MCIS6273 Data Mining (Prof. Maull) / Fall 2017 / HW3

This assignment is worth up to 20 POINTS to your grade total if you complete it on time.

Points Possible	Due Date	Time Commitment (estimated)
20	Monday, Nov 6 @ Midnight	*up to* 12 hours

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

OBJECTIVES

- RESUBMIT HW2 PARTS 2 AND 3 (OPTIONAL): Resubmit HW2 if you did not complete it
- BUILD A DECISION TREE CLASSIFIER: Work with Pandas and Scikit to build a decision tree classifier on a real world dataset
- BUILD A BAYESIAN CLASSIFIER: Work with Pandas and Scikit to explore the statistical features of a real world dataset using Bayesian classification

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, clone the course repository and modify the `hw3.ipynb` file in the `homework/hw3` directory. 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 Github and cloning repositories.

Turn in a copy of a `.ipynb` file, a PDF or Word Document to Blackboard with the code or answers to the questions /tasks labeled with the § sign.

ASSIGNMENT TASKS

(0%) RESUBMIT HW2 PARTS 2 AND 3 (OPTIONAL): Resubmit HW2 if you did not complete it

You can resubmit HW2 (parts 2 and 3 only) if you did not complete it. If you completed it and would like to resubmit a revised version of your homework, you are free to do so. If you resubmit part 1, you will incur the usual penalty, while there is **no** penalty for parts 2 and 3.

§ Review HW2 and resumbit your work if necessary.

(50%) BUILD A DECISION TREE CLASSIFIER: Work with Pandas and Scikit to build a decision tree classifier on a real world dataset

Continuing our work with classification, we're going to work with decision trees and use the decision tree classification tools in `sklearn`. As we talked about in lecture, decision trees are highly desireable for their interpretability. It is often very easy to understand how a classification is performed and it is also very easy to use features of mixed types and avoid **some** (though not all) of the pre-processing work we labored through in the prior exercise, where data preparation was paramount.

The dataset for this task is adapted from yet another a Kaggle dataset that provides a large number of details about real diamonds for sale. The dataset is nearly 54K rows of data with 10 features, but I have enhanced it for this assignment, so you will use the file on Github for the actual task I am asking you to do in this assignment.

If you are unfamiliar with the features of diamonds please read this very informative article from the Gemological Institute of America (GIA), which should introduce you to some of the details of the attributes – though understanding all these details is certainly not critical for completing the exercise, and is merely for your personal edification.

You are a data scientist for the diamond company, Sparkling Gems Inc., and you are trying to sell diamonds to one of the most prolific buyers on the market, Mr. Dirk Diamond. Mr. Dirk buys and sells millions of dollars worth of diamonds and if we know what his purchasing criteria are, we might be able to improve our sales to him and hence our own profitablity. Someone in the industry has found a notebook of thousands of Dirk's purchasing decisions in the last few years which include the details of diamonds he has accepted and rejected for purchase.

What we believe is if we can replicate his `ACCEPT` and `REJECT` criteria, then we might be able to find diamonds within our own inventory to sell to Mr. Dirk. To determine if we can capture his purchasing rules, we have 2000 diamonds that he has accepted and rejected (roughly split down the middle).

You are to take those decisions he's made and build a binary decision tree classifier. Split the data into a test and training set, build the classifier from the training set and test it on the remaining test set.

§ Load the dataset of 2000 diamonds into a DataFrame and build a training set of 1400 diamonds. Leave the remaining diamonds alone as a test set – you will notice we're going to use a 70/30 split. You can save a lot of time by using `sklearn.model_selection.train_test_split()`, but you are free to write you own code to do the same. You will need to eliminate the column `0\: Unnamed` as it contains an index value from the original file that you will not need.

§ Use the `sklearn.tree.DecisionTreeClassifier` class to build a decision tree classifier. You can use the defaults for now (e.g. you can keep the default `criteria='gini'`). To do this, you will need to use the following pattern:

- take the training set DataFrame and split into a *features* and *labels* DataFrame, where features is the $n \times m$ matrix where m is the number of features and n the number of training samples, and *labels* is an $n \times 1$ matrix of just the label for the corresponding sample.
- use the decision classifier with the *features* as the `X` parameter and *labels* as the `y` parameter
- **NOTE:** unfortunately, and as you learned in prior tasks, `sklearn` does not support text labels and so you will need to convert the `color`, `clarity` and `cut` variables to numeric (ordinal) variables. The easiest way to do this is with `sklearn.preprocessing.LabelEncoder().fit_transform()`

§ Use the `sklearn.metrics.classification_report()` to build the classification report that shows the precison, recall and f1-score of the classifier.

§ Draw the decision tree for this classifier. To do this you will need to install *graphviz* in Anaconda. Please follow instructions found on this page. To draw the tree, once the package is installed, use sklearn.tree.export_graphiz as shown in the example on the instruction page. Make sure the tree shows up in your notebook and also export the file to PNG so I can have a separate digital copy of it. Make sure you attach this PNG to your submission.

§ Do you feel you have built a good classifier? Why or why not? Please use the evidence from the prior questions when answering this question.

§ Now that you have a classifier for Mr. Dirk, use the file `./data/sgi_diamond_inventory.csv` to classify Sparkling Gem's inventory. Find all the diamonds that are classified `ACCEPT` by your classifiers. You are free to rework the parameters of your decision tree if you think you can reduce the classification errors.

- export the data to a CSV file called `mr_dirk_inventory.csv` and make sure you attach this to your final submission.

- what is the *gross total sale amount* if we were to sell all these diamonds to Mr. Dirk?

§ **[OPTIONAL BONUS]** *Completing this question will earn you 2 bonus points.* Pricing diamonds is tricky business, but Mr. Dirk is always one step ahead of most sellers. He's very astute and most often likes to buy diamonds according to Tavernier's law; see this resource for details. Effectively, let C be the *mean price* of a 1 carat diamond, then Tavernier's Law (τ) is calculated as

$$\tau = W^2 \times C$$

where W is the weight of the diamond in carats. Please **provide code and evidence** for whether you think your diamond prices are more or less attractive to Mr. Dirk.

(50%) BUILD A BAYESIAN CLASSIFIER: Work with Pandas and Scikit to explore the statistical features of a real world dataset using Bayesian classification

In this part of the assignment you will build a relatively simple Bayesian classifier. The data we will be using comes from a subset of a large US Department of Transportation dataset of over 7M records which you can optionally explore here. The data represents traffic volume from various places around the US, captured hourly for each day of data collection. The dataset we have is a sample of less than 10% of the original data, and we'll restrict that further to capture just over 100K data points.

The DOT needs to understand traffic patterns, but in particular wants to understand the traffic patterns in urban areas. What they'd like to do is build a classifier that, given a series of volume counts, can give the day of the week. In other words given a data point representing the volume counts by hour, provide the day of the week. The primary reason for building such a classifier is that some of the station data is not collecting day of week or has become unreliable. Thus, they plan to use the reliable data to classify the missing and unreliable data as a quality control check. They will keep the classifier that cam achieve results above 0.80.

Recall, the setup of our Bayesian problem, we want to find the classifier which given data d about the volume counts, we can produce the day of the week, C, or:

$$\Pr(C \big| d_1, \ldots, d_j) = \Pr(C) \prod_{i=0}^{n} \Pr(d_i \big| C)$$

where C is the day of the week and d_j the traffic volume counts for the given time window in 1 hour increments over 24 hours. Now we talked about how Naive Bayes works best in the context of discrete features, but `sklearn` provides a Bayes classifier over continuous features assumed to be Gaussian. This assumption, while not strictly true, can be practically used in large datasets. We will be using this classifier in the assignment. Our probabilities are then computed as such:

$$\Pr(d_i \big| C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\big(- \frac{(x_i - \mu_C)^2}{2\sigma_C^2} \big)$$

To perform this task will require you to train the classifier on some labeled data, test it, then use the provided unlabelled set to classify the day of week on unseen data. The final class, $\hat{C}$ is then computed as :

$$\hat{C} = \text{argmax}_C \Pr(C) \prod_{i=0}^{n} \Pr(d_i \big| C).$$

§ Prepare the dataset that is to include **only** the traffic volume data where the `functional_classification_name` is `Urban: Principal Arterial - Interstate` since we want to make classifications **only** on urban interstate data. Use the file `data/dot_testtrain.csv`.

§ Plot the **bar chart** of traffic volume by day of week for the following three times `traffic_volume_counted_after_0700_to_0` `traffic_volume_counted_after_2000_to_2100`, `traffic_volume_counted_after_1500_to_1600`. That is group by `day_of_week` and plot the data in a single chart using `DataFrame.plot(kind='bar')` as in prior homeworks.

§ Build the classifier over the `data/dot_testtrain.csv`, creating a 70/30 split of the data as in the prior exercise. You will need to use the `sklearn.naive_bayes.GaussianNB`. Train the data on 70% of the samples and test on the remaining 30%.

- What is the overall classifier accuracy? Show your code.

§ Now that you have a classifier, load the data in `data/dot_validate` and run your classifier on the data. Add the class label back to the unlabeled DataFrame onto a column `nb_classifier_label` and save the file back as `dot_validate_labeled.csv`. Make sure you attach the file to your submission.

§ The DOT stakeholder would like to know whether you achieved the desired result of 0.80 accuracy. Did you achieve this with your classifier? Explain with the appropriate level of evidence. Produce a classification report and show the precision, recall and f1-score as well.

§ [**OPTIONAL BONUS**]: *Doing this bonus can earn you up to 5 extra points varying on completeness and correctness of your effort.* Train the classifier to classify `functional_classification_name`. That is, the class you're trying to predict is the `functional_classification_name`. You are free to add back in the `day_of_week` should you find it useful and you should add back in all the other data (use the full data file and not just the urban data as above). You will, of course, need to remove `functional_classification_name` as it is the class you're trying to predict.