Homework 4

April 26, 2020

1 Homework 4 - Classification and Clustering

1.1 *Ran Ju*

Netid: rj133

2 1) Conceptual questions

2.0.1 (1.5 points total)

(a) Considering a binary problem: Is it possible to compute an ROC curve for a 5-Nearest Neighbor classifier? (0.5 points) Is it possible to compute an ROC curve for a decision tree with no limitation on the maximal depth? (0.5 points) Explain your answers.

(b) You produce ten bootstrapped samples from a data set containing classes A and B. Then you use a pre-trained logistic regression classifier on each sample and produce ten estimates of P(class = A): {0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75}. What are the predictions of a soft voting and a hard voting classifier? (0.5 points)

ANSWER

(a) It is **possible** to compute an ROC curve for a 5-Nearest Neighbor classifier because becausthee output of the can be probability thus we can compute the curve. It is **impossible** to compute an ROC curve for a decision tree with no limitation on the maximal depth because when there is no limit of depth, the output probability will be hard to varied, the final output is more like a label rather than a probability.

(b) The prediction of a hard voting is class A, because in hard voting, the result follows with the majority. Here, six votes for A and four votes for not A, $4 > 6$, so the result is **A**. The prediction of a soft voting is not class A, because the result is based on average probability. Here average is $\frac{0.1+0.15+0.2+0.2+0.55+0.6+0.6+0.65+0.7+0.75}{10} = 0.45 < 0.5$, so the result is **not A**.

3 2) k-means clustering

3.0.1 (5 points total)

(a) Use make_blobs from `scikit-learn` to create two datasets with 2 and 5 cluster centers, each containing 5000 samples with 2 features (0.5 points).

(b) For both datasets run the k-means algorithm with k=5. Plot the results using different colors to indicate the clusters (0.5 points).

(c) Use the cluster centers from (b) to compute the sum of square error (i.e. loop over all datapoints) (2 points).

(d) For both dataset run the k-means algorithm for values of k from 1 to 10 and then plot the “elbow curve” using the sum of square error. (1.5 points)

(e) For both dataset, discuss if the elbow method works (0.5 points).

ANSWER

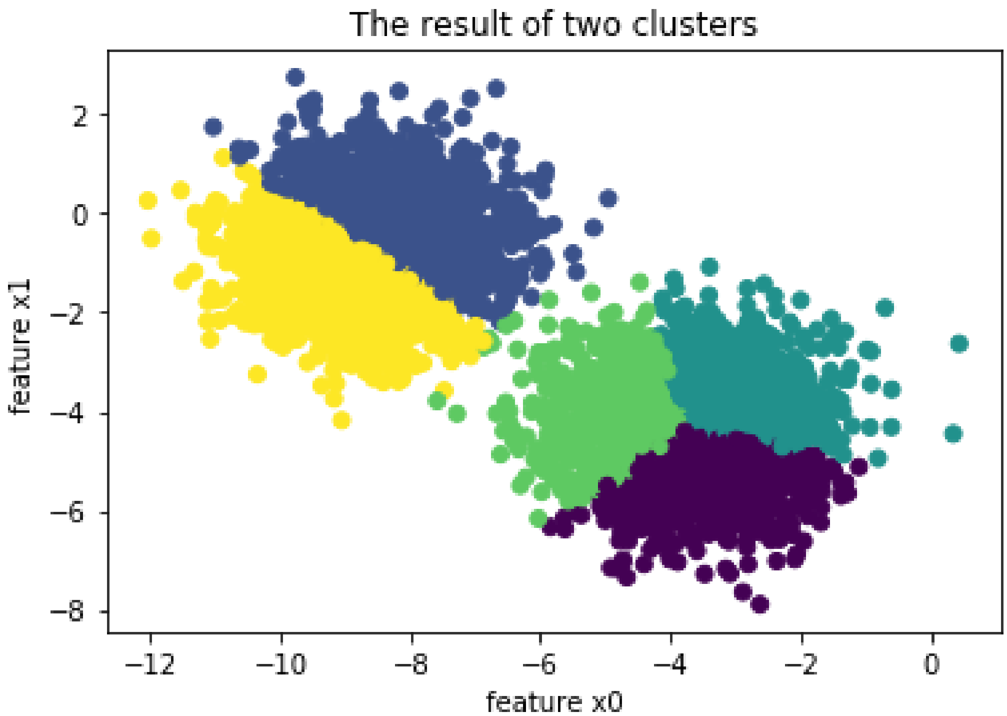
[1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
from sklearn.datasets import make_moons
from sklearn.datasets import make_blobs
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.cluster import KMeans
import math
```

[2]:
```
#(a) create datasets
X2, y2=make_blobs(n_samples=5000, n_features=2, centers=2)#2 clusters
X5, y5=make_blobs(n_samples=5000, n_features=2, centers=5)#5 clusters
```

[3]:
```
#(b) k-menas
kmodel2=KMeans(n_clusters=5)
kmodel5=KMeans(n_clusters=5)
y_pred2=kmodel2.fit_predict(X2)# prediction for the 2-clusters data
center2=kmodel2.cluster_centers_#the center
y_pred5=kmodel5.fit_predict(X5)# prediction for the 5-clusters data
center5=kmodel5.cluster_centers_#the center
```

[4]:
```
plt.scatter(X2[:, 0], X2[:, 1], c =y_pred2)
plt.xlabel('feature x0')
plt.ylabel('feature x1')
plt.title('The result of two clusters')
```

[4]:
```
Text(0.5, 1.0, 'The result of two clusters')
```



[5]:
```
plt.scatter(X5[:, 0], X5[:, 1], c =y_pred5)
plt.xlabel('feature x0')
plt.ylabel('feature x1')
plt.title('The result of five clusters')
```

[5]: Text(0.5, 1.0, 'The result of five clusters')

[6]:
```
def sum_square_error(k,x,ypredict,center):
    #k:number of clusters
    #x:the features
    #ypredict: the predict
    #center: cluster center
    sum=0
    for i in range(k):
        dis=np.sum((x[ypredict==i]-center[i])**2)# the distance between each␣
 ↪point to the sample
        sum+=dis
    return sum
```

[7]:
```
# (c) mean square error
SSE2 = sum_square_error(5,X2,y_pred2,center2)
SSE5 = sum_square_error(5,X5,y_pred5,center5)
print('The mean square error of two clusters is '+str(SSE2))
print('The mean square error of five clusters is '+str(SSE5))
```

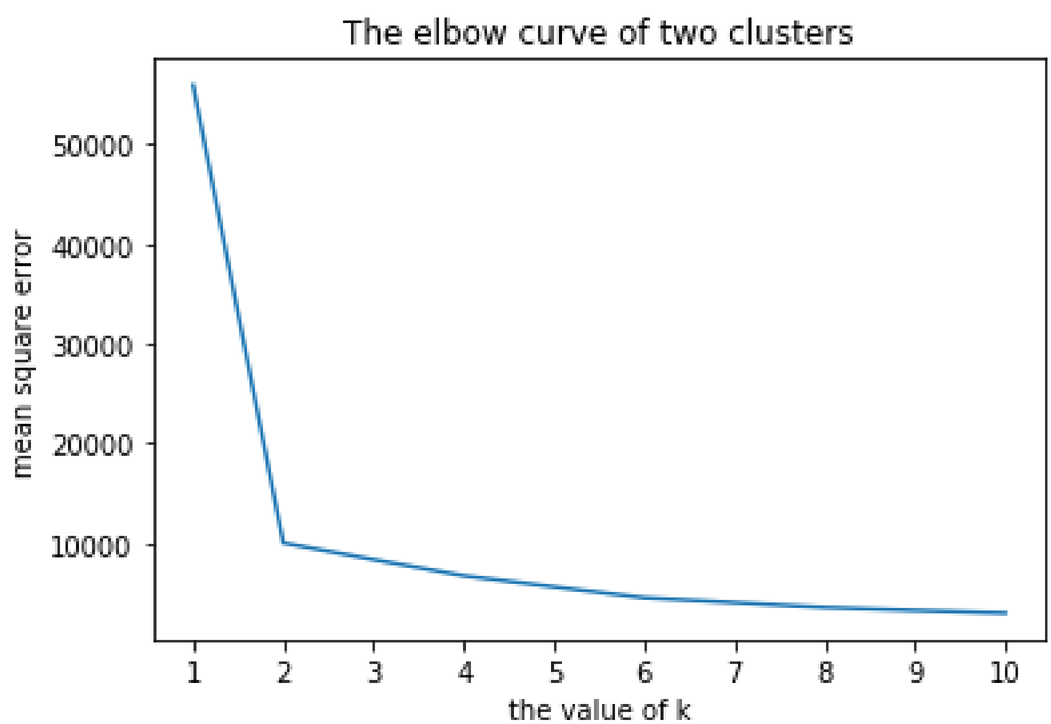
```
The mean square error of two clusters is 5629.700974914748
The mean square error of five clusters is 9693.473182630245
```

[8]:
```
# (d) elbow curve
# calculate the mean square error for each k value
```

```
SSE2=[]
SSE5=[]
K=[i for i in range(1,11)]
for k in K:
    kmodel2=KMeans(n_clusters=k)#set up model
    kmodel5=KMeans(n_clusters=k)
    y_pred2=kmodel2.fit_predict(X2)#predict
    center2=kmodel2.cluster_centers_#cluster center
    y_pred5=kmodel5.fit_predict(X5)
    center5=kmodel5.cluster_centers_
    SSE2.append(sum_square_error(k,X2,y_pred2,center2))# calculate the sum␣
↪square error
    SSE5.append(sum_square_error(k,X5,y_pred5,center5))
```

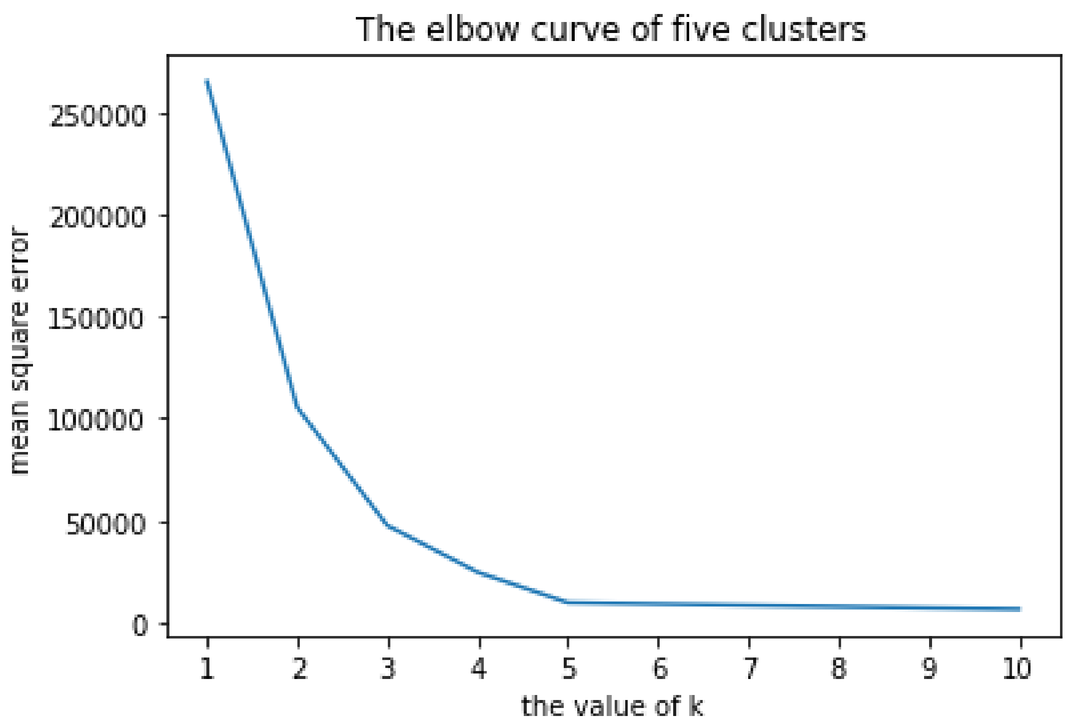
[9]:
```
plt.plot(K, SSE2)
plt.locator_params('x',nbins=10)# show ten numbers in x axis
plt.xlabel('the value of k')
plt.ylabel('mean square error')
plt.title('The elbow curve of two clusters')
```

[9]:
```
Text(0.5, 1.0, 'The elbow curve of two clusters')
```



[10]:
```
plt.plot(K, SSE5)
plt.locator_params('x',nbins=10)# show ten numbers in x axis
plt.xlabel('the value of k')
plt.ylabel('mean square error')
plt.title('The elbow curve of five clusters')
```

[10]:
```
Text(0.5, 1.0, 'The elbow curve of five clusters')
```



(e) For the two clusters case, the elbow method **works**, there is a clear elbow at k=2. For the five clusters case, the elbow method seems **do not work**, we can see there are several elbows here, so it is hard to decide which is the real elbow.

4 3) Bias-variance tradeoff for the kNN classifier

4.0.1 (3.5 points total)

(a) use make moon to create a dataset of 1000 random samples with noise=0.35. Scatterplot the dataset. (0.5 points)

(b) Select 400 of the 1000 data points at random. Use this dataset to train three k-Nearest Neighbor classifiers with $k = \{1, 20, 140\}$ (1 point).

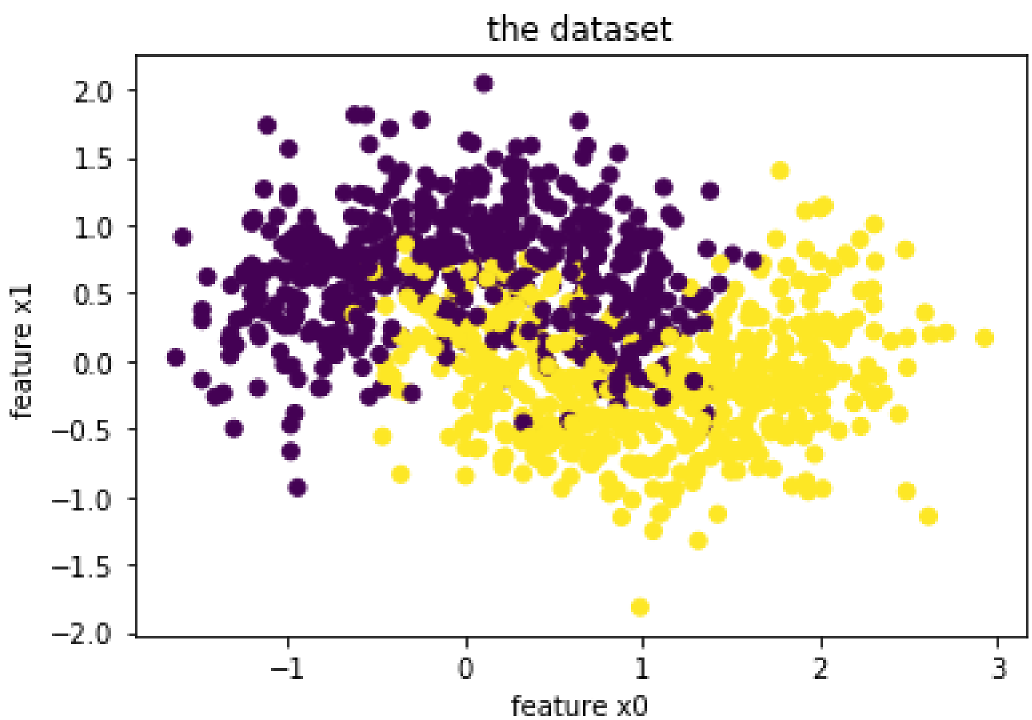
(c) Create three plots showing the three decision boundaries together with the training data (0.5 points).

(d) Split the dataset from (a) in two equal sized test and training datasets. Train a kNN classifier on your training set for k=1,2,...140. Apply each of these trained classifiers to both your training dataset and your test dataset and plot the classification error (fraction of mislabeled datapoints) using a logarithmic x-axis (1.5 points).

ANSWER

[11]:
```python
#(a) create dataset
X, y = make_moons(n_samples=1000, noise=0.35)
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.xlabel('feature x0')
plt.ylabel('feature x1')
plt.title('the dataset')
```

[11]:
```
Text(0.5, 1.0, 'the dataset')
```



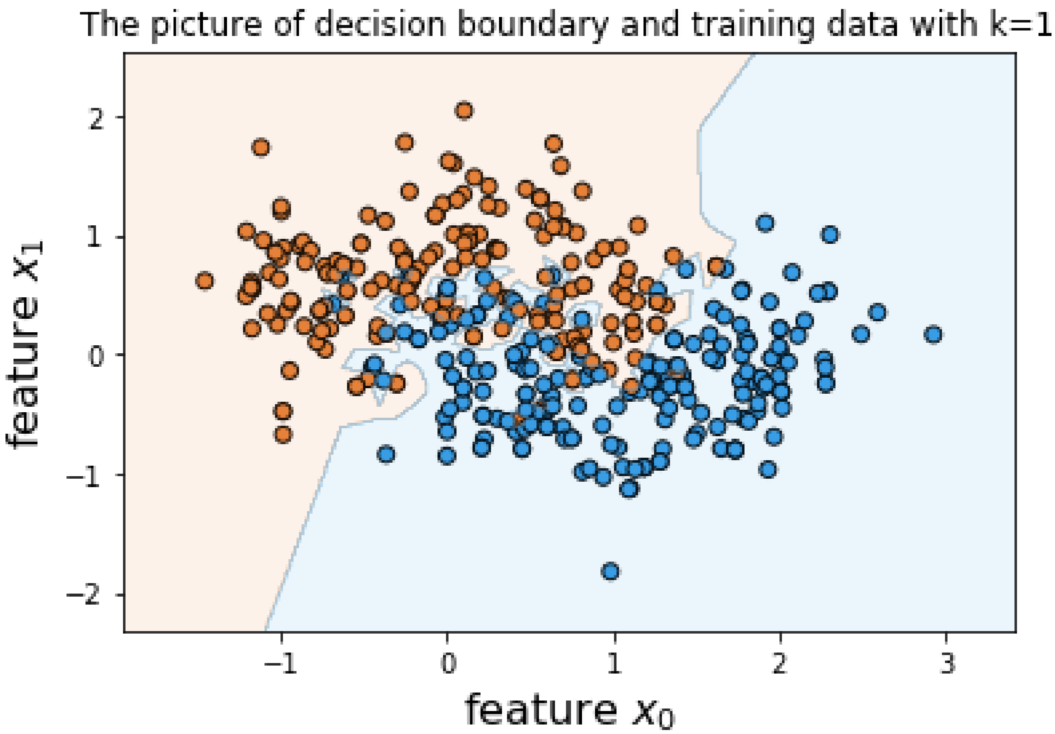
[12]:
```python
# (b) train KNN
m=np.random.randint(0,1000,400)#choose 400 indexs out of 1000 randomly
xtrain=X[m]
ytrain=y[m]
# create the KNN for each k and see their performance
knn1 = KNeighborsClassifier(n_neighbors=1)
knn1.fit(xtrain,ytrain)
print('The training score of k-Nearest Neighbor classifiers with  =1:\n',knn1.
 ↪score(xtrain,ytrain))
```

```
knn20 = KNeighborsClassifier(n_neighbors=20)
knn20.fit(xtrain,ytrain)
print('The training score of k-Nearest Neighbor classifiers with  =20:
 ↪\n',knn20.score(xtrain,ytrain))
knn140 = KNeighborsClassifier(n_neighbors=140)
knn140.fit(xtrain,ytrain)
print('The training score of k-Nearest Neighbor classifiers with  =140:
 ↪\n',knn140.score(xtrain,ytrain))
```

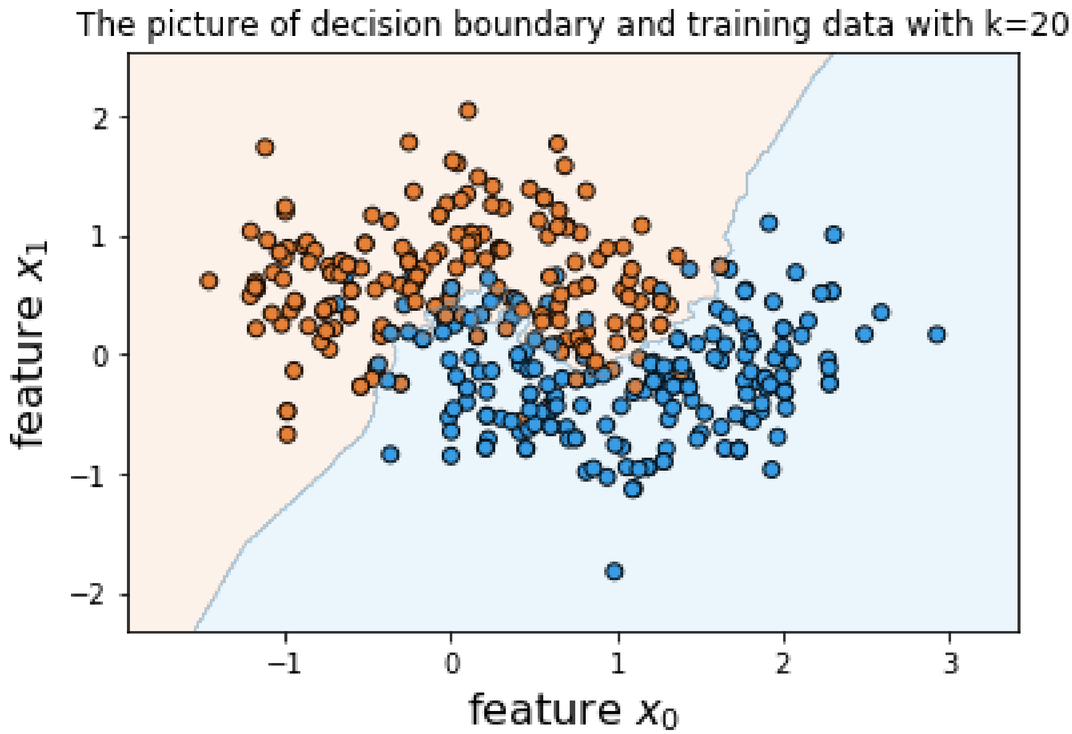
```
The training score of k-Nearest Neighbor classifiers with  =1:
 1.0
The training score of k-Nearest Neighbor classifiers with  =20:
 0.87
The training score of k-Nearest Neighbor classifiers with  =140:
 0.84
```

[13]:

```
#(c) show the decision boundaries
cm_tree = ListedColormap(['#e58139','#399de5'])
h = .01  # step size
# determine boundaries
x1_min, x1_max = xtrain[:, 0].min() - .5, xtrain[:, 0].max() + .5
x2_min, x2_max = xtrain[:, 1].min() - .5, xtrain[:, 1].max() + .5
# assign predictions to each mesh point
xx1, yy1 = np.meshgrid(np.arange(x1_min, x1_max, h), np.arange(x2_min, x2_max,␣
 ↪h))
Z1 = knn1.predict(np.c_[xx1.ravel(), yy1.ravel()])
Z1 = Z1.reshape(xx1.shape)
# plot training data
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=ytrain,  cmap=cm_tree, edgecolors='k')
# plot decision boundary
plt.contourf(xx1, yy1, Z1, cmap=cm_tree, alpha=.1)
plt.xlabel('feature $x_0$', size=16)
plt.ylabel('feature $x_1$', size=16)
plt.title('The picture of decision boundary and training data with k=1')
plt.show()
```

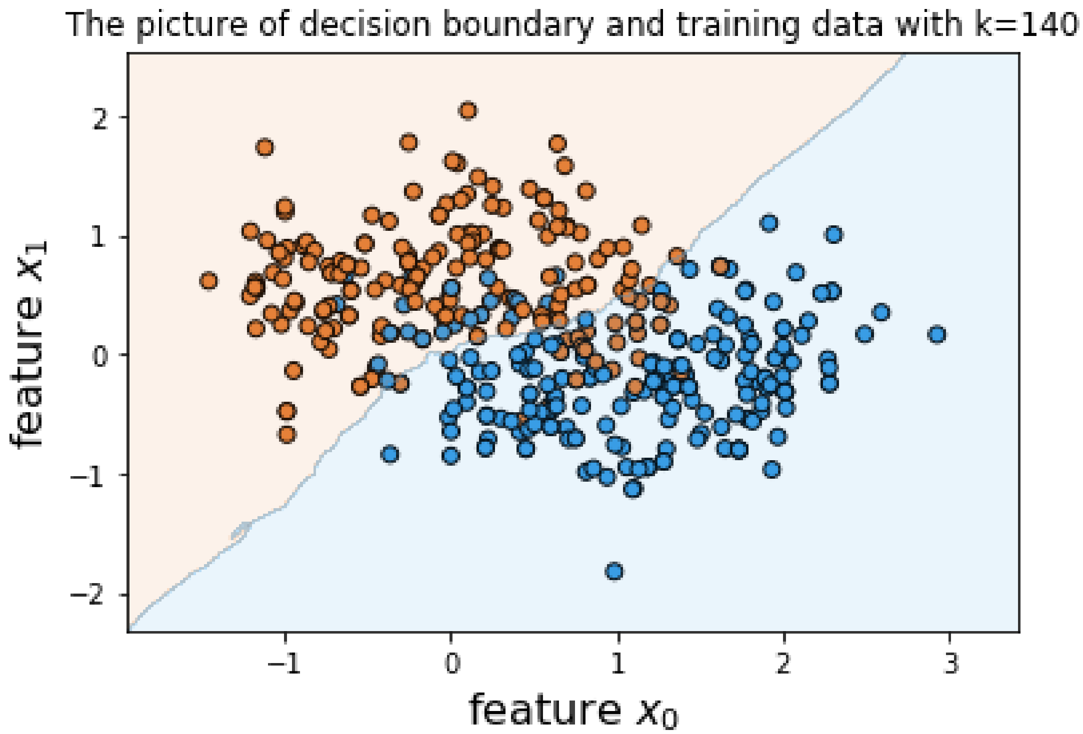


[14]:
```
# assign predictions to each mesh point
xx20, yy20 = np.meshgrid(np.arange(x1_min, x1_max, h), np.arange(x2_min,␣
↪x2_max, h))
Z20 = knn20.predict(np.c_[xx20.ravel(), yy20.ravel()])
Z20 = Z20.reshape(xx20.shape)
# plot training data
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=ytrain,  cmap=cm_tree, edgecolors='k')
# plot decision boundary
plt.contourf(xx20, yy20, Z20, cmap=cm_tree, alpha=.1)
plt.xlabel('feature $x_0$', size=16)
plt.ylabel('feature $x_1$', size=16)
plt.title('The picture of decision boundary and training data with k=20')
plt.show()
```

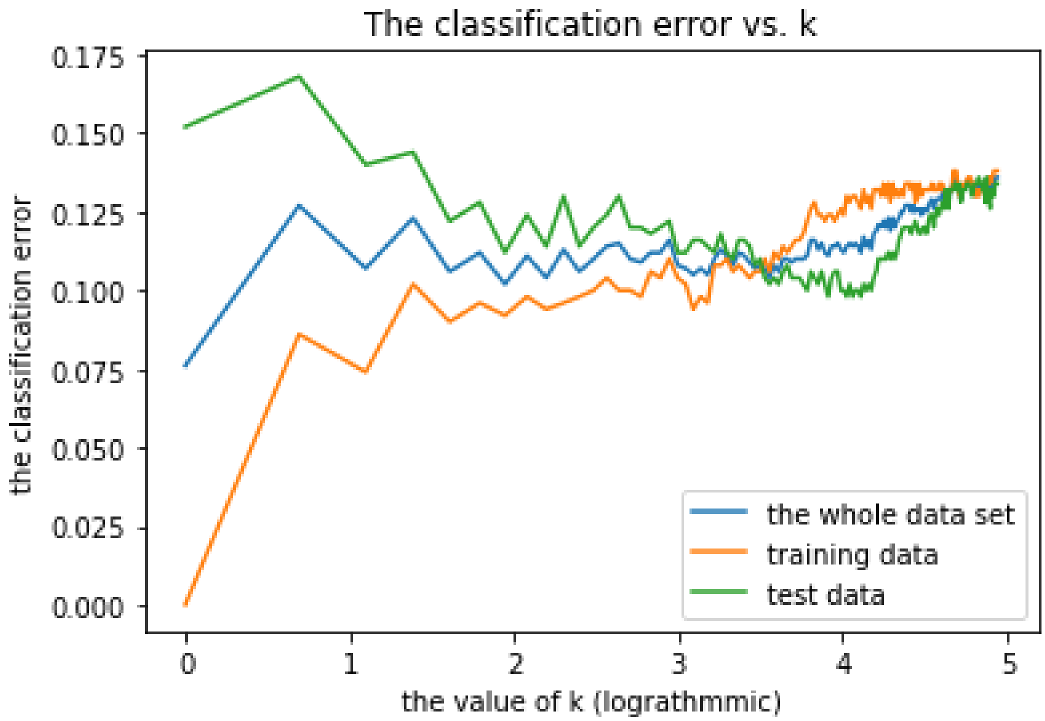


[15]:
```
# assign predictions to each mesh point
xx140, yy140 = np.meshgrid(np.arange(x1_min, x1_max, h), np.arange(x2_min,␣
↪x2_max, h))
Z140 = knn140.predict(np.c_[xx140.ravel(), yy140.ravel()])
Z140 = Z140.reshape(xx140.shape)
# plot training data
plt.scatter(xtrain[:, 0], xtrain[:, 1], c=ytrain,  cmap=cm_tree, edgecolors='k')
# plot decision boundary
plt.contourf(xx140, yy140, Z140, cmap=cm_tree, alpha=.1)

plt.xlabel('feature $x_0$', size=16)
plt.ylabel('feature $x_1$', size=16)
plt.title('The picture of decision boundary and training data with k=140')
plt.show()
```



[16]:
```
#(d) plot classification error
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)# split␣
↪the data set
K=[i for i in range(1,141)]
totalfraction=[]
train=[]
test=[]
for k in K:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    y_pred = knn.predict(X)
    totalfraction.append(1-accuracy_score(y, y_pred))# fraction of error␣
↪classification for the whole data
    train.append(1-accuracy_score(y_train,knn.predict(X_train)))
    test.append(1-accuracy_score(y_test,knn.predict(X_test)))
plt.plot([math.log(i) for i in K],totalfraction,label='the whole data set')
plt.plot([math.log(i) for i in K],train,label='training data')
plt.plot([math.log(i) for i in K],test,label='test data')
plt.xlabel('the value of k (lograthmmic)')
plt.ylabel('the classification error')
plt.title('The classification error vs. k')
plt.legend()
plt.show()
```



The classification error vs. k
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
the classification error
0
1
2
3
4
5
the value of k (lograthmmic)
the whole data set
training data
test data