UDACITY

Return to "Data Scientist Nanodegree" in the classroom

Data Scientist Capstone

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hopefully you have learned a bunch throughout this capstone project (as I can image that you have by reading your report) and you can take some of these techniques further.

If this is your final report, I would like to be the first one to congratulate you on completing this nano-degree! Wish you the best of luck in your future!

Project Definition

Student provides a high-level overview of the project. Background information such as the problem domain, the project origin, and related data sets or input data is provided.

Good intro to your project here as this is a great opening as it gets the reader excited about going further.

Still. Would suggest also providing some research / links for other machine learning prediction problems similar to yours. It is always important to provide similiar research on such a topic to give some backing to your claims.

The problem which needs to be solved is clearly defined. A strategy for solving the problem, including discussion of the expected solution, has been made.

Great theoretical workflow for your problem statement here. Definitely gives the reader a solid understanding in what is to come in the rest of your project and how you plan on tackling this classification problem.

Metrics used to measure performance of a model or result are clearly defined. Metrics are justified based on the characteristics of the problem.

> "This evaluation metric is preferred because the training data which we will talk about in the fourth part is highly unbalanced."

Perfect! As using accuracy is not the best for unbalanced datasets based on the accuracy paradox.

Analysis

Features and calculated statistics relevant to the problem have been reported and discussed related to the dataset, and a thorough description of the input space or input data has been made. Abnormalities or characteristics about the data or input that need to be addressed have been identified.

Very nice job describing your dataset. Great job describing your dataset here and good work computing the min, max and std for the features. This is always a good first step in any machine learning model.

Build data visualizations to further convey the information associated with your data exploration journey. Ensure that visualizations are appropriate for the data values you are plotting.

All great visuals to show here! The distribution of target variable and correlation plots are always a good idea. Maybe another idea would be to combine the target variable with these features to get a better understanding of how each correlate to the target variable with a seaborn factorplot.

Also glad that you label your Figures here, such as 'Figure 1', etc... As this allows you to quickly reference them throughout your project.

Methodology

All preprocessing steps have been clearly documented. Abnormalities or characteristics about the data or input that needed to be addressed have been corrected. If no data preprocessing is necessary, it has been clearly justified.

The process for which metrics, algorithms, and techniques were implemented with the given datasets or input data has been thoroughly documented. Complications that occurred during the coding process are discussed.

Good choice in your K value based on your visual. We don't necessarily always see an actual 'elbow' in the curve (it is quite nice when we do!). Therefore we often need to make our best judgment based on the domain knowledge, the decline of average distance, and other factors.

Another popular metric for evaluating clustering models is Silhouette Coefficient. The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is 2 <= n_labels <= n_samples - 1.

The process of improving upon the algorithms and techniques used is clearly documented. Both the initial and final solutions are reported, along with intermediate solutions, if necessary.

One option for a smarter implementation of hyperparameter tuning is to combine random search and grid search:

- Use random search with a large hyperparameter grid
- Use the results of random search to build a focused hyperparameter grid around the best performing hyperparameter values.
- Run grid search on the reduced hyperparameter grid.
- Repeat grid search on more focused grids until maximum computational/time budget is exceeded.

Or could even look into using hyperopt. Here might be a good example of how to use this bayesian optimization technique in python.

Results

If a model is used, the following should hold: The final model's qualities — such as parameters — are evaluated in detail. Some type of analysis is used to validate the robustness of the model's solution.

Alternatively a student may choose to answer questions with data visualizations or other means that don't involve machine learning if a different approach best helps them address their question(s) of interest.

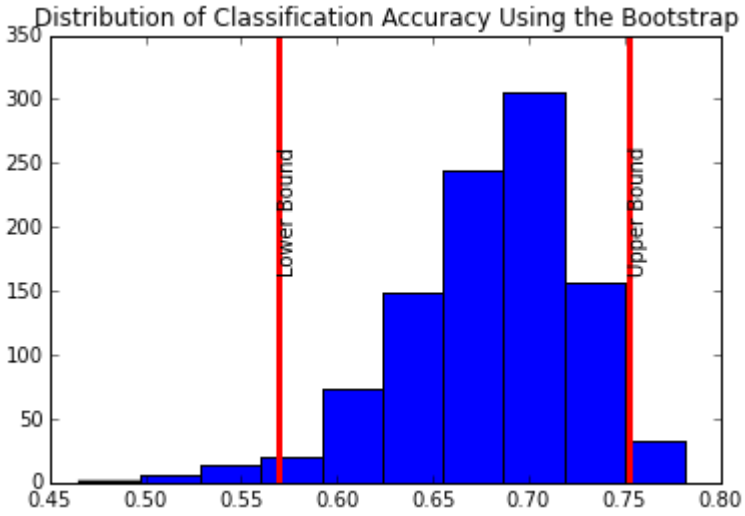
Great analysis of your final models.

Maybe one other idea would be to plot and 95% confidence interval to determine if the model is robust with bootstrapping. Here might be an example with the boston housing dataset.

```
from sklearn.utils import resample
```

```
import matplotlib.pyplot as plt

data = pd.read_csv('housing.csv')
values = data.values
# configure bootstrap
n_iterations = 1000
n_size = int(len(data) * 0.50)

# run bootstrap
stats = []
for i in range(n_iterations):
    # prepare train and test sets
    train = resample(values, n_samples=n_size)
    test = np.array([x for x in values if x.tolist() not in train.tolist()])
    # model
    model = DecisionTreeRegressor(random_state=100)
    model.fit(train[:,:-1], train[:,-1])
    score = performance_metric(test[:,-1], model.predict(test[:,:-1]))
    stats.append(score)

# confidence intervals
alpha = 0.95
p = ((1.0 - alpha) / 2.0) * 100
lower = max(0.0, np.percentile(stats, p))
p = (alpha + ((1.0 - alpha) / 2.0)) * 100
upper = min(1.0, np.percentile(stats, p))

# plot
plt.hist(stats)
plt.axvline(lower, color='red', lw=3)
plt.text(lower, n_iterations // 4, 'Lower Bound', rotation=90)
plt.axvline(upper, color='red', lw=3)
plt.text(upper, n_iterations // 4, 'Upper Bound', rotation=90)
plt.title('Distribution of Classification Accuracy Using the Bootstrap')
plt.show()

print('%.1f confidence interval %.1f%% and %.1f%%' % (alpha*100, lower*100, upper*10
0))
```

(https://machinelearningmastery.com/calculate-bootstrap-confidence-intervals-machine-learning-results-python/)



The final results are discussed in detail.
Exploration as to why some techniques worked better than others, or how improvements were made are documented.

For your tree based model, could also plot some feature importance (http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html)

Conclusion

Student adequately summarizes the end-to-end problem solution and discusses one or two particular aspects of the project they found interesting or difficult.

Discussion is made as to how at least one aspect of the implementation could be improved. Potential solutions resulting from these improvements are considered and compared/contrasted to the current solution.

Might want to also check out using an Xgboost or LightGBM. Here might be a couple of examples of how to implement these powerful algorithms.

- Xgboost example
- LightGBM example

Deliverables

If the student chooses to provide a blog post the following must hold: Project report follows a well-organized structure and would be readily understood by a technical audience. Each section is written in a clear, concise and specific manner. Few grammatical and spelling mistakes are present. All resources used to complete the project are cited and referenced.

If the student chooses to submit a web-application, the following holds: There is a web application that utilizes data to inform how the web application works. The application does not need to be hosted, but directions for how to run the application on a local machine should be documented.

Student must have a Github repository of their project. The repository must have a README.md file that communicates the libraries used, the motivation for the project, the files in the repository with a small description of each, a summary of the results of the analysis, and necessary acknowledgements. If the student submits a web app rather than a blog post, then the Project Definition, Analysis, and Conclusion should be included in the README file, or in their Jupyter Notebook. Students should not use another student's code to complete the project, but they may use other references on the web including StackOverflow and Kaggle to complete the project.

Code is formatted neatly with comments and uses DRY principles. A README file is provided that provides. PEP8 is used as a guideline for best coding practices.

Best practices from software engineering and communication lessons are used to create a phenomenal end product that students can be proud to showcase!

DOWNLOAD PROJECT

RETURN TO PATH