SI 206 Final Project Report

Team Name: Data DynamoSquad
Team Members:
- ➔ William Wentrack (wwentrac@umich.edu)
- ➔ Han Byun (habyun@umich.edu)
- ➔ Erin Lee (erinclee@umich.edu)

Project Goals:

The goal of our project is to analyze the relationship between weather conditions and Michigan State University (MSU) football games. Specifically, we wanted to analyze how the weather such as temperature, precipitation, and wind speed during game time affects game attendance and the amount of points scored by MSU in the MSU games. To do this, we planned to use two main data sources. The first was the Weather API (weatherstack.com), which provided us with detailed weather data including temperature, precipitation, and wind speed during game times. We also used Sports API (collegefootballdata.com), which gave us information about points scored in MSU football games.

We want to mention how our project scope evolved significantly from its initial plan. Originally, we planned to analyze how weather conditions, specifically precipitation, affected ticket prices and expected field goal points at MSU. The initial plan involved using Ticketmaster's API for ticket pricing data and specialized field goal statistics. However, after our group discovered certain technical limitations that are mentioned later on, we pivoted our project goals to analyze broader patterns of game attendance and overall scoring. This provided us with more comprehensive and accessible data sets.

Achieved Goals:

We successfully brought together three different sources of data to create a complete picture of MSU football games. From the Visual Crossing Weather API (using key YG3BWARZ7M7W7TADTU38X6J53), we were able to gather detailed weather information for East Lansing from 2005-2023. This included not just basic temperature readings, but also precipitation levels and wind conditions for every game day. The College Football Data API (key: u3N5YeQ4CFO09Gf4n8/V8QCHouOPUB1w9Ml6Y3CAGmfJIwgrAPMhn36F/dbH7C+L) provided us with comprehensive game statistics, which showed us how many points MSU scored in each home game. To complete our dataset, we developed a web scraping system to collect attendance figures from Wikipedia, covering all MSU home games from 2009 to 2023. This combination of data sources gave us a thorough view of how weather, attendance, and team performance interact.

Problems Faced:

Our biggest challenge came from our original plan to use the Ticketmaster API. When we discovered it worked better with JavaScript than Python, we needed to find a different approach. Rather than seeing this as a setback, we turned to Wikipedia for attendance data. We think that this was a change that ended up helping us a lot as Wikipedia provided more historical data than we could have gotten from Ticketmaster, giving us attendance records for over a decade of games.
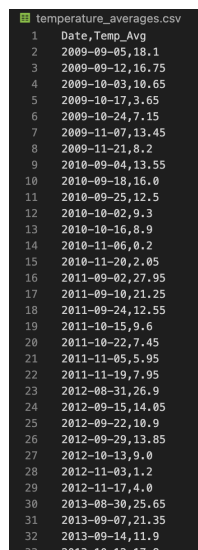
We also faced a learning curve with web scraping which was a technique we hadn't originally planned to use. Once we understood how to implement it effectively using 'BeautifulSoup' in Python, we were able to gather attendance data much more efficiently rather than just manually collecting it. As we worked with the data, we realized that looking at overall points scored instead of just field goals gave us a better picture of how the weather might affect game outcomes.

Calculations from Data in the Database:

Our analysis revealed several interesting patterns in the data. After looking at temperatures, we found games were played in very diverse weather conditions. The warmest game was played at 27.95°C (September 2, 2011), while the coldest dropped to -5.55°C (November 19, 2022). After doing calculations, we found that the average game temperature was 11.5°C. This showed us that MSU football spans both warm early season games and cold late season games.

We found that the attendance numbers told a very interesting story as out of 103 analyzed games, 87 games ended up having between 70,000-79,999 fans. This high attendance stayed consistent regardless of the weather, which was very surprising to us. We also only found 4 games with attendance below 10,000. This outlier happened during the Covid Pandemic, during the 2021 season.

Looking at scoring patterns, we analyzed games like the 73-point performance against Eastern Michigan in 2014 and several lower-scoring games. Something that was interesting is that we found no strong connection between temperature and scoring. Teams seemed to perform similarly whether it was warm or cold which challenged our original common belief that weather significantly affects scoring.

Visualizations Created:

This visualization shows how many fans attended MSU football games from 2009 to 2023. We created a <u>histogram</u> that groups games by attendance ranges, from under 10,000 to over 70,000 fans. The graph makes it clear that most games drew large crowds, with a dramatic spike in the 70,000-79,999 range. This visual representation helps show how consistently MSU fills its stadium, regardless of weather conditions.
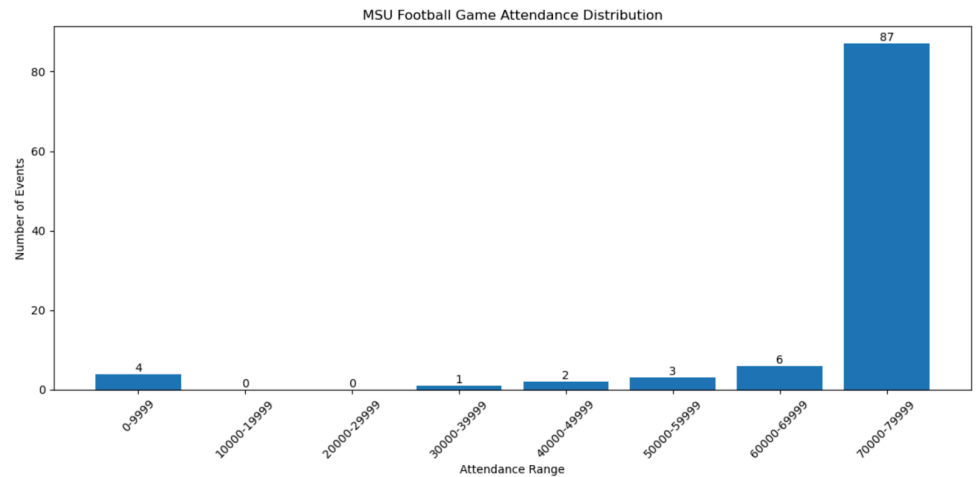

We used 'matplotlib' to create this visualization, choosing a clear bar chart format to make the patterns easy to spot. The blue bars show the number of games in each attendance range, while the numbers above each bar give the exact count. This helps readers quickly see that the majority of games (87) had attendance in the highest range.

We also created <u>a scatter plot</u> that compared MSU's points scored against the average game temperature. With a correlation coefficient of r=0.28, the weak positive trend line suggests to us that temperature has minimal impact on scoring. The data points also show wide variation, from a maximum of 73 points to a minimum of 0, with an average of 29.16 points scored across all temperatures.
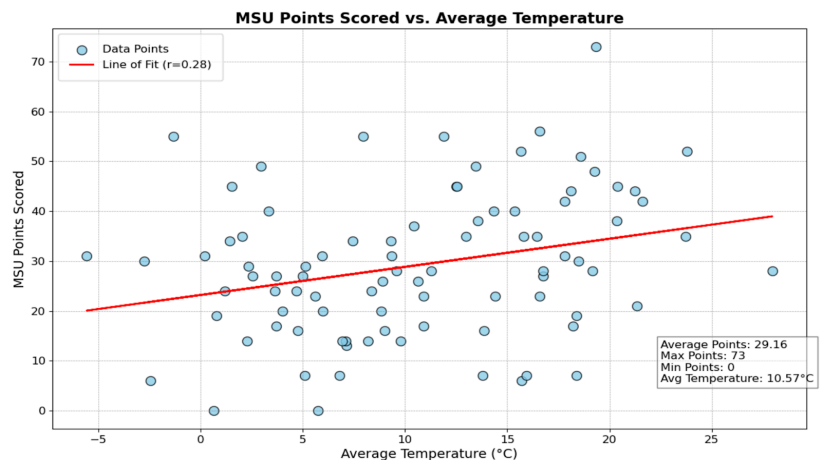
Finally, we examined how temperature affects attendance through another <u>scatter plot.</u> We found that the correlation coefficient of r=0.19 indicated an even weaker relationship between temperature and attendance. Most of the MSU games maintained high attendance between 70,000-75,000 fans regardless of temperature with an
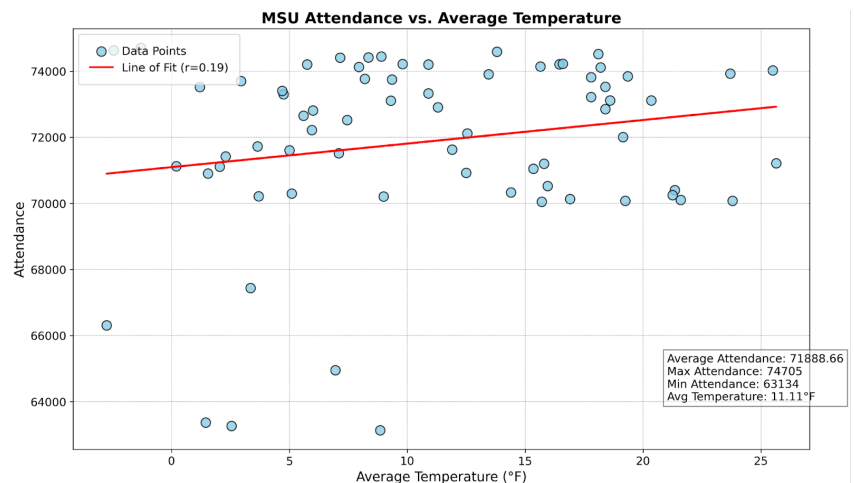
average attendance of 71,888 fans. So in conclusion, even in cold weather, attendance remained strong, only dropping below 65,000 in rare cases.

Instructions for Running Code:

First, run Attendance.py to create the attendance database. Then run weather.py to create the weather database. Then run sports.py to create the MSU home game scores database. Next, run temperature_calculate.py. Then, run process_attendance.py followed by msu_weather_score_analysis.py. Finally, run attendance_temp.py.

Documentation for Functions Written:

Our code is organized into several key Python files, each serving a specific purpose. Below is a detailed breakdown of the functions.

From temperature_calculate.py:

1. calculate_average_temperature()

Purpose: This function calculates the average temperature for each date using the maximum and minimum temperature values from the database. The results are saved to a CSV file for further analysis.

Inputs: None directly. The function queries the database to fetch date, temp_max, and temp_min values.

Outputs: A CSV file named temperature_averages.csv containing the date and corresponding average temperature rounded to 2 decimal places.

2. connect_to_database()

Purpose: This function establishes a connection to the weather_data.db SQLite database.

Inputs: None

Outputs: Returns a database connection object if successful; otherwise, raises an error.

From sports.py:

1. fetch_msu_home_games(start_year, end_year, max_records=25)
Purpose: This function fetches Michigan State University (MSU) home football game data for the specified range of years using the CFBD API. It filters the results for home games and inserts the next batch of up to 25 new records into the SQLite database.
Inputs:
   ○ start_year (integer): The starting year of the range for fetching games.

- ○ end_year (integer): The ending year of the range for fetching games.
- ○ max_records (integer, optional): The maximum number of new records to fetch and store in this run. Defaults to 25.

Outputs: None. Inserts up to max_records new rows into the msu_home_games table. Outputs messages about the progress of the fetching and insertion process.

2. count_existing_records()

Purpose: This function counts the number of records currently stored in the msu_home_games table in the SQLite database.
Inputs: None.
Outputs: An integer representing the total number of rows (records) in the msu_home_games table.

From process_attendance.py:

1. process_attendance_intervals()

Purpose: This function processes attendance data into defined intervals of 10,000. It calculates the number of games falling within each range.

Inputs: None; data is read from the full_attendance_data.db SQLite database.

Outputs: A text file (attendance_intervals.txt) containing the count of games in each attendance range. It also generates and saves a bar graph as attendance_distribution.png.

2. read_attendance_data()

Purpose: This function reads attendance records from the AttendanceHistory table in the SQLite database.

Inputs: None.

Outputs: Returns a pandas DataFrame containing attendance data.

3. generate_bar_graph()

Purpose: This function creates a bar graph visualizing attendance intervals and their respective game counts.

Inputs: A dictionary containing attendance ranges as keys and their counts as values.

Outputs: Saves the bar graph as attendance_distribution.png.

From attendance.py:

1. read_dates()

Purpose: This function reads game dates from an input text file. It processes the file to extract and clean the dates.

Inputs: file_path (string): Path to the text file containing game dates.

Outputs: A list of dates in string format. Returns an empty list if the file cannot be read.

2. convert_date_format()

Purpose: This function converts dates from Month, Day, Year format to Month Day format, which is required for Wikipedia URL generation.

Inputs: date (string): A date in Month, Day, Year format.

Outputs: A formatted date string. Returns None if conversion fails.

3. normalize_date()

Purpose: continues with convert_date_format to ensure the correct dates.

4. scrape_attendance()

Purpose: This function scrapes attendance data for Michigan State football games from Wikipedia pages. It navigates through the HTML structure to extract attendance values for a given date.

Inputs: year (integer): Season year.

date (string): Game date for which attendance needs to be extracted.

Outputs: An integer representing attendance for the specified game. Returns None if attendance data is not found or scraping fails.

5. scrape_special_years()

Purpose: Same as scrape_attendance but is used for years with a different format

6. insert_data()

Purpose: Inserts scraped data from wikipedia into the table created in create_table

Inputs: year, date, attendance

Outputs: N/A; Simply changes the table

From weather.py:

1. create_table()

Purpose: This function creates a table in the SQLite database for storing weather data.

Inputs: None.

Outputs: A table named FilteredWeatherHistory with columns for date, temp_max, temp_min, precipitation, and description.

2. fetch_data_for_date()

Purpose: This function fetches weather data for a given date from the Visual Crossing API. It processes API responses to extract temperature, precipitation, and descriptions.

Inputs: date (string): The date for which weather data is needed.

Outputs: A dictionary containing weather data such as temp_max, temp_min, precipitation, description. Returns None if the API request fails.

3. count_existing_records()

Purpose: This function counts the number of weather records currently stored in the database table.
Inputs: None.
Outputs: An integer representing the total number of rows (records) in the FilteredWeatherHistory table.

4. insert_data(day)
Purpose: This function inserts a single day's weather data into the FilteredWeatherHistory table in the SQLite database.
Inputs: day (dictionary): A dictionary containing weather data for a specific date with keys such as datetime, tempmax, tempmin, precip, and description.
Outputs: None. The function adds the data to the database and skips duplicates with a message if the entry already exists.

Documentation of Resources:

Date	Issue Description	Location of Resource	Result (Did it solve the issue?)
11/24/24	Learning to use College Football Data API for retrieving MSU football statistics	https://collegefootballdata.com/api/docs	Yes - We successfully implemented the API to gather MSU football scores and game data across multiple seasons
12/1/24	Transitioning from Ticketmaster API to attendance data	Wikipedia pages for MSU Football seasons https://en.wikipedia.org/wiki/2023_Michigan_State_Spartans_football_team)	Yes - Web scraping Wikipedia was proved more effective than the Ticketmaster API. This gave us access to comprehensive historical attendance data
12/5/24	Implementing temperature calculation methods	https://www.visualcrossing.com/resources/documentation/weather-api/timeline-weather-api/	Yes - We successfully built a system to calculate and store average temperatures for game days
12/8/24	Learning how to use the weather API	https://open-meteo.com/en/docs/historical-weather-api#hourly=temperature_2m,relative_humidity_2m	Yes - We successfully implemented the API to collect historical weather data for East Lansing game days

Key Findings:

Through our analysis of MSU football games spanning multiple seasons, we discovered that weather conditions had less impact on both attendance and scoring than we initially expected. The data showed consistent attendance patterns, with most of the MSU football games having crowds of 70,000+ fans regardless of temperature or precipitation. Even during games with extreme temperatures (like the -5.55°C game in November 2022), fan attendance remained strong.

Our shift from studying ticket prices to analyzing attendance actually gave us better insights into fan behavior. Instead of just seeing how the weather affected ticket costs, we could see how it influenced actual fan turnout. Similarly, examining total points rather than just field goals provided a more complete picture of game performance across different weather conditions.

Github Link

https://github.com/willwentrack/206_Final