Brief File Information

To fetch and load large datasets (activeReviewDf.pkl, activeUserDf.pkl, inactiveReviewDf.pkl, inactiveUserDf.pkl, test.pkl, train.pkl) stored remotely using git-lft, need to set up git-lft at https://git-lfs.github.com/

Parsing of the review and user data set from json to csv (dropped columns we think useless from json):
https://github.com/g1isgone/mlFinalYelp/blob/master/ParseUserJson.ipynb
https://github.com/g1isgone/mlFinalYelp/blob/master/ParseReviewJson.ipynb

Sync user data with review data (includes how we sample active users and inactive users and how we form training data and testing data):
https://github.com/g1isgone/mlFinalYelp/blob/master/SyncUserAndReview.ipynb

Text classification using naive Bayes model and Support Vector Machine:
https://github.com/g1isgone/mlFinalYelp/blob/master/ReviewClassification.ipynb

Preprocessing of the restaurant data set (includes detailed commentary and thought process regarding decision on how to narrow down the dataset):
https://github.com/g1isgone/mlFinalYelp/blob/master/restaurantsDataPreProcessing.ipynb

Decision Tree & Random Forest Models of Restaurant dataset (and some data preprocessing):
https://github.com/g1isgone/mlFinalYelp/blob/master/restaurantAnalysis.Rmd
https://github.com/g1isgone/mlFinalYelp/blob/master/restaurantAnalysisWriteUp.pdf

Clarification

We are doing two subprojects the Yelp dataset, one on user's reviews, another one on business, and do comparative analysis in the end. So far, we've finished all model building in both our subprojects, but haven't compared the results of the two projects.
More codes: https://github.com/g1isgone/mlFinalYelp

Subproject 1: User & Review Analysis

Introduction (3 points)

- A **bulleted list of changes** you've made to your model since FP2.
    - Update the measure of user activity: Abandon measuring user activity by review_count * 1000 / (most_recent + 1 - yelping_since) because it is confusing and make new Yelp users outliers. Instead, we just measure user activity by review_count
    - More data processing: sync users with their corresponding reviews
    - Modeling on user activity and review texts: In FP2, we only did linear regression on average_stars and activity. In FP3, we get to modeling on the correlation between activity and review texts with naive Bayes classifier and support vector machine.
- A brief **summary of the problem** (i.e. what is your question, where did your data come from, who is your audience, etc.).
    - Can we predict whether a user is active or not using the text of a user review
    - The data come from user.json and review.json from the Yelp dataset challenge https://www.yelp.com/dataset/documentation/json
    - The potential audience is Yelp and all business owners, who should be interested in increasing user activity.
- A clear description of your final **data preprocessing chain** (in this section, please **do not** include things you tried that ultimately didn't work out)**.** This may take the form of detailed comments in your code OR an annotated diagram.

Model description (4 points)

- Code: https://github.com/g1isgone/mlFinalYelp/blob/master/ReviewClassification.ipynb
- An **explanation of your modeling techniques**
    - Tf-idf (Term Frequency times Inverse Document Frequency): tokenize texts and count word frequency using scikit-learn's CountVectorizer and TfidfTransformer. The result is a sparse matrix that stores the weight of each word in each review. This technique can downscale weights for words such as 'is', 'a', 'are', that occur in many reviews and are therefore less informative than those that occur only in a smaller portion of all the reviews.
    - Naive bayes classifier: we started with sklearn's naive bayes classifier to provide a baseline for predicting the category of the user, because naive bayes classifier can usually be much faster than more sophisticated methods though it assume independence between every pair of features. Specifically, we used MultinomialNB, which implements the naive Bayes algorithm for data that have multinomial distribution
    - Support Vector Machine: we then tried SVM, because it's one of the classic text classification algorithms, though slightly slower than naive Bayes.
- A brief description of the **assumptions** the model makes - they all make at least one!
    - Since I choose the data i used for training and testing consist only of reviews that belong to users whose review_count>=90 or review_count==1, so they are very active users and very inactive users. If I have better machine, or can do parallel computing, I am able to train more users. Also, users with low review_count might not actually be inactive, they might have just joined Yelp very recently.
    - Naive bayes classifier assumes independence between every pair of features
    - Support vector machine is quite tolerant of the input data
- A description of what you've done to **validate whether that assumption holds** for your data.
    - We attempted to incorporative yelping_since, which is the date the user join Yelp in measuring user activity, but it is not performing well, so we are sticking to measuring activity just using review_count
    - Since naive bayes classifier's performance is not as good as SVM (though still not bad), so pairwise independence might not hold

- A **checklist** of what pieces of your model have already been built, and what is still in process.
  - Tf-idf, DONE! have built the tf-idf sparse matrix and visualize the result for a subset of active users and a subset of inactive users
  - Naive Bayes classifier: DONE building the model; no plan to visualize it because its performance is not as good as SVM
  - SVM: DONE building linear SVM but haven't tried other kernels. Haven't visualization of the result because PCA cannot deal with the size of the tf-idf matrix
  - Comparative analysis with the model of business data set is still IN PROGRESS, we will sync reviews of active/inactive users with restaurants analyzed in our second subproject using the business_id for each review. Looking to find trends among active/inactive users and distinctions between active users and inactive users, and compare the result with the result of tf-idf

Discussion (3 points)

- A brief defense of why your chosen technique makes sense.
  - Our chosen technique is SVM. SVM makes sense because theoretically it is very good at dealing with high dimensional data (dimension in our case is the number of unique words among all reviews). SVM doesn't have many assumptions on the input data, and turn out performing quite well both on the training set and test set.
- The pros and cons of the technique.
  - Tf-idf pros: clear and memory efficient; good for comparative analysis with the business data; filtered out many uninformative words; prepare the data for naive Bayes and SVM well
  - Tf-idf cons: not perfect, words such as we, he, she are not filtered out
  - Naive Bayes pros: fast, a good head start
  - Naive bayes cons:
  - SVM pros: effective in high dimensional spaces, even when the number of dimensions dimensions is greater than the number of samples; it uses a subset of training points in the support vectors so is memory efficient
  - SVM cons: do not directly provide probability estimates as decision tree model; hard to visualize and not very intuitive
- Regrets / future work (i.e. what are you **not** going to have time to do, even though it might be interesting)

    - Clustering on all users, not just classifying a subset of users (whom we already know are either very inactive, or very active) into two categories, might classify them into “cheap eats lovers”, “asian food lovers”, “service lovers”, etc
    - More user modeling -> so Yelp can provide more accurate searching results and advertisements/recommendations that cater to users’ interests

Subproject 2: Restaurant Analysis

Introduction (3 points)

- **Bulleted List of Changes since FP2**
    - We decided to focus just on the restaurant businesses
    - We decided not to do comparisons by states because we did not have a lot of observations for the restaurant data, instead we decided to use all the restaurant dataset
    - We decided to make our models in R instead of Python because it dealt with categorical data better and it was easier to order the levels of the factors
    - We were going to prune, but it was not possible because we had so few predictors and some of our models did not let us do so
    - We did 2 regression trees and 1 classification tree instead of making them all classification decision trees
- **Summary of the Problem**
    - We try to identify what components define a "successful" restaurant business. Our goal for exploring the restaurant business data is to use two types of models: decision trees and random forests to see if we can identify what are some potential factors that define a "successful" business. We planned to do this by identifying potential patterns that appear from the decision tree and random forest models. For example, we hypothesized that some factors may be indicated as factors of high importance for both the decision tree and random forests. If so, we further reasoned that those factors could potentially define what makes a restaurant business successful.

- **Data Processing Chain**
    - Please look at https://github.com/g1isgone/mlFinalYelp/blob/master/restaurantsDataPreProcessing.ipynb for more details
    - Extraction of Data

    - The raw data set had each of the line as a json, but the entire file was not in json format. Thus we needed to extract the json and convert it into a dataframe.
- Expanding DataFrame: Unnesting
    - A lot of our data is nested within each dataframe, particularly, the 'attributes', 'hours', and 'categories' columns, so those required unnesting
    - This required regrouping of the categories
    - We also unnested json for the ‘attributes’ and ‘hours’ columns
- Dropping Columns
    - We do this because some of the columns' data is unnested and thus repetitive and did not need as fine grain data (i.e. address)
- We then subsetted the businesses with Restaurants labels only to create a new dataset
- Dropping columns with more than 20% of Data Missing:
    - Determining which columns to delete was actually an iterative process. We continuously modeled and saw if we had enough data. From practice, it seemed that if there were more than 20% of the data missing, it was hard to create a comprehensible decision tree because it would split into Unknown too frequently to provide much meaningful data. Additional complications seemed to arise because we had too few observations such that having a 1/3 of its data as Unknown was hard to provide analysis on.
- Merging and Grouping Parking columns
    - We had four columns related to parking that were missing ~15% of the data, so we merged those together
- Filling missing data with “Unknown”
    - The rest of the dataset had at most ~15% of its observations missing per column. We filled those in with “unknown” label.
- Converting Data Type of “is_open”
    - We converted the binary 1/0 data type to "Open" and "Closed" for the is_open column. This was just for convenience when we export it to R so that R understands the 'is_open' column as one with categorical data types.
- Conclusively we have 12- 1 = 11 columns/properties that describe our restaurant data set from the Python Model
- Changing Factors to Numeric Data

        - "stars" and "review_count" are factor types in R initially. However, they are numbers on an ordinal scale and have a numeric ordering. Thus, we convert the data types of those columns to numeric.
    - Setting Orders to Factor Labels
        - The columns "BusinessAcceptsCreditCards", "RestaurantsPriceRange2", and "RestaurantsTakeOut" can be considered categorical data types. We extracted the columns above by manually looking through the labels of each columns.
    - Pre-processing in R
        - We eventually dropped the columns ‘X’, ‘index’, and all regional data: ‘business_id’, ‘city’, ‘neighborhood’, and ‘state’
    - The models were really run on cleanDf which had the following columns: is_open, review_count, stars, BusinessAcceptsCreditCards, RestaurantsPriceRange2, RestaurantsTakeOut, and Parking

Model description (4 points)

- An **explanation of your modeling technique(s)**.
    - Our modeling technique was an iterative process. The size of the dataset was chosen after we realized that we needed a few number of predictors to create a comprehensible model. We also made some models and went back to group some labels to produce models that made more sense. We chose to do both a decision tree and random forest modeling for the following reasons: (1) we wanted to identify what factors indicated a successful business (2) we wanted to see if we could corroborate those factors by comparing our results from decision trees and random forests. From here our plan was to expand the number of predictors manually after figuring out what models may be interesting.
- A brief description of the **assumptions** the model makes - they all make at least one!
    - We recognize that success can be determined by various categories. In our case we assumed that the success of a restaurant based on the yelp dataset could be objectively determined by "is_open", "stars", and "review_count". Our assumption was that successful restaurants most likely is still open, has good star ratings, and a lot of review counts.
- A description of what you've done to **validate whether that assumption holds** for your data

    - To see if our assumption was true, we looked at variable importance and decision trees for "is_open", "stars", and "review_count" and saw what predictors were important. We found out that predicting the review_count produced a comprehensible decision tree model. The review_count decision tree model seemed to indicate that restaurants that potentially have parking, are open, and have a "high" or above 1.0 price range have more reviews. This intuitively makes sense because restaurants that are successful probably have parking information, are still open, and have a "pricey" range.
- A **checklist** of what pieces of your model have already been built, and what is still in process.
    - Classification decision tree predicting whether a restaurant is still open or not (in operation) - built, but perhaps more analysis is needed
    - Regression decision tree predicting restaurants star rating - completed, but it is a bad model
    - Regression decision tree predicting restaurants review rating - completed, model is fine, but perhaps more analysis is needed
    - Random Forest tree predicting is_open - built, perhaps more thorough analysis
    - Random Forest tree predicting stars - built, but does not seem meaningful
    - Random Forest tree predicting review_count, built, perhaps more thorough analysis

Discussion (3 points)

- **A brief defense of why your chosen technique makes sense.**
    - We chose our techniques because we did not want to bias (although we did) the dataset and wanted to see if the machine itself could find out what the most important factors/features were for a successful business by doing models for 3 different columns
- **The pros and cons of the technique.**
    - **Pros**
        - It allows us to identify multiple potential important factors
        - Cross-comparison gives us higher faith in our correlation with the importance of some features
        - It allows the model to suggest some interesting features
        - It shows that some models are incorrect, for example the assumption that stars denote a good restaurant is questionable given the models we’ve made
        - The decision trees are easy to read
    - **Cons**
        - It is hard to analyze

        - It does not perform well with too many or too little observations and or columns
        - The models do not deal categorical and numeric data simultaneously well together
        - There is no conclusive causational relationship
- **Regrets / future work (i.e. what are you not going to have time to do, even though it might be interesting)**
    - I wish I did not have to spend so much time trying to learn both the Python packages and R. I had to switch very last minute to make the R models instead of Python ones so I have had a learning curve trying to code and that wasted a lot of time. If I knew what type of dataset I was dealing with and what models I were making and the limitations in both languages beforehand, I believe the transition may have been smoother.
    - I would like to add more columns to the dataset. We know that 6~7 predictors is not a lot of predictors. I wished we had more observations for restaurants only. I also wished we could explore more on the stars dataset and see if we can find anything more meaningful. We also wonder how some of the decision tree models, particularly the one regarding review_count will change if we take the unknown data out. We will also try and do cross analysis between the user dataset models and the business dataset models to see if we can make sense of any of our models. I also think we may need to do more analysis.

Comparative Analysis

We don’t have time to do this part yet...