Report

Introduction

For this project, data were gathered and used to create a model to predict the daily maximum temperature of four locations: Central Park, New York; Chicago Midway Airport, Illinois; Austin Bergstrom International Airport, Texas; and Miami International Airport, Florida. In addition to the prediction model, the predictions are used to make daily trades for three weeks on the trading platform, Kalshi. This report summarizes the data collection process, the model used to predict the maximum temperature, and the results of the model and Kalshi trading.

Data Collection Process

Data Collection

The five data sources I found for this task are Open-Meteo's Historical Weather API, National Oceanic and Atmospheric Administration (NOAA)'s Daily Summaries dataset, Visual Crossing API, Meteostat, and the Florida Climate Center. Each data source had its limitations. The Florida Climate Center had very few features for its weather data, Meteostat is tedious to use, the Visual Crossing API had a very low number of free API calls, NOAA's dataset is a bit outdated, and the Historical Weather API had a low number of free daily API calls. Also, since most of the data sources had the same features and data, for the model's training I decided to just use NOAA and the Historical Weather API for each of the four locations. I used NOAA to gather weather data from January 1, 1998, to March 1, 2024, and used it for the training and validation sets, whereas I used the Historical Weather API to collect data from March 2, 2024, to March 22, 2024, and used it for the testing set. The code for the Historical Weather API calls is provided in the API-open-meteo.ipynb file.

Data-Preprocessing

I did some preliminary analysis of the data and found that there was much missing data. Thus, I removed all the columns that had 40% or more missing data and I filled any missing value by the value in the previous row. I did this instead of removing any rows because the data needed to be sequential. The main features of the data are maximum temperature, minimum temperature, precipitation level, and snow depth. The data was also normalized to ensure that each feature contributes proportionally to the model's learning process.

Model

Week 2

For week 2, I made a simple linear regression model with L2 regularization and five-fold cross-validation. I used scikit-learns's linear_model.RidgeCV with values of alpha, 10^{-3}, 10^{-2}, 10^{-1}, and 1. Since this is not the final model I made, I did not include it in the submission.

Week 3

During the week, I decided to implement a more complex model. Thus, I created an LSTM recurrent neural network and included this model in my final submission. The code is in the file temperature_prediction_rnn.ipynb. I made the LSTM model using the Keras library because I was already familiar with the library. The sequential model takes groups of 7 days of data as input and contains 7 layers, three LSTM layers each followed by a dropout layer, and a final dense layer for the output. The model uses the ADAM gradient descent optimizer and uses the mean squared error as the loss function. The model was trained for 50 epochs with a batch size of 64 and used 10% of the data for the validation set. The reason I chose 50 epochs for the training is because the model did not noticeably improve after 50 epochs and I did not want to overfit the model to the training set.

Results

Week 1

During week 1, I made manual predictions using information from various websites. The results of the trading are shown in the table below. More information on the trades is available in the 'Kalshi Trading Log.xlsx' file.

Date	Start Balance	End Balance	Location	Temperature Prediction	Correct
Feb 26, 2024	10000	10009.1	Austin	No, 80 or below	True
			Chicago	No, 62 or below	True
				No, 63 to 64	True

			Miami	No, 75 to 76	False
			NYC	No, 51 to 52	True
Feb 27, 2024	10009.1	10477.03	Austin	No, 85 to 86	True
			Chicago	No, 70 or below	True
			Miami	No, 74 to 75	True
				No, 76 to 77	True
			NYC	No, 54 to 55	False
				No, 58 to 49	True
Feb 28, 2024	10477.03	10505.24	Austin	No, 58 or below	True
				No, 59 to 60	True
			Chicago	No, 30 to 31	True
				No, 75 to 76	True
			Miami	No, 77 to 78	True
			NYC	No, 59 to 60	True
Feb 29, 2024	10505.24	10736.48	Austin	No, 55 to 56	True
			Chicago	No, 43 to 44	True
			Miami	Yes, 77 to 78	False
				No, 79 to 80	True
			NYC	No, 41 to 42	True
				No, 43 or above	False
March 1, 2024	10736.48	10820.67	Austin	No, 74 to 75	True
			Chicago	No, 48 to 49	True
			Miami	No, 77 to 78	True
			NYC	No, 46 to 47	True

I was quite cautious about my decisions and mostly bought events for ‘No’ which has an 83% chance of being correct if I chose randomly. Thus, I was able to procure some money.

Week 2

During week 2, I made my decisions using the predictions by the linear regression model described previously. The results of the trading are shown below.

Date	Start Balance	End Balance	Location	Temperature Prediction	Correct
March 4, 2024	10820.67	10807.17	Austin	Yes, 77 to 78	False
			Chicago	Yes, 74 to 75	False
			Miami	Yes, 79 to 80	False
			NYC	Yes, 58 to 59	True
March 5, 2024	10807.17	10795.17	Austin	Yes, 85 to 86	False
			Chicago	Yes, 48 to 49	False
			Miami	No, 78 to 79	False
			NYC	Yes, 46 or below	False
March 6, 2024	10795.17	10783.53	Austin	Yes, 78 or below	False
			Chicago	Yes, 43 or below	False
			Miami	Yes, 85 or above	False

			NYC	Yes, 52 to 53	True
March 7, 2024	10783.53	10776.79	Austin	Yes, 75 to 76	True
			Chicago	Yes, 49 to 50	True
			Miami	Yes, 83 to 84	False
			NYC	Yes, 52 to 53	False
March 8, 2024	10776.79	10763.79	Austin	Yes, 79 to 80	False
			Chicago	Yes, 49 to 50	False
			Miami	Yes, 87 or above	False
			NYC	Yes, 56 to 57	True

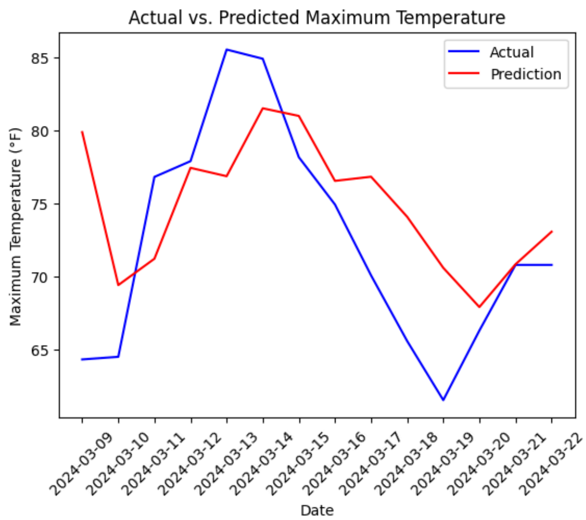
For this week, I mostly purchased events for 'Yes'. The model did not do well and only predicted correctly ¼ of the time.

Week 3

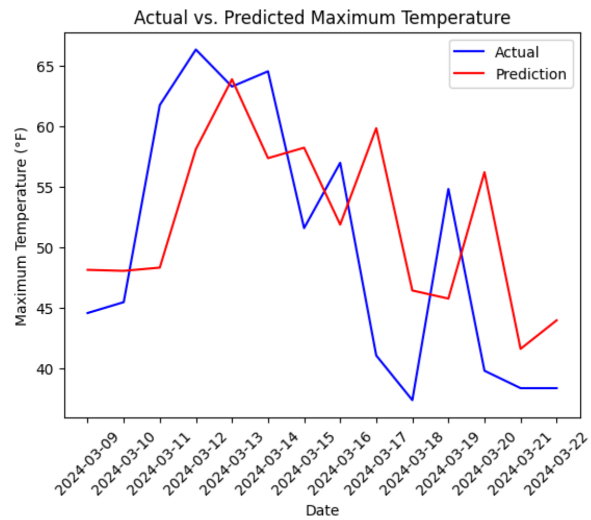
During week 3, I used the LSTM model to make the decisions and used the Kalshi API to automatically trade. I modified the KalshiTradingV2.ipynb file in the KalshiAPIStarterCode folder to implement the trades. The table below summarizes the results of the trading.

Date	Start Balance	End Balance	Location	Temperature Prediction	Correct
March 4, 2024	10763.79	10752.79	Austin	Yes, 64 or below	False
			Chicago	Yes, 36 to 37	True
			Miami	Yes, 89 to 90	False
			NYC	Yes, 55 or above	False
March 5, 2024	10752.79	10734.79	Austin	Yes, 67 or above	False
			Chicago	Yes, 50 or below	False
			Miami	Yes, 80 or above	False
			NYC	Yes, 53 or above	False
March 6, 2024	10734.79	10711.75	Austin	Yes, 62 or below	False
			Chicago	Yes, 37 to 38	False
			Miami	Yes, 80 to 81	False
			NYC	Yes, 51 to 52	False
March 7, 2024	10711.75	10701.61	Austin	Yes, 67 or below	True
			Chicago	Yes, 44 or above	False
			Miami	Yes, 81 to 82	True
			NYC	Yes, 44 to 45	False
March 8, 2024	10701.61	10688.92	Austin	Yes, 72 to 73	False
			Chicago	Yes, 41 or below	True
			Miami	Yes, 79 to 80	False
			NYC	Yes, 49 or above	False

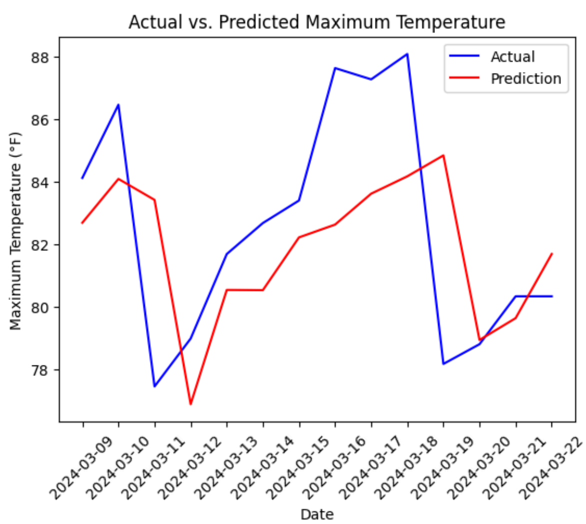
For this week, only events for ‘Yes’ were purchased. The model performed poorly and only made a correct prediction ⅕ of the time. The graphs below demonstrate the history of the model’s prediction compared to the actual value.



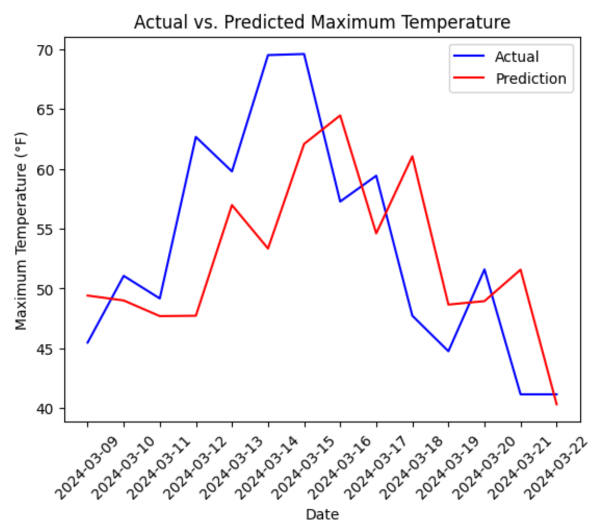
Temperature Prediction in Austin



Temperature Prediction in Chicago



Temperature Prediction in Miami



Temperature Prediction in NYC

From the graphs, we can see that the prediction of the model is mostly influenced by the previous day and the model is unable to predict large fluctuations in temperature. However, the model’s predictions remain close to the actual value and do not deviate far.

Conclusion

I primarily used data from Open-Meteo's Historical Weather API and NOAA's Daily Summaries dataset to train the model. The final model that I used is a 7-layer LSTM recurrent neural network that is trained using the ADAM optimizer and a mean-squared error loss function. The final model did not perform well on the Kalshi trading platform but the predictions do not deviate far from the actual maximum temperature. To further improve my model I would focus more on feature extraction and collecting data on other weather features. For example, I could create a new feature that stores the monthly maximum and minimum temperatures, so that the model can find more patterns. Also, I should have spent more time finding sources with fewer missing data on weather variables such as sunlight duration, humidity, and average wind speed.