Data cleaning (so far)

Good ideas

- .describe() – panda function that extensively summarizes data
- Keep units consistent (especially across data sources)
- Multivariate analysis checks connections between pairs of variables
- Check all rows have the same numbers of columns
- Normalized data storage to ensure consistency

Here in my garage… Domain knowledge

- Constraints
- Ranges
- Values and how they make sense within context
- Handling missing values
  - Should guide the decision process on how to go about any and all missing values
- Detecting cycles
  - As an example, consider a house’s temperature. The cyclitic nature of seasons and weather will play a role and thus should be removed from the data to obtain the thermostat changes independent of the weather outside.

Handling missing values

1. Forward/Backward-fill: assume the value is the same as the value before/after it

2. Local regression: averaging values (locally or globally)

3. Time sensitive regression: places more emphasis on time periods and historical data

4. Correlation imputation: using a correlated column to compute the missing value **NOTE** these columns are important to consider whenever missing values are filled

Benfords law

Leading digits in numbers are more likely to smaller. While this rule does not need to be strictly adhered to, data that does not follow this general trend may need further analysis

i.e
1 = 100 occurrences

2 = 20 occurrences

3 = 10 occurrences

4 = 10 occurrences etc…

Statistical outliers and overfitting data

Outliers can make a model too specific and overfit the data to the current data set (as opposed to generalizing trends). Outliers can be detected as follows:

Z scores: used when the data is normal (bell curve) a value greater than 3 or less than -3 is considered an outlier with the following equation

$$\frac{attribute\ value\ - average}{standard\ deviation}$$

IQR: used for multipeaked data. Outliers are the values larger than Q3 + 1.5(IQR) or less than Q1 – 1.5(IQR) respectively

- Q1 = (0.25)(n) - *round up if non whole number*
- Q3 = (0.75)(n) - *round up if non whole number*
- IQR = Q3 – Q1

Feature engineering

What is? Creating synthetic or derived data attributes with data aggregates.
For example:

- Bins/frequency distributions:
  - Simplifies granular data into percentages and distinctive categories such as a scale of 1 - 10
- Binarization:
  - Similar to Bins except that binarization deals with Booleans and their associated conditions
- Class imbalances:
  - Duplicating relevant data that may have low representational cases in the data set

Dealing with many dimensions

Unfortunately, data models can become less effective with too many dimensions. Enter Cart (R) and SciKitLearn (Python

Python snippet using breast cancer data:

```
1. load the data
cancer = load_breast_cancer()
X_raw = MinMaxScaler().fit_transform(cancer.data)          # data preprocessing
Y = cancer.target                                           # part of the dataset - the classification target

2. Transform the dataa
poly = dict()
X_poly = dict()

For n in [2, 3, 4, 5]:
        poly[n] = PolynomialFeatures(n) # Generate feature matrix of polynomial combinations with degree
```

NOTE: Polynomial features are those features created by raising existing features to an exponent   CONTINUED…

```
X_poly[n] = poly[n].fit_transform(X_raw)              # preprocesses data for future steps
```

3. reduce model dimensions to most relevant. NOTE Ranking of feature importance is not always relevant in specific models i.e k-nearest neighbors

```
# creates an arbitrary model
model = RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=4, random_state=2)
rfecv = RFECV(estimator=model, n_jobs=1)          # apply feature elimination/cross-validation to model
best_feat = rfecv.fit(X_poly[2], y)
X_support = X_poly[2][:, best_feat.support_]       # X_support now automatically holds the best subset
X_support.shape                                    # tells you the best dimensions to use
```

Testing

With regards to machine learning two different sources of fault exist: the data and the model

K-fold: data set is divide into an arbitrary number of portions. Excluding one subset for testing, all other portions generate the mode. Ideally, the omitted data yields satisfactory output from the newly generated model. This process generally repeats itself until every portion of data is tested. Next, all averages are summed thus returning the models overall effectiveness

Leave-one-out comparisons: similar to K-fold except that in addition, one data set is excluded entirely. The advantage of this method is that irrelevant data may show itself as once removed, the model's accuracy remains unaffected

Monte Carlo runs multiple iterations of tests in which data is divided randomly per run and the results are averaged

Monte Carlo has lower variance but is more biased. This trade off is common and called the the Bias-Variance tradeoff. In most cases, K-fold comparisons will be good enough and is computationally less expensive.