DI 501: Introduction to Data Informatics
Assignment 4

Volga Sezen

Due by 09.06.2024

In this assignment you will work with an expanded version of the ELODIE dataset to classify and analyze the distribution of stars with different models. **Be sure to check out the ReadMe.txt file with new and existing column descriptions.**

Deliverables

You will be asked to present your experiments and findings in a pdf report, and your code in a well annotated .ipynb file. While there is no strict format, you can follow the IEEE conference manuscript format.

Submission and Grading Principles

- This is an individual assignment. Please refrain from collaboration. Upon any hurdle, you can contact your TA. It is important you adhere to academic integrity principles.
- Whenever a decision is made, justify it by referencing metrics and different trials.
- Offer commentary on the results, but keep it relevant and concise. Results with no commentary will lose up to 50% of the grade of that part.
- Your results must be shown in tables or other appropriate structures. Do not copy and paste code or outputs from your notebook for the report. This will result in grade deductions.
- Your report must include a minimum of two graphs and tables. The maximum number for both is six. Make sure the graphs you provide are exported at 300 dpi and not copied and pasted. Failing to do so or not staying within limits will result in grade reductions.

1. EDA and Preprocessing (15 pts)

To begin, get to know the expanded version of the dataset and prepare the data for training.

- Briefly analyze the features given for possible correlations with each other and the target class. Also observe the target distribution. For the most correlated features, consider engineering features out of them.
- Identify erroneous values with the help of observation quality flags. Either drop rows with those values, or impute them using a technique discussed in class.
- Divide the dataset into training and test splits with 80% and 20% of observations such that each split contains similar proportions of each class.
- Analyze the feature ranges in the training set and pick a preprocessing technique to ensure each feature spans the same scale.

2. Baseline and Scoring (10 pts)

Come up with a simple baseline method for predicting the star's main spectral class. (Which should not involve any learning!) Determine a single value metric and set up functions to print classification reports and confusion matrices.

Evaluate the performance of the simple baseline method according to the metrics defined. Explain the assumption underlying the baseline, and why it was picked.

3. Non-Neural Network Model (25 pts)

Choose a supervised machine learning model (either covered in class or not) to classify stars. Initialize it with a random state (if applicable), and any parameter that will not be changed. Find optimal hyper-parameters by designing a parameter search:

- Pick hyper-parameters that are relevant to the model complexity.
- Decide on a range of values for each hyper-parameter.
- Fit a model with either a grid or randomized search with built in cross validation.
- Ensure the best possible fit is achieved by re-initializing the search if the best score for a given combination has values on the edge of the parameter grid.
- Once the best model configuration is found, evaluate it on the test set, and compare the performance level with the simple baseline.

Then answer the following: Is this model better than the simple baseline? If so, by how much? Discuss its' performance in identifying each class. Are there any weaknesses, or strengths?

Note: If the model of choice utilizes gradient descent, make sure each model is given enough iterations to converge. Failure in convergence can be tracked with function printouts and performance levels.

4. Fully Connected Neural Network Optimization (30 pts)

Train neural networks with scikit-learn's `MLPClassifier()` module. These networks follow the structure of a multi-layered perceptron.

They comprise of many hidden layers, and **each layer can have the same number of neurons** or they can change to constrict and expand the flow of information. To keep the search space manageable, focus on the **first case** only.

Apply the same pre-processing steps before training. Initialize the MLP with a sufficient amount of iterations, (not the default value), a random state and other fixed parameters.

Find optimal values of number of layers and number of neurons per layer by utilizing a **grid search** specifically. (Having a small search space should be beneficial here.) Some tips:

- Using the following line one can generate all unique combinations of number of layers and number of neurons.

```
np.array(np.meshgrid(hidd, nlr)).T.reshape(-1, 2)
```

- Each combination will need to be converted to a format that `MLPClassifier()` can take as input.

After finding the best neural network configuration, answer the following:

- Test the final model's performance against the test set and compare your results with the baseline and the non-neural network model. Are there any improvements?
- How does model depth and width affect the performance levels? Cross validation results can be plotted using heatmaps, 3D graphs or simple line graphs to illustrate this point.
- Finally, using a technique of your choosing (permutation importance, SHAP values, tree feature importances), explain how much these models rely on each feature overall. Do different models prioritize different features?

Bonus (10 pts)

- What type of loss function is used in `MLPClassifier()` by default? Recalling that our target class is ordinal in nature, how appropriate is this loss function for this task?
- What was the default learning rate of `MLPClassifier()` ? Plot the loss curve for the optimal model. Change the learning rate to 0.1 and 0.000001 and observe the loss curve and model performance afterwards. How does high and low learning rate affect model performance?

5. Clustering Analysis (20 pts)

Analyze how these stars occupy the feature plane by using clustering algorithms and visualisations.

- Use the K-Means clustering algorithm to determine the optimal number of clusters using the sum of squared errors and silhouette scores. Does the optimal cluster number represent the number of classes in our dataset?
- Calculate two dimensional approximations of the feature space with the PCA and T-SNE algorithms. With a scatter plot, show the two components and color each point by the original label.

  Which type of manifold captures more visually distinct clusters? Can feature scales affect the clustering? If so, unify feature scales. Choose one technique for further analyses.
- With scatter plots, show the two components from PCA/T-SNE and color each point by the predicted clusters in k=optimal and k=7 scenarios in K-Means. Are any fundamental characteristics captured by the model with optimal number of clusters? Is there any overlap between the clusters and original labels? *This can also be compared with differences of predictions and true labels (whether a particular class always takes the unrelated cluster number).*
- Utilize the DBSCAN algorithm and find the optimal value for the distance metric that determines whether two points belong to different clusters by checking the silhouette scores and using the PCA/T-SNE plots. How many clusters are predicted by the optimal model? Observe which stars take the unknown class, and whether their attributes are abnormal.