美赛C题第一问思路

①数据预处理（省略）

②首先进行基本的随机森林和XGBoost方法进行预测：

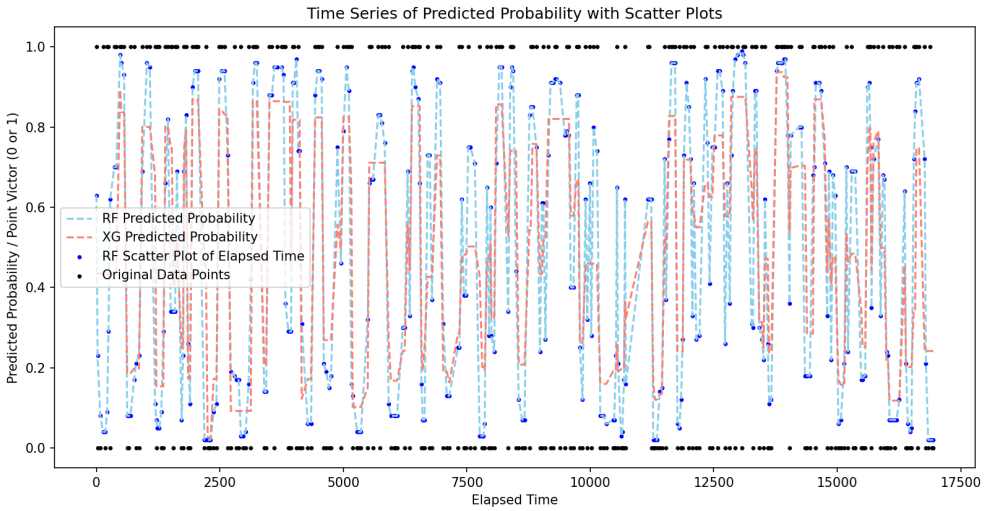
随机森林预测结果：

Random Forest Accuracy: 0.43564356435643564
Random Forest Confusion Matrix:
[[22 28]
 [29 22]]

XGBoost预测结果：

XGBoost Accuracy: 0.45544554455445546
XGBoost Confusion Matrix:
[[23 27]
 [28 23]]

拟合效果很差：



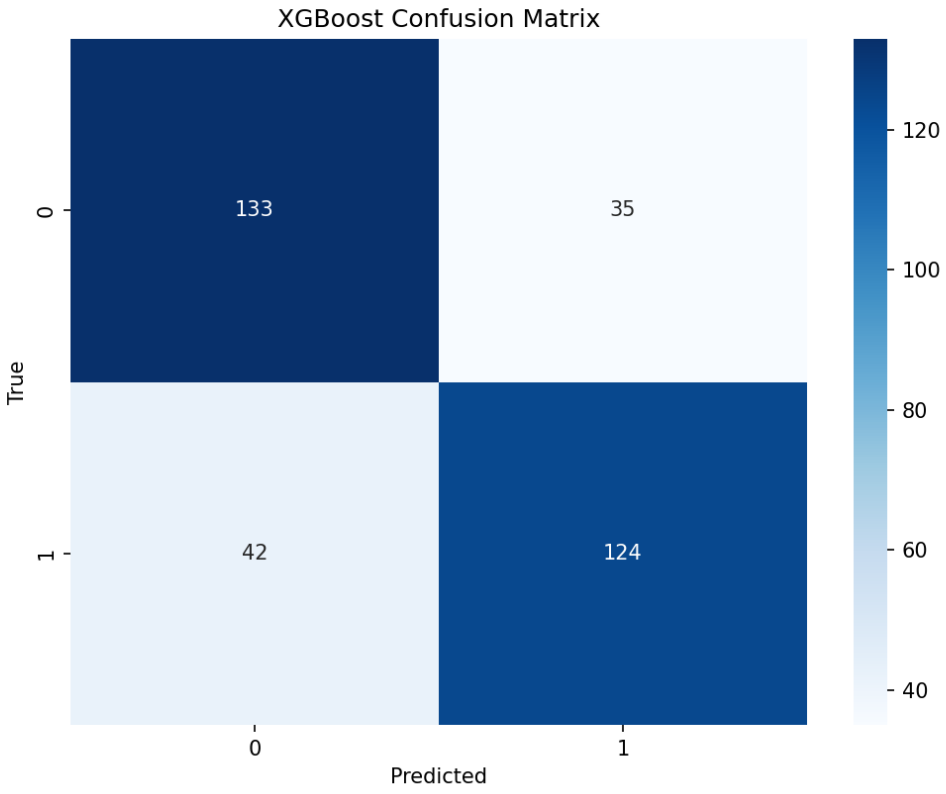

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, confusion_matrix

# Load data from '数据处理.csv' with specific columns
file_path = "c:/Users/92579/Documents/GitHub/Mathematical-Modeling/比赛记录/2024 MCM/美赛/2024_MCM-ICM_Problems/2024_MCM-ICM_Problems/数据处理.csv"
columns_to_read = ['match_id', 'player1', 'player2', 'elapsed_time', 'p1_sets', 'p2_sets', 'p1_games', 'p2_games',
                   'score_lead', 'Tie_breakers', 'server', 'serve_no', 'point_victor', 'game_victor', 'set_victor',
```

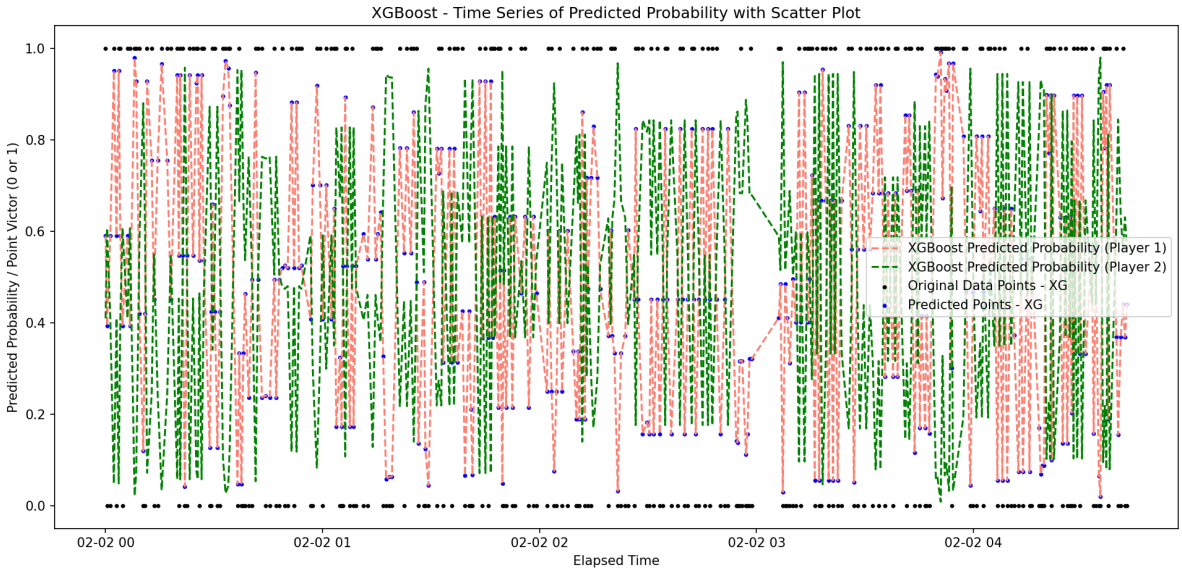
```
                    'p1_ace', 'p2_ace', 'p1_winner', 'p2_winner',
'p1_double_fault', 'p2_double_fault', 'p1_unf_err',
                    'p2_unf_err', 'p1_net_pt', 'p2_net_pt', 'p1_net_pt_won',
'p2_net_pt_won', 'p1_break_pt', 'p2_break_pt',
                    'p1_break_pt_won', 'p2_break_pt_won', 'p1_break_pt_missed',
'p2_break_pt_missed', 'p1_distance_run',
                    'p2_distance_run', 'rally_count']

df = pd.read_csv(file_path, usecols=columns_to_read)

# Choose features and target variable
features = ['elapsed_time', 'p1_sets', 'p2_sets', 'p1_games', 'p2_games']
target = 'point_victor'

# Convert 'elapsed_time' to string and then to timedelta
df['elapsed_time'] =
pd.to_timedelta(df['elapsed_time'].astype(str)).dt.total_seconds()

# Process other non-numeric features (dummy encoding)
df = pd.get_dummies(df, columns=['server', 'serve_no'])

# Encode 'point_victor' column
le = LabelEncoder()
df[target] = le.fit_transform(df[target])

# Split data into training and testing sets
train_size = 0.7
train, test = train_test_split(df, test_size=1 - train_size, random_state=42)

# Train Random Forest model on the first 70% of data
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(train[features], train[target])

# Use Random Forest model for predictions on the entire dataset
df['rf_predictions'] = rf_model.predict_proba(df[features])[:, 1]

# Train XGBoost model on the first 70% of data
xg_model = XGBClassifier(random_state=42)
xg_model.fit(train[features], train[target])

# Use XGBoost model for predictions on the entire dataset
df['xg_predictions'] = xg_model.predict_proba(df[features])[:, 1]

# Inverse transform 'point_victor' for interpretation
df['point_victor'] = le.inverse_transform(df['point_victor'])

# Sort DataFrame by 'elapsed_time' for time series plot
df.sort_values('elapsed_time', inplace=True)

# Plot the time series of predicted probabilities for Random Forest
plt.figure(figsize=(10, 6))
plt.plot(df['elapsed_time'], df['rf_predictions'], label='RF Predicted
Probability', linestyle='dashed', color='skyblue')

# Plot the time series of predicted probabilities for XGBoost
plt.plot(df['elapsed_time'], df['xg_predictions'], label='XG Predicted
Probability', linestyle='dashed', color='salmon')
```

```
plt.xlabel('Elapsed Time')
plt.ylabel('Predicted Probability')
plt.title('Time Series of Predicted Probability')
plt.legend()
plt.show()



# Random Forest predictions on the test set
rf_test_predictions = rf_model.predict(test[features])

# Calculate accuracy for Random Forest
rf_accuracy = accuracy_score(test[target], rf_test_predictions)
print("Random Forest Accuracy:", rf_accuracy)

# Confusion matrix for Random Forest
rf_conf_matrix = confusion_matrix(test[target], rf_test_predictions)
print("Random Forest Confusion Matrix:")
print(rf_conf_matrix)

# XGBoost predictions on the test set
xg_test_predictions = xg_model.predict(test[features])

# Calculate accuracy for XGBoost
xg_accuracy = accuracy_score(test[target], xg_test_predictions)
print("\nXGBoost Accuracy:", xg_accuracy)

# Confusion matrix for XGBoost
xg_conf_matrix = confusion_matrix(test[target], xg_test_predictions)
print("XGBoost Confusion Matrix:")
print(xg_conf_matrix)
```

③进行X-Prophet预测：（看情况用不用）

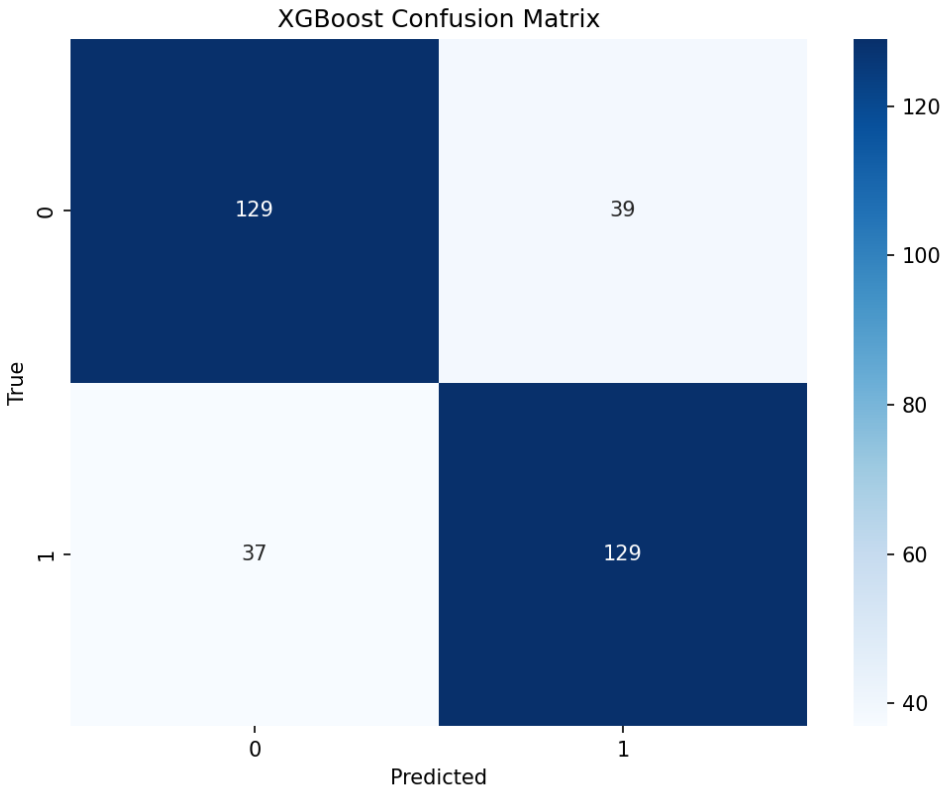
XGBoost Accuracy: 0.7694610778443114

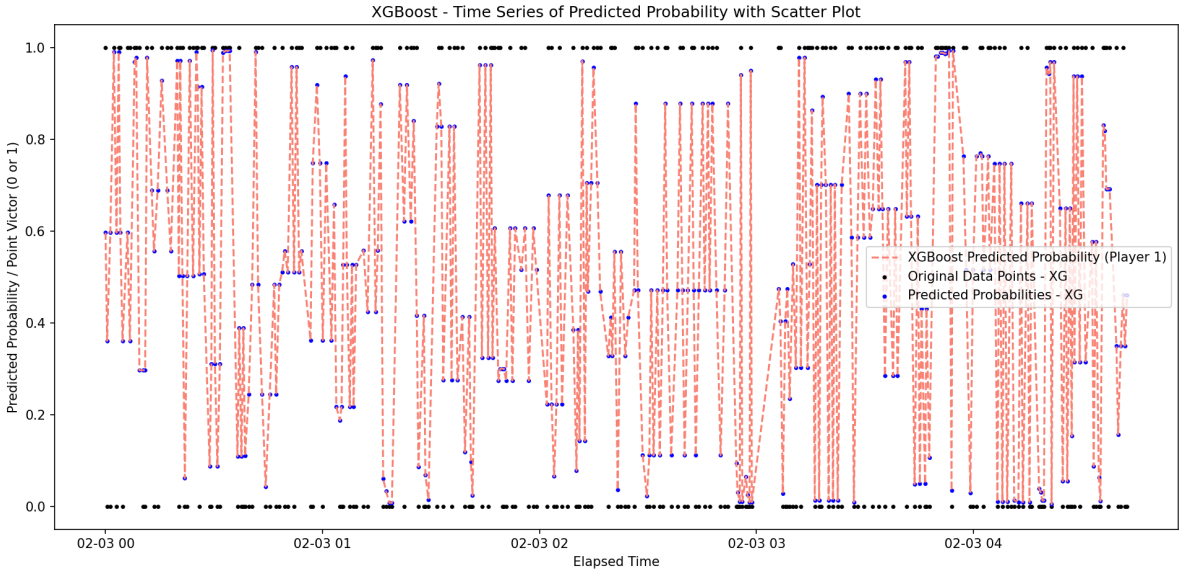




④进行XGBoost + 随机森林预测：

XGBoost Accuracy: 0.7724550898203593






```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load data from '数据处理.csv' with specific columns
file_path = "c:/Users/92579/Documents/GitHub/Mathematical-Modeling/比赛记录/2024 MCM/美赛/2024_MCM-ICM_Problems/2024_MCM-ICM_Problems/数据处理.csv"
columns_to_read = ['match_id', 'player1', 'player2', 'elapsed_time', 'p1_sets', 'p2_sets', 'p1_games', 'p2_games',
```

```
                    'score_lead', 'Tie_breakers', 'server', 'serve_no',
'point_victor', 'game_victor', 'set_victor',
                    'p1_ace', 'p2_ace', 'p1_winner', 'p2_winner',
'p1_double_fault', 'p2_double_fault', 'p1_unf_err',
                    'p2_unf_err', 'p1_net_pt', 'p2_net_pt', 'p1_net_pt_won',
'p2_net_pt_won', 'p1_break_pt', 'p2_break_pt',
                    'p1_break_pt_won', 'p2_break_pt_won', 'p1_break_pt_missed',
'p2_break_pt_missed', 'p1_distance_run',
                    'p2_distance_run', 'rally_count']
df = pd.read_csv(file_path, usecols=columns_to_read)

# Convert 'elapsed_time' to datetime
df['elapsed_time'] = pd.to_datetime(df['elapsed_time'])

# Choose features and target variable
features_rf = ['p1_sets', 'p2_sets', 'p1_games', 'p2_games', 'score_lead']
features_xg = features_rf + ['rf_predictions']  # Include Random Forest
predictions as a feature for XGBoost
target = 'point_victor'

# Encode 'point_victor' column
le = LabelEncoder()
df[target] = le.fit_transform(df[target])

# Split data into training and testing sets
train_size = 0.7
train, test = train_test_split(df, test_size=1 - train_size, random_state=42)

# Train Random Forest model on the first 70% of data
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(train[features_rf], train[target])

# Use Random Forest model for predictions on the entire dataset
df['rf_predictions'] = rf_model.predict(df[features_rf])

# Train XGBoost model on the entire dataset with 'rf_predictions' as a feature
xg_model = XGBClassifier(random_state=42)
xg_model.fit(df[features_xg], df[target])

# Use XGBoost model for predictions on the entire dataset
df['xg_predictions'] = xg_model.predict(df[features_xg])

# Inverse transform 'point_victor' for interpretation
df['point_victor'] = le.inverse_transform(df['point_victor'])

# Calculate accuracy for XGBoost
xg_accuracy = accuracy_score(df[target], df['xg_predictions'])
print("XGBoost Accuracy:", xg_accuracy)

# Confusion matrix for XGBoost
xg_conf_matrix = confusion_matrix(df[target], df['xg_predictions'])
plt.figure(figsize=(8, 6))
sns.heatmap(xg_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['0',
'1'], yticklabels=['0', '1'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('XGBoost Confusion Matrix')
plt.show()
```

```
# ...

# Use XGBoost model for predictions on the entire dataset
df['xg_probabilities'] = xg_model.predict_proba(df[features_xg])[:, 1]

# Inverse transform 'point_victor' for interpretation
df['point_victor'] = le.inverse_transform(df['point_victor'])

# Calculate accuracy for XGBoost
xg_accuracy = accuracy_score(df[target], df['xg_predictions'])
print("XGBoost Accuracy:", xg_accuracy)

# Plot the time series of predicted probabilities for XGBoost
plt.figure(figsize=(15, 8))

# Subplot 1: XGBoost Predicted Probability and Original Data Points
plt.plot(df['elapsed_time'], df['xg_probabilities'], label='XGBoost Predicted
Probability (Player 1)', linestyle='dashed', color='salmon')
plt.scatter(df['elapsed_time'], df['point_victor'], marker='o', s=5,
color='black', label='Original Data Points - XG')
plt.scatter(df['elapsed_time'], df['xg_probabilities'], marker='o', s=5,
color='blue', label='Predicted Probabilities - XG')  # Blue points at predicted
points
plt.xlabel('Elapsed Time')
plt.ylabel('Predicted Probability / Point Victor (0 or 1)')
plt.title('XGBoost - Time Series of Predicted Probability with Scatter Plot')
plt.legend()

plt.tight_layout()
plt.show()
```

其中：随机森林的算法原理：

在论文中，对随机森林（Random Forest）和XGBoost的算法原理进行深入分析时，可以包含以下内容。请注意，以下是一种简要的描述，详细的算法推导和公式可能需要更多的篇幅和详细讨论。

随机森林（Random Forest）：

基本概念：

- 随机森林是一种集成学习方法，基于决策树构建。
- 通过构建多个决策树，并对它们的预测进行集成，随机森林能够提高模型的泛化性能。

算法原理：

1. **构建决策树：**
    - 使用随机的子样本和特征选择的方式构建多个决策树。
    - 对于每个树的训练样本，随机抽样一部分数据，同时对特征进行随机选择。
2. **投票集成：**
    - 对每个决策树的预测进行投票或平均，得到最终的随机森林模型的预测结果。
3. **随机性引入：**
    - 引入随机性有助于防止过拟合，并增加模型的多样性。

公式推导（简化）：

对于一个具体的决策树模型，可以定义为： $T(x; \Theta)$

[T(x; \Theta)]
其中 (x) 是输入特征向量， (\Theta) 是树的参数。

对于随机森林的预测： $RF(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x; \Theta_i)$

RF(*x*)=*N*1∑*i*=1*N***T***i*(*x*;Θ*i*)
[RF(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x; \Theta_i)]

XGBoost:

基本概念：

- XGBoost是一种梯度提升算法，通过集成多个弱学习器来构建一个强大的模型。
- 通过迭代训练，每次迭代学习一个弱学习器来纠正前一轮的错误。

算法原理：

1. **Boosting过程：**
   - 每轮迭代学习一个新的弱学习器，使其专注于前一轮模型预测错误的样本。
2. **目标函数优化：**
   - XGBoost的训练过程是通过优化一个目标函数来实现的，该目标函数包含了模型的拟合误差和正则化项。
3. **弱学习器的加权集成：**
   - 每个弱学习器都被加权并与之前的学习器进行组合。

公式推导（简化）：

定义第 (t) 轮的模型为 (F_t(x))，目标函数为：
[\text{Obj}^{(t)} = \sum*{i=1}^{n} L(y_i, F*{t-1}(x_i)) + \sum_{i=1}^{t} \Omega(f_i)]
其中 (L) 是损失函数， (\Omega) 是正则化项。

每轮迭代时，目标函数的负梯度近似表示了前一轮模型的残差：
[\text{Residuals}^{(t)} = - \left[\frac{\partial L(y_i, F*{t-1}(x_i))}{\partial F*{t-1}(x_i)} \right]*{F*{t-1}(x_i)=F_{t-1}(x_i)}]

新的弱学习器 (f_t(x)) 学习这些残差。

XGBoost的预测为前 (t) 轮模型和新学习器的组合：
[F_t(x) = F_{t-1}(x) + \eta f_t(x)]
其中 (\eta) 是学习率。

3. **弱学习器的加权集成：**
   - 每个弱学习器都被加权并与之前的学习器进行组合。

公式推导（简化）：

定义第 t 轮的模型为 $F_t(x)$，目标函数为：
$\text{Obj}^{(t)} = \sum_{i=1}^{n} L(y_i, F_{t-1}(x_i)) + \sum_{i=1}^{t} \Omega(f_i)$
其中 L 是损失函数，Ω 是正则化项。

每轮迭代时，目标函数的负梯度近似表示了前一轮模型的残差：
$\text{Residuals}^{(t)} = -\left[\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}\right]_{F_{t-1}(x_i)=F_{t-1}(x_i)}$

新的弱学习器 $f_t(x)$ 学习这些残差。

XGBoost的预测为前 t 轮模型和新学习器的组合：
$F_t(x) = F_{t-1}(x) + \eta f_t(x)$
其中 η 是学习率。