Project 2: Supervised Learning

Building a Student Intervention System

1. Classification vs Regression

Your goal is to identify students who might need early intervention - which type of supervised machine learning problem is this, classification or regression? Why?

2. Exploring the Data

Let's go ahead and read in the student dataset first.

*To execute a code cell, click inside it and press **Shift+Enter**.*

In [1]:
```
# Import libraries
import numpy as np
import pandas as pd
import seaborn as sns
```

```
:0: FutureWarning: IPython widgets are experimental and may change in the futu
re.
```

In [2]:
```
# Read student data
student_data = pd.read_csv("student-data.csv")
print "Student data read successfully!"
# Note: The last column 'passed' is the target/label, all other are feature co
lumns
```

```
Student data read successfully!
```

Now, can you find out the following facts about the dataset?

- Total number of students
- Number of students who passed
- Number of students who failed
- Graduation rate of the class (%)
- Number of features

*Use the code block below to compute these values. Instructions/steps are marked using **TODO**s.*

In [3]:
```
# TODO: Compute desired values - replace each '?' with an appropriate expressi
on/function call
n_students = student_data.shape[0]
n_features = student_data.shape[1]-1
n_passed = (student_data['passed']=="yes").sum()
n_failed = (student_data['passed']=="no").sum()
grad_rate = (n_passed / float(n_passed + n_failed)) * 100
print "Total number of students: {}".format(n_students)
print "Number of students who passed: {}".format(n_passed)
print "Number of students who failed: {}".format(n_failed)
print "Number of features: {}".format(n_features)
print "Graduation rate of the class: {:.2f}%".format(grad_rate)
```

```
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%
```

3. Preparing the Data

In this section, we will prepare the data for modeling, training and testing.

Identify feature and target columns

It is often the case that the data you obtain contains non-numeric features. This can be a problem, as most machine learning algorithms expect numeric data to perform computations with.

Let's first separate our data into feature and target columns, and see if any features are non-numeric.
Note: For this dataset, the last column (`'passed'`) is the target or label we are trying to predict.

In [4]:
```
# Extract feature (X) and target (y) columns
feature_cols = list(student_data.columns[:-1])  # all columns but last are fea
tures
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:-"
print X_all.head()  # print the first 5 rows
```

```
Feature column(s):-
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjo
b', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'scho
olsup', '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

[5 rows x 30 columns]
```

Preprocess feature columns

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() (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html?highlight=get_dummies#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', 'reas
on_course', 'reason_home', 'reason_other', 'reason_reputation', 'guardian_fath
er', 'guardian_mother', 'guardian_other', 'traveltime', 'studytime', 'failure
s', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'interne
t', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'abse
nces']
```

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 order to split the data I have used the StratifiedShuffleSplit. It is a random permutation cross-validation iterator. It generates indices to split data into training and test sets. This type of split is useful when the target variable is unevenly distributed. This would be helpful in our dataset as our dataset is relatively small.

In [6]:
```
# 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 trainin
g and test sets
from sklearn.cross_validation import StratifiedShuffleSplit

splitGen = StratifiedShuffleSplit(y=y_all,
                                        n_iter=3,
                                        train_size=num_train,
                                        test_size=num_test,
                                        random_state=42)
train_index, test_index = next(iter(splitGen))
# Note: Shuffle the data or randomly select samples to avoid any bias due to o
rdering in the dataset
X_train = X_all.iloc[train_index]
y_train = y_all.iloc[train_index]
X_test = X_all.iloc[test_index]
y_test = y_all.iloc[test_index]
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
```

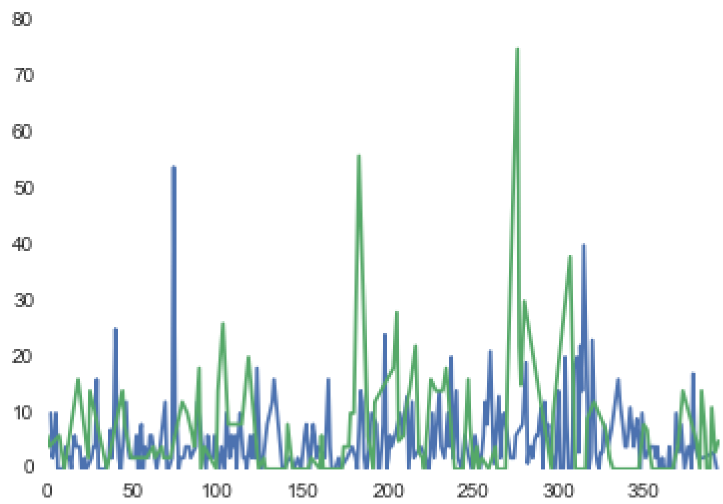
```
Training set: 300 samples
Test set: 95 samples
```

Visualize the basic data

We explore that data a little further for better understanding of the data.

In [7]:
```
%matplotlib inline
student_data[student_data['passed'] == 'yes']['absences'].plot()
student_data[student_data['passed'] == 'no']['absences'].plot()
```

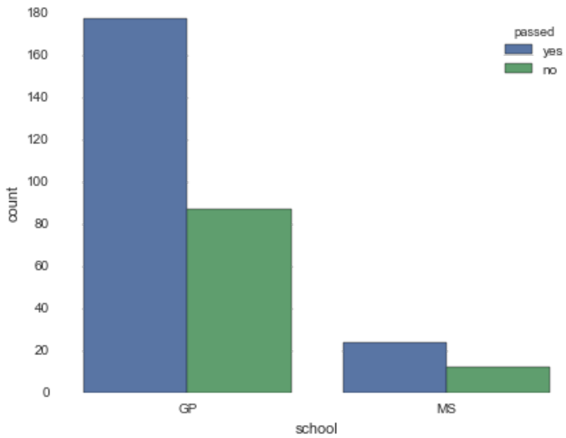
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0xecddb00>

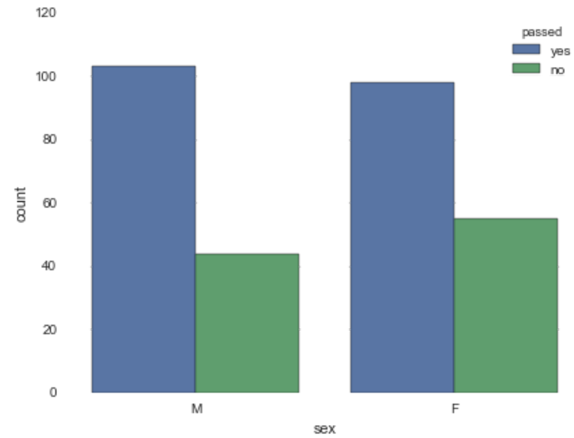


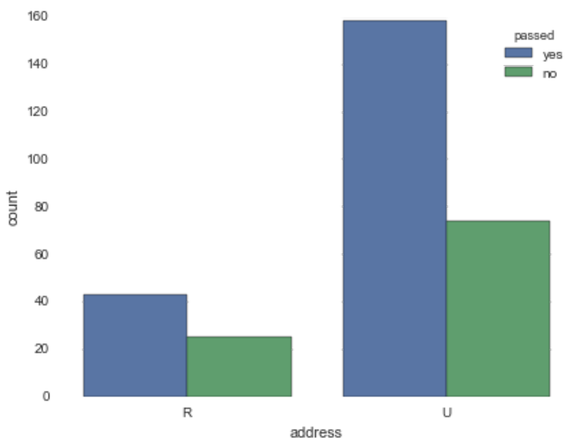
In [8]:
```
feature_full_names = {"school": "student's school",
                      "sex": "student's sex",
                      "age": "student's age",
                      "address": "student's home address type",
                      "famsize": "family size",
                      "Pstatus": "parent's cohabitation status",
                      "Medu": "mother's education",
                      "Fedu": "father's education",
                      "Mjob": "mother's job",
                      "Fjob": "father's job",
                      "reason": "reason to choose this school",
                      "guardian": "student's guardian",
                      "traveltime": "home to school travel time",
                      "studytime": "weekly study time",
                      "failures": "number of past class failures",
                      "schoolsup": "extra educational support",
                      "famsup": "family educational support",
                      "paid": "extra paid classes within the course subject",
                      "activities": "extra-curricular activities",
                      "nursery": "attended nursery school",
                      "higher": "wants to take higher education",
                      "internet": "Internet access at home",
                      "romantic": "with a romantic relationship",
                      "famrel": "quality of family relationships",
                      "freetime": "free time after school",
                      "goout": "going out with friends",
                      "Dalc": "workday alcohol consumption",
                      "Walc": "weekend alcohol consumption",
                      "health": "current health status",
                      "absences": "number of school absences",
                      "passed": "did the student pass the final exam"}
```

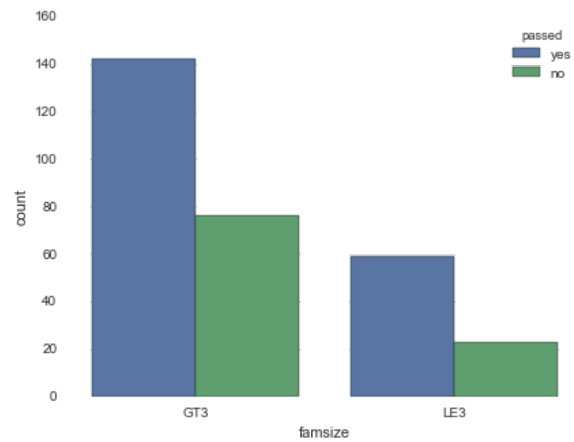
In [9]:
```
X_train_explore = student_data.iloc[train_index]
test_df = X_train_explore.select_dtypes(include=['object'])
count_col= "passed"
```

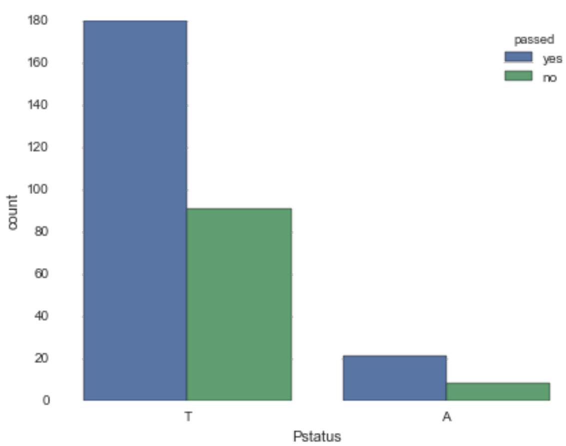
In [10]:
```
import matplotlib.pyplot as plt
for i, col in enumerate(test_df.columns):
    plot_index = i%2
    #f, axes = plt.subplots(figsize=(18, 5))
    #sns.despine(left=True)

    if plot_index ==0:
        f, axes = plt.subplots(1, 2, figsize=(15, 5))
        sns.despine(left=True)
    #print i, col

    sns.countplot(data=test_df, x=col,hue=count_col, ax=axes[plot_index])
```

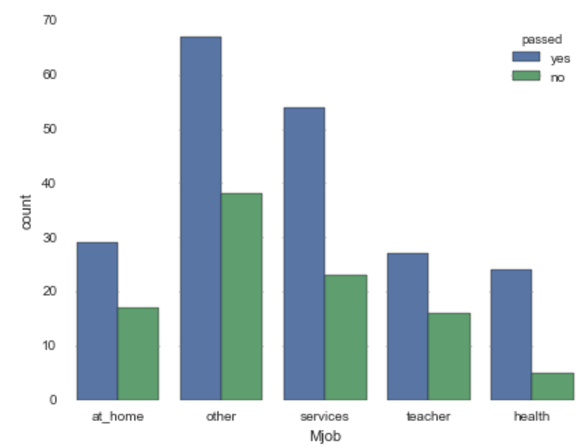

passed
yes
no
count
0
20
40
60
80
100
120
140
160
180
GP
MS
school

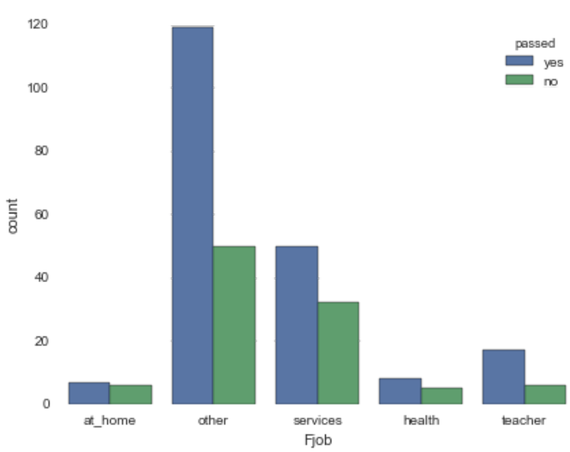

passed
yes
no
count
0
20
40
60
80
100
120
M
F
sex

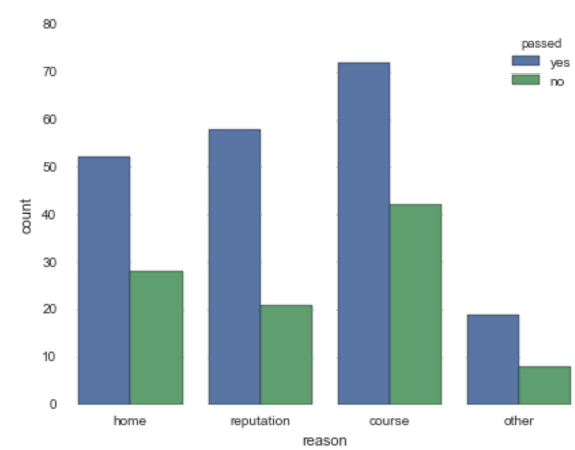

passed
yes
no
count
0
20
40
60
80
100
120
140
160
R
U
address

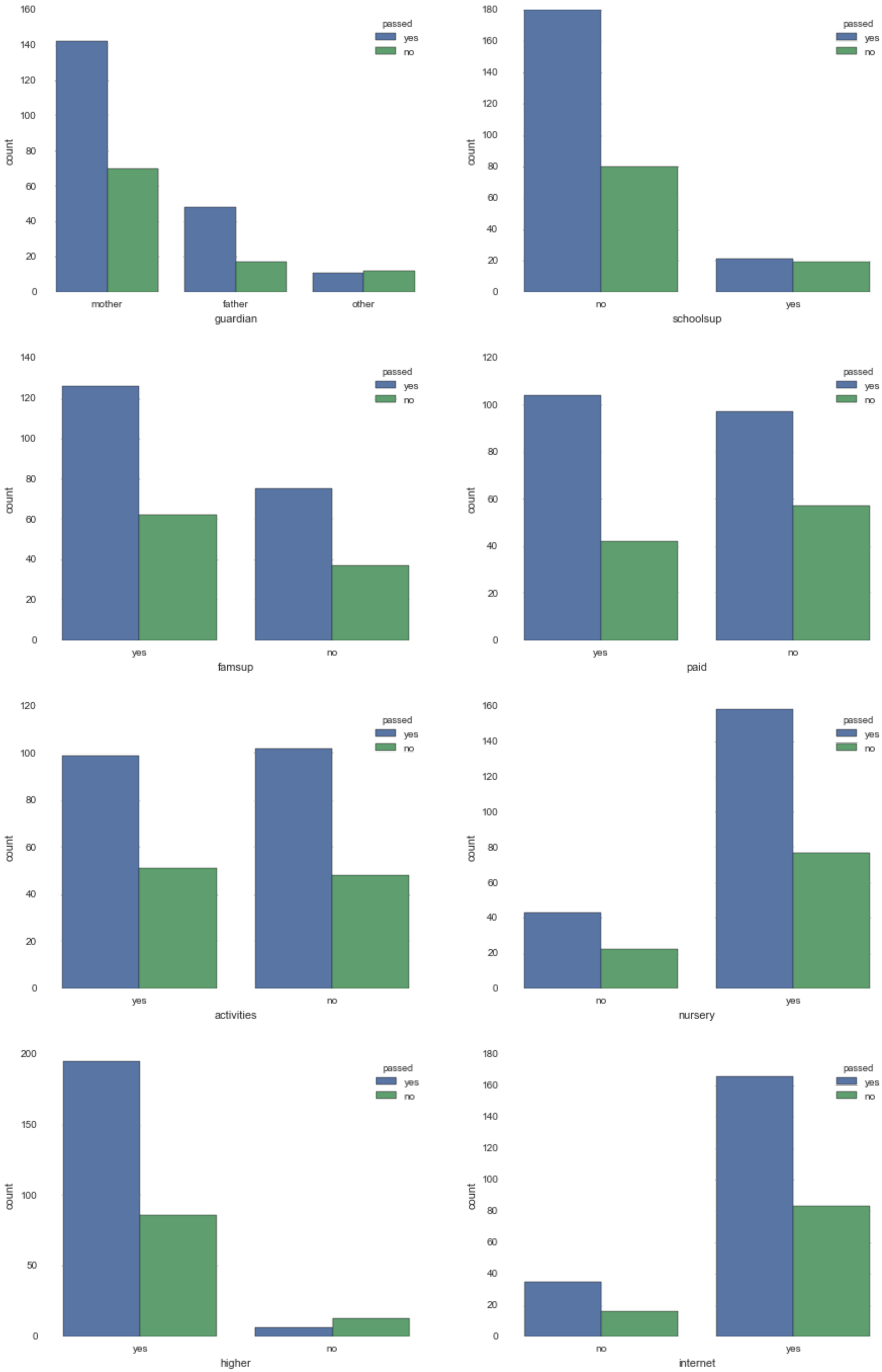

passed
yes
no
count
0
20
40
60
80
100
120
140
160
GT3
LE3
famsize

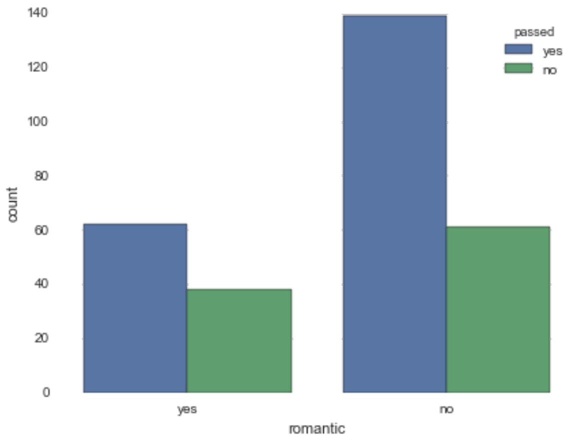

passed
yes
no
count
0
20
40
60
80
100
120
140
160
180
T
A
Pstatus

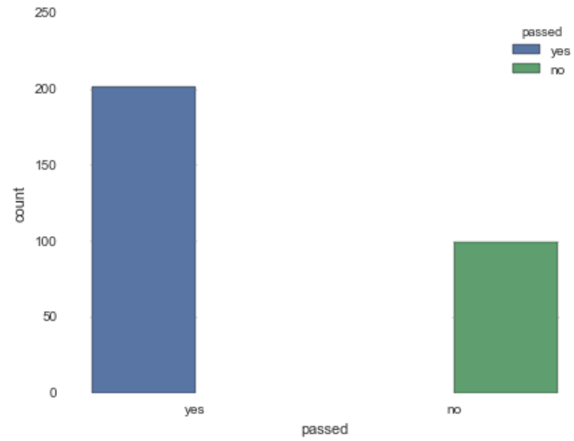

passed
yes
no
count
0
10
20
30
40
50
60
70
at_home
other
services
teacher
health
Mjob


passed
yes
no
count
0
20
40
60
80
100
120
at_home
other
services
health
teacher
Fjob


passed
yes
no
count
0
10
20
30
40
50
60
70
80
home
reputation
course
other
reason


passed
yes
no
count
guardian
mother
father
other
schoolsup
famsup
paid
activities
nursery
higher
internet

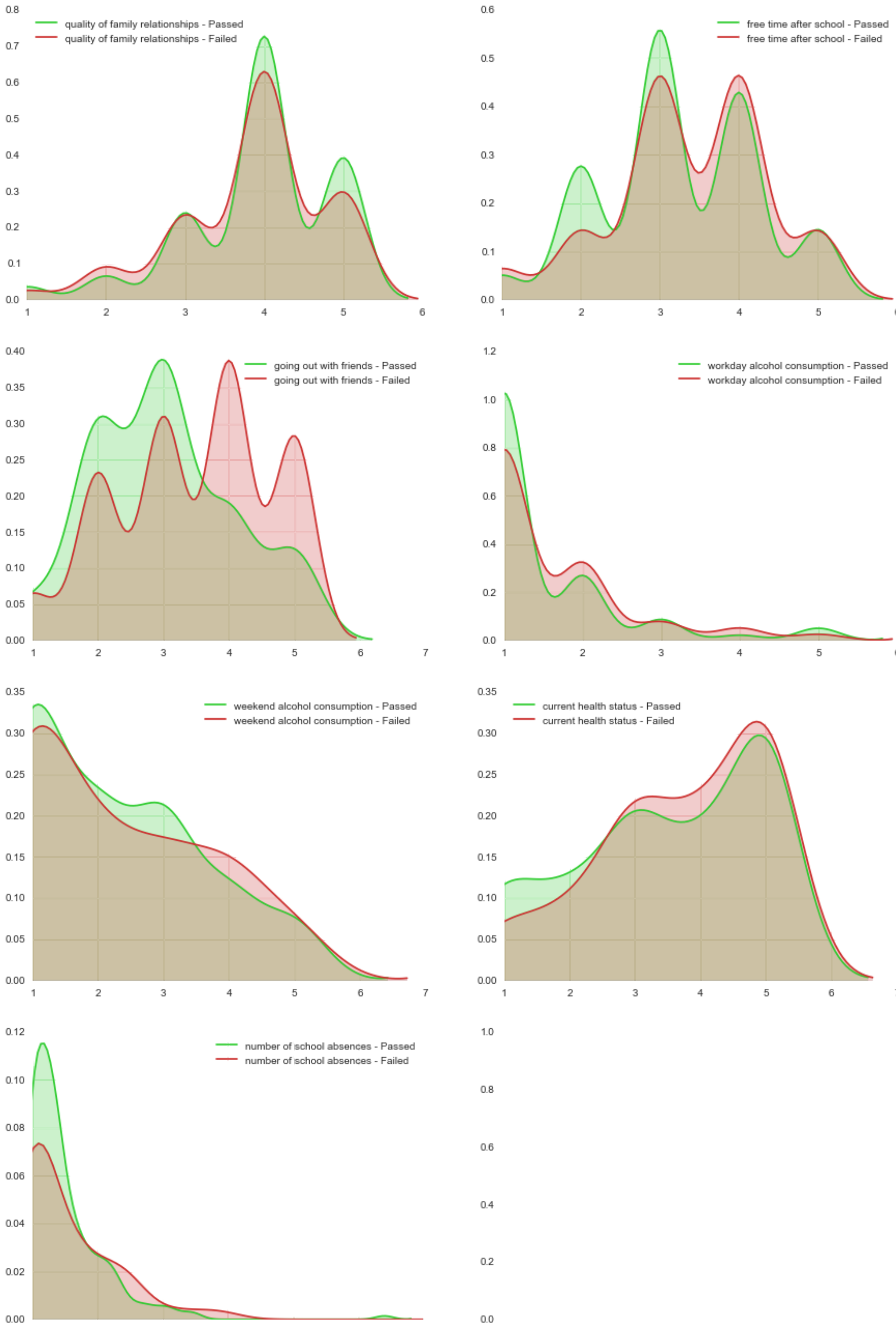




In [11]:
```
test2_df =  X_train_explore.select_dtypes(exclude=['object'])
test2_df = test2_df.join(X_train_explore['passed'])
```

In [12]:
```
factor_col='passed'


plots_per_row =2
for i, col in enumerate(test2_df.columns):
    plot_index = i%2
    if col == factor_col:
        continue
    #f, axes = plt.subplots(figsize=(18, 5))
    #sns.despine(left=True)

    if plot_index ==0:
        f, axes = plt.subplots(1, plots_per_row, figsize=(15, 5))
        sns.despine(left=True)
    #print i, col

    pass_yes = test2_df.loc[test2_df[factor_col] == "yes"]
    yes_label = '{0} - Passed'.format(feature_full_names[col])

    pass_no = test2_df.loc[test2_df[factor_col] == "no"]
    no_label = '{0} - Failed'.format(feature_full_names[col])

        # Plot each kernel density plot and overlay them.
    sns.kdeplot(pass_yes[col],
                    ax=axes[plot_index],
                    shade=True,
                    label=yes_label,
                    color='#32cd33').set(xlim=(min(pass_no[col]))) # Limit the
 x-label to the min.

    sns.kdeplot(pass_no[col],
                    ax=axes[plot_index],
                    shade=True,
                    label=no_label,
                    color='#cd3332').set(xlim=(min(pass_no[col])))
```

```
C:\Users\aw634c\AppData\Local\Continuum\Anaconda\lib\site-packages\matplotlib
\collections.py:590: FutureWarning: elementwise comparison failed; returning s
calar instead, but in the future will perform elementwise comparison
  if self._edgecolors == str('face'):
```


quality of family relationships - Passed
quality of family relationships - Failed
free time after school - Passed
free time after school - Failed
going out with friends - Passed
going out with friends - Failed
workday alcohol consumption - Passed
workday alcohol consumption - Failed
weekend alcohol consumption - Passed
weekend alcohol consumption - Failed
current health status - Passed
current health status - Failed
number of school absences - Passed
number of school absences - Failed

4. Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem. For each model:

- What are the general applications of this model? What are its strengths and weaknesses?
- Given what you know about the data so far, why did you choose this model to apply?
- Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F_1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.

Model Selection

Logistic Regression

Logistic regression is classification machine learning algorithm. Logistic regression measures the relationship between the categorical dependent variable (y) and one or more independent variables (X) by estimating the probabilities using logistic function (ex- sigmoid curve) which is the cumulative logistic distribution. The dependent variable (y) is a discrete variable (0 or 1), called the class. The estimated probabilities is used to predict a given example or given independent variables whether the example belongs to class "1" or class "0". The 2 class of "0" or "1" belongs to the binary classification problems. The logistic regression model can be used for multi class claification also.

Logistic Function or Sigmoid Function

The logistic function or logistic curve also called sigmoid curve is used to estimating the probabilities for logistic regression model. The equation for a sigmoid function is given below.

$f(x) = \frac{1}{1+exp^(-x)}$

Estimating conditional Probability with Logistic Function

$$P(y_i|\mathbf{x}_i, \theta) = \frac{1}{1 + \exp(-\mathbf{\Theta}^T(\mathbf{x}_i))}$$

Strengths

Logistic regression is one of widely used classification model.

- Logistic Regression is very straightforward and easy to implement.
- Logistice Regression being a linear classifier works well with high dimensional data.
- Logistic Regression along with regulization is a convex function. This convexity ensures there are no local minima.
- Convexity of the function ensures convergence, that is solution is fast.

Weakness

- Logistic Regression works well for discrete outcomes but not for continuous outcomes.
- Each data points in Logistic Regression needs to be independent of other data points.
- Logistic Regression models are vulnerable to overfitting.
- Logistic Regression requires a lot of data.
- Data needs to be normalized for convergence.

Applications Of Logistic Regression

Logistic Regression models are used

- Credit Scoring Models.
- Sentiment Classifier.
- Marketing Campaigns.
- Image Classifications.

Logistic Regression Applied to Student Intervention

Logistic regression with regularization is a good supervised learning model that can be used to predict the pass/fail of the students. With regularization logistic regression will keep the most important features by giving them higher weights and suppresing the less significant features. Logistic Regression with regularization helps in avaoiding over-fitting of the data, when we have very small data set, which is the case in our current student intervention data set.

The biggest weakness is that logistic regression assumes that the data set is linearly separeble by the weighted sum of the features that have been measured. This logistic regression model with regularization generates a low bias model and less variance as compared to a Decision Tree.

Decision Trees

Decision Tree learning model is a learning algorithm that can be used for classification or regression. Decision tree models are represent an inverted tree, where each branch represents outcome of the logical results(yes/no) and each leaves represent the values of the labels. Topmost node of the inverted tree is called the root node. The different metrics used in decision trees are

- Gini Impurity
- Information Gain
- Classification Error

Decision Tree models predicts the value of the dependent variable discrete or continous.

Strengths

- Decision Trees are simple to understand and interpret.
- Decision Trees does not need a lot of data as compared to Logistic Regression.
- Data does not need to be normalized.
- Decision Trees can be used to predict both discrete (class lables) or coontinous values.

Weakness

- Decision Trees tend to overfit the data.
- Decision Trees works well on training data but poorly on test data due to overfitting.
- Pruning and Boosting techniques used to prevent overfiting by Decision Tree Learning Models.

Applications Of Decision Trees

Decision Tree models are used

- Credit Scoring Models and Financial Analysis.
- Medical Diagnosis.
- Control Systems.
- Object Recognition (Kinect).
- Text Classification.
- Sentiment Analysis.

Decision Trees Applied to Student Intervention

The biggest strength of Decision Trees is that they are very easy to interpret and hence is another choice as supervised learning model. Decision trees give a inverted tree which can be walked down based on decision splits at certain features in order to identify whether a student is likely pass or fail. Decision Trees does not change with the scale of features. Outliers do not affect the Decision Tree models. The one weakness of Decision Trees are prone to high variance as the trees can grow deep. This high variance cause the Decision Trees to overfit the data. This causes the Decision tree to do very weel on the training data, but predicts very poorly on the test data. Cross-validation is required while tuning Decision Tree and tuning of the hyper paramaters such as Max depth.

Support vector Machines (SVM)

Support Vector Machines are learning models that can be used for classification or regression. SVMs are non-probablistic learning models that categorizes a data point into oone or the other category. SVMs are capable of doing linear classification as well as non-linear classification. Non-linear classification is done by using kernels. Gaussian kernel is one of the most comonly used kernels. SVMs are also called large margin classifiers. SVMs gives a direct prediction of the lables (0/1 in binary classifiers) as compared to the logistic regression which is probablistic model.

Strengths

- SVMs works very well on data that are not lineraly separable.
- SVMs are not affected by local minima as compared to Logistic Regression.
- SVMs works very well with high dimesnional data and does not suffer the curse of dimensionality.
- SVMs can be applied for a classification or regression problem.
- SVMs Convex Optimization function gurantees convergence to global slution.

Weakness

- SVMs are very sensitive to noise.
- Mislabelled examples will decrease the performance.
- Choice of Kernel(Gaussian, Polynomial etc).
- Kernel parameters required to fine tune the SVMs model and this is a time consuming process.

Applications Of Decision Trees

Decision Tree models are used

- Image Classification
- Medical Diagnosis(cancer classification).
- Bioinformatics(Protein Classification).
- Character Recognition(Hand writting).
- Text Classification.
- Sentiment Analysis.

Support Vector Machines Applied to Student Intervention

Support Vector Machines would be able to predict better and have higher accuracies in predicting whether the student would pass or fail. The biggest issue in this model is that we would only be predicting whether a student passes or fails. This model would not give us any insight into the effect of the features or what features we might need to address so as to improve the passing rate of the students. Also support vector machines do take longer to train. In our student intervention data set the size of the data is small and hence SVM can be used along with cross-validation to improve accuracy.

Produce a table showing training time, prediction time, F_1 score on training set and F_1 score on test set, for each training set size.

Note: You need to produce 3 such tables - one for each model.

In [13]:
```
# Train a model
import time

def train_classifier(clf, X_train, y_train):
    print "Training {}...".format(clf.__class__.__name__)
    start = time.time()
    clf.fit(X_train, y_train)
    end = time.time()
    print "Done!\nTraining time (secs): {:.3f}".format(end - start)
    return (end-start)

# TODO: Choose a model, import it and instantiate an object
from sklearn import svm
clf = svm.SVC(kernel='rbf',random_state=42)

# Fit model to training data
trainTime =train_classifier(clf, X_train, y_train)  # note: using entire train
ing set here
print clf  # you can inspect the learned model by printing it
print "Training time (secs): {:.3f}".format(trainTime)
```

```
Training SVC...
Done!
Training time (secs): 0.006
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.0,
  kernel='rbf', max_iter=-1, probability=False, random_state=42,
  shrinking=True, tol=0.001, verbose=False)
Training time (secs): 0.006
```

In [14]:
```
# Predict on training set and compute F1 score
from sklearn.metrics import f1_score
import time

def predict_labels(clf, features, target):
    print "Predicting labels using {}...".format(clf.__class__.__name__)
    start = time.time()
    y_pred = clf.predict(features)
    end = time.time()
    print "Done!\nPrediction time (secs): {:.6f}".format(end - start)
    return f1_score(target.values, y_pred, pos_label='yes'), (end-start)

train_f1_score, predictTime = predict_labels(clf, X_train, y_train)
#predict_labels(clf, X_train, y_train)
print "F1 score for training set: {}".format(train_f1_score)
print "Prediction time (secs): {:.6f}".format(predictTime)
```

```
Predicting labels using SVC...
Done!
Prediction time (secs): 0.004000
F1 score for training set: 0.866379310345
Prediction time (secs): 0.004000
```

In [15]:
```
# Predict on test data

print "F1 score for test set: {}".format(predict_labels(clf, X_test, y_test)
[0])
```

```
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.805194805195
```

In [16]:

```
# Train and predict using different training set sizes
def train_predict(clf, X_train, y_train, X_test, y_test):
    print "------------------------------------------"
    print "Training set size: {}".format(len(X_train))
    #trainTime = train_classifier(clf, X_train, y_train)
    train_classifier(clf, X_train, y_train)
    print "F1 score for training set: {}".format(predict_labels(clf, X_train,
y_train)[0])
    print "F1 score for test set: {}".format(predict_labels(clf, X_test, y_tes
t)[0])
    train_f1_score, predictTimeTrain = predict_labels(clf, X_train, y_train)
    test_f1_score, predictTimeTest = predict_labels(clf, X_test, y_test)

    F1_scores = {'F1_train': train_f1_score,
                 'F1_test': test_f1_score}

    timeTaken = {'Training Time': trainTime,'Predict Train Time': predictTimeT
rain,'Predict Test Time': predictTimeTest}

    return F1_scores, timeTaken

setSize =[100,200,300]
#Reference Stack Flow

rowList =[]
# TODO: Run the helper function above for desired subsets of training data
for size in setSize:
    splitGen = StratifiedShuffleSplit(y=y_all,
                                      n_iter=3,
                                      train_size=size,
                                      test_size=num_test,
                                      random_state=42)
    train_index, test_index = next(iter(splitGen))
    X_train = X_all.iloc[train_index]
    y_train = y_all.iloc[train_index]
    X_test = X_all.iloc[test_index]
    y_test = y_all.iloc[test_index]
    #X_train, X_test1, y_train, y_test1= train_test_split(X_all,y_all,train_si
ze= size, random_state=42)
    F1_scores, predicttime = train_predict(clf, X_train, y_train, X_test, y_te
st)
    print size
    one_row ={"Training Size":size}
    one_row.update(F1_scores)
    one_row.update(predicttime)

    rowList.append(one_row)
    #dfTest.from_dict
    #print F1_scores, predicttime
# Note: Keep the test set constant
print "________________________________________________________________________
_________________________"
print clf
svmModel= pd.DataFrame(rowList)
svmModel
```

```
------------------------------------------
Training set size: 100
Training SVC...
Done!
Training time (secs): 0.001
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.864516129032
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.81045751634
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.000000
100
------------------------------------------
Training set size: 200
Training SVC...
Done!
Training time (secs): 0.003
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.861736334405
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.823529411765
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
200
------------------------------------------
Training set size: 300
Training SVC...
Done!
Training time (secs): 0.005
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005000
F1 score for training set: 0.866379310345
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.805194805195
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005000
Predicting labels using SVC...
```

```
Done!
Prediction time (secs): 0.002000
300
______________________________________________________________________________
__________________
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.0,
  kernel='rbf', max_iter=-1, probability=False, random_state=42,
  shrinking=True, tol=0.001, verbose=False)
```

Out[16]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.810458	0.864516	0.000	0.001	100	0.006
1	0.823529	0.861736	0.001	0.002	200	0.006
2	0.805195	0.866379	0.002	0.005	300	0.006

In [17]:
```
# TODO: Train and predict using two other models
#Decision Tree Classifier. Use Decision Tree to classify the data
from sklearn import tree
clfTree = tree.DecisionTreeClassifier(random_state=42)
modelRow={}
rowList =[]
for size in setSize:
    splitGen = StratifiedShuffleSplit(y=y_all,
                                    n_iter=3,
                                    train_size=size,
                                    test_size=num_test,
                                    random_state=42)
    train_index, test_index = next(iter(splitGen))
    X_train = X_all.iloc[train_index]
    y_train = y_all.iloc[train_index]
    X_test = X_all.iloc[test_index]
    y_test = y_all.iloc[test_index]
    #X_train, X_test1, y_train, y_test1= train_test_split(X_all,y_all,train_si
ze= size, random_state=42)
    F1_scores, predicttime = train_predict(clfTree, X_train, y_train, X_test,
y_test)
    print size
    modelRow ={"Training Size":size}
    modelRow.update(F1_scores)
    modelRow.update(predicttime)

    rowList.append(modelRow)
    #dfTest.from_dict
    #print F1_scores, predicttime
# Note: Keep the test set constant
print "_______________________________________________________________________
_________________________"
print clfTree
TreeModel=pd.DataFrame(rowList)
TreeModel
```

```
------------------------------------------
Training set size: 100
Training DecisionTreeClassifier...
Done!
Training time (secs): 0.001
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 1.0
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.6875
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
100
------------------------------------------
Training set size: 200
Training DecisionTreeClassifier...
Done!
Training time (secs): 0.001
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 1.0
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.772727272727
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.001000
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
200
------------------------------------------
Training set size: 300
Training DecisionTreeClassifier...
Done!
Training time (secs): 0.002
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 1.0
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.683760683761
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000000
Predicting labels using DecisionTreeClassifier...
```

```
Done!
Prediction time (secs): 0.000000
300
______________________________________________________________________________
__________________
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            random_state=42, splitter='best')
```

Out[17]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.687500	1	0	0.000	100	0.006
1	0.772727	1	0	0.001	200	0.006
2	0.683761	1	0	0.000	300	0.006

In [18]:
```
#Create a Clssifier Using Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_test_scaled = pd.DataFrame(X_test_scaled, columns=X_test.columns)
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train.columns)

clfLogReg = LogisticRegression(penalty='l1', random_state=42)
modelRow={}
rowList =[]
for size in setSize:
    splitGen = StratifiedShuffleSplit(y=y_all,n_iter=3,train_size=size,test_size=num_test,random_state=42)
    train_index, test_index = next(iter(splitGen))
    X_train_scaled = X_all.iloc[train_index]
    y_train_scaled = y_all.iloc[train_index]
    X_test_scaled = X_all.iloc[test_index]
    y_test_scaled = y_all.iloc[test_index]
    #X_train, X_test1, y_train, y_test1= train_test_split(X_all,y_all,train_size= size, random_state=42)
    F1_scores, predicttime = train_predict(clfLogReg, X_train_scaled, y_train_scaled, X_test_scaled, y_test_scaled)
    print size
    modelRow ={"Training Size":size}
    modelRow.update(F1_scores)
    modelRow.update(predicttime)

    rowList.append(modelRow)
    #dfTest.from_dict
    #print F1_scores, predicttime
# Note: Keep the test set constant
print "____________________________________________________________________________________________"
print clfLogReg
logRegModel =pd.DataFrame(rowList)
logRegModel
```

```
------------------------------------------
Training set size: 100
Training LogisticRegression...
Done!
Training time (secs): 0.300
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.192000
F1 score for training set: 0.895104895105
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.755555555556
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
100
------------------------------------------
Training set size: 200
Training LogisticRegression...
Done!
Training time (secs): 0.005
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.842465753425
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.814285714286
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
200
------------------------------------------
Training set size: 300
Training LogisticRegression...
Done!
Training time (secs): 0.006
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.835616438356
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.753623188406
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
```

```
Done!
Prediction time (secs): 0.000000
300
______________________________________________________________________________
__________________
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l1', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0)
```

Out[18]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.755556	0.895105	0	0.001	100	0.006
1	0.814286	0.842466	0	0.000	200	0.006
2	0.753623	0.835616	0	0.000	300	0.006

In [19]:

```
#Support Vector Machine Model
svmModel
```

Out[19]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.810458	0.864516	0.000	0.001	100	0.006
1	0.823529	0.861736	0.001	0.002	200	0.006
2	0.805195	0.866379	0.002	0.005	300	0.006

In [20]:

```
#Decision Tree Model
TreeModel
```

Out[20]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.687500	1	0	0.000	100	0.006
1	0.772727	1	0	0.001	200	0.006
2	0.683761	1	0	0.000	300	0.006

In [21]:

```
#Logistic Regression Model
logRegModel
```

Out[21]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.755556	0.895105	0	0.001	100	0.006
1	0.814286	0.842466	0	0.000	200	0.006
2	0.753623	0.835616	0	0.000	300	0.006

Computation Cost of Each Model

The three models I choose were also studied for the computation time and how the F1_score of the train set and test set varies with respect to the data set size.

In [22]:
```
subset_sizes = xrange(100, 301, 10)
```

In [23]:
```
def subset_train_predict(clf, X_train, y_train, X_test, y_test, subset_sizes):
    row_list =[]
    X_train
    for i in subset_sizes:
        row_new = {'Training Size':i}
        X_train_subset = X_train[:i]
        y_train_subset = y_train[:i]
        F1_scores, predicttime = train_predict(clf, X_train_subset, y_train_su
bset, X_test, y_test)
        #print "I am in subset Predict",F1_scores
        row_new.update(F1_scores)
       # print "I am in subset Predict",row_new
        row_new.update(predicttime)
        row_list.append(row_new)

    print row_list
    return pd.DataFrame(row_list)
```

In [53]:
```
def time_plot(modelStats):
        fig, ax = plt.subplots(figsize=(12, 9))
        ax.plot(modelStats['Training Size'],modelStats['Predict Test Time'],la
bel='Test Prediction Time')
        ax.plot(modelStats['Training Size'],modelStats['Predict Train Time'],
'--',label='Train Prediction Time')
        ax.plot(modelStats['Training Size'],modelStats['Training
Time'],label='Training Time')
        legend = ax.legend(loc='best')
        ax.set_ybound(min(modelStats['Predict Train Time']) - 0.001, max(model
Stats['Predict Train Time']) + 0.005)
        ax.set_xticks(subset_sizes)
        ax.set_xticklabels(subset_sizes, rotation='vertical')
        ax.set_ylabel('Seconds')
        ax.set_xlabel('Training Set Size')
        ax.set_title('Training/Prediction Times')
        plt.show()
```

In [48]:
```
def F1_plot(modelStats):
    fig, ax = plt.subplots(figsize=(12, 9))
    ax.plot(modelStats['Training Size'], modelStats['F1_test'], label='Test F1
 score')
    ax.plot(modelStats['Training Size'], modelStats['F1_train'], '--',
label='Train F1 score')
    legend = ax.legend(loc='best')
    ax.set_ybound(min(modelStats['F1_test']) - 0.05,
max(modelStats['F1_train']) + 0.05)
    ax.set_xticks(subset_sizes)
    ax.set_xticklabels(subset_sizes, rotation='vertical')
    ax.set_ylabel('F1 Score')
    ax.set_xlabel('Training Set Size')
    ax.set_title('F1 Scores for each sample size training set')
    plt.show()
```

In [ ]:
```
### Logistic Regression

The data set was standarized using the standard scaler which standizes the fea
tures by centering around the mean and scaling to
unit variance.
Below we plot calculate the F1_scores for train and test data, the training ti
me and the prediction time on the data sets.
```

In [24]:
```
logReg_stats = subset_train_predict(LogisticRegression(penalty='l1', random_st
ate=42),
                                    X_train_scaled, y_train_scaled,
                                    X_test, y_test,
                                    subset_sizes=subset_sizes)
```

```
------------------------------------------
Training set size: 100
Training LogisticRegression...
Done!
Training time (secs): 0.001
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.868965517241
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.701492537313
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 110
Training LogisticRegression...
Done!
Training time (secs): 0.001
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.876543209877
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.724637681159
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 120
Training LogisticRegression...
Done!
Training time (secs): 0.002
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.868131868132
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.760563380282
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
```

```
----------------------------------------
Training set size: 130
Training LogisticRegression...
Done!
Training time (secs): 0.002
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.873684210526
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.757142857143
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
----------------------------------------
Training set size: 140
Training LogisticRegression...
Done!
Training time (secs): 0.001
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.847290640394
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.742857142857
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
----------------------------------------
Training set size: 150
Training LogisticRegression...
Done!
Training time (secs): 0.003
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.003000
F1 score for training set: 0.844036697248
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.757142857143
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
```

```
------------------------------------------
Training set size: 160
Training LogisticRegression...
Done!
Training time (secs): 0.003
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.830508474576
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.760563380282
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 170
Training LogisticRegression...
Done!
Training time (secs): 0.003
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.824489795918
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.731343283582
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 180
Training LogisticRegression...
Done!
Training time (secs): 0.002
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.78431372549
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.724637681159
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
```

```
------------------------------------------
Training set size: 190
Training LogisticRegression...
Done!
Training time (secs): 0.003
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.805653710247
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.755244755245
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 200
Training LogisticRegression...
Done!
Training time (secs): 0.003
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.806779661017
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.753623188406
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 210
Training LogisticRegression...
Done!
Training time (secs): 0.004
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.815533980583
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.765957446809
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
```

```
------------------------------------------
Training set size: 220
Training LogisticRegression...
Done!
Training time (secs): 0.004
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.813664596273
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.765957446809
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 230
Training LogisticRegression...
Done!
Training time (secs): 0.006
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.824925816024
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.760563380282
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 240
Training LogisticRegression...
Done!
Training time (secs): 0.004
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.829971181556
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.765957446809
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
```

```
------------------------------------------
Training set size: 250
Training LogisticRegression...
Done!
Training time (secs): 0.005
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.822222222222
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.757142857143
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 260
Training LogisticRegression...
Done!
Training time (secs): 0.006
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.828877005348
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.757142857143
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 270
Training LogisticRegression...
Done!
Training time (secs): 0.006
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.832487309645
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.742857142857
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
```

```
------------------------------------------
Training set size: 280
Training LogisticRegression...
Done!
Training time (secs): 0.007
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.843902439024
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.757142857143
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 290
Training LogisticRegression...
Done!
Training time (secs): 0.004
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.838407494145
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.724637681159
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 300
Training LogisticRegression...
Done!
Training time (secs): 0.007
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.83295194508
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.753623188406
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
```

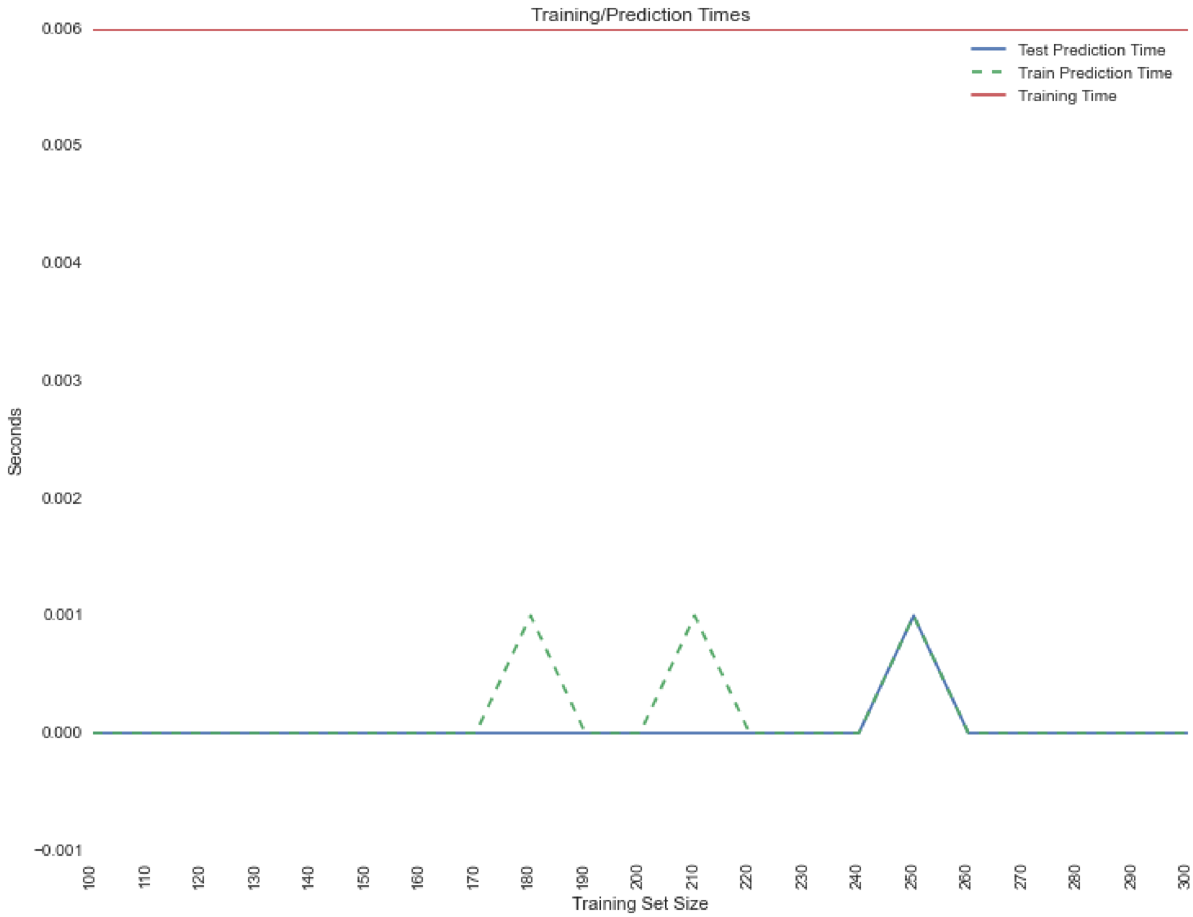
[{'F1_test': 0.70149253731343286, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.86896551724137938, 'Training Size': 100}, {'F1_test': 0.72463768115942029, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.87654320987654322, 'Training Size': 110}, {'F1_test': 0.76056338028169024, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.86813186813186816, 'Training Size': 120}, {'F1_test': 0.75714285714285723, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.87368421052631573, 'Training Size': 130}, {'F1_test': 0.74285714285714299, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.84729064039408875, 'Training Size': 140}, {'F1_test': 0.75714285714285723, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.84403669724770636, 'Training Size': 150}, {'F1_test': 0.76056338028169024, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.8305084745762713, 'Training Size': 160}, {'F1_test': 0.73134328358208955, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.82448979591836724, 'Training Size': 170}, {'F1_test': 0.72463768115942029, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0010001659393310547, 'Predict Test Time': 0.0, 'F1_train': 0.78431372549019607, 'Training Size': 180}, {'F1_test': 0.75524475524475521, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.80565371024734977, 'Training Size': 190}, {'F1_test': 0.75362318840579712, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.8067796610169492, 'Training Size': 200}, {'F1_test': 0.76595744680851063, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0009999275207519531, 'Predict Test Time': 0.0, 'F1_train': 0.81553398058252424, 'Training Size': 210}, {'F1_test': 0.76595744680851063, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.81366459627329191, 'Training Size': 220}, {'F1_test': 0.76056338028169024, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.82492581602373871, 'Training Size': 230}, {'F1_test': 0.76595744680851063, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.82997118155619587, 'Training Size': 240}, {'F1_test': 0.75714285714285723, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0009999275207519531, 'Predict Test Time': 0.0009999275207519531, 'F1_train': 0.82222222222222219, 'Training Size': 250}, {'F1_test': 0.75714285714285723, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.82887700534759357, 'Training Size': 260}, {'F1_test': 0.74285714285714299, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.8324873096446701, 'Training Size': 270}, {'F1_test': 0.75714285714285723, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.84390243902439033, 'Training Size': 280}, {'F1_test': 0.72463768115942029, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.83840749414519911, 'Training Size': 290}, {'F1_test': 0.75362318840579712, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0, 'Predict Test Time': 0.0, 'F1_train': 0.83295194508009152, 'Training Size': 300}]

In [56]:
```
logReg_stats
```

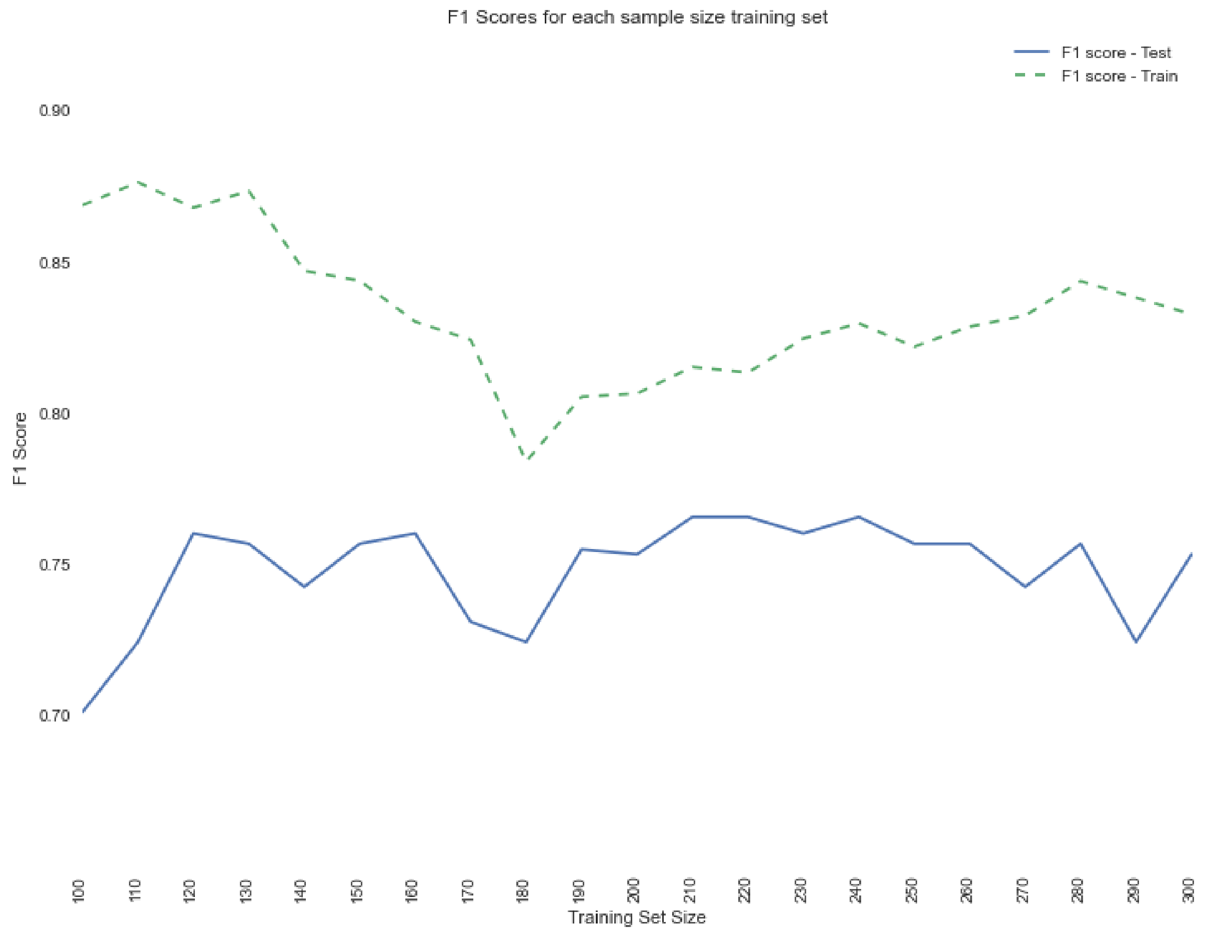
Out[56]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.701493	0.868966	0.000	0.000	100	0.006
1	0.724638	0.876543	0.000	0.000	110	0.006
2	0.760563	0.868132	0.000	0.000	120	0.006
3	0.757143	0.873684	0.000	0.000	130	0.006
4	0.742857	0.847291	0.000	0.000	140	0.006
5	0.757143	0.844037	0.000	0.000	150	0.006
6	0.760563	0.830508	0.000	0.000	160	0.006
7	0.731343	0.824490	0.000	0.000	170	0.006
8	0.724638	0.784314	0.000	0.001	180	0.006
9	0.755245	0.805654	0.000	0.000	190	0.006
10	0.753623	0.806780	0.000	0.000	200	0.006
11	0.765957	0.815534	0.000	0.001	210	0.006
12	0.765957	0.813665	0.000	0.000	220	0.006
13	0.760563	0.824926	0.000	0.000	230	0.006
14	0.765957	0.829971	0.000	0.000	240	0.006
15	0.757143	0.822222	0.001	0.001	250	0.006
16	0.757143	0.828877	0.000	0.000	260	0.006
17	0.742857	0.832487	0.000	0.000	270	0.006
18	0.757143	0.843902	0.000	0.000	280	0.006
19	0.724638	0.838407	0.000	0.000	290	0.006
20	0.753623	0.832952	0.000	0.000	300	0.006

In [54]:
```
time_plot(logReg_stats)
```



In [34]:
```
F1_plot(logReg_stats)
```



As we see that training time does not show any significant change in the time taken for training the model using logistic regression. This is due to the small data set size. The predcition time on the train and test data are very small and we see small pertubations only. We see that plot of the F1_scores on the data set is better metric. We see that F1_score increases as the data set increases , we see a drop for a data set size of 180 and then we see the score increaseing. In contrast the F1_score on the test set increases with the increase in data set size and and we see the model converging.

In [57]:
```
D_Tree_stats = subset_train_predict(tree.DecisionTreeClassifier(random_state=4
2,cv=5),
                                    X_train, y_train,
                                    X_test, y_test,
                                    subset_sizes=subset_sizes)
```

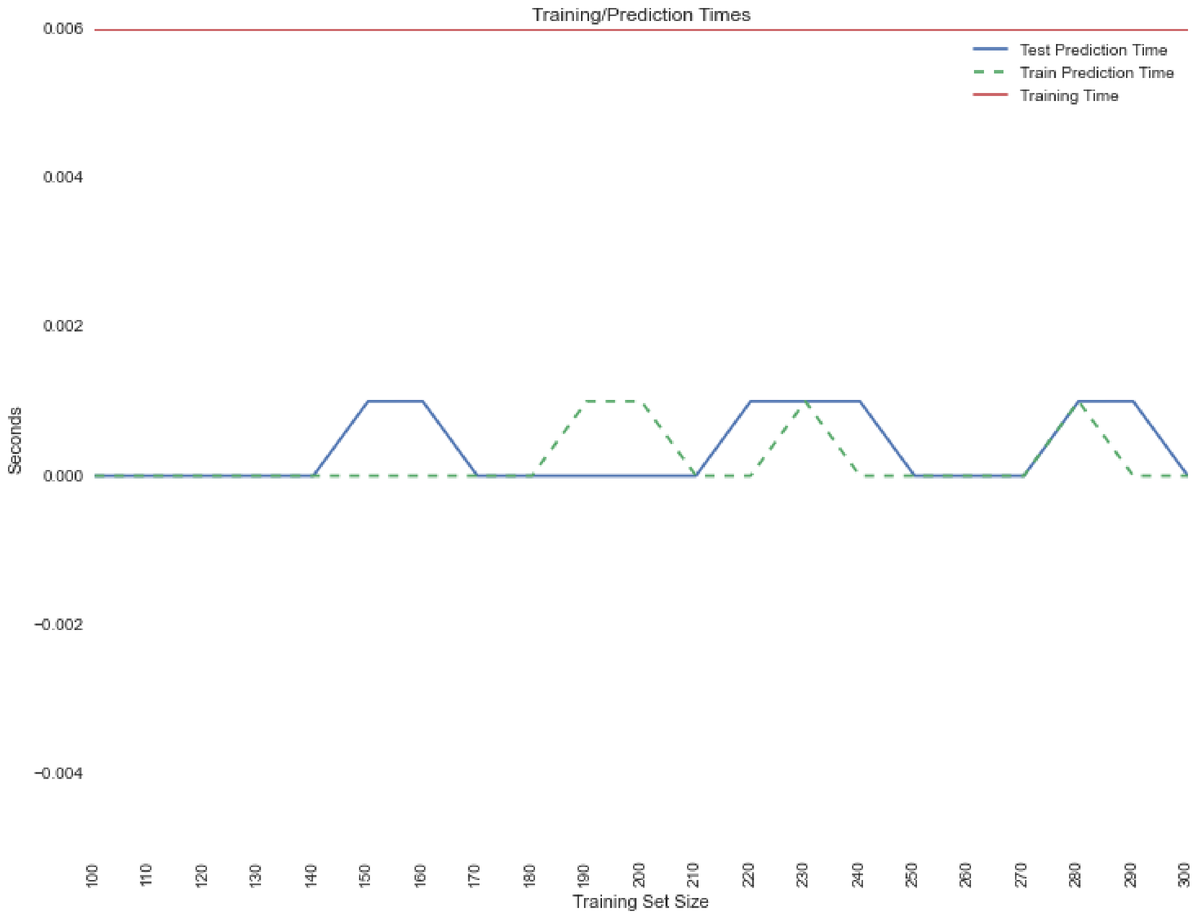
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-57-910e0cd93f2f> in <module>()
----> 1 D_Tree_stats = subset_train_predict(tree.DecisionTreeClassifier(random
_state=42,cv=5),
      2                                     X_train, y_train,
      3                                     X_test, y_test,
      4                                     subset_sizes=subset_sizes)

TypeError: __init__() got an unexpected keyword argument 'cv'
```

In [28]:
```
D_Tree_stats
```

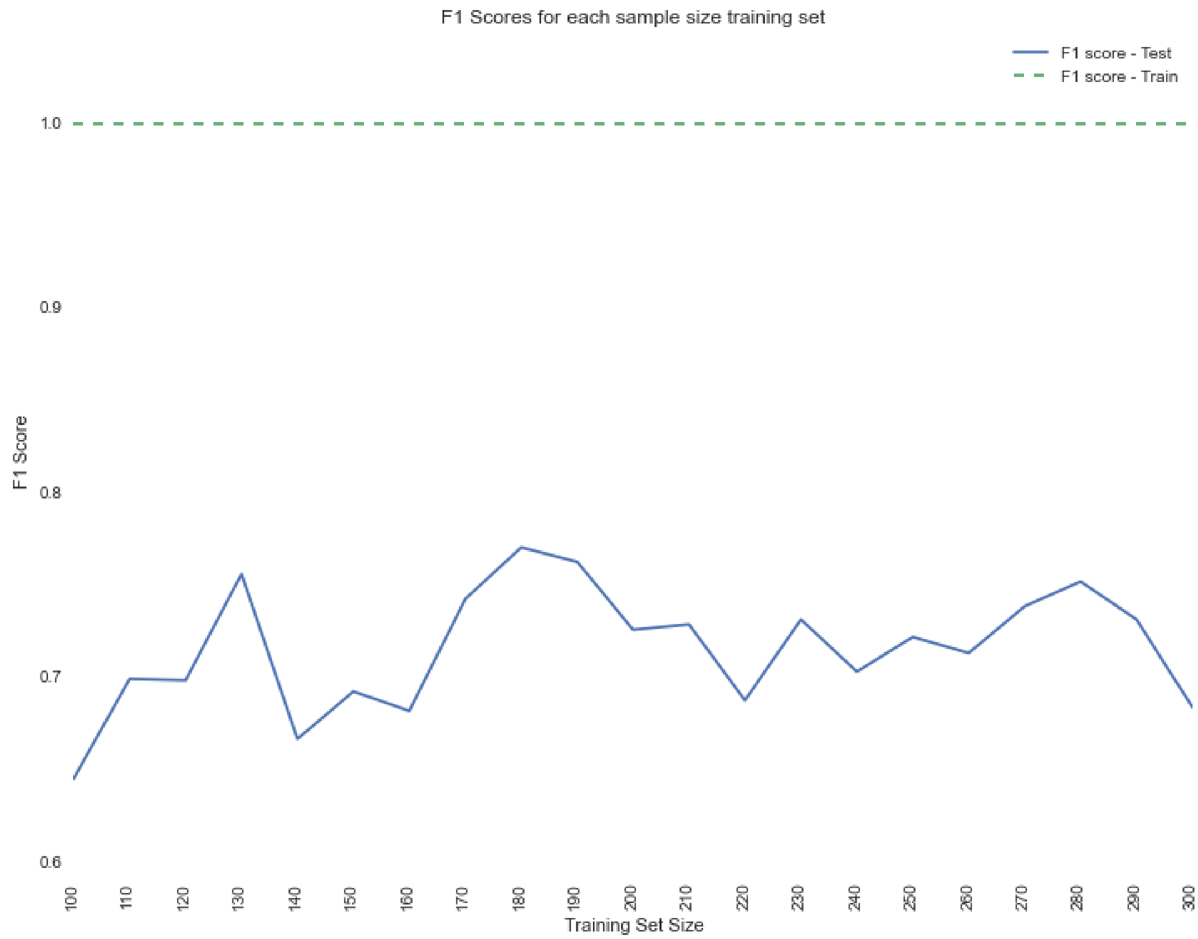
Out[28]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.645161	1	0.000	0.000	100	0.006
1	0.699187	1	0.000	0.000	110	0.006
2	0.698413	1	0.000	0.000	120	0.006
3	0.755906	1	0.000	0.000	130	0.006
4	0.666667	1	0.000	0.000	140	0.006
5	0.692308	1	0.001	0.000	150	0.006
6	0.681818	1	0.001	0.000	160	0.006
7	0.742424	1	0.000	0.000	170	0.006
8	0.770370	1	0.000	0.000	180	0.006
9	0.762590	1	0.000	0.001	190	0.006
10	0.725806	1	0.000	0.001	200	0.006
11	0.728682	1	0.000	0.000	210	0.006
12	0.687500	1	0.001	0.000	220	0.006
13	0.731343	1	0.001	0.001	230	0.006
14	0.703125	1	0.001	0.000	240	0.006
15	0.721805	1	0.000	0.000	250	0.006
16	0.713178	1	0.000	0.000	260	0.006
17	0.738462	1	0.000	0.000	270	0.006
18	0.751880	1	0.001	0.001	280	0.006
19	0.731343	1	0.001	0.000	290	0.006
20	0.683761	1	0.000	0.000	300	0.006

In [52]:
```
time_plot(D_Tree_stats)
```



In [35]:
```
F1_plot(D_Tree_stats)
```



In [ ]:
```
For the Decision Tree model we also see that training time does not show any s
ignificant change in the time taken for training
the model using logistic regression. This is due to the small data set size. T
he predcition time on the train and test data are
very small and we see small pertubations only.
But as we look at F1_score for the train data it shows us a F1_score of 1.00,
which indicates a 100% accurate model.
But as we look at the F1_score on the test set, the F1_score increases with th
e increase in data set size and at data set size
of 180 the F1_score starts to decrease. This behaviour of the f1_score on the
test data and the F1_score on the train data
does indicate an overfir model.
```

In [27]:
```
svm_stats = subset_train_predict(clf, X_train, y_train,
                                         X_test, y_test,
                                         subset_sizes=subset_sizes)
```

```
------------------------------------------
Training set size: 100
Training SVC...
Done!
Training time (secs): 0.001
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.835443037975
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.802547770701
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.000000
------------------------------------------
Training set size: 110
Training SVC...
Done!
Training time (secs): 0.001
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.862275449102
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.8
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 120
Training SVC...
Done!
Training time (secs): 0.001
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.870967741935
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.812903225806
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
```

```
------------------------------------------
Training set size: 130
Training SVC...
Done!
Training time (secs): 0.001
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.857142857143
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.807947019868
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 140
Training SVC...
Done!
Training time (secs): 0.002
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for training set: 0.861244019139
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.8
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 150
Training SVC...
Done!
Training time (secs): 0.002
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.863436123348
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.807947019868
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
```

```
------------------------------------------
Training set size: 160
Training SVC...
Done!
Training time (secs): 0.002
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.858299595142
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.797385620915
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 170
Training SVC...
Done!
Training time (secs): 0.002
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.849420849421
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.797385620915
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 180
Training SVC...
Done!
Training time (secs): 0.002
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.838235294118
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.789473684211
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
```

```
------------------------------------------
Training set size: 190
Training SVC...
Done!
Training time (secs): 0.003
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.843537414966
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.797385620915
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 200
Training SVC...
Done!
Training time (secs): 0.003
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.843137254902
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.81045751634
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 210
Training SVC...
Done!
Training time (secs): 0.004
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for training set: 0.844036697248
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.805194805195
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
```

```
----------------------------------------
Training set size: 220
Training SVC...
Done!
Training time (secs): 0.003
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
F1 score for training set: 0.850439882698
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.794701986755
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
----------------------------------------
Training set size: 230
Training SVC...
Done!
Training time (secs): 0.004
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
F1 score for training set: 0.85393258427
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.797385620915
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
----------------------------------------
Training set size: 240
Training SVC...
Done!
Training time (secs): 0.004
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
F1 score for training set: 0.844919786096
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.812903225806
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
```

```
------------------------------------------
Training set size: 250
Training SVC...
Done!
Training time (secs): 0.004
Predicting labels using SVC...
Done!
Prediction time (secs): 0.004000
F1 score for training set: 0.854922279793
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.805194805195
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
------------------------------------------
Training set size: 260
Training SVC...
Done!
Training time (secs): 0.005
Predicting labels using SVC...
Done!
Prediction time (secs): 0.003000
F1 score for training set: 0.853598014888
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.805194805195
Predicting labels using SVC...
Done!
Prediction time (secs): 0.004000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
------------------------------------------
Training set size: 270
Training SVC...
Done!
Training time (secs): 0.006
Predicting labels using SVC...
Done!
Prediction time (secs): 0.004000
F1 score for training set: 0.855791962175
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.812903225806
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
```

```
----------------------------------------
Training set size: 280
Training SVC...
Done!
Training time (secs): 0.006
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005000
F1 score for training set: 0.857798165138
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.812903225806
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
----------------------------------------
Training set size: 290
Training SVC...
Done!
Training time (secs): 0.006
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005000
F1 score for training set: 0.862831858407
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
F1 score for test set: 0.812903225806
Predicting labels using SVC...
Done!
Prediction time (secs): 0.004000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001000
----------------------------------------
Training set size: 300
Training SVC...
Done!
Training time (secs): 0.006
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005000
F1 score for training set: 0.866379310345
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
F1 score for test set: 0.805194805195
Predicting labels using SVC...
Done!
Prediction time (secs): 0.004000
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002000
```

[{'F1_test': 0.80254777070063688, 'Training Time': 0.00599980354309082, 'Predi
ct Train Time': 0.0009999275207519531, 'Predict Test Time': 0.0, 'F1_train':
 0.83544303797468344, 'Training Size': 100}, {'F1_test': 0.79999999999999993,
 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.000999927520751
9531, 'Predict Test Time': 0.0009999275207519531, 'F1_train': 0.86227544910179
643, 'Training Size': 110}, {'F1_test': 0.81290322580645158, 'Training Time':
 0.00599980354309082, 'Predict Train Time': 0.0009999275207519531, 'Predict Te
st Time': 0.0009999275207519531, 'F1_train': 0.87096774193548399, 'Training Si
ze': 120}, {'F1_test': 0.80794701986754958, 'Training Time': 0.005999803543090
82, 'Predict Train Time': 0.002000093460083008, 'Predict Test Time': 0.0009999
275207519531, 'F1_train': 0.8571428571428571, 'Training Size': 130}, {'F1_tes
t': 0.80000000000000016, 'Training Time': 0.00599980354309082, 'Predict Train
 Time': 0.0009999275207519531, 'Predict Test Time': 0.0010001659393310547, 'F1
_train': 0.86124401913875603, 'Training Size': 140}, {'F1_test': 0.80794701986
754958, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.00099992
75207519531, 'Predict Test Time': 0.0009999275207519531, 'F1_train': 0.8634361
2334801756, 'Training Size': 150}, {'F1_test': 0.79738562091503273, 'Training
 Time': 0.00599980354309082, 'Predict Train Time': 0.0009999275207519531, 'Pre
dict Test Time': 0.0010001659393310547, 'F1_train': 0.85829959514170051, 'Trai
ning Size': 160}, {'F1_test': 0.79738562091503273, 'Training Time': 0.00599980
354309082, 'Predict Train Time': 0.0009999275207519531, 'Predict Test Time':
 0.0010001659393310547, 'F1_train': 0.8494208494208495, 'Training Size': 170},
 {'F1_test': 0.78947368421052633, 'Training Time': 0.00599980354309082, 'Predi
ct Train Time': 0.002000093460083008, 'Predict Test Time': 0.00099992752075195
31, 'F1_train': 0.83823529411764719, 'Training Size': 180}, {'F1_test': 0.7973
8562091503273, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0
02000093460083008, 'Predict Test Time': 0.0010001659393310547, 'F1_train': 0.8
4353741496598644, 'Training Size': 190}, {'F1_test': 0.8104575163398694, 'Trai
ning Time': 0.00599980354309082, 'Predict Train Time': 0.003000020980834961,
 'Predict Test Time': 0.0010001659393310547, 'F1_train': 0.84313725490196068,
 'Training Size': 200}, {'F1_test': 0.80519480519480513, 'Training Time': 0.00
599980354309082, 'Predict Train Time': 0.0019998550415039062, 'Predict Test Ti
me': 0.0009999275207519531, 'F1_train': 0.84403669724770636, 'Training Size':
 210}, {'F1_test': 0.79470198675496684, 'Training Time': 0.00599980354309082,
 'Predict Train Time': 0.003000020980834961, 'Predict Test Time': 0.0020000934
60083008, 'F1_train': 0.85043988269794712, 'Training Size': 220}, {'F1_test':
 0.79738562091503273, 'Training Time': 0.00599980354309082, 'Predict Train Tim
e': 0.003000020980834961, 'Predict Test Time': 0.0010001659393310547, 'F1_trai
n': 0.8539325842696629, 'Training Size': 230}, {'F1_test': 0.8129032258064515
8, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0030000209808
34961, 'Predict Test Time': 0.0010001659393310547, 'F1_train': 0.8449197860962
5673, 'Training Size': 240}, {'F1_test': 0.80519480519480513, 'Training Time':
 0.00599980354309082, 'Predict Train Time': 0.003000020980834961, 'Predict Tes
t Time': 0.0010001659393310547, 'F1_train': 0.85492227979274615, 'Training Siz
e': 250}, {'F1_test': 0.80519480519480513, 'Training Time': 0.0059998035430908
2, 'Predict Train Time': 0.004000186920166016, 'Predict Test Time': 0.00199985
50415039062, 'F1_train': 0.85359801488833742, 'Training Size': 260}, {'F1_tes
t': 0.81290322580645158, 'Training Time': 0.00599980354309082, 'Predict Train
 Time': 0.005000114440917969, 'Predict Test Time': 0.0010001659393310547, 'F1_
train': 0.8557919621749408, 'Training Size': 270}, {'F1_test': 0.8129032258064
5158, 'Training Time': 0.00599980354309082, 'Predict Train Time': 0.0050001144
40917969, 'Predict Test Time': 0.002000093460083008, 'F1_train': 0.85779816513
761464, 'Training Size': 280}, {'F1_test': 0.81290322580645158, 'Training Tim
e': 0.00599980354309082, 'Predict Train Time': 0.004000186920166016, 'Predict
 Test Time': 0.0009999275207519531, 'F1_train': 0.86283185840707965, 'Training
 Size': 290}, {'F1_test': 0.80519480519480513, 'Training Time': 0.005999803543
09082, 'Predict Train Time': 0.003999948501586914, 'Predict Test Time': 0.0020

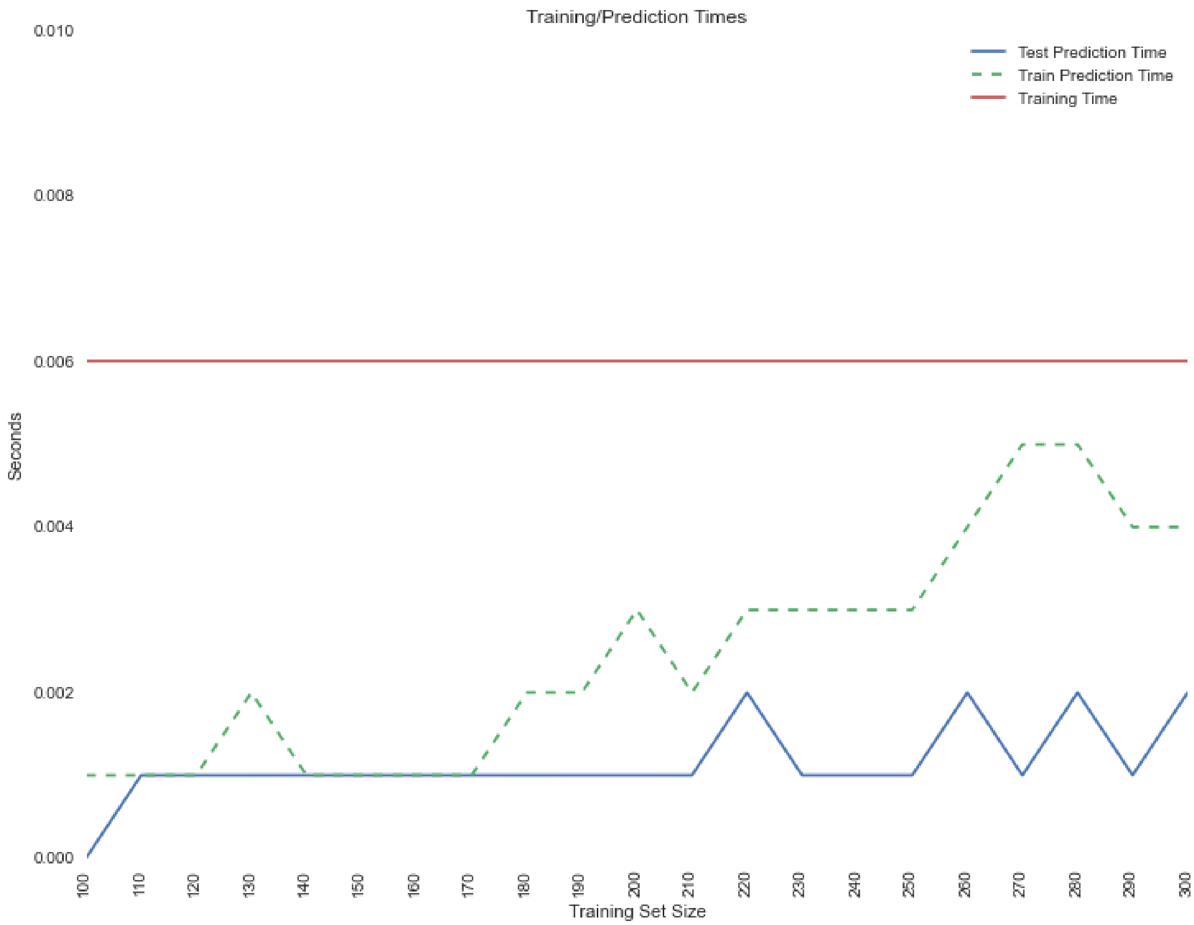
```
00093460083008, 'F1_train': 0.86637931034482762, 'Training Size': 300}]
```

In [31]:
```
svm_stats
```

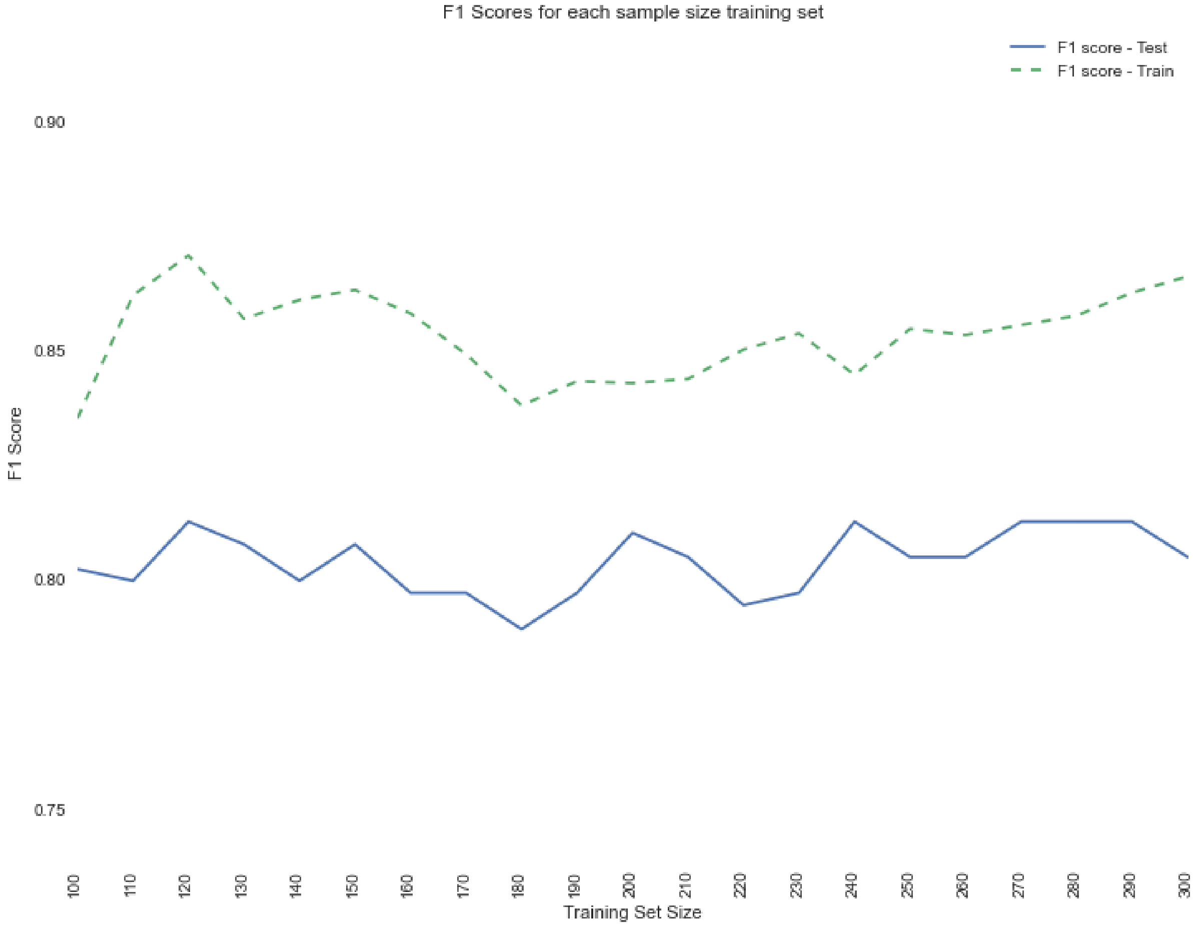
Out[31]:

	F1_test	F1_train	Predict Test Time	Predict Train Time	Training Size	Training Time
0	0.802548	0.835443	0.000	0.001	100	0.006
1	0.800000	0.862275	0.001	0.001	110	0.006
2	0.812903	0.870968	0.001	0.001	120	0.006
3	0.807947	0.857143	0.001	0.002	130	0.006
4	0.800000	0.861244	0.001	0.001	140	0.006
5	0.807947	0.863436	0.001	0.001	150	0.006
6	0.797386	0.858300	0.001	0.001	160	0.006
7	0.797386	0.849421	0.001	0.001	170	0.006
8	0.789474	0.838235	0.001	0.002	180	0.006
9	0.797386	0.843537	0.001	0.002	190	0.006
10	0.810458	0.843137	0.001	0.003	200	0.006
11	0.805195	0.844037	0.001	0.002	210	0.006
12	0.794702	0.850440	0.002	0.003	220	0.006
13	0.797386	0.853933	0.001	0.003	230	0.006
14	0.812903	0.844920	0.001	0.003	240	0.006
15	0.805195	0.854922	0.001	0.003	250	0.006
16	0.805195	0.853598	0.002	0.004	260	0.006
17	0.812903	0.855792	0.001	0.005	270	0.006
18	0.812903	0.857798	0.002	0.005	280	0.006
19	0.812903	0.862832	0.001	0.004	290	0.006
20	0.805195	0.866379	0.002	0.004	300	0.006

In [55]:
```
time_plot(svm_stats)
```



In [36]:
```
F1_plot(svm_stats)
```



In [ ]:
```
The support vector machine model as compared to the Logistic Regression and th
e Decision Tree model the prediction train and
test time increase as the data set size increases.
As we look at F1_score for the train and test data thhe F1_score is less than
the it shows us a F1_score of 1.00, which indicates a 100% accurate model.
But as we look at the F1_score on the train set increases. The F1_score on the
 test score dos have some perturbations.
```

Based on these observations on the three models, the logistic regression is the model of choice for predicting the student

graduation rate.

5. Choosing the Best Model

- Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model. Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?
- In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a Decision Tree or Support Vector Machine, how does it make a prediction).
- Fine-tune the model. Use Gridsearch with at least one important parameter tuned and with at least 3 settings. Use the entire training set for this.
- What is the model's final F_1 score?

Based on the three models used for classification F1 scores obtained from Support vector Machine and Logistic Regression were identical. Hence Grid Search CV was used to fine tune the model for the support vector machine and logistic regression.

In [37]:
```
from sklearn.metrics import f1_score
from sklearn.metrics import make_scorer
f1_scorer = make_scorer(f1_score, pos_label='yes')
```

In [38]:
```
# TODO: Fine-tune your model and report the best F1 score
"""
from sklearn import grid_search
from sklearn.metrics import f1_score
from sklearn.metrics import make_scorer
from sklearn.cross_validation import StratifiedShuffleSplit



cv = StratifiedShuffleSplit(y_train, random_state=42)

clf = svm.SVC()
param_grid = [
  {'C': [1, 10, 100, 200, 300, 400, 500, 600, 700],
   'gamma': [1e-2, 1e-3, 1e-4, 1e-5, 1e-6],
   'kernel': ['rbf','linear'], 'tol':[1e-3, 1e-4, 1e-5, 1e-6]
  }
 ]

#regressor = grid_search.GridSearchCV(clf, param_grid,cv=cv, scoring='f1_weigh
ted')
regressor = grid_search.GridSearchCV(clf, param_grid,cv=cv, scoring=f1_scorer)
regressor.fit(X_train, y_train)
reg = regressor.best_estimator_
print reg
train_f1_score = predict_labels(reg, X_train, y_train)[0]
print "F1 score for training set: {}".format(train_f1_score)

print "F1 score for test set: {}".format(predict_labels(reg, X_test, y_test)
[0])
"""
```

```
Out[38]: '\nfrom sklearn import grid_search\nfrom sklearn.metrics import f1_score\nfrom
          sklearn.metrics import make_scorer\nfrom sklearn.cross_validation import Stra
         tifiedShuffleSplit\n\n\n\ncv = StratifiedShuffleSplit(y_train, random_state=4
         2)\n\nclf = svm.SVC()\nparam_grid = [\n  {\'C\': [1, 10, 100, 200, 300, 400, 5
         00, 600, 700],\n   \'gamma\': [1e-2, 1e-3, 1e-4, 1e-5, 1e-6],\n   \'kernel\':
          [\'rbf\',\'linear\'], \'tol\':[1e-3, 1e-4, 1e-5, 1e-6]\n  }\n ]\n\n#regressor
          = grid_search.GridSearchCV(clf, param_grid,cv=cv, scoring=\'f1_weighted\')\nr
         egressor = grid_search.GridSearchCV(clf, param_grid,cv=cv, scoring=f1_scorer)
         \nregressor.fit(X_train, y_train)\nreg = regressor.best_estimator_\nprint reg
         \ntrain_f1_score = predict_labels(reg, X_train, y_train)[0]\nprint "F1 score f
         or training set: {}".format(train_f1_score)\n\nprint "F1 score for test set:
          {}".format(predict_labels(reg, X_test, y_test)[0])\n'
```

In [39]:
```
from sklearn import grid_search
from sklearn.metrics import f1_score
from sklearn.metrics import make_scorer
from sklearn.cross_validation import StratifiedShuffleSplit

cv = StratifiedShuffleSplit(y_train, random_state=42)

logReg = LogisticRegression(random_state=42)
param_grid = {'penalty':['l1','l2'],'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]
}

regressor_LogReg = grid_search.GridSearchCV(logReg, param_grid, scoring= f1_sc
orer, cv=cv)

regressor_LogReg.fit(X_train_scaled, y_train_scaled)
regLogRef = regressor_LogReg.best_estimator_
print regLogRef
train_f1_score_LogReg = predict_labels(regLogRef, X_train_scaled, y_train_scal
ed)[0]
print "F1 score for training set: {}".format(train_f1_score_LogReg)

print "F1 score for test set: {}".format(predict_labels(regLogRef, X_test_scal
ed, y_test_scaled)[0])
```

```
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l1', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0)
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for training set: 0.82905982906
Predicting labels using LogisticRegression...
Done!
Prediction time (secs): 0.000000
F1 score for test set: 0.789115646259

C:\Users\aw634c\AppData\Local\Continuum\Anaconda\lib\site-packages\sklearn\met
rics\classification.py:958: UndefinedMetricWarning: F-score is ill-defined and
 being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [40]:
```
print "BestTuned Parameters for Logistic Regression"
print regressor_LogReg.best_params_
```

```
BestTuned Parameters for Logistic Regression
{'penalty': 'l1', 'C': 0.1}
```

In [41]:
```
regressor_LogReg
```

Out[41]:
```
GridSearchCV(cv=StratifiedShuffleSplit(labels=['yes' 'yes' ..., 'yes' 'no'], n
_iter=10, test_size=0.1, random_state=42),
       error_score='raise',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, fit_
intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0),
       fit_params={}, iid=True, loss_func=None, n_jobs=1,
       param_grid={'penalty': ['l1', 'l2'], 'C': [0.001, 0.01, 0.1, 1, 10, 10
0, 1000]},
       pre_dispatch='2*n_jobs', refit=True, score_func=None,
       scoring=make_scorer(f1_score, pos_label=yes), verbose=0)
```

Choice of the Best Model

Logistic regresssion Model was chosen as the model to predict student intervention. Logistic regression model as seen had a better train prediction time as compared to the SVM. The logistic regression model with the regularization did not overfit the model as compared to the Decision Tree model. A F1 score of 0.83 was obtained using the logitic regression model on the train data and 0.78 on the test data.

SVM and Logistic Regression took almost the same time to train and test the data.

Based on the analysis Logistic Regression with L1 penalty provides the best model in predicting the graduation rates of the students.

The Logistic Regression model multiplies the variables with the weights obtained into a final score. This final score we get is between 0 and 1. This value of between 0 and 1 gives the probability of the student passing.

If the score or the probablity is greater than 0.5 the probability, the model predicts tha the student will pass and if not the model predicts the student fails.

Based on this model if we look at the coefficients of the model we see that Logistic Regression forces some of the coefficients to zero.

By looking at the coeffcient values and the values associated with it the logistic regression model will help us in improving the graduation rate.

The features failures, goout, absences and reason_other have negative coefficients. If the school could work on improving these features the student graduation rate will improve. On the same grounds the same can be said of 4 other features Walc, age , farmrel and Medu which have positive coeeficients. Improving these features of the participating students would also increase the graduation rate.

The final tuned F1 score on the test set using Logistic Regression with L2 penalty is 0.78.

In [42]:
```
regLogRef.coef_
```

Out[42]:
```
array([[ 0.        ,  0.        ,  0.        ,  0.        ,  0.07234218,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.04485458,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        , -0.82986468,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.01110423,  0.        , -0.15189878,  0.        ,
         0.07401609,  0.        , -0.01334525]])
```

In [43]:
```
lr_coeffs = pd.DataFrame({'Feature': X_train.columns,
                          'Coefficient': regLogRef.coef_[0]},
                            index=X_train.columns)
```

In [44]:
```
lr_coeffs
```

Out[44]:

	Coefficient	Feature
school_GP	0.000000	school_GP
school_MS	0.000000	school_MS
sex_F	0.000000	sex_F
sex_M	0.000000	sex_M
age	0.072342	age
address_R	0.000000	address_R
address_U	0.000000	address_U
famsize_GT3	0.000000	famsize_GT3
famsize_LE3	0.000000	famsize_LE3
Pstatus_A	0.000000	Pstatus_A
Pstatus_T	0.000000	Pstatus_T
Medu	0.044855	Medu
Fedu	0.000000	Fedu
Mjob_at_home	0.000000	Mjob_at_home
Mjob_health	0.000000	Mjob_health
Mjob_other	0.000000	Mjob_other
Mjob_services	0.000000	Mjob_services
Mjob_teacher	0.000000	Mjob_teacher
Fjob_at_home	0.000000	Fjob_at_home
Fjob_health	0.000000	Fjob_health
Fjob_other	0.000000	Fjob_other
Fjob_services	0.000000	Fjob_services
Fjob_teacher	0.000000	Fjob_teacher
reason_course	0.000000	reason_course
reason_home	0.000000	reason_home
reason_other	0.000000	reason_other
reason_reputation	0.000000	reason_reputation
guardian_father	0.000000	guardian_father
guardian_mother	0.000000	guardian_mother
guardian_other	0.000000	guardian_other
traveltime	0.000000	traveltime
studytime	0.000000	studytime
failures	-0.829865	failures

schoolsup	0.000000	schoolsup

In [45]:
```
lr_coeffs.sort(['Coefficient'],ascending=[False] ).head()
```

Out[45]:

	Coefficient	Feature
Walc	0.074016	Walc
age	0.072342	age
Medu	0.044855	Medu
famrel	0.011104	famrel
school_GP	0.000000	school_GP

In [62]:
```
lr_coeffs.sort(['Coefficient'],ascending=[True] ).head()
```

Out[62]:

	Coefficient	Feature
failures	-0.829865	failures
goout	-0.151899	goout
absences	-0.013345	absences
reason_other	0.000000	reason_other
reason_reputation	0.000000	reason_reputation

In [60]:

In [ ]: