UDACITY

PROJECT

# Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

## Meets Specifications

CONGRATULATIONS!

Outstanding job with the report, I'm impressed with how you've iterated on the project.

For another look at how you could approach a Python data analysis, you can step through this example ML notebook. And if you haven't been recommended Sebastian Raschka's Python ML book yet, it's definitely worth a look. 😎

### 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.**

**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

...
```

---

See more tips about using python in this pandas cheat sheet and the below links:

- http://dataconomy.com/14-best-python-pandas-features/
- https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python
- http://www.labri.fr/perso/nrougier/from-python-to-numpy/

## 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.**

DONE!

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

Excellent work implementing the pipeline! 😃

- With this pipeline you can see how the performance of the 3 models changes when using different training sizes passed to the `sample_size` parameter.
- You could also experiment on your own with making predictions on the training set using `sample_size`, but for the sake of speed we only use the first 300 training points here.

**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 — Random Forest 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.

(FYI, if you're interested in the best predictive performance, the highest scores I've seen generated by students used either Gradient Boosting or AdaBoost)

**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 random forest model, although the explanation could probably still be expanded for a non-technical audience. (e.g., explain what "decision trees" are)

Regardless, the discussion demonstrates that you understand the basic principles of random forests, which is the purpose for having you write it.

---

For further ideas on describing common ML models in plain english, see here:
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 best_params`. 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 `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, and benchmark models correctly in the table provided. Student compares the final model results to previous results obtained.**

Pro tip: randomized search
For future reference, another common parameter tuning approach to try out is randomized search. With something like sklearn's `RandomizedSearchCV` method you can often get comparable results at a much lower run time.

## Feature Importance

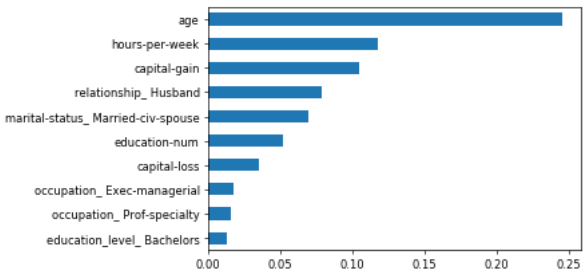
**Student ranks five features which they believe to be the most relevant for predicting an individual's' income. Discussion is provided for why these features were chosen.**

DONE!

**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.**

To explore beyond the 5 most important features, you can try using pandas plotting to look at some of the other feature importances as well...

```
# look at top "n" feature importances
n = 10
fi = model.feature_importances_
pd.Series(fi, index=X_train.columns).sort_values()[-n:].plot(kind='barh');
```




---

Read more on feature selection techniques here:
http://machinelearningmastery.com/feature-selection-machine-learning-python/

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.

DONE!

DOWNLOAD PROJECT

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