

PROJECT

Finding Donors for CharityML  
A part of the Machine Learning Engineer Nanodegree Program


PROJECT REVIEW

CODE REVIEW

NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

 Excellent job with the report, I'm impressed with how you've grasped the main concepts of supervised learning and completed all the project specs with your first submission. 😎


For an excellent guide to approaching almost any ML problem, you can also check out this [blog post by a Kaggle grandmaster](#).

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 getting the dataset stats!

 Tip: imbalanced target classes

As you can see we have an [imbalanced proportion](#) of individuals making more than \$50k vs those making less, and will want to make sure the metric we're using for model evaluation is capturing how well the model is actually doing. (more info [here](#))

In this project we use the precision, recall, and F-beta scores, but we could also consider [F1 score](#) (which is equivalent to using F-beta with `beta=1` ).

Preparing the Data

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

Good job encoding the features and target labels. 😎

We can also use sklearn's [LabelEncoder](#) to convert the non-numeric target labels. This would come in handy for performing [multi-class](#) predictions...

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
income = le.fit_transform(income_raw)

# if needed we can reverse the transform
print (le.inverse_transform(income))
```

Evaluating Model Performance

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

Terrific work calculating the accuracy and F-score of a naive predictor — this will serve as a useful benchmark to evaluate how well our model is performing.

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 discussion of the 3 models and why you chose them. 😊

Rate this review

https://review.udacity.com/?utm\_medium=email&utm\_campaign=review...bsft\_txnid=926921c7-ae86-4d94-9647-007a3fa4583d#!/reviews/836471

Page 1 of 3

- In general, with [model selection](#) it's a good idea to try out simpler methods like Logistic Regression or Naive Bayes as a benchmark, and then move on to [non-linear classifiers](#) such as your choice of SVM, Decision Trees, and Adaboost.
- For additional info you can also check out this [microsoft azure guide](#) on choosing an algorithm, or [this one from SAS](#).

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

Nice job in general creating the pipeline, but we want to calculate the [fbeta\\_scores](#) by using the default `average='binary'` setting — this will give us a score for just the positive `1` labels...

```
# Compute F-score on the the first 300 training samples
results['f_train'] = fbeta_score(y_train[:300], predictions_train, beta=0.5
)

...
```

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

Great job generating the results with the 3 sample sizes, and setting random states on the classifiers to make your results reproducible.

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.

Good justification of your model choice by looking at factors such as the accuracy/F-scores and computational cost/time. 😊

Adaboost is a good option to use here, and it's a good bet that we can also improve the model's performance even further with some parameter tuning.

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.

Nice discussion of the adaboost model in a way that's understandable by a non-technical audience. Well done!

For more ideas on describing common ML models in plain english, you can also check out these descriptions:  
<https://rayli.net/blog/data/top-10-data-mining-algorithms-in-plain-english>

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.

Excellent job tuning the model, and it's also great that you outputted the best parameters found with `print(grid_fit.best_score_)`. 😊

- You can also look at the scores of each of the parameter settings with `grid_obj.grid_scores_` or `grid_obj.cv_results_` ...

`display(pd.DataFrame(grid_obj.cv_results_))`

- If you keep the default `cv` parameter on the grid search object `grid_obj`, the grid search will default to a stratified K-Fold cross validation with 3 folds.

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.

Good work discussing the tuned model, and congrats on improving the accuracy and F-score. 😊

For future reference, another common parameter tuning approach to try out is [randomized search](#) — with sklearn's `RandomizedSearchCV` method you can often get [comparable results](#) at a much lower run time.

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 rankings of the features you think are important — all of them seem to be closely related to an individual's income level.

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.

Excellent job extracting the feature importances of the gradient boosting model — you could have also simply used the feature importances of the `best_clf` model we just tuned...

```
# Extract the feature importances
importances = best_clf.feature_importances_
...
```

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.

Good examination of the model's performance with the reduced feature set — the results are clearly worse, but we could always try adding back just a few more "important" features to see if the accuracy improves. (e.g., try using 8 or 10)

```
# Reduce the feature space
N = 8 # choose number of features to use

X_train_reduced = X_train[X_train.columns.values[(np.argsort(importances)[::-1])[:N]]]

X_test_reduced = X_test[X_test.columns.values[(np.argsort(importances)[::-1])[:N]]]

...
```

 [DOWNLOAD PROJECT](#)

[RETURN TO PATH](#)

[Student FAQ](#)