Classifying Muffins and Cupcakes with SVM

Step 1: Import Packages

In [11]:

```
# Packages for analysis
import pandas as pd
import numpy as np
from sklearn import svm

# Packages for visuals
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(font_scale=1.2)

# Allows charts to appear in the notebook
%matplotlib inline

# Pickle package
import pickle
```

Step 2: Import Data

In [12]:

```
# Read in muffin and cupcake ingredient data
recipes = pd.read_csv('recipes_muffins_cupcakes.csv')
recipes
```

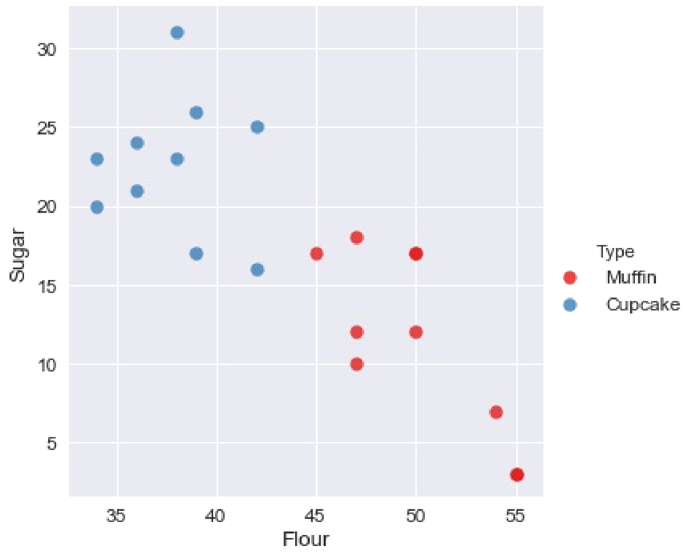
Out[12]:

	Type	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Salt
0	Muffin	55	28	3	7	5	2	0	0
1	Muffin	47	24	12	6	9	1	0	0
2	Muffin	47	23	18	6	4	1	0	0
3	Muffin	45	11	17	17	8	1	0	0
4	Muffin	50	25	12	6	5	2	1	0
5	Muffin	55	27	3	7	5	2	1	0
6	Muffin	54	27	7	5	5	2	0	0
7	Muffin	47	26	10	10	4	1	0	0
8	Muffin	50	17	17	8	6	1	0	0
9	Muffin	50	17	17	11	4	1	0	0
10	Cupcake	39	0	26	19	14	1	1	0
11	Cupcake	42	21	16	10	8	3	0	0
12	Cupcake	34	17	20	20	5	2	1	0
13	Cupcake	39	13	17	19	10	1	1	0
14	Cupcake	38	15	23	15	8	0	1	0
15	Cupcake	42	18	25	9	5	1	0	0
16	Cupcake	36	14	21	14	11	2	1	0
17	Cupcake	38	15	31	8	6	1	1	0
18	Cupcake	36	16	24	12	9	1	1	0
19	Cupcake	34	17	23	11	13	0	1	0

Step 3: Prepare the Data

In [13]:

```
# Plot two ingredients
sns.lmplot('Flour', 'Sugar', data=recipes, hue='Type',
           palette='Set1', fit_reg=False, scatter_kws={"s": 70});
```



In [14]:

```
# Specify inputs for the model
# ingredients = recipes[['Flour', 'Milk', 'Sugar', 'Butter', 'Egg', 'Baking Powder', 'Vanilla', 'Salt']].as_matrix
()
ingredients = recipes[['Flour','Sugar']].as_matrix()
type_label = np.where(recipes['Type']=='Muffin', 0, 1)

# Feature names
recipe_features = recipes.columns.values[1:].tolist()
recipe_features
```

Out[14]:

```
['Flour', 'Milk', 'Sugar', 'Butter', 'Egg', 'Baking Powder', 'Vanilla', 'Salt']
```

Step 4: Fit the Model

In [15]:

```
# Fit the SVM model
model = svm.SVC(kernel='linear')
model.fit(ingredients, type_label)
```

Out[15]:

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

Step 5: Visualize Results

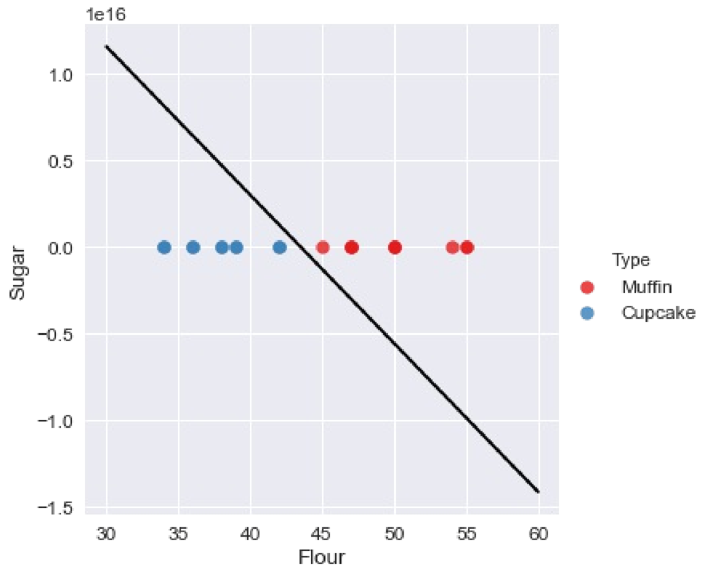
In [16]:

```
# Get the separating hyperplane
w = model.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(30, 60)
yy = a * xx - (model.intercept_[0]) / w[1]

# Plot the parallels to the separating hyperplane that pass through the support vectors
b = model.support_vectors_[0]
yy_down = a * xx + (b[1] - a * b[0])
b = model.support_vectors_[-1]
yy_up = a * xx + (b[1] - a * b[0])
```

In [18]:

```
# Plot the hyperplane
sns.lmplot('Flour', 'Sugar', data=recipes, hue='Type', palette='Set1', fit_reg=False, scatter_kws={"s": 70})
plt.plot(xx, yy, linewidth=2, color='black');
```

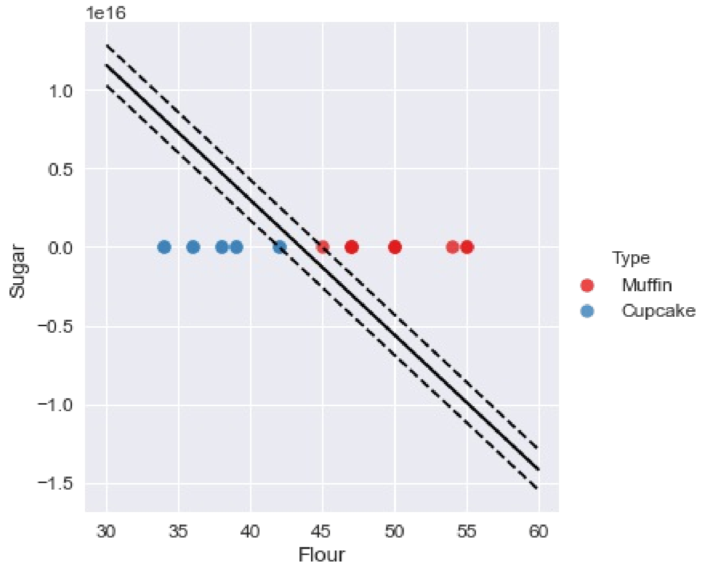


In [19]:

```
# Look at the margins and support vectors
sns.lmplot('Flour', 'Sugar', data=recipes, hue='Type', palette='Set1', fit_reg=False, scatter_kws={"s": 70})
plt.plot(xx, yy, linewidth=2, color='black')
plt.plot(xx, yy_down, 'k--')
plt.plot(xx, yy_up, 'k--')
plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
            s=80, facecolors='none');
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x10b034f60>
```



Step 6: Predict New Case

In [20]:

```
# Create a function to guess when a recipe is a muffin or a cupcake
def muffin_or_cupcake(flour, sugar):
    if(model.predict([[flour, sugar]]))==0:
        print('You\'re looking at a muffin recipe!')
    else:
        print('You\'re looking at a cupcake recipe!')
```

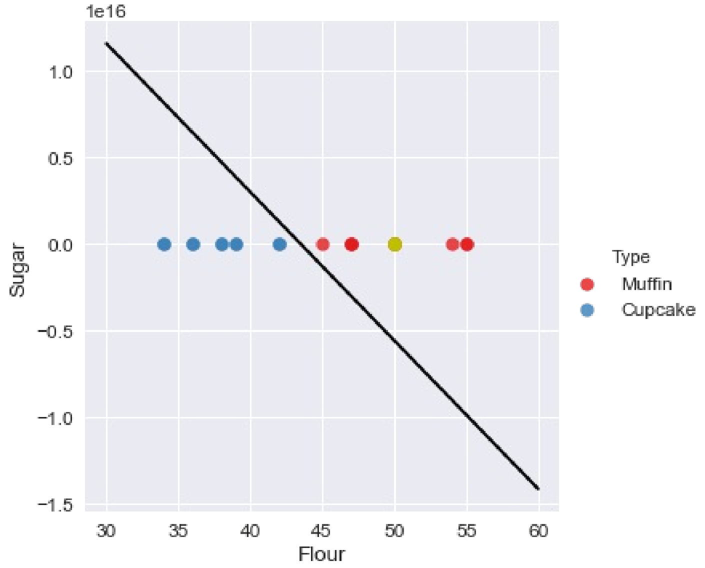
In [21]:

```
# Predict if 50 parts flour and 20 parts sugar
muffin_or_cupcake(50, 20)
```

```
You're looking at a muffin recipe!
```

In [23]:

```
# Plot the point to visually see where the point lies
sns.lmplot('Flour', 'Sugar', data=recipes, hue='Type', palette='Set1', fit_reg=False, scatter_kws={"s": 70})
plt.plot(xx, yy, linewidth=2, color='black')
plt.plot(50, 20, 'yo', markersize='9');
```



In [24]:

```
# Predict if 40 parts flour and 20 parts sugar
muffin_or_cupcake(40,20)
```

```
You're looking at a cupcake recipe!
```

In [25]:

```
muffin_cupcake_dict = {'muffin_cupcake_model': model, 'muffin_cupcake_features': ['Flour','Sugar'], 'all_features'
: recipe_features}
```

In [26]:

```
muffin_cupcake_dict
```

Out[26]:

```
{'all_features': ['Flour',
  'Milk',
  'Sugar',
  'Butter',
  'Egg',
  'Baking Powder',
  'Vanilla',
  'Salt'],
 'muffin_cupcake_features': ['Flour', 'Sugar'],
 'muffin_cupcake_model': SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
   decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
   max_iter=-1, probability=False, random_state=None, shrinking=True,
   tol=0.001, verbose=False)}
```

In [210]:

```
# Pickle
pickle.dump(muffin_cupcake_dict, open("muffin_cupcake_dict.p", "wb"))
```

In [211]:

```
# S = String
pickle.dumps(muffin_cupcake_dict)
```

Out[211]:


```
"(dp0\nS'muffin_cupcake_features'\np1\n(lp2\nS'Flour'\np3\naS'Sugar'\np4\nasS'muffin_cupcake_model'\np5\ncc
opy_reg\n_reconstructor\np6\n(csklearn.svm.classes\nSVC\np7\nc__builtin__\nobject\np8\nNtp9\nRp10\n(dp11\nS
'_impl'\np12\nS'c_svc'\np13\nsS'kernel'\np14\nS'linear'\np15\nsS'verbose'\np16\nI00\nsS'probability'\np17\n
I00\nsS'classes_'\np18\ncnumpy.core.multiarray\n_reconstruct\np19\n(cnumpy\nndarray\np20\n(I0\ntp21\nS'b'\n
p22\ntp23\nRp24\n(I1\n(I2\ntp25\ncnumpy\ndtype\np26\n(S'i8'\np27\nI0\nI1\ntp28\nRp29\n(I3\nS'<'\np30\nNNNI-
1\nI-1\nI0\ntp31\nbI00\nS'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x01\\x00\\x00\\x00\\x00\\x00\\x00\\x00'
\np32\ntp33\nbsS'support_'\np34\ng19\n(g20\n(I0\ntp35\ng22\ntp36\nRp37\n(I1\n(I3\ntp38\ng26\n(S'i4'\np39\nI
0\nI1\ntp40\nRp41\n(I3\nS'<'\np42\nNNNI-1\nI-1\nI0\ntp43\nbI00\nS'\\x03\\x00\\x00\\x00\\x0b\\x00\\x00\\x00\
\x0f\\x00\\x00\\x00'\np44\ntp45\nbsS'dual_coef_'\np46\ng19\n(g20\n(I0\ntp47\ng22\ntp48\nRp49\n(I1\n(I1\nI3\
ntp50\ng26\n(S'f8'\np51\nI0\nI1\ntp52\nRp53\n(I3\nS'<'\np54\nNNNI-1\nI-1\nI0\ntp55\nbI00\nS'\\x19;\\x16\\x8
1\\xfdo\\xcc\\xbf5\\xdf\\xda9\\x1aG\\xc9? \\xdf\\xda9\\x1aG\\x99?'\np56\ntp57\nbsS'shrinking'\np58\nI01\nsS
'class_weight'\np59\nNsS'_gamma'\np60\nF0.5\nsS'probA_'\np61\ng19\n(g20\n(I0\ntp62\ng22\ntp63\nRp64\n(I1\n(
I0\ntp65\ng53\nI00\nS''\np66\ntp67\nbsS'_sparse'\np68\nI00\nsS'class_weight_'\np69\ng19\n(g20\n(I0\ntp70\ng
22\ntp71\nRp72\n(I1\n(I2\ntp73\ng53\nI00\nS'\\x00\\x00\\x00\\x00\\x00\\x00\\xf0?\\x00\\x00\\x00\\x00\\x00\\
x00\\xf0?'\np74\ntp75\nbsS'random_state'\np76\nNsS'_sklearn_version'\np77\nS'0.18.1'\np78\nsS'tol'\np79\nF0
.001\nsS'coef0'\np80\nF0.0\nsS'nu'\np81\nF0.0\nsS'n_support_'\np82\ng19\n(g20\n(I0\ntp83\ng22\ntp84\nRp85\n
(I1\n(I2\ntp86\ng41\nI00\nS'\\x01\\x00\\x00\\x00\\x02\\x00\\x00\\x00'\np87\ntp88\nbsS'shape_fit_'\np89\n(I2
0\nI2\ntp90\nsS'C'\np91\nF1.0\nsS'support_vectors_'\np92\ng19\n(g20\n(I0\ntp93\ng22\ntp94\nRp95\n(I1\n(I3\n
I2\ntp96\ng53\nI00\nS'\\x00\\x00\\x00\\x00\\x00\\x80F@\\x00\\x00\\x00\\x00\\x00\\x001@\\x00\\x00\\x00\\x00\
\x00\\x00E@\\x00\\x00\\x00\\x00\\x00\\x000@\\x00\\x00\\x00\\x00\\x00\\x00E@\\x00\\x00\\x00\\x00\\x00\\x009@
'\np97\ntp98\nbsS'_dual_coef_'\np99\ng19\n(g20\n(I0\ntp100\ng22\ntp101\nRp102\n(I1\n(I1\nI3\ntp103\ng53\nI0
0\nS'\\x19;\\x16\\x81\\xfdo\\xcc?5\\xdf\\xda9\\x1aG\\xc9\\xbf \\xdf\\xda9\\x1aG\\x99\\xbf'\np104\ntp105\nbs
S'degree'\np106\nI3\nsS'epsilon'\np107\nF0.0\nsS'max_iter'\np108\nI-1\nsS'decision_function_shape'\np109\nN
sS'fit_status_'\np110\nI0\nsS'_intercept_'\np111\ng19\n(g20\n(I0\ntp112\ng22\ntp113\nRp114\n(I1\n(I1\ntp115
\ng53\nI00\nS'\\xe9\\xbcm\\xd12\\xfe<\\xc0'\np116\ntp117\nbsS'intercept_'\np118\ng19\n(g20\n(I0\ntp119\ng22
\ntp120\nRp121\n(I1\n(I1\ntp122\ng53\nI00\nS'\\xe9\\xbcm\\xd12\\xfe<@'\np123\ntp124\nbsS'probB_'\np125\ng19
\n(g20\n(I0\ntp126\ng22\ntp127\nRp128\n(I1\n(I0\ntp129\ng53\nI00\ng66\ntp130\nbsS'cache_size'\np131\nI200\n
sS'gamma'\np132\nS'auto'\np133\nsbsS'all_features'\np134\n(lp135\nS'Flour'\np136\naS'Milk'\np137\naS'Sugar'
\np138\naS'Butter'\np139\naS'Egg'\np140\naS'Baking Powder'\np141\naS'Vanilla'\np142\naS'Salt'\np143\nas."
```