Bespoke Almanacs for Monterey Bay Event Planners

Jeff Treviño, 2019

Contents

I. Problem Statement and Proposal

Background + The Client's Decision (The Problem)

To decide when to schedule events, Monterey peninsula event planners for outdoor weddings and cultural festivals, such as the Monterey Jazz Festival, must often guess whether or not a given weekend will be sunny, sometimes up to a year in advance. But the area's climate can be notoriously volatile and counterintuitive: sunny weather ends sometime in May or June and begins again sometime in September, after a few months of fog throughout the summer season, with gaps of sun between

The Action

I propose to develop a predictive model based on weather data for the city of Monterey for the last ten years, to assign a sunniness probability for each calendar weekend. I will need to align calendar weekends across years, i.e., "the first weekend in March," and create an average sunniness score for a given weekend that takes into account three component days (Friday, Saturday, and Sunday) and averages across ten different corresponding weekends through time; a "calendar weekend" data structure will include data for thirty different calendar days (Friday, Saturday, and Sunday for ten years into the past). The dataset will be ordered from the National Center for Environmental Information, and will include several daily summary fields: total daily precipitation, average wind speed, and (most importantly) daily weather type. The data requested spans a period of May 14th, 2008 to May 14th 2019.

The project requires several considered modeling choices. The reality of climate change problematizes the question of timeframe: how many past years of data really reflect a weather trend relevant to the immediate future, rather than pre-global-warming patterns that no longer apply? But also how many years of data will be necessary to avoid overfitting to this small amount of data? One challenge of interpretation will be the quantification of qualitative, categorical data: how should a “partly cloudy” or “partly sunny” day impact a weekend’s overall sunniness score? The model will also require binarization, as the goal is to determine a “sunny/not sunny” binary value for each calendar weekend — should this binarization occur for each day, for each weekend, for each day throughout history, or for each weekend throughout history?

Deliverable

Because the aim of the project is to improve event planners’ future event scheduling decisions, I will create a calendar that displays sunniness probabilities for each weekend, an interface that will allow planners to make more informed choices when scheduling outdoor events in the area.

The Value of an Improved Decision

While this won’t enable us to predict the future, it should allow event and festival planners in the Monterey area to make more informed event planning choices in the future. This might be especially useful to those planners who have recently moved to the area and aren’t yet familiar with the area’s peculiar weather patterns, and to smaller businesses or individual entrepreneurs who can’t afford data consultants.

II. Data Collection and Wrangling

I obtained my dataset as a CSV file from the freely available National Oceanic and Atmospheric Administration's online database. The data contained hourly sky condition and visibility, collected from the Monterey airport between 2009 and the present. The data arrived via email request as a CSV file, which I imported into Pandas as a dataframe.

Cleaning Steps

Converted each row's date-time column to a Python datetime object in a new column.

Set this new column to be a datetime index for the dataframe.

Convert strings to float across dataset where possible with to_numeric. This coerced suspect values (see below), asterisk, and T values as floats if possible (for single letter numeric value suffixes) or as NaNs (for alphanumeric strings).

Backfilled the maximum daily temperature, minimum daily temperature, daily precipitation, and maximum daily wind speed columns throughout the day.

For hourly sky condition, learned to interpret the sky obscuration okta scale (1-9) and the abbreviation codes that appear in the hourly sky condition column, and then replaced the string representation of the list of sky conditions with a more accessible list of dictionaries; each dictionary's key is a condition code, and each dictionary's value is a SkyCondition namedtuple containing obscuration and vertical_distance fields.[1]

[1] This structure works better than a single dictionary for this data, because condition codes occur multiple times for a single hours sometimes, which would prevent a single dictionary from having unique keys.

Missing Values

The presence of a NaN for a particular value means a couple different things in this dataset. Because the hourly weather type column tracks only the presence of rain, mist, or fog, a missing value in this column indicates the absence of one of these three types of weather. Otherwise, a NaN more generally means that no measurement is available.

The DailyPrecipitation column records the day's rainfall in inches on each day at 11:59 PM but otherwise contains all missing values. This reading sometimes contains a "T", "Ts", or "0:00s" values, which appear to indicate "trace amounts" of precipitation; these have been replaced with zero values and then backfilled with the other daily measurements.

The hourly temperature and hourly wind speed columns are missing about four thousand values each. It would be strange to impute these values, as the point of tracking hourly temperature is to characterize a single particular hour, so I've elected to leave these missing values as NaNs. The presence of daily temperature max and min, as well as the ability to calculate a daily average, should provide solutions if this becomes a problem.

The sky condition representation contained an especially challenging missing value: a clear day lacks a following integer to represent the vertical visibility from ground to clouds, but no NaN appeared for this value, as it occurred within a string that required custom parsing. The parsing algorithm handles this by replacing the missing value with a 0, as would normally trail the clear day code to indicate a missing value, as in "CLR:00 00".

The HourlyWindSpeed and HourlyDryBulb temperature columns contained many missing values indicated with an asterisk string ('*'). These have been replaced with NaNs.

To avoid processing errors on the hourly sky condition column, the NaN values in that column have been converted to empty lists.

Peculiar to this dataset, NOAA defines a value with an “s” suffix indicates a “suspect value.” In these cases, the value has nonetheless been coerced as a floating point value without its suspect suffix.

Outliers

There are some outliers in various respects. The maximum and minimum daily temperatures tend to have from four to six spikes in a given year. And any Californian will tell you that daily precipitation has seen some historically surprising highs and lows in the past several years.

III. Exploratory Data Analysis Summary

A first plot of sky obscuration by hour of the day revealed that 10 AM to 4 PM are generally clearer than all other hours; for this reason, further analysis focused on these hours for each day in the data set. Obscuration histograms reveal that there are generally more cloudy than sunny days in a given year, and that clear days fall disproportionately in the months of January and November. A moving two-week temperature average reveals a temperature increase from January through November, followed by a sudden drop to winter temperatures. Correlation plots revealed some obvious correlations: wind correlates with rain, and obscuration correlates with cool (but not cold) temperatures. Brief exploration of rainfall reveals erratic patterns in the last decade. The data show that some winter days are much more likely to be rainy than others. A moving two-week obscuration average provides a first look at obscuration seasonality, which aligns roughly with temperature seasonality. Lower obscuration from 10 AM to 4 PM also matches the portion of the daily humidity curve when humidity drops below 70%. Finally, a view of relative obscuration risk — the probability of a calendar day's obscuration over the rest of the month's days' average obscuration probability — provides a look at which days in each month are relatively more or less likely to be cloudy.

IV. Time Series Analysis and Modeling

The modeling stage of the project benchmarks several verified time series analysis algorithms against one another, progressing from simpler to more complex.

First, a preliminary exploration of stationarity confirms that temperature, humidity, and obscuration are stationary time series. Next, correlation and autocorrelation plots provide insight into the order of best-fitting ARMA models, the order of which is then algorithmically determined. Then ARMA models fit and predict temperature, obscuration, and humidity data.

Next, the models add seasonal components. Naive season-level-trend decomposition provides an initial view of trend and seasonality, and analysis proceeds to a comparison of exponential smoothing models against a generalized additive model.

In fitting exponential smoothing models to temperature data, as predicted by Hyndman and Athanasopoulos, the model with damped trend and multiplicative seasonality performs the best by both error and Akaike Information Criterion (AIC) measures. In the case of the humidity data, the damped trend with additive trend and seasonality performs best by both error and AIC measures.

The facebook prophet algorithm extracts two levels of hierarchical seasonality (weekly and annually) as Fourier harmonics, and models trend according to a piecewise linear or logarithmic function smoothed according to optimized Bayesian priors. This algorithm yielded the lowest root mean square error, and its predictions were used for calendar output.

V. Calendar Export

To create the final deliverable relative obscuration risk ratio statistics for each calendar day were exported, along with facebook prophet predictions for each day’s temperature, humidity, and obscuration. A Python script packs these values into an event body string and posts a prediction event to the corresponding calendar day using the Google Calendar API, to a publicly available calendar.