UDACITY

Return to Classroom

# Finding Donors for CharityML

REVIEW

CODE REVIEW

HISTORY

## Meets Specifications

Dear student

Great start on this project! You've clearly understood the material from the tutorials and you've done a great job applying it to this real-world dataset. Congratulations on passing quickly and good luck with the next section of the course.

Cheers!

Suggested reading for how to code a decision tree model from scratch:

https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/

You might also want to read up on how a randomized search cross validation works:

https://jamesrledoux.com/code/randomized_parameter_search

This can allow you to search large parameter spaces very efficiently.

## 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

```
Total number of records: 45222
Individuals making more than $50,000: 11208
Individuals making at most $50,000: 34014
Percentage of individuals making more than $50,000: 24.78439697492371%
```

Perfect!

## Preparing the Data

✓

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

```
income = (income_raw == ">50K").astype(int)
```

Nice job! You can also use a lambda function to encode the labels too:

```
income = income_raw.apply(lambda x: 0 if x == '<=50K' else 1)
```

## Evaluating Model Performance

✓

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

> Naive Predictor: [Accuracy score: 0.2478, F-score: 0.2917]

One interesting aspect to this predictor is that precision is equivalent to accuracy, and recall is always one. Hence a simpler implementation:

```
accuracy = n_greater_50k / n_records
fscore = (1.25) * accuracy / (0.25 * accuracy + 1)
```

✓

**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.**

Great job explaining and justifying your choice of models!

✓

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

Well done constructing a pipeline for training and tuning your models!

✓

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

```
clf_A = GaussianNB()
clf_B = GradientBoostingClassifier(random_state=10)
clf_C = LogisticRegression(random_state=10, max_iter = 300)
```

Nice job setting the `random_state` parameter so that each model outputs the same result each time it is run.

## 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.**

> Out of all the models tested, the Gradient Boosting algorithm is the most appropriate for the task of identifying individuals that make more than $50,000.

I think that you've done a really good job justifying that GradientBoosting is the best tradeoff between speed and performance. None of these models are particularly computationally expensive, so it makes sense to pick the model that has the best accuracy/F scores.

✓

**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 Gradient Boosting classifier uses a combination of multiple simple models to create one large well performing model. This is similar to how in a company, the individual employees each have certain strengths and weaknesses, but when working together they can compensate for each others weaknesses. Similarly to how a new employee with his own strengths needs to be hired when the current combination of employees is unable to adequately handle the task, when training the Gradient Boosting classifier, new simple models are added to correct the prediction errors. Each simple model in the classifier asks the data questions such as "what is the person's relationship status?" or "what is their education level?" similarly to the 20 questions game. The result of each game is then pooled together to create the final model's answer.

Nice job using a metaphor to explain how the model works in everyday terms! The 'story' will help a non-technical person to retain at least a few of the details. Being able to do this well can be worth its weight in gold in industry. There are many times when we have to explain the 'big idea' to an employer or client in a way that makes sense to them without seeming like we're talking down to them.

**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.**

```
clf = GradientBoostingClassifier(random_state=10)
```

Looks good!

```
parameters = {'learning_rate': [0.01,0.05,0.1,0.5],
              'n_estimators': [100,300,500],
              'max_depth': [2,3,5]}
```

I'd also recommend trying even higher values for the `n_estimators` parameter (500-1000).

**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.**

> The resulting accuracy and F-score are only slightly better than the unoptimized model.

Clearly the default parameter settings were pretty good. Nice job squeezing a bit more performance out of your model with tuning!

# 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.

You've done a good job explaining and justifying your choice of "most relevant" features.

✓

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.

Keep in mind that these feature importances may be somewhat 'model specific'. In other words, if you re-ran the analysis with a AdaBoostClassifier, you'd get a different list of 'most important' features. While it's important to know which features are helping the model the most, keep in mind that this may not necessarily indicate something that is fundamental about the dataset itself.

✓

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.

Congratulations...you've passed!

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START