UDACITY

PROJECT

Building a Student Intervention System

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Very nice adjustments in this report, as you have a very solid understanding of these techniques. I always recommend diving a bit deeper into the pros and cons of these models to get a better understanding of when to use these models in the future. But great and wish you the best of luck in your future!!

Classification vs Regression

✓

Student is able to correctly identify which type of prediction problem is required and provided reasonable justification.

Spot on!! Excellent job here as this is definitely a classification problem. As it is a binary classification problem where we have the column `passed` containing yes and no.

Exploring the Data

✓

Student response addresses the most important characteristics of the dataset and uses these characteristics to inform their decision making. Important characteristics must include:

- **Number of data points**
- **Number of features**

- **Number of graduates**
- **Number of non-graduates**
- **Graduation rate**

Nice adjustment and all correct here. It is definitely a good idea to get an idea of the distribution of our target variable here. As we now know that we have an imbalance of students, so we can plan accordingly. Also we know that if we were evaluating based on accuracy, our "dumb" classifier should predict 67%.

Preparing the Data

✓

Code has been executed in the iPython notebook, with proper output and no errors.

Everything runs fine!!!

✓

Training and test sets have been generated by randomly sampling the overall dataset.

Great work here with Sklearn's `train_test_split`, especially with the `stratify = y_all`, as this method makes it much easier to split your data.

Note: Also great work setting a `random_state` here as well with your splitting function, as this allows for reproducible split of the data, thus your training and testing samples are not randomly generated each time you run this.

Code Note: Instead of having to calculate `test_size = splitPercentage`. We could also simply assign the exact number we want to pass in with `test_size = 95` or `test_size = num_test`

Training and Evaluating Models

✓

Three supervised models are chosen with reasonable justification. Pros and cons for the use of each model are provided, along with discussion of general applications for each model.

Nice adjustments and great job!!

KNN

- Correct as this is a lazy learning, therefore we simply memorize the training data, thus training is fast however classification is slow
- The space of the learning rate of KNN is O(n) and the query rate is O(1). Since KNN is a lazy learning method classification time is nonlinear

Logistic Regression

- Correct with "*It is not heavily affected by noise in the data.*"
- Another downfall is that this is a linear classifier

- It can also be a more interpretable model

SVM

- Typically much slower, but the kernel trick makes it very awesome to find non-linearity in the data.
- Also note here the SVM output parameter are not really interpretable
- With the comment of "*Do not rely on entire data*", as this makes it very memory efficient.

✓

All the required time and F1 scores for each model and training set sizes are provided within the chart given. The performance metrics are reasonable relative to other models measured.

Nice work running your code with the different desired algorithms and your outputs look reasonable.

Choosing the Best Model

✓

Justification is provided for which model seems to be the best by comparing the computational cost and accuracy of each model.

Nice job here, as it seems as you have considered a faster algorithm. As this would be good regarding the scalability of your model. As this would be much more appropriate if we ever had to classify thousands or hundreds of thousands of students. Nice ideas!!

✓

Student is able to clearly and concisely describe how the optimal model works in laymen terms to someone what is not familiar with machine learning nor has a technical background.

I would recommend avoiding terms such as "*minimizing the squared error*", as this could still be a bit advanced for someone who is not familiar with machine learning nor has a technical background, but nice job here as you have a great representation of how we could predict new students.

As here could be a basic blueprint as well

- After the initial training using the training data Logistic Regression gives each feature a weight by minimizing a cost function.
- When we are predicting we take all feature values multiply them with their weights accordingly
- We then sum up all the results.
- This final result is applied to a special function called Sigmoid/Logistic function which only outputs values between 0-1 which can be interpreted as the probability of our positive label.

✓

The final model chosen is correctly tuned using gridsearch with at least one parameter using at least three settings. If the model does not need any parameter tuning it is explicitly stated with reasonable justification.

We really only needed to run one GridSearch here, but nice idea to try all. I would also recommend checking out the parameter gamma with your SVM. As a small gamma will give you low bias and high variance while a large gamma will give you higher bias and low variance.

Note: As with using an unbalanced dataset like this one with only 395 rows, and about 33% of students failing and 67% passing it is a great idea to use sklearn's StratifiedShuffleSplit to make sure the labels are evenly split between the validation sets.

✓

The F1 score is provided from the tuned model and performs approximately as well or better than the default model chosen.

Quality of Code

✓

Code reflects the description in the documentation.

DOWNLOAD PROJECT

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Rate this review

Student FAQ