In [38]:

```
##Import the relevant libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
```

In [39]:

```
### Load the data
data = pd.read_csv("D:\\Projects\\Market segmentation - Clustering\\DATA.csv")
```

In [40]:

```
### Check the data
data
```

Out[40]:

	Satisfaction	Loyalty
0	4	-1.33
1	6	-0.28
2	5	-0.99
3	7	-0.29
4	4	1.06
5	1	-1.66
6	10	-0.97
7	8	-0.32
8	8	1.02
9	8	0.68
10	10	-0.34
11	5	0.39
12	5	-1.69
13	2	0.67
14	7	0.27
15	9	1.36
16	8	1.38
17	7	1.36
18	7	-0.34
19	9	0.67
20	10	1.18
21	3	-1.69
22	4	1.04
23	3	-0.96
24	6	1.03
25	9	-0.99

	Satisfaction	Loyalty
26	10	0.37
27	9	0.03
28	3	-1.36
29	5	0.73

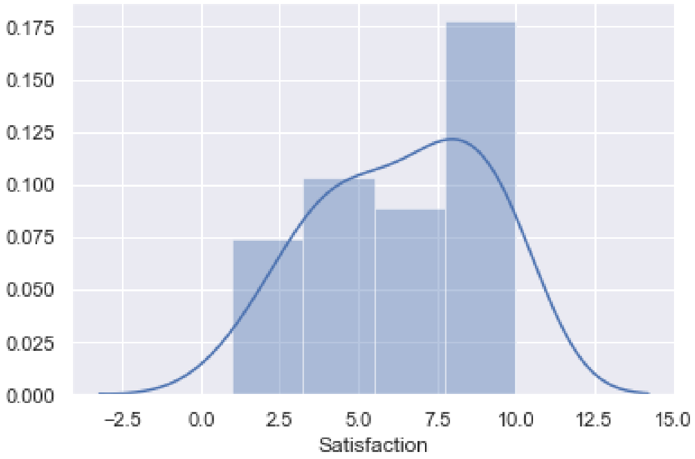
In [43]:

```
sns.distplot(data['Satisfaction'])
```

```
C:\Users\Administrator\Anaconda2\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimen
sional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index,
`arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

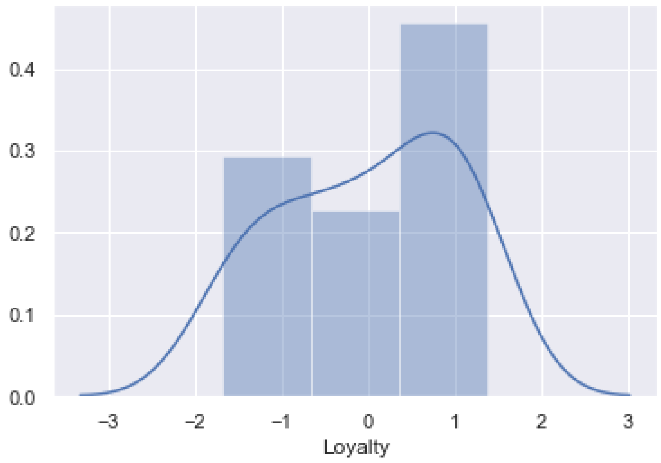
Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xdac4438>
```



```
In [44]: sns.distplot(data['Loyalty'])
```

```
Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0xdd4b128>
```



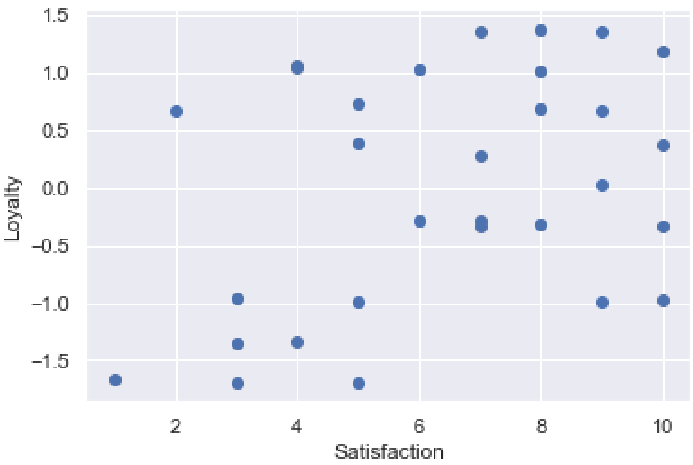
In [45]:

```


## Plot the data
# We are creating a scatter plot of the two variables
plt.scatter(data['Satisfaction'],data['Loyalty'])
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
```

Out[45]:

```
Text(0,0.5,'Loyalty')
```



In [46]:

```


#select the features
# Select both features by creating a copy of the data variable
x = data.copy()
x
```

Out[46]:

```
    Satisfaction  Loyalty
 0             4    -1.33
 1             6    -0.28
 2             5    -0.99
 3             7    -0.29
 4             4     1.06
 5             1    -1.66
 6            10    -0.97
 7             8    -0.32
 8             8     1.02
 9             8     0.68
10            10    -0.34
11             5     0.39
12             5    -1.69
13             2     0.67
14             7     0.27
15             9     1.36
16             8     1.38
17             7     1.36
18             7    -0.34
19             9     0.67
20            10     1.18
21             3    -1.69
22             4     1.04
23             3    -0.96
```

	Satisfaction	Loyalty
24	6	1.03
25	9	-0.99
26	10	0.37
27	9	0.03
28	3	-1.36
29	5	0.73

In [47]:

```
##Clustering results
# Create a copy of the input data
clusters = x.copy()
# Take note of the predicted clusters
clusters['cluster_pred']=kmeans.fit_predict(x)
clusters
```

Out[47]:

	Satisfaction	Loyalty	cluster_pred
0	4	-1.33	1
1	6	-0.28	1
2	5	-0.99	1
3	7	-0.29	0
4	4	1.06	1
5	1	-1.66	1
6	10	-0.97	0
7	8	-0.32	0
8	8	1.02	0
9	8	0.68	0
10	10	-0.34	0

In [28]:

```

# Create an object - kmeans for 2 clusters
kmeans = KMeans(2)
kmeans.fit(x)
```

Out[28]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

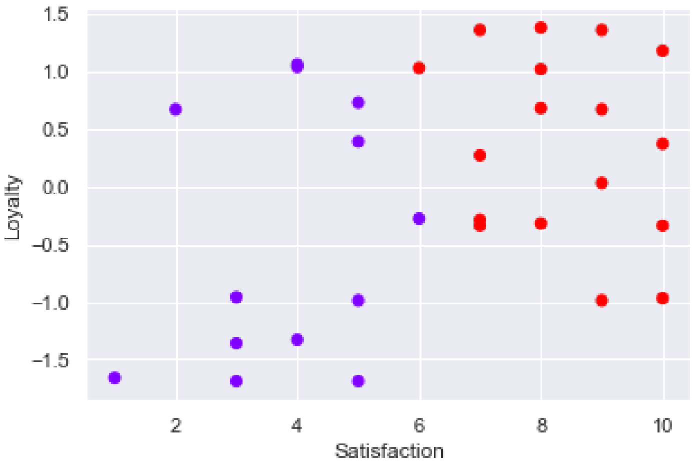
In [42]:

```
# Plot the data using the longitude and the latitude
plt.scatter(clusters['Satisfaction'],clusters['Loyalty'],c=clusters['cluster_pred'],cmap='rainbow')
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')

## most probably algorithum consider satisfaction only as a feature. boundry line is 6.
##Because we did not standardize the variable the satisfaction values are much higher than those of loyalty
```

Out[42]:

```
Text(0,0.5,'Loyalty')
```



In [32]:

```
## To solve above problem we will standardize variables.
##Standardize the variables
from sklearn import preprocessing
x_scaled = preprocessing.scale(x)
x_scaled
```

```
C:\Users\Administrator\Anaconda2\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: Data with input dtype int64, floa
t64 were all converted to float64 by the scale function.
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[32]:

```
array([[-0.93138063, -1.3318111 ],
       [-0.15523011, -0.28117124],
       [-0.54330537, -0.99160391],
       [ 0.23284516, -0.29117733],
       [-0.93138063,  1.05964534],
       [-2.09560642, -1.6620122 ],
       [ 1.39707095, -0.97159172],
       [ 0.62092042, -0.32119561],
       [ 0.62092042,  1.01962097],
       [ 0.62092042,  0.67941378],
       [ 1.39707095, -0.3412078 ],
       [-0.54330537,  0.38923705],
       [-0.54330537, -1.69203048],
       [-1.70753116,  0.66940768],
       [ 0.23284516,  0.26916393],
       [ 1.00899568,  1.35982816],
```

In [48]:

```
##Elbow method
# Createa an empty list
wcss =[]

for i in range(1,10):
    kmeans = KMeans(i)
    kmeans.fit(x_scaled)
    wcss.append(kmeans.inertia_)


wcss
```

Out[48]:

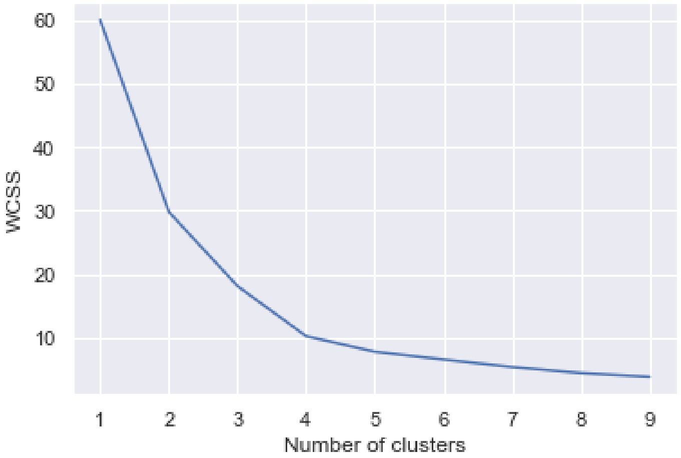
```
[60.0,
 29.818973034723147,
 18.129659446063226,
 10.247181805928422,
 7.792695153937187,
 6.569489487091783,
 5.398758288946922,
 4.4492366515918995,
 3.860881437312899]
```

In [49]:

```
# Plot the number of clusters vs WCSS
plt.plot(range(1,10),wcss)
# Name your axes
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
```

Out[49]:

```
Text(0,0.5,'WCSS')
```



In [50]:

```
## Explore clustering solutions and select the number of clusters
kmeans_new = KMeans(4)
kmeans_new.fit(x_scaled)
clusters_new = x.copy()
clusters_new['cluster_pred'] = kmeans_new.fit_predict(x_scaled)
```

In [51]:

```python
# Check if everything seems right
clusters_new
```

Out[51]:

	Satisfaction	Loyalty	cluster_pred
0	4	-1.33	0
1	6	-0.28	2
2	5	-0.99	0
3	7	-0.29	2
4	4	1.06	1
5	1	-1.66	0
6	10	-0.97	2
7	8	-0.32	2
8	8	1.02	3
9	8	0.68	3
10	10	-0.34	2
11	5	0.39	1
12	5	-1.69	0
13	2	0.67	1
14	7	0.27	2
15	9	1.36	3
16	8	1.38	3
17	7	1.36	3
18	7	-0.34	2
19	9	0.67	3
20	10	1.18	3
21	3	-1.69	0
22	4	1.04	1
23	3	-0.96	0
24	6	1.03	1
25	9	-0.99	2

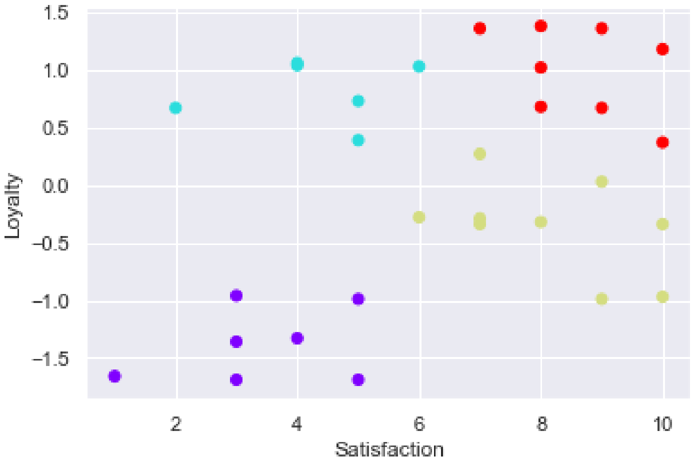
	Satisfaction	Loyalty	cluster_pred
26	10	0.37	3
27	9	0.03	2
28	3	-1.36	0
29	5	0.73	1

In [52]:

```
# Plot
plt.scatter(clusters_new['Satisfaction'],clusters_new['Loyalty'],c=clusters_new['cluster_pred'],cmap='rainbow')
plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
```

Out[52]:

```
Text(0,0.5,'Loyalty')
```



In [ ]:

```

```