Homework 3 - DSC 478

For this assignment you will experiment with various regression approaches and you'll get your feet wet with some clustering. We will rely on subsets of some real-world data sets and on tools from the scikit-learn machine learning package for Python as well as modules from the textbook code (Machine Learning in Action, Chapters 8 and 10).

Check out the additional notes on this homework below.

1. [5 pts] Load and preprocess the data using Pandas or Numpy and, if necessary, preprocessing functions from scikit-learn. For this problem you do not need to normalize or standardize the data. However, you may need to handle missing values by imputing those values based on variable means. Compute and display basic statistics (mean, standard deviation, min, max, etc.) for the variables in the data set. Separate the target attribute for regression. Use scikit-learn to create a 20%-80% randomized split of the data. Set aside the 20% test portion; the 80% training data partition will be used for cross-validation on various tasks specified below.
2. [10] Perform **standard multiple linear regression** on data **using the implementation for Ch. 8 of MLA**. Compute the RMSE values on the full training data (the 80% partition). Also, plot the correlation between the predicted and actual values of the target attribute. Display the obtained regression coefficients (weights) and plot them using matplotlib. Finally, perform 10-fold cross-validation on the training partition and compare the cross-validation RMSE to the training RMSE (for cross validation, you should use the KFold module from sklearn.model\_selection). **Note** if you cannot get the book's version working, use scikit-learn Linear Regression instead for a 3pt deduction)
3. **[15 pts] Feature Selection:**  use the scikit-learn regression model from **sklearn.linear\_model** with a subset of features to perform linear regression. For feature selection, write a script or function that takes as input the training data, target variable; the model; and any other parameters you find necessary, and returns the optimal percentage of the most informative features to use. Your approach should use k-fold cross-validation on the training data (you can use k=5). You can use **feature\_selection.SelectPercentile** to find the most informative variables. Show the list of most informative variables and their weights [**Note:** since this is regression not classification, you should use [feature\_selection.f\_regression](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html" \l "sklearn.feature_selection.f_regression" \o "sklearn.feature_selection.f_regression) as scoring function rather than [chi2](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2)). Next, plot the model's mean absolute error values  on cross-validation relative to the percentage of selected features (See scikit-learn's [metrics.mean\_absolute\_error](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html" \l "sklearn.metrics.mean_squared_error" \o "sklearn.metrics.mean_squared_error) ). In order to use **cross\_validation.cross\_val\_score** with regression you'll need to pass to it **scoring='neg\_mean\_absolute\_error'**as a parameter. [**Hint:** for an example of a similar feature selection process please review the [class example notebook](https://d2l.depaul.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=859723&type=content&rcode=depaul-3332180). Also, review [scikit-learn documentation for feature selection](http://scikit-learn.org/stable/modules/feature_selection.html" \t "_blank).]
4. [10 pts] Next, using the original train-test split in part (a), perform **Ridge Regression** and **Lasso Regression** using the modules from sklearn.linear\_model. In each case, perform systematic**model selection** to identify the optimal alpha parameter. You should create a function that takes as input the data and target variable; the parameter to vary and a list of its values; the model to be trained; and any other relevant input needed to determine the optimal value for the specified parameter. The model selection process should perform k-fold cross validation (k should be a parameter, but you can select k=5 for this problem). For each model, you should also plot the error values on the training and cross-validation splits across the specified values of the alpha parameter. Finally, using the best alpha values, train the model on the full training data and evaluate it on the set-aside test data. Discuss your observations and conclusions, especially about the impact of alpha on bias-variance trade-off.  [**Hint:** for an example of a similar model selection process please review the [class example notebook](https://d2l.depaul.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=859723&type=content&rcode=depaul-3332180).]
5. [10 pts] Next, perform regression using Stochastic Gradient Descent for regression. For this part, you should use the **[SGDRegessor](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html" \t "_blank)** module from **sklearn.linear\_model**. Again, start by a creating randomized 80%-20% train-test split. SGDRegessor requires that features be standardized (with 0 mean and scaled by standard deviation). Prior to fiting the model, perform the scaling using **StandardScaler** from **sklearn.preprocessing**. For this problem, perform a grid search (using **GridSearchCV** from **sklearn.grid\_search**) Your grid search should compare combinations of two penalty parameters ('l2', 'l1') and different values of alpha (alpha could vary from 0.0001 which is the default to relatively large values, say 10). Using the best parameters, apply the model to the set-aside test data. Finally, perform model selection (similar to part d, above) to find the best "l1\_ratio" parameter using SGDRegressor with  the "elasticnet" penalty parameter. [**Note:** "l1\_ratio" is The Elastic Net mixing parameter, with 0 <= l1\_ratio <= 1;  l1\_ratio=0 corresponds to L2 penalty, l1\_ratio=1 to L1 penalty; defaults to 0.15.] Using the best mixing ratio, apply the Elastic Net model to the set-aside test data. Provide a summary of your findings from the above experiments.

For this problem you will use a different subset of the[**20 Newsgroup data set**](https://d2l.depaul.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=859723&type=content&rcode=depaul-3335889) that you used in Assignment 2  (see the [description of the full dataset](http://qwone.com/~jason/20Newsgroups/)). The subset for this assignment includes 2,500 documents (newsgroup posts), each belonging to one of 5 categories **windows** (0), **crypt** (1), **christian** (2), **hockey** (3), **forsale** (4). The documents are represented by 9328 terms (stems). The dictionary (vocabulary) for the data set is given in the file "terms.txt" and the full term-by-document matrix is given in "matrix.txt" (comma separated values). The actual category labels for the documents are provided in the file "classes.txt". Your goal in this assignment is to perform clustering on the documents and compare the clusters to the actual categories.

Your tasks in this problem are the following [**Note:** for the clustering part of this assignment you should use the [kMeans module form Ch. 10](https://d2l.depaul.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=859723&type=content&rcode=depaul-4984155" \t "_self) of MLA (use the version provided here as it includes some corrections to the book version). You may also use Pandas and other modules from scikit-learn that you may need for preprocessing or evaluation.]

1. [5 pts] Create your own distance function that, instead of using Euclidean distance, uses Cosine similarity. This is the distance function you will use to pass to the kMeans function.
2. [10 pts] Load the data set [**Note:** the data matrix provided has terms as rows and documents as columns. Since you will be clustering documents, you'll need to take the transpose of this matrix so that your main data matrix is a document x term matrix. In Numpy, you may use the "**.T**" operation to obtain the transpose.] Then, split the data set (the document x term matrix) and set aside 20% for later use (see below). Use the 80% segment for clustering in the next part. The 20% portion must be a random subset. Next, as in the previous assignment, perform TFxIDF transformation on these data sets.
3. [20 pts] Perform Kmeans clustering on the transformed training data from part (b) [**Note:** if you have difficulty with TFxIDF conversion, then use the original non-transformed data for the remainder of this assignment]. Perform a qualitative analysis of the clusters by examining top features in each cluster and identifying patterns in the data. To facilitate your analysis of the clusters, write a function to display the top N terms in each cluster sorted by the cluster DF values for each term, the centroid weights for each term in the top N terms in the cluster (mean TFxIDF weight of the term), and the size of the cluster. The cluster DF value for a term **t** in a cluster **C** is the percentage of docs in cluster **C** in which term **t** appears (so, if a cluster has 500 documents, and term "game" appears in 100 of those 500 documents, then DF value of "game" in that cluster is 0.2 or 20%).  Here is [an example of how this output might look like](https://d2l.depaul.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=859723&type=content&rcode=depaul-4541767) (here the top 10 terms for some clusters and the data are displayed in decreasing order of cluster DF values).

**Important Note:** for this problem you should try several values of k for the number of clusters (try values of k from 4 through 8) and in each case try several runs in order to obtain clusters that seem more meaningful. In some cases, you may find some small clusters containing noise documents, which is not unusual. The point is to experiment with different runs and cluster numbers until you find at least several clusters that seem to capture some of the key topics in the documents. You do not need to provide the results of all your runs; you should only provide the results of your best clustering along with a brief discussion of your experimentation and your final observations.

[**Extra Credit - 5pt:** use your favorite third party tool, ideally with a Python based API, to create a word cloud for each cluster.]

d. [5pts] Using the cluster assignments from your Kmeans clustering and the original cluster labels for the training document, compare your clusters to the re-assigned classes by computing the***Completeness***and ***Homogeneity***values. You should do this for the best values of k and the best clustering run you settled on in the previous part.

**[Extra Credit - 5 pts:** You can try other clustering runs with values of k between 4 and 8 and in each case compute Completeness and Homogeneity. This experiment will indicate which clustering provides the best representation of the original newsgroup categories.**]**

e. [10 pts] Finally, using your cluster assignments as class labels, categorize each of the documents in the 20% set-aside data into each of the appropriate cluster. Your categorization should be based on Cosine similarity between each test document and cluster centroids. For each test document show the predicted class label as well as Cosine similarity to the corresponding cluster.