Duvall Pinkney WEEK 4 homework: classification Evaluate k-nearest-neighbor method with different k values on the dataset used in Week 03 notebook (Source: https://raw.githubusercontent.com/empathy87/The-Elements-of-Statistical-Learning-Python-Notebooks/master/data/mixture.txt (https://raw.githubusercontent.com/empathy87/The-Elements-of-Statistical-Learning-Python-Notebooks/master/data/mixture.txt))

1. Load the data as a Pandas data frame.
2. Split the data into 80% training data and 20% test data.
3. Build three k-nearest-neighbor model with k = 1, 5, 25, respectively.
4. Train the models on the training set, and obtain the model predictions on the test set.
5. Calculate the test accuracy score for each model. Which k value give the best accuracy score?

In [51]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [54]:
```
url = "https://pjreddie.com/media/files/mnist_train.csv"

train_data = pd.read_csv(url, header=None, sep=',')
train_data
```

Out[54]:

	0	1	2	3	4	5	6	7	8	9	...	775	776	777	778	779	780	781	782	783	784
0	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59995	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59996	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59997	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59998	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59999	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

60000 rows × 785 columns

In [55]:
```
url2 = "https://pjreddie.com/media/files/mnist_test.csv"

test_data = pd.read_csv(url2, header=None, sep=',')
test_data
```

Out[55]:

	0	1	2	3	4	5	6	7	8	9	...	775	776	777	778	779	780	781	782	783	784
0	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9996	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9997	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9998	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9999	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10000 rows × 785 columns

In [56]:
```
print("Shape:",test_data.shape)
print("Columns names", test_data.columns)
print("Data types: ", test_data.dtypes)
```

```
Shape: (10000, 785)
Columns names Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
            ...
            775, 776, 777, 778, 779, 780, 781, 782, 783, 784],
           dtype='int64', length=785)
Data types:  0      int64
1      int64
2      int64
3      int64
4      int64
       ...
780    int64
781    int64
782    int64
783    int64
784    int64
Length: 785, dtype: object
```

In [42]:
```
print("Shape:",train_data.shape)
print("Columns names", train_data.columns)
print("Data types: ", train_data.dtypes)
```

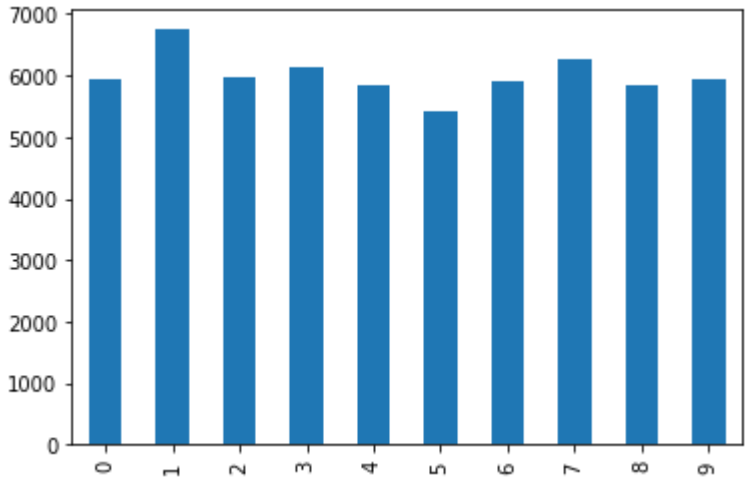
```
Shape: (59999, 785)
Columns names Index(['5', '0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8',
       ...
       '0.608', '0.609', '0.610', '0.611', '0.612', '0.613', '0.614', '0.615',
       '0.616', '0.617'],
      dtype='object', length=785)
Data types:  5        int64
0        int64
0.1      int64
0.2      int64
0.3      int64
         ...
0.613    int64
0.614    int64
0.615    int64
0.616    int64
0.617    int64
Length: 785, dtype: object
```

In [57]:
```
data = train_data.rename({0: 'label'}, axis=1)
data['label'].value_counts().sort_index()
```

Out[57]:
```
0    5923
1    6742
2    5958
3    6131
4    5842
5    5421
6    5918
7    6265
8    5851
9    5949
Name: label, dtype: int64
```

In [59]:
```
data['label'].value_counts().sort_index().plot.bar()
```

Out[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bfa8742a88>
```



In [60]:
```
def split_train_test(data, test_ratio):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]
```

In [23]:
```
testing_data = split_train_test(test_data, 0.2)
len(testing_data)
```

Out[23]:
```
2
```

In [24]:
```
training_data = split_train_test(train_data, 0.8)
len(training_data)
```

Out[24]:
```
2
```

In [62]:
```
# Visualize data as images
ind = 123 # Going to show the image on this row
input_features = [x for x in data.columns if x != "label"]
# print(input_features)
data_example = data.loc[ind, input_features] # Use .loc[] expression to extract data from a data frame.
print(data_example.shape)
```

```
(784,)
```

In [63]:
```
# Convert the data example to a numpy array
data_example_array = data_example.values
print(data_example_array.shape)
```

```
(784,)
```

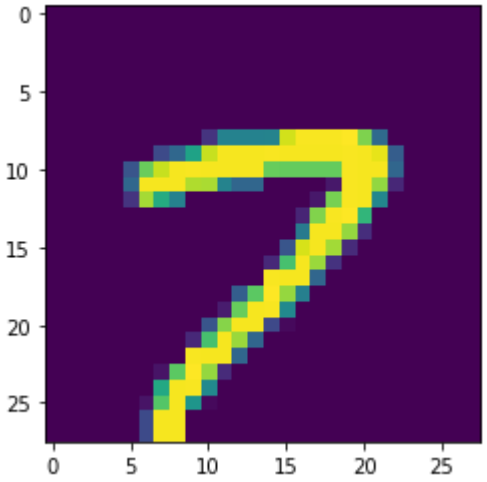
In [64]:
```
# Transform the array to a 28*28 2D array
data_example_array_transformed = data_example_array.reshape([28, 28])
print(data_example_array_transformed.shape)
print(data_example_array_transformed)
```

```
(28, 28)
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  38 113 113 113 114 238 253 253
  253 255 206  88   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  57  85 147 234 252 252 252 253 252 252 252
  252 253 252 246  75   0   0   0   0   0]
 [  0   0   0   0   0  67 197 234 252 253 252 252 252 252 196 195 195 195
  195 253 252 252  84   0   0   0   0   0]
 [  0   0   0   0   0  85 252 252 252 225 223 114  84  84   0   0   0   0
   16 253 252 214  28   0   0   0   0   0]
 [  0   0   0   0   0  38 221 157 112   0   0   0   0   0   0   0   0  16
  203 253 252 118   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  26 207
  253 255 168   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 104 252
  252 215  33   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  67 240 252
  220  31   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  29 181 252 217
   37   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 253 252 252  84
    0   0   0   0   0   0   0   0   0   0]
```

```
[  0   0   0   0   0   0   0   0   0   0   0   0  76 191 255 215 110   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0  19 196 252 253 121   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0  67 209 252 252  56   6   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0  29 181 252 217  84   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0 253 252 252  84   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0  13 191 255 215  31   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0 154 252 253 121   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0  13 187 252 133   6   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0  57 252 252   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0  57 252 252   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]]
```

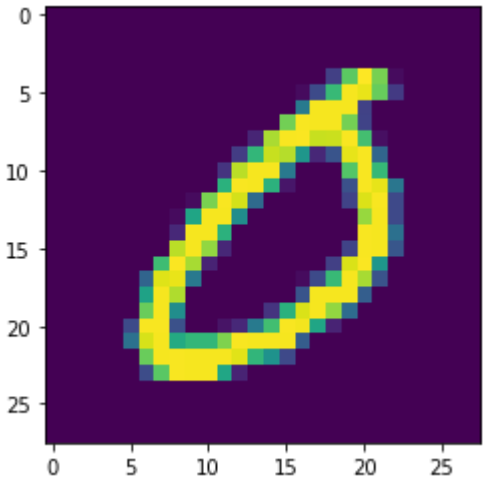
In [65]:
```
plt.imshow(data_example_array_transformed)
```

Out[65]:
```
<matplotlib.image.AxesImage at 0x1bfa8bc8b08>
```



In [66]:
```
# Write a function to automate the process
def get_image(data, ind):
    # Use data.loc to extract the 784 pixel values

    input_features2 = [column for column in data.columns if column != 'label']
    data_example = data.loc[ind, input_features2]


    # Convert the list to a numpy array
    data_example_numpyArray = data_example.values

    # Change the shape to [28, 28]
    data_example_numpyArray_transformed = data_example_numpyArray.reshape([28,28])

    # Use imshow() to display the image.
    plt.imshow(data_example_numpyArray_transformed)

#     return data_example_array_transformed
```

In [67]:
```
ind = 4321
get_image(data, ind)
```



In [39]:

```
# How to evaluate the kNN model?

# Visualize the decision region.

# 1. create a 100 * 100 grid of points to cover the entire plane.
plot_data = pd.DataFrame()
x1_coordinates = np.linspace(-3, 4, 100) # sample 100 points per row
x2_coordinates = np.linspace(-2, 3, 100) # sample 100 rows (total: 100 * 100 = 10,000 points)
for x1 in x1_coordinates:
    for x2 in x2_coordinates:
        plot_data = plot_data.append({'x1': x1,
                                      'x2': x2},
                                     ignore_index=True)
plot_data

# 2. use the kNN model to make a prediction on each of the 10,000 points.
plot_data['prediction'] = model_3nn.predict(plot_data)
plot_data.head()
```

Out[39]:

	x1	x2
0	-3.0	-2.000000
1	-3.0	-1.949495
2	-3.0	-1.898990
3	-3.0	-1.848485
4	-3.0	-1.797980

In [68]:
```
# Create a smaller training set to reduce training time
sample_size = 6000
samples = np.random.choice(data.index, sample_size, replace=False)
mnist_train_small = data.loc[samples]
print(mnist_train_small.shape)
```

```
(6000, 785)
```

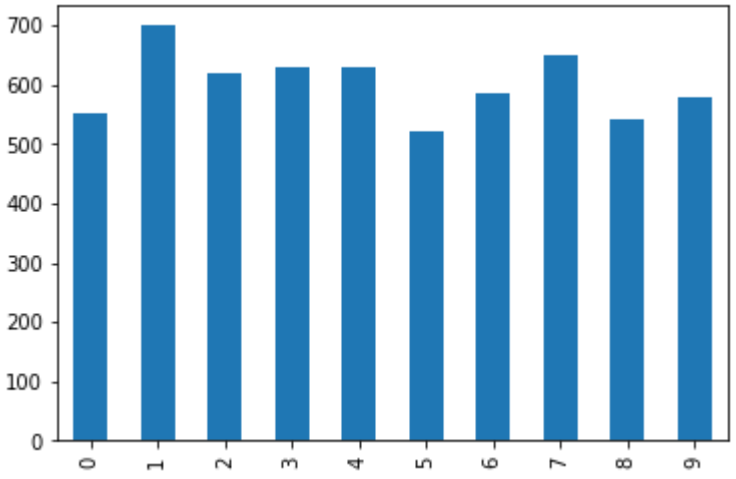
In [69]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

Out[69]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [49]:
```
input_features = [ x for x in plot_data.columns if x != "label"]
```

In [70]:
```
# Verify mnist_train_small still contains enough training examples for each label

mnist_train_small['label'].value_counts().sort_index().plot.bar()
```

Out[70]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bfd2036788>
```



In [71]:
```
#K nearest neighbor model
#Build three k-nearest-neighbor model with k = 1, 5, 25

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(mnist_train_small[input_features], mnist_train_small['label'])
```

Out[71]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

In [72]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(mnist_train_small[input_features], mnist_train_small['label'])
```

Out[72]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [73]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=25)
knn.fit(mnist_train_small[input_features], mnist_train_small['label'])
```

Out[73]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=25, p=2,
                     weights='uniform')
```

In [ ]:
```

```

In [86]:
```
# Build a linear SVM classifier
from sklearn.svm import LinearSVC
model_svm = LinearSVC()
model_svm.fit(mnist_train_small[input_features], mnist_train_small['label'])
```

```
C:\Users\thees\Anaconda3\lib\site-packages\sklearn\svm\base.py:929: ConvergenceWarning: Liblinear failed to co
nverge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[86]:
```
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 [87]:
```
# Use the model to make predictions on the test images
test_data['prediction_svm'] = model_svm.predict(test_data[input_features])
```

In [88]:
```
# Calculate accuracy score

accuracy_score(test_data['label'], test_data['prediction_svm'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-88-03c111e3d8d0> in <module>
      1 # Calculate accuracy score
      2
----> 3 accuracy_score(test_data['label'], test_data['prediction_svm'])

NameError: name 'accuracy_score' is not defined
```

In [77]:
```
test_data.head()
```

Out[77]:

	0	1	2	3	4	5	6	7	8	9	...	775	776	777	778	779	780	781	782	783	784
0	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

In [85]:
```
# Extract rows that are mis-classified

filter1 = (test_data['label'] != test_data['prediction'])
test_data[filter1][['label', 'prediction']]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'label'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-85-167524130096> in <module>
      1 # Extract rows that are mis-classified
      2
----> 3 filter1 = (test_data['label'] != test_data['prediction'])
      4 test_data[filter1][['label', 'prediction']]

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2993             if self.columns.nlevels > 1:
   2994                 return self._getitem_multilevel(key)
-> 2995             indexer = self.columns.get_loc(key)
   2996             if is_integer(indexer):
   2997                 indexer = [indexer]

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(key))
```

```
   2900         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:
```

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'label'

In [79]:
```
# Show one image that is incorrectly classified

# Create a copy of test_data without the prediction and prediction_svm column
test_data2 = test_data.drop(['prediction', 'prediction_svm'], axis=1)

ind = 9922
get_image(test_data2, ind)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-79-5adc46439460> in <module>
      2
      3 # Create a copy of test_data without the prediction and prediction_svm column
----> 4 test_data2 = test_data.drop(['prediction', 'prediction_svm'], axis=1)
      5
      6 ind = 9922

~\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, inpla
ce, errors)
   4115             level=level,
   4116             inplace=inplace,
-> 4117             errors=errors,
   4118         )
   4119

~\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level, inp
lace, errors)
   3912         for axis, labels in axes.items():
   3913             if labels is not None:
-> 3914                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3915
   3916         if inplace:

~\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors)
   3944                 new_axis = axis.drop(labels, level=level, errors=errors)
   3945             else:
-> 3946                 new_axis = axis.drop(labels, errors=errors)
   3947             result = self.reindex(**{axis_name: new_axis})
   3948

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   5338         if mask.any():
```

```
   5339             if errors != "ignore":
-> 5340                 raise KeyError("{} not found in axis".format(labels[mask]))
   5341             indexer = indexer[~mask]
   5342         return self.delete(indexer)

KeyError: "['prediction' 'prediction_svm'] not found in axis"
```

In [76]:
```
# Show the label and the prediction of this image
print("Label:", test_data.loc[ind, 'label'])
print("Prediction by kNN:", test_data.loc[ind, 'prediction'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index_class_helper.pxi in pandas._libs.index.Int64Engine._check_type()

KeyError: 'label'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-76-5fe95549b521> in <module>
      1 # Show the label and the prediction of this image
----> 2 print("Label:", test_data.loc[ind, 'label'])
      3 print("Prediction by kNN:", test_data.loc[ind, 'prediction'])

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(self, key)
   1416                 except (KeyError, IndexError, AttributeError):
   1417                     pass
-> 1418             return self._getitem_tuple(key)
   1419         else:
   1420             # we by definition only have the 0th axis

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_tuple(self, tup)
    803     def _getitem_tuple(self, tup):
    804         try:
--> 805             return self._getitem_lowerdim(tup)
    806         except IndexingError:
    807             pass

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_lowerdim(self, tup)
```

```
    959                     return section
    960                 # This is an elided recursive call to iloc/loc/etc'
--> 961                 return getattr(section, self.name)[new_key]
    962
    963         raise IndexingError("not applicable")

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(self, key)
   1422
   1423             maybe_callable = com.apply_if_callable(key, self.obj)
-> 1424             return self._getitem_axis(maybe_callable, axis=axis)
   1425
   1426     def _is_scalar_access(self, key: Tuple):

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self, key, axis)
   1848         # fall thru to straight lookup
   1849         self._validate_key(key, axis)
-> 1850         return self._get_label(key, axis=axis)
   1851
   1852

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get_label(self, label, axis)
    154             # but will fail when the index is not present
    155             # see GH5667
--> 156             return self.obj._xs(label, axis=axis)
    157         elif isinstance(label, tuple) and isinstance(label[axis], slice):
    158             raise IndexingError("no slices here, handle elsewhere")

~\Anaconda3\lib\site-packages\pandas\core\generic.py in xs(self, key, axis, level, drop_level)
   3735             loc, new_index = self.index.get_loc_level(key, drop_level=drop_level)
   3736         else:
-> 3737             loc = self.index.get_loc(key)
   3738
   3739             if isinstance(loc, np.ndarray):

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()
```

```
pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index_class_helper.pxi in pandas._libs.index.Int64Engine._check_type()

KeyError: 'label'
```

In [ ]: