Support Vector Machines project I - Cancer Data set

October 27, 2016

1 Support Vector Machines with Python

1.1 Import Libraries

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

1.2 Get the Data

```
In [52]: from sklearn.datasets import load_breast_cancer
```

```
In [54]: cancer = load_breast_cancer()
```

The data set is presented in a dictionary form:

```
In [55]: cancer.keys()
```

```
Out[55]: dict_keys(['DESCR', 'target', 'data', 'target_names', 'feature_names'])
```

```
In [56]: cancer['feature_names']
```

```
Out[56]: array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
                'mean smoothness', 'mean compactness', 'mean concavity',
                'mean concave points', 'mean symmetry', 'mean fractal dimension',
                'radius error', 'texture error', 'perimeter error', 'area error',
                'smoothness error', 'compactness error', 'concavity error',
                'concave points error', 'symmetry error', 'fractal dimension error',
                'worst radius', 'worst texture', 'worst perimeter', 'worst area',
                'worst smoothness', 'worst compactness', 'worst concavity',
                'worst concave points', 'worst symmetry', 'worst fractal dimension'],
               dtype='<U23')
```

1.3 Set up DataFrame

```
In [12]: df_feat = pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
         df_feat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
mean radius                569 non-null float64
mean texture               569 non-null float64
mean perimeter             569 non-null float64
mean area                  569 non-null float64
mean smoothness            569 non-null float64
mean compactness           569 non-null float64
mean concavity             569 non-null float64
mean concave points        569 non-null float64
mean symmetry              569 non-null float64
mean fractal dimension     569 non-null float64
radius error               569 non-null float64
texture error              569 non-null float64
perimeter error            569 non-null float64
area error                 569 non-null float64
smoothness error           569 non-null float64
compactness error          569 non-null float64
concavity error            569 non-null float64
concave points error       569 non-null float64
symmetry error             569 non-null float64
fractal dimension error    569 non-null float64
worst radius               569 non-null float64
worst texture              569 non-null float64
worst perimeter            569 non-null float64
worst area                 569 non-null float64
worst smoothness           569 non-null float64
worst compactness          569 non-null float64
worst concavity            569 non-null float64
worst concave points       569 non-null float64
worst symmetry             569 non-null float64
worst fractal dimension    569 non-null float64
dtypes: float64(30)
memory usage: 133.4 KB
```

```
In [14]: cancer['target']
```

```
Out[14]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
                1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
                1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
                0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
                0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
                0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
                0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
                0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
                1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
                1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
                1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
                1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
                0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
                1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1,
                1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
                1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

```
In [16]: df_target = pd.DataFrame(cancer['target'],columns=['Cancer'])
```

Now let's actually check out the dataframe!

```
In [8]: df.head()
```

```
Out[8]:    mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
        0        17.99         10.38          122.80     1001.0          0.11840
        1        20.57         17.77          132.90     1326.0          0.08474
        2        19.69         21.25          130.00     1203.0          0.10960
        3        11.42         20.38           77.58      386.1          0.14250
        4        20.29         14.34          135.10     1297.0          0.10030

           mean compactness  mean concavity  mean concave points  mean symmetry  \
        0           0.27760          0.3001              0.14710         0.2419
        1           0.07864          0.0869              0.07017         0.1812
        2           0.15990          0.1974              0.12790         0.2069
        3           0.28390          0.2414              0.10520         0.2597
        4           0.13280          0.1980              0.10430         0.1809

           mean fractal dimension           ...             worst radius  \
        0                 0.07871           ...                    25.38
        1                 0.05667           ...                    24.99
        2                 0.05999           ...                    23.57
        3                 0.09744           ...                    14.91
        4                 0.05883           ...                    22.54

           worst texture  worst perimeter  worst area  worst smoothness  \
        0          17.33           184.60      2019.0            0.1622
        1          23.41           158.80      1956.0            0.1238
        2          25.53           152.50      1709.0            0.1444
        3          26.50            98.87       567.7            0.2098
        4          16.67           152.20      1575.0            0.1374

           worst compactness  worst concavity  worst concave points  worst symmetry  \
        0             0.6656           0.7119                0.2654          0.4601
        1             0.1866           0.2416                0.1860          0.2750
        2             0.4245           0.4504                0.2430          0.3613
        3             0.8663           0.6869                0.2575          0.6638
        4             0.2050           0.4000                0.1625          0.2364

           worst fractal dimension
        0                  0.11890
        1                  0.08902
        2                  0.08758
        3                  0.17300
        4                  0.07678

        [5 rows x 30 columns]
```

1.4 Train Test Split

```
In [57]: from sklearn.cross_validation import train_test_split
```

```
In [58]: X_train, X_test, y_train, y_test = train_test_split(df_feat, np.ravel(df_target), test_size=0.
```

2 Train the Support Vector Classifier

```
In [59]: from sklearn.svm import SVC
```

```
In [60]: model = SVC()
```

```
In [61]: model.fit(X_train,y_train)
```

```
Out[61]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)
```

2.1 Predictions and Evaluations

```
In [27]: predictions = model.predict(X_test)
```

```
In [45]: from sklearn.metrics import classification_report,confusion_matrix
```

```
In [46]: print(confusion_matrix(y_test,predictions))
```

```
[[  0  66]
 [  0 105]]
```

```
In [62]: print(classification_report(y_test,predictions))
```

```
precision    recall  f1-score   support

          0       0.00      0.00      0.00        66
          1       0.61      1.00      0.76       105

avg / total       0.38      0.61      0.47       171
```

```
/Users/marci/anaconda/lib/python3.5/site-packages/sklearn/metrics/classification.py:1074: UndefinedMetr
  'precision', 'predicted', average, warn_for)
```

We can search for parameters using a GridSearch!

3 Gridsearch

```
In [63]: param_grid = {'C': [0.1,1, 10, 100, 1000], 'gamma': [1,0.1,0.01,0.001,0.0001], 'kernel': ['rbf
```

```
In [64]: from sklearn.grid_search import GridSearchCV
```

```
In [65]: grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=3)
```

```
In [40]: grid.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 25 candidates, totalling 75 fits
[CV] gamma=1, C=0.1, kernel=rbf ...
[CV] ... gamma=1, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=0.1, kernel=rbf ...
[CV] ... gamma=1, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=0.1, kernel=rbf ...
[CV] ... gamma=1, C=0.1, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.1, C=0.1, kernel=rbf ...
[CV] ... gamma=0.1, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=0.1, kernel=rbf ...
[CV] ... gamma=0.1, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=0.1, kernel=rbf ...
[CV] ... gamma=0.1, C=0.1, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.01, C=0.1, kernel=rbf ...
[CV] ... gamma=0.01, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=0.1, kernel=rbf ...
[CV] ... gamma=0.01, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=0.1, kernel=rbf ...
[CV] ... gamma=0.01, C=0.1, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.001, C=0.1, kernel=rbf ...
[CV] ... gamma=0.001, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.001, C=0.1, kernel=rbf ...
[CV] ... gamma=0.001, C=0.1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.001, C=0.1, kernel=rbf ...
[CV] ... gamma=0.001, C=0.1, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.0001, C=0.1, kernel=rbf ...
[CV] ... gamma=0.0001, C=0.1, kernel=rbf, score=0.902256 -   0.0s
[CV] gamma=0.0001, C=0.1, kernel=rbf ...
[CV] ... gamma=0.0001, C=0.1, kernel=rbf, score=0.962406 -   0.0s
[CV] gamma=0.0001, C=0.1, kernel=rbf ...
[CV] ... gamma=0.0001, C=0.1, kernel=rbf, score=0.916667 -   0.0s
[CV] gamma=1, C=1, kernel=rbf ...
[CV] ... gamma=1, C=1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=1, kernel=rbf ...
[CV] ... gamma=1, C=1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=1, kernel=rbf ...
[CV] ... gamma=1, C=1, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.1, C=1, kernel=rbf ...
[CV] ... gamma=0.1, C=1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=1, kernel=rbf ...
[CV] ... gamma=0.1, C=1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=1, kernel=rbf ...
[CV] ... gamma=0.1, C=1, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.01, C=1, kernel=rbf ...
[CV] ... gamma=0.01, C=1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=1, kernel=rbf ...
[CV] ... gamma=0.01, C=1, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=1, kernel=rbf ...
[CV] ... gamma=0.01, C=1, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.001, C=1, kernel=rbf ...
[CV] ... gamma=0.001, C=1, kernel=rbf, score=0.902256 -   0.0s
[CV] gamma=0.001, C=1, kernel=rbf ...
[CV] ... gamma=0.001, C=1, kernel=rbf, score=0.939850 -   0.0s
[CV] gamma=0.001, C=1, kernel=rbf ...
```

```
[CV] ... gamma=0.001, C=1, kernel=rbf, score=0.954545 -   0.0s
[CV] gamma=0.0001, C=1, kernel=rbf ...
[CV] ... gamma=0.0001, C=1, kernel=rbf, score=0.939850 -   0.0s
[CV] gamma=0.0001, C=1, kernel=rbf ...
[CV] ... gamma=0.0001, C=1, kernel=rbf, score=0.969925 -   0.0s
[CV] gamma=0.0001, C=1, kernel=rbf ...
[CV] ... gamma=0.0001, C=1, kernel=rbf, score=0.946970 -   0.0s
[CV] gamma=1, C=10, kernel=rbf ...
[CV] ... gamma=1, C=10, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=10, kernel=rbf ...
[CV] ... gamma=1, C=10, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=10, kernel=rbf ...
[CV] ... gamma=1, C=10, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.1, C=10, kernel=rbf ...
[CV] ... gamma=0.1, C=10, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=10, kernel=rbf ...
[CV] ... gamma=0.1, C=10, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=10, kernel=rbf ...
[CV] ... gamma=0.1, C=10, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.01, C=10, kernel=rbf ...
[CV] ... gamma=0.01, C=10, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=10, kernel=rbf ...
[CV] ... gamma=0.01, C=10, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=10, kernel=rbf ...
[CV] ... gamma=0.01, C=10, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.001, C=10, kernel=rbf ...
[CV] ... gamma=0.001, C=10, kernel=rbf, score=0.894737 -   0.0s
[CV] gamma=0.001, C=10, kernel=rbf ...
[CV] ... gamma=0.001, C=10, kernel=rbf, score=0.932331 -   0.0s
[CV] gamma=0.001, C=10, kernel=rbf ...
[CV] ... gamma=0.001, C=10, kernel=rbf, score=0.916667 -   0.0s
[CV] gamma=0.0001, C=10, kernel=rbf ...
[CV] ... gamma=0.0001, C=10, kernel=rbf, score=0.932331 -   0.0s
[CV] gamma=0.0001, C=10, kernel=rbf ...
[CV] ... gamma=0.0001, C=10, kernel=rbf, score=0.969925 -   0.0s
[CV] gamma=0.0001, C=10, kernel=rbf ...
[CV] ... gamma=0.0001, C=10, kernel=rbf, score=0.962121 -   0.0s
[CV] gamma=1, C=100, kernel=rbf ...
[CV] ... gamma=1, C=100, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=100, kernel=rbf ...
[CV] ... gamma=1, C=100, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=100, kernel=rbf ...
[CV] ... gamma=1, C=100, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.1, C=100, kernel=rbf ...
[CV] ... gamma=0.1, C=100, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=100, kernel=rbf ...
[CV] ... gamma=0.1, C=100, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=100, kernel=rbf ...
[CV] ... gamma=0.1, C=100, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.01, C=100, kernel=rbf ...
[CV] ... gamma=0.01, C=100, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=100, kernel=rbf ...
[CV] ... gamma=0.01, C=100, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=100, kernel=rbf ...
```

```
[CV] ... gamma=0.01, C=100, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.001, C=100, kernel=rbf ...
[CV] ... gamma=0.001, C=100, kernel=rbf, score=0.894737 -   0.0s
[CV] gamma=0.001, C=100, kernel=rbf ...
[CV] ... gamma=0.001, C=100, kernel=rbf, score=0.932331 -   0.0s
[CV] gamma=0.001, C=100, kernel=rbf ...
[CV] ... gamma=0.001, C=100, kernel=rbf, score=0.916667 -   0.0s
[CV] gamma=0.0001, C=100, kernel=rbf ...
[CV] ... gamma=0.0001, C=100, kernel=rbf, score=0.917293 -   0.0s
[CV] gamma=0.0001, C=100, kernel=rbf ...
[CV] ... gamma=0.0001, C=100, kernel=rbf, score=0.977444 -   0.0s
[CV] gamma=0.0001, C=100, kernel=rbf ...
[CV] ... gamma=0.0001, C=100, kernel=rbf, score=0.939394 -   0.0s
[CV] gamma=1, C=1000, kernel=rbf ...
[CV] ... gamma=1, C=1000, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=1000, kernel=rbf ...
[CV] ... gamma=1, C=1000, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=1, C=1000, kernel=rbf ...
[CV] ... gamma=1, C=1000, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.1, C=1000, kernel=rbf ...
[CV] ... gamma=0.1, C=1000, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=1000, kernel=rbf ...
[CV] ... gamma=0.1, C=1000, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.1, C=1000, kernel=rbf ...
[CV] ... gamma=0.1, C=1000, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.01, C=1000, kernel=rbf ...
[CV] ... gamma=0.01, C=1000, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=1000, kernel=rbf ...
[CV] ... gamma=0.01, C=1000, kernel=rbf, score=0.631579 -   0.0s
[CV] gamma=0.01, C=1000, kernel=rbf ...
[CV] ... gamma=0.01, C=1000, kernel=rbf, score=0.636364 -   0.0s
[CV] gamma=0.001, C=1000, kernel=rbf ...
[CV] ... gamma=0.001, C=1000, kernel=rbf, score=0.894737 -   0.0s
[CV] gamma=0.001, C=1000, kernel=rbf ...
[CV] ... gamma=0.001, C=1000, kernel=rbf, score=0.932331 -   0.0s
[CV] gamma=0.001, C=1000, kernel=rbf ...
[CV] ... gamma=0.001, C=1000, kernel=rbf, score=0.916667 -   0.0s
```

```
[Parallel(n_jobs=1)]: Done  31 tasks       | elapsed:    0.3s
[Parallel(n_jobs=1)]: Done  75 out of  75 | elapsed:    0.8s finished
```

```
[CV] gamma=0.0001, C=1000, kernel=rbf ...
[CV] ... gamma=0.0001, C=1000, kernel=rbf, score=0.909774 -   0.0s
[CV] gamma=0.0001, C=1000, kernel=rbf ...
[CV] ... gamma=0.0001, C=1000, kernel=rbf, score=0.969925 -   0.0s
[CV] gamma=0.0001, C=1000, kernel=rbf ...
[CV] ... gamma=0.0001, C=1000, kernel=rbf, score=0.931818 -   0.0s
```

```
Out[40]: GridSearchCV(cv=None, error_score='raise',
                estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False),
                fit_params={}, iid=True, n_jobs=1,
                param_grid={'gamma': [1, 0.1, 0.01, 0.001, 0.0001], 'C': [0.1, 1, 10, 100, 1000], 'kerne
                pre_dispatch='2*n_jobs', refit=True, scoring=None, verbose=3)
```

Inspect the best parameters found by GridSearchCV in the best_params_ attribute, and the best estimator in the best_estimator_ attribute:

```
In [41]: grid.best_params_
```

```
Out[41]: {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
```

```
In [ ]: grid.best_estimator_
```

Then you can re-run predictions on this grid object just like you would with a normal model.

```
In [48]: grid_predictions = grid.predict(X_test)
```

```
In [49]: print(confusion_matrix(y_test,grid_predictions))
```

```
[[ 60   6]
 [  3 102]]
```

```
In [50]: print(classification_report(y_test,grid_predictions))
```

```
precision    recall  f1-score   support

          0       0.95      0.91      0.93        66
          1       0.94      0.97      0.96       105

avg / total       0.95      0.95      0.95       171
```