UDACITY

‹ Back to Machine Learning Engineer Nanodegree

# Finding Donors for CharityML

REVIEW

CODE REVIEW

HISTORY

## Requires Changes

2 SPECIFICATIONS REQUIRE CHANGES

Dear student,

Great first submission 👍

I would recommend you to spend some more time understanding each of supervised models and try to explain them in simple terms. It will help you understand the model very well and you can easily explain it to others as well.

I have provided suggestions for each of required sections. I am sure the required changes won't take much time and it is worth your time.

Keep up the good work! I look forward to next submission.

## 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**

You correctly calculated the dataset statistics. Good work!

If you observed the statistics it is clearly indicating the classes (individuals with income > $50k = 11208
and individuals with income atmost $50k = 34014) are imbalanced. Please check this link to understand how to deal with imbalanced data.

## Preparing the Data

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

Nice implementation using `get_dummies` method.

There are a couple other ways that we can encode the labels here.
One way is to use boolean indexing.

```
income = (income_raw == ">50K").astype(np.uint8)
```

We can also use the LabelEncoder class provided by sklearn. This class is especially useful when we have lots of possible output labels. We can use it for this problem as follows:

```
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.**

You correctly calculated both accuracy and f-score. Good work!

You could check this link for further understanding precision and recall.

**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.**

We can apply any classification algorithm if the outcome is binary, But here the question is what makes you choose that particular algorithm for the given dataset?

Have you observed any dataset properties by choosing this particular model? You need to answer this question in both dataset properties as well as in model strengths.

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

Nice implementation of pipeline!

**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.**

Nice explanation and I agree with your reason of selection of final model.

**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.**

Please explain your model in layman's terms. If you have included any technical terms please explain them clearly. You can assume you are explaining it to a completely unknown business person who have zero knowledge on machine learning.

The combination of weak learners resulting in a strong learner description is accurate, and it does give some information. However, it leaves out some important points, such as

- What is a decision tree?
- what is a weak learner? How it is different from random guessing?
- How weak learners are combined in boosting?
- How boosting is used for training and prediction on given census data?

These questions justify your understanding of your final model. Here is a sample explanation of weak learners :

## What is A Good Weak Learner?

A weak learner is any machine learning algorithm that gives better accuracy than simply guessing. For instance, if you are trying to classify animals at a zoo, you might have an algorithm that can correctly identify zebras most of the time, but it simply guesses for any other animal. That algorithm would be a weak learner because it is better than guessing.

If you had an algorithm that identified every animal as a zebra, then that probably is not better than guessing and so it would not be a weak learner.

For boosting problems, the best kinds of weak learners are ones that are very accurate, even if it is only over a limited scope of the problem. For instance the algorithm that correctly identifies zebras would be good. It allows you to confidently identify as least most of the zebras, allowing other weak learners to focus on the remaining animals.

## How Are Weak Learners Combined?

Boosting algorithms typically work by solving subsections of the problem, by peeling them away so future boosting iterations can solve the remaining sections.

Imagine you are hiring people to build your house, and you have 10 different big jobs that need to be done. A great way of doing it would be to get someone who is really good at foundations to build the foundation. Then hire a very good carpenter to focus on the framing. Then hire a great roofer and plumber to focus stage, a small subsection of the project is getting completely solved.

The takeaway is that weak learners are best combined in a way that allows each one to solve a limited section of the problem. Any machine learning routine can be used as a weak learner. Neural nets, support vector machines or any other would work, but the most commonly used weak learner is the decision tree

Please check this link to understand why decision trees are used as weak learners.

Now your turn to explain how boosting works and how you should use boosting for training and prediction for given census data. Make sure you use your own terms and explain in simple terms.

**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 implementation using grid search!

**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 analysis and great selection of features.

**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.**

An alternative feature selection approach consists in leveraging the power of Recursive Feature Selection to automate the selection process and find a good indication of the number of relevant features (it is not suitable for this problem because that is not what is required by the project rubric, though it is generally a very good approach).
http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.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