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

3 SPECIFICATIONS REQUIRE CHANGES

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

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

Exploring the Data

✓

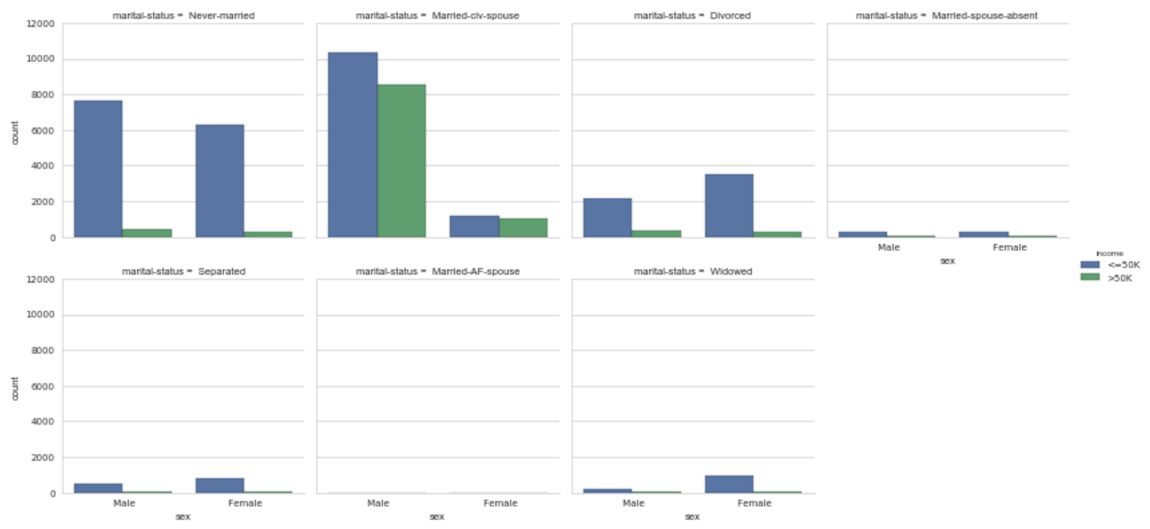
Student's implementation correctly calculates the following:

- **Number of records**
- **Number of individuals with income >$50,000**
- **Number of individuals with income <=$50,000**
- **Percentage of individuals with income > $50,000**

Pro tip: visualizing data
Although the project template doesn't go into exploratory data analysis (EDA) that much, it's usually a big part of any data analysis project.

One nice method to try is factor plots in seaborn. For example, below you can see the count of individuals making less than or more than 50K, with a breakdown by the features `sex` and `marital-status` ...

```
import seaborn as sns
sns.set(style="whitegrid", color_codes=True)
sns.factorplot("sex", col="marital-status", data=data, hue='income', kind="count", col_wrap=4);
```



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.

re: Question 1
Nice work using the full dataset size, but it looks like your calculation is mixing up the values for `precision` and `recall`...

```
# TO FIX: reassign the below variables
precision = tp_cnt / (tp_cnt+fp_cnt)  # TODO: precision value should be 0.2478
recall = tp_cnt / (tp_cnt+tn_cnt)  # TODO: recall value should be 1.0
```

✓

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.

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

You can also take a look at the specific values of the training results by displaying them in a dataframe...

```
for i in results.items():
    print i[0]
    display(pd.DataFrame(i[1]).rename(columns={0:'1%', 1:'10%', 2:'100%'}))
```

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.

DONE!

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
Nice start explaining the decision tree algorithm, but make sure the discussion is comprehensive enough for a non-technical audience to know what's going on.

Remember that your audience won't even know what features and classes are, so it would help to describe that our tree is trained with census information we already have about individuals and their known incomes, and then how it will predict incomes for new individuals.

For example...

1. The Decision Tree takes census data about individuals called "features" (age, gender, etc), and uses them to create a set of rules that can classify those with incomes over 50K vs those under 50K.
2. By looking at these features and determining which ones are important in predicting incomes, it creates a flowchart or "tree" of yes/no questions that... *<<ADD YOUR OWN DISCUSSION>>*
3. For example, the model could find that if an individual is male, then... *<<ADD YOUR OWN DISCUSSION>>*
4. Using this tree of decision points created with individuals census data we already know about, we can look at new individuals and... *<<ADD YOUR OWN DISCUSSION>>*

Again, here are some links for further 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.

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

re: Question 5
The structure of your answer looks good, so 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.

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.

DONE!

✓

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!

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