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

Hello Student,
Congratulations!!! You made it.
It is truly exceptional that you passed all the questions right in your first submission.
It shows you do understand all the concepts it taught in the lectures thoroughly.
Hope the lectures and this project in particular have effectively taught you the fundamentals of Machine Learning.
Carry on your Great Work! Cheers!

### Classification vs Regression

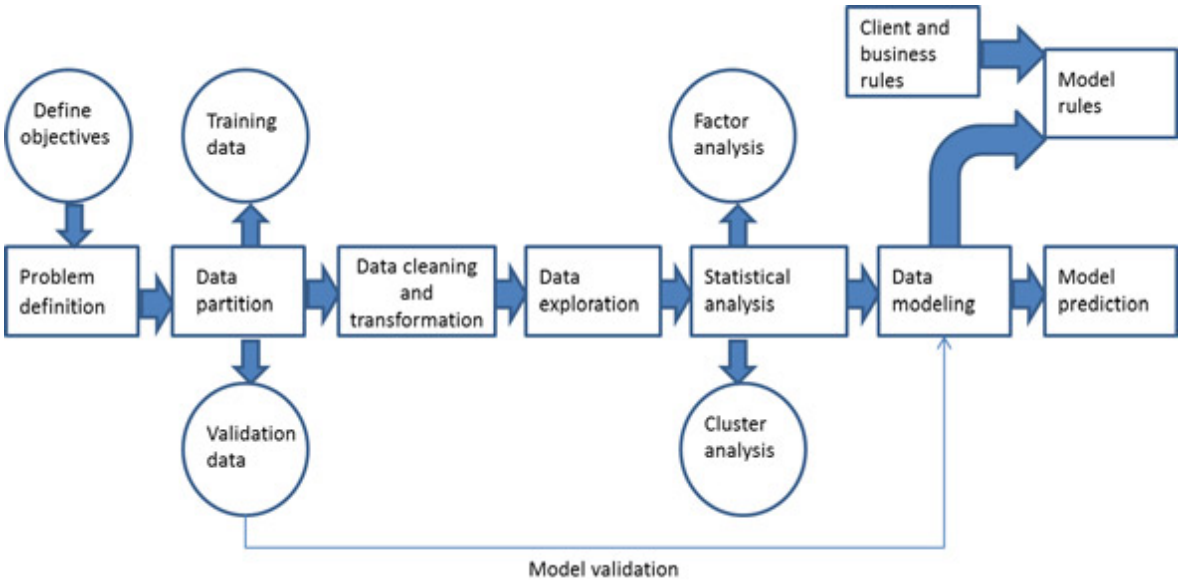
✓

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

Well done identifying the type of machine learning problem! This is important, and you should always think about this when attempting to predict a phenomenon.

### Pro Tip:

Here is a diagram of the Predictive Modelling Process, obtained from this link, just for your reference:




## 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**

Well done here! Your code is short and precise.

### Pro Tips:

**Data Set Imbalance**:

- A very peculiar characteristic of this dataset is that it's labels are quite unbalanced.
- This means that the number of positive training examples is much more than the number of negative training examples, or the number of students who passed is much more than the number of students who failed.
- It's for this reason that an F1 score is chosen as the default evaluation metric rather than an accuracy score.
- Please note that one could also use precision or recall as the evaluation metric for this classification problem.

## Preparing the Data

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

Well done! Your code is short and precise.

### Suggestions and Comments:

When writing code in iPython notebook, please remember to make sure you define a function/variable/object before using it. If not, when someone runs the cells one by one from the top, they might encounter a mistake, because the you defined the function to run the code in the cell below instead of above the said cell.

### Pro Tips:

- Mastering Markdown formatting is extremely useful if you want to produce Quality Reports in iPython Notebook.
- Please check out this link if you want to learn more about Markdown formatting.

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

Well done using the scikit-learn train_test_split function! Your code was precise and concise.

### Pro Tips:

STRATIFIED SHUFFLE SPLIT:

- You may also consider making use of Stratified Shuffle Split since we have an unbalanced dataset. Please look at here and here for more information.

SEEDING YOUR ALGORITHMS:

- In order to remove randomness of your algorithms, and to make sure your results don't differ at each run, please consider to always use a random seed to seed your algorithms.
- A standard practice I've come across is to define a random seed as a global variable in your work, and to use it throughout all the algorithms/methods which require random number generation (splitting data, decision tree initialisation, neural network weight initialisation etc).
- In sklearn, as far as I know, random seeds are provided to methods and functions using the parameter `random_state`. Please seed all of your algorithms in the future if you haven't been doing so yet

## Training and Evaluating Models

**Three supervised models are chosen with reasonable justification. Pros and cons for the use of each model are provided, along with discussion of general applications for each model.**

Fantastic Job Writing about the 4 Models in details! Below are some suggested readings regarding this section:

### Random Forest:

SUGGESTED READING

- If you wanna go deeper with Random Forests, check these out:
    - http://rstudio-pubs-static.s3.amazonaws.com/4239_fcb292ade17648b097a9806fbe026e74.html

### Logistic Regression:

SUGGESTED READING

- Check out these links if you wish to learn more about Logistic Regression:
    - http://cs229.stanford.edu/notes/cs229-notes1.pdf
    - https://en.wikipedia.org/wiki/Logistic_regression
    - http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

SVMs are indeed a fantastic choice for this problem, nice job giving info about them!

### Decision Trees:

SUGGESTED READING

- These links helped me a lot when learning about Decision Trees, hope they help you too:
    - https://www.quora.com/What-are-the-advantages-of-using-a-decision-tree-for-classification
    - https://www.quora.com/What-are-the-disadvantages-of-using-a-decision-tree-for-classification
    - http://scikit-learn.org/stable/modules/tree.html#complexity
    - http://www.cbcb.umd.edu/~salzberg/docs/murthy_thesis/survey/node32.html

### KNN:

SUGGESTED READING

- The following resources might be very useful for learning more about KNNs:
    - http://www.nickgillian.com/wiki/pmwiki.php/GRT/KNN#Disadvantages
    - https://www.youtube.com/watch?v=UqYde-LULfs
    - http://people.revoledu.com/kardi/tutorial/KNN/Strength%20and%20Weakness.htm
- KNNs don't train and store a specific model, so they don't really learn from data. Instead, they classify data by directly dividing the data into the classes in the feature space.

KNNs might be underestimated in the Machine Learning literature, but they can sometimes surprise you, good try.

**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.**

Well done here! Below are some comments and feedback just for your reference:

### Pro Tips:

- Nice job using F1 (harmonic mean of precision and recall) to evaluate your algorithms results.
- One of the advantages of using precision and recall rather than F1 score is ease of interpretability.
- For example, for this particular problem, precision can be interpreted as the number of students who really need intervention out of those picked by the model for intervention.
- While recall is the number of students identified for intervention by the model out of those who really need intervention.

## 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.**

You did awesome in this particular section!

✓

**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.**

Your description on the `Random Forest` in laymen terms is great. And I am sure now you must clearly understand its mechanism and capable to explain it to anyone. Great Job!

### Suggestions and Comments:

- To make a great discussion of the model, please talk about the following regarding the algorithm:
  - Talk about how the algorithm trains itself: what happens during training? How does the algorithm distinguish between the two classes?
  - Also, please briefly discuss about how testing is conducted with this model.
- Explaining the algorithm in layman's terms might be a non-trivial task, as it requires thorough understanding of how the algorithm works.
- I also suggest that in order to improve your description, you could use a visualisation or an illustration to actually show how the algorithm works. Visualisations are known to be highly effective in explaining complex things to people new to a field.

EXTRA LAYMEN'S TERMS EXAMPLES:

- Check out this excellent description of the adaboost algorithm .
  This one might also help:
  https://www.reddit.com/r/MachineLearning/comments/15zrpp/please_explain_support_vector_machines_svm_like_i/

✓

**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.**

Great job fine-tuning your algorithm!

✓

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

You did awesome in this particular section! Since you sound like a pro, below are some pro tips for you:

## Pro Tip:

**Using Stratified Shuffle Split**:

- As mentioned in the section about dataset exploring the data, the dataset is quite unbalanced.
- To better model unbalanced data, it's always preferable to use a stratified shuffle split.
- This is because using Stratified shuffle split, the dataset is split so as to **preserve the percentage of samples for each class.**
- This method avoids not having a single representative of the minor class in a fold.
- Also, given a small data with highly unbalanced classes, stratification provides a safeguard and more consistency among the classes in the two splits.
- This paper gives more information on stratified cross-validation. In the paper, it is stated that the main advantage of this procedure is that it reduces experimental variance, which makes it easier to identify the best methods under consideration (hyper-parameters)
- Below is a short implementation of StratifiedShuffleSplit from sklearn, using adaboost as the classifier. Note that you could also use other algorithms such as SVMs or Decision Trees to conduct grid search using stratified shuffle split.

```
# Import StratifiedShuffleSplit from sklearn
from sklearn.cross_validation import StratifiedShuffleSplit
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import make_scorer

# Define a random state to seed your algorithms
ran_state = 201

# Define your classifier to optimise. You can choose to optimise classifiers
# other than adaboost.
clf = AdaBoostClassifier(random_state=ran_state)

# Create the parameters list you wish to tune
parameters = {'n_estimators' : [5, 10, 15, 20],
              #'algorithm'    : ['SAMME', 'SAMME.R'],
              'learning_rate': [0.01, 0.50, 0.60, 0.70]}

# Create the Stratified Shuffle Split object
sss = StratifiedShuffleSplit(y_train, n_iter=10, test_size=0.24, random_state=
ran_state)
```

```
# Make an f1 scoring function using 'make_scorer'
f1_scorer = make_scorer(f1_score, pos_label="yes")

# Create a grid search object, and use the sss object in place of cv parameter
grid_obj = GridSearchCV(clf, param_grid=parameters, scoring=f1_scorer,
                        cv=sss)

# Fit the grid search object to the training data and find the optimal paramet
ers
grid_obj.fit(X_train, y_train)

# Get the estimator
clf = grid_obj.best_estimator_

# Print the parameters
print clf.get_params(), '\n'

# Report the final F1 score for training and testing after parameter tuning
print "Tuned model has a training F1 score of {:.4f}.".format(predict_labels(c
lf, X_train, y_train))
print "Tuned model has a testing F1 score of {:.4f}.".format(predict_labels(cl
f, X_test, y_test))
```

## Quality of Code

**Code reflects the description in the documentation.**

Well done, your code is very well written Below are some Pro Tips which can help you in becoming a much better programmer:

### Pro Tips:

Writing Clean code is very important and is an integral part of being a Machine Learning Engineer. If you wish to master the art of writing clean code in Python, you may want to check out the Google Python Style Guide.
You don't have to read all the guide at once; you can choose to read only the parts relevant to you at the moment.

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

Student FAQ