UDACITY

‹ Return to "Machine Learning Engineer Nanodegree" in the classroom

Finding Donors for CharityML

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

🏆 Excellent job with the report! Your strong grasp of supervised learning and your ability to implement the models clearly shone through in your work.

You've picked up skills that are in high demand and not all that common. You should be proud of your achievement. Keep it up and enjoy the rest of the course! 😎

For additional learning, this is a great guide to approaching almost any ML problem written by a Kaggle grandmaster (http://blog.kaggle.com/2016/07/21/approaching-almost-any-machine-learning-problem-abhishek-thakur/).

Exploring the Data

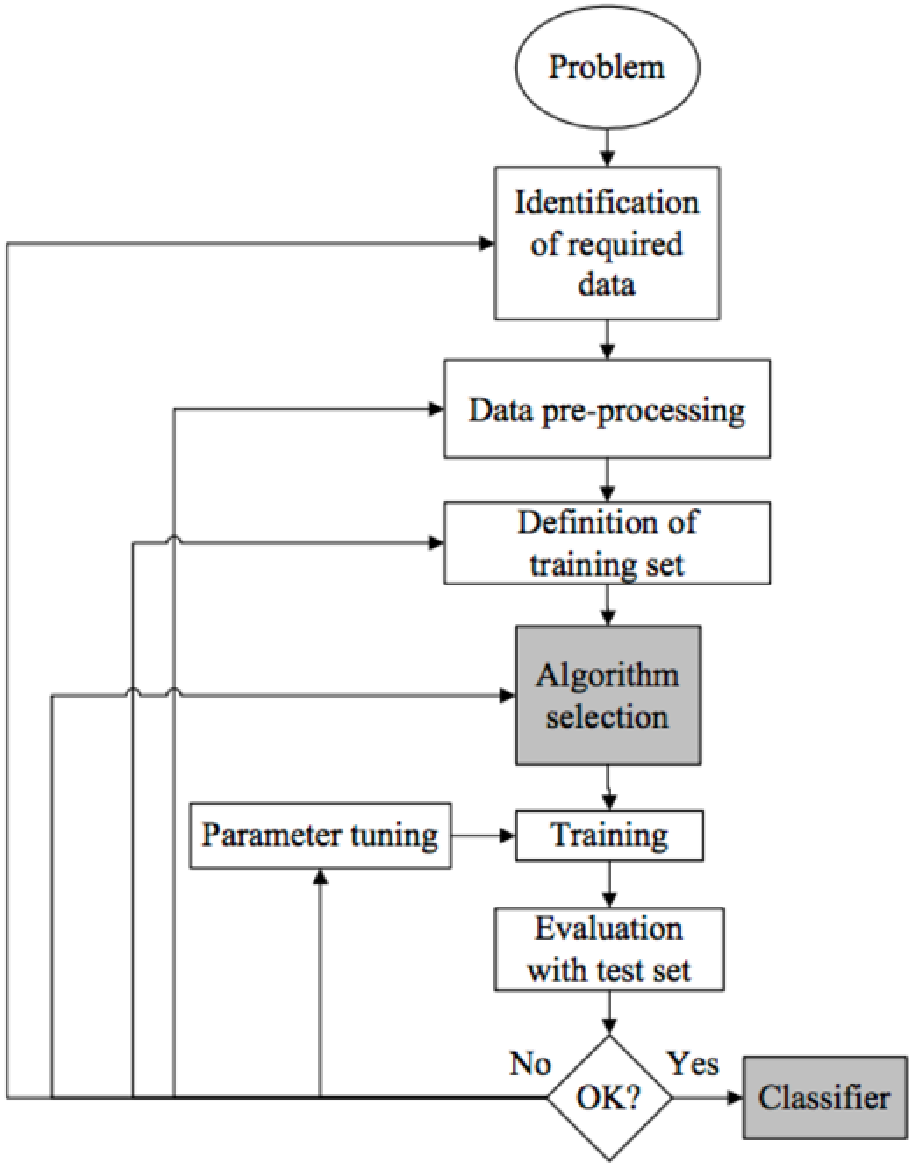
Student's implementation correctly calculates the following:

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

Great work getting the dataset stats! 👏

As you can see we have an imbalanced proportion of individuals making more than $50k vs those making less and will want to make sure the metric we're using for model evaluation is capturing how well the model is actually doing. Here are two really useful articles on handling imbalanced datasets:
https://blog.dominodatalab.com/imbalanced-datasets/
https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html

You may also find the following diagram useful for an overview of the process for developing a supervised learning classifier.



Preparing the Data

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

Good job encoding the features and target labels. 😎

You could also use Sklean's LabelEncoder. (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html) to convert the non-numeric target labels. This would come in handy for performing multi-class (http://scikit-learn.org/stable/modules/multiclass.html) predictions. If you're interested, here's an article that goes deeper into encoding categorical values in Python (http://pbpython.com/categorical-encoding.html).

Evaluating Model Performance

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

Terrific work calculating the accuracy and F-score of a naive predictor — this will serve as a useful benchmark to evaluate how well our model is performing. If you're interested, here's a great article on choosing the right metric for classification problems (https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c).

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.

Nice job describing some of the strengths and weaknesses of the models!

In general, with model selection it's a good idea to try out simpler methods like Logistic Regression or Naive Bayes as a benchmark, and then move on to non-linear classifiers such as your choice of Decision Trees and ensemble methods. It may be exciting and tempting to apply the most sophisticated models right from the start, but in real production projects, the golden rule is you only want to use as much sophistication as needed – in order to maximize efficiency.

If you're interviewing for machine learning roles, you'll find it's common for recruiters to ask questions about comparing different ML models and the most suitable use cases for each. The following additional reading will hopefully help you ace these questions.

Microsoft Azure guide on choosing an algorithm:
https://docs.microsoft.com/en-gb/azure/machine-learning/studio/algorithm-choice

Comprehensive SAS article on model selection:
https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

Great Quora thread comparing common ML models: https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

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

Excellent work implementing the pipeline! 👍 This part is the main driver for the entire project. With this pipeline you can see how the performance of the 3 models changes when using different training sizes passed to the sample_size parameter.

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

Great job generating the results with the 3 sample sizes and setting random states on the classifiers to make your results reproducible.

Random state is a super useful parameter that will save you tons of time. To better understand how it works, you can look at this comprehensive explanation here (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 of your choice by looking at factors such as the models' accuracy, fscore and computational cost/time.

FYI, the highest scores I've seen from students used either Gradient Boosting or AdaBoost. 😉 See more on boosting here (https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/).

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.

Nice discussion of your model in a way that's understandable by a non-technical audience. 👏

For additional reading, this is a useful article describing common machine learning models in plain English (https://hackerbits.com/data/top-10-data-mining-algorithms-in-plain-english/).

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 job tuning your classifier!

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 discussing the tuned model and comparing the scores against the unoptimized model and the naïve predictor. Now you can see the difference that parameter tuning has on your model of choice. 😄 With more experience, you'll develop an intuition for which models have greater potential for improvement from parameter tuning, given different datasets.

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 rankings of the features you think are important — all of them seem to be closely related to an individual's income level. Let's see if we can verify this with feature importances.

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.

Excellent job extracting the feature importances of the model and discussing how the features could be considered important to making predictions. 👍

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.

Good examination of the model's performance with the reduced feature set — the results are worse, but training and testing is computationally more efficient. How you decide on the trade-offs depends on what you're optimizing for, speed or accuracy/fscore.

We could also try adding back just a few more "important" features to see if the accuracy improves without increasing processing time significantly.

In general, feature reduction is a great way to fight the curse of dimensionality (https://medium.freecodecamp.org/the-curse-of-dimensionality-how-we-can-save-big-data-from-itself-d9fa0f872335).

Feature selection (and deciding on the number of features to use in training your classifier) is an important consideration in Machine Learning problems. In general, the more features you train on, the better your accuracy, but the greater your compute cost. Experienced Machine Learning engineers understand this dynamic and are able to make trade-offs suitable for each different use case. Also note that a good feature set contains features that are highly correlated with the class, yet uncorrelated with each other.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review