University of San Diego®
SHILEY-MARCOS
SCHOOL OF ENGINEERING

Assignment 4.1 Exercises

For this assignment, you will conduct programming tasks in Google Colab and answer provided questions in a Google Doc or MS Word document. You will submit **2** documents to Blackboard:

1. A PDF file converted from Google Colab.
2. A PDF file converted from your Assignment 4.1 Short Answer Template Word Doc (document is linked in Blackboard assignment prompt).
    - This template has a copy of all short answer questions for this assignment.

Clothing Data
For this assignment, you will work with data from a popular clothing website, modcloth.com. This gives us a chance to work with feature engineering, feature extraction, and unbalanced, multiclass datasets. You will train several different classification models based on different types of data from this dataset.

As you go through the assignment, you will create several tables and figures. After you complete the programming section, use the tables and figures you generated to answer the following questions.

Programming Section <u>(Complete this section in Google Colab)</u>

<u>Pre-Processing and Loading Data:</u>
In your Blackboard assignment prompt is the modcloth_final_data.json file, which is a series of JSON objects from a popular clothing store, modcloth.com. Load these objects into a list, and then use pd.DataFrame to convert that list into a dataframe called modcloth_data.

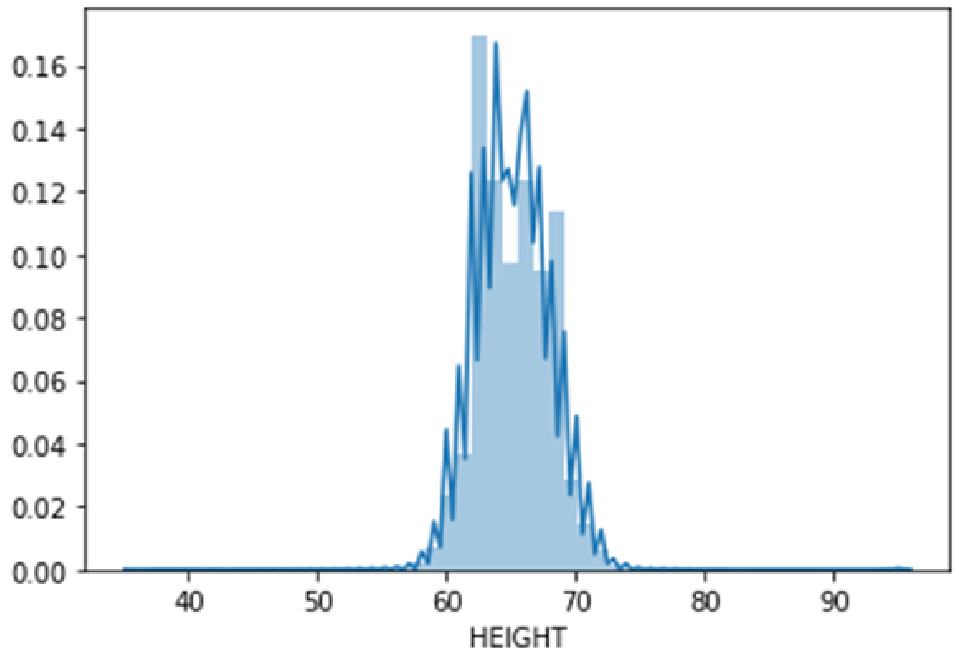
Create a variable named 'labels' using the 'quality' column. These are the labels that we will use for the rest of this assignment.

Next, you will transform these data into numerical vectors you can use for classification.

For the variables 'waist', 'size', 'hips', 'bra size', and 'shoe size' you can use `pd.to_numeric(modcloth_data[variable]` to convert them to numerical values. Store these in a data frame.

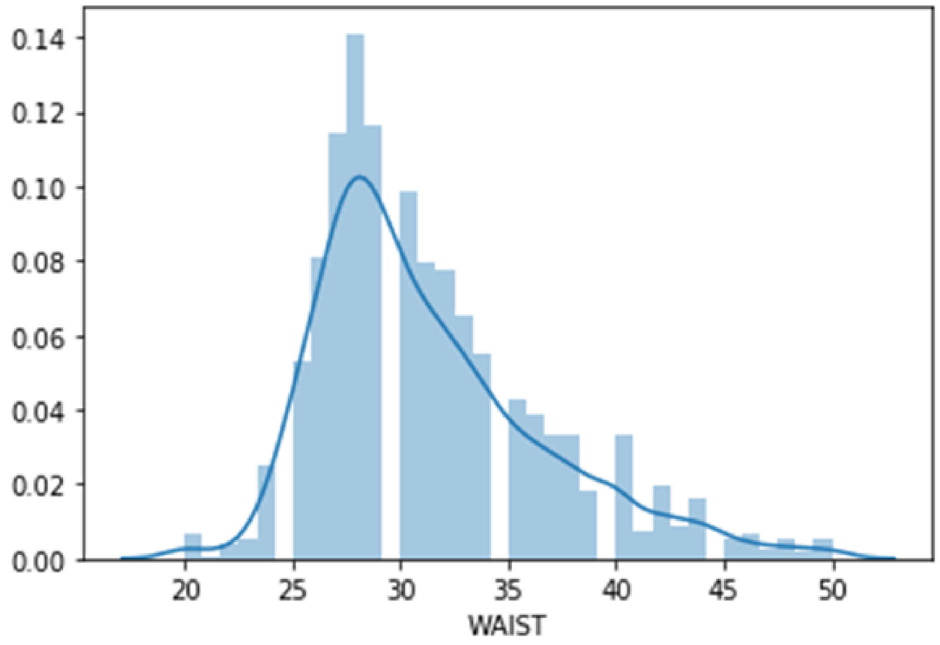
For the next variable, 'height', the strings need some processing to convert them into numeric values. You can use pd.apply to apply a transformation to the column of the data frame. Write a short python function to convert these to a meaningful categorical variable: convert the strings to height (in inches). Also, do this for the variable 'bust' - there are some outliers that could be handled by writing and applying another python function. Convert height and bust and store the numeric values to the same data frame as the other numeric features.

Plot the height using seaborn's distplot function. You should see a normal distribution of height, which should match your expectations for what the distribution of human heights should look like.



Make a plot of the waist data, which should look like this:

```
sns.distplot(new_data['WAIST'])
```



<u>Categorical Data for Reviews:</u>
For these values, ‘cup size’, ‘length’, ‘category’, use OneHotEncoder to convert these to a data frame called cat_feat.

Use make_pipeline to compare these two classifiers:

```
balanced_model =
make_pipeline(,Perceptron(class_weight='balanced'))
unbalanced_model = make_pipeline(,Perceptron())
```

Train a classifier using the data in cat_feat. Use train_test_split and test_size=0.50 to get a 50/50 train and test split.

Use metrics.plot_confusion_matrix and metrics.classification_report to report and visualize the confusion matrix and results. Perform this for both the balanced and unbalanced models.

<u>Categorial and Numeric Features:</u>

Next, you will add in data from the numeric features. Unfortunately, many of these values are missing: not every JSON object will have a value for this numeric value. You will use imputation as implemented in https://scikit-learn.org/stable/modules/impute.html to fill in the missing values before classification: use SimpleImputer.

Next, you will create three pipelines. Create one pipeline using SimpleImputer to impute the missing values and the balanced perceptron. Next, create a pipeline using preprocessing.Normalizer(). Finally, create a different pipeline using preprocessing.StandardScaler().

For each of the pipeline,I do fivefold cross-validation, using `scores = cross_val_score(model, X, y, cv=5,scoring='accuracy')` to compute the accuracy of each of the pipelines.

Create a table, with a row (labeled) for each pipeline you created. The table should report the maximum, minimum, and mean of accuracy for each of the pipelines.

Text Data from Reviews:

Finally, you will use the words in the consumer reviews to build a classifier to predict the results.

```
corpus = modcloth_data['review_text']
```

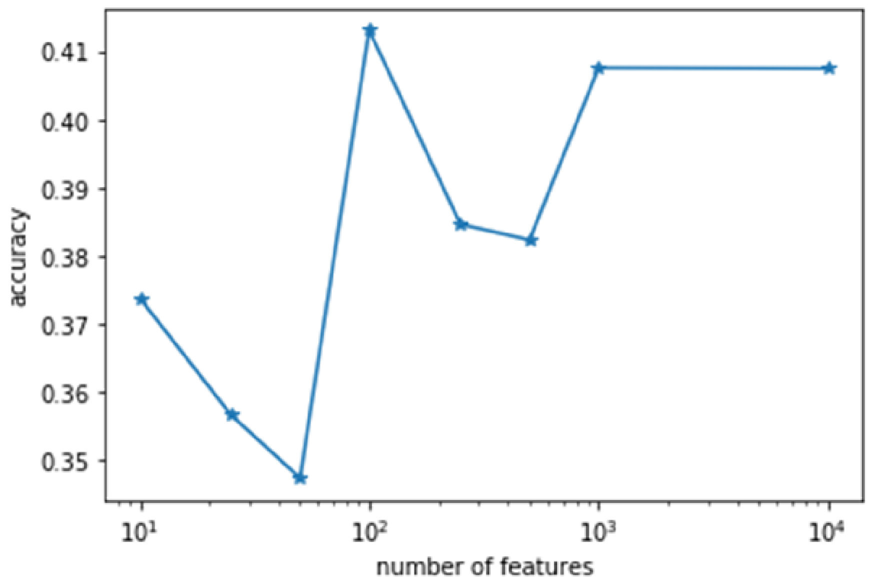
Create a new variable from the text data, after removing outliers, which are records that contain no text data (these will be nans in your data frame). This new variable is a list of strings, where every string is a consumer review.

Pass this new variable to sklearn'sTfidfVectorizer (https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction) to load the data. Create a new dataframe called X from the bag of words features, using pd.DataFrame.sparse.from_spmatrix, and subset the labels to only those which have text data present.

Next, use feature selection (https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection) to subset the text data to only the most relevant features. Be sure to do this based on the training set, then apply the transformation to the test set. Using the pipeline can make this step easier.

Use `SelectKBest` and chi-2 to select features. Make sure you have the latest version (at least 0.24 of scikit-learn, or you may get errors). Print a list of the top ten features selected. Hint: Use `vectorizer.get_feature_names(),XKBest.fit()` and then `XNew.get_support()`.

For k in the range [`10,25,50,100,250,500,1000,10000`], plot the accuracy of the classifier for each value of k. Your plot should look like the one below.



Written Questions **<u>(Complete these in the Assignment 4.1 Short Answer Template, located in your assignment prompt)</u>**

(Short answer, 2-3 sentences each):

- Using the confusion matrix and report in the Categorical Data for Reviews section, summarize the results you obtained using the balanced and unbalanced perceptron. Report it in terms of both f1 score and accuracy.
- Of all the methods used on the clothing ratings dataset, which of them performs best as a classifier? Would you recommend this system to predict product ratings?

(Short answer, 1-2 sentences each):

- In the clothing ratings, you plotted height and waist size (plots appear above). What transformation could be applied to the waist data, to make it look more like a normal distribution?
- You obtained a plot of accuracy vs. the number of words chosen for count vectorizer and tf-idf vectorizer (or refer to the plot in the last section of this document). How many features are used to obtain the best accuracy on this dataset?
- In the text classification section, you printed a list of the top ten words. What are they, and do you think these words make sense in the context of clothing reviews?