UDACITY

PROJECT

Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Student Note Response

> I'm actually looking for solid reasons on when a particular model can be used. Most of the data that I could gather are very generic.

The truth is that model selection is almost more of an art than a science. I left some "cheat sheets" in the relevant section below that serve as pretty good heuristics for model selection. In general, one of the most important things to consider is the size of the dataset - some models can be "partially fit", meaning that you can stochastically improve them by feeding them new batches of data. Models like this include neural nets and SGD. This is useful when you have too much data to fit into memory (which is pretty common nowadays). Another thing to consider is the feature type and dimensionality. For example, Gaussian Naive Bayes is a fairly strong model with text data, which is very high-dimensional.

> For our data what is the best model? What are the exact points that will help us in choosing a model. Should we clearly check the correlation between independent variables and the dependent variables, relation between each of the independent variables, size of the data, number of features etc or are there any other details that should be considered.

For this dataset, the best scores I've seen have come from Gradient Boosting and AdaBoost. In practice, you can't often know exactly which model will perform best on your dataset before trying it - this is just the nature of machine learning. Gradient Boosting models have become very popular because of their versatility and general power.

In terms of the data analysis, the more you analyze the data the better. However, there are often diminishing returns, and we don't usually have unlimited time 😉. Finding correlations and associations in the data is a great way to discover the nature of the data, and can often really help the feature engineering phase, which is probably the most important part of coming up with a powerful model (and is unfortunately somewhat neglected in this project). I really recommend checking out Kaggle kernels and discussion boards on Kaggle, as they tackle feature engineering very rigorously and very well. As I mentioned, there are some good heuristics for analyzing the data and picking a model, and I left some cheat sheets for that in a section below.

> For this problem I got a F-Score of around 0.6. What are the next steps that will help in improving it. Say, if I want an F-Score of 0.8 what are the different ways that will help improve it.

You're going to have a hard time doing much better than 0.6 with logistic regression in this setting. It turns out that some of the relationships in this dataset are fairly complex. Even though logistic regression is a solid performer when it hasn't been tuned, you can push some of the higher-variance models to higher F-Scores via hyperparameter tuning. This isn't necessarily something you can know from the outset - sometimes it more-or-less boils down to trial and error. We'll cover some dimensionality reduction techniques in the next section (unsupervised learning), such as PCA, which can let you visualize high dimensional data in lower dimensions. This can help you get a better sense of how the features might be related to each other.

> How exactly LabelEncoder and One-Hot encoding differ? Is it only the ordinal factor in LabelEncoder. If so, how do I remove ordinality when using LabelEncoder?

You're correct that the ordinality is the difference. You just wouldn't want to use the LabelEncoder if you didn't want ordinal encoding - that's the purpose which it is built for.

Project Summary

Well done here! I realized that I forgot to answer your student note in the last review (I'm sorry about that!). In one of the sections below I've covered why logistic regression can be hard to see improvement in via gridsearch. There isn't much else you could have done to push the F-Score of logistic regression significantly higher, but you could try techniques like basis expansion, which might help.

Anyway, you're definitely ready to move on in the program. Keep up the great work, and enjoy the rest of the nanodegree!

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.

Really thorough discussion! I left the cheat sheets below here, as well as an explanation for why you may have had a hard time improving logistic regression via hyperparameter tuning.

TIPS

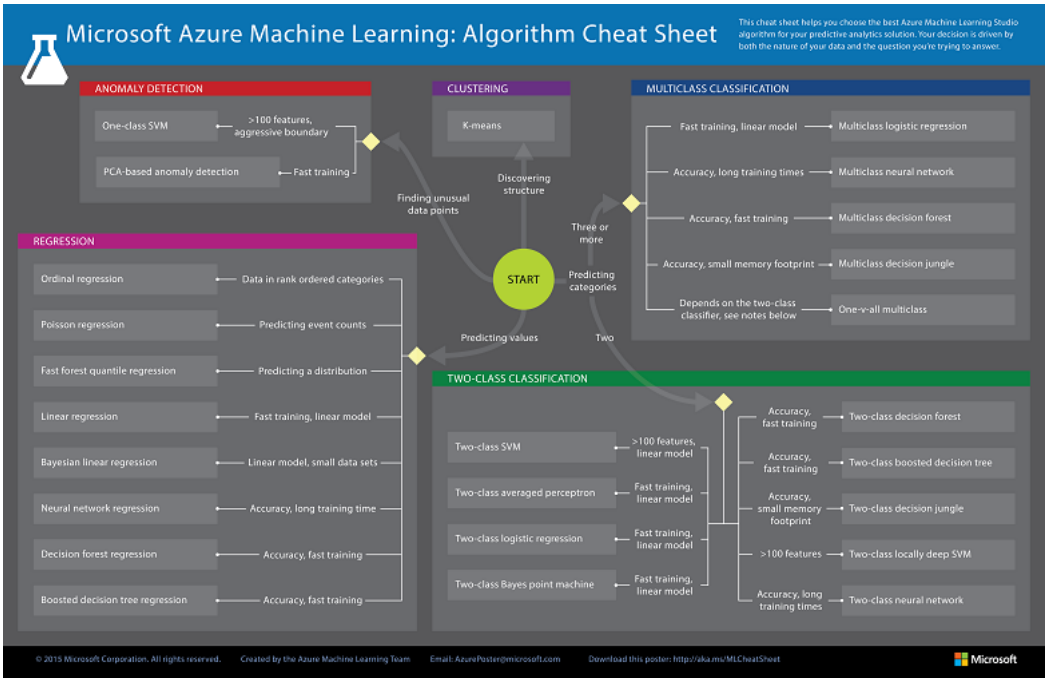
LOGISTIC REGRESSION

You might find that it's difficult to get extra performance out of this model in the hyperparameter tuning phase. This is a consequence of the high-bias nature of the model. One workaround we can use is to perform some clever *feature engineering*. By creating new features that are non-linear combinations of existing features, we can allow logistic regression to learn more complex relationships in the data.
You can use the PolynomialFeatures class in sklearn to do the legwork here.

MODEL SELECTION

Even if we carefully analyze our data, we'll likely still have some level of trial and error involved in our model selection process. Unfortunately, there's no one-size-fits-all solution either. However, there are a few great heuristics we can use to really help narrow down the process! Check out the cheat sheets posted below.
Sklearn

Azure



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.

Nice work using your results, in terms of performance and computational cost, to justify the final choice. As a suggestion, consider a confusion matrix for more detailed information about classifier performance.

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Compute confusion matrix for a model
model = clf_C
cm = confusion_matrix(y_test.values, model.predict(X_test))

# view with a heatmap
sns.heatmap(cm, annot=True, cmap='Blues', xticklabels=['no', 'yes'], yticklabels=['no', 'yes'])
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title('Confusion matrix for:\n{}'.format(model.__class__.__name__));
```

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.

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.

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.

Great analysis here!

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

Student FAQ