**河海大学 人工智能与自动化学院**

**《机器学习实践》**

**能源消耗时间序列预测（中难度）**

**学 号 \_\_\_\_2206040224\_\_\_\_\_\_\_\_\_\_\_**

**专 业 \_\_\_\_智能科学与技术\_\_\_\_\_\_\_**

**学生姓名 \_\_\_\_刘滢\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师 \_\_\_\_张卓\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**完成时间 \_\_\_\_2025.1\_\_\_****\_\_\_\_\_\_\_\_\_\_\_\_**

1. **实验目的**

本实验旨在对 XGBoost、TCN 和 LSTM 三种模型在能源消耗时间序列预测任务中的表现进行对比分析。通过使用区域每小时能源消耗数据集，评估并比较不同模型的预测精度。实验将深入探讨各模型在时间序列预测中的优缺点、适用性以及改进方向。

**二、实验环境**

操作系统：Windows 11

编程语言：Python 3.9

深度学习框架：TensorFlow

主要库：

pandas：用于数据处理

scikit-learn：用于机器学习模型、数据预处理等

xgboost：用于训练 XGBoost 模型

matplotlib：用于绘图

numpy：用于数值计算

tensorflow：用于构建和训练深度学习模型（LSTM、TCN）

joblib：用于保存模型和处理数据

硬件环境：

CPU：支持 AVX2 和 FMA 指令集优化

**三、实验数据集**

数据集名称：主要使用 Kaggle 提供的每小时能源消耗数据集 (AEP\_hourly.csv，可直接更换不同区域的数据集文件)

数据内容：

包含每小时的能源消耗数据（单位：MW）

主要包含两列：Datetime（时间戳）、MW（能源消耗量）

数据集链接：

<https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption>

**四、实验方法**

4.1 数据预处理

时间格式处理：将 'Datetime' 列转换为日期时间格式并将其设置为 DataFrame 的索引。

Python:

# 确保数据集包含 'Datetime' 列并将其转化为日期时间格式

if 'Datetime' not in df.columns:

raise ValueError("'Datetime' 列在数据中不存在，请检查数据格式。")

df['Datetime'] = pd.to\_datetime(df['Datetime']) # 转换为日期时间格式

df.set\_index('Datetime', inplace=True) # 将'Datetime'列设置为索引

特征提取：

提取时间特征：小时、日、月、年、星期几。

构造滞后特征：利用前 24 小时的能源消耗数据作为当前时间的特征。

Python:

# 特征工程：提取时间特征

for df\_part in [train\_df, test\_df]:

df\_part['hour'] = df\_part.index.hour # 小时

df\_part['day'] = df\_part.index.day # 日期

df\_part['month'] = df\_part.index.month # 月份

df\_part['year'] = df\_part.index.year # 年份

# 添加滞后特征：根据目标列的前24小时数据生成滞后特征

for lag in range(1, 25):

train\_df[f'lag\_{lag}'] = train\_df[target\_column].shift(lag) # 为训练集添加滞后特征

test\_df[f'lag\_{lag}'] = test\_df[target\_column].shift(lag) # 为测试集添加滞后特征

缺失值处理：删除含有 NaN 值的行。

数据标准化：使用 MinMaxScaler 对数据进行归一化（适用于 LSTM 和 TCN 模型）。

Python:

# 删除包含NaN的行，确保数据整洁

train\_df.dropna(inplace=True)

test\_df.dropna(inplace=True)

4.2 模型构建

XGBoost：

使用 XGBRegressor 模型。

调整参数：n\_estimators、learning\_rate、max\_depth、subsample 等。

Python:

model = xgb.XGBRegressor(

n\_estimators=1000, # 树的个数

learning\_rate=0.1, # 学习率

max\_depth=5, # 树的最大深度

subsample=0.8, # 随机采样比例

colsample\_bytree=0.8, # 树的特征采样比例

random\_state=42 # 随机种子，确保结果可复现

)

TCN：

构建因果卷积网络（TCN）模型，结合 Dropout 和全连接层。

使用 Adam 优化器和 MSE（均方误差）损失函数进行训练。

Python：

def create\_tcn\_model(input\_shape, num\_filters=64, kernel\_size=3, dropout\_rate=0.2, learning\_rate=0.001):

"""创建一个TCN模型。"""

model = keras.Sequential([

Conv1D(filters=num\_filters, kernel\_size=kernel\_size, padding='causal', activation='relu', dilation\_rate=1, input\_shape=input\_shape),

Dropout(dropout\_rate),

Conv1D(filters=num\_filters, kernel\_size=kernel\_size, padding='causal', activation='relu', dilation\_rate=2),

Dropout(dropout\_rate),

Conv1D(filters=num\_filters, kernel\_size=kernel\_size, padding='causal', activation='relu', dilation\_rate=4),

Dropout(dropout\_rate),

Flatten(),

Dense(1)

])

optimizer = tf.keras.optimizers.Adam(learning\_rate=learning\_rate)

model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])

return model

LSTM：

构建包含 LSTM 层和全连接层的 LSTM 网络。

使用 Adam 优化器和 MSE 损失函数。

采用 EarlyStopping 防止过拟合。

Python:

def create\_lstm\_model(input\_shape):

model = Sequential()

model.add(LSTM(50, activation='relu', input\_shape=input\_shape))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')

return model

4.3 模型训练与评估

数据划分：将数据集按照 80%：20% 的比例划分为训练集和测试集，不打乱数据集顺序，确保时间序列的顺序。

训练过程：使用训练集训练每个模型，并在测试集上评估模型性能。

评估指标：均方误差（MSE）、均方根误差（RMSE）和平均绝对误差（MAE）。

4.4 结果可视化

绘制预测值与真实值的曲线图，直观比较不同模型的预测效果。

**五、实验结果**

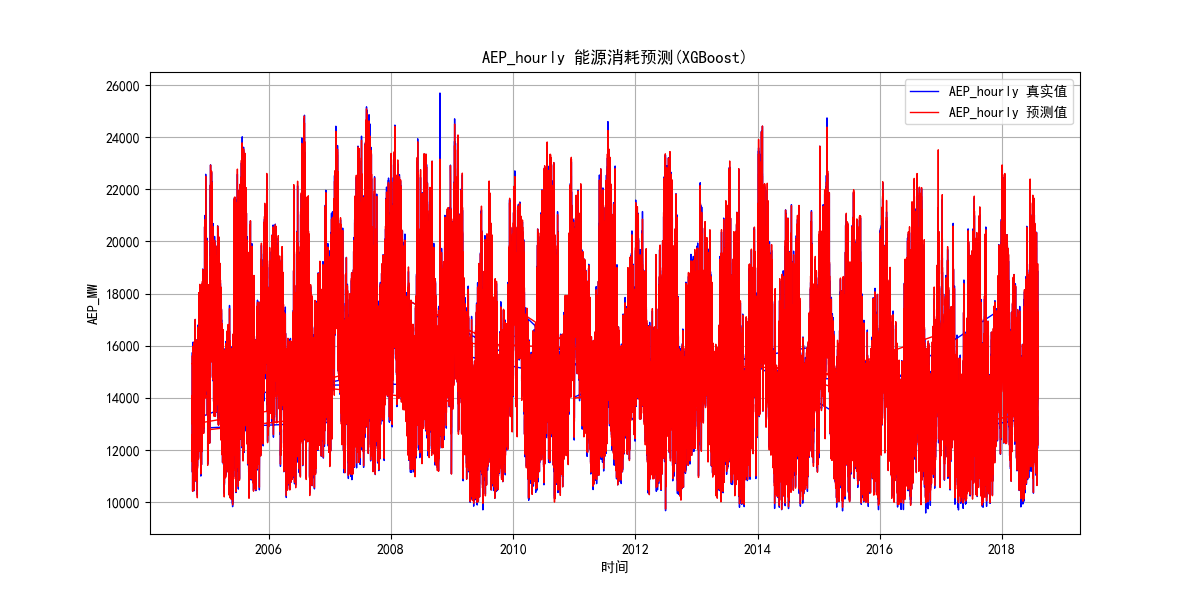
模型 MSE RMSE MAE 训练时间

XGBoost 85710.20 292.76 191.26 0:00:02.98

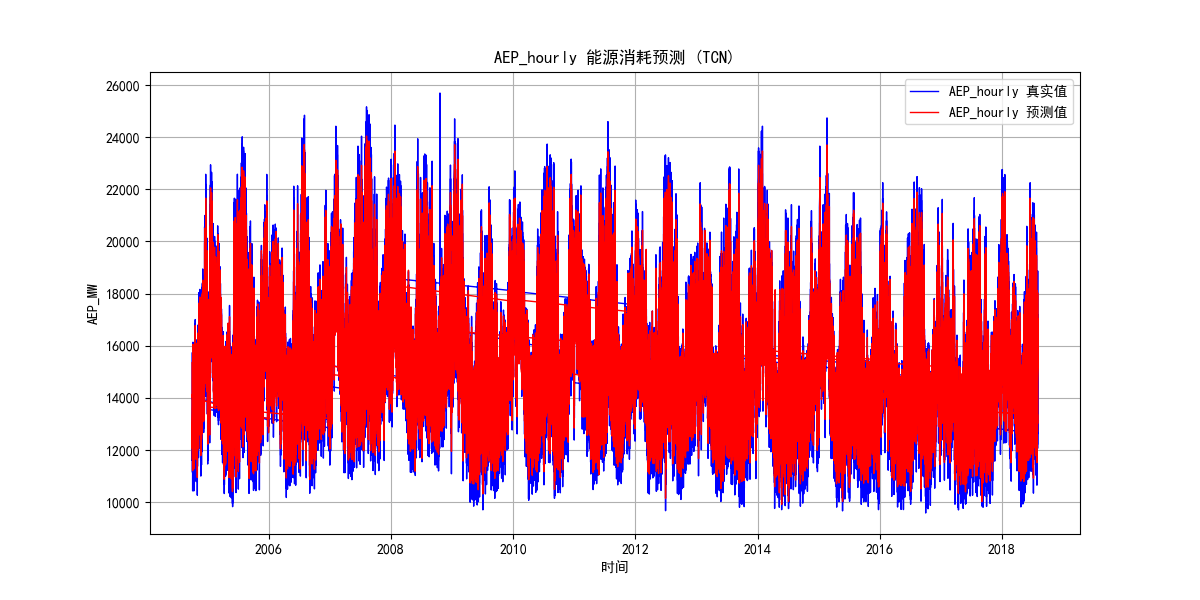
TCN 136457.49 369.40 256.80 0:03:04.07(20epoches)

LSTM 54324.12 233.08 136.03 0:05:06.73(20epoches)

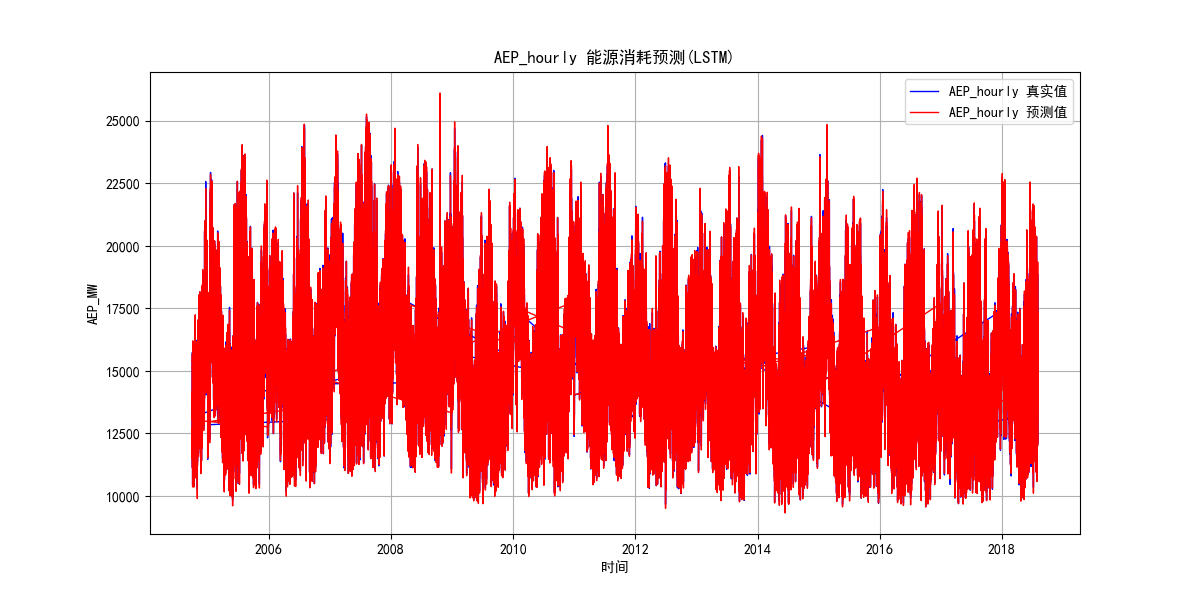
实验结果图



图一、XGBoost模型预测结果图



图二、TCN模型预测结果图



图三、LSTM模型预测结果图

XGBoost：虽然训练时间短，但预测精度相对较低，MSE、RMSE 和 MAE 均高于 LSTM。

TCN：训练时间较长，且预测效果较差，MSE、RMSE 和 MAE 均为三者中最高。

LSTM：在三种模型中表现最优，MSE、RMSE 和 MAE 均最低，表明其对时间序列数据的捕捉能力最好。

**六、结果分析与讨论**

XGBoost 作为强大的集成学习方法，在处理时间序列数据时虽然性能较好，但未能达到 LSTM 的精度。

优点：训练速度较快，适合于较大数据集。

缺点：在处理时间序列数据的长期依赖问题上存在局限性，效果较差。

LSTM 模型在本实验中的表现最佳，能有效捕捉时间序列数据中的长期依赖关系，适用于能源消耗预测这类时间序列数据。

优点：能处理长期依赖，且在复杂的非线性关系下表现较好。

缺点：训练时间较长，且对硬件要求较高。

TCN 模型的性能较差，可能是由于模型参数设置不优化或结构设计过于简单。

优点：相比 LSTM，更容易并行化训练。

缺点：在本任务中未能充分发挥其优势，可能需要更多的调优和更复杂的结构。

**七、结论**

对于能源消耗时间序列预测任务，LSTM 模型的预测性能最强，能够有效捕捉长期依赖关系，是最优选择。

XGBoost 是一个可行的备选方案，但在本实验中表现逊色于 LSTM。

TCN 模型的表现未达到预期，可能需要进一步的调优和优化才能发挥其优势。

在实际应用中，选择模型时需综合考虑数据特点、模型复杂度以及计算资源等因素。