**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***](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?**

**Chart, scatter chart

Description automatically generated** We see Summer is easily identifiable, but the other 3 clusters do not seem to correlate with the actual seasons. We saw in a scatter plot that the centers of two of the clusters were very close which is a sign that we have too many clusters.

**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?**

We see that temperature is the easiest variable to pick out in determining clusters. Rainfall was a poor indicator for the clusters as it is on a per-hour basis (i.e. it may rain once during a day in a rainy season.) Additioanlly, pressure seems to be a poor indicator of defining the clusters.

Wind speed, wind direction, and relative humidity do the best at defining cluster centers.

**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?**

One challenge with this method of clustering is that there’s no concept of memory in the classification. We see three different classifications of the season within a few days. Another range that would be interesting to check out is the flooding in 2013, but our data doesn’t go back that far.

**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?**

Using 3 clusters reduces the redundancy in our classification a bit, however, two of the clusters seem to overlap quite a bit. Reducing the cluster to two may be the best option as there are two very clear clusters with this scheme. One other option could be to preprocess the data to have better variables for classification. Some examples include “did it rain today” and the degree of variability in a data variable (i.e. huge day to day swings in winter temperature). f

**Notebook #2**

**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:**

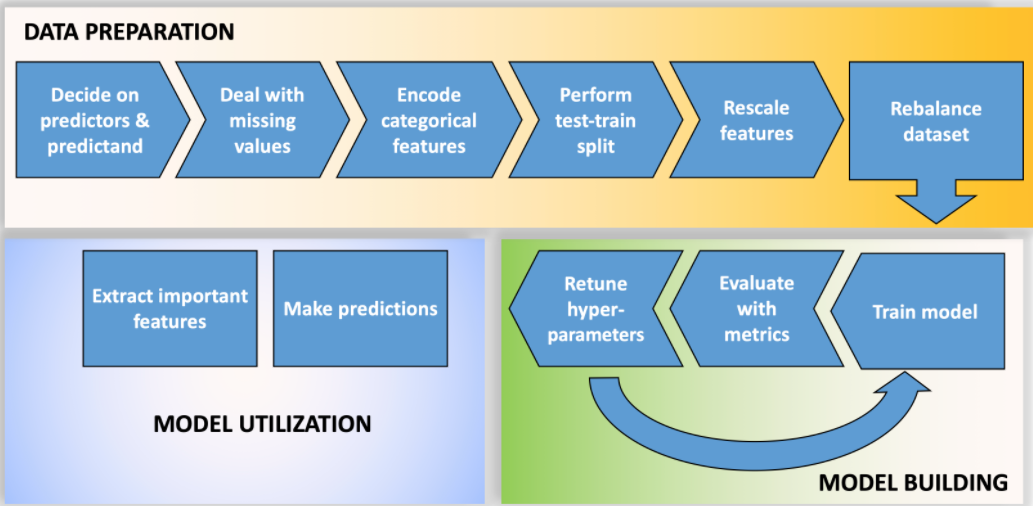
1. See an example of the data processing pipeline (workflow) required to utilize supervised machine learning techniques.
2. Implement and compare four different supervised learning algorithms
3. Understanding two outcomes of supervised learning algorithms: prediction and feature importance.
4. Start 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?

Random forest was the worst model on both accuracy and recall; SVM preformed slightly better on accuracy and had comparable recall to the other models. We selected this as the best model. We emphasize accuracy over recall as a model with 100% recall but low accuracy would not be very useful (i.e. saying it will rain every day).

1. 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 how well our model predicts the correct condition (raining or not raining). Recall is of the days where it was raining, how many days did we say that it was raining.

Prioritizing false positive vs false negative depends on the consequences. We prefer false positives when the cost of preparing for an event is minimal or the severity of the consequences of the prediction being correct are drastic. For example, if predicting a hurricane we’d rather recommend people to evacuate unnecessarily rather than not issue a warning and lose many lives.

1. 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 need to perform a train test split to ensure that our model isn’t overfitting to the training data. This gives us confidence that our model will perform well for future prediction. Along the same line, we should scale the test data after splitting because our future data may not explicitly come from the same distribution as our training data.

We also want to be able give our model to someone else with the same data and have them produce the same results. This can only happen if we normalize after splitting.

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.

We have collinear features in our dataset. For example, we saw that wind speed and maximum wind speed were very correlated. We could either remove one of these, use some dimensionality reduction such as EOFs, and overall try a different strategies and re-run the model.