UDACITY Logout

PROJECT

Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

## Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Dear student,

well done improving your submission and addressing the last remaining issues, you made it for an excellent project. There's not much I can actually further add to your good work though, if you are enthusiastic about process optimization, when it comes to machine learning, I left a Pro Tip for you in the grid search section. Please be advised that it is quite advanced, therefore don't worry if you don't get it immediately, you could keep it and use it later on as you progress in the Nanodegree.

PS if you're an algorithm enthusiast I've left a Pro Tip regarding XG boost as well (a very popular algorithm at Kaggle) you'll find it in the algorithm selection section.

Congratulations on passing your exam!

### 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

Pro Tip: Data assessment:

Though it might be not necessary here, when dealing with the new data-set, it is good practice to assess its specific characteristics and implement the cross validation technique tailored on those very characteristics.
If the dataset is **unbalanced**(some classes are overrepresented compared to others) we could address the unbalanced nature of our data set using Stratified K-Fold and Stratified Shuffle Split Cross validation, as stratification is preserving the preserving the percentage of samples for each class.
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

As for the initial train test split you can obtain stratification by simply using `stratify = income`:

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, income,stratify = income, test_size = 0.2, random_state=42)
```

### Preparing the Data

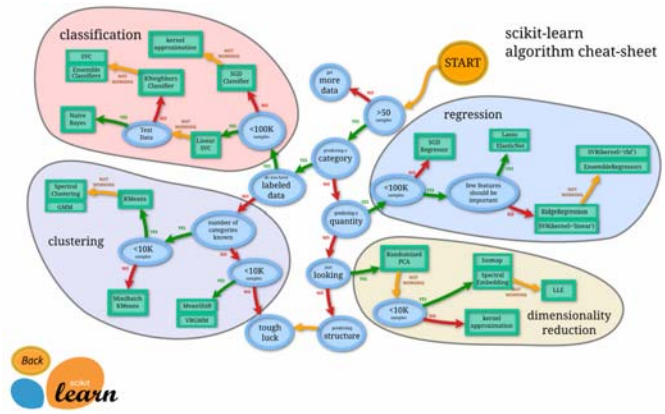
Student correctly implements one-hot encoding for the feature and income data.

### 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.

Pro Tip:
When choosing which algorithm to use this interactive map provides a good starting point: http://scikit-learn.org/stable/tutorial/machine_learning_map/




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.

Pro Tip (Advanced): Xgboost, one of Kaggle's top algorithms.
In the recent years one algorithm emerged as favourite in the machine learning community, it is actually one of the most used in Kaggle: Xgboost.
Here you can find an informative discussion on why that is the case: https://www.quora.com/Why-is-xgboost-given-so-much-less-attention-than-deep-learning-despite-its-ubiquity-in-winning-Kaggle-solutions
The algorithm is not available sci-kit learn, here is how you can start working with it:
http://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/

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.

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.

Pro Tip (Advanced): You could actually go well beyond grid search and implement 'pipelines' where the whole machine learning process becomes 'grid-searchable' and you can parameterize and search the whole process though cross validation.
http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
And yes you can try out several algorithms automatically as well too! Watch out though this is pretty advanced stuff, here is a great, informative, top notch tutorial from Zac Sewart!
http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html

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.

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.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ

Rate this review