MEM T380 – Applied Machine Learning in Mechanical Engineering

Case Studies Assignment 2
Part B - DTC

Dimitrios Fafalis, PhD
Dimitrios.Fafalis@drexel.edu

due Thursday, May 11, 2023

Students' names & ID:

1. ______________________________ ____________
2. ______________________________ ____________

Submit your files (`.ipynb`) and a report (`.pdf`) on Blackboard by due date.

Classifying Weld Defects!

CASE STUDY 1. points 80 – **Weld defects classification**
This case study is related to classifying weld defects and is based on the research article "**Lu Yang and Hongquan Jiang, 2020**, *Weld defect classification in radiographic images using unified deep neural network with multi-level features*, Journal of Intelligent Manufacturing ". This article is available in the Case Studies folder for HW-2. Although the authors of the paper used an artificial neural network (ANN) machine learning approach to classify the weld defects, in this and the following assignments you will use the k-Nearest Neighbors and the Decision Trees classification techniques. In a later lecture, we will revisit this case study and practice with ANN. Refer to this article to familiarize yourselves with the problem.

The raw data for this case study are available to you in an excel workbook posted in the assignment `weld_defect_dataset.xlsx`. The data are partitioned into five subsets, each one stored in a separate sheet named as `subset1`, ..., `subset5`.

The data come from image processing of radiographic images. The features and the target classes have been extracted from those images. The data consist of eleven (11) total features and seven (7) weld defect types. How the features were extracted from the radiographic images is a completely different topic. You are welcome to reach out to me to guide you into literature and tools on how to do that.

The goal of this case study is to develop various ML models that classify weld defects observed in radiographic images.

1 Data Exploration Tasks

0 points: this task has been performed in HW2/Part A; reuse it here as a prerequisite for the new tasks. The very first steps in developing a `Machine Learning` model are to load, explore, and preprocess the data. The goal of this task is to explore the data and try to listen to what they want to tell us!

1. using the `pandas` command `read_excel`, load the datasets available into the five sheets of the file `weld_defect_dataset.xlsx` and store them into `pandas` dataframes. It is advised to create a different dataframe for each subset so that you can do cross-validation when you explore the classification models. Refer to pandas.read_excel documentation for examples and additional options.

2. print a summary of the information included in the dataframes using the functions `df.info`, `df.dtypes` and `df.describe`.

3. identify whether there are missing values in any of the fields (columns) of the dataframes.

4. identify whether there are duplicated entries in the dataframes.

5. if there are missing values, return a part of the table that contains only the entries (rows) of the missing data (include all columns). Do you think that the missing data in the particular fields (columns) would have an important impact on interpreting the data of the other fields? Create a new dataframe that contains only non-missing data.

6. if the original dataframes had missing or duplicated entries, print the properties of the *cleaned* dataframe, using the commands `df.info` and/or `df.describe` .

7. identify which of the columns (features) could be used as `categorical` data. These could be `cell` or `string` arrays or even `numeric` data with distinct and repetitive values along all entries. Convert these columns into `categorical` type.

8. use the `seaborn` command `pairplot` to visualize `bivariate` relationships between the `numerical` fields grouping them by the `categorical` fields. Repeat this task as many times as the number of categorical features of the original dataframe (*Hint*: in our case only the target classes is categorical).

9. create a heatmap of the correlation matrix of the features.

10. identify from the figures in item 8 which `numerical` features (fields) are the **strongest** to be used in classifying the data in any of the groups in the `categorical` fields; in other words, which features distinguish the data in groups as clear as possible. Do you see any patterns or trends?

11. are your insights from item 10 justified by looking at the heatmap correlation matrix in item 9?

12. create any other figures using `matplotlib` or `seaborn` that would help you better understand your data. E.g., are they imbalanced? what is the range of values for the features, are there significant differences, etc.

2 Classification with Decision Trees (DT)

2.1 Part 1:

50 points. For the following tasks use **four** subsets as **training** data and **one** subset as **testing** data.

1. Based on the data exploration you performed in section 1, choose **two** strong features and **three** defect types that are easy to distinguish in a bivariate scatter plot. Justify your choice.

2. Create two `DT` classifier models using the class `sklearn.tree.DecisionTreeClassifier` of the `sklearn` package, and the **training** dataset. Experiment with different values of `max_depth`, `criterion`, `min_samples_leaf`, and `max_leaf_nodes`. Refer to `sklearn` documentation DecisionTreeClassifier for explanations and examples.

3. Visualize the `DT` models using the method `sklearn.tree.plot_tree`. Show graphics for a couple of models you experimented in item 2.

4. Visualize the `DT` models using the method `sklearn.tree.export_text`. Print the trees for a couple of models you experimented in item 2.

5. Visualize the `DT` models using the method `sklearn.tree.export_graphviz`. Show graphics for a couple of models you experimented in item 2.

6. Create a decision surface figure for two models you created in item 2 to help you visualize the class space. *Hint:* refer to section *Plot the decision surface of decision trees* of notebook `W4-DT-Classification-IRIS.ipynb` and on `sklearn` documentation website Plot the decision surface of decision trees trained on the iris dataset.

7. By looking at the `DT` graphical depictions in item 3 and the decision surfaces you created in item 6 give a short narrative on what you observe with respect to the boundaries of the classes' spaces . Which decision tree would you prefer and why? Which one do you think would give more accurate predictions on new data? Give some examples by picking points on the decision surface figures and elaborate.

8. Write down the mean accuracy for each model, using the method `score(X, y)`, that comes with the `DT` models you created. Report the accuracy score for both the training dataset and the testing dataset.

9. Use the trained models in item 2 to predict the weld defect types for the **testing** dataset.

10. Calculate the confusion matrix using the command `sklearn.metrics.confusion_matrix` and display it using the `seaborn` command `seaborn.heatmap`.

11. Provide the classification report using `classification_report` function of the metrics library of the `sklearn` package. Comment on the performance of the model. What is the misclassification error?

2.2 Part 2: Pruning and Overfitting Prevention

30 points. For the following tasks combine all subsets into one large dataset; just create a new pandas dataframe. Also, use all features and all target defect classes.

1. Perform a `GridSearchCV` cross-validation technique with $K = 10$ (ten) folds, to determine the optimal value for the `max_depth` hyperparameter of the `DecisionTreeClassifier` class. Increase the `max_depth` from 1 to 20.

2. Create an accuracy score plot as a function of `max_depth`, for both the training and testing subsets.

3. From the figure you created in item 2 above, identify the optimal `max_depth`.

4. Randomly split the entire dataset into training and testing subsets using the function `sklearn.model_selection.train_test_split`.

5. Create a decision tree classifier using the optimal value for `max_depth` you identified above. Fit the model with the training subset.

6. Use the trained model to predict the weld defect types for the **testing** subset.

7. Calculate the confusion matrix using the command `sklearn.metrics.confusion_matrix` and display it using the `seaborn` command `seaborn.heatmap`.

8. Provide the classification report using `classification_report` function of the metrics library of the `sklearn` package. Comment on the performance of the model. What is the misclassification error?

9. How do the metrics of the optimal `max_depth` compare with the metrics of the models you explored in Part 1?

10. From this optimal model, return, print and plot the feature importances, as shown in class. Which features seem to play a significant role in this classification problem? How do they compare with the features you selected initially in Part 1? Are there any surprises?