Value'], label='Historical Data')
plt.plot(pd.date_range(df.index[-1], periods=future_days + 1)[1:], forecast,
label='Forecast', color='red')
plt.title('Bitcoin Price Prediction')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

模型三：LSTM模型（2204883.pdf）

1. **LSTM简介：**
   - LSTM（长短时记忆网络）是循环神经网络（RNN）的一种特殊类型，解决了在RNN长序列训练中出现的梯度爆炸和梯度消失问题。当网络的层数增加时，后续节点对先前节点的感知减弱，随着时间的推移遗忘先前信息的现象逐渐发生。
   - LSTM通过在隐藏层的每个神经单元中添加一个内存单元来改进RNN，称为“细胞状态”，并使用遗忘门、输入门和输出门等结构来控制时间序列上的内存信息。这使得LSTM能够更深入地挖掘数据之间的潜在模式，从而使预测更准确可靠。
2. **LSTM的门结构：**
   - **遗忘门：** 选择性地遗忘过去的某些信息。使用Sigmoid函数来实现，输出值在0和1之间，表示是否应该遗忘先前的信息。
   - **输入门：** 选择性地记住过去的某些信息并确定当前记忆单元的值。生成两个中间变量，其中一个用于输入门的输出值，另一个用于候选记忆单元值。
   - **输出门：** 选择性地生成当前隐藏状态的值。输出值用作中间变量，通过对当前记忆单元值进行处理生成隐藏状态值。
3. **Sliding Window Algorithm（滑动窗口算法）：**
   - 在预测每天的黄金和比特币价格时，使用滑动窗口算法。仅使用前一天的价格数据作为LSTM模型的输入，滑动窗口包含当前天和其前d天的数据，用于训练模型。
4. **调整参数：**
   - 通过调整隐藏层大小l、滑动窗口大小d、训练滑动窗口大小d0、学习率λ和迭代次数ε等5个参数，使用MAPE（平均绝对百分比误差）评估黄金价格预测结果。
   - 在调整参数时，每次只更改一个参数的值，保持其他参数不变。选择MAPE值最小的参数组合作为最终的黄金和比特币价格预测结果。
5. **结果展示：**

- 最终选择的参数组合为[l=100, d=10, $d0$=5, λ=0.001, ϵ=50]，对应的MAPE值为0.007261552。通过比较预测曲线和真实曲线，证明了对黄金和比特币价格的预测效果良好。

可以得到的：



Train Score: 469.53 RMSE
Test Score: 8900.69 RMSE

- 训练集的RMSE为469.53
- 测试集的RMSE为8900.69

模型代码：

```
# 导入必要的库
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import LSTM, Dense

# 读取CSV文件
file_path = "C:/Users/92579/Documents/GitHub/Mathematical-Modeling/学习记录/2022美赛/C题/2022_Problem_C_DATA/BCHAIN-MKPRU.csv"
df = pd.read_csv(file_path, parse_dates=['Date'], index_col='Date')

# 数据预处理
df = df.dropna()  # 删除缺失值
df = df.resample('D').mean()  # 将数据按天重采样，取均值

# 归一化数据
scaler = MinMaxScaler()
df['Value'] = scaler.fit_transform(df[['Value']])

# 划分训练集和测试集
```

```
train_size = int(len(df) * 0.8)
train, test = df[:train_size], df[train_size:]

# 创建窗口数据集
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back):
        a = dataset[i:(i + look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)

look_back = 10  # 可调整窗口大小
trainX, trainY = create_dataset(np.array(train), look_back)
testX, testY = create_dataset(np.array(test), look_back)

# 调整输入数据的形状
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# 构建LSTM模型
model = Sequential()
model.add(LSTM(50, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

# 训练模型
model.fit(trainX, trainY, epochs=50, batch_size=1, verbose=2)

# 在训练集和测试集上进行预测
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)

# 反归一化预测值
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])

# 计算训练集和测试集的均方根误差
trainScore = np.sqrt(np.mean(np.square(trainY[0] - trainPredict[:, 0])))
testScore = np.sqrt(np.mean(np.square(testY[0] - testPredict[:, 0])))

print('Train Score: %.2f RMSE' % (trainScore))
print('Test Score: %.2f RMSE' % (testScore))

# 可视化训练集的预测结果
trainPredictPlot = np.empty_like(df)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(trainPredict) + look_back, :] = trainPredict

# 可视化测试集的预测结果
testPredictPlot = np.empty_like(df)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(trainPredict) + (look_back * 2):len(df), :] = testPredict

# 将原始数据、训练集和测试集的预测结果进行可视化
plt.figure(figsize=(10, 6))
```

```
plt.plot(scaler.inverse_transform(df[['Value']]), label='Original Data')
plt.plot(trainPredictPlot, label='Train Predictions')
plt.plot(testPredictPlot, label='Test Predictions')
plt.title('Bitcoin Price Prediction using LSTM')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```