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

6 SPECIFICATIONS REQUIRE CHANGES

All in all this is a very good submission and you only have a few minor adjustments to make in order to meet all the specs.

You're very close to completion, so keep at it! 😃

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!

Note: look at 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.

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 convert the `income` target labels to numerical values with `get_dummies` ...

```
pd.get_dummies(income_raw)['>50K']
```

or...

```
pd.get_dummies(income_raw, drop_first=True)
```

Evaluating Model Performance

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

re: Question 1
Nice work using the provided F-beta formula, but take another look at your definition of `precision` and `recall` , and also make sure the scores are calculated on the *full* dataset size rather

than just the test data (`X_test` , `y_test`).

For example, here's one way to approach it...

```
# generate accuracy using the stats calculated earlier
accuracy = greater_percent / 100.

# TODO: Calculate F-score using the formula above for beta = 0.5
precision = accuracy
fscore = (1 + <<INSERT BETA VALUE>>**2) * precision * <<INSERT RECALL VALUE>> /
         (<<INSERT BETA VALUE>>**2 * precision + <<INSERT RECALL VALUE>>)
```

✓

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to be explored.

Good discussion of the models and why you chose them! As you've discovered, there are several issues to consider in choosing the best machine learning algorithm for your problem, and it's not always easy to know which model to use.

With model selection it's often a good idea to try out simpler methods like Logistic Regression as a benchmark (or Naive Bayes, as you've chosen here), and then move on to non-linear classifiers such as your choice of Decision Trees and KNN.

Further reading:
For additional info you can check out the udacity blog for more about the general applications of machine learning, and this microsoft azure guide on choosing an algorithm.

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

Required: getting predictions & scores
Good work in general creating the pipeline, but be sure to generate the predictions and compute the accuracy/fbeta scores on the *first 300* training points rather than the *full* training set...

```
# TODO: Get the predictions on the test set,
#       then get predictions on the first 300 training samples
...
predictions_test = learner.predict(X_test)
predictions_train = learner.predict(X_train[<<INSERT SUBSET>>]) # TODO: use 300 pts
...

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

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

Required: set a random state
Great work fitting the models, but to make the results reproducible by others, we want to also set random states for any classifiers where they can be specified.

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 choice by looking at factors such as the models' accuracy/F-scores and computational cost/time — Decision trees (and tree-based ensembles) are a good option to use here because we can address any tendency to overfit 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.

re: Question 4
Be sure to include an answer that describes how decision trees work in laymen terms, including how it's trained and how it will be used to make predictions. The level of discussion should be for a non-technical audience like CharityML that may not even know what "features" are.

See below for ideas on explaining the model:

- https://www.quora.com/What-is-an-intuitive-explanation-of-a-decision-tree
- https://algobeans.com/2016/07/27/decision-trees-tutorial/
- 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.

Required: set a random state
Great job tuning the model, but to keep the comparison between the tuned and untuned models consistent, don't forget to also specify the same random state on the `clf` object that was used in the earlier model training.

Suggestion: You might also experiment with improving the score by tuning other parameters. For example...

```
parameters = {'min_weight_fraction_leaf': [0, .1, .5],
              'max_features': [None, 'sqrt', 'log2'],
              'max_depth':[5,8,15,None],
              'min_samples_split':[2,5,10]
              }

# The default grid search uses 3 folds; use the 'cv' param to specify more
grid_obj = GridSearchCV(clf, parameters, cv=5,
                        scoring=scorer, n_jobs=-1)
```

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.

re: Question 5
The structure of your answer looks good, so after adjusting the calculation of the naive predictor earlier in the notebook, just update the table values here for the Benchmark Predictor.

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. Let's see if we can verify this with feature importances...

✓

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 model, although it might be interesting here to simply use 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 — it appears that the results compare somewhat well with those generated using the full list of features.

Feature reduction is a great way to fight the curse of dimensionality.

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)

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Rate this review

☆☆☆☆☆

Student FAQ