University of Notre Dame, Computer Science and Engineering

CSE 40647/60647 Data Mining — Assignment 3
Due Date: April 4th, 2014 at 11:59pm ET

Classification and Regression

March 25, 2014

This assignment will require you to implement and interpret some of the classification and regression concepts that were introduced in class. **An IPython Notebook with predefined functions to assist you with this assignment is available here. Additionally, the images used in this assignment can be downloaded here.** Keep in mind that the main objective of this assignment is to highlight the insights that we can derive from applying these techniques—the coding aspect is secondary. Accordingly, you are welcome to consult any online documentation and/or code that has been posted to the course website, so long as all references and sources are properly cited. You are also encouraged to use code libraries, so long as you acknowledge any source code that was not written by you by mentioning the original author(s) directly in your source code (comment or header).

You are expected to submit a single IPython Notebook file following the same instructions and naming convention described in Assignment 0. Answers to the conceptual questions can be embedded in the Notebook as *markdown* cells, and you may use *heading* cells to further organize your document.

1 Regression (50 points)

The Data
For this portion of the assignment you will be using the *Iris Flower dataset*, available here.

The dataset consists of 50 samples from each of three species of Iris (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

The Idea: Using Linear Regression on the Iris Dataset
Often, we may want to predict one feature based upon other features. Your objective here will be to generate a linear model of one of the features (a continuous variable) in the Iris dataset using one or more of the remaining features and/or class values. In our case, we're interested in finding the best linear model among those that can be generated from this dataset.

What to Do
First, load the Iris dataset. This can be done using the following snippet of code:

```
import pandas as pd
fileURL = 'http://archive.ics.uci.edu/ml/\
machine-learning-databases/iris/iris.data'
iris = pd.read_csv(fileURL, names=['Sepal_Length', 'Sepal_Width', \
                                   'Petal_Length', 'Petal_Width',
                                   'Species'], header=None)
iris = iris.dropna()
```

Next, you can visualize the correlation between different features using the following snippet of code, which executes the provided `pairs` function:

```
pairs(iris)
```

Some pairs of features tend to be more correlated than others. Try to uncover related features by using linear regression to model the relationship between pairs of features. In other words, use one feature as a target (dependent) variable and another feature as a predictor (independent) variable. To generate a linear regression model, you may use the `linear_model.LinearRegression()` function available via the scikit-learn library. To run the model on the Iris data, first divide the dataset into training and testing sets, then fit the model on the training set and predict with the fitted model on the testing set. scikit-learn provides several functions for dividing datasets in this manner, including `cross_validation.KFold` and `cross_validation.train_test_split`.

Several statistics can be generated from a linear model. Given a fitted linear model, the following code outputs the model coefficients (the parameter values for the fitted model), the residual sum of squares (the model error), and the explained variance (the degree to which the model explains the variation present in the data):

```
# The coefficients
print "Coefficients: \n", regr.coef_
# The mean square error
print ("Residual sum of squares: %.2f" %
        np.mean((regr.predict(iris_X_test) - iris_y_test) ** 2))
# Explained variance score (1 is perfect prediction)
print ("Variance score: %.2f" % regr.score(iris_X_test, iris_y_test))
```

You can use these scores to measure the efficacy of a particular linear model.

What to Provide
Your output should contain the following:

- A scatterplot matrix of scatterplots, with one scatterplot for each pairwise combination of features.
- A plot of the linear regression models generated on each pairwise combination of features, with corresponding model statistics.
- A plot of the best overall linear regression model you were able to generate using any combination of features (including the use of multiple features used in combination to predict a single feature), with corresponding model statistics.

Given this output, respond to the following questions:

1. Based upon the linear models you generated, which pair of features appear to be most predictive for one another? Note that you can answer this question based upon the output provided for the linear models.
2. Suppose you tried to generate a classification model on this dataset, but only after removing the feature that you were best able to predict based upon other features. How would removing this feature affect the classification performance?

2 Classification (50 points)

The Data
For this portion of the assignment you will be using a set of images of the works of various artists. To acquire the large number of images, we used a script that searches for each artist's painting on Google Images and downloads it. The provided dataset consists of images obtained in this manner for several well-known artists. The image are of varying size and quality.

The provided dataset is available here.

You are welcome to use the included function (`go_google_image`) to search and retrieve paintings by your preferred artist(s), which may then be used for this assignment.

The Idea: Identifying Artists by their Paintings
Your objective here will be to perform classification on the dataset to discern how reliably the paintings can be attributed to their respective artists. Several data preprocessing steps, including dimensionality reduction and clustering, may be applied to generate features more amenable to this task. Specifically, we would like to perform classification on the images so that they are correctly attributed to this respective artists. We can assess classification performance by dividing the dataset into a training set and a testing set, and measuring the error of a classifier fitted on the training set and evaluated on the testing set.

Intuitively, since we typically view an image as a collection of pixels, we might consider using the set of pixels as a feature. In other words, each color value for each pixel would be a feature, with the collection of pixels comprising many features that collectively describe the pixel values of the image. We call the collection of features that describe a particular aspect of the image a "feature descriptor." While this feature descriptor describes each pixel of the image, the specific locations of particular color values and the size of the image directly affect the derived feature values, thus making it difficult to use this set of features to directly compare images.

We can supplement or replace the pixel-based features with a histogram of color values. Each color has 256 possible values, resulting in a histogram of 768 (256*3) color values distributed over the entire range of pixels. Each value of the histogram is the number of pixels in the image with the corresponding color value. Here, we would consider each histogram value as a feature of an image. The histogram provides a representation of the color distribution of an image, ignoring the specific location of color values within the image. Each value of the histogram then corresponds to one feature.

We can also use more complicated features, such as histograms of Oriented Gradients (HOGs). HOGs are feature descriptors used in computer vision and image processing for the purpose of object detection that count the occurrences of gradient orientation in localized portions of an image. HOG descriptors are based upon the premise that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. These properties can derived by dividing the image into small connected regions or "cells," and compiling a histogram of gradient directions or edge orientations for the pixels within each cell. The combination of these histograms then represents the HOG descriptor, which has been shown to be quite effective for the purposes of classifying images.

While these are several of the popular feature descriptors used for image classification, other feature descriptors could be incorporated in the final set of feature used to train a classification model. In addition, each of these feature descriptors could be further preprocessed for further enhance their descriptive power.

What to Do

The IPython Notebook provided with this assignment includes functions to compute the histograms and plot the images within the transformed (2-dimensional) space (`load_images` and `plot_image_space`, respectively). There are also functions to generate and plot the color palettes associated with each image (`cluster_image_colors` and `plot_color_palette`, respectively); the palettes are generated via (k-means) clustering of the pixel color values, and may be investigated at your own leisure—they are not needed to complete the assignment.

The images can be loaded and the histograms generated by running the following code snippet (which imports pandas as `pd`). Please ensure that the directory provided to the `load_images` function is correct. For example, if you have placed all the images in your base IPython Notebook directory in a folder labeled `images`, with the paintings images in a subfolder labeled `paintings`, then the (relative) path to the painting images would be `images/paintings`. The following code snippet loads all images from the 'Pablo Picasso Paintings' and 'Vincent van Gogh Paintings' subdirectories (the `painters_subdirs` variable can be changed to your painters of interest, or omitted to load all of the subdirectories) and generates feature descriptors based upon the loaded images:

```python
import pandas as pd
# Load images and generate initial feature descriptors
painters_dir = '/path/to/painting_images' # directory path
painters_subdirs = [u'Pablo Picasso Paintings',
                    u'Vincent van Gogh Paintings']
data = fetch_paintings(painters_dir, painters_subdirs)
gen_pixel_fd(data) # generate pixel-based feature descriptor
gen_hist_fd(data) # generate color histogram feature descriptor
gen_hog_fd(data) # generate HOG feature descriptor
```

Each of these feature descriptors has a different representational format. However, the classification models available via scikit-learn only accept data into a single-dimensional vector format. Accordingly, these features need to be flattened into a single long array. The following code snippet accomplishes these tasks:

```python
# Arrange feature descriptors into a single feature vector
images = [pp['image'] for p in data.keys() for pp in data[p]]
X = pd.DataFrame([pp['features'] \
               for p in data.keys() for pp in data[p]])
y = pd.DataFrame([cls for cls in [pp['class'] \
               for p in data.keys() for pp in data[p]]])
for index in pd.isnull(X).any(1).nonzero()[0]: del images[index]
y = y.drop(y.index[pd.isnull(X).any(1).nonzero()[0]])
X = X.dropna()
```

Given this feature vector, visualize the data into two-dimensions using a method of your choice. Subsequently classify the data using 5-fold cross-validation, using one or more classifiers or your choice.

What to Provide

Your output should contain the following:

- For a pair of artists of your choice, generate a two-dimensional visualization.
- For a pair of artists of your choice, generate at least three classification models that distinguish between the artists' paintings.

- Generate at least one feature descriptor (in addition to those provided) and include it in the set of features used for your classifiers. Rerun the classifiers with the newly-included feature descriptor.

Given this output, respond to the following questions:

1. What is the highest classification accuracy you achieve?
2. Of the classification models you tried, did any tend to perform better or worse than others? Any thoughts why?
3. What feature descriptor(s) did you generate? Did they improve the classification performance? If so, by how much?

3 Extra Credit Portion (+10 points)

The Idea: Good Classifiers Generalize Well
A good classifier should generalize well to data it has not yet seen. When performing any classification task, the typical objective is to minimize the generalization error that a classification model will produce on new data.

What to Do
Generate the best classifier you can for distinguishing between two different artists' paintings, per the classification problem outlined above. We define the best-performing model here as the model with the highest accuracy. We will use a new dataset from two unprovided artists to test each student's model using 5-fold cross-validation. The student with the best-performing model (the model with the highest accuracy on this new dataset) will earn 10 points extra credit; the student with the second-best model will earn 9 points; and so on, with the tenth-best model earning 1 point. Thus up to ten students will receive some degree of extra credit on this task.

What to Provide
If you're interested in participating, ensure that your model is properly delineated in and runnable from your notebook. All content needed to generate your classification model should be included in your assignment submission.