



Earthly Insights Data Science Project

By Shai Shillo & Ofek Daida



Introduction



ברוכים הבאים למצגת הפרויקט שלנו במדעי הנתונים.

נושא המחקר שלנו מתעסק בחקירת דפוסי נתוני האקלים ופרט הטמפרטורה ברחבי כדור הארץ.

שאלת המחקר היא:

האם ניתן לחזות את דפוסי הטמפרטורה השונים של כדור הארץ?

בחרנו בשאלת מחקר זאת מכיוון שלדעינו מדויק בנושא חשוב וакוטי המתמודד עם הקיום והשמירה על הכדור בו אנו חיים.

כיום, יש גיבוי רחב למגמה הקודרת של קצב הת חממות כדור הארץ.

אנו מאמינים שככל ניסיון להסביר את הציבור (ואתנו) לרכוש מידע ותובנות חדשות בנושא הוא חשוב ומשמעותי.

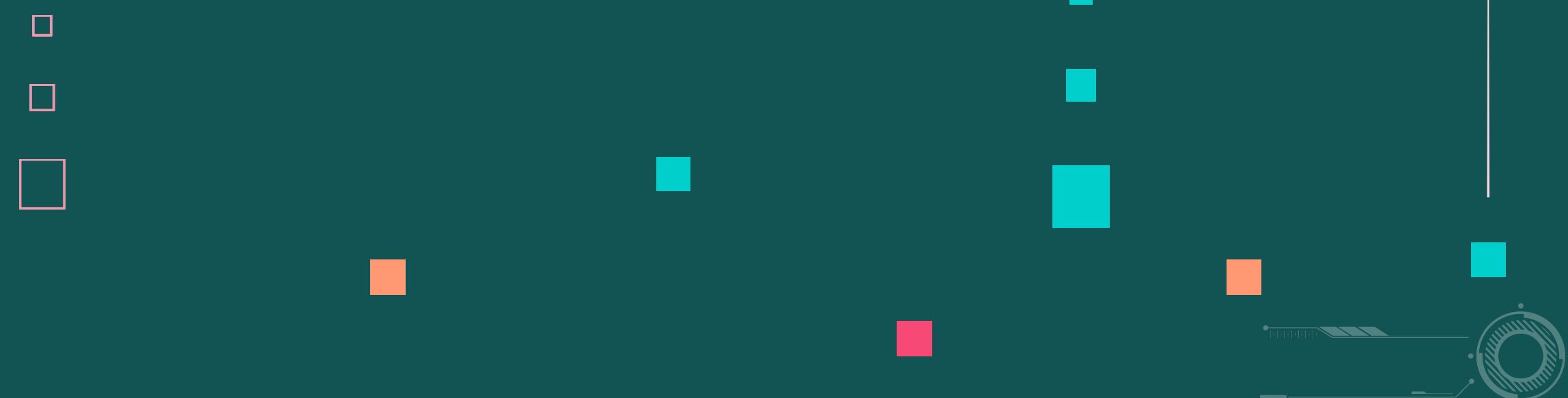


Introduction

במסגרת המחקר, אנו משתמשים במודל למדידת מכונה המסתמך על מאפיינים שונים כדי להזות את הטמפרטורה ברחבי העולם.

המטרה העיקרית היא להבין את הדפוסים השונים של הטמפרטורה, בנוסף למוגמות השונות שלהם והיכולת להזותם.

מטרת המחקר היא לתרום להבנה טובה יותר של תהליכי האקלים וקשרם בין מאפייניהם השונים בצדור הארץ.



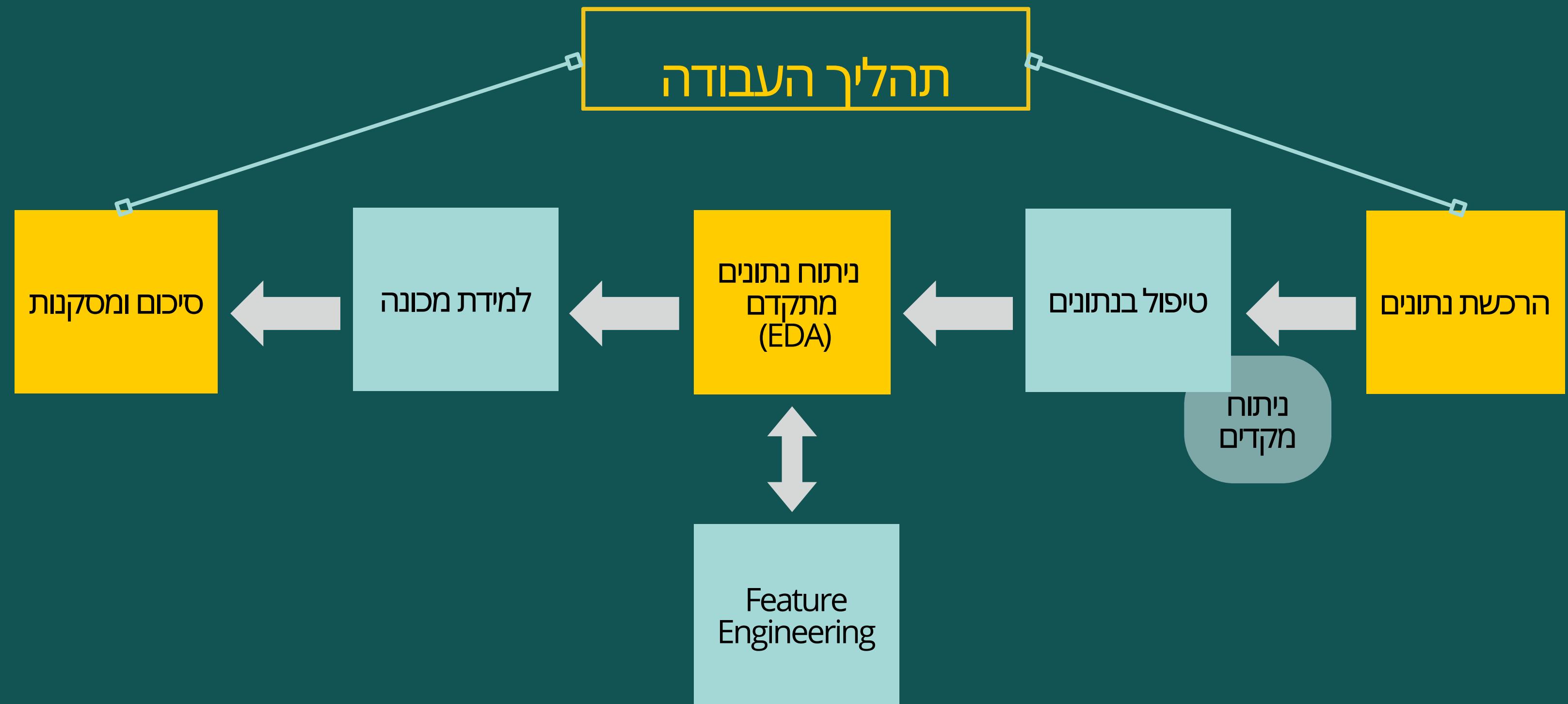
האתר ממנו בחרנו להרכיש את המידע

Weather Underground מתגראת את שיתוף המידע הקונבנציונלי על מזג האוויר מאז 1993.



הם רואים את עצמם כחלוצים בתחום, ומחפשים כל הזמן מערכי נתונים וטכנולוגיות חדשות לשיפור הגישה לנוטרי מזג אוויר משמעותיים מרחבי העולם.

במהלך ההרכשה החליטו להתרכז ב-32 מדינות הפרוסות ברחבי כדור הארץ, על מנת לנסות ולתפוס מדגם מייצג של האקלים והטמפרטורות השונות ברחבי הכדור.



שלב ראשון - הרכשת הנתונים

ניסיון לשימוש בגישה
Multi-threading

טיפול בזמן ריצה בנתונים
חסרים חישוב ממוצעים
ושכיחים ו"שיטוח" הנתונים
לרמה יומית

יצור דוח דפים שגויים/
כישלון הרכשה

ריצה
הרכשה לקונסול בזמן
קלבת נתונים על סטטוס

קוד רובוטי המאפשר
טיפול מגוון בשגיאות:

- ניתוק רשת קזרים
- דף ריק או שגוי
- נתונים חסרים בזמן
הרכשה

תנועה
אפליקטיבית
בין דפי האתר
השוניים
(Selenium & Bs4)

תנווה אפליקטיבית בין דפי האתר השונים



במהלך פיתוח הzahlן למען הרכשת הנתונים, היה חשוב לנו לפתחו בצורה המותאמת לצרכי היום יום של מדען הנתונים באתרים השונים.

אתר אותו בחרנו להרכיש: [wunderground.com](https://www.wunderground.com)

תפריט החיפוש בנוי בJavaScript עליו היינו צריכים לעורר את התנווה השונות, למעבר בין הדפים אותם נרצה להרכיש על מנת לעשות זאת השתמשנו בסיפוריות פיתון:

- Selenium
- Bs4 - Beautiful Soup

```
get_weather_data( year, month, day, city, browser):
try:
    wait = WebDriverWait(browser, 20)
    browser.get('https://www.wunderground.com/history')
    wait.until(EC.presence_of_element_located((By.ID, 'yearSelection')))
    wait.until(EC.presence_of_element_located((By.ID, 'historySearch')))

    # Select year, month, and day from the drop-down menus
    actions = ActionChains(browser)
    year_select = Select(browser.find_element(By.ID, 'yearSelection'))
    year_select.select_by_value(str((2023-year))+':'+str(year))
    month_select = Select(browser.find_element(By.ID, 'monthSelection'))
    month_select.select_by_value(str(month))
    day_select = Select(browser.find_element(By.ID, 'daySelection'))
    day_select.select_by_value(str((day-1))+':'+str(day))

    # Waiting for choosing destination
    WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.CSS_SELECTOR, "input#historySearch[name='historySearch']")))
    WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.XPATH, "//ul[@class='ui-autocomplete']/li[@class='ui-autocomplete-item'][1]")))
    WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.CSS_SELECTOR, "input#dateSubmit[value='View']"))).click()

    # Wait for the data to Load
    wait.until_not(EC.presence_of_element_located((By.XPATH, '//div[@class="loading-indicator"]')))
    wait.until(EC.visibility_of_element_located((By.CLASS_NAME, 'row')))
    wait.until(EC.presence_of_element_located((By.XPATH, '//table[@class="mat-table cdk-table mat-sort ng-star-inserted"]')))

except TimeoutError:
    raise Exception("Timeout Error: Could not load data from URL")

# Extract the data from the table using BeautifulSoup
soup = BeautifulSoup(browser.page_source, 'html.parser')
table = soup.select_one('table[class="mat-table cdk-table mat-sort ng-star-inserted"]')
rows = table.find_all('tr', {'class': 'mat-row cdk-row ng-star-inserted'})
```

קוד רובי המאפשר טיפול מגוון בשגיאות:



דאגנו כי הקוד של הzahlן יהיה רובי ויעמוד ב"איתני" הרכשה היום יומיים.

קרי, טיפול במגוון שגיאות, כאשר בפועל zahlן ינסה כמספר הפעמים אשר הוגדר לו ל"תקוף" את עמודה היעד, אשר הוגדר כ"בעית".

לבסוף, במידה ונכשל zahlן את אותו עמוד לקובץ CSV המכיל את הרשומות אותן לא הצליחו להרכיש

```
# Handle exceptions during data retrieval, retry if necessary
except Exception as e:
    max_retries = 1 # usually on - 5 tries
    retries = 0
    while retries < max_retries:
        try:
            browser.quit()
            time.sleep(15)
            browser = webdriver.Firefox()
            browser.maximize_window()
            date_data = get_weather_data(year, month, day, city, browser)
            weather_data[current_date.strftime("%Y-%m-%d"),city] = date_data
            break
        except Exception as e:
            retries += 1
            time.sleep(30)
            logging.error("An error occurred while getting data for date %s: %s", current_date, e)
    except SessionNotFoundException:
        browser.quit()
        time.sleep(15)
        browser = webdriver.Firefox()
        browser.maximize_window()
        date_data = get_weather_data(year, month, day, city, browser)
        weather_data[current_date.strftime("%Y-%m-%d"),city] = date_data
    if retries == max_retries:
        current_url = browser.current_url
        browser.quit()
        logging.error("Maximum number of retries reached for date %s", current_date)
        problematic_urls.append(f"{current_date.strftime('%Y-%m-%d')}, {city}", current_url)
        current_date += timedelta(days=1)
        current_date = start_date
```



הדגמה ריצת הצל

The screenshot shows a Visual Studio Code window with the following details:

- Title Bar:** File Edit Selection View Go Run Terminal Help • data_science_project.ipynb - Visual Studio Code
- File Explorer:** Shows one file: data_science_project.ipynb (1).
- Code Editor:** Displays Python code for a function named `get_weather_data_for_dates`. The code uses Selenium to interact with a Firefox browser to fetch weather data for specific dates. It handles exceptions, retries up to 5 times, and logs errors.
- Bottom Status Bar:** PROBLEMS 28, OUTPUT, DEBUG CONSOLE, TERMINAL, JUPYTER, powershell

```
C: > Users > ofekd > OneDrive > Desktop > DSP > data_science_project.ipynb > def get_weather_data_for_dates(start_year, start_month, start_day, end_year, end_month, end_day, browser):
+ Code + Markdown | ▶ Run All ⌘ Clear All Outputs ⌘ Go To ⌘ Restart | ⌘ Variables ⌘ Outline ...
    weather_data[current_date.strftime("%Y-%m-%d")] = date_data
except Exception as e:
    max_retries = 1 # usually on ~ 5 tries
    retries = 0
    while retries < max_retries:
        try:
            browser.quit()
            time.sleep(15)
            browser = webdriver.Firefox()
            browser.maximize_window()
            date_data = get_weather_data4(year, month, day, browser)
            weather_data[current_date.strftime("%Y-%m-%d")] = date_data
            break
        except Exception as e:
            retries += 1
            time.sleep(30)
            logging.error("An error occurred while getting data for date %s: %s", current_date, e)
    except SessionNotFoundException:
        browser.quit()
        time.sleep(15)
        browser = webdriver.Firefox()
        browser.maximize_window()
        date_data = get_weather_data4(year, month, day, browser)
        weather_data[current_date.strftime("%Y-%m-%d")] = date_data
if retries == max_retries:
```

טיפול בזמן ריצה בנתונים - חישוב ממוצעים ושיטוח הנתונים

מכיוון שהנתונים באתר ממנו הרכשנו מדוחים בrama השעיתית, היה על הזחלן לבצע תהליך של "שיטוח" טבלת הנתונים המתקבלת לשורה אחת - המיצגת תאריך מסויים, במקום מסויים, בטבלת הנתונים הסופית שלנו.

על מנת לבצע שיטוח זה,

היה על הזחלן לבצע חישובי ממוצעים ושכיחים על מנת שהשורה המתקבלת תבעה בצורה המיטבית את הנתונים עליהם היא מtabסת.

```
#calculating avg
temperature_avg = fahrenheit_to_celsius(sum(float(t) for t in temperature) / len(temperature))
precipitation_avg = sum(float(p) for p in precipitation) / len(precipitation)
wind_avg = mph_to_kmh(sum(float(w) for w in wind_speed) / len(wind_speed))
humidity_avg = sum(float(r) for r in humidity) / len(humidity)
dew_point_avg = fahrenheit_to_celsius(sum(float(r) for r in dew_point) / len(dew_point))
wind_dir_mf = most_frequent_wind_direction(wind_dir)
condition_mf = most_frequent_string(condition)

data = {'Year': year, 'Month': month, 'Day': day,
        'Temperature': temperature_avg,
        'Temp Dew Point': dew_point_avg,
        'Humidity': humidity_avg,
        'Wind Direction': wind_dir_mf,
        'Wind Speed': wind_avg,
        'Precipitation': precipitation_avg,
        'Condition': condition_mf}
return data
```

ניסוי לשימוש בagișת Multi-threading

מכיוון שהיא ברכינו להרכיש כמהות נתוניים עצימה על מנת לקבל חיזוי מהימן של מגמות האקלים.

ניסינו להבין איך אלו הופכים את התהlixir למהיר יותר. לשם כך פנו לגישה שנקראית THREADING-MULTI, שבבסיסה הוא שימוש במספר אינסטנסים (THREADS) שונים על מנת לקבל את התהlixir ההרbesch.

התדרמו רבות עם גישה זו במהלך הפroyskt, אך טרם הטמענו את המערכת באופן סופי.
(ניתן לראות דוקומנטציה נוספת בדף הפroyskt)

טיפול נתונים

נקי נתונים ועיבוד מקדים הוא שלב מכריע בכל פרויקט נתוח נתונים.

בשלב זה של הפרויקט ניקינו ועיבדנו מראש את נתונים מג האוויר על ידי:

- איחוד קבצי הנתונים
- טיפול ב-OUTLIERS
- הסרת כפליות
- ملي ערכים חסרים
- הרמת סוג נתונים (על מנת להפכם לモכנים לנתח)
- הנדסת מאפיינים (FEATURE ENGINEERING)

לאחר מכן נעשה שימוש נתונים מנוקאים ומעובדים אלו לבניית מודלים והפקת תובנות לגבי דפוסי מג האוויר ומגמות שינוי.



איחוד קבצי הנתונים

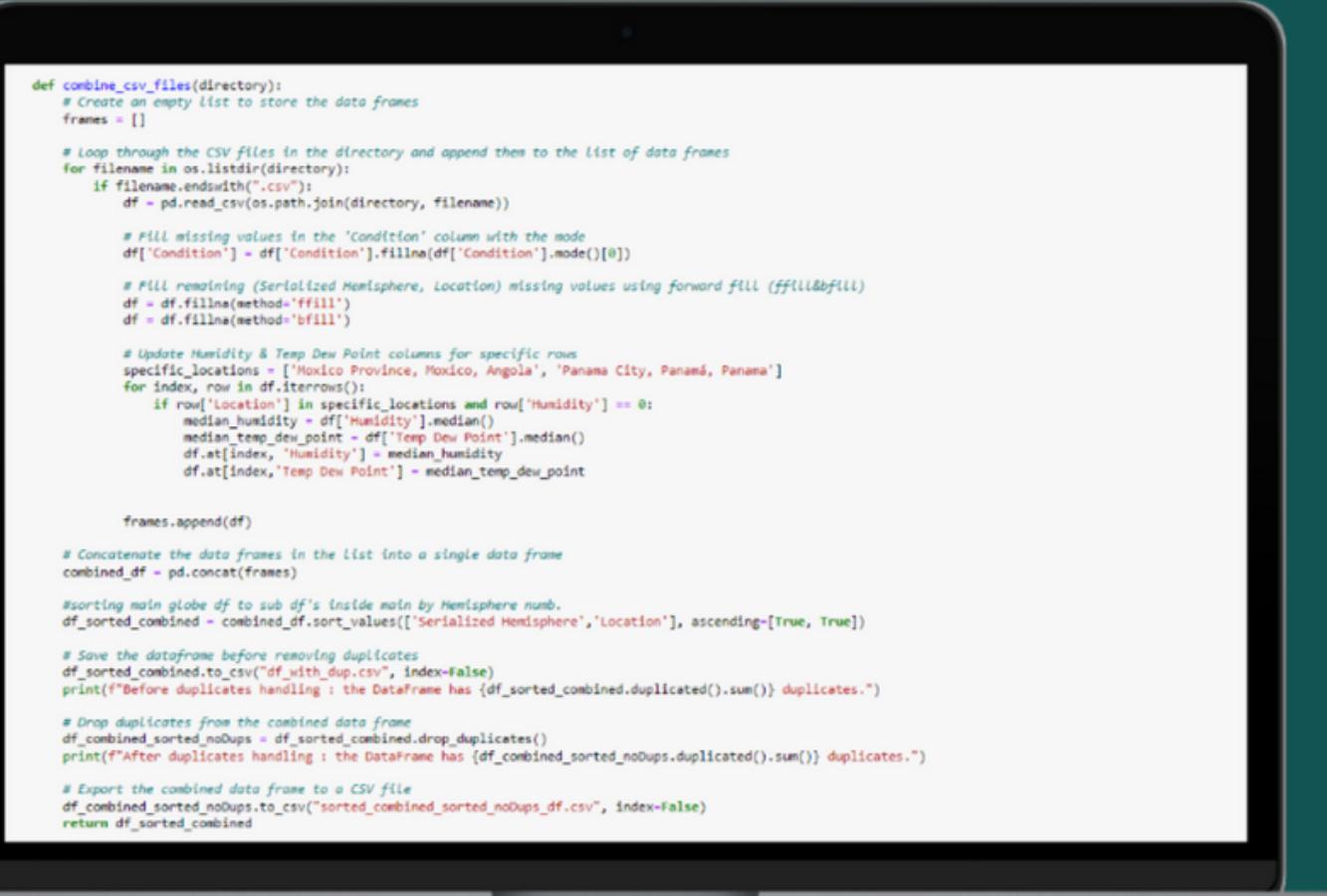
מכיוון שהיה עליינו להרכיש כמהות נתונים נזירים גדולה (כ-32 מדינות על פני 8 שנים בrama היומית)

החליטנו לפרק את התהליך ההרכשה לשלבים בהם נרכיש בכל פעם עיר אחת.

בנוסף לכך כדי לזרח את התהליך מקבלנו את ההרכשה בין כמה מחשבים. כל זאת יצר לנו מספר רב של קבצי csv. שאוותם עליינו היה לאחד בחלק זה של הפרויקט.

יתר על כן, הבנו בשלב מאוחר יותר של התהליך (EDA) כי נדרש לפונקציה הביל לטיפול נוספת נתונים ברמה הלאומית.

כל זאת ועוד בסעיף נתונים חסרים...



```
def combine_csv_files(directory):
    # Create an empty list to store the data frames
    frames = []

    # Loop through the CSV files in the directory and append them to the list of data frames
    for filename in os.listdir(directory):
        if filename.endswith('.csv'):
            df = pd.read_csv(os.path.join(directory, filename))

            # Fill missing values in the 'Condition' column with the mode
            df['Condition'] = df['Condition'].fillna(df['Condition'].mode()[0])

            # Fill remaining (Serialized Hemisphere, location) missing values using forward fill (ffill&bfill)
            df = df.fillna(method='ffill')
            df = df.fillna(method='bfill')

            # Update Humidity & Temp Dew Point columns for specific rows
            specific_locations = ['Mexico Province, Mexico, Angola', 'Panama City, Panamá, Panama']
            for index, row in df.iterrows():
                if row['location'] in specific_locations and row['Humidity'] == 0:
                    median_humidity = df['Humidity'].median()
                    median_temp_dew_point = df['Temp Dew Point'].median()
                    df.at[index, 'Humidity'] = median_humidity
                    df.at[index, 'Temp Dew Point'] = median_temp_dew_point

            frames.append(df)

    # Concatenate the data frames in the list into a single data frame
    combined_df = pd.concat(frames)

    # Sorting main globe df to sub df's inside main by Hemisphere numb.
    df_sorted_combined = combined_df.sort_values(['Serialized Hemisphere','Location'], ascending=[True, True])

    # Save the dataframe before removing duplicates
    df_sorted_combined.to_csv("df_with_dup.csv", index=False)
    print(f"Before duplicates handling : the DataFrame has {df_sorted_combined.duplicated().sum()} duplicates.")

    # Drop duplicates from the combined data frame
    df_combined_sorted_noDups = df_sorted_combined.drop_duplicates()
    print(f"After duplicates handling : the DataFrame has {df_combined_sorted_noDups.duplicated().sum()} duplicates.")

    # Export the combined data frame to a CSV file
    df_combined_sorted_noDups.to_csv("sorted_combined_sorted_noDups_df.csv", index=False)
    return df_sorted_combined
```

טיפול ב-OUTLIERS

בשלב זה לאחר חקר ראשוני של הנתונים ראיינו כי בכמה מאפיינים קיימים ערכים חריגים בהם נטפל.

```
# Select rows where 'wind speed' is greater than 216 Km/H
selected_rows = df[df['Wind Speed'] > 216]

# Iterate over the selected rows and print each one to the console
print
for index, row in selected_rows.iterrows():
    print(row)
    print('-' * 73)

# Select rows where 'Temperature' is smaller than -25 Celcius
selected_rows = df[df['Temperature'] < -25]

# Iterate over the selected rows and print each one to the console
for index, row in selected_rows.iterrows():
    print(row)
    print('-' * 48)
```

בשלב הראשון חקכנו באופן ידני את טבלאות הנתונים שלנו וראיינו כי קיימים כ-2 ערכים חריגים במאפיין SPEED WIND (ערכים העולים על 100 Km/H).

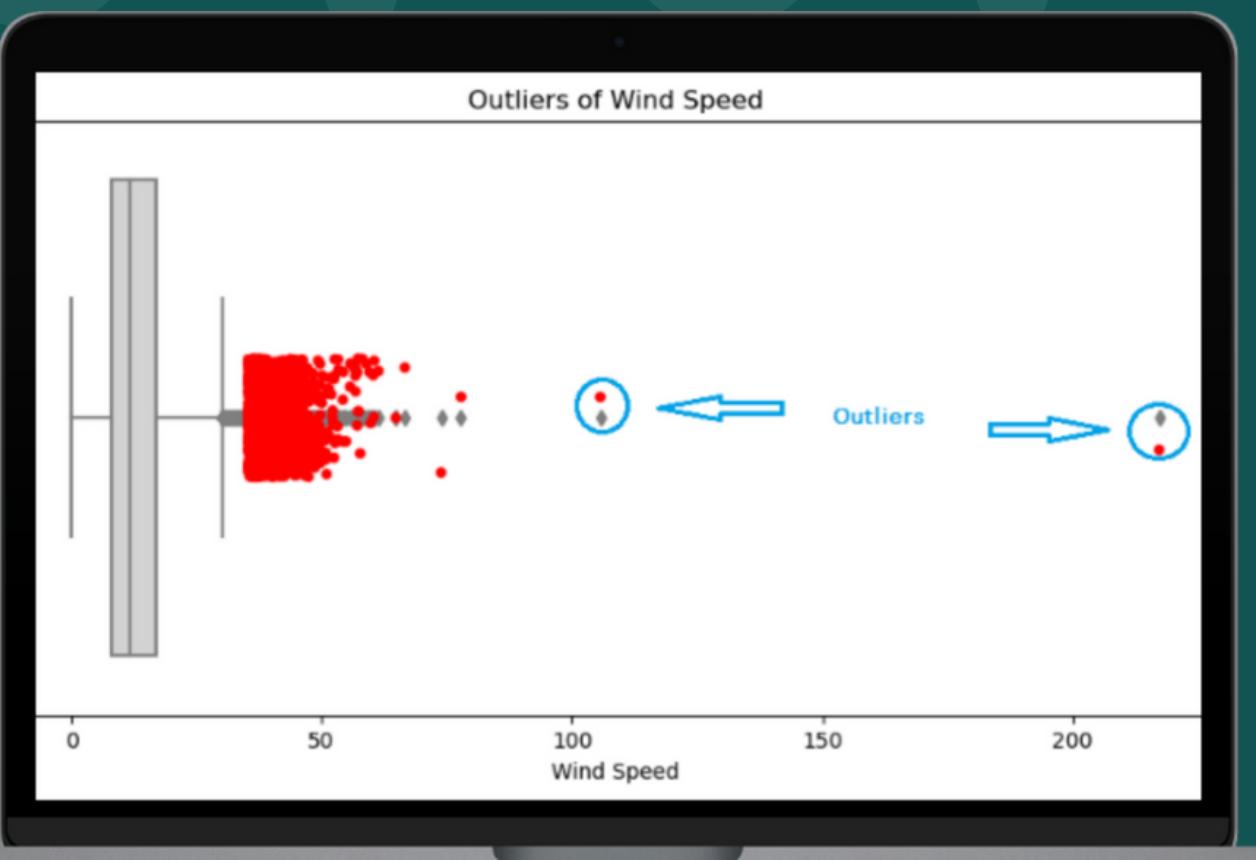
אנחנו צפינו כי מדובר בתיעודים של סופות ולאחר בדיקה מול הנתונים המקוריים באתר נמצא כי אלו נתונים אמיתיים, מכיוון שמדובר רק ב-2 מקרים. לטובת הסדר הטוב הורדנו אותם מהטבלה.

OUTPUT

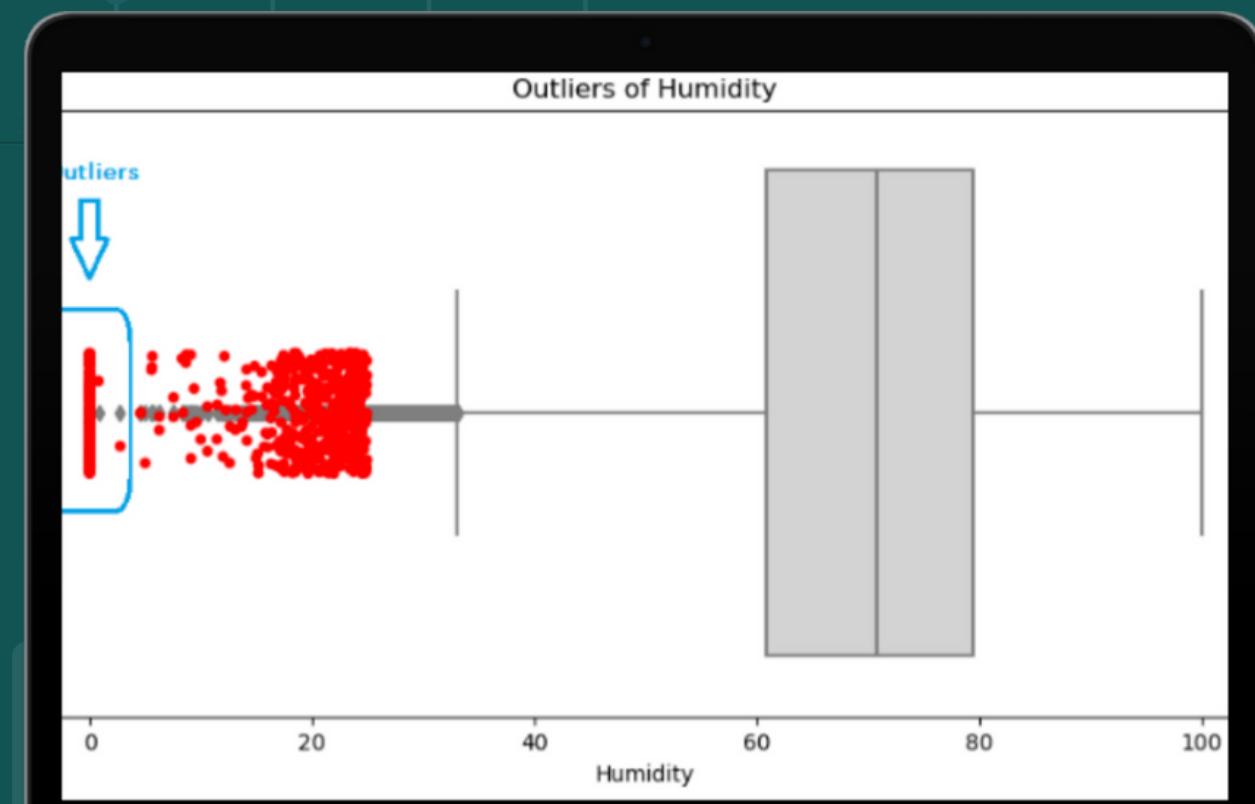
```
Year          2020
Month         10
Day           25
Temperature   31.666667
Temp Dew Point 21.984127
Humidity      56.714286
Wind Direction S
Wind Speed    217.26144
Precipitation 0.0
Condition     Fair
Location      Naypyidaw (Ela), Mandalay Region, Myanmar (VYEL)
Serialized Hemisphere 3.0
Name: 4887, dtype: object
```

```
Year          2016
Month         2
Day           13
Temperature   -25.362319
Temp Dew Point -31.449275
Humidity      57.608696
Wind Direction WNW
Wind Speed    29.947793
Precipitation 0.0
Condition     Drifting Snow
Location      Ottawa, Ontario, Canada
Serialized Hemisphere 1.0
Name: 3371, dtype: object
-----
Year          2023
Month         2
Day           3
Temperature   -25.227273
Temp Dew Point -31.919192
Humidity      53.863636
Wind Direction NW
Wind Speed    33.28416
Precipitation 0.0
Condition     Drifting Snow
Location      Ottawa, Ontario, Canada
Serialized Hemisphere 1.0
Name: 5916, dtype: object
```

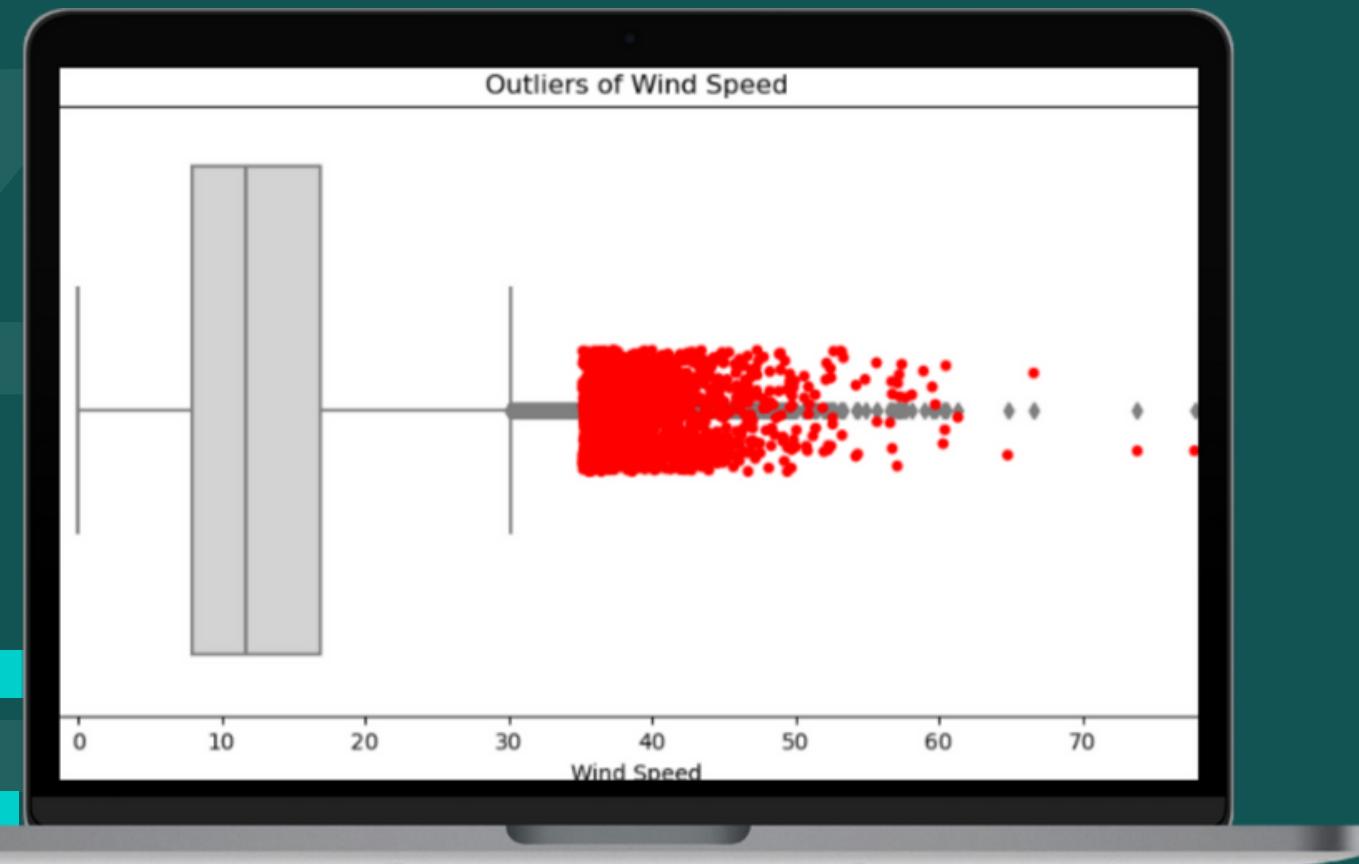
WINDSPEED OUTLIERS



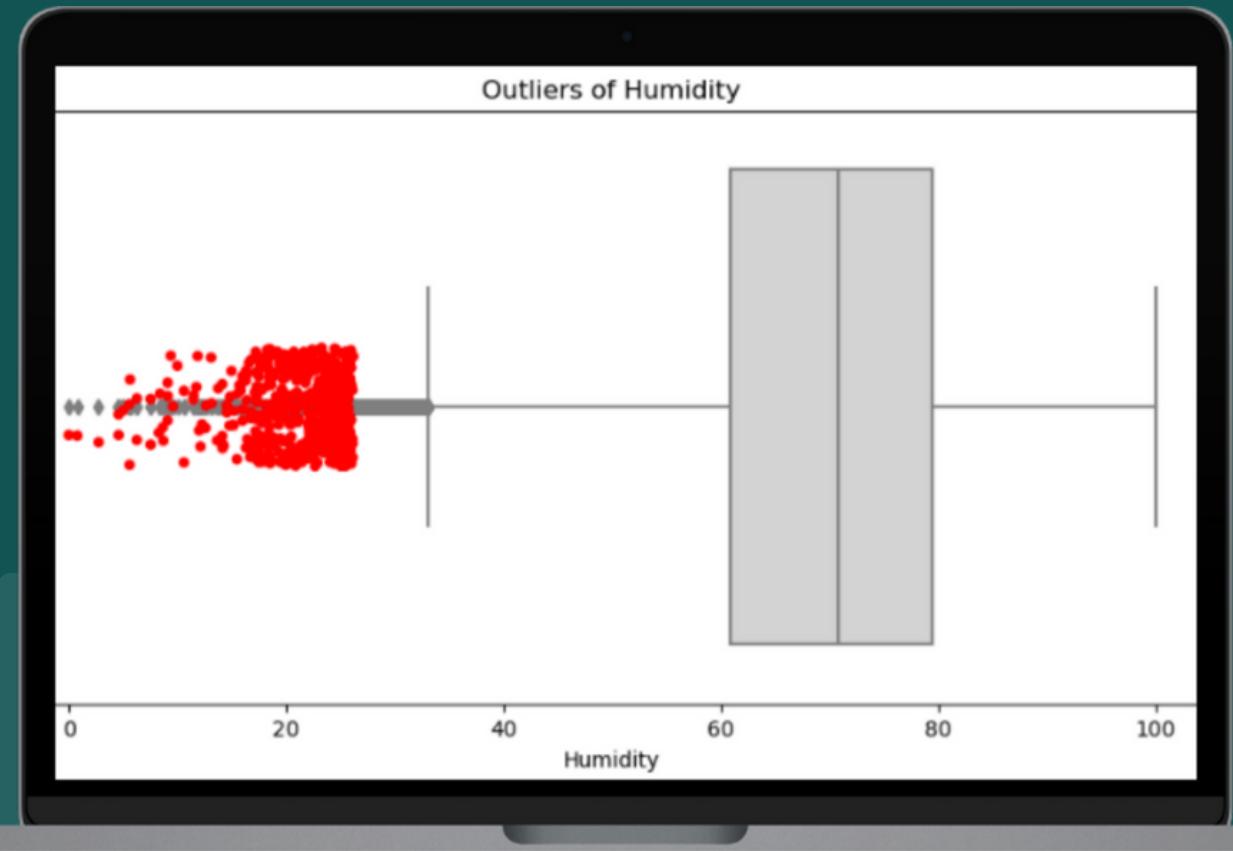
HUMIDITY OUTLIERS



WINDSPEED OUTLIERS AFTER HANDLING



HUMIDITY OUTLIERS AFTER HANDLING



המשר - טיפול בנתונים חריגים

בהתליר בדיקת הנתונים החריגים שלנו התענינו במקרים בהם הלוחות הייתה קיצונית - 100% או לחלופין - 0%.

מצאנו כי בהמון במקרים הנתון לא מופיע בפועל בטבלת הנתונים המקורית, בנוסף לכך בדקונו נתונים אלו, גילינו כי בכל פעם שהלוחות מתעדכנת ל-0% - באופן מפתיע נקודת הטל גם מתעדכנת ל(17.7778-) ללא סיבה שנייתנית לשינוי על ידי הנתונים لكن טיפול במקרה אלו והשכנו אותם למקרים חסרים (ראו טיפול במקרים חסרים).

נסיק כי הבעיה התרחשה בשלב הרכשת הנתונים.

ניתוח מקדים

בשלב זה רצינו לקבל מושג כללי על הנתונים אותם הרכשנו:

Analyze Insights

Dataset Information

The dataset contains weather information from 32 locations between 2015 and 2023. It has 102655 rows and 12 columns. There are no missing values in any of the columns.

Temperature

The temperature in the location ranges from -25.36 to 41.91 degrees Celsius with a mean temperature of 18.43 degrees Celsius.

Dew Point Temperature

The dew point temperature ranges from -31.92 to 30.30 degrees Celsius with a mean of 11.89 degrees Celsius. The Temperature Histogram shows that the temperatures typically range between 15 and 28 Celsius.

Humidity

The humidity ranges from 0 to 100 with a mean of 69.19.

Wind Speed

The wind speed ranges from 0 to 217.26 km/h with a mean of 13.23 km/h.

Data Types

The dataset contains both numerical and categorical data types. There are 6 numerical columns and 3 categorical columns.

Categorical Columns

Column Name	Unique Values	Most Frequently Occurring Value
Wind Direction	18	N
Condition	61	Fair
Location	32	Tokyo, Tokyo Prefecture, Japan

Serialized Hemisphere

The Serialized Hemisphere column is of int64 data type and has only one unique value, indicating that all the data belongs to the northern hemisphere.

Temperature & Humidity Relations -

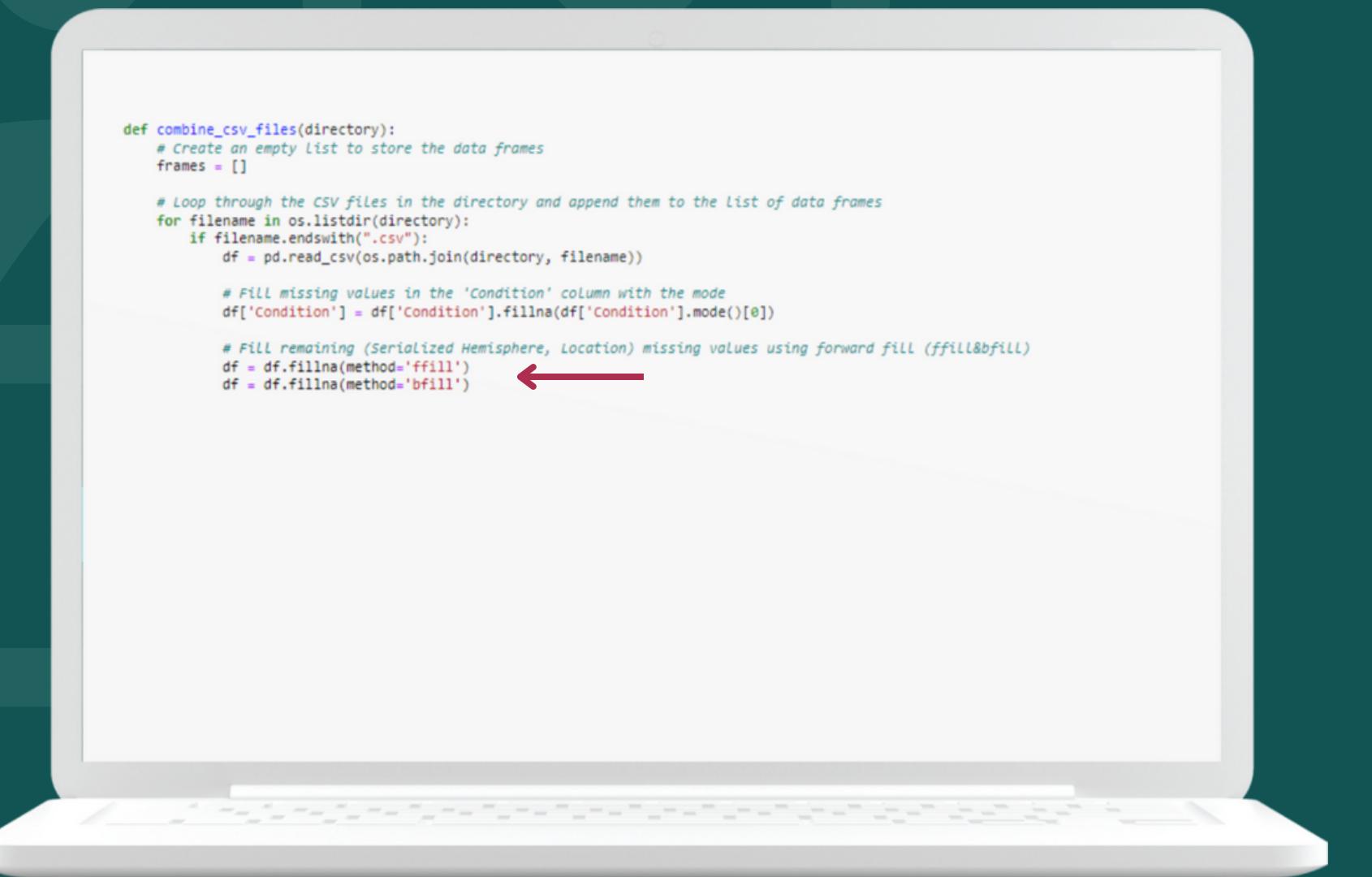
Scatter plot

We can observe that humidity and temperature have a strong direct link up to the 30 degree zone, but beyond that the relationship becomes reversible: as the temperature rises, the humidity falls.

- **שמות ומספר המאפיינים השונים של הנתונים**
- **ערכי חציון וממוצעים של אותן עמודות מאפיינים**
- **ערכים קיצוניים של המאפיינים השונים**
- **מספר הערכים החסרים בטלالة הנתונים שלנו**
- **זיהוי אליזיות בסיסיות בನיסיון לקבל הבנת ראשונית על הנתונים.**

הסרת כפליות

שלב ההסרת הכפליות היה מאד פשוט עבורנו מכיוון שההרכשה בוצעה בצורה מיטבית, במקרים שלנו נמצאו 2 שורות כפולות בלבד, אותן הורדנו



מילוי ערכים חסרים

בשלב זה גילינו כי חסרים נתונים בתונים במאפיין הלחות ונק' הטל. על מנת לטפל בכך בצורה היעילה ביותר היה علينا לחזור לשלב איחוד הקבצים ו בשלב זה - טרם איחוד הקבצים טיפולנו בערכים החסרים.

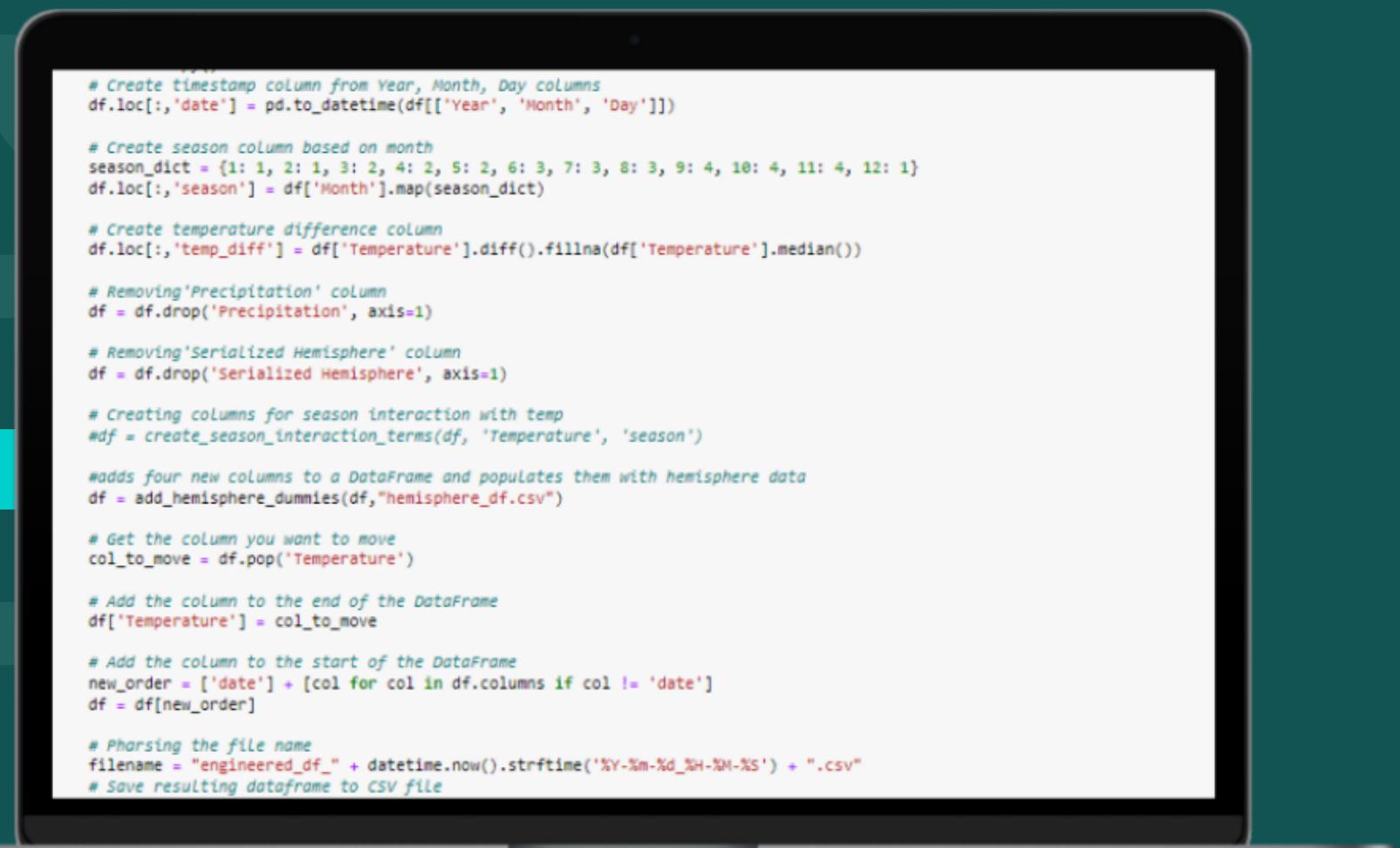
הטיפול נעשה בצורה מכוונת בשלב מוקדם זה על מנת שהערכים של החזון בו נשתמש למילוי הערכים החסרים יוחשבו בטבלת הנתונים בrama הлокאלית ולא בrama הגלובלית - בטבלאת הנתונים המאוחצת.

המרת סוג נתונים

בשלב זהו קיטלנו את המאפיינים לסוגי המשתנים השונים (שם, סדר, מנה וכו')

המרנו את המשתנים המתאימים לכך למשתנים מסוג CATEGORICAL זהן (נומינליים ואורדיינליים). בנוסף תחיל

זה חסר בזיכרון שבירת הנתונים.

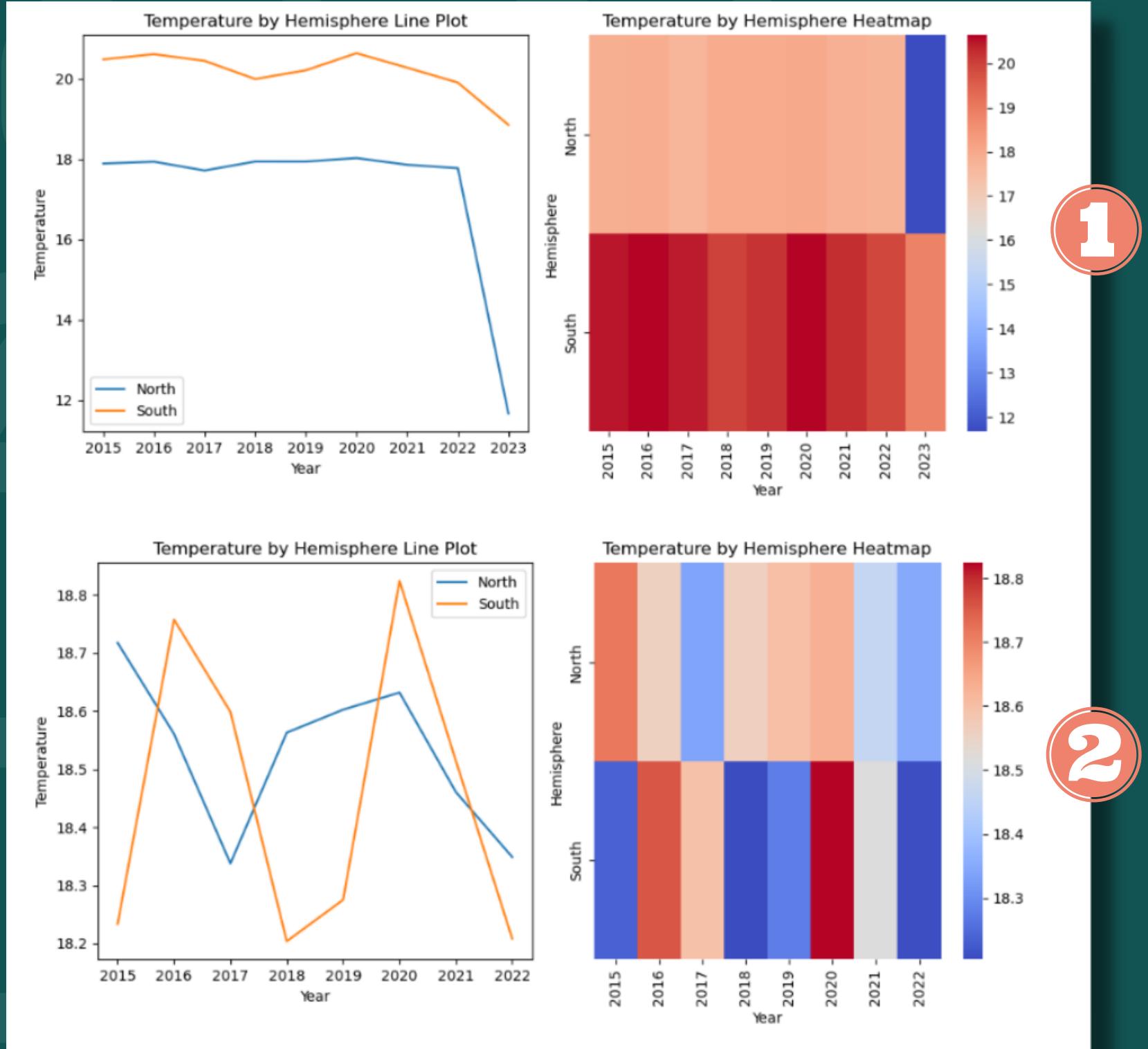


הנדסת מאפיינים

בטור התחלה לאחר ניתוח הנתונים החלקי מצאנו לנכון לתקן ולהוסיף מאפיין מסוג ימוספרה המשير את המדינה להימוספרות אליהן היא שייכת.

בנוספּנו מעתנה מסוג SEASON המייצג את עונות השנה (קיז, חורף סתיו וכו') על מנת שלמידת המכונה שלנו أولית תוכל להפיק תובנות חדשות.

תיקון הטויה טבלת הנתונים



במהלך חקר הנתונים הראשוני שלנו, נוכחנו לגלות (ראו צמד גרפים 1) כי קיימת ירידה בטמפרטורה בשנת 2023 אשר גורמת לסתירות הנתונים שלנו. כמובן, משקפת כי מגמת האקלים בעולם הינה בתהליכי ירידה, דבר אותו אנו יודעים כשגוי.

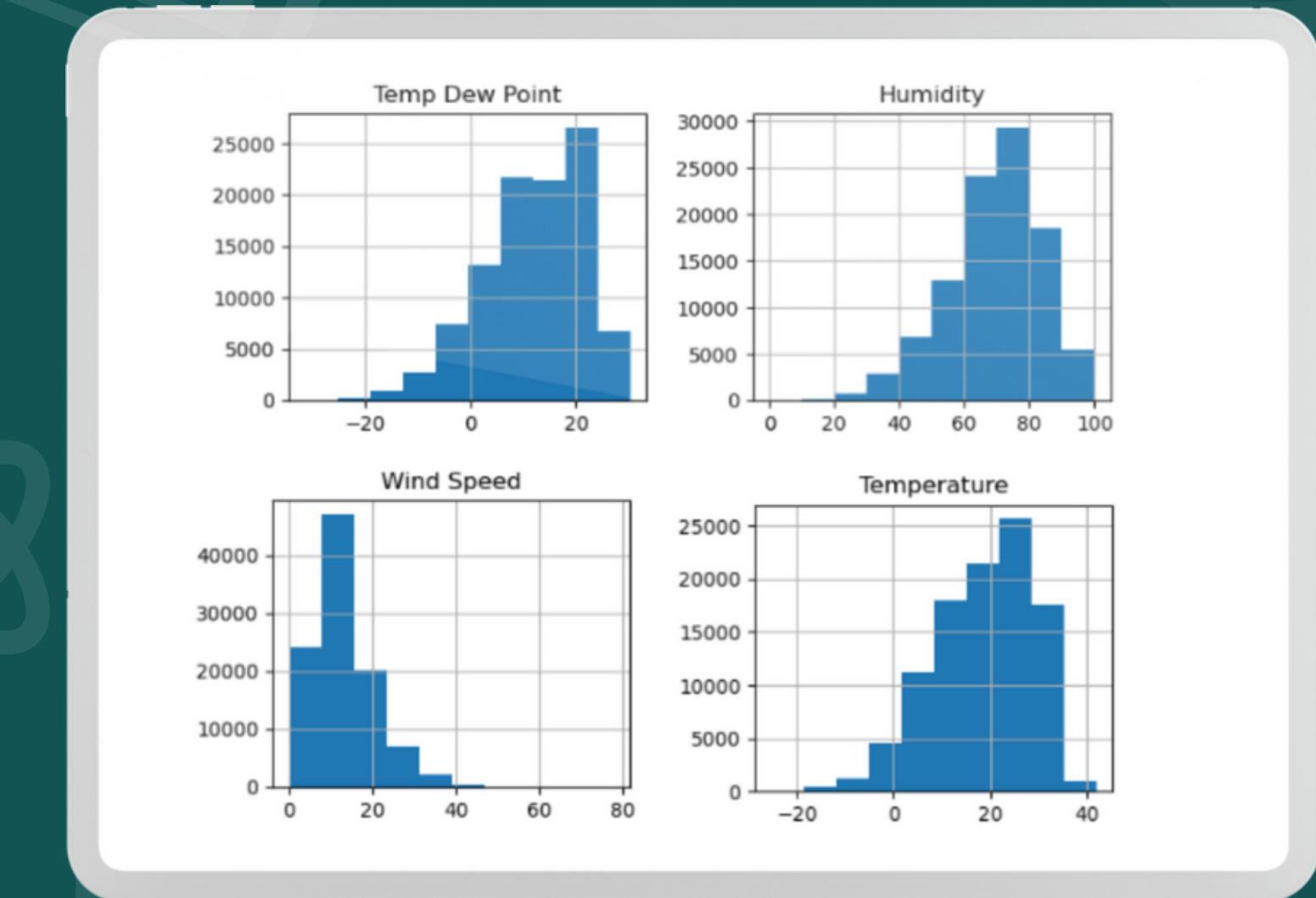
אחרי חשיבה הגענו כי הסיבה לכך היא כי בשלב הרכשה (נכון לתאריך 15.02.23) יכלנו בפועל להרכיש רק את החודשים הראשונים של השנה (חודשים המאופיינים כחודשי חורף) ברור לנו עכשו מדוע קיימת סטיית נתונים זו, נתקנה על ידי מחיקת נתונים שנת 2023.

(ナイト) נתוני מתוך (_EDA - Chiragi)

בפרק זה, ננתח מערך נתונים באמצעות טכניקות סטטיסטיות והדמיה שונות כדי להשיג הבנה טובה יותר של הנתונים, תובנות והבנת הדפוסים הבסיסיים בנתונים.

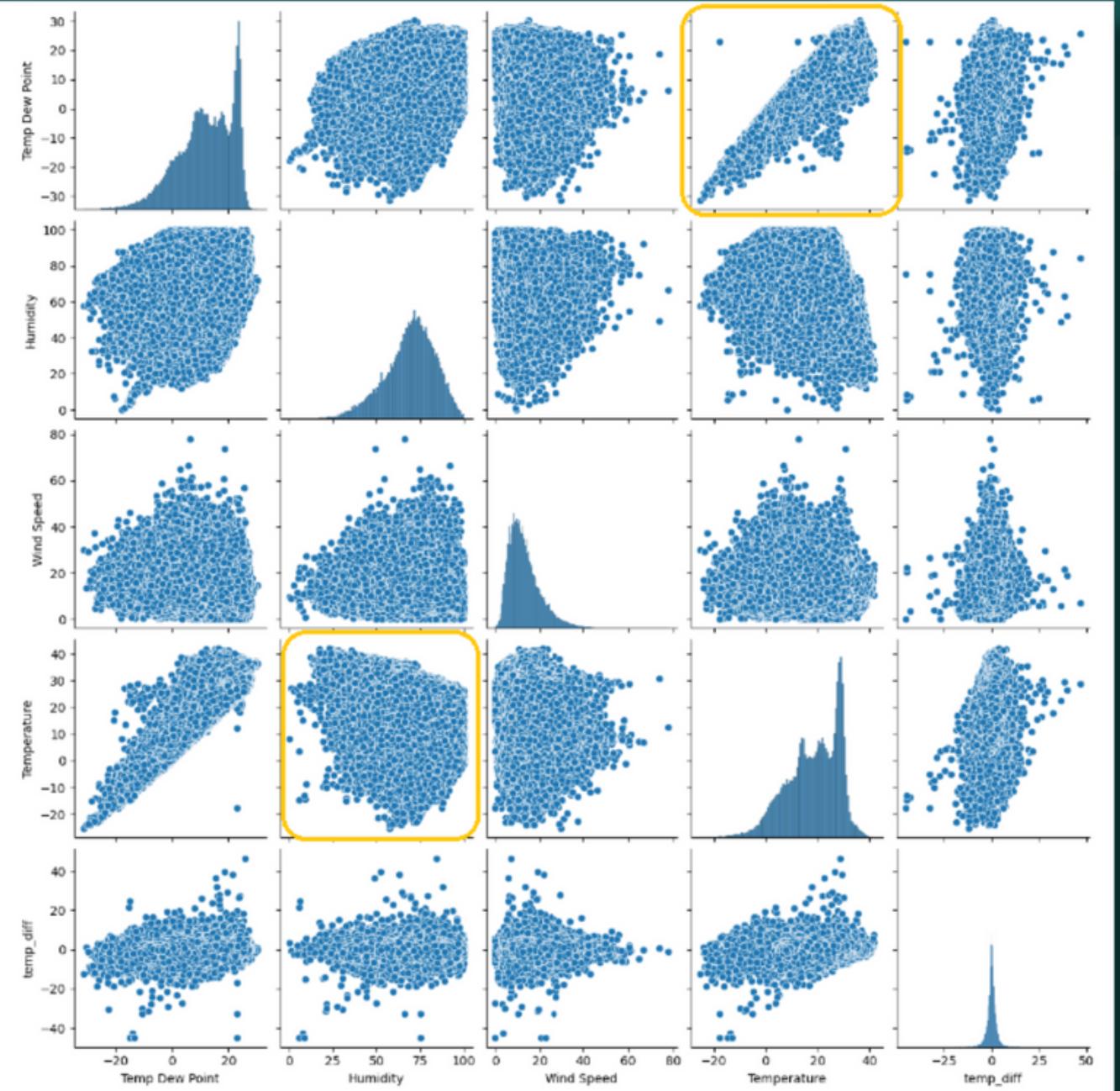
בגרפים אלו כפי שניתן לראות, התפלגות של הלחות, הטמפרטורה וטמפרטורת נקודת הטל היא נורמלית (בצורת פעמון), עם מגמה ימינה (חיובית).

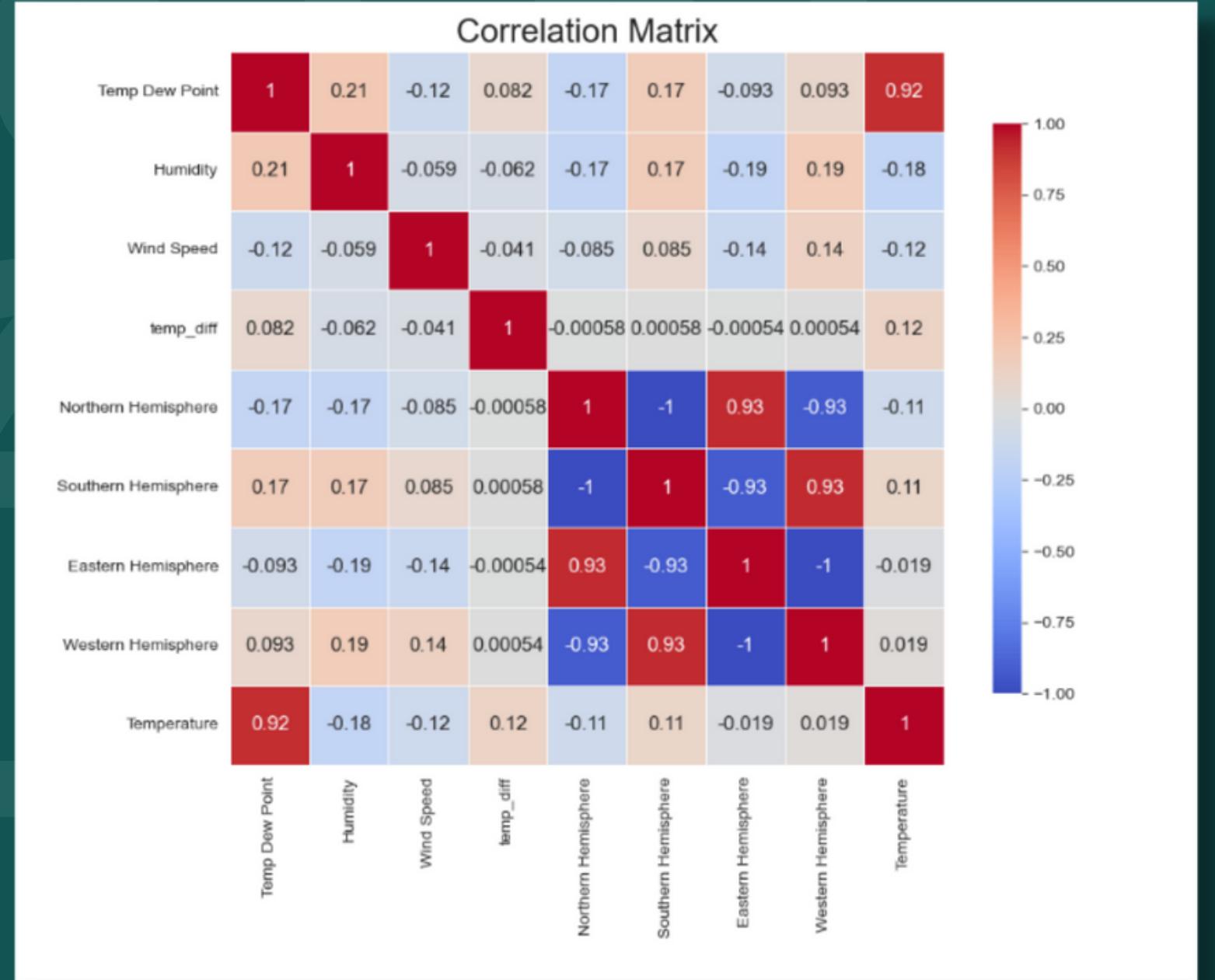
התפלגות של מהירות הרוח היא גם נורמלית, אולם היא מוטה שמאלה.



כאן אנו יכולים לראות את הצורך שלנו להפgin את ה-"SKILLS PLOTTING" שלנו, אך חשוב מכך, להשיג תובנות לגבי הקשרים והמגמות בין המשתנים המספריים הרציפים בכלל, ובין המשתנים הללו לבין עמודת המידע שלנו 'טמפרטורה' בפרט.

על ידי הדמיית הנתונים באמצעות גרפים ותרשימים שונים, אנו יכולים לאשר את ההנחהות שלנו, לחושף דפוסים ולחקרו אפקטים חדשים לניטוח נוסף.

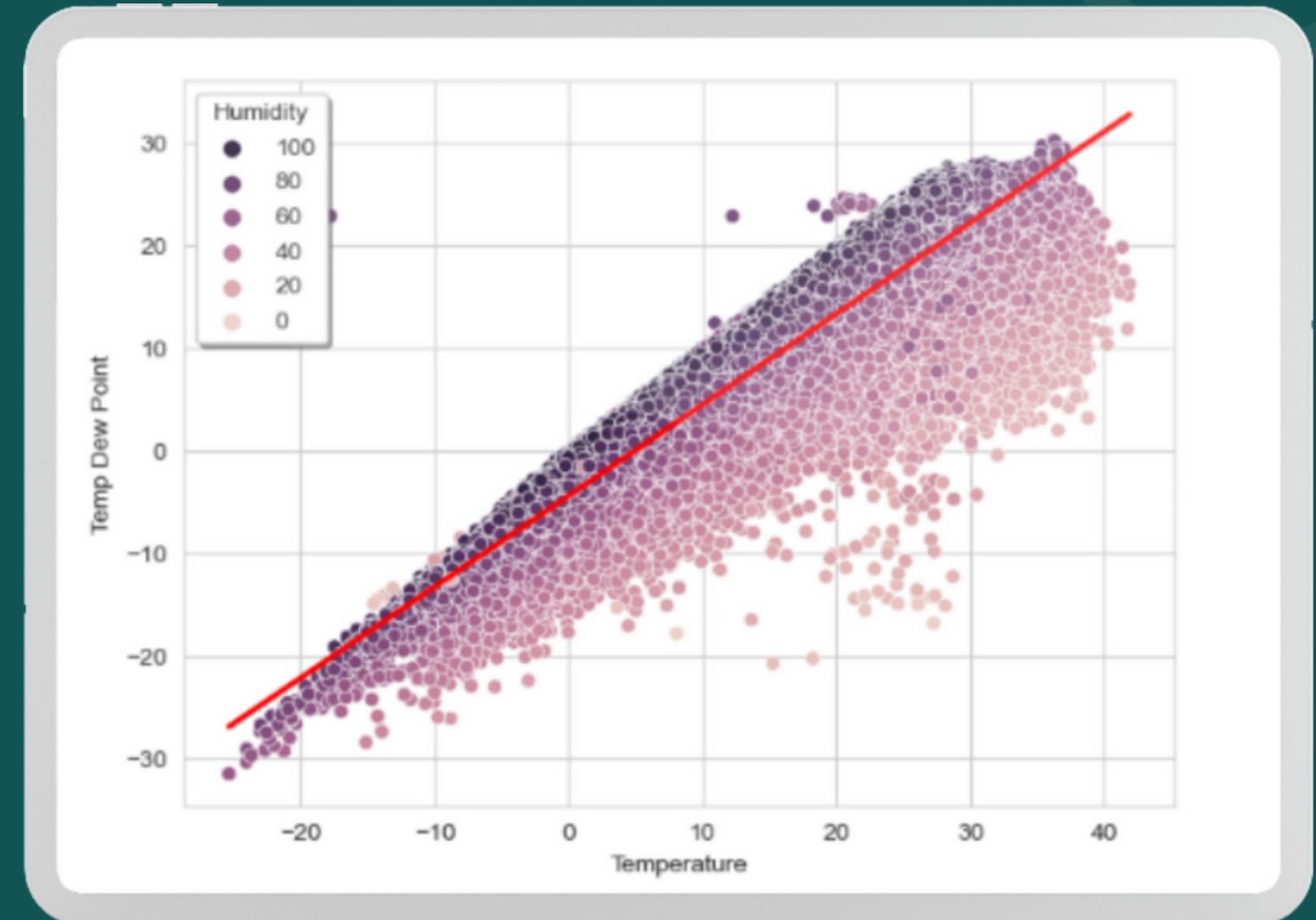


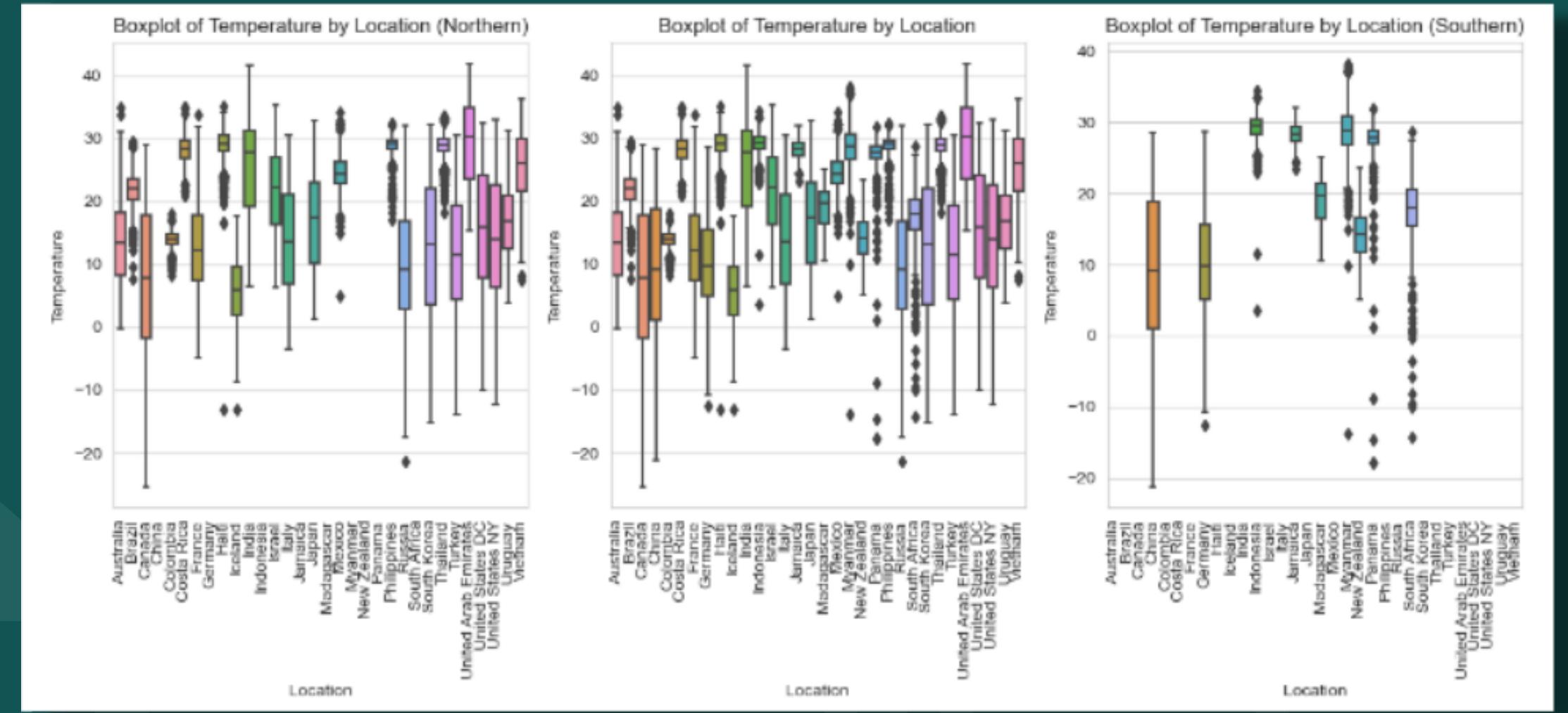


אנו מ Chapman בטבלת החום קשורים "חזקים" בין תכונות לטמפרטורה או לחילופין אינטראקציות "חזקות" בין תכונות שהקשר הישיר שלתן לטמפרטורה חלש יותר על מנת לבנות קשר חדש שיכל לשזוף קשר חזק יותר למשתנה המוסף שלנו - טמפרטורה.

כמו שניתן לראותו יצרנו קו המשקיף את משווהota
הישר המקבוב לערכי הטמפרטורה כפונקציה של נקודת
הטל ולהחות.

בגרף זה נוכל לראות את הקשר הישיר שקיים בין
טמפרטורת נקודת הטל, הלחות היחסית
וכМОВН הטמפרטורה.

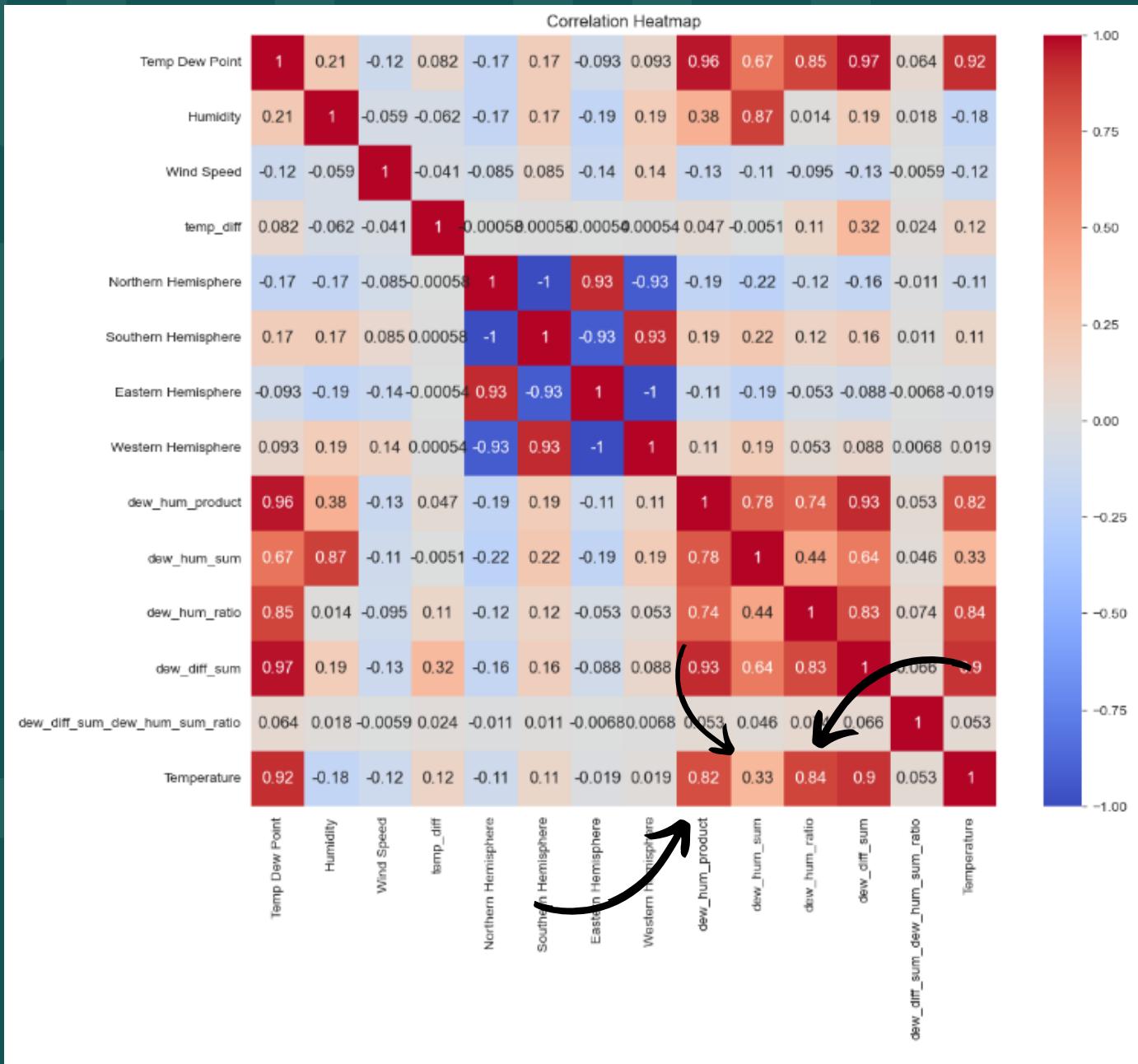




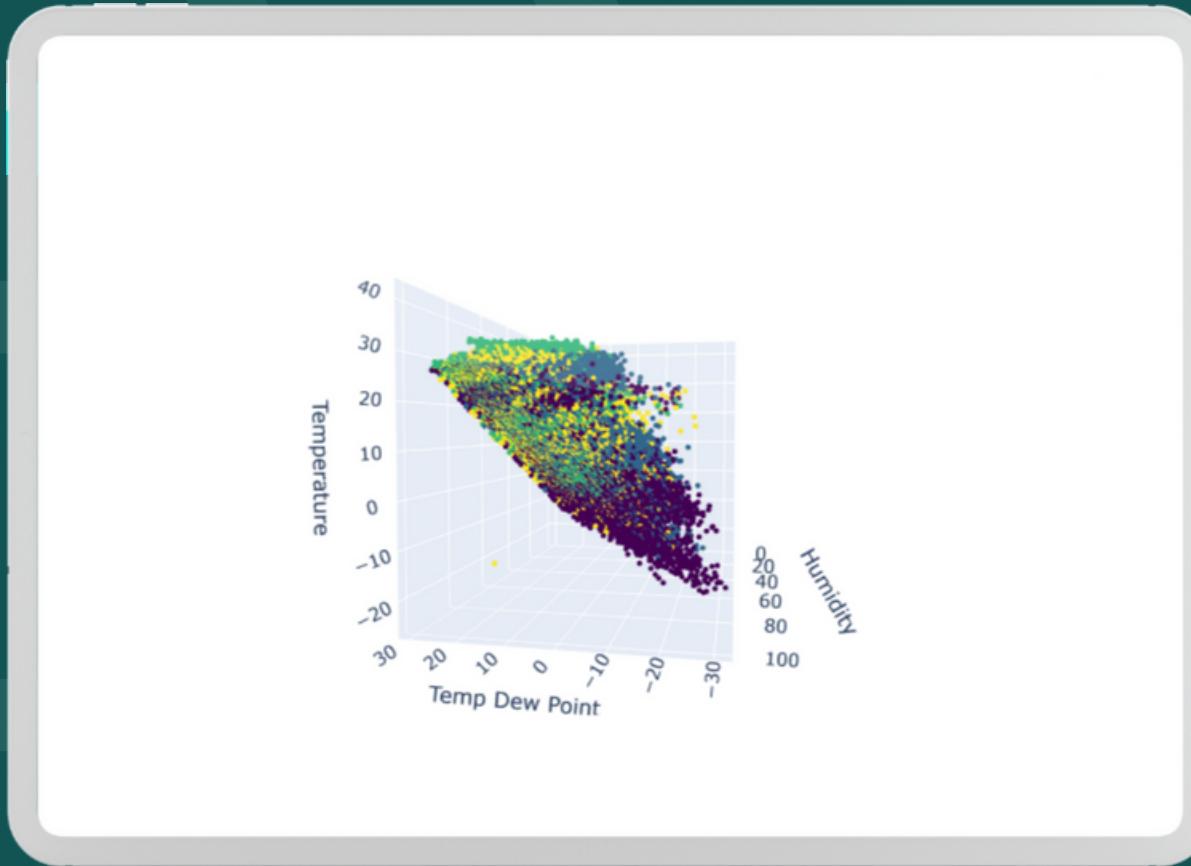
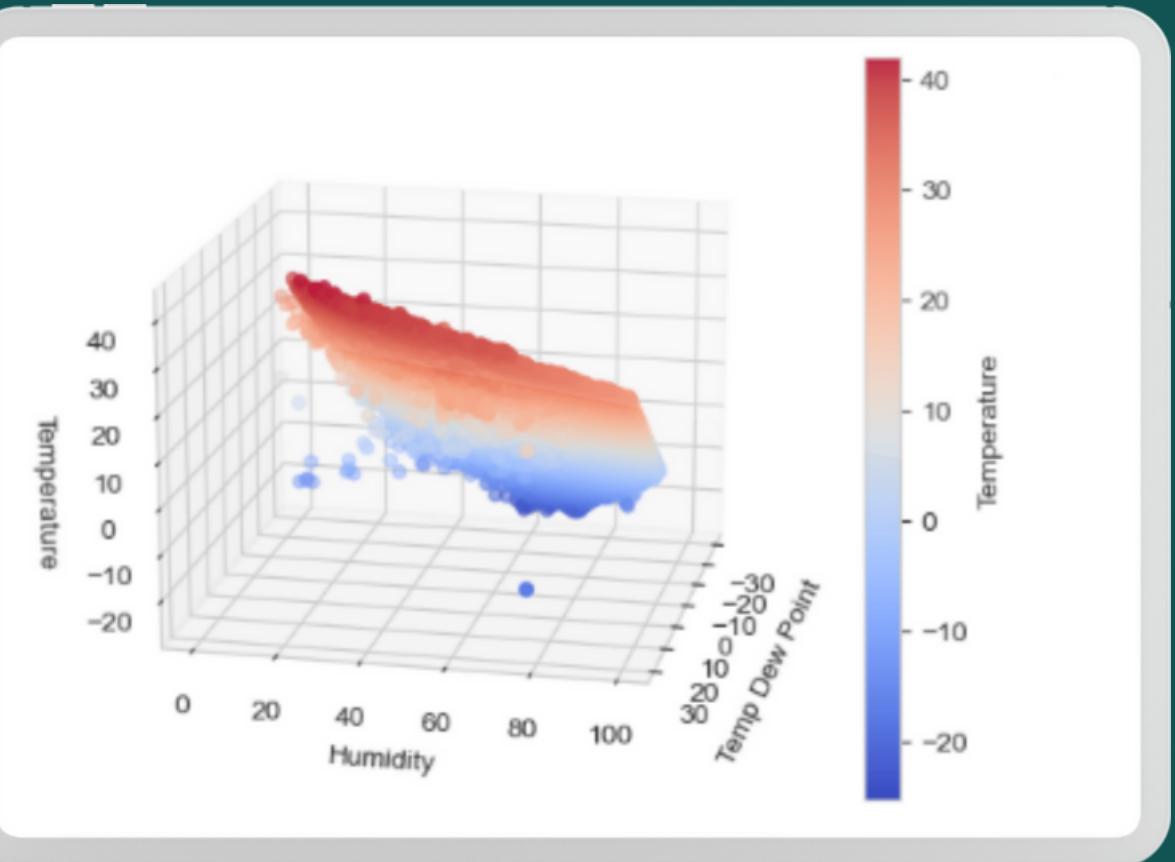
- בונוס לקובץ האומות לפי האם הן בחצי היבשת הצפוני או הדרומי, ניתן לראות כיצד הטמפרטורה משתנה לפי המדיניות השונות ומיקום הגיאוגרפי וכי הטמפרטורות הממוצעות גדולות יותר בחצי היבשת הדרומי.

*בגרף זה גם שמו לב כי הנתונים שלנו היו מוטים ברמה הגלובלית במידה מסוימת לכיוון ההימוספרה הצפונית, זאת מכיוון שהרכשנו יותר מדינות בהימוספה זאת

בנינו טבלה קורלציות מחודשת לאחר ניסיון נוסף שלנו
ל-FEATURE ENGINEERING בין מאפיין הלחות
למאפיין נקודת הטל.



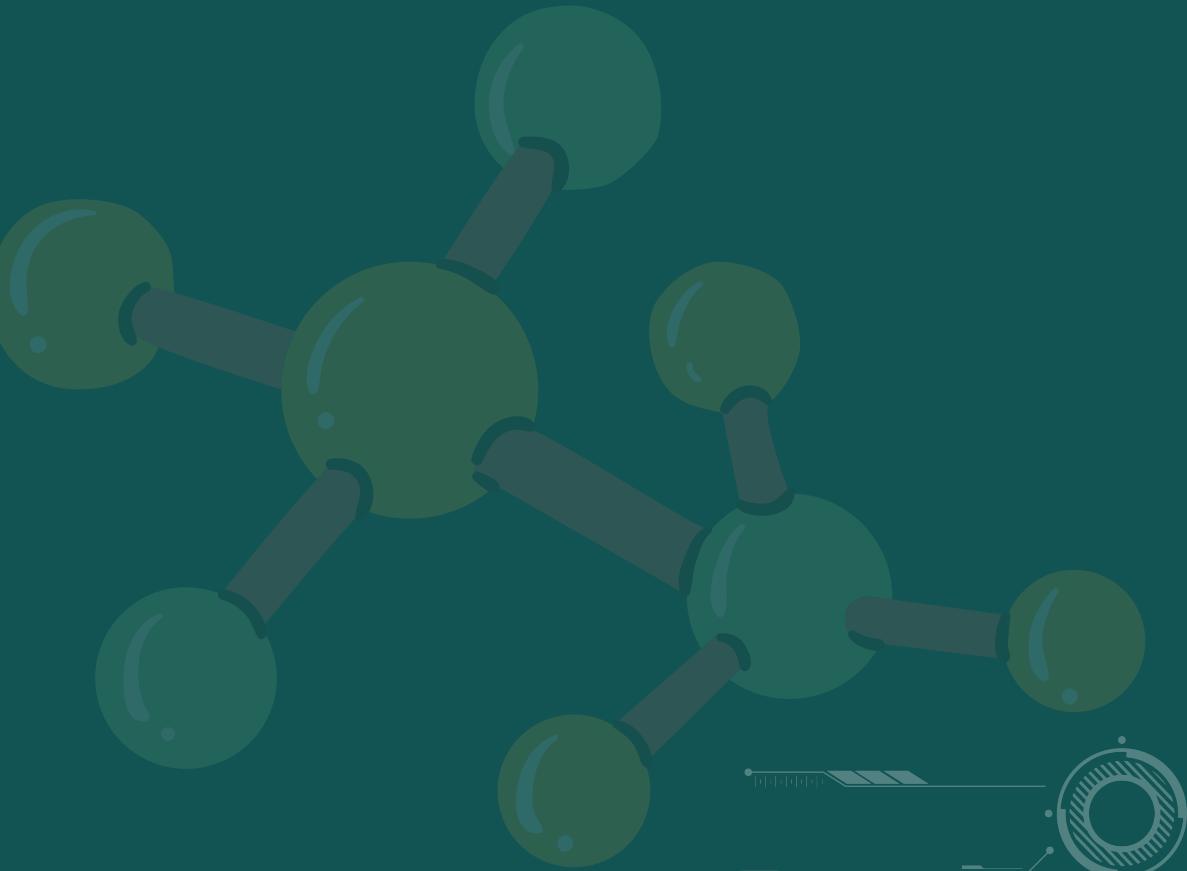
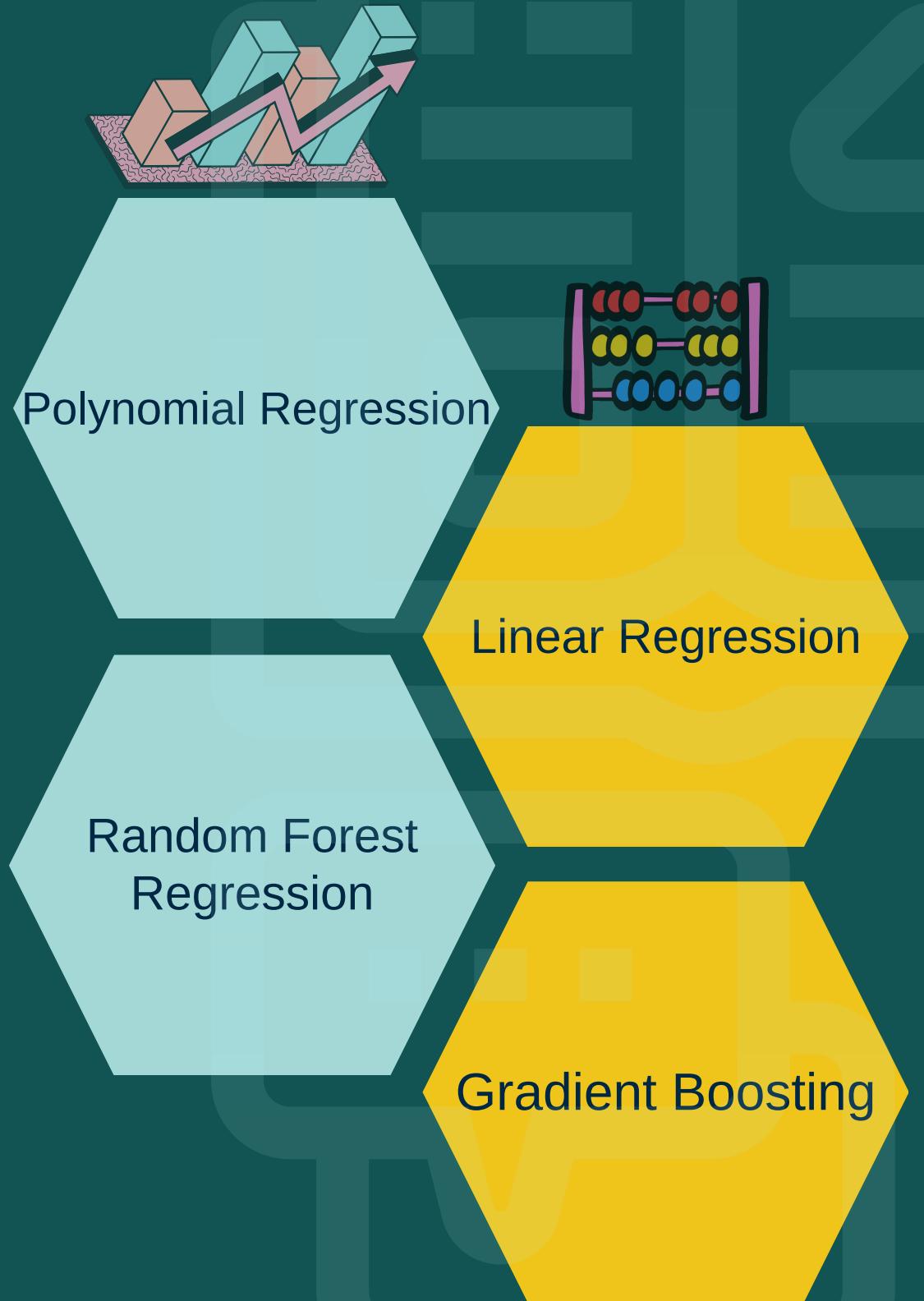
ראינו בגרפים קודמים כי קיים קשר ישיר בין משתנה
לחות למשתנה המוסבר שלנו, טמפרטורה. אם כי
נרצה לחזק על ידי "הינדוס" מאפיין הלחות שלנו עם
מאפיין נקודת הטל.



גרפים תלת ממדיים

אנו יכולים לראות בבירור כיצד התכונות הללו קשורות זו לזו: טמפרטורת נקודת טל, לחות וטמפרטורה

למידה מכונה



למידת מכונה

בנית מודלים והערכתה הם צעד מכריע בכל פרויקט מדעי נתוניים.

בפרק זה, נחקור מודלים שונים של למידת מכונה ונעריך את הביצועים שלהם באמצעות מדדים שונים כגון: דיוון, R₂ ועוד.

כמו כן, נבצע כוונון של הפרמטרים כדי ליעל את הביצועים של הדגם הנבחר.

לאחר שבחרנו את המודל הטוב ביותר, נעריך את הביצועים שלו על ערכת אימוחות ונודע שהוא מقلיל היתוב לנתחים חדשים.

כמו כן, נבחן את יכולת הפרשנות והסביר של המודל.

פונקציות הערכת מודלי למדידת המוכנה השונות

פונקציה זו לוקחת ארבעה כניסה:
מטריצות תכונות: אימון ובדיקה (X_{train} ו- X_{test}),
ומשתני יעד: אימון ובדיקה (y_{train} ו- y_{test}).

לאחר מכן הוא מתאים

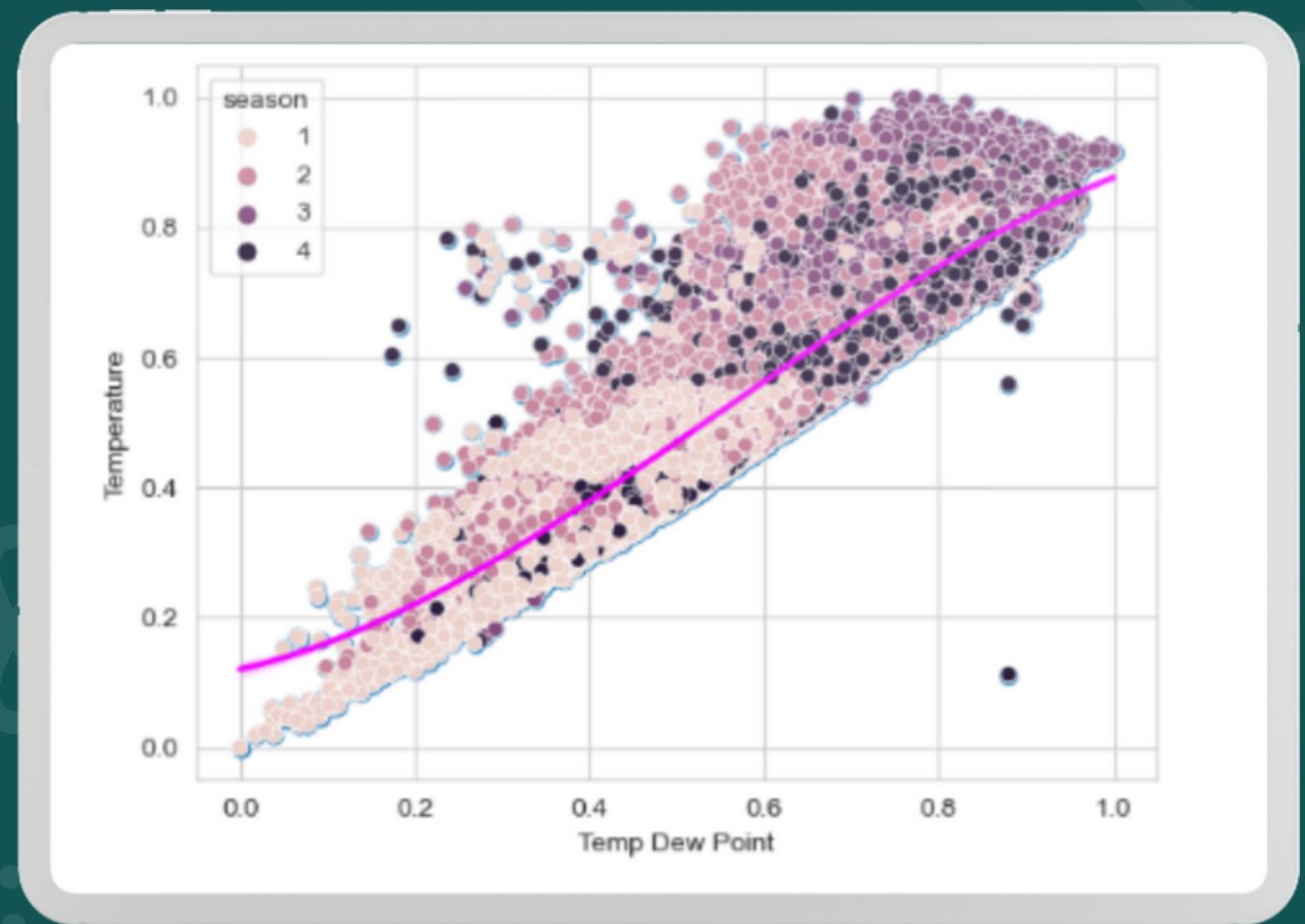
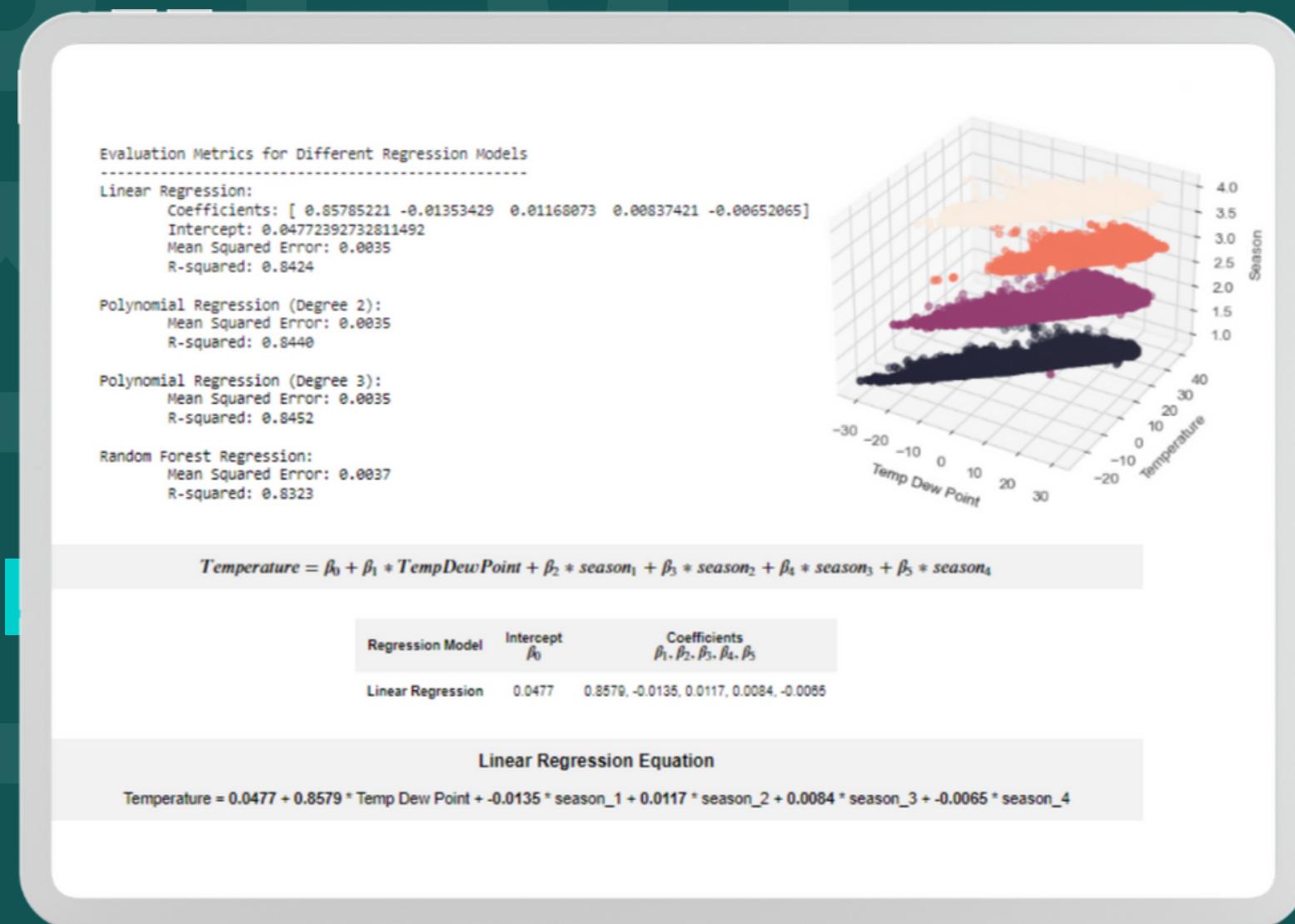
- מודל רגסיה ליניארי,
- שני מודלים של רגסיה פולינומית (עם מעלות 2 ו-3)
- מודל RANDOM FOREST REGRESSION

לנתוני האימון.

הfonקציה משתמשת בשגיאה ממוצעת בריבוע ובריבוע R כדי להעריך את הביצועים של כל מודל בנתוני הבדיקה,
ומדפיסה את מדדי ההערכה עבור כל מודל.

```
In [68]: def evaluate_regression_models(X_train, X_test, y_train, y_test):  
    # train the linear regression model  
    lr = LinearRegression()  
    lr.fit(X_train, y_train)  
  
    # predict temperature on the testing set  
    y_Pred = lr.predict(X_test)  
  
    # evaluate the model using mean squared error and R-squared  
    mse = mean_squared_error(y_test, y_pred)  
    r2 = r2_score(y_test, y_pred)  
  
    # try polynomial regression with degrees 2 and 3  
    poly2 = PolynomialFeatures(degree=2)  
    X_train_poly2 = poly2.fit_transform(X_train)  
    X_test_poly2 = poly2.transform(X_test)  
    lr_poly2 = LinearRegression()  
    lr_poly2.fit(X_train_poly2, y_train)  
    y_Pred_poly2 = lr_poly2.predict(X_test_poly2)  
    mse_poly2 = mean_squared_error(y_test, y_Pred_poly2)  
    r2_poly2 = r2_score(y_test, y_Pred_poly2)  
  
    # try polynomial regression with degree 3  
    poly3 = PolynomialFeatures(degree=3)  
    X_train_poly3 = poly3.fit_transform(X_train)  
    X_test_poly3 = poly3.transform(X_test)  
    lr_poly3 = LinearRegression()  
    lr_poly3.fit(X_train_poly3, y_train)  
    y_Pred_poly3 = lr_poly3.predict(X_test_poly3)  
    mse_poly3 = mean_squared_error(y_test, y_Pred_poly3)  
    r2_poly3 = r2_score(y_test, y_Pred_poly3)  
  
    # try random forest regression  
    rf = RandomForestRegressor(n_estimators=100, random_state=42)  
    rf.fit(X_train, y_train)  
    y_Pred_rf = rf.predict(X_test)  
    mse_rf = mean_squared_error(y_test, y_Pred_rf)  
    r2_rf = r2_score(y_test, y_Pred_rf)  
  
    # print evaluation metrics for all models  
    print("Evaluation Metrics for Different Regression Models")  
    print("-----")  
    # Linear Regression Metrics  
    y_Pred_linear_model = lr.predict(X_test)  
    mse_linear_model = mean_squared_error(y_test, y_Pred_linear_model)  
    r2_linear_model = r2_score(y_test, y_Pred_linear_model)  
    print("Linear Regression:")  
    print(f"Coeficients: {lr.coef_}")  
    print(f"Intercept: {lr.intercept_}")  
    print(f"Mean Squared Error: {mse_linear_model:.4f}")  
    print(f"R-squared: {r2_linear_model:.4f}")  
    print("Polynomial Regression (Degree 2):")  
    print(f"Mean Squared Error: {mse_poly2:.4f}")  
    print(f"R-squared: {r2_poly2:.4f}")  
    print("Polynomial Regression (Degree 3):")  
    print(f"Mean Squared Error: {mse_poly3:.4f}")  
    print(f"R-squared: {r2_poly3:.4f}")  
    print("Random Forest Regression:")  
    print(f"Mean Squared Error: {mse_rf:.4f}")  
    print(f"R-squared: {r2_rf:.4f}")
```

OUTPUT 1



כأن ניתן לראות טמפרטורה כפונקציה של עונה ונק' הטל. בחיתוך נתונים זה קיבל הרגסירה הלינארית את הסיבירות לחיזוי הגובה ביותר.

בגרף לעיל ניתן לראות את הקירוב הפולינומי של המעלה השלישי שמודל הרגסיה מייצר בפועל בתחום

OUTPUT 2

כעת בואו ננסה להשתמש בתכונות המהנדסות שלנו כדי לקבוע אם נוכל ללקט מידע שימושי כלשהו.
לאחר חקירת נתונים, בחרנו להמשיך עם 'dew_point' 'dew_hum_product' 'dew_hum_sum' בו מוקדם

```
Evaluation Metrics for Different Regression Models
-----
Linear Regression:
Coefficients: [ 1.20892779 -0.0095077 -0.58582845]
Intercept: 0.20995834717035528
Mean Squared Error: 0.0003
R-squared: 0.9882

Polynomial Regression (Degree 2):
Mean Squared Error: 0.0001
R-squared: 0.9946

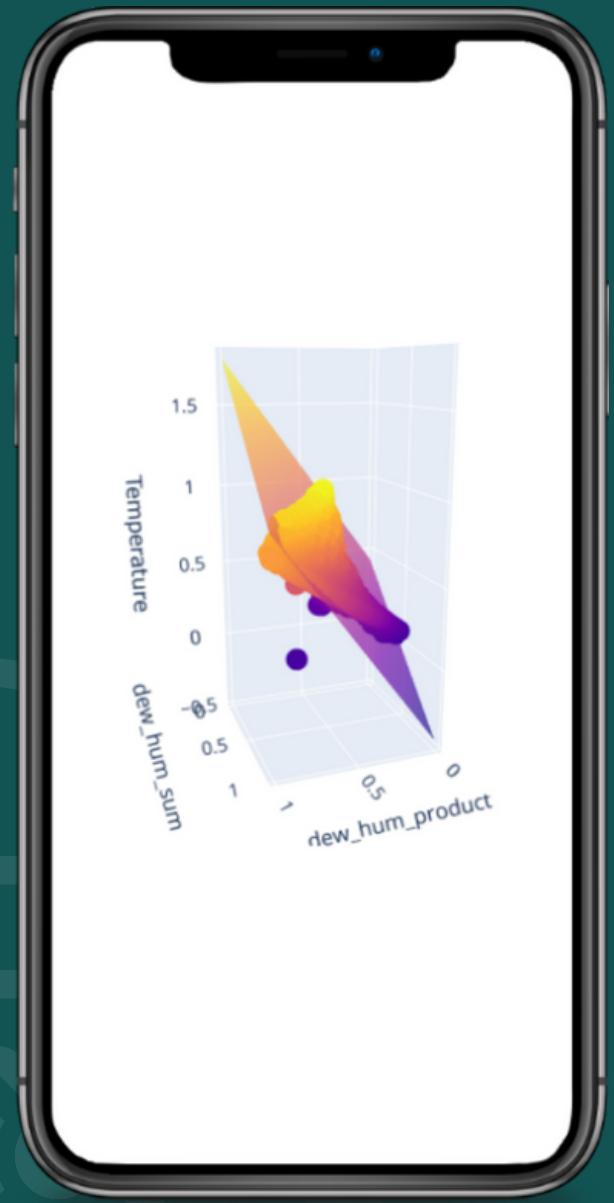
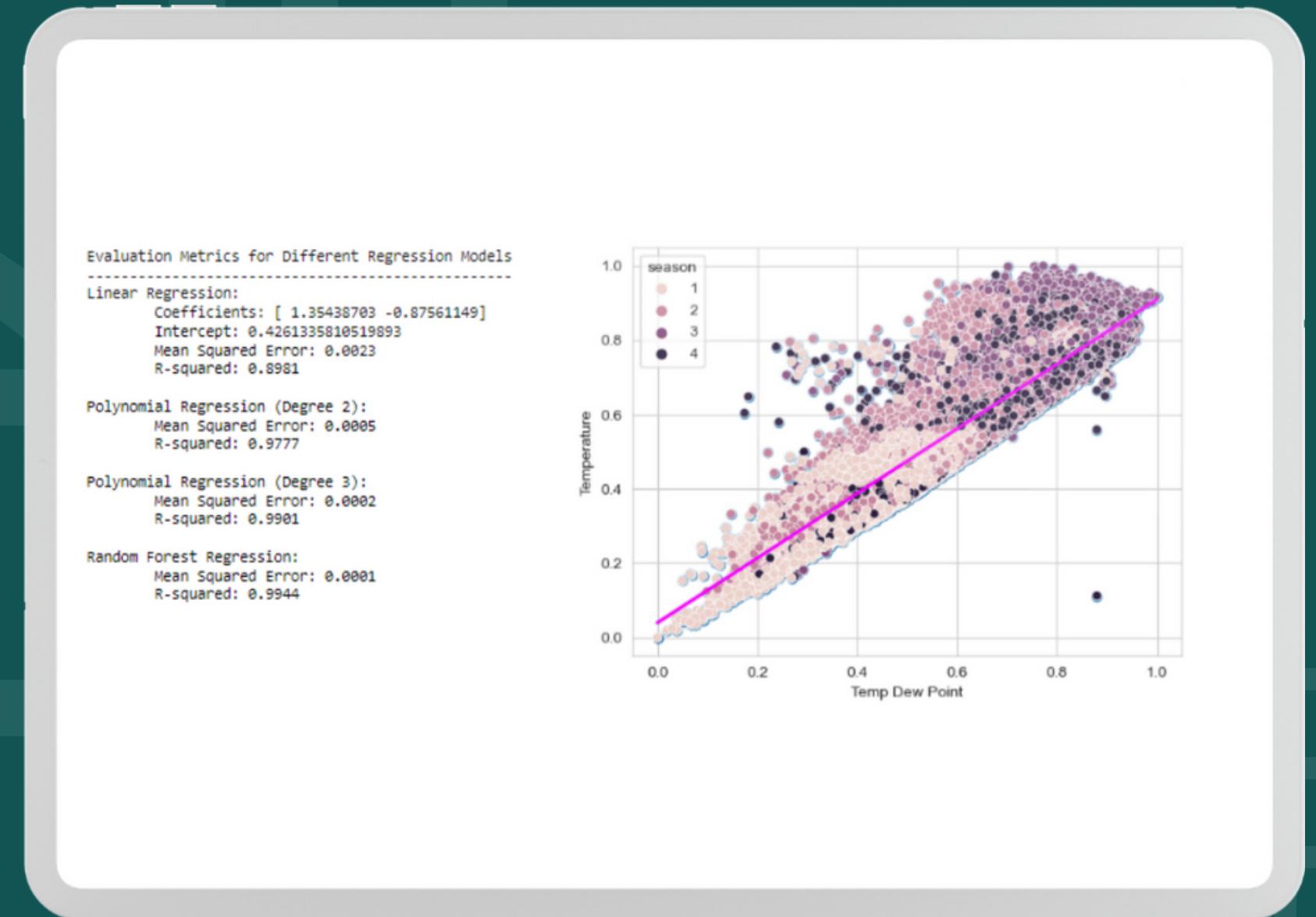
Polynomial Regression (Degree 3):
Mean Squared Error: 0.0001
R-squared: 0.9948

Random Forest Regression:
Mean Squared Error: 0.0001
R-squared: 0.9944
```

בוא ננסה להסביר את התכונה טמפרטורה, נק' הטל, שיש לה קורלציה ממש חזקה עם משתנה הטמפרטורה, ה"משתנה המוסף"
יש לנו כאן דוגמה קלאסית של Overfitting.
שלנו, כדי לראות מה קורה.

FINAL OUTPUT

3



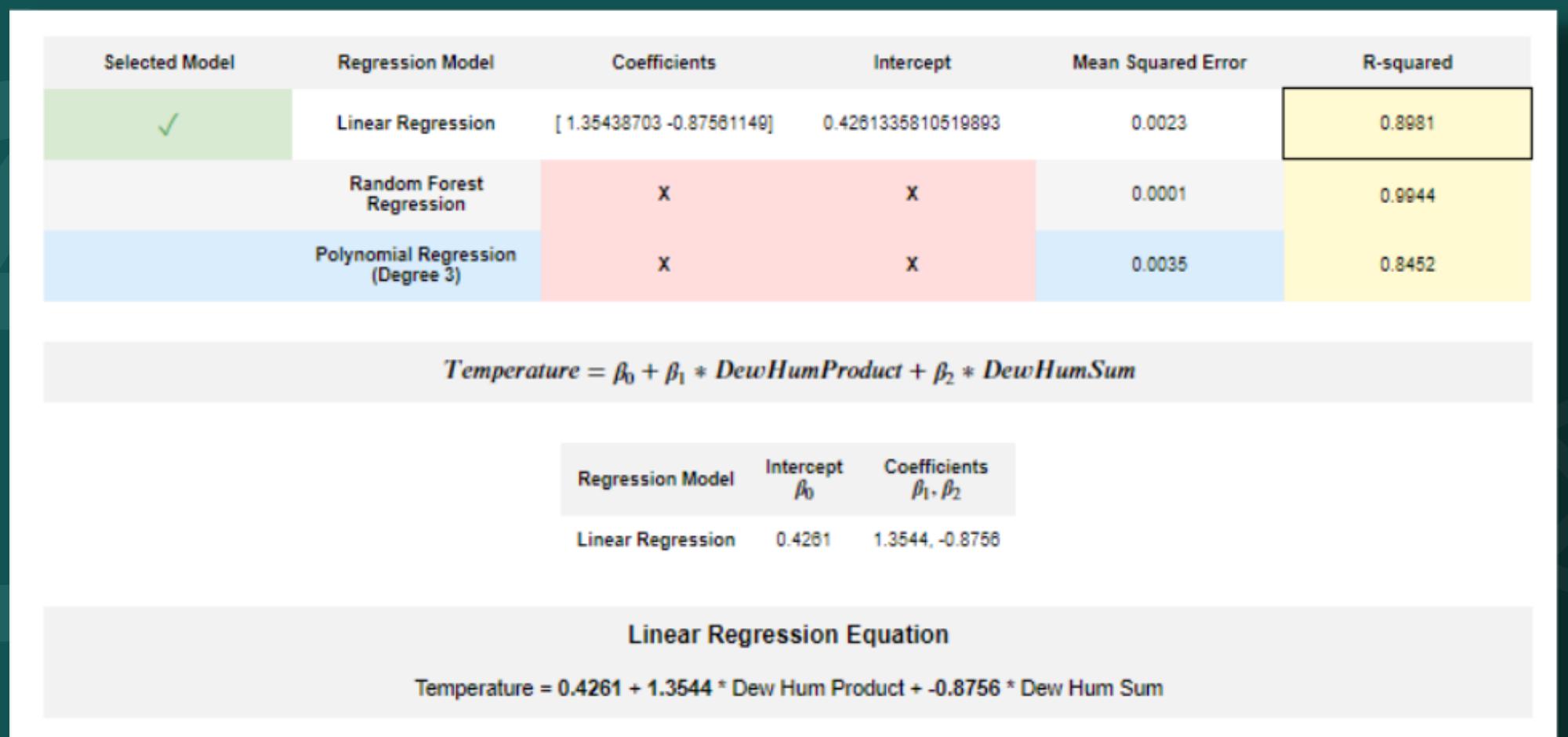
הגרף המצורף מציג את הקירוב הליניארי למודל הרגסיה באמצעות מייצר עבור התחזית

סיכום ומסקנות

לסיכום, נדון בבחירה הסופית של המודל בעל הביצועים הטובים ביותר לחיזוי טמפרטורת עתידיות.

אנו נערך את ביצועי המודל ונווודא שהוא חזק ומסוגל ליצור תחזיות מדויקות.

לאחר שבחרנו את הדגם הסופי, נפרוס אותו ליצור ונשלב אותו עם מערכות אחרות כדי לספק תחזיות טמפרטורה בזמן אמת.



סיכום

לאחר ניתוח יסודי ותהליך בחירת מודל, הגיענו למודל רגסיה ליניארית עם ציון R בריבוע של 0.8981, המעיד על התאמה טובת לנתחונים.

משמעות הרגסיה חופפת שהטמפרטורה מושפעת לטובה מ-PRODUCT HUM DEW ומושפעת לרעה מ-SUM HUM DEW.

השגיאה הממוצעת בריבוע של 0.0023 מאמתת עוד יותר את ביצועי המודל. עם זאת, ישנו מספר תחומים למחקר ופיתוח נוספים.

ראשית, שילוב של יותר משתנים או תכונות עשוי לשפר את דיוק המודל.

שניית, בחינת מודלים מורכבים יותר של מידת מכונה והשווות הביצועים שלהם למודל הרגסיה הליניארית עשוי להיות כדאית.

בנוסף, חקירת ההשפעה של טכניקות שונות של עיבוד מוקדם של נתונים וטכניקות הנדסת תכונות עשויו לשפר את ביצועי המודל.

השלבים הבאים

למרות שמודל הרגרסיה הלייניארית סיפק רמת דיוק מספקת בחיזוי טמפרטורות עתידיות, עדין ישנו אחרים למחקר ופיתוח נוספים.

ראשית, אנו יכולים לחקור מודלים אחרים של למידת מכונה כגון יער אקראיו- GRADIENT BOOSTING כדי לקבוע אם הם יכולים לספק דיוק טוב יותר בחיזוי טמפרטורות עתידיות. שנית, אנו יכולים לשאול איסוף נתונים ותכונות נוספות כדי לשפר עוד יותר את הדיוק של המודלים.

לבסוף, אנו יכולים גם לחקור מודלים של למידה عمוקה כמו רשתות ניירוניים, שיש להם פוטנציאל לספק דיוק גבוהה יותר בחיזוי טמפרטורות עתידיות. בסיכום, הבנתו שלנו סיפק תובנות לגבי הגורמים המשפיעים על הטמפרטורה ופיתח מודל חיזוי טמפרטורהאמין.

עם זאת, ניתן לבצע שיפורים נוספים, והמשרחקירה וחידוד של המודל יכולים להניב דיוק ואמינות טובים עוד יותר. ממצאים אלה יכולים לספק בסיס למחקר עתידי בחיזוי טמפרטורה ותחומי מחקר קשורים.

סיגו
תודה על ההק莎ה