UDACITY

Logo

Back to Machine Learning Engineer Nanodegree

Finding Donors for CharityML

REVIEW | CODE REVIEW | HISTORY

Requires Changes

2 SPECIFICATIONS REQUIRE CHANGES

Great work so far! I believe that you just need to polish your answers in a few sections to meet all the specifications of this project! Keep up the good work and all the very best for future projects 👍

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**

Great job! If you are just starting out with pandas, I would suggest this great resource for you!

https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python

Another suggestion is to use the python package `seaborn` for data exploratory exercise. For instance, try:

```
import seaborn as sns
sns.pairplot(data, palette='Set1', hue='income')
```

to observe the distribution of each feature across the two income classes as well as to observe the correlation between features. These exercises will allow you to understand the data better and start making decisions on how to obtain the best model for classification.

Preparing the Data

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

Great job here again! Isn't `pd.get_dummies()` a very neat function to get the encode categorical features!

Another option to encode income would be to use `replace()` function

Evaluating Model Performance

Student correctly calculates the benchmark score of the naive predictor for both

accuracy and F1 scores.

Good job! What you have implemented here is a naive predictor that always predicts a person to have income >50K. Naturally, this model has a low accuracy but a 100% recall as there are no `FN` produced by the model.

It is always a great idea to establish a benchmark in any type of problem. As these are now considered our "dumb" classfiier results, & any real model should be able to beat these scores, and if they don't we may have some model issues.

Also, you could view this as an exercise to appreciate the requirement of various evaluation metrics and how a certain metric is more suitable for a given problem. Sometimes accuracy/precision is more important than recall of vice versa. When both accuracy & recall are both important, you would look at the F-score, which is a harmonic mean of the precision, recall values.

You can refer this page on sklearn metrics to read about all the available metrics:

http://scikit-learn.org/stable/modules/model_evaluation.html#model-evaluation

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.

Good choice of models! There are however a few concerns over a couple of points. You state that:

> AdaBoost can be sensitive to noisy data and outliers. In some problems, however, it can be less susceptible to the overfitting problem than most learning algorithms.

You state that Adaboost can be sensitive to noisy data. However, with a large number of estimators in the ensemble, it can effectively combat noise and result in a model that is not overfit to the training data.

Also, for the reason to choosing this model, you mention that:

> Adaboost can use multiple instances of the same classifier with different parameters.

Adaboost using a combination of classifiers is a description of the algorithm itself. You would have to motivate your reasons as to why you are choosing it for this dataset. Further, you mention that:

> Thus, a previously linear classifier can be combined into nonlinear classifiers.

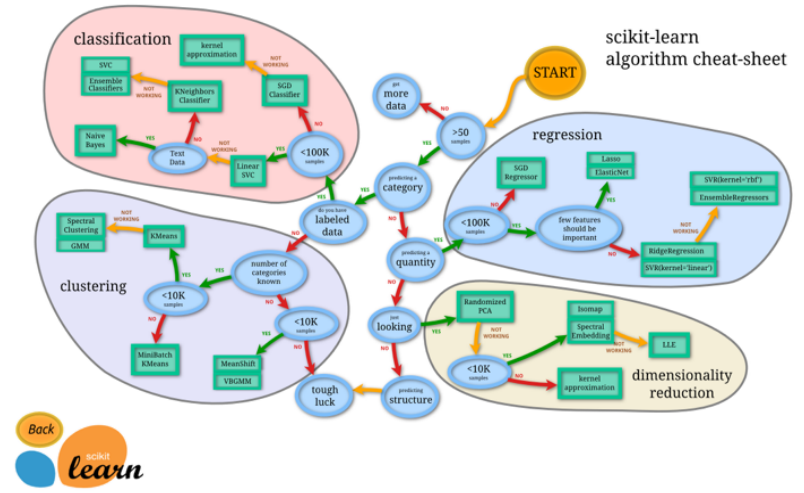
This is again applicable only to a base estimator that is linear. So if the base estimator is a decision tree, this explanation does not hold. What you could instead say is that, the combination of base estimators can result in a strong classifier that can deduce very complex and nonlinear decision boundaries as well.

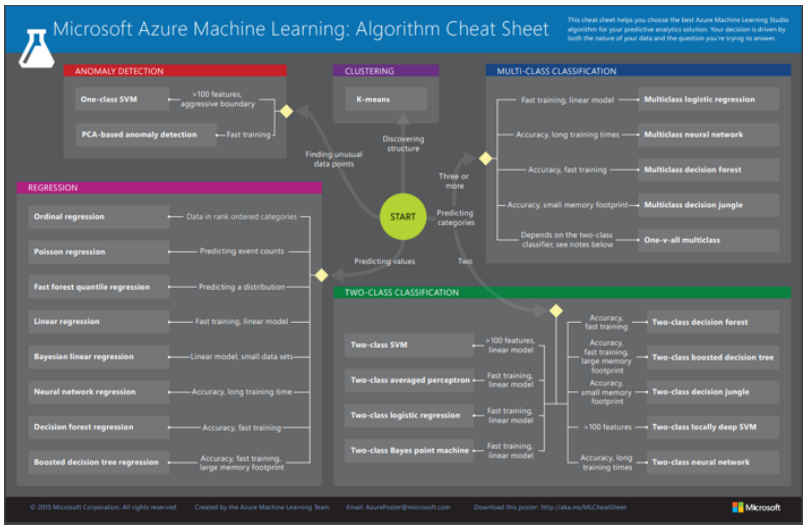
Also, with respect to SVM, you mention that:

> Because SVM is a suitable classifier for (1) large data with large number of features (variables)

SVM's biggest disadvantage is actually its computational complexity that scales as O(n3) where n is the number of free support vectors. So actually, it is not suitable for a large data with large number of features. Its suitability to this dataset can rather be connected to the powerfulness of the algorithm owing to its kernel trick and hyper parameters that can penalize overfitting.

I am attaching the following two resources that can guide you with machine learning model selection:





Rate this review

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

What you have implemented here is a function that acts as a pipeline to compare the different classifiers that you have chosen in the same manner (also avoiding any differences in the manner in which the models are trained). sklearn also has a pipeline class that allows you to build a pipeline to apply a series of scalers, reducers and estimators through just one pipeline object.

Refer this sklearn page for more information on this:

http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

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

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.

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.

Good work here! In order to further improve this answer, I have the following feedback for you:

You are spending a lot of time describing other classifiers to make your case that feature

selection is important. But you have not defined Adaboost until this point or even after that actually. So my suggestion is that you can instead introduce adaboost, describe its philosophy and then talk about its edge over the decision tree which is its base estimator in your description. Since you haven't done this the reader (a layman) does not get the idea of what this algorithm does.

Describing an ML model in layman terms is a very important skillset for a machine learning scientist and you may greatly benefit from perfecting this art! You have great ideas on how to relay the concepts, so with a little restructuring you can do really well here !

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.

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.

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.

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.

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.

Yes, as you mention, the reduced run time is definitely a good thing. But apart from that, the advantage of choosing a smaller feature space is that your model is less likely to overfit and therefore more stable. So it is always a great practice to include a feature selection exercise into your machine learning project. Often, this step comes a long way in improving the model performance even.
There are different methods available for feature selection. One is to apply algorithms like PCA, feature agglomeration which essentially projects the covariance of the feature space onto a smaller number of independent (principal) components. The other is to use algorithms like KBest, which tests a simpler algorithm on various subsets of features to decide which one would be the most suitable.
Please go through this awesome resource to know more about this:
https://machinelearningmastery.com/an-introduction-to-feature-selection/

RESUBMIT PROJECT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Student FAQ