**DATA 440 – Assignment 1: Support Vector Machines**

**Introduction**

Support Vector Machine models are supervised learning models that can be used for classification as well as for regression, but you are to use them for classification in this assignment. SVMs find optimal hyperplanes with the maximum margin (distance) that attempt to “separate out” the points of all the classes in n-dimensional space. A key part of their ability to separate points into classes is based on the “kernel trick” which consists of mapping the data into a higher dimension. Picture we had a ‘red’ class and a ‘blue’ class in two dimensions that are not linearly separable—but if we somehow map them to three-dimensional space, then we can more easily separate them:

|  |  |
| --- | --- |
|  |  |

**General steps:**

1. **Coding:**
2. Review the theoretical background and SVM implementation example that was provided in Practice Exercise 1A. You can choose to work with SVM\_440.py as a Python module in Spyder or another Python IDE or with SVM\_440.ipynb as a Jupyter notebook in Google Colab or another compatible Jupyter notebook environment.
3. For this assignment, you will use the dataset that you chose in Practice Exercise 1B.
4. You are to modify the Practice Exercise 1A code to use your chosen dataset with your choice of input variables and output variable/s:
   1. As per the sample code you will divide your full dataset into training and testing datasets.
   2. You will need to have picked a categorical output variable/s, since your are building a classification model, and these need to be mapped to dummy variables using LabelEncoder() and fit\_transform() ; see the sample code.
   3. Make sure that as in the sample code you use dropna() to remove any rows with missing data, since SVM’s cannot handle missing data.
   4. If you have categorical input variables use one-hot encoding. Assume you have a categorical input variable named body\_style, then this is the suggested approach, assuming you fetched your data set using fetch\_openml() as in the example:

#Get rid of any rows with NA's

dataset.frame = dataset.frame.dropna()

**#And this is the code you would add:**

**from** **sklearn.preprocessing** **import** OneHotEncoder

oe\_style = OneHotEncoder()

oe\_results = oe\_style.fit\_transform(dataset[["body\_style"]])

dataset = dataset.join(pd.DataFrame(oe\_results.toarray(), columns=oe\_style.categories\_))

* 1. For reference and more ideas see: <https://pbpython.com/categorical-encoding.html> and focus on the ScikitLearn OneHotEncoder section.

1. Run the code to fit your SVM model and record the results. Try with different kernel and decision\_function\_shape parameters in the svm.SVC() function.
2. Submit your python code from steps 3) to 4) as well as a text file with the output of your program.
3. **Technical Report:**
   1. Document your findings and analysis in a technical report using the template that accompanies these instructions.
4. **Deliverables with critical areas:**

**Python Code and Output (40%)**: In addition to any python code that you may include in the Appendix, you should submit all your python script as a separate file when you submit the assignment to LEO. Also submit a text file with all the output from your program. Adding some useful comments to your code (starting with ‘#’) will increase your grade in this section.

**Technical Report(60%):**

**Overview**: areas to address:

* **Problem Domain**: give some background and context about the problem domain (application area). For instance, if you are doing the analysis for predicting heart disease, provide some context about the disease and include some interesting statistics about it. Also, discuss how the method is relevant for the chosen problem.
* **Objective**: clearly state the objective of the analysis in relation to the kind of algorithm you are employing. Use specific language as to what question(s) you are trying to answer using the specific analysis/modeling type. Your answers will depend on your choice of input and output variables.

**Analysis**: areas to address:

* **Exploratory Analysis**: describe the data including the source, the collection method and variables. Perform exploratory analysis. Also, select few key variables (including the target variable for supervised learning) and study their distributions using plots such as histograms, box plot, bar chart, etc.
* **Preprocessing**: describe how you prepared the data especially for categorical inputs and categorical outputs.
* **Model Fitting**: explain the key steps and activities you perform to fit the model. Experiment (as appropriate) with parameters tuning: kernel and decision\_function\_shape parameters in the svm.SVC() function. This is key, what separates highly accurate model from a less accurate ones is the amount of performance tuning performed.

**Results**: areas to address:

* **Model Properties:** explain the components of the fitted model and their characteristics. Leverage functions to summarize the model properties. Also, leverage visualization as possible.
* **Output Interpretation**: explain the result and interpret the final model output using terms that reflect the application area and in relation to the stated objective. This is where you check whether or not the stated objective is met.
* **Evaluation**: describe the metrics used to quantitatively evaluate the performance of the fitted model that are in your code: confusion matrices from the function confusion\_matrix() and other accuracy statistics from the function classification\_report(): precision, recall, f1-score and support. Interpret these statistics in the context of your model.

**Conclusion**: areas to address:

* **Summary**: highlight the main findings in relation to the stated objective. You don’t need to discuss the details of the analysis and the model such as accuracy here, just focus on the key findings.
* **Limitations & Improvement areas**: discuss the limitations of the analysis and identify potential improvement areas for future work. This could be related to the data, algorithm, or a combination of the two.

**Miscellaneous**:

* Use the template that accompanies these instructions to submit your responses to each section, with Python code and any extended model outputs submitted as separate files and you can include code snippets in the Appendix.
* Proofread your report for correct structure, grammar, and spelling.
* The report should be entirely in your own words, no direct quotes from any source. However, keep in mind that any original ideas, information or interpretation of your dataset or regarding the general use of any algorithm, method, or model that you may discover from a source must be cited. Follow appropriate APA format and provide all necessary references. If you have any questions about this requirement, please ask your instructor for clarification.
* Graphics, figures, or tables should be titled and explained. For example, screen captures generated should be assigned a figure title and label (e.g. Figure 1.xxx) and have a description associated with that figure providing details and context for the image.

***The total length of the report should be 6-9 pages, single-spaced within the text areas of the template provided, excluding the appendix, and python script. Large code snippets and graphs should be in the appendix.***