UDACITY

PROJECT

Building a Student Intervention System

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Dear student,

I've reviewed your previous submission, thanks for improving the content and the clarity of the pros and cons section and well done successfully completing your project!
If your enthusiastic about process optimization, when it comes to machine learning, I've left a Pro Tip for you in the code section, please be advised that it is quite advanced therefore don't worry if you don't get it immediately, you could keep it and use it later on as you progress in the Nanodegree.

Congratulations on passing your exam!

Classification vs Regression

Student is able to correctly identify which type of prediction problem is required and provided reasonable justification.

Exploring the Data

Student response addresses the most important characteristics of the dataset and uses these characteristics to inform their decision making. Important characteristics must include:

- **Number of data points**
- **Number of features**
- **Number of graduates**
- **Number of non-graduates**
- **Graduation rate**

Pro Tip:
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, in our case there are two main elements:

1. Our dataset is **small**.
2. Our dataset is slightly **unbalanced**. (There are more passing students than on passing students)

We could take advantage of K-fold cross validation to exploit small data sets. Even though in this case it might not be necessary, should we have to deal with heavily unbalance datasets, 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.cross_validation.StratifiedShuffleSplit.html
http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedKFold.html

Preparing the Data

Code has been executed in the iPython notebook, with proper output and no errors.

Training and test sets have been generated by randomly sampling the overall dataset.

Training and Evaluating Models

The pros and cons of application for each model is provided with reasonable justification why each model was chosen to explore.

Excellent job, thanks for improving the section and dramatically improving its readability.

All the required time and F1 scores for each model and training set sizes are provided within the chart given. The performance metrics are reasonable relative to other models measured.

Choosing the Best Model

Justification is provided for which model seems to be the best by comparing the computational cost and accuracy of each model.

Student is able to clearly and concisely describe how the optimal model works in laymen terms to someone what is not familiar with machine learning nor has a technical background.

The final model chosen is correctly tuned using gridsearch 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.

The F1 score is provided from the tuned model and performs approximately as well or better than the default model chosen.

Pro Tip:
We can use a stratified shuffle split data-split which preserves the percentage of samples for each class and combines it with cross validation. This could be extremely useful when the dataset is strongly imbalanced towards one of the two target labels.

http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html

In this example a support vector classifier is used, any other classifier would do, please remind to change the parameters adapting them to the specific classifier you intend to deploy.

```
from sklearn.grid_search import GridSearchCV
from sklearn.svm import SVC
from sklearn.cross_validation import StratifiedShuffleSplit
from sklearn.metrics import f1_score
from sklearn.metrics import make_scorer
```

```
f1_scorer = make_scorer(f1_score, pos_label="yes")
parameters = { 'C' : [ 0.1, 1, 10, 100, 1000 ], 'gamma' : [ 0.0001, 0.0
01, 0.1, 10, 100 ] } # Some SVC parameters
ssscv = StratifiedShuffleSplit( y_train, n_iter=10, test_size=0.1) # 1.
Let's build a stratified shuffle object
grid = GridSearchCV( SVC(), parameters, cv = ssscv , scoring=f1_scorer)
# 2. Let's now we pass the object and the parameters to grid search
grid.fit( X_train, y_train ) # 3. Let's fit it
best = grid.best_estimator_ # 4. Let's reteieve the best estimator foun
d
y_pred = best.predict( X_test ) # 5. Let's make predictions!
print "F1 score: {}".format( f1_score( y_test, y_pred, pos_label = 'ye
s' ))
print "Best params: {}".format( grid.best_params_ )
```

Quality of Code

Code reflects the description in the documentation.

Pro Tip (Advanced): You could actually go well beyond grid search and implement 'pipelines' where the whole machine learning process becomes 'grid-searchable' and you can parameterize and search the whole process though cross validation.
http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
And yes you can try out several algorithms automatically as well too! Watch out though this is pretty advanced stuff, here is a great, informative, top notch tutorial from Zac Sewart!
http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html

DOWNLOAD PROJECT

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Rate this review