Week 3: Text/Sentiment Analysis, Categorical Data, and Dates/Times

Introduction

Contents of the Week

Topics

- A Text processing
- B Sentiment analysis
- C Handling categorical data
- D Working with dates and times

Readings

Read the following:

- Chapters 5-7 of *Machine Learning with Python Cookbook*

Supplemental Materials

All of the materials below are from external sources. Authorship and ownership are indicated within the sources themselves.

Readings **Videos**

- What is natural language processing? Introduction to NLP
- Sentiment Analysis: A Definitive Guide
- TextBlob: Simplified Text Processing
- Sentiment Analysis using TextBlob
- Python | Sentiment Analysis using VADER
- Pandas Apply Function
- Handling Categorical Data, The Right Way
- How to Create Dummy Variables in Python with Pandas?
- Time Series/Date Functionality in Pandas
- How To Build Your Own Chatbot Using Deep Learning
- GPT-3 Powers the Next Generation of Apps
- Chatbot, Machine Learning and Artificial Intelligence in Pharmacovigilance: Maintaining Privacy, Optimizing Efficiency

3.1 Discussion/Participation

Here are optional topics for discussion via Teams this week. Remember, these topics aren't required, but if you are struggling to know what to post about, these can be used to initiate discussion!

1. What is NLP?
2. What is meant by sentiment analysis?
3. What are some prebuilt Python libraries to perform sentiment analysis?
4. Why is it sometimes necessary to preprocess text?
5. What is tf-idf vectorization?
6. What are some methods for dealing with categorical features when building a model?

3.2 Exercise: Sentiment Analysis and Preprocessing Text

Download the labeled training dataset from this link: Bag of Words Meets Bags of Popcorn.

Part 1: Using the TextBlob Sentiment Analyzer

1. Import the movie review data as a data frame and ensure that the data is loaded properly.
2. How many of each positive and negative reviews are there?
3. Use TextBlob to classify each movie review as positive or negative. Assume that a polarity score greater than or equal to zero is a positive sentiment and less than 0 is a negative sentiment.
4. Check the accuracy of this model. Is this model better than random guessing?
5. For up to five points extra credit, use another prebuilt text sentiment analyzer,

Submission Instructions

Click the title above to submit your assignment.

This exercise is due by Sunday 11:59 PM.

Submit your code, output, and answers at the link above. Comment all your code and answer any questions that are asked in the instructions. It is perfectly fine to answer a question by displaying output from your code, but make sure you are displaying the appropriate output to answer the question. I would recommend using and submitting a Jupyter Notebook, but this is not required.

View the rubric for this Assignment by clicking on the link below:

Exercise Rubric

e.g., VADER, and repeat steps (3) and (4).

Part 2: Prepping Text for a Custom Model

If you want to run your own model to classify text, it needs to be in proper form to do so. The following steps will outline a procedure to do this on the movie reviews text.

1. Convert all text to lowercase letters.
2. Remove punctuation and special characters from the text.
3. Remove stop words.
4. Apply NLTK's PorterStemmer.
5. Create a bag-of-words matrix from your stemmed text (output from (4)), where each row is a word-count vector for a single movie review (see sections 5.3 & 6.8 in the Machine Learning with Python Cookbook). Display the dimensions of your bag-of-words matrix. The number of rows in this matrix should be the same as the number of rows in your original data frame.
6. Create a term frequency-inverse document frequency (tf-idf) matrix from your stemmed text, for your movie reviews (see section 6.9 in the Machine Learning with Python Cookbook). Display the dimensions of your tf-idf matrix. These dimensions should be the same as your bag-of-words matrix.

Additional Comments

The bag-of-words and tf-idf matrix

- The bag-of-words and tf-idf matrices are stored as sparse matrices because most entries are zero.
- Each row in the bag-of-words/tf-idf matrices corresponds to a movie review.
- The columns in the bag-of-words/tf-idf matrices correspond to unique words appearing in the movie reviews.
- Entries in the bag-of-words matrix are the number of times a word appears in a review.
- Entries in the tf-idf matrix are numbers representing the word importance in a review.
- The bag-of-words/tf-idf matrices are possible feature (input) matrices for model building.
- We will revisit this preprocessed text data to build a custom model in the future.