Finding Donors for CharityML

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

Very impressive submission here, as you have good understanding of these techniques and you now have a solid understanding of the machine learning pipeline. Check out corresponding sections based on your submitted comment.
Hopefully you can learn a bit more from this review and wish you the best of luck in your future!
Here are a few more resources you might find useful for further learning.

- Exploring Supervised Machine Learning Algorithms
- Supervised Learning With Python
- Practical Machine Learning Tutorial with Python Introduction
- Beginner's Guide to Machine Learning with Python

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

Preparing the Data

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

Evaluating Model Performance

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

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

Please list all the references you use while listing out your pros and cons.

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

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

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.

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.

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.

Great use of GridSearch here! GridSearch is not the only technique available to us though! Another similar technique
worth looking is RandomizedSearchCV
With an unbalanced dataset like this one, one idea to make sure the labels are evenly split between the validation sets a
great idea would be to use sklearn's StratifiedShuffeSplit

Student reports the accuracy and F1 score of the optimized, unoptimized, models correctly in the table provided. Student compares the final model results to previous results obtained.

Good work comparing your tuned model to the untuned one. The scores of the optimized model are relatively the same as
that of the unoptimized model, as evidenced by the table above.
We could also examine the final confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or
"classifier") on a set of test data for which the true values are known

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
%matplotlib inline
pred = best_clf.predict(X_test)
sns.heatmap(confusion_matrix(y_test, pred), annot = True, fmt = '')
```

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.

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.

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.

DOWNLOAD PROJECT

RETURN TO PATH