DTSC 680: Applied Machine Learning

Assignment 6: Mushroom Classifier

Overview:

The purpose of this assignment is for you to gain experience with the KNN and PCA algorithms, as well as to assess your ability to construct a well-organized and well-documented notebook. In this assignment you will not receive a notebook template.

Upon submission, be sure that your notebook is well documented - this is required for code development and collaboration in industry, and is needed for grading your work. You will be heavily graded on the clarity of your notebook. How easy is it to understand the meaning of your computations? Do the names of your variables clearly communicate their meaning? Is your notebook documented completely? **Document/Comment your code so that someone who didn't know about your assignment would be able to follow what you were doing every step of the way.**

At the beginning of your notebook, you must include an overview of your work. You should offer a detailed explanation of all steps taken in the notebook, and you should explain your process thoroughly. Give special attention to any deviations from the below outline - there may be steps you need to add!

You will be working with the mushroom data set. You should download the necessary files at the supplied link - I have also posted the materials to Brightspace (zip file called mushrooms), in case you have difficulties downloading the data for yourself. Begin by inspecting all of the materials and understanding the data set (note that you will not use all of the supplied files).

Directions:

The following outlines, roughly, what you aim to accomplish in this assignment:

1. Import the Mushroom data set.

2. Use the KNN algorithm to impute missing values in the dataset. ***Note: You are not permitted to use the KNNImputer class from Scikit-learn.*** Instead, you must explicitly write the code to perform the imputation using the KNeighborsClassifier algorithm (use the default settings).

The first step is to think through what will be your feature data and what will be your response data for this imputation step. You will want to one-hot-encode your feature data and label encode your response data. Next, you should train your KNN model on those instances that don't have missing values, then have the model make predictions for those instances that have missing values. This is how you will have the KNN model impute missing data.

When you have computed the missing values, create a data structure (i.e. a list) called `missing_values` that contains all of the imputed values (in terms of the original categorical/letter data and not the encoded numeric data) in order of increasing index from the original data set. You must then print the first 10 instances of `missing_values` to the screen so we can check your work. Finally, you will impute the missing values back into the original data set before continuing, so that the next step starts fresh with a complete data set in terms of the raw data values.

Graded Concept Question #1 (include a section in your notebook): Would it still be possible to train the KNN model if you one-hot encoded the response data instead? Why or why not?

3. Train a RandomForestClassifier as well as a LogisticRegression model to predict whether a mushroom is edible or poisonous given this data set of nominally-valued characteristics. Train the model on the feature data supplied to you after you've one-hot encoded it. You should label encode the response data.

Graded Concept Question #2 (include a section in your notebook): Would it still be possible to train these two models if you one-hot encoded the response data instead, being careful to specify that the drop parameter of the OneHotEncoder class is set to 'first'? Why or why not?

You do not need to perform a grid search to find the optimal hyperparameter values - some of these models will take a long time to train, so we will not focus on that in this assignment. Accordingly, you should use the default hyperparameter settings for both models, along with a random_state of 42 for the RandomForestClassifier. (Note: In the real world, you would want to perform a grid search, but since we covered this in other assignments and want to cut down on the necessary time, you will not include this.)

4. When you train both the RandomForestClassifier and LogisticRegression models, use the magic command %%time to time how long it takes to complete training. See the Python Data Science Handbook for more information about this magic command.

5. Compute the accuracy, precision, and recall scores for a test set. Briefly discuss the performance of your models in terms of these values.

6. Perform dimensionality reduction using PCA and keep 95% of the variance. By what percentage were you able to reduce the number of dimensions of the training set? How many features (i.e. dimensions) are you left with after reducing dimensionality?

7. Train two new models, a RandomForestClassifier and a LogisticRegression model, on this reduced dataset to predict whether a mushroom is edible or not. Again, time the training of these two models on the reduced dataset. You will again use the default hyperparameter values for both models, with a random state parameter of 42 for the RandomForestClassifier.

8. Compute the accuracy, precision, and recall scores for the model trained on the reduced data set using the same test set as before. What conclusions can you make about these models and the PCA process? How do the models compare? Discuss the trends observed for model training time and performance for the full and reduced data sets. Tabulate the following information in your final analysis:

Models	**Item**	**Full Data**	**PCA Reduced**
Random Forest	**Accuracy**	<Value>	<Value>
	Precision	<Value>	<Value>
	Recall	<Value>	<Value>
	Time	<Value>	<Value>
Logistic Regression	**Accuracy**	<Value>	<Value>
	Precision	<Value>	<Value>
	Recall	<Value>	<Value>
	Time	<Value>	<Value>

Below is a simple rubric indicating how you will be graded:

Clarity and Documentation 20%
How well-organized is your notebook? Do you have enough comments? Is your notebook documented completely? How easy is it to understand the meaning of your computations? Do the names of your variables clearly communicate their meaning? Is your notebook overview sufficiently descriptive? Could someone not familiar with the assignment follow your code?

Perform KNN Classification to Impute Missing Values 30%
Were you able to correctly implement encoding algorithms to prepare data for the KNN algorithm? Were you able to correctly fit a KNN model and make predictions for the missing values in the data? Did you print the missing values (in letter form) to the screen?

Fit RandomForestClassifier & Logistic Regression models 10%
Did you successfully fit a `RandomForestClassfier` and `LogisticRegression` model to the data for the full and reduced data sets (i.e. before and after PCA) to predict whether a mushroom is edible or poisonous? Did you time how long it took to train each respective model on the full and reduced data sets?

Graded Concept Questions 10%
Were you able to answer the following questions:

1. Would it still be possible to train the KNN model if you one-hot encoded the response data instead? Why or why not?
2. Would it still be possible to train the two models on the RandomForestClassifier & Logistic Regression algorithms if you one-hot encoded the response data instead, being careful to specify that the drop parameter of the OneHotEncoder class is set to 'first'? Why or why not?

Perform PCA 15%
Were you able to successfully perform PCA? By what percentage were you able to reduce the number of dimensions of the training set? How many features (i.e. dimensions) are you left with afterwards?

Measuring Generalization Error and Model Analysis 15%
Did you successfully calculate the accuracy, precision and recall for the full and reduced models on the training and test sets? Did you analyze the possibility of overfitting the data and discuss this? Did you make conclusions about the two models? How do the full and reduced models compare?