PROJECT

Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

SHARE YOUR ACCOMPLISHMENT

You're almost there! Just one minor adjustment remains, so keep at it! 😃

Rate this review

Exploring the Data

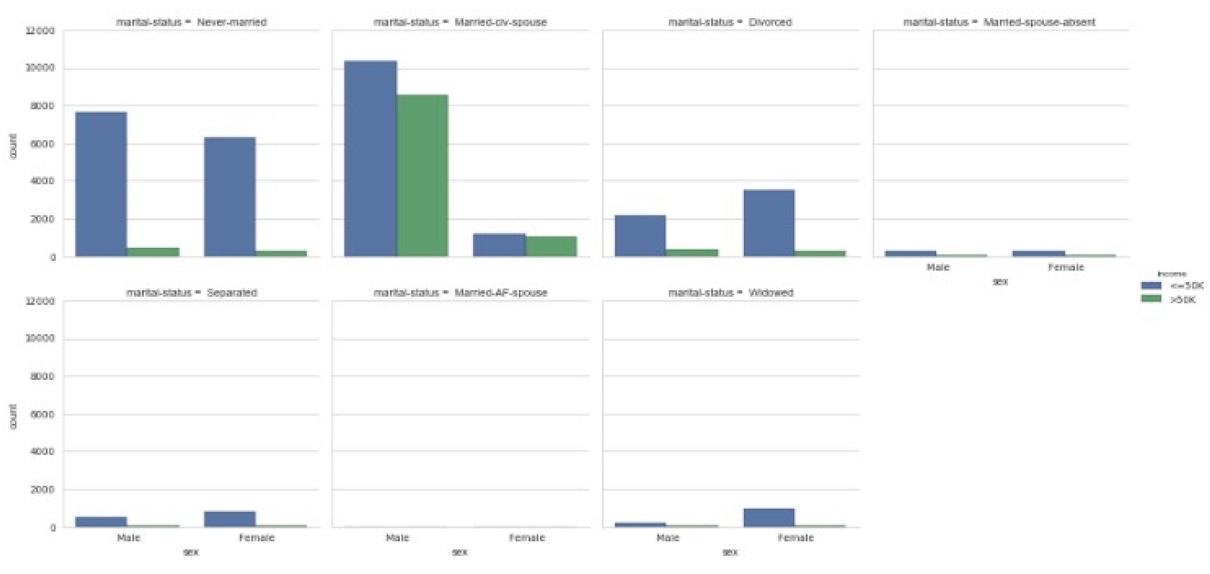
✓ **Student's implementation correctly calculates the following:**

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

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.**

For more tips about using python, check out 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.**

DONE!

✓ **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.

Required: getting predictions
Good work in general creating the pipeline and fitting the `learner` with slices of the training data using `sample_size`.

Just be sure to compute the prediction 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 VALUE>>]) # TODO: use 300 pts
```

```
# Compute accuracy on the first 300 training samples
results['acc_train'] = accuracy_score(y_train[<<INSERT VALUE>>], predictions_train)
```

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

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

DONE!

Improving Results

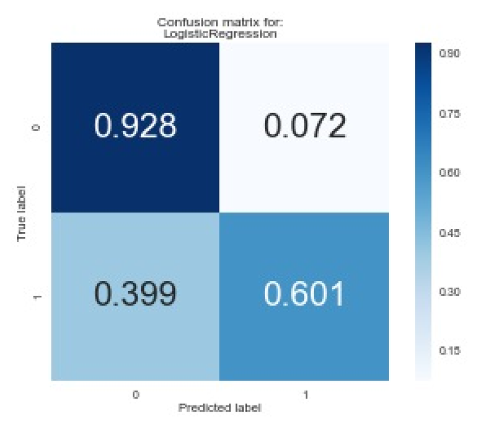
✓ **Justification is provided for which model appears to be the best to use given computational cost, model performance, and the characteristics of the data.**

To further justify your model choice, you can get a closer look at how the models are predicting the labels by plotting a confusion matrix using seaborn...

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
sns.set(style="whitegrid", color_codes=True)
import matplotlib.pyplot as plt

# Compute confusion matrix for a model
model = clf_C
cm = confusion_matrix(y_test, model.predict(X_test))
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] # normalize the data

# view with a heatmap
sns.heatmap(cm, annot=True, annot_kws={"size":30},
            cmap='Blues', square=True, fmt='.3f')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title('Confusion matrix for:\n{}'.format(model.__class__.__name__));
```



✓ **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.**

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.

✓ **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.**

DONE!

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.**

Note: LR odds ratio
If we wanted to interpret the "important features" for a model such as Logistic Regression, we could look at the Odds Ratio (OR) for the coefficients...

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression().fit(X_train, y_train)

coefs = model.coef_.ravel()
coefs = pd.DataFrame({'Coef': coefs, 'Odds Ratio': np.exp(coefs)})
coefs.index = X_train.columns

display(coefs.sort_values('Odds Ratio', ascending=False)[:10])
```

```
                                    Coef    Odds Ratio
capital-gain                   18.590313  1.184870e+08
capital-loss                    2.657013  1.425365e+01
hours-per-week                  2.609884  1.359747e+01
education-num                   1.682740  5.380277e+00
age                             1.673601  5.331333e+00
...
```

✓ **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 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