知能情報実験 III（データマイニング班）
来年の沖縄が蒸し暑くなるのはいつ?

225767A 荒井隆, 215731E 神谷嘉人, 215704H 柳原克己,
205716G 川口朝陽, 215757J イデル・テムーレン

提出日：2024 年 7 月 25 日

目次

概要

毎年暑くなると多くの人が熱中症により命を落としている。総務省の令和 5 年 (5 月から 9 月) の熱中症による救急搬送状況 [1] によると、令和 5 年の 5 月から 9 月の全国の熱中症による搬送者数は 91,467 人で、死亡者数は 107 人である。この現状をデータマイニングの授業で学んだ技術を使い、過去の熱中症になりやすくなる不快指数が高くなる日から、来年の不快指数が高くなり、熱中症になりやすくなる日を予測し、熱中症に備えることで、熱中症の患者および死者数が減少するようにしたいと考えたのが、本実験を始めた動機である。実験方法としては、データとして気象庁の過去 30 年分を使う。1994 年から 2023 年は 1 年分のデータを取った。2024 年は 1 月 1 日から 6 月 30 日までの気温と湿度データを用いて、ランダムフォレストを用いて予測モデルを構築し、次に 2024 年の気温と湿度データを生成して予測を行い、結果をグラフで可視化し、不快指数が 80 を超える予測日を報告する。その結果、今年の不快指数が 80 を超え、熱中症のリスクが高くなる日は、6 月 22 日であるという結果が得られた。この実験結果より、来年は 6 月 22 日からこまめに水を飲んだり、エアコンを適切に使用するなどの熱中症対策を行う必要があると考えられる。また、今後の課題として、気温と湿度のみならず、日光や風の有無などのパラメータを含めて実験を行うことでより精度の高い予測ができると考えられる。

1　実験の目的と達成目標

今年のいつから熱中症になりやすくなるのか予測し、熱中症にそなえることを実験目的とする。達成目標としては、このプロジェクトの達成目標は、過去 30 年分の気温と湿度データを用いて沖縄の不快指数が 80 を超える日を予測し、2024 年の 3 月 1 日から 5 月 31 日までの期間に適用することである。まず、データセットをクレンジングし、気温と湿度を基に不快指数の計算を行う。その後、データをトレーニングセットとテストセットに分割し、ランダムフォレストを用いて予測モデルを構築する。モデルの精度を評価し、2024 年の気温と湿度データを生成して予測を行う。結果はグラフで可視化し、不快指数が 80 を超える予測日を報告する。この実験を通して、データの前処理、特徴量設計、モデル構築、結果の可視化を行い、熱中症の貢献に寄与する。

2　テーマ「来年の沖縄が蒸し暑くなるのはいつ?」とは

本グループでは熱中症予測 (熱中症になりやすくなる日を予測) することで、熱中症に備えることを対象問題として設定した。熱中症とは気象庁 [2] 高温、多湿、風が弱いなどの環境や、激しい労働や運動によって体にたまる熱などに体が十分に対応できず体内の水分や塩分のバランスが崩れ、また体温の調節機構が破綻するなどの原因で起こる症状の総称であり、来年の蒸し暑くなる日を予測することで熱中症予防に寄与する。また、熱中症は健康に重大な影響を与える問題である。特に高齢者や子供、持病を持つ人々にとっては命に関わるリスクとなる。

そこで、本実験では来年の蒸し暑くなる日を機械学習を用いて予測することで事前に対策を講じ、多くの命を守ることに寄与することを目的とする。例えば、公共施設やイベントの運営方針の見直し、労働環境の調整、学校の体育活動のスケジュール変更などが考えられる。よって、今回の実験を行うことで、「熱中症になりやすい」という予測が出たとき、様々なリスクに対して事前に備えることができると考えられるため、今回の実験テーマにすることを決めた。

3　理論

本章では、今回の実験で必要な知識について説明する。

3.1　不快指数

不快指数は、気温と湿度の組み合わせから算出される指標で、人間がどの程度蒸し暑さを感じるかを表します。75 以上では半数以上が、80 以上では全員が不快を感じるとされている [3]。また、不快指数は $0.81 \times$ 気温 $+ 0.01 \times$ 湿度 $\times (0.99 \times$ 気温 $- 14.3) + 46.3$ という公式によって計算することができる。

3.2　線形回帰モデル

線形回帰とは、既知のデータ値を使用して別の未知のデータの値を予測するデータ分析手法であり、未知の従属変数と既知の独立変数を線形方程式としてモデル化する [4]。

3.3　Lasso 回帰モデル

Lasso 回帰は過学習を抑えるために L1 正則化を取り入れた線形回帰モデルであり，目的変数の予測能力が低い変数を回帰モデルに多く含めて推定を行っても，それらの回帰係数が 0 になりやすい性質を持つ。[5] そのため、特徴量の選択が重要な場合や、モデルの解釈性を高めたい場合に適している。Lasso 回帰には Alpha というパラメータがある。Alpha は正則化強度を調整し、A が大きいほど正則化効果が強くなるためグリッドサーチを用いて最適化できる。そして Interval と

いう独自の変数は値を大きくすると誤差が拡張しやすいが、予測した気温に対して、傾向をつかみやすい。Interval の役割として例えば Interval を 6 に設定すると過去 6 日間の気温からその翌日の気温を予測することができる。Interval が大きいと、より長期的なトレンドや季節の変動をとらえる。短期的なノイズの影響を減少させるメリットがある一方、計算コストの増加というデメリットがある。

4　実験方法

本実験では以下の手順で行った。

1. データセット構築
2. モデルの選定
3. 気温・湿度の予測
4. 不快指数の計算

4.1　データセット構築

本実験で用いたデータセットは気象庁の過去 30 年分の気温と湿度データ [6] を用いた。表 1 に取得するデータのサンプルを示す。これは気象庁から取得した 1994 年の 1 月 1 日から 1 月 3 日のデータである。 1994 から 2023 年までの気温, 降水量, 日照時間, 天気概要 (天気概要/昼: 6 時から 18 時) をダウンロードした。また、予測した値と実際の値を比較するため，気温，湿度のみ 2024 年の 1/1〜6/30 まで追加で取得して、データセット構築を行った。なお、今回取得した気温や湿度は 1 日の平均気温と平均湿度である。

表 1　気象庁から取得したデータのサンプル

年月日	平均湿度 (%)	平均気温 (℃)	天気概要
1994/1/1	53	16.7	晴時々曇
1994/1/2	64	17.9	晴
1994/1/3	74	19.4	晴時々曇一時雨

4.2　モデル選定

本実験では予測モデルとして最初に線形回帰を用いて、そのあと Lasso 回帰を選んだ。線形回帰を用いた理由としては連続値を学習し、気温や湿度やほかの特徴量との線形関係を仮定して、予測するので連続値である気温や湿度データの学習に適しているからであり、Lasso 回帰を用いた理由としては、線形回帰では過学習を起こした可能性があったため、過学習を防ぐ正則化という手法が使用されている Lasso 回帰を用いて精度の向上を図った。

4.3　気温・湿度予測

リスト 1 で 2023 年までのデータから 2024 の気温を予測している．読み込んだデータに前処理として標準化を施し，作成した訓練データを用いて線形回帰のモデルを訓練し，実際に 2024 年の気温を予測した．線形回帰と Lasso 回帰の二つの学習機を用いて予測を行い、不快指数の精度誤差が、一般的な気温予測の精度目標である 1.2 から 2.0 度 [7] という基準を満たしているのか比較する。また、線形回帰モデルで不快指数を求めた際に気温と湿度以外の特徴量を追加して精度誤差がどのようになるかについても実験を行った。

ソースコード 1　気温の変化

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 気温データを読み込む
temper_data = pd.read_csv('new_naha_kion.csv', encoding="utf-8")

# 過去 6日分を特徴量､その翌日の気温を目的変数とする訓練データを作成する関数
def make_data(data, interval):
    x = []
    y = []
    temps = list(data["気温"])
    for i in range(len(temps)):
        if i < interval: continue
        y.append(temps[i])
        xa = []
        for p in range(interval):
            d = i + p - interval
            xa.append(temps[d])
        x.append(xa)
    return (x, y)

# 1994年から 2023年までのデータを訓練データとして使用
train_year = (temper_data["年"] <= 2023)
interval = 6

train_x, train_y = make_data(temper_data[train_year], interval)

# 訓練データの標準化
```

```
scaler = StandardScaler()
train_x_scaled = scaler.fit_transform(train_x)

# 線形回帰モデルの訓練
lr = LinearRegression()
lr.fit(train_x_scaled, train_y)

# 2023年のデータを使用して2024年の気温を予測
test_year_2023 = (temper_data["年"] == 2023)
test_x_2023, _ = make_data(temper_data[test_year_2023], interval)
test_x_2023_scaled = scaler.transform(test_x_2023)
pre_y_2024 = lr.predict(test_x_2023_scaled)

# 予測した2024年の気温データを保存
# ここで月と日を生成
test_data_2023 = temper_data[test_year_2023].iloc[interval:] # 最初の
    interval日を除く
months_2024 = test_data_2023['月'].values
days_2024 = test_data_2023['日'].values

predicted_2024_df = pd.DataFrame({
    '年': 2024,
    '月': months_2024,
    '日': days_2024,
    '予測気温': pre_y_2024
})

# 2023年の気温データを取得
actual_2023_df = temper_data[test_year_2023][['月', '日', '気温']].iloc[
    interval:]

# 2023年と2024年の気温データを比較するためにプロット
plt.figure(figsize=(10, 6), dpi=100)
plt.plot(actual_2023_df['気温'].values, c='r', label='2023年の実際の気温')
plt.plot(pre_y_2024, c='b', label='2024年の予測気温')
plt.legend()
plt.xlabel('日数')
plt.ylabel('気温 (℃)')
plt.title('2023年の実際の気温と2024年の予測気温の比較')
plt.savefig('tenki-kion-2023_vs_2024.png')
plt.show()

```

```
# 予測した2024年の気温データをCSVファイルとして保存
predicted_2024_df.to_csv('predicted_2024_temperatures.csv', index=False,
    encoding='utf-8')


from sklearn.metrics import mean_absolute_error

# 実際の値
actual_values = actual_2023_df['気温'].values

# MAEを計算
mae = mean_absolute_error(actual_values, pre_y_2024)
print(f"平均絶対誤差 (MAE): {mae:.2f} ℃")
```

次に、リスト 2 に 2023 年までのデータから 2024 の湿度を予測するプログラムを示す．なお、リスト 1 と重なるところは省略し差分のみを示した。リスト 1 と異なる点としては、データの読み込み部分、関数内の特徴量と目的変数の列名やプロットのラベルと軸ラベルなどであり、基本的な動作はいずれも同じである。このプログラムでは、まず年月日と湿度の情報が含まれている取得した CSV ファイルを読み込む。次に、make_data 関数で過去 6 日分の湿度を特徴量とし、その翌日の湿度を目的変数として訓練データを作成する。また、interval は過去何日分のデータを使用するかを指定する。この関数は、特徴量のリスト x と目的変数のリスト y を返す。そして、1994 年から 2023 年までのデータを使用して訓練データを作成する。このとき、interval を 6 日として、特徴量と目的変数を生成する。次に特徴量を標準化する。データの平均を 0、分散を 1 にして標準化することで、異なるスケールの特徴量を持つデータの学習が容易になる。その次に、標準化した訓練データを用いて線形回帰モデルを訓練し、最後に 2023 年のデータを使用して、2024 年の湿度を予測する。

ソースコード 2　湿度の変化

```
# 湿度データを読み込む
temper_data = pd.read_csv('new_naha_shitsudo.csv', encoding="utf-8")

# 過去6日分を特徴量､その翌日の湿度を目的変数とする訓練データを作成する関数
def make_data(data, interval):
    x = []
    y = []
    temps = list(data["湿度"])
    for i in range(len(temps)):
        if i < interval: continue
        y.append(temps[i])
        xa = []
        for p in range(interval):
```

```
            d = i + p - interval
            xa.append(temps[d])
        x.append(xa)
    return (x, y)

# 1994年から2023年までのデータを訓練データとして使用
train_year = (temper_data["年"] <= 2023)
interval = 6

train_x, train_y = make_data(temper_data[train_year], interval)

# 訓練データの標準化
scaler = StandardScaler()
train_x_scaled = scaler.fit_transform(train_x)

# 線形回帰モデルの訓練
lr = LinearRegression()
lr.fit(train_x_scaled, train_y)

# 2023年のデータを使用して2024年の湿度を予測
test_year_2023 = (temper_data["年"] == 2023)
test_x_2023, _ = make_data(temper_data[test_year_2023], interval)
test_x_2023_scaled = scaler.transform(test_x_2023)
pre_y_2024 = lr.predict(test_x_2023_scaled)

# 予測した2024年の湿度データを保存
# ここで月と日を生成
test_data_2023 = temper_data[test_year_2023].iloc[interval:] # 最初の
    interval日を除く
months_2024 = test_data_2023['月'].values
days_2024 = test_data_2023['日'].values

predicted_2024_df = pd.DataFrame({
    '年': 2024,
    '月': months_2024,
    '日': days_2024,
    '予測湿度': pre_y_2024
})

# 2023年の湿度データを取得
actual_2023_df = temper_data[test_year_2023][['月', '日', '湿度']].iloc[
    interval:]
```

```
54
55 # 2023年と 2024年の湿度データを比較するためにプロット
56 plt.figure(figsize=(10, 6), dpi=100)
57 plt.plot(actual_2023_df['湿度'].values, c='r', label='2023年の実際の湿度')
58 plt.plot(pre_y_2024, c='b', label='2024年の予測湿度')
59 plt.legend()
60 plt.xlabel('日数')
61 plt.ylabel('湿度 (%)')
62 plt.title('2023年の実際の湿度と 2024年の予測湿度の比較')
63 plt.savefig('tenki-shitsudo-2023_vs_2024.png')
64 plt.show()
65
66 # 予測した 2024年の湿度データをCSV ファイルとして保存
67 predicted_2024_df.to_csv('predicted_2024_humidity.csv', index=False, encoding
      ='utf-8')
```

4.4 不快指数の計算

気温と湿度の予測結果を用いて不快指数の計算を行った。その際に用いたプログラムをリスト 3 に示す。このプログラムでは、気温と温度の両データを同じ日付ごと取得し気温と湿度から不快指数を求める公式 $0.81 \times$ 気温 $+ 0.01 \times$ 湿度 $\times (0.99 \times$ 気温 $- 14.3) + 46.3$ に代入しグラフを表示している。

ソースコード 3　不快指数

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import matplotlib.dates as mdates
4
5  # 予測した気温データと湿度データを読み込む
6  temperature_df = pd.read_csv('predicted_2024_temperatures.csv', encoding="utf
       -8")
7  humidity_df = pd.read_csv('predicted_2024_humidity.csv', encoding="utf-8")
8
9  # 不快指数を計算する関数
10 def calculate_discomfort_index(temp, hum):
11     return 0.81 * temp + 0.01 * hum * (0.99 * temp - 14.3) + 46.3
12
13 # データをマージして不快指数を計算する
14 merged_df = pd.merge(temperature_df, humidity_df, on=['年', '月', '日'])
15 merged_df['不快指数'] = merged_df.apply(lambda row: calculate_discomfort_index
       (row['予測気温'], row['予測湿度']), axis=1)
```

```
16
17 # 列名を英語に変更
18 merged_df.rename(columns={'年': 'year', '月': 'month', '日': 'day'}, inplace=
       True)
19
20 # 日付カラムを追加
21 merged_df['date'] = pd.to_datetime(merged_df[['year', 'month', 'day']])
22
23 # 結果を表示
24 print(merged_df.head())
25
26 # グラフの作成
27 plt.figure(figsize=(15, 8))
28 plt.plot(merged_df['date'], merged_df['不快指数'], label='2024年の不快指数',
       color='purple')
29 plt.xlabel('日付')
30 plt.ylabel('不快指数')
31 plt.title('2024年の不快指数の推移')
32 plt.legend()
33 plt.grid(True)
34
35 # 日付フォーマットと目盛りの設定
36 plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=1)) # 1週間間
       隔で目盛りを設定
37 plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d')) # 日付フ
       ォーマットを設定
38
39 plt.xticks(rotation=45) # 日付ラベルを 45度回転して表示
40 plt.tight_layout() # レイアウトを調整
41 plt.savefig('discomfort_index_2024.png')
42 plt.show()
```

5 実験結果

気温の予測結果と実際の予測結果を比較した図 1 から、ある程度傾向をつかめていることがわかる。3 章に示した実験方法を用いて実験を行った結果不快指数は図 2 に示した通りになった。その結果、今年の不快指数が 80 を超え、熱中症のリスクが高くなる日は、6 月 22 日であるという結果が得られた。次に、実際の不快指数とこの結果を比較すると図 3 のようになった。これより、予測した不快指数ではある程度傾向がつかめていることがわかる。また、実際の不快指数と予測した不快指数間で回帰分析を用いた場合誤差が 2.04 度と良好な精度を示した。また、これと同様に

Lasso 回帰を用いた場合では誤差が 1.73 度となり、一般的な気温予測の精度目標である 1.2 から 2.0 度という基準を満たしていた。

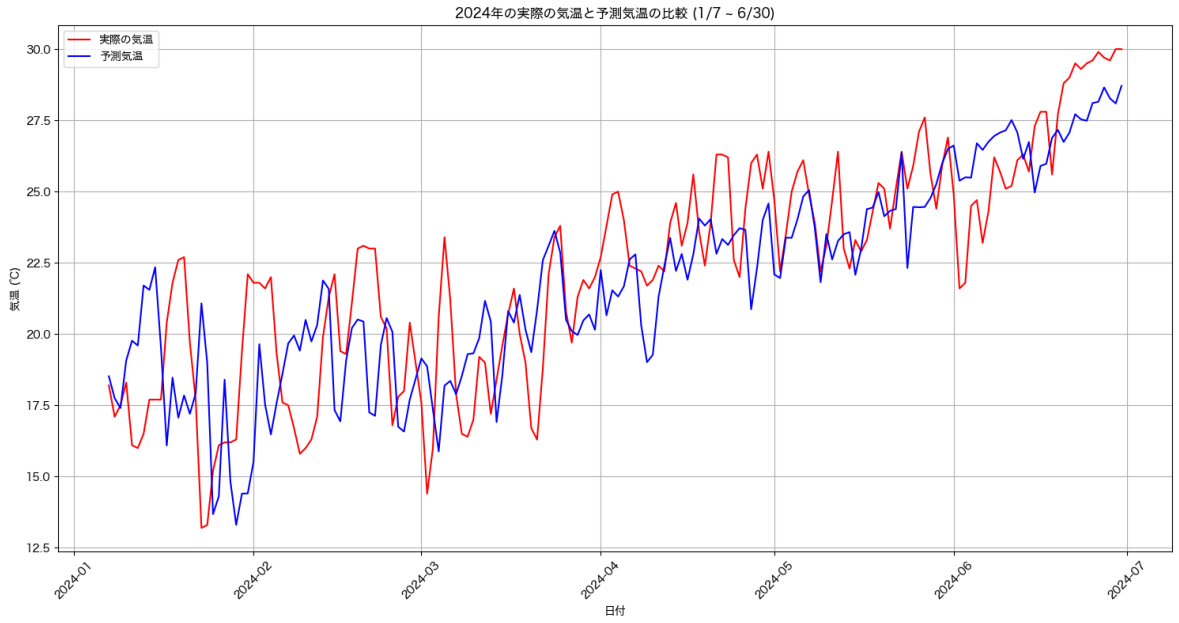

図 1　気温予測結果と実際の気温の比較

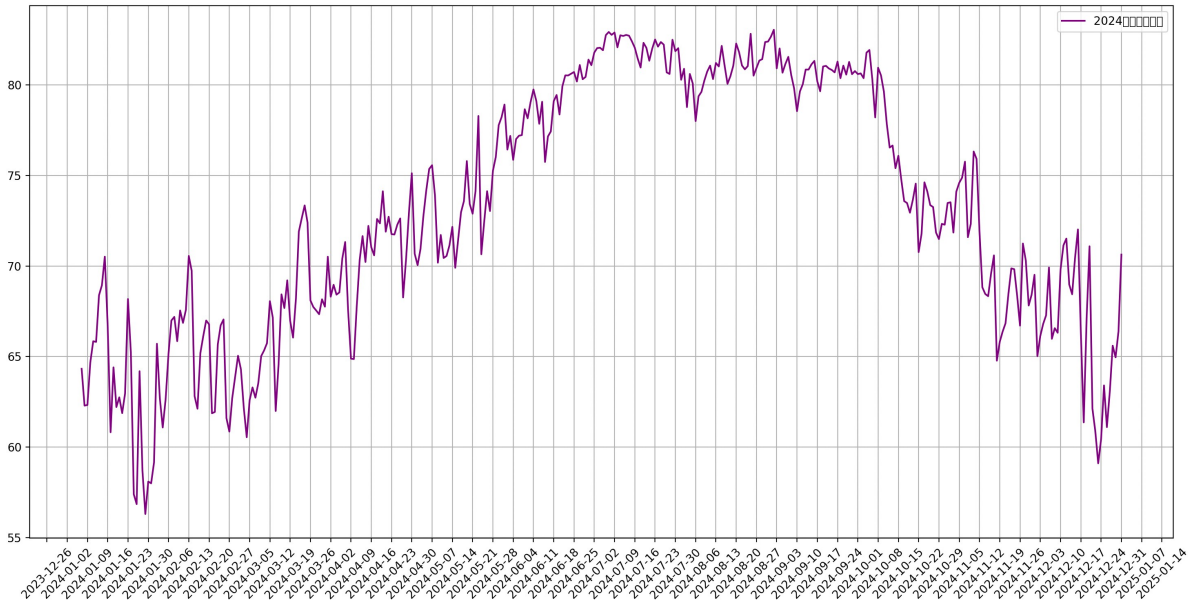

図 2　不快指数予測結果

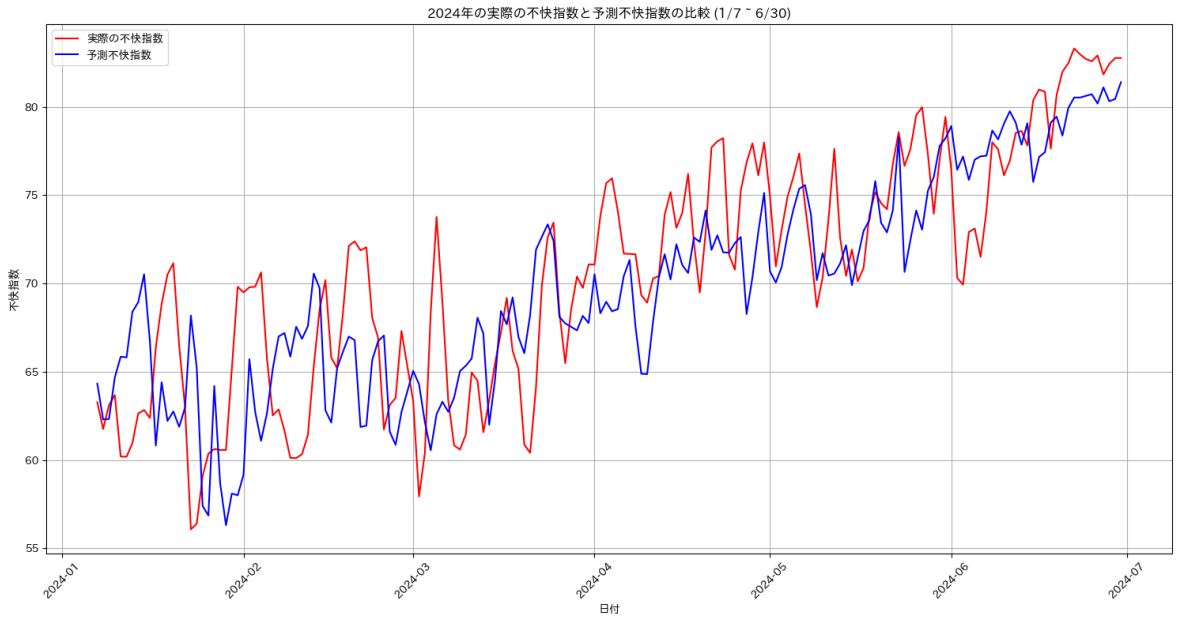

図 3　予測した不快指数と実際の不快指数の比較

6　考察

本実験より、気温の予測においては線形回帰および Lasso 回帰が気温の予測に適していたのではないかと考えられる。その理由として、使用した Lasso 回帰モデルは、過去の気温データの変動パターンが比較的安定しており、回帰分析による予測が効果的に機能したことが示されていることから、不要な特徴量をゼロにすることでモデルの解釈性と予測精度を高める効果があったからであると考えられる。一方で、湿度の予測では、気温の予測に比べ精度が悪かったが、これは 1 ヶ月後の天気予報が限界の中、天候に非常に影響を受ける湿度を最大 1 年後まで予測するのがかなり厳しいことが原因であると考えられる。したがって、長期的な予測が可能なモデルへの変更 (LSTM や GRU) や、非線形性を捉える能力が高いランダムフォレスト、XGBoost などを検討することでさらに高い精度で予測できると考えられる。そして、気温と湿度のデータから得られる不快指数の予測では、気温の予測精度が高かったためある程度正確に予測することができていたが、湿度の予測を改善させることで、さらに不快指数の予測向上に期待できると考えられる。以上の考察より、今後の課題として、湿度予測モデルの改善、湿度に影響を与える他の気象要素（例：気圧、風速など）などの新しい特徴量の導入をすることで、予測精度の向上を図る必要があると考えられるが、本実験では、新しい特徴量を追加することでむしろ予測精度が下がってしまったため、ニューラルネットワークなどのより複雑なモデルを検討し、予測の精度を向上させることが必要であると考えられる。そして、那覇以外の名護などの地域のデータを収集し、より多くのデータを活用することで、予測モデルの精度と信頼性を向上させることが可能になると考えられる。また、今回の実験では、

データを得ることができなかったため、１日の平均気温、平均湿度から不快指数を導出したが、日中のみのデータが取得できれば、さらに精度を上げることができるのみならず、データ量が減少することから計算速度が速くなると考えられる。次に、実用化への展開として、本分析の成果を基に、一般の方々にも役立つアプリを開発することができると考えられる。具体的には、イベントの日程調整、クーラーの設置時期の調整、熱中症予防などの情報を提供するアプリが考えられる。

7　まとめ

この実験は、過去の気象データから気温・湿度の予測を行い、熱中症に備えることを目的としている．ここでは”不快指数”という蒸し暑さを表す指数を用いる．不快指数は気温・湿度により求めることができるため，過去の気温・湿度データから 2024 年の気温・湿度を Lasso 回帰により予測し，不快指数を計算した．また気温の予測に湿度，最高気温，降水量，日照時間，天気概況の特徴量を用いて精度の向上を図った．

2024 の実際のデータ（1/1 から 6/30）と予測したデータを比較すると，気温の精度（MAE）は比較的良好であり、決定係数という実測値と予測値の一致度を示す指標についても 0.55 と中程度の精度になっており過去データと 2024 年における予測値との間に一定の相関関係があったが，それに比べて湿度の精度が MAE が 10.231% で非常に高い誤差率となっており、各日の平均で 10% もの誤差があり、誤差が非常に大きく、決定係数も-0.06 と負の値になっており、過去データと 2024 年の予測値に相関関係があるとは考えにくく、湿度の予測の精度は極めて悪かった．それに伴い不快指数の精度も課題が残る結果となった．そのため、不快指数の精度の向上を目指すために，湿度の予測で適切なパラメータの設定や，湿度に影響を及ぼす気圧，風速などの特徴量の追加を検討し改良する必要がある．

また、熱中症予防については現行の天気予報アプリでも出していることを踏まえると本当に役に立つのか、という点については、不快指数ではなく暑さ指数を用い、かつ、風の強さや日の強さなど様々なデータを使うことで精度を向上させ、熱中症予防に特化した予測アプリを作ることで差別化を図っていけるのではないかと考えられる。

参考文献

[1] 令和５年（５月から９月）の熱中症による救急搬送状況 ,https://www.fdma.go.jp/pressrelease/houdou/items/d7e69bd2038c29f307d3b517fe2ee80610badb63.pdf, 2020/06/06.

[2] 気象庁 熱中症対策に関する用語, https://www.jma.go.jp/jma/kishou/know/yougo_hp/nettyusho.html, 2020/05/30.

[3] Tenki.jp, https://tenki.jp/indexes/discomfort/, 2020/05/30.

[4] Amazon Web Services, 線形回帰とは何ですか?, https://aws.amazon.com/jp/what-is/linear-regression/, 2020/06/06.

[5] 内田俊博,Lasso 回帰による幸福度分析, https://econo.chukyo-u.ac.jp/academicInfo/cerPdf/cer32_01.pdf,2020/06/06.

[6] 気象庁, 過去の気象データ・ダウンロード, https://www.data.jma.go.jp/gmd/risk/obsdl/, 2020/05/30.

[7] 気象庁, 天気予報の精度の例年値とその特徴, https://www.data.jma.go.jp/yoho/kensho/expln_reinen.html, 2020/06/30.