UDACITY

PROJECT

Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Awesome job! I've left some tips and tricks below, but you've definitely demonstrated that you're ready to move on to the next section. Keep up the great work! Enjoy the rest of the program

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 work here!

You can also use the `value_counts` method to get your label counts.

```
n_at_most_50k, n_greater_50k = data.income.value_counts()
```

Preparing the Data

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

Great job implementing `get_dummies`!

TIP

It doesn't provide a ton of benefit in this situation, but it's definitely worth checking out the LabelEncoder class provided by sklearn. This class will do all the legwork in ordinally encoding your variables, so it's massively helpful when you have a ton of output classes 😄
You can apply it like so

```
encoder = LabelEncoder()
income = encoder.fit_transform(income_raw)
```

Evaluating Model Performance

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

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.

Really thorough discussion! You seem to have a pretty solid understanding of the models you've selected. I just have a couple model tips below to supplement your learning 😄

TIPS

DECISION TREES

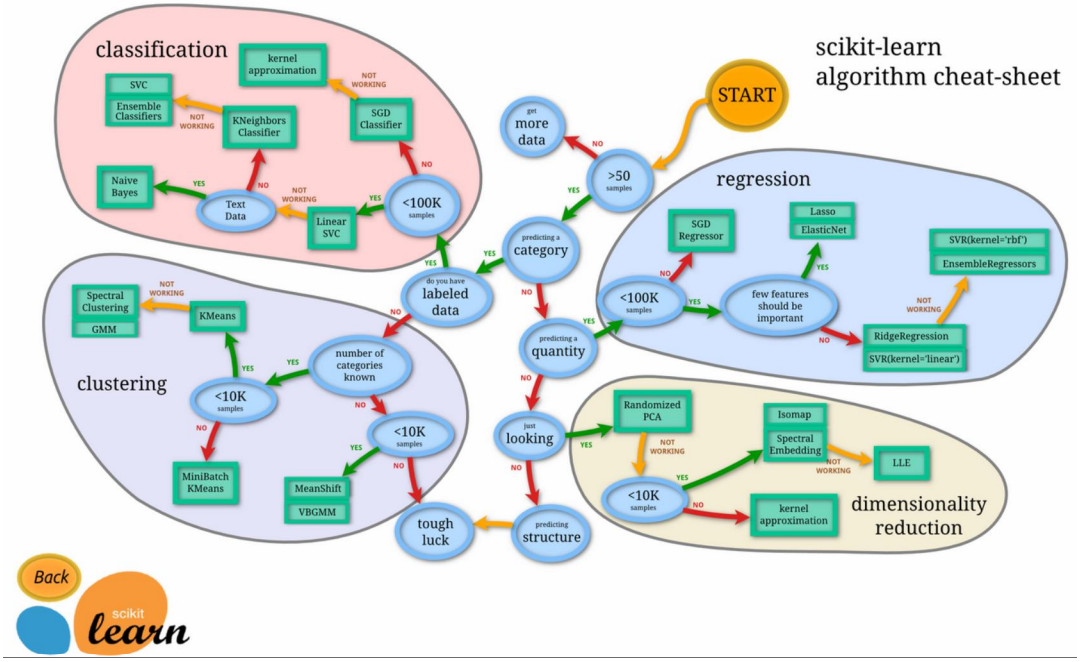
You're right that they can tend to suffer from over-fitting unless we're careful about tuning them. As you probably remember from the model validation project,the `max_depth` parameter can be a useful starting point for reducing overfitting.
Another popular method is pruning:
https://en.wikipedia.org/wiki/Pruning_(decision_trees)

Although pruning is not strictly supported in sklearn, tuning the `min_impurity_split` hyper-parameter serves is a great replacement.
http://blog.nelsonliu.me/2016/08/05/gsoc-week-10-scikit-learn-pr-6954-adding-pre-pruning-to-decisiontrees/

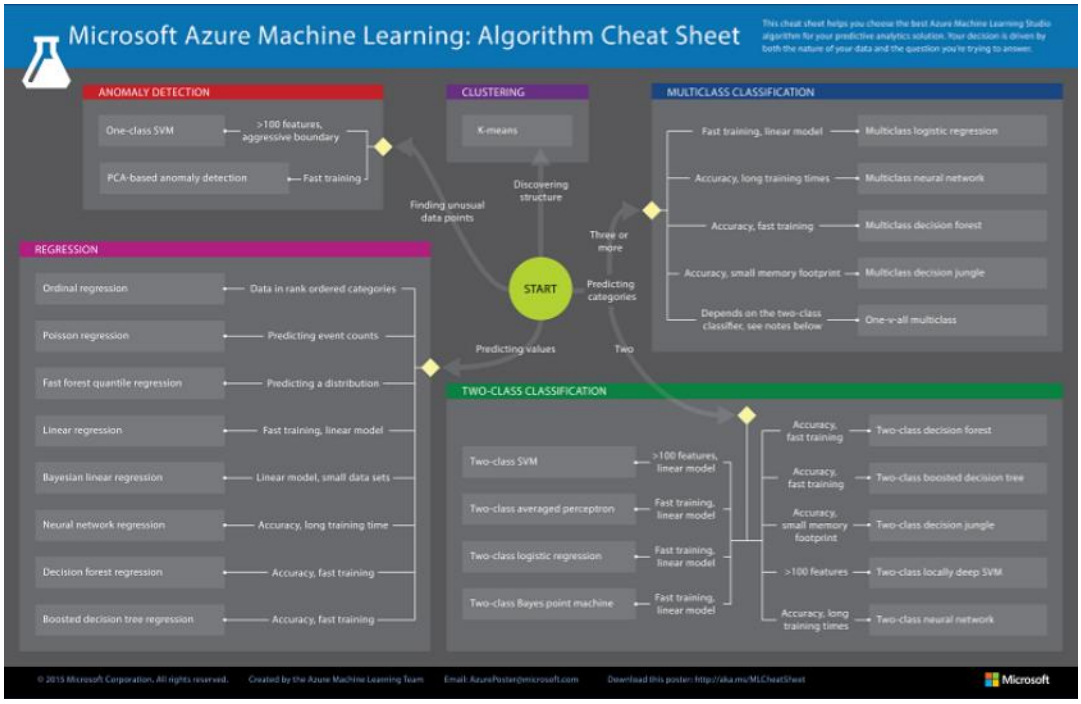
MODEL SELECTION

Even if we carefully analyze our data, we'll likely still have some level of trial and error involved in our model selection process. Unfortunately, there's no one-size-fits-all solution either. However, there are a few great heuristics we can use to really help narrow down the process! Check out the cheat sheets posted below 😄

Sklearn



Azure



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

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.

Really great work here, this is probably the hardest part of the project 👍

Being able to explain a model in simple terms is a really important asset in business, as many of the people you'll be working with won't have the technical background you do. For this reason you should try to get a similar handle on other popular ML models.
You can use the resources provided below to get a head start in this 😄
https://rayli.net/blog/data/top-10-data-mining-algorithms-in-plain-english/
http://blog.echen.me/2011/03/14/laymans-introduction-to-random-forests/
https://prateekvjoshi.com/2014/05/05/what-is-adaboost/
http://xgboost.readthedocs.io/en/latest/model.html

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.

Great work!

TIPS

Another search technique worth looking at is the RandomizedSearchCV. The difference between this method and the typical grid search method is that the Randomized Search actually doesn't search every value. Instead, it samples random subsets.

At first glance, you might wonder why we'd ever use this over a grid search, but it turns out that in practice this method often gets *very close* to an optimal value in a fraction of the time. So it's great when we have a ton of hyperparameters. Another benefit is that it can sample from value *distributions* rather than lists of pre-determined values.

If you're interested, I highly recommend reading the blog post below. It does a phenomenal job of illustrating why you'd want to use a Random Search over a Grid Search.
https://medium.com/rants-on-machine-learning/smarter-parameter-sweeps-or-why-grid-search-is-plain-stupid-c17d97a0e881

Student reports the accuracy and F1 score of the optimized, unoptimized, and benchmark 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.

Nice job! Your feature choices are logical and intuitive 👍

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.

Great work here! It's always great to try an intuitively analyze our data in addition to our raw quantitative analysis. Getting a good feel for the data is almost always a benefit in practice 😄

TIPS

There are a ton of other methods we can use to extract the feature importances in our data. You might remember some of the simpler univariate methods like SelectKBest and SelectPercentile.

We can also try more complex methods like Recursive Feature Elimination. This technique recursively cuts out the feature that contributes least to the model, and then re-evaluates the model after the elimination. This process repeats until we have the desired number of features.

If you're interested in checking this technique or others, take a look at the posts below
http://blog.datadive.net/selecting-good-features-part-iv-stability-selection-rfe-and-everything-

side-by-side/
https://topepo.github.io/caret/recursive-feature-elimination.html

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.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

Student FAQ Reviewer Agreement