UDACITY

Return to "Machine Learning Engineer Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Finding Donors for CharityML

REVIEW

CODE REVIEW

HISTORY

Requires Changes

4 SPECIFICATIONS REQUIRE CHANGES

Dear Student,

Congratulations on your submission!. Overall, good work on those parts of the project you completed. You'll find some sections are marked as not meeting specifications, I hope you find my argumentation reasonable and clear and helps you to continue your work on such sections. However, if you still have questions/comments, please don't hesitate to reach us, we'll be glad to help you.

Keep up your good work!

Exploring the Data

Student's implementation correctly calculates the following:

- **Number of records**
- **Number of individuals with income >$50,000**
- **Number of individuals with income <=$50,000**
- **Percentage of individuals with income > $50,000**

Nice job getting these numbers.

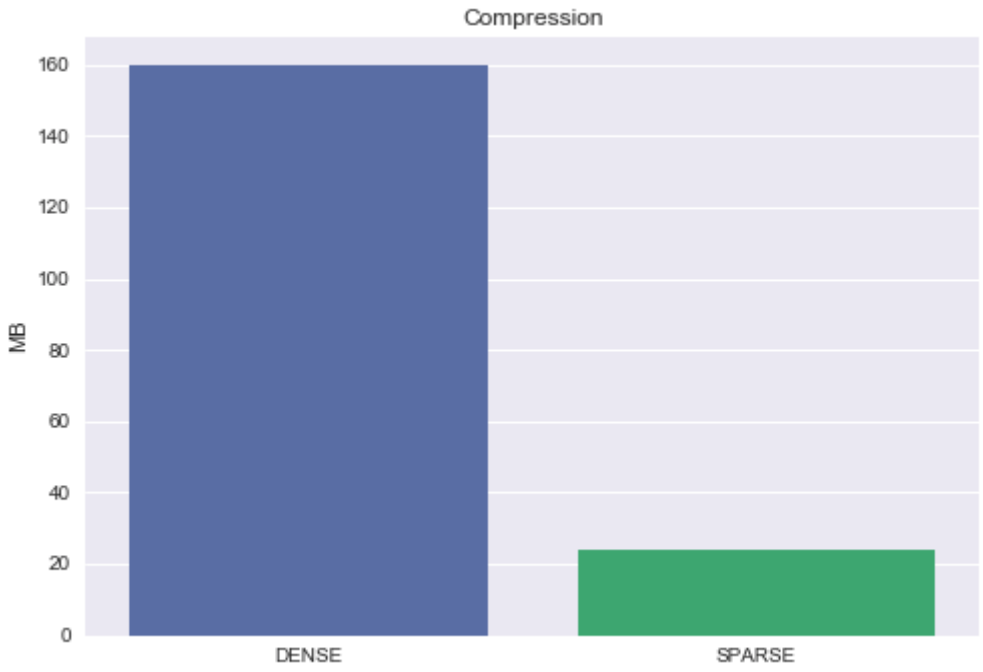
Preparing the Data

Student correctly implements one-hot encoding for the feature and income data.

Excellent implementation of one-hot encoding.

For your reference, this is a great post that demonstrates different ways to handle categorical variables.

Also, when you work with huge datasets, you might benefit of using Sparse matrices which are a memory efficient way to store matrices that are almost 99% zeroes and 1% ones.



Evaluating Model Performance

Student correctly calculates the benchmark score of the naive predictor for both accuracy and F1 scores.

Good first attempt!. However, Precision, Recall and Accuracy need to be recalculated:

```
# TODO: Calculate accuracy, precision and recall
accuracy = float(TP / (TP + FN))
recall = float(TP / (TP + FN))
precision = float(TP / (TP + FP))

# TODO: Calculate F-score using the formula above for beta = 0.5 and correct values for precision and recall.
beta = 0.5
fscore = (1+beta**2) * (precision * recall) / ((beta**2 * recall) + recall)

# Print the results
print("Naive Predictor: [Accuracy score: {:.4f}, F-score: {:.4f}]".format(accuracy, fscore))
```

```
Naive Predictor: [Accuracy score: 1.0000, F-score: 0.2478]
```

Being accuracy the number of times your model is right and your naive predictor always predicts more than $50,000, which would be your Accuracy?. What about Recall? note recall is the portion of people that make more than $50,000 correctly identified by your model and your model always predicts more than $50,000. And what about Precision? it is the percent of times your model is right on its more than $50,000 predictions..... Please reconstruct accuracy, precision and recall using the terms already calculated: `n_greater_50k`, `n_records` or using the confusion matrix (PT, FP, TN, FN) terms. To verify your results, you could use Scikit metrics for precision, recall and accuracy, but make sure in your written response it is included a definition of these metrics and F0.5 scores in terms of `n_greater_50k`, `n_records` as required in the rubric.

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to be explored.

Please list all the references you use while listing out your pros and cons.

Great discussion of the pros and cons for the different algorithms attempted, as well as their main applications and reasons to use them in this problem.

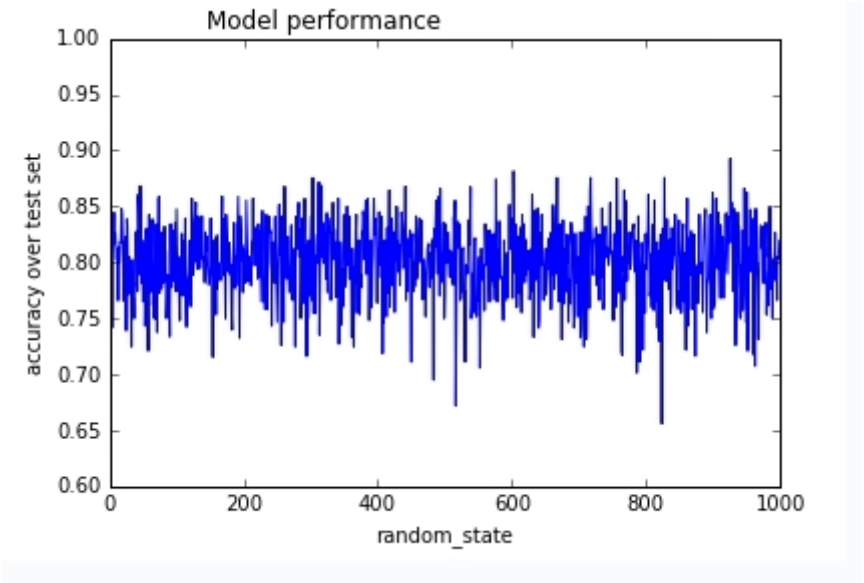
As a side comment, you choose a great stack of models to attack your problem!. In my opinion, GaussianNB is always a good choice since it doesn't require tuning and can reach decent results even if features are not fully independent, it could be your benchmark to explore further algorithms. Decision Trees are also a good choice since they provide feature importances which can help you to understand the predictive power associated with each feature and also an intuitive visualization of the decision criteria. Another right choice is the Logistic regression since there are just two classes, this classifier is quite robust and also provides probability estimates so you can understand the model confidence when labeling a student with a particular class. For more accurate results, but computationally more expensive, ensemble models are the right choice, even you can consider a Voting Classifier to combine the output of several classifiers!. As you can see, there are plenty of options!, it is a matter of the dataset characteristics and your main goals to achieve to select the right algorithms.

Student successfully implements a pipeline in code that will train and predict on the supervised learning algorithm given.

Excellent implementation of the pipeline to train and validate the different classifiers attempted.

Student correctly implements three supervised learning models and produces a performance visualization.

Well done testing the different classifiers attempted for different training sizes. However, make sure you define the `random_state` number for those algorithms that call for it. This would allow us to reproduce your results as well as validate the increase of performance after tuning since models are initialized with the same conditions. Note `random_state` affects the model performance, so for reproducibility purposes, it is required to define a particular value. In the example below, it is tested the same algorithm but using values for `random_state` in the interval `[0, 1000]`:



Improving Results

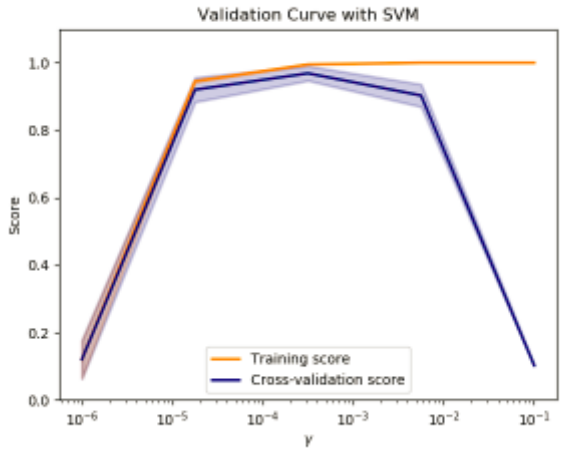
Justification is provided for which model appears to be the best to use given computational cost, model performance, and the characteristics of the data.

Nice work using your results, in terms of performance and computational cost, to justify the final choice.

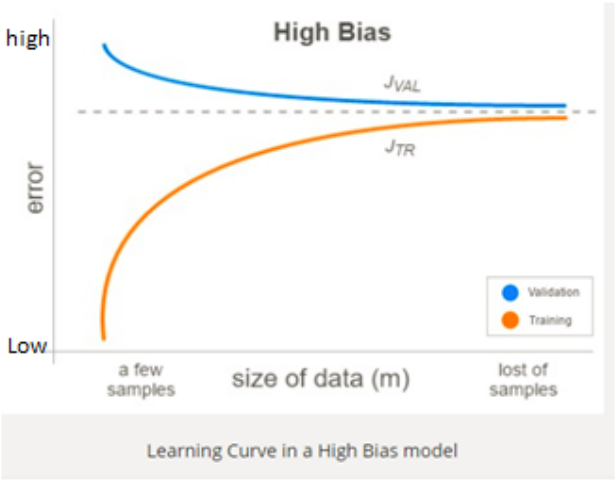
As a side comment, Scikit includes a whole chapter on validation.

Validation curves are used when it is helpful to plot the influence of a single hyperparameter on the training score and the validation score to find out whether the estimator is overfitting or underfitting for some

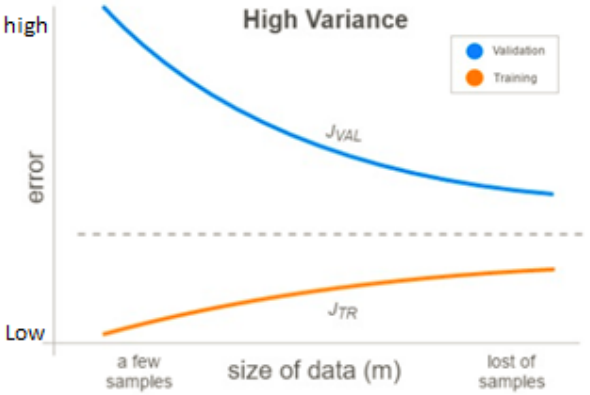
hyperparameter values.



On the other hand, learning plots are used to find whether our model is overfitted or underfitted. Basically, this is something that we did in this exercise by comparing the train versus test sets performances, if the performance on the train set is high but our test set score is significantly lower, the model is **overfitted**, that is, with the aim to maximize the performance on the train set, the model identified during the training phase those patterns that are particular just to this set and are not replicable on other sets (remember, a dataset is always an **approximation** of the whole population and is always biased). This scenario is a huge problem since the model learned these particularities that are just present in the train set and not those patterns that could make it generalize properly.



Learning Curve in a High Bias model



Learning Curve in a High Variance model

For more insights, check this great post on Learning curves.

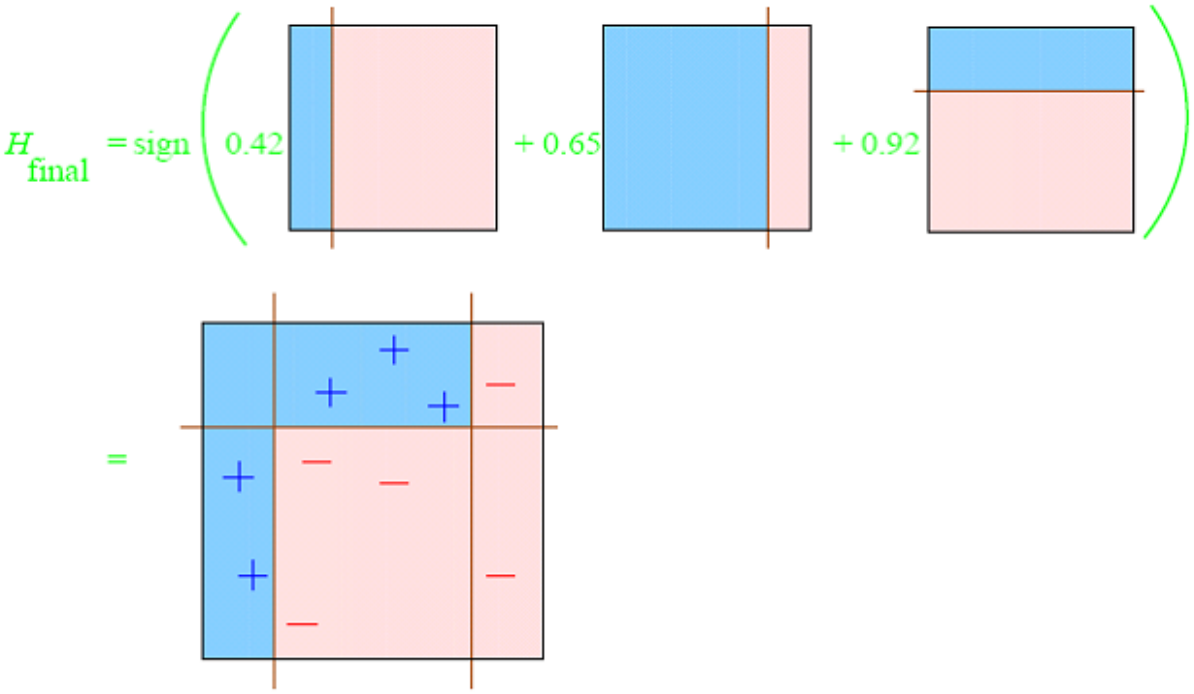
Student is able to clearly and concisely describe how the optimal model works in layman's terms to someone who is not familiar with machine learning nor has a technical background.

Although the basics of how AdaBoost works are well described, well done!. However, some more details are required to better understand how learning process happens during the training phase or how a new person is categorized based on his characteristics and the previously learned information, of course, these descriptions need to be provided in non-technical terms.

So, your response should address:

1. How the algorithm learns during the training phase
2. How the algorithm classifies a new person in the predictive phase

With regards to the "non-technical" description of the model, please have a look at this blog where it is provided a layman explanation of how Random Forest works. While this is a different algorithm as Adaboost is boosting and Random Forest bagging (this is a great threat to better understand the differences between bagging and boosting), I like this particular example since it demonstrates that using plain examples (there are no terms like "training", "weak learners", etc) to explain technical concepts is a wise option since the audience is familiar with such context (In this particular case, it would be great if you use this current scenario!). Note you can also include visualizations and use such visualizations to explain how the algorithm works:



For example, how would you explain this visualization to your audience in this particular context? what does sign represent? what the different colors?.... translating this image to the current context is the way to allow your agnostic audience to understand how the classifier works. Here is an example of an explanation of the image above.

The final model chosen is correctly tuned using grid search 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.

Nice work with GridCV, however, the reason for marking this section off is because `random_state` needs to be defined. Make sure you use the same value used when testing the three different algorithms. Note you want to compare the benefits of tuning, for such purpose you need to compare the same algorithms being initialized with the same random states.

Student reports the accuracy and F1 score of the optimized, unoptimized, models correctly in the table provided. Student compares the final model results to previous results obtained.

Good discussion of the performances obtained along the whole process.

Feature Importance

Student ranks five features which they believe to be the most relevant for predicting an individual's' income. Discussion is provided for why these features were chosen.

Well done summarizing those features that intuitively seem important, it seems individual's income level is a key factor.

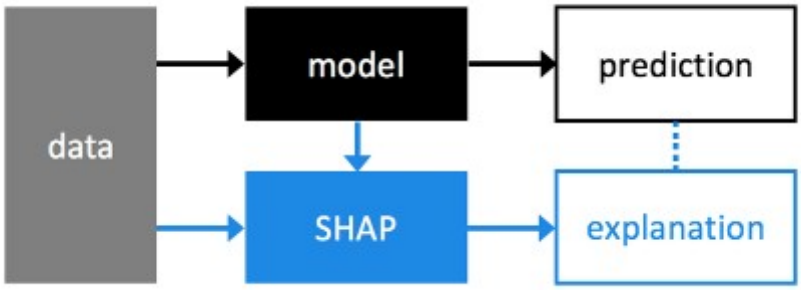
Student correctly implements a supervised learning model that makes use of the `feature_importances_` attribute. Additionally, student discusses the differences or similarities between the features they considered relevant and the reported relevant features.

Well done getting the feature importances. As you can see in this example, intuition about the feature importances in any ML problem is a good initial approach, but a thorough method, like the one here employed, is a better approach since its conclusions are based on the data relations between features and label. This is a general rule in any ML problem, feature selection is critical and should be mostly based on mathematical methods instead of intuition.

Student analyzes the final model's performance when only the top 5 features are used and compares this performance to the optimized model from Question 5.

Good analysis. Note feature selection process is key in Machine Learning problems, the idea behind it is that you want to have the minimum number of features that capture trends and patterns in your data. A good feature set contains features that are highly correlated with the class, yet uncorrelated with each other. Your machine learning algorithm is just going to be as good as the features you put into it. For that reason, this is definitely a critical step in any ML problem. In this case, there is no gain in terms of performance, but in terms of computational costs and model explicability, there is a significant gain!.

BTW, here more info on Shap.



RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH