ATOC5860 – Application Lab #6
Machine Learning with Weather Data
Spring 2022

Note: You will need to use the python environment provided (environment.yml), especially for notebook #2. These notebooks were written by Eleanor Middlemas in 2020 (https://github.com/e-middlemas/ML_application_lab). They were last adapted/updated for use in ATOC5860 during Spring 2022.

Notebook #1

ATOC5860_applicationlab6_cluster_mesa_data.ipynb

LEARNING GOALS

1) Use k-means clustering as an example of unsupervised (grouping events into different categories) machine learning
2) Become familiar with the limits and applicability of K-means clustering to detect seasons in Boulder, Colorado
3) Assess sensitivity of K-means to standardization, changing the variables used for the clustering (also called "features"), and number of clusters (4 vs. 3).

DATA and UNDERLYING SCIENCE:

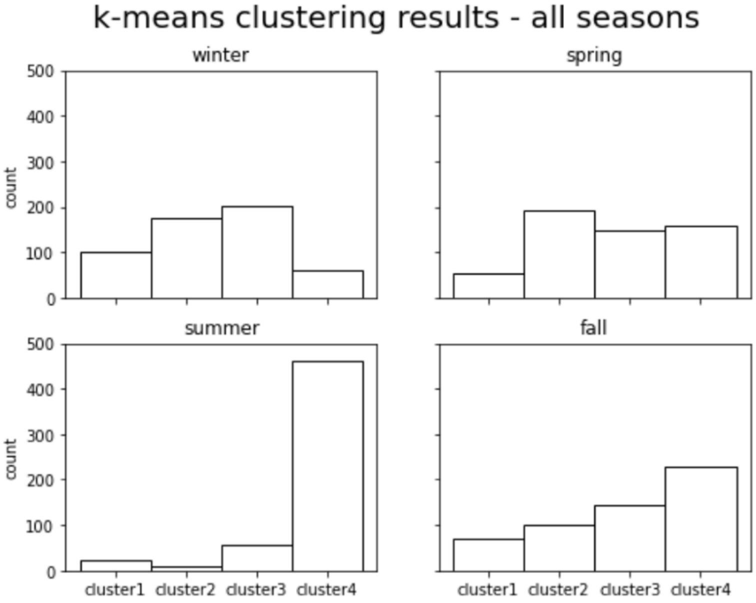
You will be working with weather data from the NCAR Mesa Laboratory in Boulder, Colorado. We'll call this dataset the "Mesa dataset". The data go from 2016-2021. Information on the site and the instruments is here: https://www.eol.ucar.edu/content/ncar-foothills-lab-weather-station-information. Real-time data from the site is here: https://archive.eol.ucar.edu/cgi-bin/weather.cgi?site=ml. Note: Each year in this dataset has 365 days. Leap year data (i.e., Feb. 29 data for 2016 and 2020 have been excluded.)

In this notebook, you use K-means clustering to classify the mesa dataset weather data into different clusters. Why would we cluster weather observations? We already know which observations are in which season by looking at the date. But we all know that a day in February sometimes feels like summer and a day in September can feel like winter. We often have multiple seasons in a single week... So this could be quite fun to see how the algorithm decides how to cluster our data and assign each day to a "season". :) Will each cluster will look like a season – On Va Voir (We'll See)!

Questions to guide your analysis of Notebook #1:

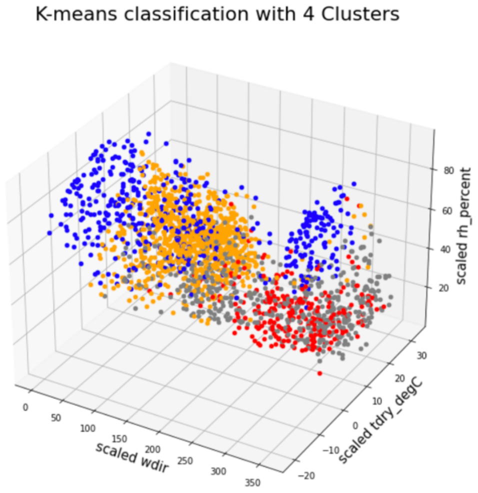
1) Start with 4 clusters. Cluster the data at 17 UTC (mid-day in Colorado). What is the seasonal occurrence of the 4 clusters? Do the 4 clusters correspond to Fall, Winter, Spring, and Summer? Why or why not?

The clusters somewhat correspond to the seasons. For example, cluster 4 corresponds super closely to the Summer!



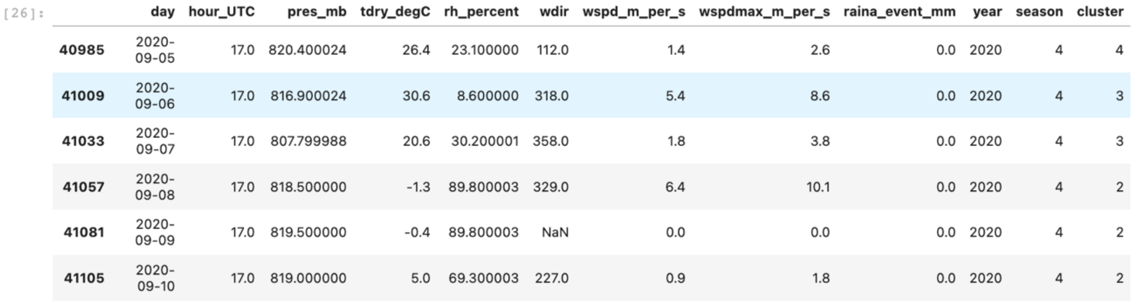
2) Based on 2D and 3D scatter plots of the cluster centers and the data – Which weather variables help (or NOT help) define the clusters?

I would say that the scale temperature is less important than expected! The scale RH and WDir might be more important than I expected! Wind speed is the least important!



3) What do the clusters show during the time period from September 5-15, 2020 (Labor Day 2020)? Are the cluster assignments consistent with the weather experienced over that time period? Are there other date ranges that you would like to check out?

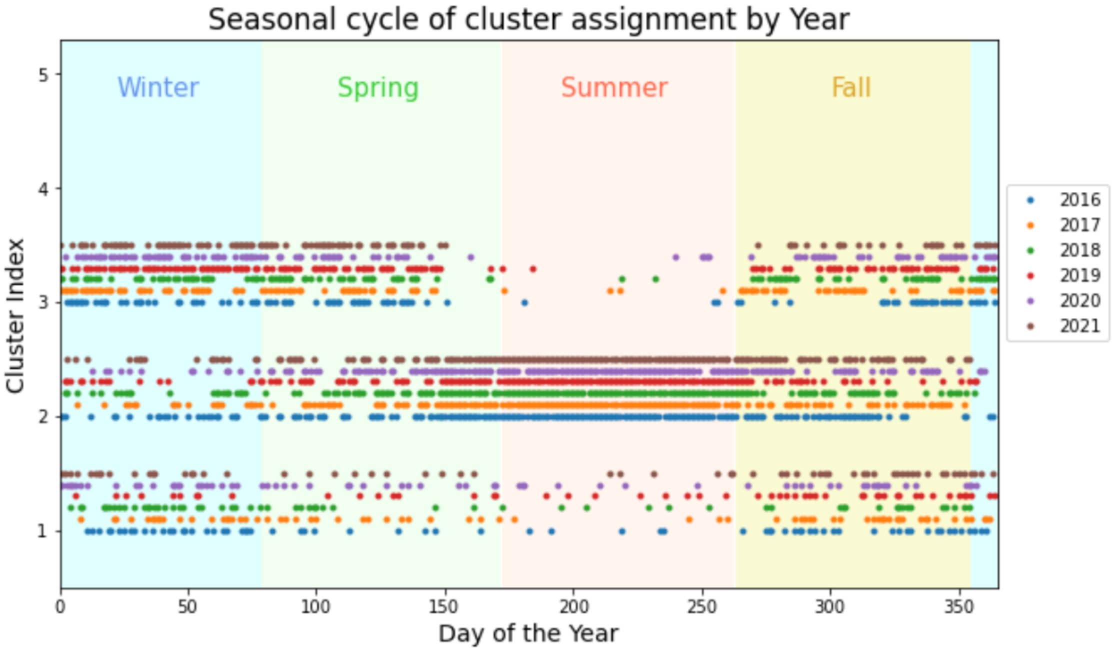
The cluster assignment is not consistent with what actually happened with the weather! I would be curious to look at a time period in the middle of the winter, during one of Boulder's crazy 24 hour temperature swings.



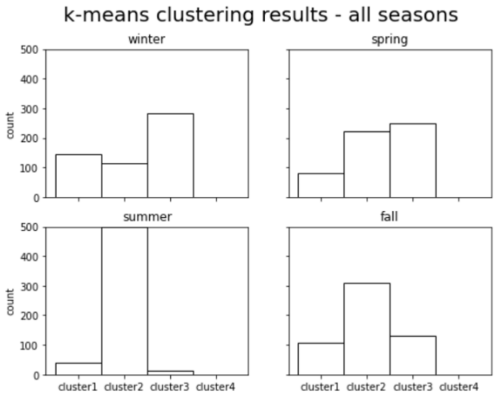
[26]:

	day	hour_UTC	pres_mb	tdry_degC	rh_percent	wdir	wspd_m_per_s	wspdmax_m_per_s	raina_event_mm	year	season	cluster
40985	2020-09-05	17.0	820.400024	26.4	23.100000	112.0	1.4	2.6	0.0	2020	4	4
41009	2020-09-06	17.0	816.900024	30.6	8.600000	318.0	5.4	8.6	0.0	2020	4	3
41033	2020-09-07	17.0	807.799988	20.6	30.200001	358.0	1.8	3.8	0.0	2020	4	3
41057	2020-09-08	17.0	818.500000	-1.3	89.800003	329.0	6.4	10.1	0.0	2020	4	2
41081	2020-09-09	17.0	819.500000	-0.4	89.800003	NaN	0.0	0.0	0.0	2020	4	2
41105	2020-09-10	17.0	819.000000	5.0	69.300003	227.0	0.9	1.8	0.0	2020	4	2

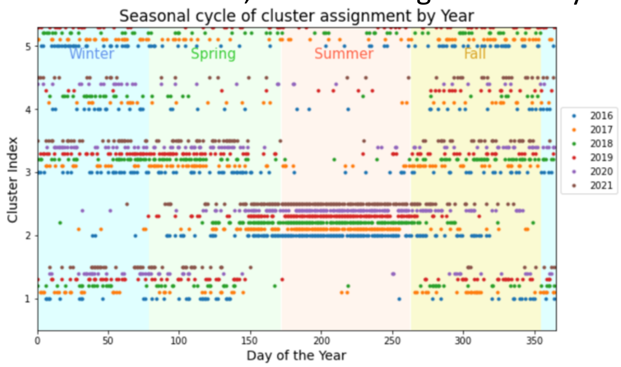
4) Re-run the analysis. But now use three clusters instead of four clusters. Compare your cluster analyses for 4 clusters and 3 clusters. Do the results for 4 clusters or 3 clusters make more sense to you based on your analysis and also your experience living in Boulder, Colorado? Which number of clusters provides a better fit to the data?



I think three clusters does a better job at defining the seasons, although only summer is very distinctly clustered (yet again).



Even with 5 clusters, summer is again the only super well-defined season.



<u>Notebook #2</u>

OPTION #1: ATOC5860_applicationlab6_supervised_ML.ipynb – Use environment.yml or

OPTION #2: supervised.ipynb – Run in Google CoLabs

Note: You will need to change the google drive paths to match those on your computer.

LEARNING GOALS:

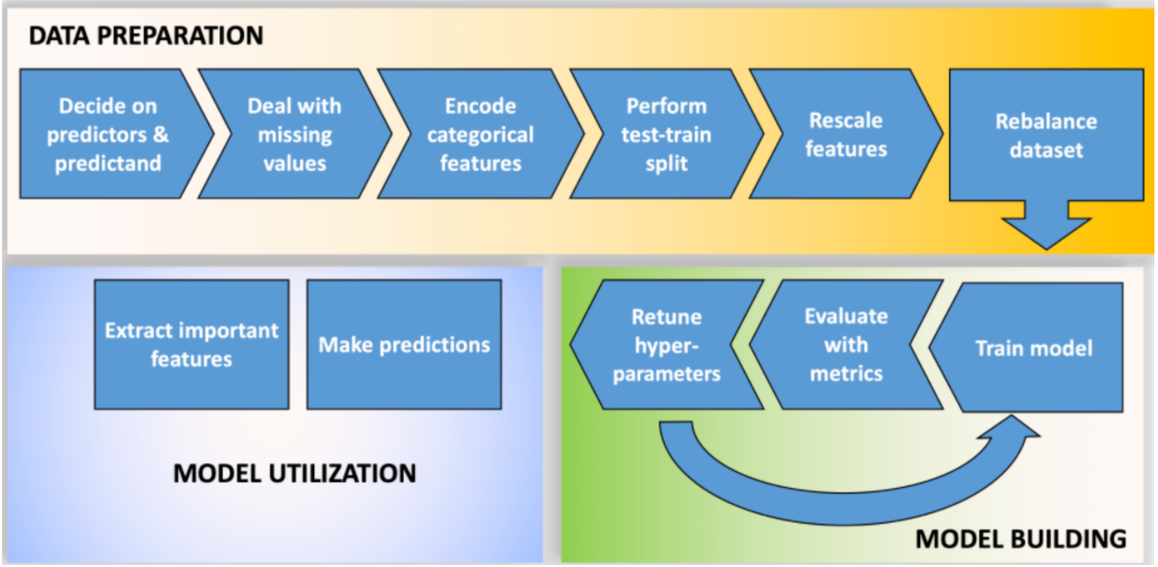
1) See an example of the data processing pipeline (workflow) required to utilize supervised machine learning techniques.
2) Implement and compare four different supervised learning algorithms
3) Understanding two outcomes of supervised learning algorithms: prediction and feature importance.
4) Start building a foundation for future machine learning including the following terms: cross-validation, training vs. testing data, metrics (accuracy, recall, precision, f1 score, etc.), overfitting/underfitting, balancing datasets, hyperparameters, & feature importance. Some future learning resources are provided… but there's a lot available! *Share resources you find valuable.*

DATA and UNDERLYING SCIENCE:

We will use the Christman dataset which contains weather observations from Fort Collins, Colorado for the year 2016. We will build and train four machine learning models to predict something we already know from the dataset: ***Is it raining?***. The point is not to conduct cutting-edge research or make novel predictions. Instead, the purpose here is to showcase supervised machine learning (ML) models and methods. By the end, we hope you can walk away with more confidence to learn and apply these tools to new problems.

Let's say you want to determine which features or atmospheric variables are the best predictors of rainfall. Often, one simply regresses some metric of precipitation onto various atmospheric variables. Then, you assume that whatever returns the highest regression coefficient is the best predictor. While this approach with linear regression presents a fine first guess, it poses a few problems. Linear regression assumes: 1) atmospheric variables are linearly related to precipitation, 2) atmospheric variables are uncorrelated. Yet, both are false assumptions. While a linear relationship between predictor & predictand is a good first guess, why limit yourself to linearity when you can just as easily relax that assumption using supervised Machine Learning...

This notebook will step through the following Machine Learning model pipeline:



After prepping the data, we will build and train four machine learning models and make predictions with them. The four machine learning models we will implement are: Logistic regression, Random Forest, Singular vector machines/classifier, Neural Network. Finally, we will determine which variable ("feature") is the best predictor, i.e., we will assess "feature importance".

Pros/Cons of these Methods (from Eleanor Middlemas)

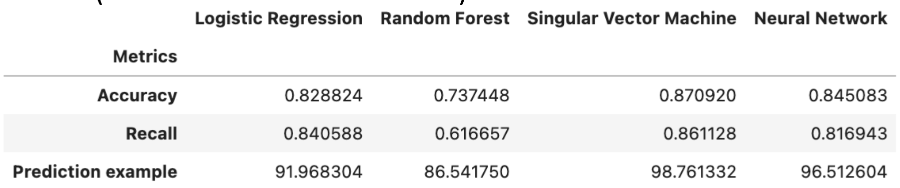
1. Logistic regression tends to overgeneralize or underfit data, but is easy to implement, to understand and easy to back out feature importance.

2. Singular Vector Machines are great at capturing complex relationships, but cannot back out feature importance. Plus, the use of the kernel makes them hard to interpret.
3. Random forests are easier to understand, generally do not overfit, and can capture complex relationships, and can provide feature importance, but they can be slow to train and there are a lot of hyperparameters to choose from.
4. Neural Networks are great at capturing complex relationships. But they are slow to train and are susceptible to overfitting.

Questions to guide your analysis of Notebook #2 – See also questions at the end of supervised.ipynb:

1) Which machine learning model performs the best to predict rainfall? What metrics did you use to make this assessment?

Of the four ML methods, the singular vector machine was the best at predicting the rainfall (for all metrics we looked at).



	Logistic Regression	Random Forest	Singular Vector Machine	Neural Network
Metrics				
Accuracy	0.828824	0.737448	0.870920	0.845083
Recall	0.840588	0.616657	0.861128	0.816943
Prediction example	91.968304	86.541750	98.761332	96.512604

2) Describe the difference between accuracy and recall. Why did we choose to use accuracy, recall, and predicted precipitation probability as a way to compare models? In forecasting: when is a false positive (you said it would rain, it didn't rain) preferred over a false negative (you said it wouldn't rain, it did rain)?

Accuracy is the proportion of precipitating hours or non-precipitating hours that are correctly predicted by the model. This tells us the percentage correct!

Recall is the proportion of precipitating hours that are correctly predicted by the model. This only tells us if we are good at predicting precipitation alone.

We use these metrics to tell us if our model is working! We would prefer a false positive potentially for an event that would be held outside where we cannot have rain.

3) One important "gotcha" in a machine learning workflow or pipeline is the order of data preparation. **Why should one should perform the train-test split before feature scaling and rebalancing?** *Hint: think about using a trained model for future predictions.* Do you want your scaling of the testing data to depend on the training data? Why perform a test-train split at all?

We must do the test-train split so that the biases inherent in the data don’t influence the skill of the model to predict. If we were to scale and rebalance the data before rebalancing the data, our statistics would be much different (mean and standard deviation for example, would not be 1).

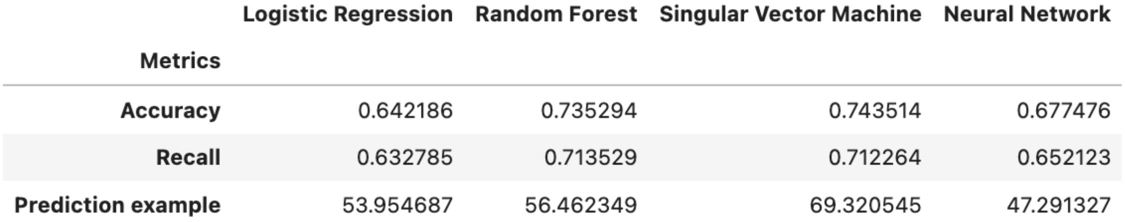
The testing data must then also be scaled separately to not have biases baked in.

4) Collinearity, or non-zero correlation among features, results in a model that is overly complex, reduces the statistical significance of the fit of the model, and prevents one from correctly identifying the importance of features. ***Are there features included in our machine learning models to predict rain in the Christman dataset that are collinear?*** If so, how do you think we should address this collinearity? A couple of suggestions: If we don't have that many features, we could use our meteorological expertise to simply remove one of the features that shares collinearity with other features. Another way to address collinearity is to use feature regularization, or add weights that penalize features that add noise, ultimately reducing model complexity.

All of our features would be collinear! All of temperature, RH and pressure would be greatly correlated with rainfall itself! And correlated with each other.

To address this collinearity, we could potentially remove our strongest collinear models with rainfall, e.g. relative humidity and pressure.

I removed dew-point temperature and relative humidity then re-trained all of our models. Our model skill dropped a ton when removing all the background information about



	Logistic Regression	Random Forest	Singular Vector Machine	Neural Network
Metrics				
Accuracy	0.642186	0.735294	0.743514	0.677476
Recall	0.632785	0.713529	0.712264	0.652123
Prediction example	53.954687	56.462349	69.320545	47.291327