MCIS6273 Data Mining (Prof. Maull) / Fall 2020 / HW2

This assignment is worth up to 10 POINTS to your grade total if you complete it on time.

Points Possible	Due Date	Time Commitment (estimated)
10	Monday, November 9 @ Midnight	*up to* 20 hours

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by University or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

OBJECTIVES

- Perform unsupervised learning with nearest-neighbors classification and regression
- Perform Bayesian text classification

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, make a directory in your Lab environment called `homework/hwN`. Put all of your files in that directory. Then zip that directory, rename it with your name as the first part of the filename (e.g. `maull_hwN_files.zip`), then download it to your local machine, then upload the `.zip` to Blackboard.

If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using zip in Linux.

If you choose not to use the provided notebook, you will still need to turn in a `.ipynb` Jupyter Notebook and corresponding files according to the instructions in this homework.

ASSIGNMENT TASKS

(50%) Perform unsupervised learning with nearest-neighbors classification and regression

In the last homework we work with the diamonds dataset and learned a lot about how to understand and visualize some of the features of the data.

We also learned that maybe the price of diamonds is only partially due to the rated characteristics of the diamond. There ar surely more that meets the eye when pricing diamonds and there are qualitative as well as quantitative measures which factor into the price of a diamond.

We're going to continue with this dataset on two routes. We're first going to build a classicification model based on the nearest neighbors in the data. That is to say, given an arbitrary unseen diamond, we would like to have a model trained from the actual data which gives us a class label for that diamond.

For the sake of the exercise, we're going to choose 5 labels for the diamonds all based on price. Instead of arbitrary cuts on those labels, we're going to use the 2015 US Federal income tax brackets as our guide.

If you look at the brackets, there are distinct cutoffs which could serve as bins for our class labels. We don't care necessarily what the labels are called, so we will just use letters 'A', 'B', 'C' and so on.

Once we have obtained the labels (and assigned them), we will then use the `sklearn.neighbors.KNeighborsClassifier` to develop a model for classifying the data. Remember from our lectures we need to split the data into test, training/validation and a *holdout* set. The training set will be used to train the model, while the test set will be used to test the model. The holdout set will be our "unseen" data upon which real classification decisions will be

made. We can evaluation the fitness of the final model on the holdout set and (hopefully) gain confidence that the model will do well in the real world.

§ Create labels for the data based on the first 5 tax brackets from the 2015 US Federal Income brackets. You only need to consider the *Single* 10%, 15%, 25%, 28% and 33% brackets which ends at $411,500. **Do not use the married or head of household brackets**. The full bracket information can be found here: https://www.bankrate.com/finance/taxes/2015-tax-bracket-rates.aspx.

1. For each bracket compute a value φ which is 10% of half the difference between the top and bottom of the bracket. For example,

$$\varphi_c = \frac{b_{max} - b_{min}}{2} \times 0.10$$

.

So if the top of a bracket $b_{max} = 5000$ and the bottom of the bracket $b_{min} = 2500$, then $\varphi = 125$. This number will represent the cutoff of the class, so that the range for the class extends from the previous class to the current one just calculated. The classes will ultimately look like this $C_1 = [0, \varphi_{c_1}], C_2 = (\varphi_{c_1}, \varphi_{c_2}], \ldots, C_n = (\varphi_{c_{n-1}}, \varphi_{c_n}]$.

2. Now label the original data from your diamonds data with the labels 'A' ... 'E' where 'A' is the first class, 'B', the second and so on. You will need to create a function that checks the class ranges for the price and applies that function accordingly to produce a new feature `class`. You can do this very simply with the `apply` function.

3. Store the labeled data (the new DataFrame with the class) into a file for all rows.

§ Now that we have the dataset, let's create a training and test set and beging model building. The easiest way to create your holdout set is to take a random sample of 25% of the data and use that as the holdout. You can can take the remaining 75% of the data and use it for the test/train split. While there are not steadfast rules for how much data you should use to train, test and validate with, a good rule of thumb is to train on no less than 20% of the data, if possible, and validate on between 20-25% of the data. If you have a large enough dataset (millions of instances), you may be able to train on 10% of the data. Of course, your choice of classifier may also help guide the decision on splitting data.

1. Create a 25% validation set and use the remaining data to do a 30%/70% test/train split. We have enough instances to create a dataset that should yield good accuracy. The `sklearn.model_selection.train_test_split()` is a great place to start and will save you a great deal of time.

2. Store the data into 3 files `test.csv`, `train.csv` and `validate.csv`.

§ Now that we have data that we need, we can perform the classification task we originally wanted. Recall, the goal is to make sure we are able to train a model that is capable of taking arbitrary unseen data and classifying it with a high degree of accuracy.

We will use the KNearestNeighbor classifier which finds a predefined (k) number training samples and produces a label prediction from those the majority agreement of the k votes. While the k does not have to be user-defined as in the case of radius-based neighbor learning, we will stick with a predefined K of 10. While considered a *non-generalizing* machine learning technique, it can be very successful in cases where class boundaries are irregularly shaped. You can learn more about the `sklearn` tools for nearest neighbors here: https://scikit-learn.org/stable/modules/neighbors.html#neighbors.

You will train, test and validate in this part of the assignment.

1. Write the code in your notebook to build the classifier using the `KNeighborsClassifier`. Here are a few tips:

- make sure you split the class labels correctly (i.e. do not include them in the feature set expected in the X parameter of the `fit()` method).

2. Use the `score` method to test your model's accuracy. Your notebook must include an individual assessment of the score, which is the mean accuracy of the `predict` method over the test data.
3. Please comment on the accuracy of your model. Would you trust it in the wild? Why or Why not? If the accuracy is not what you would think is worthy, please go back and make updates to the k parameter which

you have control over in the previous step. When you have found a k that is satisfactory, go to the next step (NOTE: this step may not be necessary, if for example, your classifier achieves > 0.80 accuracy).

§ Evaluating and reflecting on your model is an important and relevant step to take when building them. Answer the following questions:

1. Now that you have a model and may be fairly happy with it, please score the data on the holdout set. Make sure you have a cell in your notebook that shows the accuracy of the model on the holdout.
2. Do you think the your model is ready for prime time? Why or why not?

§ We spent time early on *creating* classes for our model. But with the diamonds what we really might like to have is a model that can give us a price, given the input features. While humans still control much of the evaluation and pricing process of the diamond business, machines might still be useful in situations where fast decisions might be made on less expensive diamonds or where margins may be so large that even suboptimal pricing is acceptable.

Using what we learned before, we're going to build upon the same concepts except this time, we won't need the labels. In building a *regressor* instead of predicting discrete class labels, we are going to predict a *continuous value* or in our case, the one that matters most: *price*. Like the classifier the regressor will make predictions based on the mean value of the k neighbors.

1. You will adapt your code above to produce the regressor (it will be very easy, instead of the class label as the input to the y parameter of `fit`, use the price, of course making sure you drop it and the label from your X input parameter.

To complete this study the `sklearn.neighbors.KNeighborsRegressor`. Use the same splits as the classifier above.

2. The `predict` method works similarly as before except this time it returns an R^2 value of the prediction – recall the closer R^2 is to 1.0, the better the fit. Make sure you do the prediction on the test data. Play with the model a bit if necessary to determine the impact of changes on the score.
3. Take the validation set and again determine the score. Do you feel the model is ready for prime time? Please be specific on why you feel that way?
4. Using your model, please take the sample file hw2_price_test_sample.csv from the course repo and produce prices for each input instance. The output will be a file with the price as a feature (output column in your file). Make sure you name the file `price_predictions.csv` so I can look at it.