Finding Donors for CharityML

Requires Changes

7 SPECIFICATIONS REQUIRE CHANGES

Very well done in your first submission!
However, a few minor changes are required in order to meet our rubric. Keep doing this great job!

Cheers!!

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**

Nice job! It is really great that you used pandas column slicing to get the required numbers.

Please note that these numbers indicates that the classes are imbalanced. I would encourage you to check the following resources to know more about how to deal with imbalanced data -
https://www.quora.com/In-classification-how-do-you-handle-an-unbalanced-training-set
https://blog.dominodatalab.com/imbalanced-datasets/

I would further encourage you to preform some extra exploratory data analysis for the features. You could check out the library Seaborn. For example -

```
import seaborn as sns
sns.factorplot('income', 'capital-gain', hue='sex', data=data, kind='bar'
)
```

Preparing the Data

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

Great job encoding the features using get_dummies and converting the target labels to numerical values!!

It is also possible to use Label Encoder as an alternative, specially when we are dealing with a Multi Class prediction case. Here is an example on how to use LabelEncoder -

```
encoder = LabelEncoder()
income = encoder.fit_transform(income_raw)
```

Evaluating Model Performance

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

Great work on correctly calculating the Accuracy, Precision and Recall!
It is always a great idea to establish a benchmark in any type of problem. As these are now considered our "dumb" classifier results, as any real model should be able to beat these scores, and if they don't we may have some model issues.

Also, well done getting the right calculation of F-score.

Please note that with TN and FN = 0 the Accuracy and Precision don't differ!

I would encourage you to read the below article about metrics - https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b

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

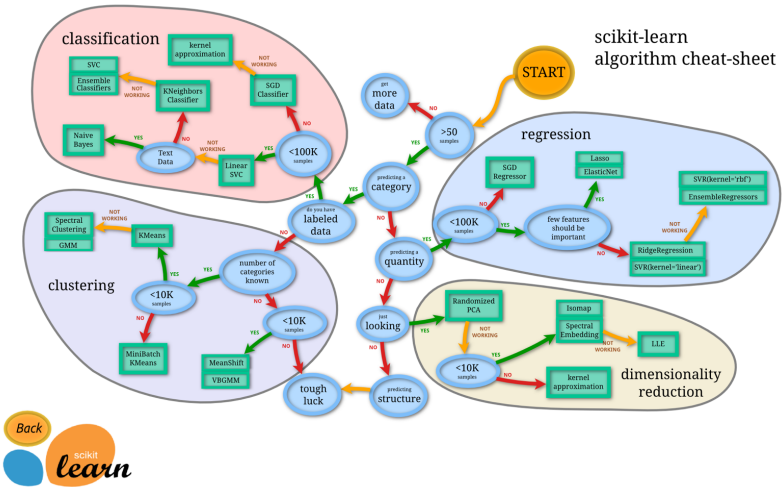
Please list all the references you use while listing out your pros and cons.

Very nice job mentioning some real-world application, strengths/weaknesses! You really have a very good understanding of the models!

However, please clarify on why you think SVM will be suitable for this data set. You mentioned that SVM is suitable for classificaiton task, in fact, every classifier is suitable for classification. I encourage you to develop solid understanding on which model to use for which application. Below, are some ideas to think about and look for in the data/model regarding which to choose -

- the predictive power of the model
- the runtime of the model and how it will scale to much more data
- the interpretability of the model
- how often we will need to run the model and/or if it supports online learning.
- how many categorical features / numerical? What is the distribution of target variable?
- is the data linear?
- is the data non-linear?
- are there outliers present in the data set?
- if there is any missing data?
- the amount of data or the data set size

You might also check out this flowchart from Sklearn as a rough guideline (take it with a grain of salt however) when selecting a model.



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

Great job! Very nice implementation of the `train_predict()` function!

One thing to be aware of in the future, in `fbeta_score()`, you can't switch the `y_true` and `y_pred` parameter arguments. `y_true` has to come first. Please check out this example for a demonstration -

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, fbeta_score
clf = DecisionTreeClassifier(max_depth=5, random_state=1)
clf.fit(X_train, y_train)
# the correct way
print fbeta_score(y_test, clf.predict(X_test), 0.5)
# the incorrect way
print fbeta_score(clf.predict(X_test), y_test, 0.5)
```

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

Good job implementing the supervised models and creating visualizations to compare the performances of the models.

However, you should use `random_state` while initializing Decision Tree Classifier and SVC. Random seeds help in debugging and reproducing the same result at a later time. Please check out the below resource to know more about this - https://machinelearningmastery.com/randomness-in-machine-learning/

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.

Good justification on choosing SVM as the best model among the three choices you made.

However, please rewrite this section once you use `random_seeds` as suggested in previous cell. The scores will probably change.

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.

Great discussion on why SVC is performing good for this data set. However, the question was to explain the inner workings of the algorithm in simple non-technical terms. Your description should be complete and very easy to follow. For, SVC, you may want to cover the following -

- How is the model represented? Is it a separating line or a list of trees?
- How does the algorithm build this model from training data?
- What the model is trying to optimize?
- How does the algorithm use the trained model to make predictions on new data?

Also, please note that when trying to explain any algorithm to a non-technical person, it is always a good idea to include some example and try running step by step of the algorithm. Another great idea is to include pictures which can make the description visually rich and easy to understand.

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.

Nice use of GridSearch, you surely know how to use gridsearch. However, you need to make the following changes to meet the rubric requirement - you need to use `random_seed` while initializing SVC, and you need to do grid search on at least one hyperparameter with at least three values.

Please note that the purpose of gridsearch is to tune the hyperparameters to get a better performance than the unoptimized model. If you check your implementation, you can find out that the F-score on testing set actually degraded a little bit and the accuracy didn't change at all.

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.

Please rewrite this section once you make suggested changes in the previous cell. The results will probably change.

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.

Nice intuition, these are some great features to check out.

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.

Great job implementing AdaBoost classifier to get feature importance!

Please note that each model with `feature_importances_` might return different top predictive features depending on their internal algorithm implementation and you can also use your optimized model here as shown below - `importances = best_clf.feature_importances_`

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.

Nice job analyzing the final model's performance with only the top 5 features.

But, please rewrite this section once you include `random_seed` in SVC before doing grid search. The performance scores will probably change.

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)

RETURN TO PATH

Student FAQ