Classification vs Regression

Classification vs Regression

Which type of supervised machine learning problem is this, classification or regression? Why?

This is a classification problem becasuse we are trying to predict if student will drop out or not based on avialble data

Exploring the Data

Data exploration

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

```
Total number of students: 395
Number of students who passed: 265
Number of students who failed: 130
Number of features: 30
Graduation rate of the class: 67.09%
```

Preparing the Data

Identify feature and target columns

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

```
In [4]: # Extract feature (X) and target (y) columns
        feature_cols = list(student_data.columns[:-1])  # all columns but last are features
        target_col = student_data.columns[-1]  # last column is the target/label
        print "Feature column(s):-\n{}".format(feature_cols)
        print "Target column: {}".format(target_col)

        X_all = student_data[feature_cols]  # feature values for all students
        y_all = student_data[target_col]  # corresponding targets/labels
        print "\nFeature values:-"
        X_all.head()  # print the first 5 rows

Feature column(s):-
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'trav
eltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'roma
ntic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']
Target column: passed

Feature values:-
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher
1     GP   F   17       U     GT3       T     1     1  at_home     other
2     GP   F   15       U     LE3       T     1     1  at_home     other
3     GP   F   15       U     GT3       T     4     2   health  services
4     GP   F   16       U     GT3       T     3     3    other     other

  ...    higher internet  romantic  famrel  freetime goout Dalc Walc health  \
0  ...       yes       no        no       4         3     4    1    1      3
1  ...       yes      yes        no       5         3     3    1    1      3
2  ...       yes      yes        no       4         3     2    2    3      3
3  ...       yes      yes       yes       3         2     2    1    1      5
4  ...       yes       no        no       4         3     2    1    2      5

  absences
0        6
1        4
2       10
3        2
4        4
```

Preprocess feature columns

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

As you can see, there are several non-numeric columns that need to be converted! Many of them are simply yes/no, e.g. `internet`. These can be reasonably converted into `1`/`0` (binary) values.

Other columns, like `Mjob` and `Fjob`, have more than two values, and are known as *categorical variables*. The recommended way to handle such a column is to create as many columns as possible values (e.g. `Fjob_teacher`, `Fjob_other`, `Fjob_services`, etc.), and assign a `1` to one of them and `0` to all others.

These generated columns are sometimes called *dummy variables*, and we will use the `pandas.get_dummies()` function to perform this transformation.

```
In [5]: # Preprocess feature columns
        def preprocess_features(X):
            outX = pd.DataFrame(index=X.index)  # output dataframe, initially empty

            # Check each column
            for col, col_data in X.iteritems():
                # If data type is non-numeric, try to replace all yes/no values with 1/0
                if col_data.dtype == object:
                    col_data = col_data.replace(['yes', 'no'], [1, 0])
                # Note: This should change the data type for yes/no columns to int

                # If still non-numeric, convert to one or more dummy variables
                if col_data.dtype == object:
                    col_data = pd.get_dummies(col_data, prefix=col)  # e.g. 'school' => 'school_GP', 'school_MS'

                outX = outX.join(col_data)  # collect column(s) in output dataframe

            return outX

        X_all = preprocess_features(X_all)
        print "Processed feature columns ({}):-\n{}".format(len(X_all.columns), list(X_all.columns))

Processed feature columns (48):-
['school_GP', 'school_MS', 'sex_F', 'sex_M', 'age', 'address_R', 'address_U', 'famsize_GT3', 'famsize_LE3', 'Pstatus_
A', 'Pstatus_T', 'Medu', 'Fedu', 'Mjob_at_home', 'Mjob_health', 'Mjob_other', 'Mjob_services', 'Mjob_teacher', 'Fjob_
at_home', 'Fjob_health', 'Fjob_other', 'Fjob_services', 'Fjob_teacher', 'reason_course', 'reason_home', 'reason_other
', 'reason_reputation', 'guardian_father', 'guardian_mother', 'guardian_other', 'traveltime', 'studytime', 'failures'
, 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'go
out', 'Dalc', 'Walc', 'health', 'absences']
```

Split data into training and test sets

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

Split data into training and test sets

So far, we have converted all *categorical* features into numeric values. In this next step, we split the data (both features and corresponding labels) into training and test sets.

```
In [124]: from sklearn.cross_validation import train_test_split
          # First, decide how many training vs test samples you want
          num_all = student_data.shape[0]  # same as len(student_data)
          num_train = 300  # about 75% of the data
          num_test = num_all - num_train

          # TODO: Then, select features (X) and corresponding labels (y) for the training and test sets
          # Note: Shuffle the data or randomly select samples to avoid any bias due to ordering in the dataset
          # I am using train_test_split which  wraps input validation and next(iter(ShuffleSplit(n_samples))) and application to

          X_train , X_test,  y_train , y_test = train_test_split(X_all, y_all, test_size = num_test, random_state=0 )

          print "Training set: {} samples".format(X_train.shape[0])
          print "Test set: {} samples".format(X_test.shape[0])
          # Note: If you need a validation set, extract it from within training data

Training set: 300 samples
Test set: 95 samples
```

Training and Evaluating Models

Time and Space Complexity

Both the big-O notation for the space complexity to represent the model and the time for the algorithm to make a prediction are provided, or a list of several of the major factors that affect the time & space complexities are presented with a description of the largest driving factor as constant, linear, logarithmic, polynomial, etc in nature. Student presents resources or reasonable justification for their response.

Due to small dataset the prediction and training times of the model are negleble. Driving factor of compuptionn is liner in nature, meaning is we increase number of samples, the time it will take to execute will increase. As such driving factor is Liner.https://en.wikipedia.org/wiki/Time_complexity#Polynomial_time

Model Application

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

- What are the general applications of this model?

Decision Tree Classifier applies a straitforward idea to solve the classification problem. it poses a series of carefully crafted questions about the attributes of the test record. Each time time it receive an answer, a follow-up question is asked until a conclusion about the calss label of the record is reached.

- What are its strengths and weaknesses?

Advantages: Decision trees are simple to use, easy to understand.
Disadvantages: Even a small change in input data can at times, cause large changes in the tree.
Decision trees are also prone to errors in classification, owing to differences in perceptions and the limitations of applying statistical tools.

- Given what you know about the data so far, why did you choose this model to apply?

Predictive Power, Relative Simplicity

Model Performance Metrics

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.

Student Intervention System - Decision Tree Classifier

	Training set size		
	100	200	300
Training time (secs)	0	0	0
Prediction time (secs)	0	0	0
F1 score for training set	0.805	0.812	0.813
F1 score for test set	0.802	0.802	0.802

Student Intervention System - LinearSVC Classifier

	Training set size		
	100	200	300
Training time (secs)	0.007	0.018	0.024
Prediction time (secs)	0	0	0
F1 score for training set	0.874	0.825	0.841
F1 score for test set	0.724	0.782	0.796

Student Intervention System - KNeighborsClassifier Classifier

	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.002	0.002	0.006
F1 score for training set	0.833	0.835	0.748
F1 score for test set	0.763	0.782	0.774

Choosing the Best Model

Choosing the Optimal Model

Justification is provided for which model seems to be the best to use given computational cost and model accuracy.

Based on tests performed, I compared 3 models, Decision tree clasifier, Random forest, and Support Vector Classification. After evaluating performance of each model I choose Decision Tree Classifier Model for number of reasons:

- Accuracy: Model predicts with high F1 Score. F1 score conveys the balance between the precision and the recall.

- Peformance: Obtainings training and test prediction fastes

Describing the Model in Layman's Terms

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.
Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. It is called Classification trees because tree models predicts if the student drops out or not.

Model Tuning

The final model chosen is fined tuned using at least one parameter with at least three settings.
I have tuned each model to find best parmaeter. Please refer to outputs of each model

Tuned F1 Score

The F1 score is provided from the tuned model and performs better than the default model chosen.
Best F1 score for test set is 0.802

Quality of Code

Functionality

Code reflects the description in the documentation.