Project 5 Summary:
This Week's Florcast

Project Design:
I used time series analysis to predict daily order counts for my aunt's flower shop for up to 1 week in advance.

As a small business owner who personally customizes each order, my aunt sometimes struggles with planning her inventory. For example, if she orders too many flowers, she may waste inventory and lose money. If she does not order enough roses for Valentine's Day, she would need to turn away business. This model could be used by my aunt to plan her inventory for the week.

I merged multiple CSV files for Products Ordered and Basic Orders Reports to get time stamps and product descriptions for each order. I then used a combination of spaCy text preprocessing tools, customized stop words, and other text cleaning methods with the aim to get actual <u>quantities</u> of various flowers used in each arrangement. I was not able to get this full level of detail, however, and moved forward with my final project of getting <u>order counts</u> of:

1. Total number of orders
2. Roses
3. Lilies
4. Orchids
5. Designer's Choice
6. I also created a column called "Other" to illustrate that there are many other types of flowers that can be ordered, and my project at this stage is a proof of concept. "Other" is the difference between the total and the individual order counts (roses+lilies+orchids+designer's choice)

Tools

- Data Collection
    - Kiyo's (my aunt's flower shop) records
- Python
    - Data Cleaning & Analysis: Pandas
- NLP
    - spaCy
- Time Series
    - Naïve (yt+1 = yt)
    - Statsmodels: Simple Exponential Smoothing
    - Facebook Prophet
- Visualization of Results
    - Matplotlib
    - Flask

Data
I obtained my data by downloading CSV files from my aunt's order tracking system. I removed personal information, dropped duplicates, and merged the products ordered and order reports together to get timestamps and product descriptions.

I also obtained Census data for Sacramento county—from which I collected annual counts (for 2014-2019) of households with married couples and households with children. Unfortunately, I was not able to successfully add these as regressors in the Prophet model because I would need future counts and these were static counts for 2014-2019 (not daily).

Methods
I used spaCy and a lot of manual python code to clean to clean the text—to correct typos, misspellings, punctuations that were part of the words, etc. I also customized stop words.

For modeling, I did the following:

1. Naïve (use yesterday to predict today or last year to predict this year)—both of which resulted in ~2x the MAE compared to Facebook Prophet.
2. As another comparator, I used a Simple Exponential Smoothing (SES) model from Statsmodels—which predicts a weighted average, giving more weight to recent history. This model is nice because it has only one hyperparameter (alpha)—which it can optimize on its own.
3. I also used Facebook Prophet's tool. This is a fairly intuitive model, runs pretty quickly, and can identify its own seasonality and trend. Many blogs state that it performs better than an expertly-tuned ARIMA model. The other nice thing about Prophet is there are some tuning options—such as adding holidays.

Kristen Tokunaga

Project 5 Summary:
This Week's Florcast

The tricky parts about doing time series were:

- Train-test splitting and cross validation
  - SKLearn has a timeseries split, which I used, to make 5 folds. It creates different lengths—but does include some of the same training data in each fold. For example, my data was split by SKLearn grouping the first 304 rows into the first train set, then the first 608 rows (inclusive of the first 304 rows), and so on.
  - Prophet also has some cross validation methods that I used to compare to my manual train-test splitting—both had similar results. The nice thing about Prophet's cross-validation method is it will calculate MAEs for as many days as you allow it to predict (the horizon). I set my model to predict up to 7 days in advance, since flowers are perishable (they bloom, they wilt, they die).
- Moving the training period forward: rolling vs. extending windows
  - The extending window is sort-of covered with SKLearn's timeseries split and FB Prophet's cross-validation options, but if I wanted to test different train sizes and rolling windows, I had to code this on my own. After I got it running, I found that extending window performed better.

To try to get Prophet to perform better:

1. I added holidays (Valentine's Day and Mother's Day). The challenge with adding holidays is we need to know the future holidays—which is possible for Valentine's Day and Mother's Day; however, I would have liked to add other factors like graduations or special events—which are harder to predict. This improved my model's performance (from ~6.3 to 3.7), but for some reason past Valentine's Days had 100-150 orders, but my model predictions ~80-90 orders for future Valentine's Days.
2. I did a log transform on the 'y'. Because there were a lot of zero counts, I actually used log1p—which adds 1 to each value before taking the log. This also significantly improved my MAE from ~3.7 to 0.4.

horizon	mse	rmse	mae
1 days	0.345806	0.588053	0.414531
2 days	0.338155	0.581511	0.411543
3 days	0.329321	0.573865	0.407059
4 days	0.329823	0.574302	0.407897
5 days	0.329437	0.573966	0.408056
6 days	0.322552	0.567936	0.404272
7 days	0.323145	0.568458	0.404438

My aunt's average order count in 2019 was 8—so these were decent results. However, I would like to further tune the model to more accurately predict Valentine's Day and Mother's Day.

I also used this model to predict number of ORDERs containing roses, lilies, orchids, and "designer's choice".
Below are my results:

Roses

horizon	mse	rmse	mae
1 days	0.175645	0.419100	0.343461
2 days	0.176135	0.419684	0.342961
3 days	0.176144	0.419696	0.342781
4 days	0.174172	0.417339	0.340553
5 days	0.173879	0.416989	0.340241
6 days	0.174871	0.418175	0.341126
7 days	0.177061	0.420786	0.343784

Lilies

horizon	mse	rmse	mae
1 days	0.212878	0.461387	0.385847
2 days	0.212356	0.460821	0.385395
3 days	0.212394	0.460862	0.385891
4 days	0.212786	0.461288	0.386096
5 days	0.211748	0.460161	0.384629
6 days	0.211904	0.460330	0.384823
7 days	0.213009	0.461529	0.386519

Orchids

horizon	mse	rmse	mae
1 days	0.305018	0.552285	0.443345
2 days	0.303506	0.550914	0.441335
3 days	0.301329	0.548935	0.439811
4 days	0.301751	0.549319	0.440062
5 days	0.299276	0.547061	0.437409
6 days	0.299247	0.547035	0.437641
7 days	0.299239	0.547027	0.437343

Designer's Choice

horizon	mse	rmse	mae
1 days	0.339661	0.582805	0.443183
2 days	0.337230	0.580715	0.441366
3 days	0.336057	0.579704	0.438946
4 days	0.335897	0.579566	0.439244
5 days	0.336353	0.579959	0.438564
6 days	0.337288	0.580765	0.439859
7 days	0.337161	0.580655	0.439710

Kristen Tokunaga

Unfortunately, her average number of roses, lilies, orchids, and designer's choice orders are closer to 1-2 per day—so these MAE errors are relatively high.

Finally, I put my model into production by creating a Flask app—called "The Florcaster". I also used a CSS style editor to prettify my app.

Below is a sample output:

The FLORcaster

Prediction:

Date	Total Orders	Roses	Lilies	Orchids	Designer's Choice	Other
Friday, 12/06	9	0	0	1	2	6
Saturday, 12/07	8	0	0	1	1	6
Sunday, 12/08	1	0	0	0	0	1
Monday, 12/09	6	0	0	0	2	4
Tuesday, 12/10	7	0	0	1	2	4
Wednesday, 12/11	9	0	0	1	2	6
Thursday, 12/12	8	0	0	1	2	5

KIYO'S
FLORAL DESIGN

What I Would Do Differently
If I had more time, I would have liked to:

- Explore other models, such as SARIMAX or LSTM
- Add more exogenous features
  - Weather
  - More holidays (e.g., graduations)
- Add more functionality to my app:
  - Specifically, my goal would be to add an option to have my aunt upload the new "products ordered" and "order report" files—and automatically make predictions on the new data. I did not have time to complete to this level of functionality, but I look forward to trying in the future.

Future Work

- Expand to predict actual quantities of each flower—rather than order counts
  - Then add inventory records: the difference between predicted quantities and inventory-on-hand can assist in ordering
- Develop app further to be able to upload a file and predict on the newest file load

Project 5 Summary:

This Week's Florcast

- After getting a more accurate model: I'd like to optimize further by narrowing the prediction windows for yhat_upper and yhat_lower, so that I can get more confident predictions – and learn how I can make a model that produces a yhat that minimizes waste