In [13]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import KFold
```

In [14]:
```
df = pd.read_csv('Auto.csv')
df_copy = df.copy()
```

In [15]:
```
# Eliminates the rows (instances) with '?' as a predictor value
df_copy['horsepower'] = pd.to_numeric(df_copy['horsepower'], errors='coerce')
df_copy = df_copy.dropna()
# df_copy['name'] = df_copy['name'].str.split(' ').str[0]

df_copy = df_copy.drop('name', 1)
df_copy.head()
```

Out[15]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	1
1	15.0	8	350.0	165.0	3693	11.5	70	1
2	18.0	8	318.0	150.0	3436	11.0	70	1
3	16.0	8	304.0	150.0	3433	12.0	70	1
4	17.0	8	302.0	140.0	3449	10.5	70	1

Problem a

In [16]:
```
mpg_median = df_copy['mpg'].median()

# Create 'mpg01' column, 1 => mpg > mpg_median, 0 => mpg < mpg_median
df_copy.loc[df_copy['mpg'] >= mpg_median, 'mpg01'] = 1
df_copy.loc[df_copy['mpg'] < mpg_median, 'mpg01'] = 0
df_copy = df_copy.drop('mpg', 1)
```

In [ ]:
```

```

Problem b

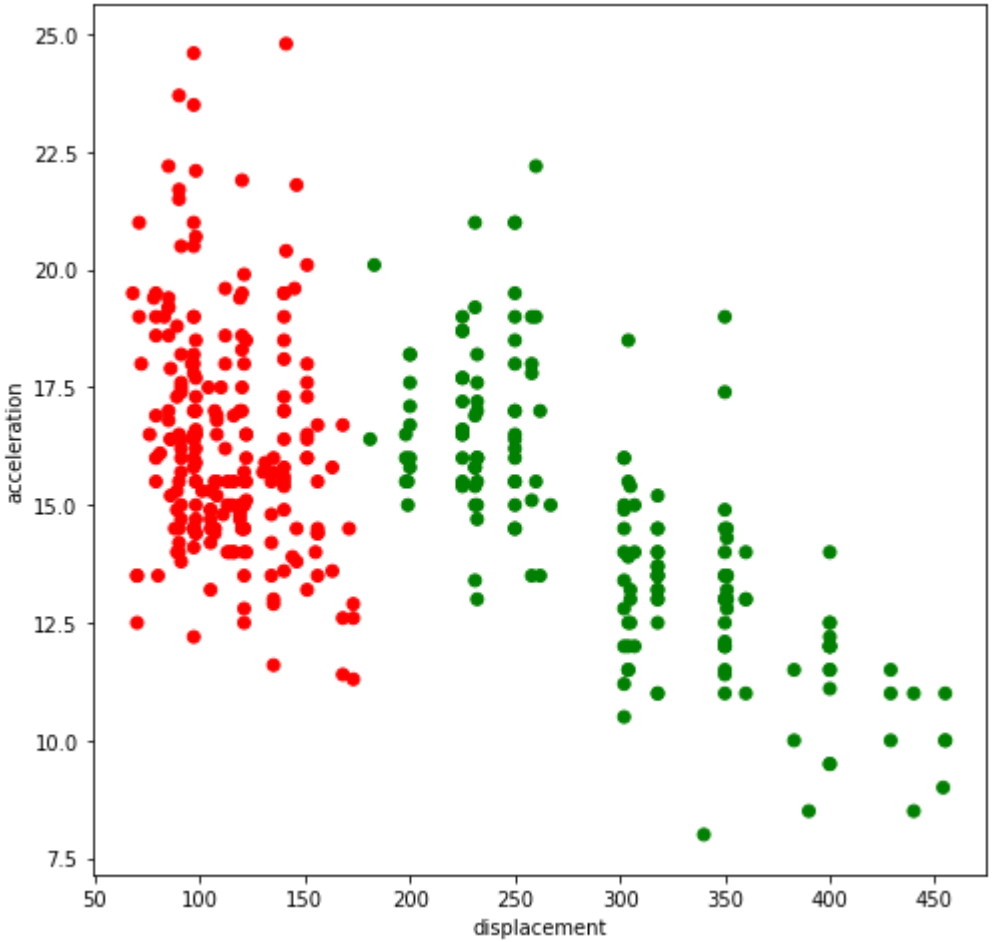
In [17]:
```
# Extract predictor x and response y as arrays
y = np.asarray(df_copy['mpg01'])
X = np.asarray(df_copy[['displacement', 'acceleration']])

K=10
# K-Fold splitter
kfold = KFold(n_splits=K, shuffle=True)
sum_test_errors = 0 # Initialize the total test error
for train_index, test_index in kfold.split(X): # For each group
    # Obtain training and testing data
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Create Linear Regression model and fit to the training data
    SV_classifier = SVC(kernel='linear')
    SV_classifier.fit(X_train, y_train);

    # Make predictions
    y_pred = SV_classifier.predict(X_test)
    y_pred = 1*(y_pred >= 0.5)

    tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
    test_error = (fp+fn)/(tn+fp+fn+tp)
    sum_test_errors = sum_test_errors + test_error

# cross-validated test error
cv_error = sum_test_errors/K

print("The Cross-Validation Error is " + repr(cv_error))
```

```
The Cross-Validation Error is 0.10237179487179486
```

In [18]:
```
y = df_copy['mpg01']
X = df_copy[['displacement', 'acceleration']]

SV_classifier = SVC(kernel='linear')
SV_classifier.fit(X, y);

# Make predictions
y_pred = SV_classifier.predict(X)
y_pred = 1*(y_pred >= 0.5)

tn, fp, fn, tp = confusion_matrix(y, y_pred).ravel()
accuracy = (tp+tn)/(tn+fp+fn+tp)
print('The accuracy rate for a SVC with a Linear Kernel is ' + repr(accuracy))

# Define colors for the class labels: RED for 1, GREEN for 0
colors = ['green','red']

fig = plt.figure(figsize=(8,8))
plt.scatter(X['displacement'], X['acceleration'], c=y_pred, cmap=matplotlib.colors.ListedColormap(colors))
plt.xlabel('displacement')
plt.ylabel('acceleration')
```

```
The accuracy rate for a SVC with a Linear Kernel is 0.8979591836734694
```

Out[18]: Text(0,0.5,'acceleration')

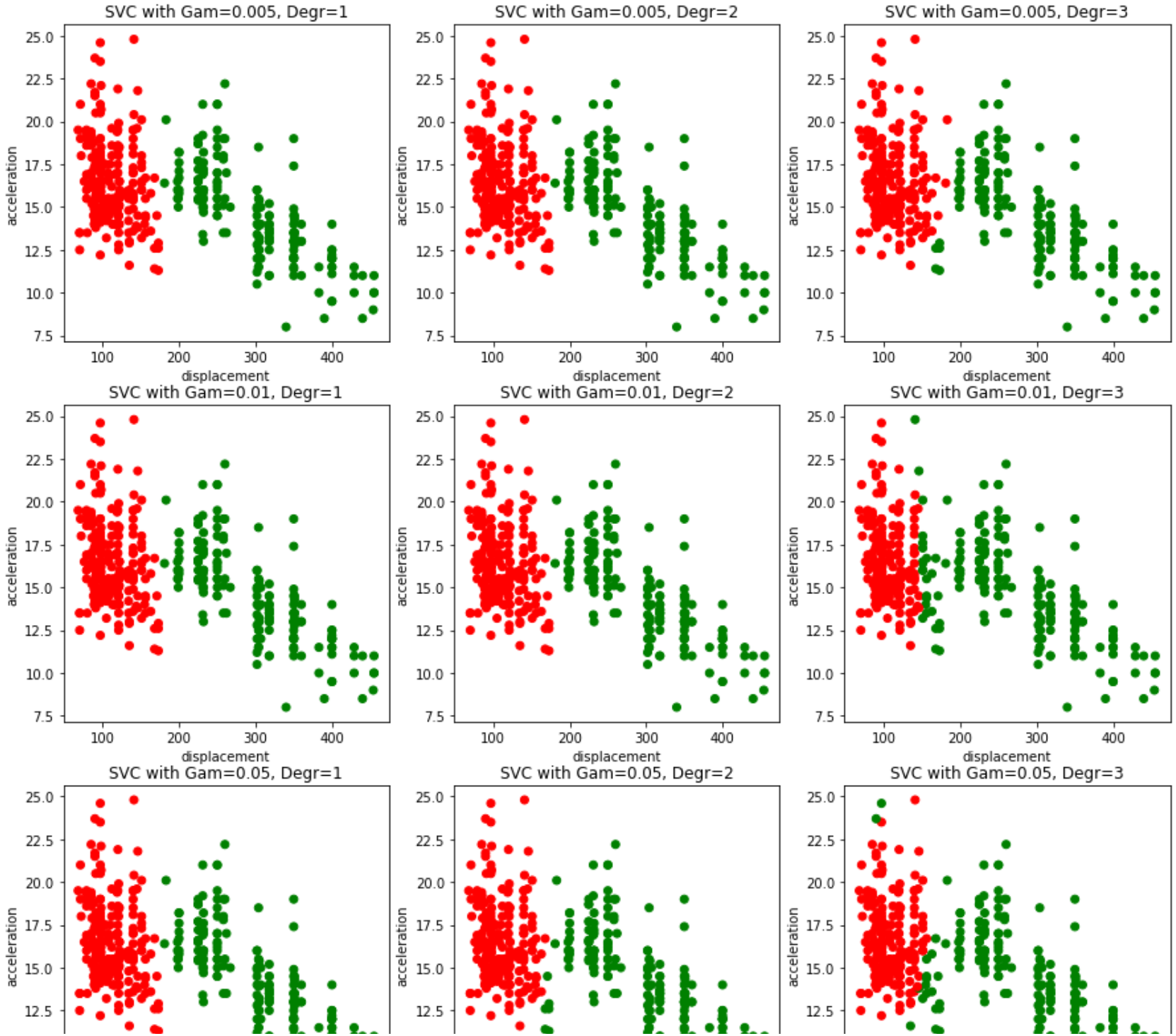


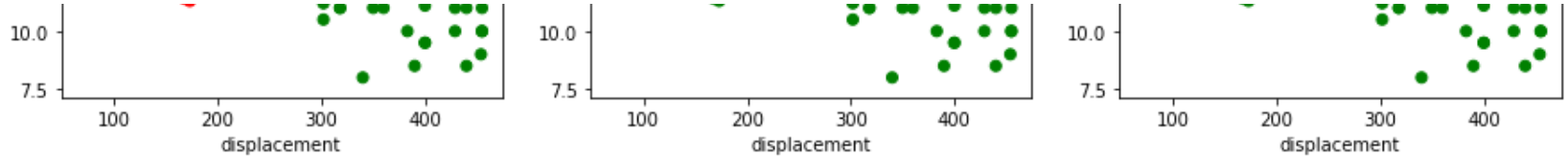
Problem c

Polynomial Kernel

In [19]:
```
gamma_values_list = [0.005, 0.01, 0.05]
degrees_list = [1, 2, 3]

# Create four polar axes and access them through the returned array
fig, axes = plt.subplots(len(gamma_values_list), len(degrees_list), figsize=(15,15))
row = 0
for gamma in (gamma_values_list):
    column = 0
    for degree in (degrees_list):
        # Obtain dataset and response
        y = df_copy['mpg01']
        X = df_copy[['displacement', 'acceleration']]

        # Create SVC model with polynomial kernel and fit to data
        SV_classifier = SVC(kernel='poly', gamma=gamma, degree=degree)
        SV_classifier.fit(X, y);

        print('Successfullly fit the data')

        # Make predictions
        y_pred = SV_classifier.predict(X)
        y_pred = 1*(y_pred >= 0.5)

        # Define colors for the class labels: RED for 1, GREEN for 0
        colors = ['green','red']

        axes[row, column].scatter(X['displacement'],
                                  X['acceleration'],
                                  c=y_pred,
                                  cmap=matplotlib.colors.ListedColormap(colors))
        axes[row, column].set_xlabel('displacement')
        axes[row, column].set_ylabel('acceleration')
        axes[row, column].set_title('SVC with Gam='+repr(gamma)+', Degr='+repr(degree))

        column = column + 1
    row = row + 1
```

```
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
```

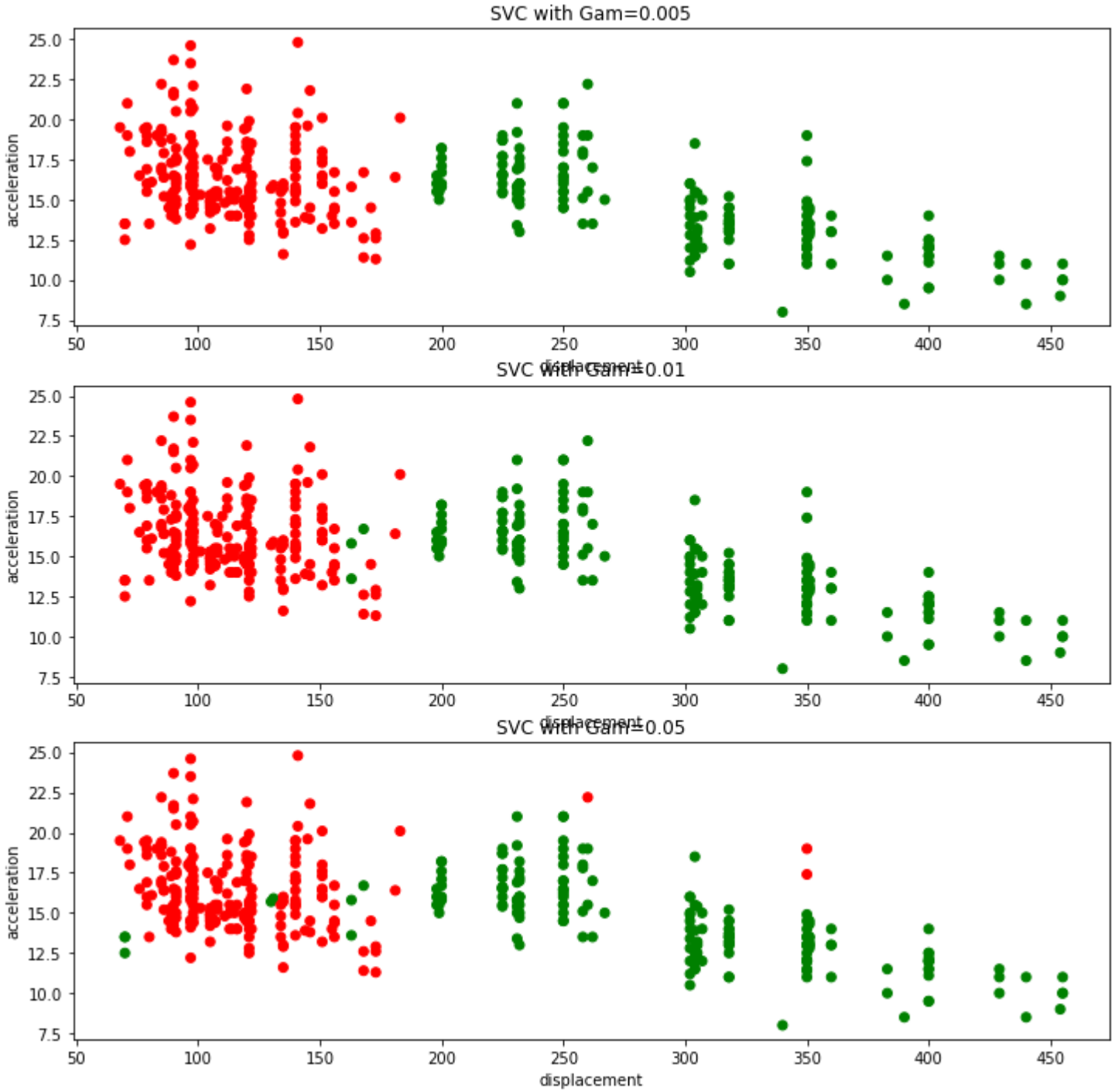

SVC with Gam=0.005, Degr=1
SVC with Gam=0.005, Degr=2
SVC with Gam=0.005, Degr=3
SVC with Gam=0.01, Degr=1
SVC with Gam=0.01, Degr=2
SVC with Gam=0.01, Degr=3
SVC with Gam=0.05, Degr=1
SVC with Gam=0.05, Degr=2
SVC with Gam=0.05, Degr=3
acceleration
displacement



Radial Basis Kernel

In [21]:
```
gamma_values_list = [0.005, 0.01, 0.05]

# Create four polar axes and access them through the returned array
fig, axes = plt.subplots(len(gamma_values_list), 1, figsize=(12,12))
i = 0
for gamma in (gamma_values_list):
    # Obtain dataset and response
    y = df_copy['mpg01']
    X = df_copy[['displacement', 'acceleration']]

    # Create SVC model with radial basis kernel and fit to data
    SV_classifier = SVC(kernel='rbf', gamma=gamma)
    SV_classifier.fit(X, y);

    print('Successfullly fit the data')

    # Make predictions
    y_pred = SV_classifier.predict(X)
    y_pred = 1*(y_pred >= 0.5)

    # Define colors for the class labels: RED for 1, GREEN for 0
    colors = ['green','red']

    axes[i].scatter(X['displacement'],
                    X['acceleration'],
                    c=y_pred,
                    cmap=matplotlib.colors.ListedColormap(colors))
    axes[i].set_xlabel('displacement')
    axes[i].set_ylabel('acceleration')
    axes[i].set_title('SVC with Gam='+repr(gamma))

    i = i + 1
```

```
Successfullly fit the data
Successfullly fit the data
Successfullly fit the data
```


SVC with Gam=0.005
SVC with Gam=0.01
SVC with Gam=0.05
acceleration
displacement
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
50
100
150
200
250
300
350
400
450

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: