PROJECT

Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

Requires Changes

SHARE YOUR ACCOMPLISHMENT

1 SPECIFICATION REQUIRES CHANGES

Excellent progress with this submission! You've done a very good job with the report so far and only have a minor adjustment to make in order to meet all the specs.

You're just about there, so stick with it! 😉

Rate this review

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**

We can also simplify the implementation a bit with...

```
n_greater_50k = sum(data['income'] == '>50K')
n_at_most_50k = sum(data['income'] == '<=50K')
```

This calculates the stats directly from our existing dataframe rather than having to create a new dataframe — the performance gain is small, but could become more important with larger datasets.

Preparing the Data

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

Pro tip: value counts
In case you wanted to examine how many different values there are for the non-numeric features before encoding them, we could take a look with value_counts...

```
# look at first few non-numeric columns
for col in data.columns[:5]:
    if data[col].dtype == 'O':
        display(data[col].value_counts())
```

```
Private             33307
Self-emp-not-inc     3796
Local-gov            3100
State-gov            1946
Self-emp-inc         1646
Federal-gov          1406
Without-pay            21

...
```

Evaluating Model Performance

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

DONE!

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

Pro tip: check out xgboost
It's outside the scope of this project, but if you haven't checked it out already, xgboost is a popular version of the tree-based ensemble method gradient boosting that's being used extensively in kaggle competitions.

Here are some guides to get you started:

- https://jessesw.com/XG-Boost/ (uses our census income dataset)
- http://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

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

Required:
Good work in general setting up the pipeline, but be sure to fit the `learner` to slices of the training data using `sample_size`...

```
# TODO: Fit the learner to the training data using slicing with 'sample_size'
learner.fit(X_train[:<<INSERT VARIABLE>>], y_train[:<<INSERT VARIABLE>>])
```

Then later, we'll get the predictions on just the first 300 training points...

```
# 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
...
```

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

DONE!

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 — Logistic regression is a good option to use here for its simplicity, and it's possible 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.**

DONE!

✓ **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 "\n",best_clf`. Bravo! 😎

Pro tip: Look at grid scores
You can also look at the scores of each of the parameter settings with `grid_obj.grid_scores_` (or `grid_obj.cv_results_` in sklearn 0.18)...

```
from IPython.display import display
display(pd.DataFrame(grid_obj.grid_scores_))
```

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

You can change the number of folds by setting the `cv` parameter with an integer...

```
# 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.**

If you're interested, you could also try different regularization methods (L1, L2) to see how they generate different results — see this quora thread for more info.

```
parameters = {
    'C': np.logspace(-2, 2, 13),
    'penalty':['l1', 'l2']
    }
```

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.**

Good work extracting the feature importances of the model, but be sure to set a `random_state` on the classifier so that your results are reproducibly by others.

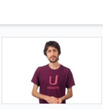
✓ **Student analyzes the final model's performance when only the top 5 features are used and compares this performance to the optimized model from Question 5.**

Good examination of the model's performance with the reduced feature set — the results are clearly worse than those generated using the full list of features, although this might be acceptable if we had a much larger dataset and computational cost were a bigger issue.

Other methods you can try out in scikit-learn for performing feature selection include recursive feature elimination (RFE) and SelectKBest.

RESUBMIT PROJECT

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