Lesson 16 – Feature Engineering:

**Questions for Mentor:**

**Encoding Categorical Data:**

* 7 types of data
  + Useless — useless for machine learning algorithms, that is — discrete
  + Nominal — groups without order — discrete
  + Binary — either/or — discrete
  + Ordinal — groups with order — discrete
  + Count — the number of occurrences — discrete
  + Time — cyclical numbers with a temporal component — continuous
  + Interval — positive and/or negative numbers without a temporal component — continuous
* Want to encode nominal and ordinal data
* Nominal data usually uses one-hot although not exclusively
* Ordinal data can be rank ordered
* Cardinality – a lot of unique values
* Ordinal data is usually used to encode with 1 through k then another encoder for nominal data
* One hot encoding
  + New column for each different value
* Binary
  + Combo b/w one-hot and hashing encoders
  + Fewere features created than one-hot
  + Works better with higher dimensionality data
  + Converts number of values into binary and adds that many columns
  + Spreads binary 1s and 0s across row
* BaseN
  + Base 1 = one hot
  + Base 2 = binary
  + Can use BaseN in sklearn gridsearchCV
  + Not useful
* Hashing
  + Fewer new dimensions than one hot

**Feature Engineering in Python:**

* Different types of data
  + Continuous
  + Categorical
  + Ordinal
  + Boolean
  + datetime
* df.select\_dtypes(include=[‘int’]) allows you to select only columns with a certain data type (int in this case)
* categorical features
  + encoding
    - one-hot
      * converts n categories into n features
    - dummies
      * dummies omits the base values because its encoded by the omission by all other columns
      * including all features can hurt your model (i.e. male female example of collinearity)
  + create a mask for categories that don’t occur often
    - syntax: mask = countries.isin(country\_counts[country\_counts < 10].index)
      * selects countries that occur less than 10 times in the data
* Numeric variables
  + Binary adjustments (i.e. restaurant violations)
  + Binning
    - Similar to binary adjustments but more than 2 bins
    - Pd.cut(column, bins= , labels= )
    - Bins can be a list of the boundaries
    - Labels is an equal sized list with the label added to the column
* Text Encoding
  + [a-zA-Z] : all letter characters
  + [^a-zA-Z] : all non letter characters
  + Can replace using .replace() method
  + Want to standardize the case of the text
  + Chain .str with the methods to manipulate/standardize
  + Text to columns
    - New column for each word and the counts associated
    - CountVectorizer creates a new column for words and their counts
      * Min\_df argument is minimum fraction of documents the word must occur in
      * Max\_df argument is minimum fraction of documents the word must occur in
  + Tf-Idf representation – normalizes word frequency
    - Term frequency – inverse document frequency
      * Max\_features argument limits features
      * Stop\_words argument omits stop words
    - Should inspect transforms for reasonability
    - Test data should be preprocessed using the same process as done on training data
    - ­­­­can analyze with multiple words to avoid misinterpretation using n-grams (bigram happy vs not happy example)
      * Syntax: ngram\_range= (n1, n2) argument in CountVectorizer for bi-gram
* Messy data and Missing values
  + Can’t always do list-wise deletion (much data lost)
  + Can fill with a certain value for categorical columns
  + For continuous columns – can use mean or median or other measure, but be careful as it can skew measurements (i.e. MSE)
* Other data Issues
  + Unwanted characters
    - Dollar signs in data making it object instead of numeric
    - Use string methods to fix
      * i.e. str.replace(‘,’, ‘’)
    - also convert to relevant type
      * .astype(‘float’)
    - Can also use .tonumeric(errors=’coerce’)
      * Coerce means it will change to numeric, if it can’t change to numeric, it changes to NaN
    - Can chain methods to perform action with one line of code
    - Use .apply() to cast a custom function when chaining methods
* Scaling data
  + Min max scaling – scale so that all values are between 0 and 1
  + Standardization – finds mean of data and centers distribution around it
  + Log transformation transforms features with highly skewed data (i.e. salaries or age of population
* Removing outliers
  + Quantile based detection
    - Remove X% of dataset (top 5%
  + Standard deviation based detection
    - Remove all values >X Standard devs from mean
* Scaling and transforming new data
  + Must transform test data using same scaler as with training data
    - Must use the fit on training data on test data

**Text Mining:**

* Process of deriving meaningful information from natural language text
* Natural Language Processing(NLP) is a part of computer science and artificial intelligence which deals with human languages.
* Tokenization – breaking down a sentence into pieces
* Can find frequencies of words occurring in a sentence
* Stemming gets to the root of a word (i.e. waiting, waited and waits stem from ‘wait’)
  + Often does not convert to base form (i.e. giv vs give)
* Lemmatization – process of converting word to base form
  + Converts to meaningful base form (i.e. giving would be give instead of giv)
* Stop words – common words (i.e. ‘is’, ‘the’, ‘a’ etc) and are dropped from text
* Part of speech tagging tags the part of speech of each word
* Named entity recognition can recognize if words are a name of a person, event, location etc
* Chunking is picking up individual pieces and grouping them into bigger pieces (i.e. ‘the’ ‘yellow’ ‘dog’ can be chunked into ‘the yellow dog’)

**Image Analysis:**

* Image feature extraction
  + Feature descriptor - It is a simplified representation of the image that contains only the most important information about the image.
  + Histogram of Oriented Gradients (HOG)
    - Focuses on shape of an object
    - Extracts edges as well as direction of edges
    - The HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an image
    - Create small (8x8, 16x16 etc) histograms of a part of the image which then allows us to analyze the whole image
* Image processing in Python
  + Purpose
    - Visualization
    - Image sharpening and restoration
    - Image retrieval
    - Measurement of pattern
    - Image recognition
  + Uses scikit-image in python
  + Often uses RGB layers
  + Grayscale uses only black and white and is based on intensity, not color
  + Numpy can be useful or analyzing images

**Automated Feature Engineering:**

* Biggest hurdle is the need for processed data
* Deep Feature Synthesis
  + Features derived from relationships between data points in dataset
  + Across datasets, many features derived by using similar mathematical operations
  + New features often composed from utilizing previously derived features
  + DFS offers a way to begin creating interpretable features for smaller datasets that humans can manually validate
* Featuretools API build features for any set of labels without requiring changes to the code
* Automated feature engineering enables data scientists to build better predictive models in a fraction of the time
* Uses primitives (mean, median, max, min etc) that are the building blocks for the feature engineering