Team Number:	apmcm22xxxxx
Problem Chosen:	C

2023 APMCM summary sheet

In recent years, the rising trend of temperature in many countries has become more and more obvious. In this paper, we mainly use multiple statistics and neural networks to establish an explainable fitting model of the global temperature change trend and use some optimized correlation coefficient methods to dig out key factors affecting global warming. Finally, some scientific plans and suggestions are given based on the established global warming analysis and prediction model.

Aiming at problem 1, ...

Aiming at problem 2, ...

Aiming at problem 3, ...

Keywords:

Contents

I. Problem background

Global warming is ...

II. Problem analysis

2.1 Question1:

In order to ...

- a) First, we ...
- b) Based on the historical data, ...
- c) Using the two mathematical models in b to predict ...
- d) The optimal model is ...

2.2 Question 2:

In order to ...

- a) In order to analyze the correlation between temperature, time, longitude, and dimensions, we ...
- b) We first collected and integrated data on ...
- c) Combined with the above questions a and b, we then ...
- d) Based on the results of the analysis in c, we ...

III. Symbol description

Table 1 Symbol description

Symbol	Description	Symbol	Description
p		R	
q		W_i	

IV. General Assumptions

- **Assumption 1:**
- **Assumption 2:**
- **Assumption 3:**
- **Assumption 4:**

V. Model and solution of problem one

5.1 Data processing and analysis

Here we preprocess the data and then perform statistical analysis to solve problem a.

5.1.1 Pre-processing

5.1.2 Statistical analysis

1. **The tropical, temperate, and global temperatures all show an upward trend over time (Figure 3):**
2. **Fluctuation of global temperature (Figure 4):**
3. **Abnormal changes(Figure 4):**

5.2 Change in temperature increase for problem a

We can conclude that ...

- **The 2013-2022 temperature change has the largest increase in any decade in the last 150 years(1873 to 2022).**
- **But the temperature change from 2013-2022 is not the largest decadal increase in history.**

5.3 ARIMA Model to describe and predict

ARIMA is ...

5.3.1 Data pre-porcessing

5.3.2 Establishment of ARIMA model

- **Tocorrelation function:**
- **Partial autocorrelation function:**
- **Determination of the p, q order:**

- **Parameter estimation:**
- **AIC:**
- **BIC:**

5.3.3 Solution of ARIMA model

1. **Parameter solving:**
    - **Determination of p,q:**
    - **Determination of d:**
2. **Model results:**

    Autoregressive moving average model ARMA:

    Differential autoregressive moving average model ARIMA:
3. **The solution of problem b:**

    The description of the past and future:
4. **The solution of problem c:**

5.4 Transformer Model to describe and predict

The model will be used to solve problems B and C in the description and prediction.

5.4.1 Data pre-processing

5.4.2 Establishment of TransFormer model

The overall framework ...

- **Self Attention:**
- **Multi-Head Attention:**
- **decoder:**
- **Masked Multi-Head Attention**

5.4.3 Solution of TransFormer model

The following part of the model is established for problem b

Describe the past:

Describe the future:

The following part of the model is established for problem c:

Global temperature forecast from 2050.1-2050.12 and 2100.1-2100.12:

5.5 Evaluation of the Model

In this section, we use RMSE and MAE as metrics to evaluate model accuracy

VI. Model and solution of problem two

6.1 Question a

In this question, we mainly use statistical correlation coefficient analysis to analyze the correlation between location, time, and temperature. Here we first introduce the three coefficients of correlation analysis.

6.1.1 Data pre-processing

6.1.2 Establishment of Correlation analysis Model

Three correlation coefficients are summarized as follows:

- **Person correlation coefficient:**
- **Formula derivation:**
- **Spearman correlation coefficient:**
- **Kendall correlation coefficient:**
- **Model choose:**

6.1.3 Solution of Correlation analysis Model

6.2 Question b

6.2.1 Data pre-porcessing

6.2.2 Establishment of model

6.2.3 Solution of model

6.3 Question c

6.3.1 Data pre-processing

6.3.2 Establishment of PCA Model

PCA (Principal Component Analysis) is a multivariate statistical method...

- **PCA principle implementation steps:**

6.3.3 Solution of PCA Model

6.4 Question d

Global Warming, also called Climate Change, is one of the most difficult issues that the world is facing today ...

6.4.1 Suggestion

VII. Non-technical report for the APMCM

Dear Organizeling Competition:

It's a great honor to convey our ideas to you!

Our survey results show ...

1. **Analysis of the causes of global temperature rise from 1743 to 2022:**

   Our suggestion:
2. **Impact of latitude on air temperature:**
3. **Impact of carbon emissions on temperature:**

   Our suggestions:
4. **Volcanic eruption makes the temperature rise:**

VIII. Sensitivity analysis

8.1 ARIMA Sensitivity analysis

8.2 TransFormer Sensitivity analysis

IX. Evaluation, Improvement and Promotion of the Model

9.1 Evaluation of the Model

- **Advantage:**

  1)

  2)

  3)
- **Shortcoming:**

  1)

  2)

9.2 Improvement and promotion of the Model

(1)

(2)

X. Reference

[1]

[2]

[3]

XI. Appendix

Detailed Steps of PCA:

Step1: Standardize sample data

Step2: The correlation coefficient is calculated to obtain the correlation coefficient matrix

Step3: Calculate the eigenroots of matrix R and the corresponding eigenvectors

Step4: Calculate principal components:

Step5: Principal ingredient selection:

Code:

```
% Transformer Model Code:
import torch
import torch.nn as nn
import numpy as np
import time
import math
from matplotlib import pyplot
import pandas as pd
torch.manual_seed(0)
np.random.seed(0)

input_window = 100
output_window = 1
batch_size = 10 # batch size
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

class PositionalEncoding(nn.Module):

    def __init__(self, d_model, max_len=5000):
        super(PositionalEncoding, self).__init__()
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len,
            dtype=torch.float).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2).float() *
            (-math.log(10000.0) / d_model))
        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
        pe = pe.unsqueeze(0).transpose(0, 1)
        #pe.requires_grad = False
        self.register_buffer('pe', pe)

    def forward(self, x):
        return x + self.pe[:x.size(0), :]


class TransAm(nn.Module):
    def __init__(self,feature_size=250,num_layers=1,dropout=0.1):
        super(TransAm, self).__init__()
        self.model_type = 'Transformer'

        self.src_mask = None
        self.pos_encoder = PositionalEncoding(feature_size)
        self.encoder_layer =
            nn.TransformerEncoderLayer(d_model=feature_size,
            nhead=10, dropout=dropout)
        self.transformer_encoder =
            nn.TransformerEncoder(self.encoder_layer,
            num_layers=num_layers)
        self.decoder = nn.Linear(feature_size,1)
        self.init_weights()

    def init_weights(self):
        initrange = 0.1
        self.decoder.bias.data.zero_()
        self.decoder.weight.data.uniform_(-initrange, initrange)

    def forward(self,src):
        if self.src_mask is None or self.src_mask.size(0) != len(src):
            device = src.device
            mask = self._generate_square_subsequent_mask(len(src))\
            .to(device)
            self.src_mask = mask

        src = self.pos_encoder(src)
        output = self.transformer_encoder(src,self.src_mask)#,
            self.src_mask)
        output = self.decoder(output)
        # DenNet(out)
        # softmax()
        return output

    def _generate_square_subsequent_mask(self, sz):
        mask = (torch.triu(torch.ones(sz, sz)) == 1).transpose(0, 1)
        mask = mask.float().masked_fill(mask == 0,
            float('-inf')).masked_fill(mask == 1, float(0.0))
        return mask

% PCA Model Code:
    df = pd.read_csv(r"pca_final_data.txt", encoding='gbk',
        index_col=0).reset_index(drop=True)
from factor_analyzer.factor_analyzer import
    calculate_bartlett_sphericity
chi_square_value, p_value = calculate_bartlett_sphericity(df)
print(chi_square_value, p_value)

from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all, kmo_model = calculate_kmo(df)
print(kmo_all)



def meanX(dataX):
    return np.mean(dataX,axis=0)
average = meanX(df)
print(average)

m, n = np.shape(df)
print(m,n)

data_adjust = []
avgs = np.tile(average, (m, 1))
print(avgs)

data_adjust = df - avgs
print(data_adjust)

covX = np.cov(data_adjust.T)
print(covX)

featValue, featVec = np.linalg.eig(covX)
featValue = sorted(featValue)[::-1]
```

```python
print(featValue)
plt.scatter(range(1, df.shape[1] + 1), featValue)
plt.plot(range(1, df.shape[1] + 1), featValue)
plt.title("Scree Plot")
plt.xlabel("Factors")
plt.ylabel("Eigenvalue")
plt.grid()
plt.show()
gx = featValue/np.sum(featValue)
print(gx)
lg = np.cumsum(gx)
print(lg)
k=[i for i in range(len(lg)) if lg[i]<0.97]
k = list(k)
print(k)
selectVec = np.matrix(featVec.T[k]).T
selectVe=selectVec*(-1)
print(selectVec)
finalData = np.dot(data_adjust,selectVec)
print(finalData)
plt.figure(figsize = (7,5))
ax = sns.heatmap(selectVec, annot=True, cmap="BuPu")
ax.yaxis.set_tick_params(labelsize=15)
plt.title("Factor Analysis", fontsize="xx-large")
plt.ylabel("Sepal Width", fontsize="xx-large")
plt.show()
# plt.savefig("factorAnalysis", dpi=500)
print()
```