Weather Data Analysis and Prediction

Uros Babic

May 31, 2024

1 Abstract

This report describes the extraction, preprocessing, training, and evaluation of machine learning models to predict temperature based on weather data. Random Forest and LSTM models are used, and their performances are compared. The study utilizes historical weather data from the OpenWeather API.

2 Introduction

In a project focused on determining the optimal placement for windmills, I discovered the OpenWeather API as a valuable source for historical weather data. This project extends that work by using the API to collect data for training machine learning models to predict temperature. The models used in this study are Random Forest and LSTM.

3 Data Description

The dataset contains the following instances:

- **Temperature (°C)**: The ambient temperature.
- **Humidity (%)**: The percentage of humidity in the air.
- **Wind Speed (m/s)**: The speed of the wind.
- **Weather Conditions**: A description of the weather conditions (e.g., clear sky, rain).

4 Data Mining Techniques

This section describes the data mining techniques used in this study: Random Forest and Long Short-Term Memory (LSTM) neural networks.

4.1 Random Forest

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the mean prediction of the individual trees. It helps improve the predictive accuracy and control overfitting. The model aggregates the votes from various decision trees to decide the final class of the test object.

4.2 LSTM Neural Networks

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies. LSTM networks are well-suited to predicting time series data, where the current prediction is influenced by previous data points. They contain memory cells that can maintain information for long periods, which makes them ideal for sequential data with temporal dependencies.

5 Data Extraction

The `data_extraction.py` script extracts historical weather data using the OpenWeather API for a specific location and time period. The data is then saved to a CSV file. The data is extracted from Belgrade, Serbia.

Listing 1: data_extraction.py

```
import requests
import pandas as pd
from datetime import datetime, timedelta

api_key = "4e45ef1900365aa87ed1695a28ac9c41"
latitude = 44.7866
longitude = 20.4489

end_date = datetime.now()
start_date = end_date - timedelta(days=25)

all_data = []

for day in range((end_date - start_date).days):
    current_date = start_date + timedelta(days=day)
    current_timestamp = int(current_date.timestamp())
    response = requests.get(f"https://history.openweathermap.
----org/data/2.5/history/city?lat={latitude}&lon={longitude}&
----type=hour&start={current_timestamp}&end={current_timestamp
----+86400}&appid={api_key}")
    data = response.json()
    if response.status_code == 200:
```

```
        for hour_data in data['list']:
            temperature = hour_data['main']['temp'] - 273.15
            humidity = hour_data['main']['humidity']
            wind_speed = hour_data['wind']['speed']
            weather_conditions = hour_data['weather'][0]['description']
            all_data.append([temperature, humidity, wind_speed,
            weather_conditions])
    else:
        print(f"Error: {data['message']}")

df = pd.DataFrame(all_data, columns=['Temperature', 'Humidity',
'Wind Speed', 'Weather Conditions'])
df.to_csv("weather_data.csv")
```

6 Data Preprocessing

The `data_preprocessing.py` script handles missing values and creates data windows for model training. It splits the data into training and test sets.

Listing 2: data_preprocessing.py

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

def preprocess_data(df, window_size=5):
    df = df.dropna()
    X, y = [], []
    features = ['Humidity', 'Wind Speed']
    target = 'Temperature'

    for i in range(len(df) - window_size):
        X.append(df[features].iloc[i:i + window_size].values)
        y.append(df[target].iloc[i + window_size])

    X = np.array(X)
    y = np.array(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

    return X_train, X_test, y_train, y_test
```

7 Model Training

The `model_training.py` script trains a Random Forest model and an LSTM model on the training data.

Listing 3: model_training.py

```
from sklearn.ensemble import RandomForestRegressor
from keras.models import Sequential
from keras.layers import LSTM, Dense
import joblib

def train_model(X_train, y_train):
    features = X_train.shape[2]
    rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    X_train_flat = X_train.reshape(X_train.shape[0], -1)
    rf_model.fit(X_train_flat, y_train)
    joblib.dump(rf_model, 'rf_trained_model.pkl')

    lstm_model = Sequential()
    lstm_model.add(LSTM(50, activation='relu',
    input_shape=(X_train.shape[1], features)))
    lstm_model.add(Dense(1))
    lstm_model.compile(optimizer='adam', loss='mse')
    lstm_model.fit(X_train, y_train, epochs=200, verbose=0)
    lstm_model.save('lstm_trained_model.h5')

    return rf_model, lstm_model
```

8 Model Evaluation

The `model_evaluation.py` script evaluates the trained models using the test data and calculates the Mean Absolute Error (MAE) for both models.

Listing 4: model_evaluation.py

```
from sklearn.metrics import mean_absolute_error
import joblib
import numpy as np

def evaluate_model(rf_model, lstm_model, X_test, y_test):
    X_test_flat = X_test.reshape(X_test.shape[0], -1)
    rf_predictions = rf_model.predict(X_test_flat)
    rf_mae = mean_absolute_error(y_test, rf_predictions)
    lstm_predictions = lstm_model.predict(X_test)
    lstm_mae = mean_absolute_error(y_test, lstm_predictions)
```

```
    return rf_mae, lstm_mae
```

9 Main Program

The main program loads the data, preprocesses it, trains the models, and evaluates them.

Listing 5: main.py

```
import pandas as pd
import data_preprocessing
import model_training
import model_evaluation

df = pd.read_csv('weather_data.csv')
X_train, X_test, y_train, y_test =
data_preprocessing.preprocess_data(df, window_size=5)
rf_model, lstm_model = model_training.train_model(X_train, y_train)
rf_mae, lstm_mae = model_evaluation.evaluate_model
(rf_model, lstm_model, X_test, y_test)

print(f"Random Forest Mean Absolute Error: {rf_mae}")
print(f"LSTM Mean Absolute Error: {lstm_mae}")
```

10 Test Results

The following test results were obtained for different periods:

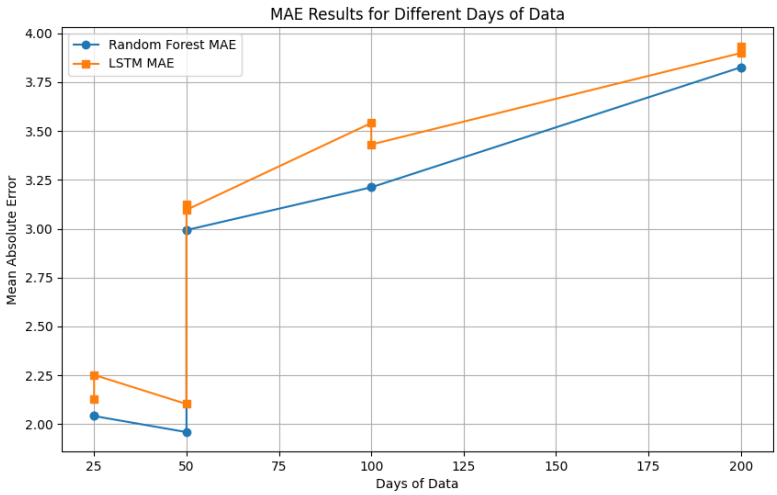

Figure 1: MAE Results for Different Days of Data

Test (Days)	Random Forest MAE	LSTM MAE
Test 1 (50 days)	1.9596	2.1033
Test 2 (50 days)	2.9930	3.1264
Test 3 (50 days)	2.9930	3.0965
Test 4 (100 days)	3.2118	3.5408
Test 5 (100 days)	3.2118	3.4308
Test 6 (200 days)	3.8257	3.8987
Test 7 (200 days)	3.8257	3.9326
Test 8 (25 days)	2.0418	2.1307
Test 9 (25 days)	2.0418	2.2522

11 Conclusion

From the results, it can be observed that as we increase the number of days for prediction, the Mean Absolute Error (MAE). Likely to the difference between different weather behaviour in different seasons. Since the weather in Belgrade changes a lot depending on what season it is.

12 Code Repository

The code for this project can be found at the following GitHub repository: GitHub.

References

[1] OpenWeatherMap, *Historical weather data*, `https://openweathermap.org/history`

[2] F. Chollet et al., *Keras*, `https://keras.io`,

[3] Jason Brownlee PhD *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras* `https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/`

[4] W. McKinney, *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56, 2010.

[5] S. van der Walt, S. C. Colbert, and G. Varoquaux, *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science & Engineering, 13(2), 22-30, 2011.

[6] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, 2825-2830, 2011.