Assignment 3

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).

---

1. **Linear Regression** [**Dataset**: **communities.zip**]

For this problem you will experiment with linear regression models to make predictions with numerical data. You will also explore more systematic methods for feature selection and for optimizing model parameters (model selection). The data set you will use is a subset of the "Communities and Crime" data set that combines information from the 1990 census data as well as FBI crime data from 1995. Please read the **full description of the data**, including the description and statistics on different variables. The target attribute for regression purposes is "ViolentCrimesPerPop". The two identifier attributes "state" and "community name" should be excluded for the regression task.

Your tasks in this problem are the following [**Note:** for these tasks you will use the available linear-models from scikit-learn as well as the implementations of the relevant approaches from the Ch. 8 of MLA] .

a. Load and preprocess the data using Pandas or Numpy and, if necessary, preprocessing functions from scikit-learn. The provided data is already normalized (see **description**), so there is no need for additional normalization. Compute and display basic statistics (mean, standard deviation, min, max, etc.) for each of the variables in the data set. Separate the target attribute for regression.

b. Perform **standard linear regression** on data using the implementation for Ch. 8 of MLA. Compute the RMSE value on the full training data. Also, plot the correlation between the predicted and actual values of the target attribute. Display the obtained regression coefficients (weights). Finally, perform 10-fold cross-validation and compare the cross-validation RMSE to the training RMSE (for cross validation, you should use the KFoldmodule from `sklearn.cross_validation`).

c. 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** as scoring function rather than **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**). In order to use `cross_validation.cross_val_score` with regression you'll need to pass to it `scoring='mean_absolute_error'` as a parameter. Review **scikit-learn documentation for feature selection**.

d. Next, 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. First, create a 20%-80% randomized split of the data. Set aside the test portion; the model selection process should be performed using the 80% training data partition. 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). 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` value, run the model on the set-aside test data. Discuss your observation and conclusions.

e.Next, perform regression using Stochastic Gradient Descent for regression. For this part, you should use the **`SGDRegessor`** 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.

2.**Automatic Document Clustering** [**Dataset**: **newsgroups5.zip**]

For this problem you will use a different subset of the **20 Newsgroup data set** that you used in Assignment 2 (see the **description of the full dataset**). 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** 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.]

a.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.

b.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.

c.Perform Kmeans clustering on the training data. Write a function to display the top N terms in each cluster along with the cluster DF values for each 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%). Sort the terms for each cluster in decreasing order of the DF percentage. Here is **an example of how this output might look like** (here the top 10 terms for 3 of the 5 clusters are displayed in decreasing order of cluster DF values, but the mean frequnecy from the cluster centroid is also shown). [**Extra Credit:** use your favorite third party tool, ideally with a Python based API, to create a word cloud for each cluster.]

d.Using the cluster assignments from Kmeans clustering, compare your 5 clusters to the 5 pre-assigned classes by computing the Completeness and Homogeneity values.

e.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.

Notes on Submission: You must submit your Jupyter Notebook (similar to examples in class) which includes your documented code, results of your interactions, and any discussions or explanations of the results. Please organize your notebook so that it's clear what parts of the notebook correspond to which problems in the assignment. Please submit the notebook in both IPYNB and HTML formats (along with any auxiliary files). Your assignment should be submitted via D2L.