In [17]:
```
from lesson_functions import *
import glob
from os.path import join
from sklearn.metrics import precision_score, classification_report, accu
racy_score
from sklearn.svm import LinearSVC, SVC, NuSVC
from sklearn.preprocessing import StandardScaler
import time
import pickle
```

In [2]:
```
def train_test_split(X, y, test_size=0.2):
    i = int((1 - test_size) * X.shape[0]) + 1
    X_train, X_test = np.split(X, [i])
    y_train, y_test = np.split(y, [i])
    return X_train, X_test, y_train, y_test
```

Parameters

While a lot of the parameters were taken from lessons in the class, others came from forum discussions, and others were purely from experimentation.

In [3]:
```
color_space = 'HLS'  # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 9  # HOG orientations
pix_per_cell = 16  # HOG pixels per cell
cell_per_block = 2  # HOG cells per block
hog_channel = "ALL"  # Can be 0, 1, 2, or "ALL"
spatial_size = (16, 16)  # Spatial binning dimensions
hist_bins = 16  # Number of histogram bins
spatial_feat = True  # Spatial features on or off
hist_feat = True  # Histogram features on or off
hog_feat = True  # HOG features on or off
```

Data Source

In [4]:
```
data_dir = "./images/"
cars = glob.glob(join(data_dir, "vehicles/*/*.png"))
notcars = glob.glob(join(data_dir, "non-vehicles/*/*.png"))
```

Extract Features for Cars and Non-Cars Using Class-Provided Methods and Passing in Appropriate Parameters

In [5]:
```
car_features = extract_features(cars, color_space=color_space,
                                spatial_size=spatial_size, hist_bins=his
t_bins,
                                orient=orient, pix_per_cell=pix_per_cell
,
                                cell_per_block=cell_per_block,
                                hog_channel=hog_channel, spatial_feat=sp
atial_feat,
                                hist_feat=hist_feat, hog_feat=hog_feat)
notcar_features = extract_features(notcars, color_space=color_space,
                                   spatial_size=spatial_size, hist_bins=
hist_bins,
                                   orient=orient, pix_per_cell=pix_per_c
ell,
                                   cell_per_block=cell_per_block,
                                   hog_channel=hog_channel, spatial_feat
=spatial_feat,
                                   hist_feat=hist_feat, hog_feat=hog_fea
t)
```

```
/Users/seth.bunke/anaconda/envs/carnd-term1/lib/python3.5/site-package
s/skimage/feature/_hog.py:119: skimage_deprecation: Default value of `b
lock_norm`==`L1` is deprecated and will be changed to `L2-Hys` in v0.15
  'be changed to `L2-Hys` in v0.15', skimage_deprecation)
```

Split Data

In [7]:
```
car_X_train, car_X_test, car_y_train, car_y_test = \
    train_test_split(np.array(car_features), np.ones(len(car_features)),
 test_size=0.2)

noncar_X_train, noncar_X_test, noncar_y_train, noncar_y_test = \
    train_test_split(np.array(notcar_features), np.zeros(len(notcar_feat
ures)), test_size=0.2)

X_train = np.vstack((car_X_train, noncar_X_train))
y_train = np.hstack((car_y_train, noncar_y_train))
X_test = np.vstack((car_X_test, noncar_X_test))
y_test = np.hstack((car_y_test, noncar_y_test))
```

Scaling the Data

In [8]:
```
X_scaler = StandardScaler().fit(np.vstack((X_train, X_test)))
X_train = X_scaler.transform(X_train)
X_test = X_scaler.transform(X_test)
```

Training the Classifier

Through disucssions on the class forum as well as my own testing I selected SVC as the model. The comparison of some models is shown below. The tuning of paramters did not make an appreciable difference in accuracy. Some of that tuning was based on this dicussion:
https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel (https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel)

In [13]:
```
def train_model_report_accuracy(model):
    global X_train
    global y_train
    global X_test
    global y_test
    # Check the training time for the SVC
    t = time.time()
    model.fit(X_train, y_train)
    t2 = time.time()
    print(round(t2 - t, 2), 'Seconds to train model...')
    t = time.time()
    p = model.predict(X_test)
    t2 = time.time()
    # Check the prediction time
    print(round(t2 - t, 4), 'Seconds to predict with model...')
    # Check the score of the SVC
    print('Test accuracy: {:.4f}'.format(accuracy_score(y_test, p)))
    print("Test precision: {:.4f}".format(precision_score(y_test, p)))
    print("")
    print(classification_report(y_test, p, digits=4))
    return model
```

In [14]:
```
model = LinearSVC(dual=True, C=0.01)
```

In [15]:
```
train_model_report_accuracy(model)
```

```
3.97 Seconds to train model...
0.6391 Seconds to predict with model...
Test accuracy: 0.9772
Test precision: 0.9635

             precision    recall  f1-score   support

        0.0     0.9914    0.9632    0.9771      1793
        1.0     0.9635    0.9915    0.9773      1758

avg / total     0.9776    0.9772    0.9772      3551
```

Out[15]:
```
LinearSVC(C=0.01, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
```

In [21]:
```
model_1 = SVC()
train_model_report_accuracy(model_1)
```

```
54.37 Seconds to train model...
12.6779 Seconds to predict with model...
Test accuracy: 0.9904
Test precision: 0.9826

             precision    recall  f1-score   support

        0.0     0.9983    0.9827    0.9904      1793
        1.0     0.9826    0.9983    0.9904      1758

avg / total     0.9905    0.9904    0.9904      3551
```

Out[21]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [22]:
```
from sklearn import tree
model_2 = tree.DecisionTreeClassifier()
train_model_report_accuracy(model_2)
```

```
76.22 Seconds to train model...
0.05 Seconds to predict with model...
Test accuracy: 0.9285
Test precision: 0.8841

             precision    recall  f1-score   support

        0.0     0.9831    0.8734    0.9250      1793
        1.0     0.8841    0.9846    0.9316      1758

avg / total     0.9340    0.9285    0.9283      3551
```

Out[22]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=N
one,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=N
one,
            splitter='best')
```

Train Using All Data

In [23]:
```
X = np.vstack((X_train, X_test))
y = np.hstack((y_train, y_test))

t = time.time()
final_model = SVC()
final_model.fit(X, y)
t2 = time.time()
print(round(t2 - t, 2), 'Seconds to train final model...')
```

```
74.11 Seconds to train final model...
```

Save Model

In [24]:
```
with open("final_model.p", "wb") as ofile:
    pickle.dump([final_model, X_scaler], ofile)
```

In [ ]:
```

```