In [1]:
```
from matplotlib.colors import ListedColormap
from sklearn import model_selection, datasets, metrics, neighbors

%matplotlib inline

import numpy as np
```

In [12]:
```
%pylab inline
```

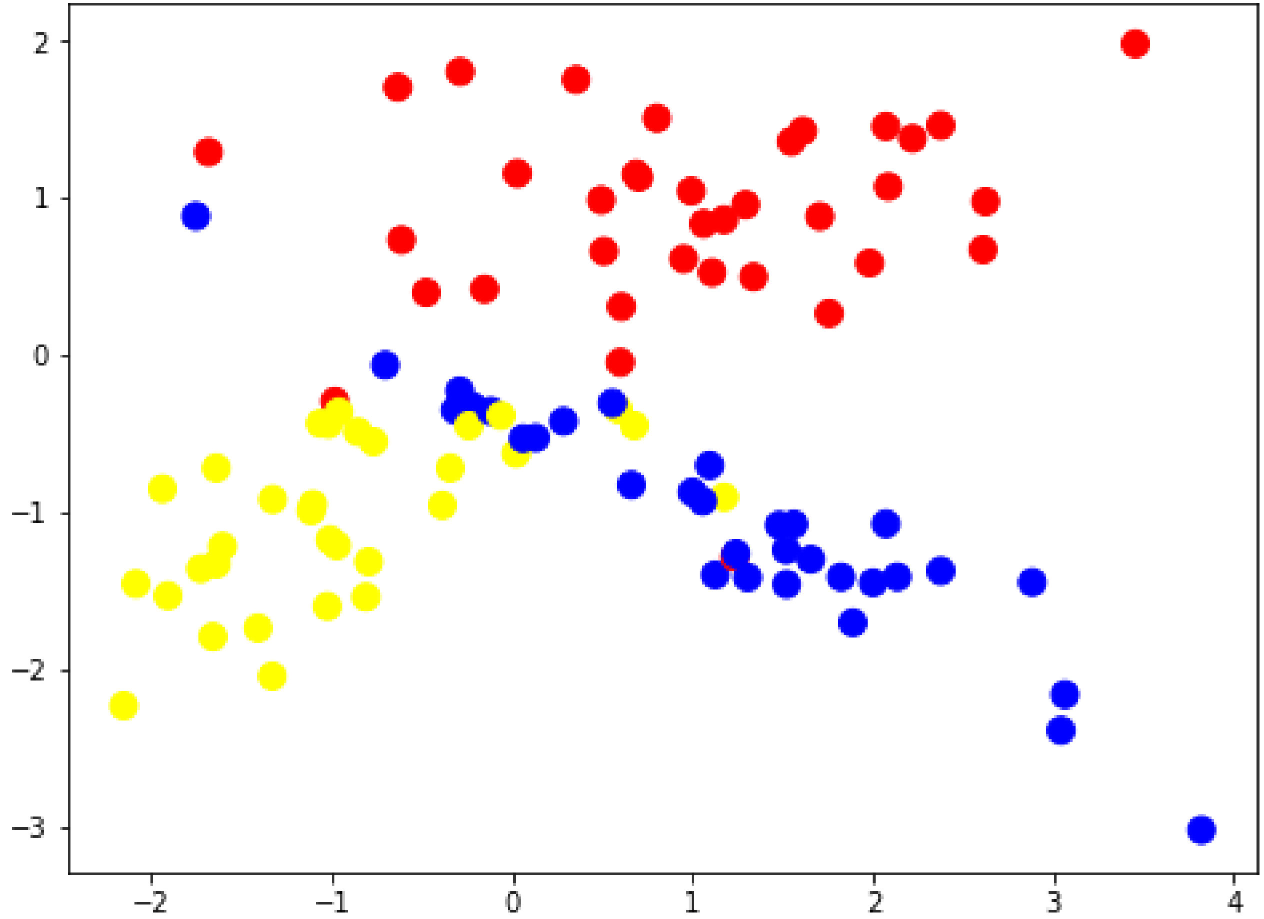
```
Populating the interactive namespace from numpy and matplotlib
```

In [13]:
```
classification_problem = datasets.make_classification(n_samples=100, n_featur
es =2, n_informative = 2,
                                                      n_classes = 3, n_redund
ant=0,
                                                      n_clusters_per_class=1,
random_state=3)
```

In [14]:
```
colors = ListedColormap(['red', 'blue', 'yellow'])
light_colors = ListedColormap(['lightcoral', 'lightblue', 'lightyellow'])
```

In [15]:
```
pylab.figure(figsize=(8,6))
pylab.scatter(map(lambda x: x[0], classification_problem[0]), map(lambda x: x
[1], classification_problem[0]),
              c=classification_problem[1], cmap=colors, s=100)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x7f7fa5233190>
```



In [16]:
```
train_data, test_data, train_labels, test_labels = cross_validation.train_tes
t_split(classification_problem[0],

classification_problem[1],

test_size = 0.3,

random_state = 1)
```

In [17]:
```
clf = neighbors.KNeighborsClassifier()
clf.fit(train_data, train_labels)
```

Out[17]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

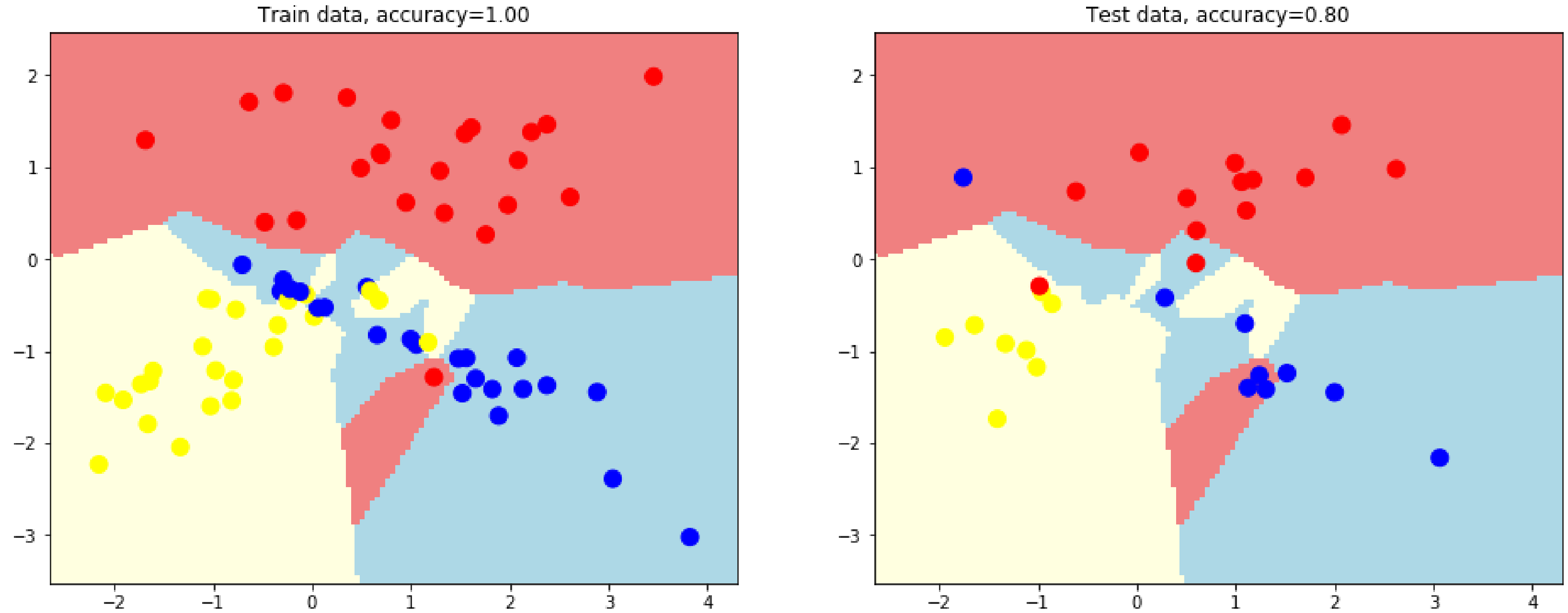
In [18]:
```
def get_meshgrid(data, step=.05, border=.5,):
    x_min, x_max = data[:, 0].min() - border, data[:, 0].max() + border
    y_min, y_max = data[:, 1].min() - border, data[:, 1].max() + border
    return np.meshgrid(np.arange(x_min, x_max, step), np.arange(y_min, y_max,
step))
```

In [19]:

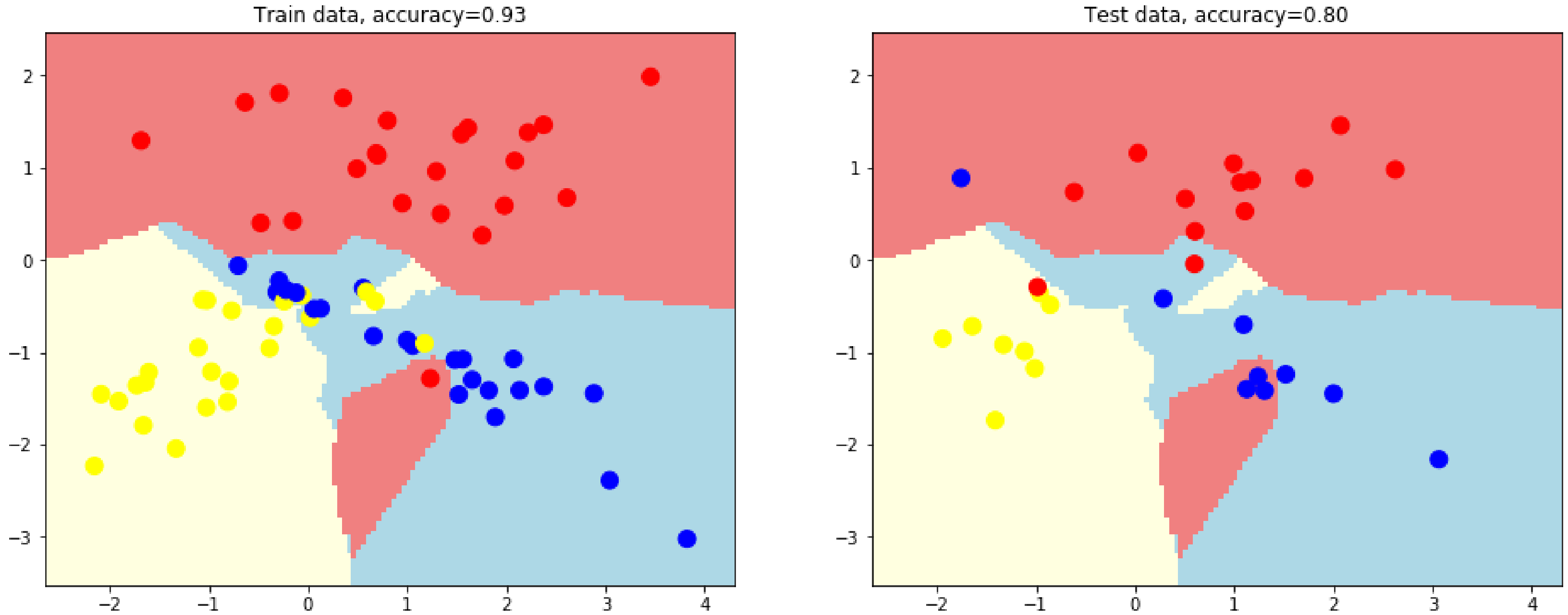
```
def plot_decision_surface(estimator, train_data, train_labels, test_data, tes
t_labels,
                          colors = colors, light_colors = light_colors):
    #fit model
    estimator.fit(train_data, train_labels)

    #set figure size
    pyplot.figure(figsize = (16, 6))

    #plot decision surface on the train data
    pyplot.subplot(1,2,1)
    xx, yy = get_meshgrid(train_data)
    mesh_predictions = np.array(estimator.predict(np.c_[xx.ravel(), yy.ravel(
)])).reshape(xx.shape)
    pyplot.pcolormesh(xx, yy, mesh_predictions, cmap = light_colors)
    pyplot.scatter(train_data[:, 0], train_data[:, 1], c = train_labels, s =
100, cmap = colors)
    pyplot.title('Train data, accuracy={:.2f}'.format(metrics.accuracy_score(
train_labels, estimator.predict(train_data))))

    #plot decision surface on the test data
    pyplot.subplot(1,2,2)
    pyplot.pcolormesh(xx, yy, mesh_predictions, cmap = light_colors)
    pyplot.scatter(test_data[:, 0], test_data[:, 1], c = test_labels, s = 100
, cmap = colors)
    pyplot.title('Test data, accuracy={:.2f}'.format(metrics.accuracy_score(t
est_labels, estimator.predict(test_data))))
```

In [20]:

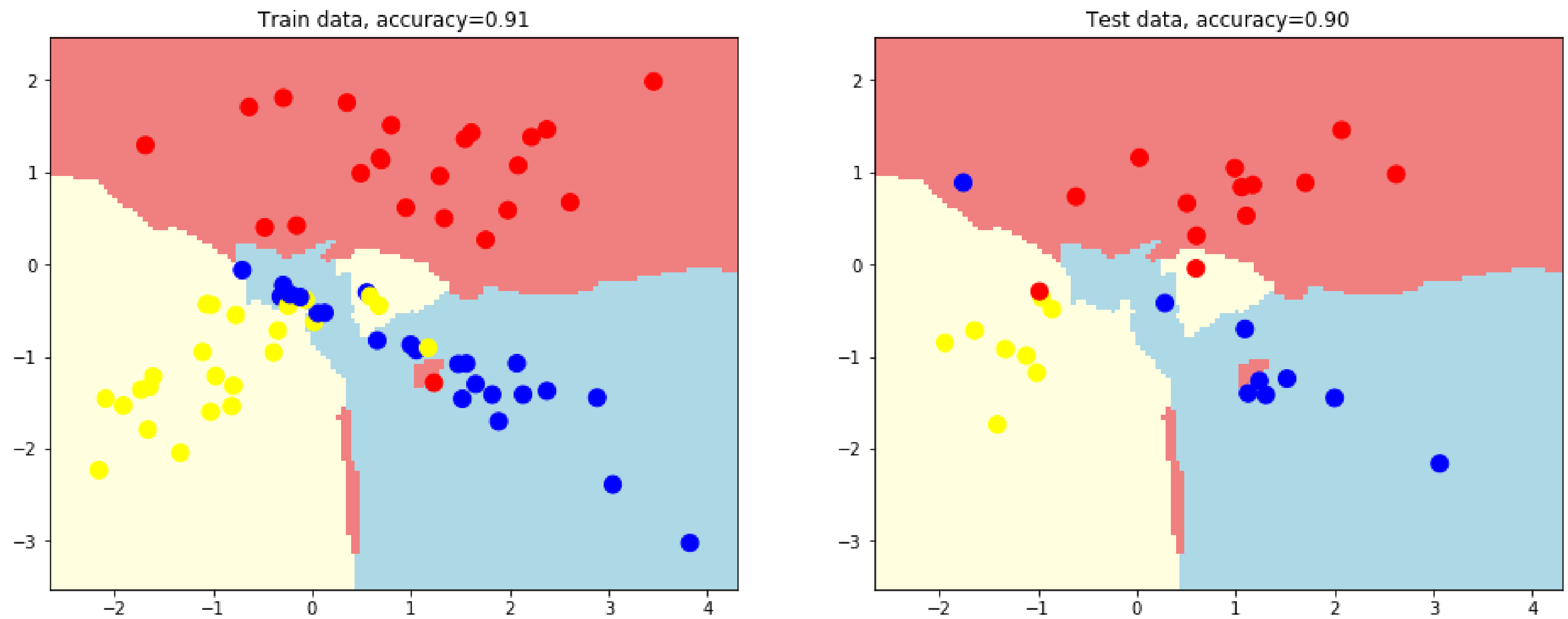
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=1)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```



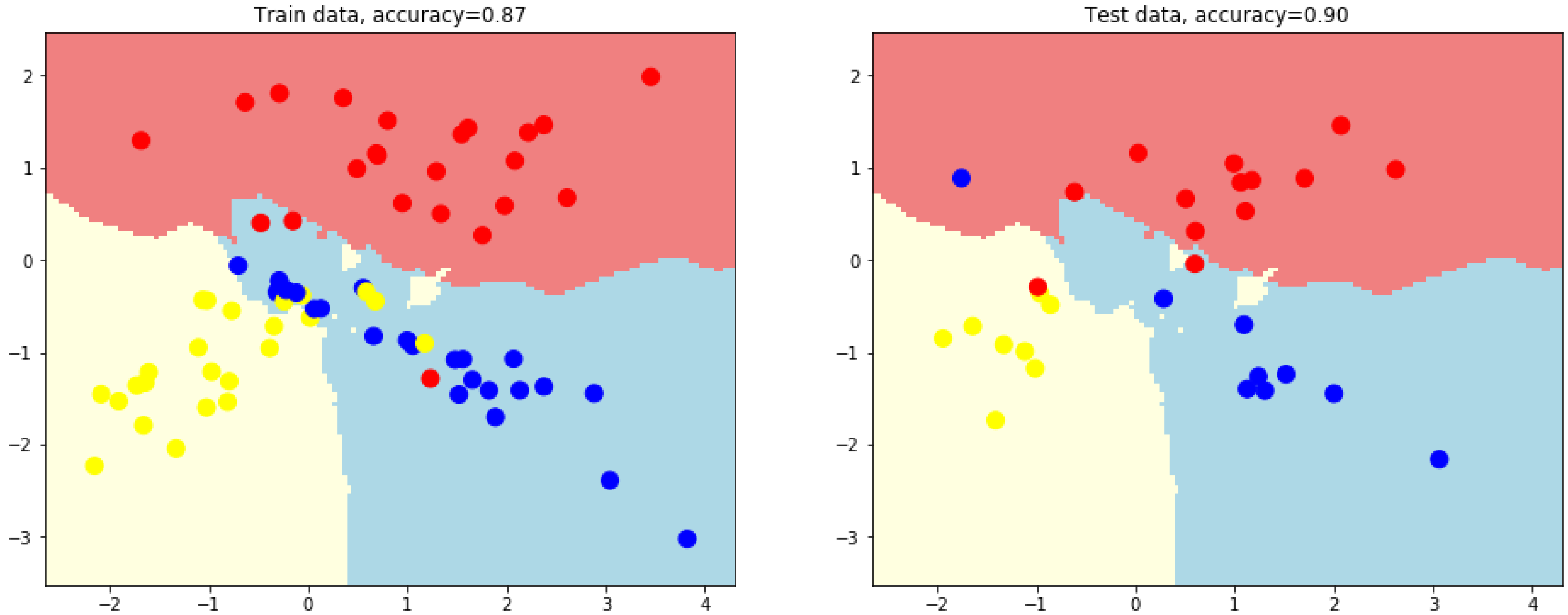
In [21]:
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=2)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```



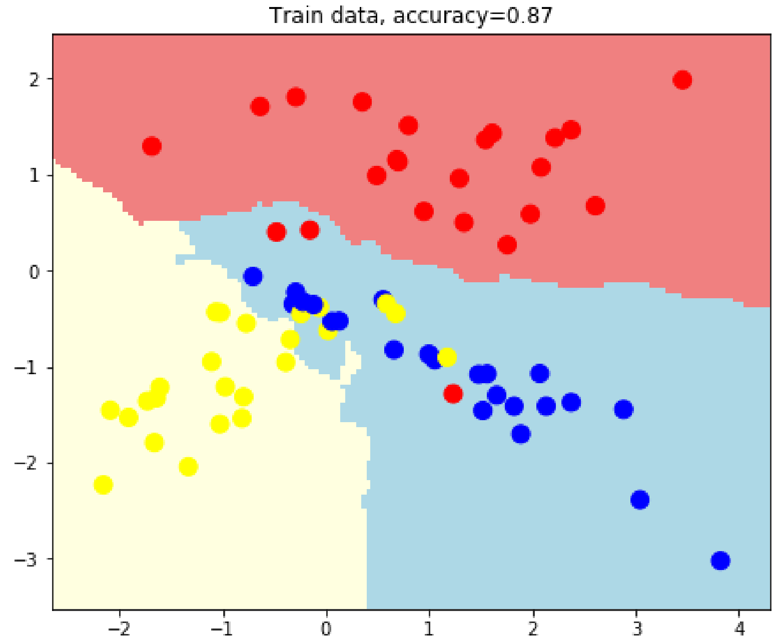
In [22]:
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=3)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```

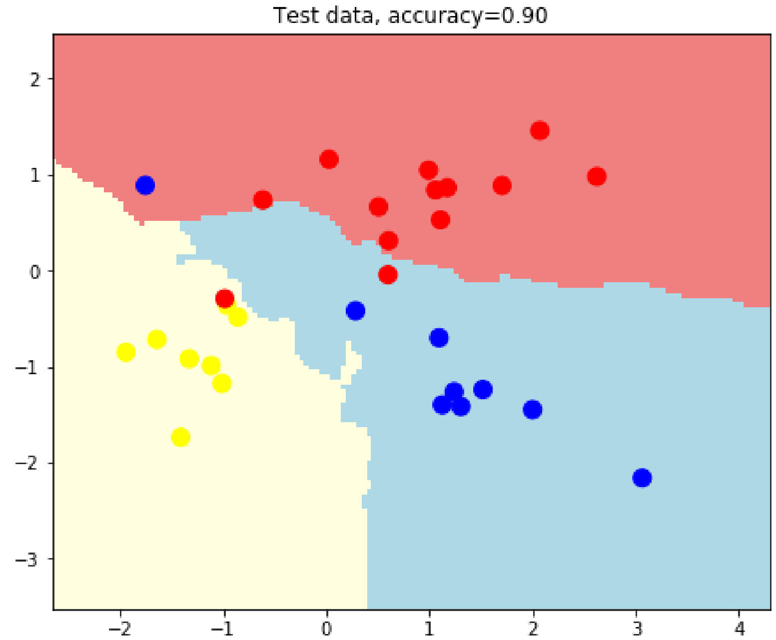


In [23]:
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=5)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```

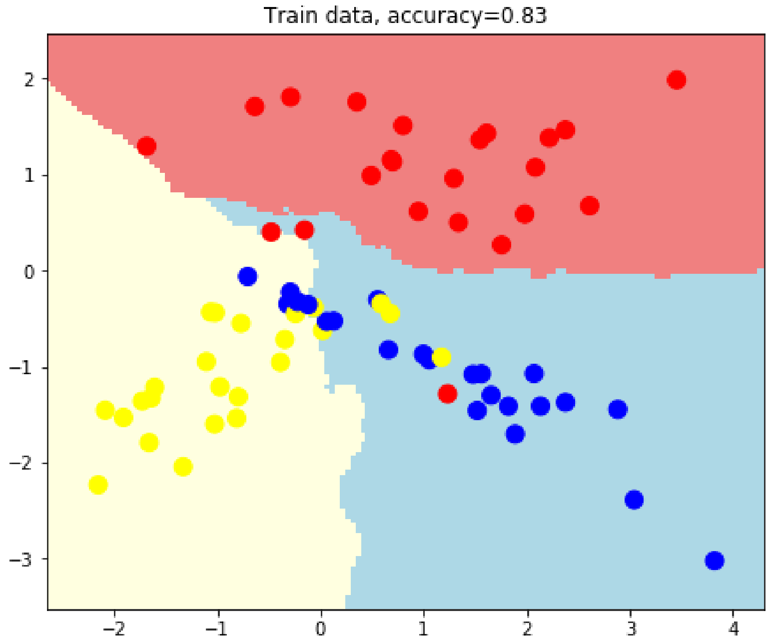


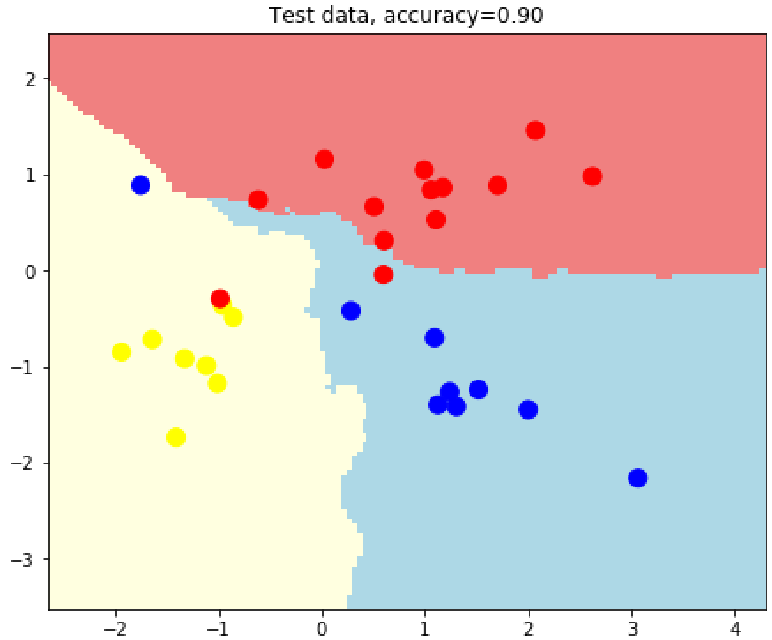
In [24]:
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=10)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```



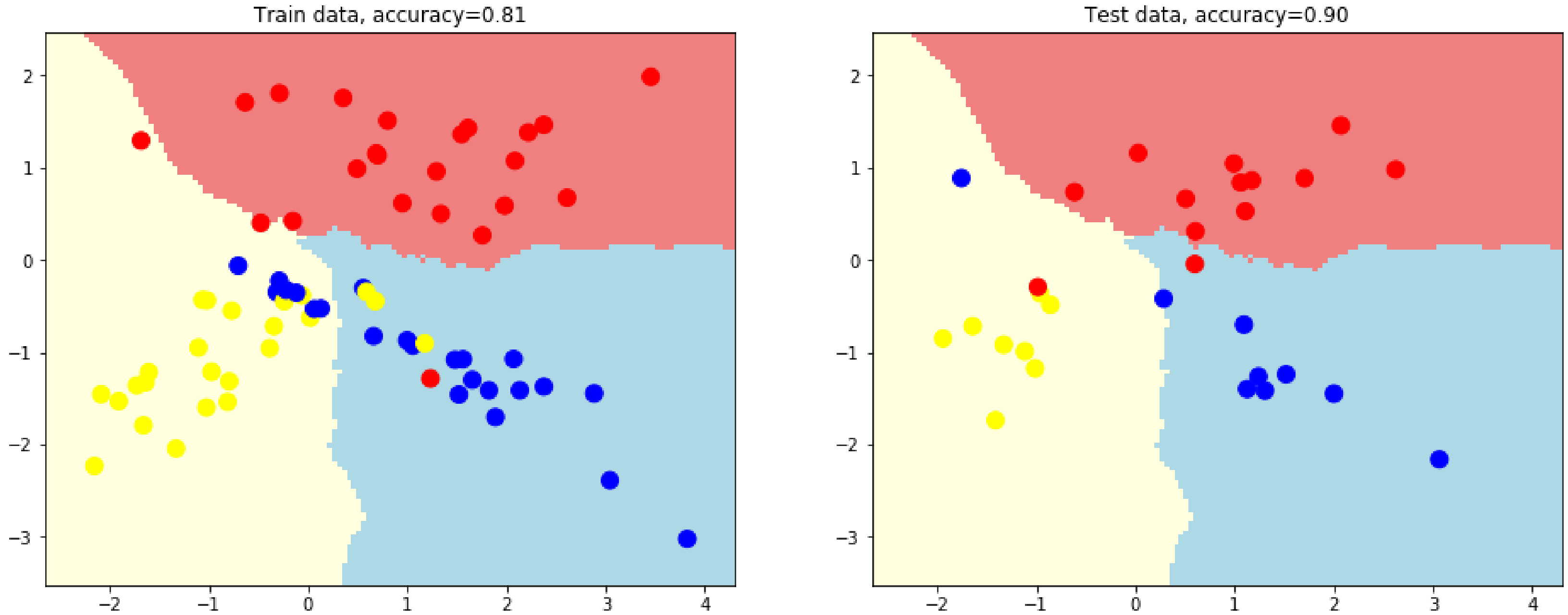


In [25]:
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=20)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```

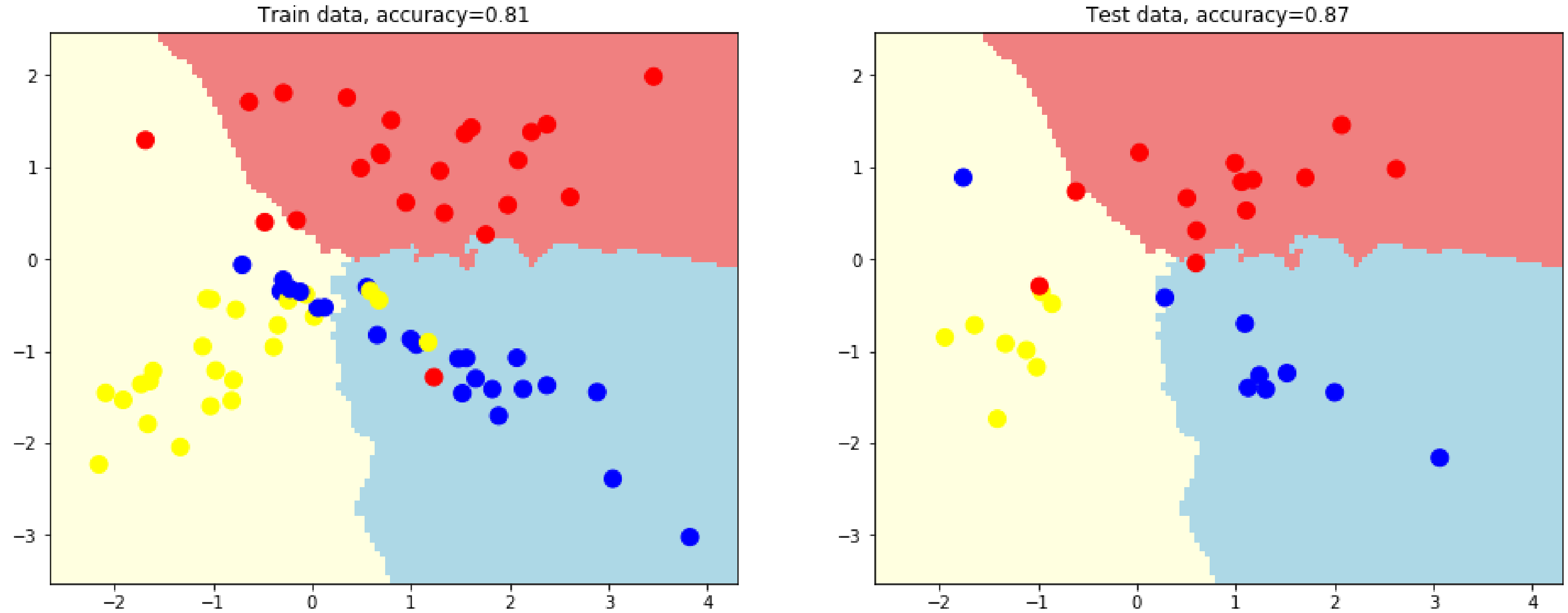




In [26]:
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=30)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```



In [27]:
```
estimator = neighbors.KNeighborsClassifier(n_neighbors=40)

plot_decision_surface(estimator, train_data, train_labels, test_data, test_la
bels)
```

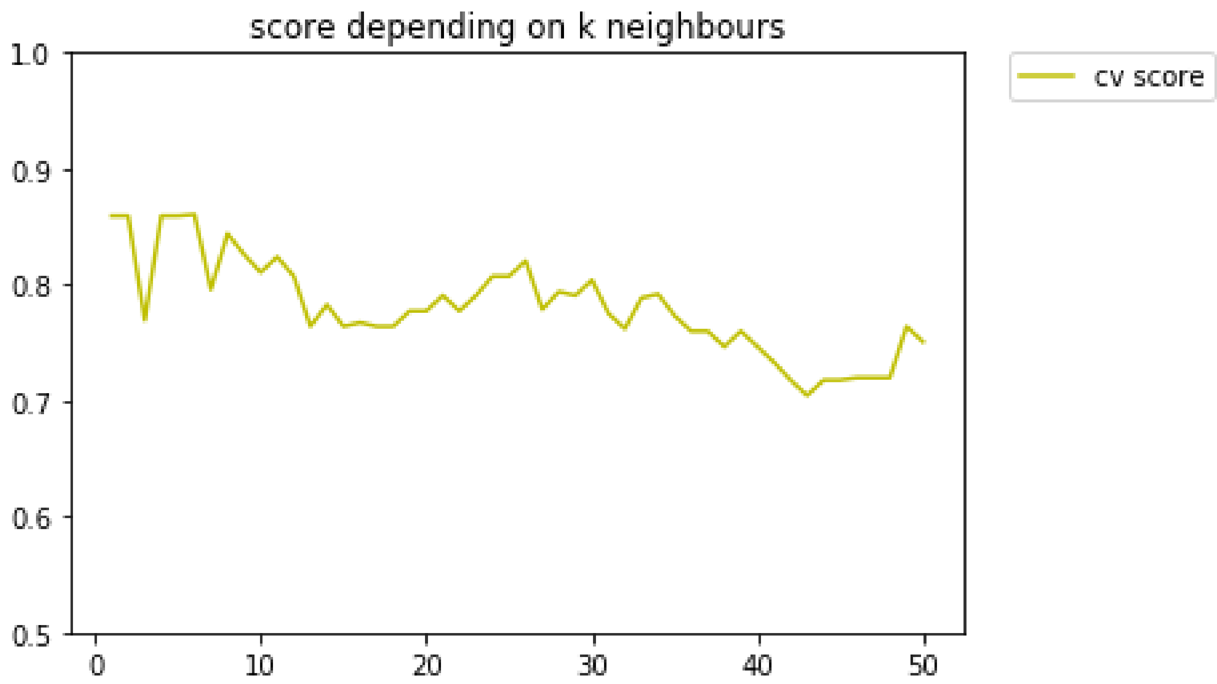


In [28]:
```
cross_val_scores = []
```

In [29]:
```
for k in range(1, 51):
    estimator = neighbors.KNeighborsClassifier(n_neighbors=k)
    cross_val_scores.append((model_selection.cross_val_score(estimator, train_data, train_labels, cv=5).mean()))
#     test_scores.append(est)
```

In [30]:
```
plt.plot(np.arange(1, 51), cross_val_scores, c="y", label="cv score")
plt.ylim([0.5, 1.0])
plt.title("score depending on k neighbours")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[30]:
```
<matplotlib.legend.Legend at 0x7f7fa4e41cd0>
```



In [39]:
```
np.where(cross_val_scores == cross_val_scores[np.argmax(cross_val_scores)])
```

Out[39]:
```
(array([5]),)
```

Наилучшее k, согласно кросс-валидации, равно 5.

In [ ]: