UDACITY

PROJECT

Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

Overall really solid work!
There's just a couple sections that need some points to be revised. Keep up the good work, I look forward to the revision.

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.

Overall really good work. There are just a couple points that need to be re-visited in your analysis.

KNN

> I chose k-NN as my first model as it is one of the simplest and easy to use models.

This would certainly be the case if you were coding your algorithms from scratch. However, given that all of these algorithms are included in sklearn, they're all about the same in terms of how easy they are to implement.

LOGISTIC REGRESSION

In here, you mention "low variance" as a strength and "high bias" as a weakness. This isn't actually so clear-cut - in some situations you'd want a lot of bias, and in others a lot of variance. This is fairly problem-specific, so it's not really the case that one is necessarily better than the other, though there are other things to consider for each.

> I chose this model as a candidate for this problem based on the knowledge that our data has been cleaned and now has 2-labels (0 and 1) specifying whether the donor earns more that 50k or less.

This isn't really an argument for using logistic regression, as this is a benefit to all classification models.

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.

Fair points here. It's certainly true that higher-variance models like RF tend to get a bigger boost from hyperparameter optimization than high-bias models like logistic regression.

The only problem is that it's not really clear which model you choose - Logistic Regression or RF. It seems like you believe that RF might perform better after tuning, but logistic regression was implemented below. Just make sure it's clear which one you think is optimal for this problem.

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.

You've done a pretty good job of explaining the entire ML pipeline, but in this section you should explain how *logistic regression* works in detail. We don't really need information on this particular problem, data preprocessing or other. This means talking about how the model is trained (how it learns), and how we can predict on it. Ideally, you should be explaining it in terms that a layman can understand.

If you'd like some resources as a jumping-off point, check out the links below.
https://codesachin.wordpress.com/2015/08/16/logistic-regression-for-dummies/

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!

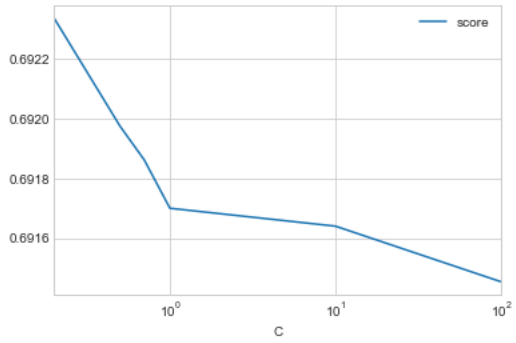
TIPS

One really interesting technique we can apply after a grid search is actually analyzing our model results. This sort of visualization can be really useful in trying to figure out if the value space we searched for our hyperparameters actually got us close to an optimum. It might be the case that we should search more values, and this sort of visualization can

give us insight as to where those values might lie 😄
This can be tricky with models that use lots of hyperparameters, but we can perform it here like so:

```
results = grid_fit.grid_scores_
results_df = pd.DataFrame([[r[0]['C'],r[1]] for r in results],
                          columns=['C','score'])

results_df.plot(x='C', y='score', logx=True)
```



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, 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.

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Student FAQ