DUT STID, Université de la Côte d'Azur

Data mining

Classification avec SVM

Prof. Dario Malchiodi

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

UNIVERSITÉ CÔTE D'AZUR

In [1]:

```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
#details at https://archive.ics.uci.edu/ml/datasets/banknote+authentication

banknotes = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/
                         '00267/data_banknote_authentication.txt',
                         names=['Variance', 'Skewness', 'Curtosis', 'Entropy', 'Clas
                         header=None)
```

In [3]:

```
banknotes.head()
```

Out[3]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

In [4]:

```
banknotes.shape
```

Out[4]:

```
(1372, 5)
```

In [59]:

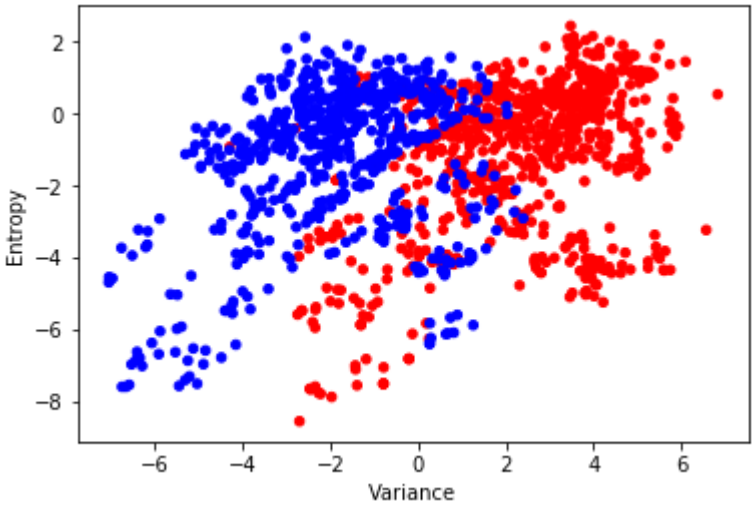
```
X_banknotes_2d = banknotes[['Variance', 'Entropy']]
y_banknotes = banknotes['Class'].map(str)
```

In [62]:

```
color = ['r' if y=='0' else 'b' for y in y_banknotes]
```

In [63]:

```
X_banknotes_2d.plot.scatter('Variance', 'Entropy', color=color)
plt.show()
```



In [64]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_banknotes_2d, y_banknotes)
```

Out[64]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linea
r',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [65]:

```
svclassifier.coef_, svclassifier.intercept_
```

Out[65]:

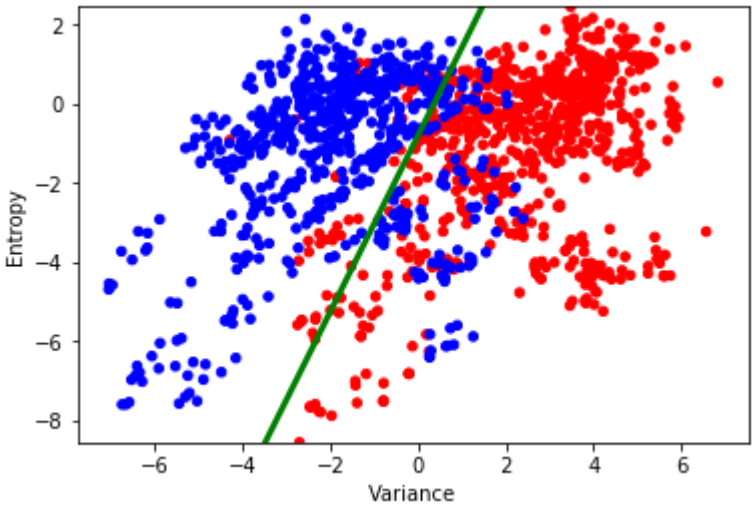
```
(array([[-0.75753259,  0.34043408]]), array([0.27382843]))
```

In [66]:

```
wx, wy = svclassifier.coef_[0]
w0 = svclassifier.intercept_

def decision_line(x):
    return (-w0 -wx*x)/wy
```

In [67]:

```
X_banknotes_2d.plot.scatter('Variance', 'Entropy', color=color)

xmin = X_banknotes_2d['Variance'].min()
xmax = X_banknotes_2d['Variance'].max()
plt.plot([xmin, xmax], [decision_line(xmin), decision_line(xmax)], c='g', linewidth

plt.ylim((X_banknotes_2d['Entropy'].min(), X_banknotes_2d['Entropy'].max()))
plt.show()
```



In [68]:

```
X_banknotes = banknotes.drop('Class', axis=1)
```

In [69]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_banknotes, y_banknotes, test_
```

In [70]:

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

Out[70]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linea
r',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [71]:

```
y_pred = svclassifier.predict(X_test)
```

In [72]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
```

```
[[148   3]
 [  0 124]]
```

In [73]:

```
print(classification_report(y_test,y_pred))
```

```
             precision    recall  f1-score   support

          0       1.00      0.98      0.99       151
          1       0.98      1.00      0.99       124

avg / total       0.99      0.99      0.99       275
```

In [74]:

```
from sklearn.metrics import accuracy_score

def svc_experiment(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
    svclassifier = SVC(kernel='linear')
    svclassifier.fit(X_train, y_train)
    y_pred = svclassifier.predict(X_test)
    return accuracy_score(y_test, y_pred)
```

In [75]:

```
svc_experiment(X_banknotes, y_banknotes)
```

Out[75]:

```
0.9709090909090909
```

In [76]:

```
np.mean([svc_experiment(X_banknotes, y_banknotes) for _ in range(10)])
```

Out[76]:

```
0.9905454545454544
```

In [77]:

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

# Assign colum names to the dataset
colnames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

# Read dataset to pandas dataframe
irisdata = pd.read_csv(url, names=colnames)
```

In [78]:

```
X_iris = irisdata.drop('Class', axis=1)
y_iris = irisdata['Class']
```

In [23]:

```
results = pd.DataFrame([np.mean([svc_experiment(X_iris, y_iris) for _ in range(10)]
                        index=['Linear SVC (no model selection)'],
                        columns=['Average accuracy'])
results
```

Out[23]:

	Average accuracy
Linear SVC (no model selection)	0.97

In [24]:

```
from sklearn.svm import SVC
X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size = 0.2
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(X_train, y_train)
```

Out[24]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=8, gamma='auto', kernel='pol
y',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [25]:

```
y_pred = svclassifier.predict(X_test)
```

In [26]:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[ 5  0  0]
 [ 0  9  2]
 [ 0  0 14]]
```

In [27]:

```
print(accuracy_score(y_test, y_pred))
```

```
0.9333333333333333
```

In [28]:

```
def svc_poly_experiment(X, y, deg):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
    svclassifier = SVC(kernel='poly', degree=deg)
    svclassifier.fit(X_train, y_train)
    y_pred = svclassifier.predict(X_test)
    return accuracy_score(y_test, y_pred)

def svc_poly_avg_accuracy(deg):
    return np.mean([svc_poly_experiment(X_iris, y_iris, deg) for _ in range(10)])
```

In [29]:

```
poly_res = [(d, svc_poly_avg_accuracy(d)) for d in range(1, 9)]
```

In [30]:

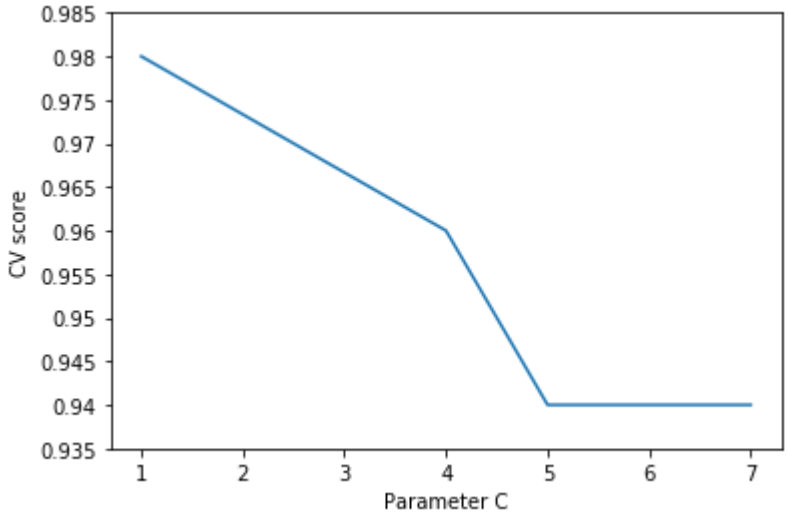
```
pd.DataFrame(poly_res, columns=['Degree', 'Avg. accuracy'])
```

Out[30]:

	Degree	Avg. accuracy
0	1	0.980000
1	2	0.953333
2	3	0.950000
3	4	0.966667
4	5	0.940000
5	6	0.956667
6	7	0.946667
7	8	0.960000

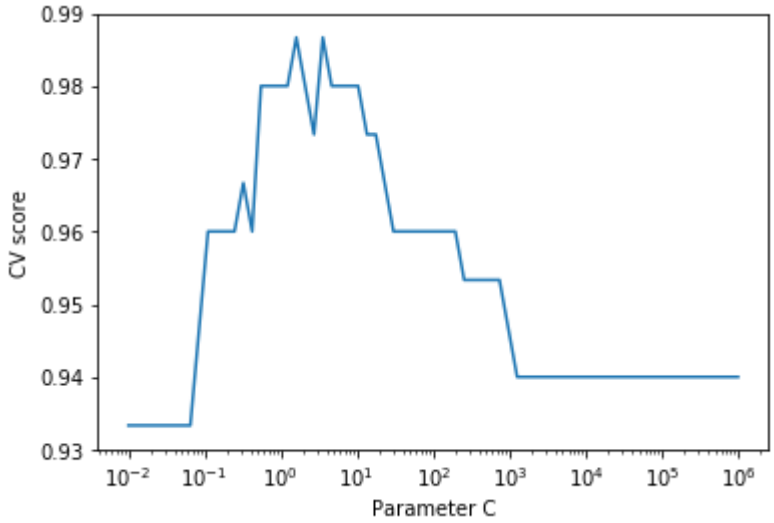
In [31]:

```
from sklearn.model_selection import cross_val_score

svc = SVC(kernel='poly')
deg_values = range(1, 8)

scores = []
for d in deg_values:
    svc.degree = d
    this_scores = cross_val_score(svc, X_iris, y_iris, cv=5, n_jobs=1)
    scores.append(np.mean(this_scores))


plt.plot(deg_values, scores)

locs, labels = plt.yticks()
plt.yticks(locs, list(map(lambda x: "%g" % x, locs)))
plt.ylabel('CV score')
plt.xlabel('Degree')
plt.show()
```



In [32]:

```
svc = SVC()
c_values = np.logspace(-2, 6, 70)

scores = []
for c in c_values:
    svc.C = c
    this_scores = cross_val_score(svc, X_iris, y_iris, cv=5, n_jobs=1)
    scores.append(np.mean(this_scores))


plt.semilogx(c_values, scores)

locs, labels = plt.yticks()
plt.yticks(locs, list(map(lambda x: "%g" % x, locs)))
plt.ylabel('CV score')
plt.xlabel('Parameter C')
plt.show()
```



In [33]:

```
from sklearn.model_selection import GridSearchCV, cross_val_score

X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size = 0.2

Cs = np.logspace(-6, -1, 10)
clf = GridSearchCV(estimator=SVC(), param_grid=dict(C=Cs),
                   n_jobs=-1)
clf.fit(X_train, y_train)

print(clf.best_score_)

print(clf.best_estimator_.C)

clf.score(X_test, y_test)
```

```
0.925
0.1
```

Out[33]:

```
0.9
```

In [34]:

```
def svc_linear_cv_experiment():

    X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size =

    c_values = np.logspace(-6, -1, 10)
    clf = GridSearchCV(estimator=SVC(), param_grid=dict(C=c_values),
                   n_jobs=-1)
    clf.fit(X_train, y_train)

    return clf.score(X_test, y_test)
```

In [35]:

```
new_row = pd.DataFrame(np.mean([svc_linear_cv_experiment() for _ in range(10)]),
                       index=['Linear SVC (model selection on C)'],
                       columns=['Average accuracy'])

results = pd.concat((results, new_row))
results
```

Out[35]:

	Average accuracy
Linear SVC (no model selection)	0.970000
Linear SVC (model selection on C)	0.953333

In [36]:

```
X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size = 0.2

c_values = np.logspace(-6, -1, 10)
deg_values = range(1, 8)
clf = GridSearchCV(estimator=SVC(kernel='poly'), param_grid=dict(C=c_values, degree
                   n_jobs=-1)
clf.fit(X_train, y_train)

print('best score: {}'.format(clf.best_score_))

print('best C={}'.format(clf.best_estimator_.C))

print('Test score: {}'.format(clf.score(X_test, y_test)))
```

```
best score: 0.9833333333333333
best C=1.2915496650148827e-05
Test score: 0.9
```

In [37]:

```
def svc_poly_cv_experiment():

    X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size =

    c_values = np.logspace(-6, -1, 10)
    deg_values = range(1, 8)
    clf = GridSearchCV(estimator=SVC(kernel='poly'), param_grid=dict(C=c_values, de
                       n_jobs=-1)
    clf.fit(X_train, y_train)

    return clf.score(X_test, y_test)
```

In [38]:

```
svc_poly_cv_experiment()
```

Out[38]:

```
0.9666666666666667
```

In [39]:

```
new_row = pd.DataFrame(np.mean(np.mean([svc_poly_cv_experiment() for _ in range(10)
                       index=['Polynomial SVC (model selection on C and deg)'],
                       columns=['Average accuracy'])

results = pd.concat((results, new_row))
results
```

Out[39]:

	Average accuracy
Linear SVC (no model selection)	0.970000
Linear SVC (model selection on C)	0.953333
Polynomial SVC (model selection on C and deg)	0.943333

In [1]:

```
def svc_rbf_cv_experiment():

    X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size =

    c_values = np.logspace(-6, 2, 10)
    gamma_values = np.logspace(10**-3, 300, 10)
    clf = GridSearchCV(estimator=SVC(kernel='rbf'), param_grid=dict(C=c_values, gam
                       n_jobs=-1)
    clf.fit(X_train, y_train)

    return clf.score(X_test, y_test)
```

In [2]:

```
svc_rbf_cv_experiment()
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-2-a9668731442e> in <module>()
----> 1 svc_rbf_cv_experiment()

<ipython-input-1-64e61e3085db> in svc_rbf_cv_experiment()
      1 def svc_rbf_cv_experiment():
      2
----> 3     X_train, X_test, y_train, y_test = train_test_split(X_iris
, y_iris, test_size = 0.20)
      4
      5     c_values = np.logspace(-6, 2, 10)

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

In [57]:

```
new_row = pd.DataFrame(np.mean([svc_rbf_cv_experiment() for _ in range(10)]),
                       index=['Gaussian SVC (model selection on gamma)'],
                       columns=['Average accuracy'])

results = pd.concat((results, new_row))
results
```

Out[57]:

	Average accuracy
Linear SVC (no model selection)	0.970000
Linear SVC (model selection on C)	0.953333
Polynomial SVC (model selection on C and deg)	0.943333
Gaussian SVC (model selection on gamma)	0.966667
Sigmoid SVC (model selection on gamma)	0.263333
Gaussian SVC (model selection on gamma)	0.956667

In [43]:

```
def svc_sigmoid_cv_experiment():

    X_train, X_test, y_train, y_test = train_test_split(X_iris, y_iris, test_size =

    c_values = np.logspace(-6, -1, 10)
    gamma_values = np.logspace(10**-3, 100, 10)
    clf = GridSearchCV(estimator=SVC(kernel='sigmoid'), param_grid=dict(C=c_values,
                       n_jobs=-1)
    clf.fit(X_train, y_train)

    return clf.score(X_test, y_test)
```

In [44]:

```
new_row = pd.DataFrame(np.mean([svc_sigmoid_cv_experiment() for _ in range(10)]),
                       index=['Sigmoid SVC (model selection on gamma)'],
                       columns=['Average accuracy'])

results = pd.concat((results, new_row))
results
```

Out[44]:

	Average accuracy
Linear SVC (no model selection)	0.970000
Linear SVC (model selection on C)	0.953333
Polynomial SVC (model selection on C and deg)	0.943333
Gaussian SVC (model selection on gamma)	0.966667
Sigmoid SVC (model selection on gamma)	0.263333

In [ ]:

```
```