Support Vector Machines to predict Wine color

Support Vector Machine is a supervised machine learning algorithm. We are using a dataset which cointains chemical properties (volatile_acidity , total_sulphur_dioxide etc) to determine wine color

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

In [3]:
```
data = pd.read_csv("data/Wine_Quality_Data.csv")
```

In [4]:
```
print(data.shape)
```

```
(6497, 13)
```

In [5]:
```
#Print no of integers, floats and strings
data.dtypes.value_counts()
```

Out[5]:
```
float64    11
object      1
int64       1
dtype: int64
```

In [6]:
```
data.head()
```

Out[6]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur
0	7.4	0.70	0.00	1.9	0.076	11.0	
1	7.8	0.88	0.00	2.6	0.098	25.0	
2	7.8	0.76	0.04	2.3	0.092	15.0	
3	11.2	0.28	0.56	1.9	0.075	17.0	
4	7.4	0.70	0.00	1.9	0.076	11.0	

Preprocessing Steps

1. Select Features .

1. Select Features and Normalize data.

In [10]:
```
#Changing Target Variable y to int (1- red and 0 - yellow)
y = (data['color'] == 'red').astype(int)
y[1:10]
```

Out[10]:
```
1    1
2    1
3    1
4    1
5    1
6    1
7    1
8    1
9    1
Name: color, dtype: int64
```

In [12]:
```
# extracting all fields except "color"
fields = list(data.columns[:-1])
#checking which features are more corealted to x
correlations = data[fields].corrwith(y)
correlations.sort_values(inplace=True)
correlations
```

Out[12]:
```
total_sulfur_dioxide   -0.700357
free_sulfur_dioxide    -0.471644
residual_sugar         -0.348821
citric_acid            -0.187397
quality                -0.119323
alcohol                -0.032970
pH                      0.329129
density                 0.390645
fixed_acidity           0.486740
sulphates               0.487218
chlorides               0.512678
volatile_acidity        0.653036
dtype: float64
```

Pairplot allow us to identify how each features are related to each ohter and also to target variable. Distribution of each feature is shown diagonally.

In [15]:
```
import seaborn as sns
sns.pairplot(data, hue="color")
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x103552ed0>

LINEAR SVC CLASSIFIER

In [17]:
```
# Extracting only two features which are highly corealted
#Scaling is important here - Distance is important and We can't have a feat
from sklearn.preprocessing import MinMaxScaler

fields = correlations.map(abs).sort_values().iloc[-2:].index
print(fields)
X = data[fields]
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
X = pd.DataFrame(X, columns=['%s_scaled' % fld for fld in fields])
print(X.columns)
```

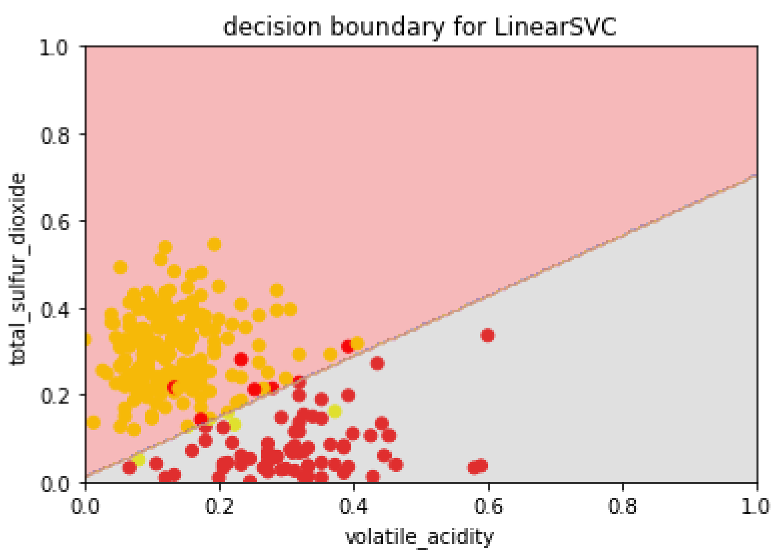
```
Index(['volatile_acidity', 'total_sulfur_dioxide'], dtype='object')
Index(['volatile_acidity_scaled', 'total_sulfur_dioxide_scaled'], dtype
='object')
```

In [18]:
```
from sklearn.svm import LinearSVC

LSVC = LinearSVC()
LSVC.fit(X, y)
```

Out[18]:
```
LinearSVC(C=1.0, 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)
```

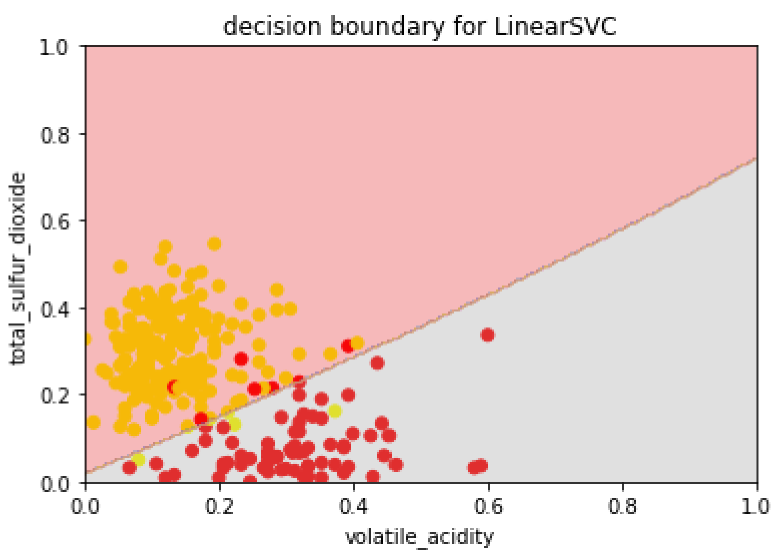
In [27]:
```
X_color = X.sample(300, random_state=45)
y_color = y.loc[X_color.index]
y_color = y_color.map(lambda r: 'red' if r == 1 else 'yellow')
ax = plt.axes()
ax.scatter(
    X_color.iloc[:, 0], X_color.iloc[:, 1],
    color=y_color, alpha=1)
# -----------
x_axis, y_axis = np.arange(0, 1.005, .005), np.arange(0, 1.005, .005)
xx, yy = np.meshgrid(x_axis, y_axis)
xx_ravel = xx.ravel()
yy_ravel = yy.ravel()
X_grid = pd.DataFrame([xx_ravel, yy_ravel]).T
y_grid_predictions = LSVC.predict(X_grid)
y_grid_predictions = y_grid_predictions.reshape(xx.shape)
ax.contourf(xx, yy, y_grid_predictions, cmap=plt.cm.Set1, alpha=.3)
# -----------
ax.set(
    xlabel=fields[0],
    ylabel=fields[1],
    xlim=[0, 1],
    ylim=[0, 1],
    title='decision boundary for LinearSVC');
```



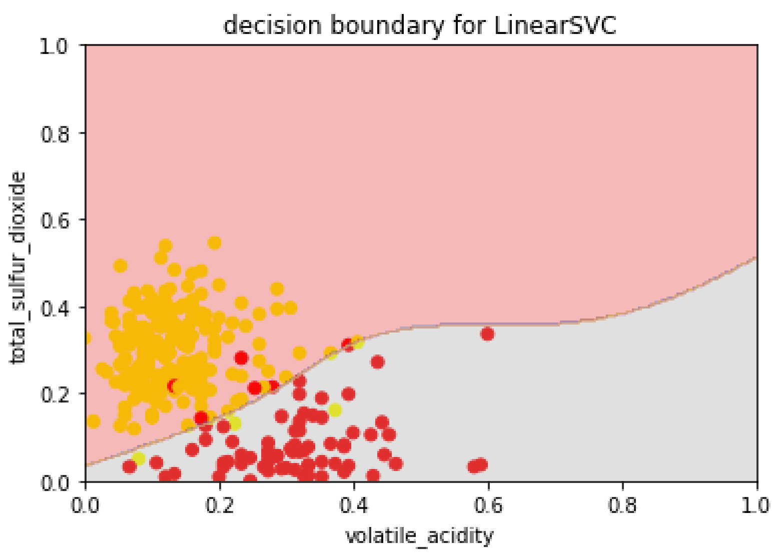
Gaussian Kernel

Tryring to fit a gaussian Kernel with two gamma parameters(.5 and 10)

In [33]:
```
from sklearn.svm import SVC
gamma = .5
SVC_Gaussian = SVC(kernel = 'rbf', gamma =gamma)
SVC_Gaussian.fit(X, y)
X_color = X.sample(300, random_state=45)
y_color = y.loc[X_color.index]
y_color = y_color.map(lambda r: 'red' if r == 1 else 'yellow')
ax = plt.axes()
ax.scatter(
    X_color.iloc[:, 0], X_color.iloc[:, 1],
    color=y_color, alpha=1)
# -----------
x_axis, y_axis = np.arange(0, 1.005, .005), np.arange(0, 1.005, .005)
xx, yy = np.meshgrid(x_axis, y_axis)
xx_ravel = xx.ravel()
yy_ravel = yy.ravel()
X_grid = pd.DataFrame([xx_ravel, yy_ravel]).T
y_grid_predictions = SVC_Gaussian.predict(X_grid)
y_grid_predictions = y_grid_predictions.reshape(xx.shape)
ax.contourf(xx, yy, y_grid_predictions, cmap=plt.cm.Set1, alpha=.3)
# -----------
ax.set(
    xlabel=fields[0],
    ylabel=fields[1],
    xlim=[0, 1],
    ylim=[0, 1],
    title='decision boundary for LinearSVC');
```



In [34]:
```
from sklearn.svm import SVC
gamma = 10
SVC_Gaussian = SVC(kernel = 'rbf', gamma =gamma)
SVC_Gaussian.fit(X, y)
X_color = X.sample(300, random_state=45)
y_color = y.loc[X_color.index]
y_color = y_color.map(lambda r: 'red' if r == 1 else 'yellow')
ax = plt.axes()
ax.scatter(
    X_color.iloc[:, 0], X_color.iloc[:, 1],
    color=y_color, alpha=1)
# -----------
x_axis, y_axis = np.arange(0, 1.005, .005), np.arange(0, 1.005, .005)
xx, yy = np.meshgrid(x_axis, y_axis)
xx_ravel = xx.ravel()
yy_ravel = yy.ravel()
X_grid = pd.DataFrame([xx_ravel, yy_ravel]).T
y_grid_predictions = SVC_Gaussian.predict(X_grid)
y_grid_predictions = y_grid_predictions.reshape(xx.shape)
ax.contourf(xx, yy, y_grid_predictions, cmap=plt.cm.Set1, alpha=.3)
# -----------
ax.set(
    xlabel=fields[0],
    ylabel=fields[1],
    xlim=[0, 1],
    ylim=[0, 1],
    title='decision boundary for LinearSVC');
```



SUMMARY

1. Plotted a linear decision boundary of a LinearSVC classifier on dataset .
2. Tried to fit Gaussian kernel by changing values of gamma .
3. Gamma is less (.5) high regularization and less complex classifier
4. Gamma is more (10) less regularization and more complex model which tries to overfit.

In [ ]: