PacktPublishing / Learning-Data-Mining-with-Python Public

Code Issues Pull requests 1 Actions Security Insights

Learning-Data-Mining-with-Python / Chapter 2 / Ionosphere Nearest Neighbour.ipynb

taabishk Added files via upload b3bf230 · 9 years ago

410 lines (410 loc) · 126 KB

Learning-Data-Mining-with-Python / Chapter 2 / Ionosphere Nearest Neighbour.ipynb Top

Preview Code Blame Raw

In [6]:

```
%matplotlib inline
```

In [7]:

```
import os
home_folder = os.path.expanduser("~")
print(home_folder)
```

```
/home/bob
```

In [8]:

```
# Change this to the location of your dataset
data_folder = os.path.join(home_folder, "Data", "Ionosphere")
data_filename = os.path.join(data_folder, "ionosphere.data")
print(data_filename)
```

```
/home/bob/Data/Ionosphere/ionosphere.data
```

In [9]:

```
import csv
import numpy as np

# Size taken from the dataset and is known
X = np.zeros((351, 34), dtype='float')
y = np.zeros((351,), dtype='bool')

with open(data_filename, 'r') as input_file:
    reader = csv.reader(input_file)
    for i, row in enumerate(reader):
        # Get the data, converting each item to a float
        data = [float(datum) for datum in row[:-1]]
        # Set the appropriate row in our dataset
        X[i] = data
        # 1 if the class is 'g', 0 otherwise
        y[i] = row[-1] == 'g'
```

In [10]:

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=14)
print("There are {} samples in the training dataset".format(X_train.shape[0]))
print("There are {} samples in the testing dataset".format(X_test.shape[0]))
print("Each sample has {} features".format(X_train.shape[1]))
```

```
There are 263 samples in the training dataset
There are 88 samples in the testing dataset
Each sample has 34 features
```

In [11]:

```
from sklearn.neighbors import KNeighborsClassifier

estimator = KNeighborsClassifier()
```

In [12]:

```
estimator.fit(X_train, y_train)
```

Out[12]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_neighbors=5, p=2, weights='uniform')
```

In [13]:

In [13]:

```
y_predicted = estimator.predict(X_test)
accuracy = np.mean(y_test == y_predicted) * 100
print("The accuracy is {0:.1f}%".format(accuracy))
```

```
The accuracy is 86.4%
```

In [14]:

```
from sklearn.cross_validation import cross_val_score
```

In [15]:

```
scores = cross_val_score(estimator, X, y, scoring='accuracy')
average_accuracy = np.mean(scores) * 100
print("The average accuracy is {0:.1f}%".format(average_accuracy))
```

```
The average accuracy is 82.3%
```

In [21]:

```
avg_scores = []
all_scores = []
parameter_values = list(range(1, 21))  # Including 20
for n_neighbors in parameter_values:
    estimator = KNeighborsClassifier(n_neighbors=n_neighbors)
    scores = cross_val_score(estimator, X, y, scoring='accuracy')
    avg_scores.append(np.mean(scores))
    all_scores.append(scores)
```

In [24]:

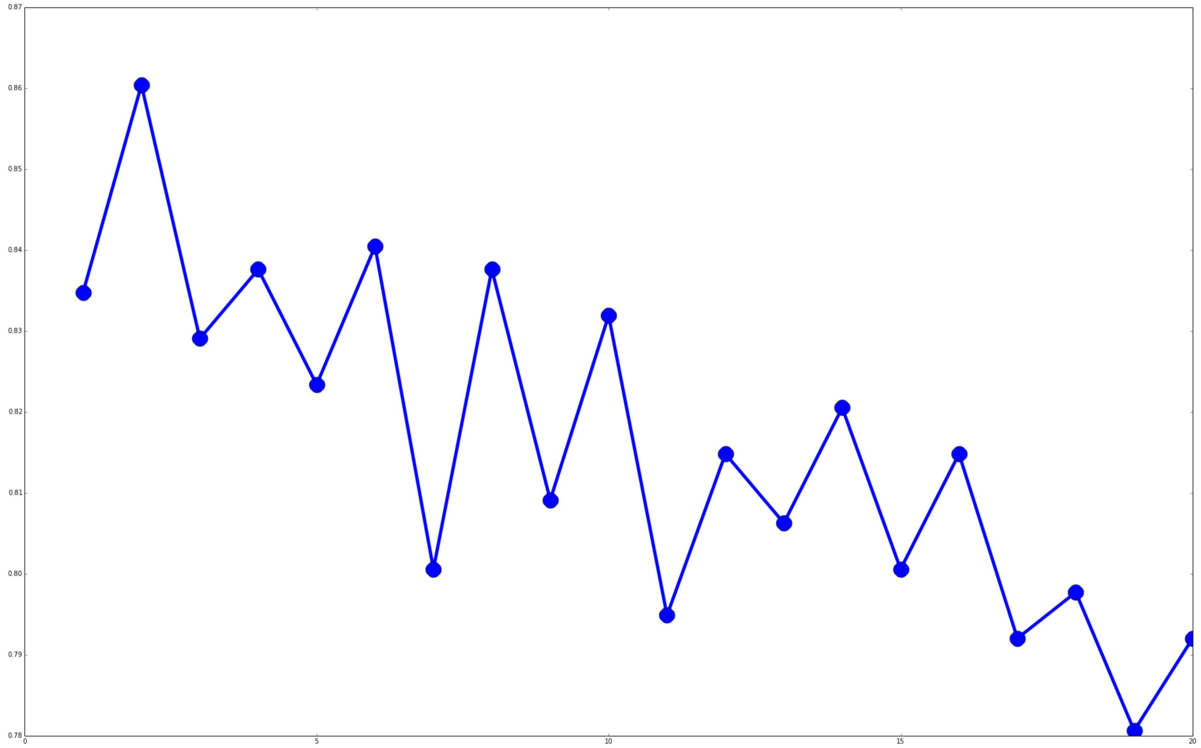
```
plt.plot?
```

In [26]:

```
from matplotlib import pyplot as plt
plt.figure(figsize=(32,20))
plt.plot(parameter_values, avg_scores, '-o', linewidth=5, markersize=24)
#plt.axis([0, max(parameter_values), 0, 1.0])
```

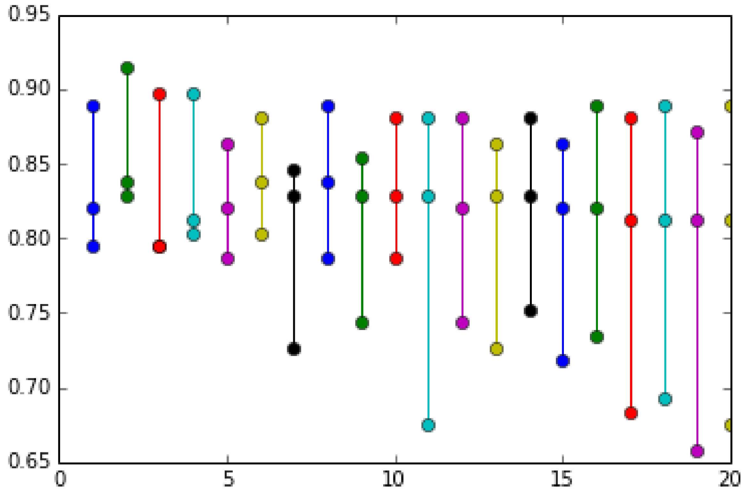
Out[26]:

```
[<matplotlib.lines.Line2D at 0x7f58cf956ac8>]
```



In [18]:

```
for parameter, scores in zip(parameter_values, all_scores):
    n_scores = len(scores)
    plt.plot([parameter] * n_scores, scores, '-o')
```

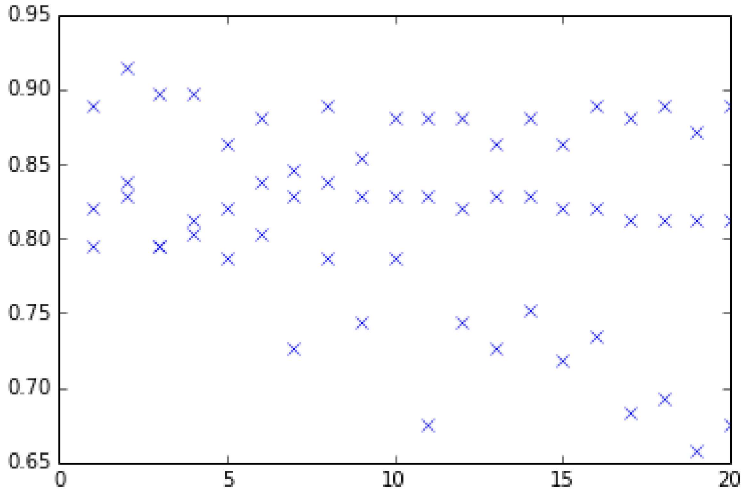


In [19]:

```
plt.plot(parameter_values, all_scores, 'bx')
```

Out[19]:

```
[<matplotlib.lines.Line2D at 0x7f58cfb47b38>,
 <matplotlib.lines.Line2D at 0x7f58cfb47eb8>,
 <matplotlib.lines.Line2D at 0x7f58cfb4b080>]
```



In [20]:

```
from collections import defaultdict
all_scores = defaultdict(list)
parameter_values = list(range(1, 21))  # Including 20
for n_neighbors in parameter_values:
    for i in range(100):
        estimator = KNeighborsClassifier(n_neighbors=n_neighbors)
        scores = cross_val_score(estimator, X, y, scoring='accuracy', cv=10)
        all_scores[n_neighbors].append(scores)
for parameter in parameter_values:
    scores = all_scores[parameter]
    n_scores = len(scores)
    plt.plot([parameter] * n_scores, scores, '-o')
```

```
KeyboardInterrupt                          Traceback (most recent call last)
<ipython-input-20-53d2156a8105> in <module>()
      5     for i in range(100):
      6         estimator = KNeighborsClassifier(n_neighbors=n_neighbors)
----> 7         scores = cross_val_score(estimator, X, y, scoring='accuracy', cv=1
0)
      8         all_scores[n_neighbors].append(scores)
      9 for parameter in parameter_values:

/usr/local/lib/python3.4/dist-packages/sklearn/cross_validation.py in cross_val_sc
ore(estimator, X, y, scoring, cv, n_jobs, verbose, fit_params, score_func, pre_dis
patch)
   1149                                               train, test, verbose, None,
   1150                                               fit_params)
-> 1151                       for train, test in cv)
   1152     return np.array(scores)[:, 0]
   1153 

/usr/local/lib/python3.4/dist-packages/sklearn/externals/joblib/parallel.py in __c
all__(self, iterable)
    650                 os.environ[JOBLIB_SPAWNED_PROCESS] = '1'
    651             self._iterating = True
--> 652             for function, args, kwargs in iterable:
    653                 self.dispatch(function, args, kwargs)
    654 

/usr/local/lib/python3.4/dist-packages/sklearn/cross_validation.py in <genexpr>(.
0)
   1149                                               train, test, verbose, None,
   1150                                               fit_params)
-> 1151                       for train, test in cv)
   1152     return np.array(scores)[:, 0]
   1153 

/usr/local/lib/python3.4/dist-packages/sklearn/externals/joblib/parallel.py in del
ayed(function, check_pickle)
    118     # using with multiprocessing:
    119     if check_pickle:
--> 120         pickle.dumps(function)
    121 
    122     def delayed_function(*args, **kwargs):

KeyboardInterrupt: 
```

In [ ]:

```
plt.plot(parameter_values, avg_scores, '-o')
```

In [ ]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [ ]:

```
```