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

4 SPECIFICATIONS REQUIRE CHANGES

Dear student,

well done with your good submission, there are only a few issues to be addressed in order to meet requirements, please refer to my comments for more hints. I've left a few pro tips for your convenience, I hope you might find them interesting.

Keep up your good work!

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: Data assessment:

Though it might be not necessary here, when dealing with the new data-set, it is good practice to assess its specific characteristics and implement the cross validation technique tailored on those very characteristics.
If the dataset is **unbalanced**(some classes are overrepresented compared to others) we could address the unbalanced nature of our data set using Stratified K-Fold and Stratified Shuffle Split Cross validation, as stratification is preserving the preserving the percentage of samples for each class.
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

As for the initial train test split you can obtain stratification by simply using **stratify = income**:

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, income,stratify = income, test_size = 0.2, random_state=42)
```

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 notebook throws an error, please make sure you address that and that the IPython notebook runs without errors before resubmitting:

```
# Print the results
print "Naive Predictor: [Accuracy score: {:.4f}, F-score: {:.4f}]".format(accuracy, fscore)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-7af178dd71d6> in <module>()
      4 from sklearn.tree import DecisionTreeClassifier
      5
----> 6 clf = GradientBoostingClassifier(random_state=1)
      7 clf.fit(X_train,y_train)
      8

NameError: name 'GradientBoostingClassifier' is not defined
```

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.

Well done with your answer, there are only a few issues here:

1. As for gradient boosting I'm not sure why the following statement is relevant as a justification for choosing the algorithm: "Trees are ok for census data because the features can be dependent on each other. For example age, workclass can impact the income level." Could you please provide more details and explain the rationale behind it?
2. As for KNN the following statement is not finished: "As per Udacity Videos it Needs domain knowledge for the grouping algorithm (i.e...)
3. As for support vector machines the rationale for choosing it is quite unclear, you write: "As there are many contours that need to be grouped and classified we can use the kernel trick to group needed contours to make a classification of the picture (i.e. cars / faces / objects)" why are pictures " (i.e. cars / faces / objects)" mentioned here? This is not a visual recognition task.

Hints: Are the pros of the specific algorithm helpful in our case considering the dataset and the problem at hand? Are the weaknesses not regarding our dataset? Are you interested in seeing how these algorithms performed against one another for some reason?

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.

Please set a random state for those algorithms that allow for it as required in: *"Use a random state for each model you use, if provided."*. This allows to obtain reproducible results, which is very important when drawing conclusion on specific figures. You could achieve that by simply specifying the random state as an algorithm's parameter like in:

```
clf_B = RandomForestClassifier(random_state = 42)
```

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.

Pro Tip (Advanced): Xgboost, one of Kaggle's top algorithms.
In the recent years one algorithm emerged as favourite in the machine learning community, it is actually one of the most used in Kaggle: Xgboost.
Here you can find an informative discussion on why that is the case: https://www.quora.com/Why-is-xgboost-given-so-much-less-attention-than-deep-learning-despite-its-ubiquity-in-winning-Kaggle-solutions
The algorithm is not available sci-kit learn, here is how you can start working with it:
http://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/

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 explanation is a bit too vague, by reading the provided description of the algorithm I'm not fully able to understand specifically how it actually works. The goal here should be to thoroughly explain how the algorithm works in a clear and simple way so that someone that is not accustomed to machine learning would be able to understand and describe the mechanism behind the specific algorithm.

Here is an example regarding logistic regression that might be helpful in providing you with a blueprint of what is expected:

1. After the initial training using the training data Logistic Regression gives each feature a weight according to their importance (This is achieved by minimizing a cost function though this detail can be omitted when addressing a layman's audience)
2. When we are predicting we take all feature values multiply them with their weights accordingly
3. We then sum up all the results.
4. This final result is applied to a special function called Sigmoid/Logistic function which only outputs values between 0-1 which can be interpreted as the probability of our positive label.

Gradient-Boosting: https://www.quora.com/What-is-an-intuitive-explanation-of-Gradient-Boosting

http://stats.stackexchange.com/questions/140920/the-difference-between-logistic-regression-and-support-vector-machines

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.

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.

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.

Pro Tip:
An alternative feature selection approach consists in leveraging the power of Recursive Feature Selection to automate the selection process and find a good indication of the number of relevant features (it is not suitable for this problem because that is not what is required by the project rubric, though it is generally a very good approach).
http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

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.

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Student FAQ