1 HW: Machine Learning in Finance LAB

1.1 due 2023-01-29

- Yu-Ching Liao ycliao3@illinois.edu (mailto:ycliao3@illinois.edu)

2 Importing Package

In [1]:
```
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import numpy as np
```

executed in 1.73s, finished 09:39:04 2023-01-27

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import preprocessing
from sklearn.linear_model import SGDClassifier
from sklearn import metrics
```

executed in 226ms, finished 09:39:10 2023-01-27

3 Classification: Iris

In [347]:
```
iris = datasets.load_iris()
X_iris, y_iris = iris.data, iris.target

print(X_iris.shape, y_iris.shape)
print(X_iris[0], y_iris[0])
```

executed in 6ms, finished 14:34:55 2023-01-27

```
(150, 4) (150,)
[5.1 3.5 1.4 0.2] 0
```

In [348]:
```
X, y = X_iris[:, :2], y_iris
```

executed in 3ms, finished 14:34:56 2023-01-27

In [349]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=33)
print(X_train.shape, y_train.shape)
```

executed in 4ms, finished 14:34:57 2023-01-27

```
(112, 2) (112,)
```

In [342]:
```
X_train
```

executed in 7ms, finished 14:33:04 2023-01-27

Out[342]:
```
array([[5. , 2.3],
       [4.9, 3.1],
       [6.3, 2.3],
       [5.8, 2.6],
       [6.2, 2.9],
       [4.7, 3.2],
       [4.6, 3.4],
       [5.1, 2.5],
       [4.8, 3.4],
       [7.9, 3.8],
       [5.1, 3.4],
       [5.1, 3.7],
       [5.6, 2.9],
       [6.5, 3. ],
       [5.4, 3.9],
       [7. , 3.2],
       [5.8, 2.8],
       [7.7, 2.6],
       [5.5, 2.5],
       [5. , 2.2],
```

In [343]:

```
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

executed in 3ms, finished 14:33:12 2023-01-27

In [344]:

```
X_train
```

executed in 7ms, finished 14:33:14 2023-01-27

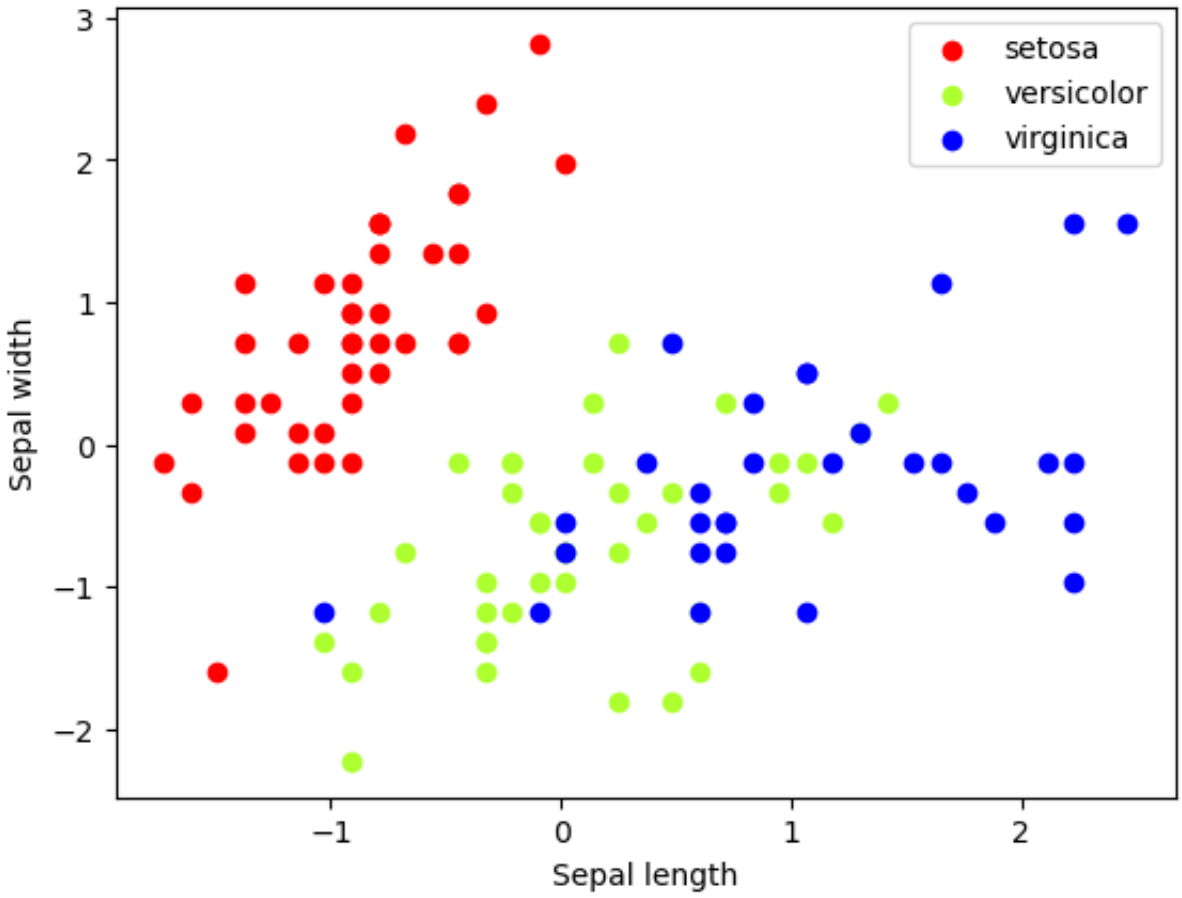
Out[344]:

```
array([[-0.91090798, -1.59775374],
       [-1.0271058 ,  0.08448757],
       [ 0.59966379, -1.59775374],
       [ 0.01867465, -0.96691325],
       [ 0.48346596, -0.33607276],
       [-1.25950146,  0.29476773],
       [-1.37569929,  0.71532806],
       [-0.79471015, -1.17719341],
       [-1.14330363,  0.71532806],
       [ 2.45882905,  1.55644871],
       [-0.79471015,  0.71532806],
       [-0.79471015,  1.34616854],
       [-0.21372101, -0.33607276],
       [ 0.83205945, -0.1257926 ],
       [-0.44611666,  1.76672887],
       [ 1.41304859,  0.29476773],
       [ 0.01867465, -0.54635292],
       [ 2.22643339, -0.96691325],
       [-0.32991883, -1.17719341],
       [ 0.13487248,  0.29476773]
```

In [171]:
```
colors = ['red', 'greenyellow', 'blue']
for i in range(len(colors)):
    xs = X_train[:, 0][y_train == i]
    ys = X_train[:, 1][y_train == i]
    plt.scatter(xs, ys, c =colors[i])
plt.legend(iris.target_names)
plt.xlabel("Sepal length")
plt.ylabel("Sepal width")
```

executed in 146ms, finished 11:08:04 2023-01-27

Out[171]: Text(0, 0.5, 'Sepal width')



In [8]:
```
clf = SGDClassifier()
clf.fit(X_train, y_train)
```

executed in 9ms, finished 09:39:58 2023-01-27

Out[8]: SGDClassifier()

In [9]:
```
print(clf.coef_)
print(clf.intercept_)
```

executed in 4ms, finished 09:40:06 2023-01-27

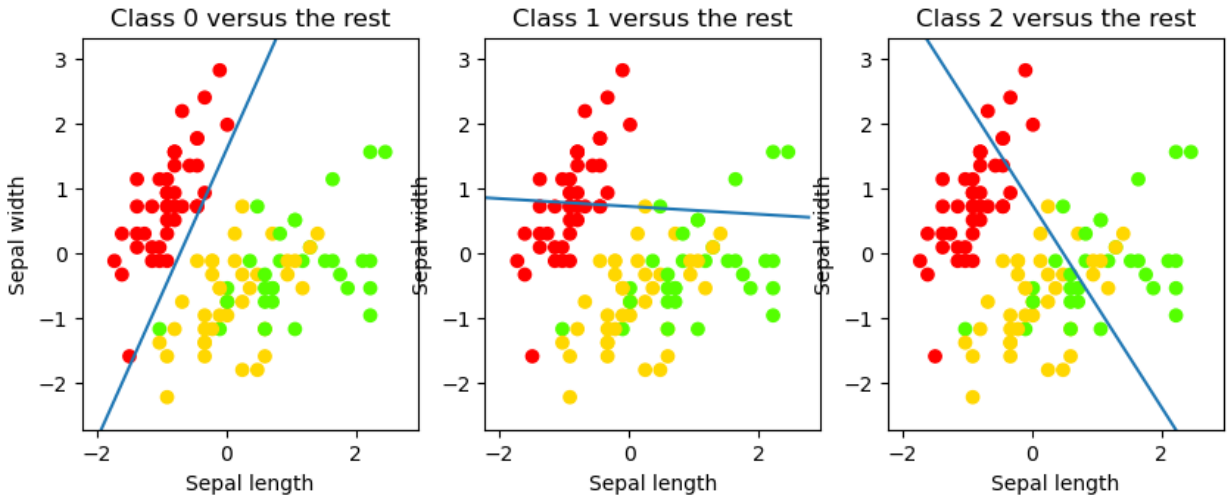
```
[[-25.68822452  11.49559386]
 [ -0.26152082  -4.32167831]
 [  5.72344877   3.65112163]]
[-18.15624742   3.09647045  -2.75692331]
```

In [10]:
```
x_min, x_max = X_train[:, 0].min() - .5, X_train[:, 0].max() + .5
y_min, y_max = X_train[:, 1].min() - .5, X_train[:, 1].max() + .5
```

executed in 4ms, finished 09:40:14 2023-01-27

In [11]:
```
xs = np.arange(x_min, x_max, 0.5)
fig, axes = plt.subplots(1, 3)
fig.set_size_inches(10, 6)
for i in [0, 1, 2]:
    axes[i].set_aspect('equal')
    axes[i].set_title('Class '+ str(i) + ' versus the rest')
    axes[i].set_xlabel('Sepal length')
    axes[i].set_ylabel('Sepal width')
    axes[i].set_xlim(x_min, x_max)
    axes[i].set_ylim(y_min, y_max)
    #error here need plt.
    plt.sca(axes[i])
    plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm
    ys = (-clf.intercept_[i] - xs * clf.coef_[i, 0]) / clf.coef_[i,
    plt.plot(xs, ys)
```

executed in 239ms, finished 09:40:21 2023-01-27



In [12]:
```
print( clf.predict(scaler.transform([[4.7, 3.1]])) )
print( clf.decision_function(scaler.transform([[4.7, 3.1]])) )
```

executed in 4ms, finished 09:40:30 2023-01-27

```
[0]
[[15.16934361  3.06072822 -9.65714102]]
```

In [14]:
```
y_train_pred = clf.predict(X_train)
print( metrics.accuracy_score(y_train, y_train_pred) )
y_pred = clf.predict(X_test)
print( metrics.accuracy_score(y_test, y_pred) )

print( metrics.classification_report(y_test, y_pred, target_names=ir
print( metrics.confusion_matrix(y_test, y_pred) )
```

executed in 10ms, finished 09:40:44 2023-01-27

```
0.8035714285714286
0.6052631578947368
              precision    recall  f1-score   support

      setosa       1.00      0.88      0.93         8
  versicolor       0.42      0.73      0.53        11
   virginica       0.67      0.42      0.52        19

    accuracy                           0.61        38
   macro avg       0.70      0.67      0.66        38
weighted avg       0.67      0.61      0.61        38

[[ 7  0  1]
 [ 0  8  3]
 [ 0 11  8]]
```

4 Classification: Tresasury

In [350]:
```
T = pd.read_csv('Treasury Squeeze raw score data.csv')
T = T.drop(['rowindex', 'contract'], axis=1)
```

executed in 7ms, finished 14:35:04 2023-01-27

In [351]:
```
l = list(T.columns)
X_l = l[:-1]
y_l = list(l[-1:])

X = T[X_l]
y = T[y_l]

for i in range(len(y)):
    if y['squeeze'][i] == True:
        y['squeeze'][i] = 1
    else:
        y['squeeze'][i] = 0
X = np.array(X)
y = np.array(y['squeeze'])
```

executed in 44ms, finished 14:35:05 2023-01-27

```
/var/folders/9z/csd85yv12f10212nnmpxfd0h0000gn/T/ipykernel_20534/4203
951657.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  y['squeeze'][i] = 1
/var/folders/9z/csd85yv12f10212nnmpxfd0h0000gn/T/ipykernel_20534/4203
951657.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  y['squeeze'][i] = 0
```

In [353]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=33)
print(X_train.shape, y_train.shape)
```

executed in 5ms, finished 14:35:07 2023-01-27

```
(675, 9) (675,)
```

In [355]:
```
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

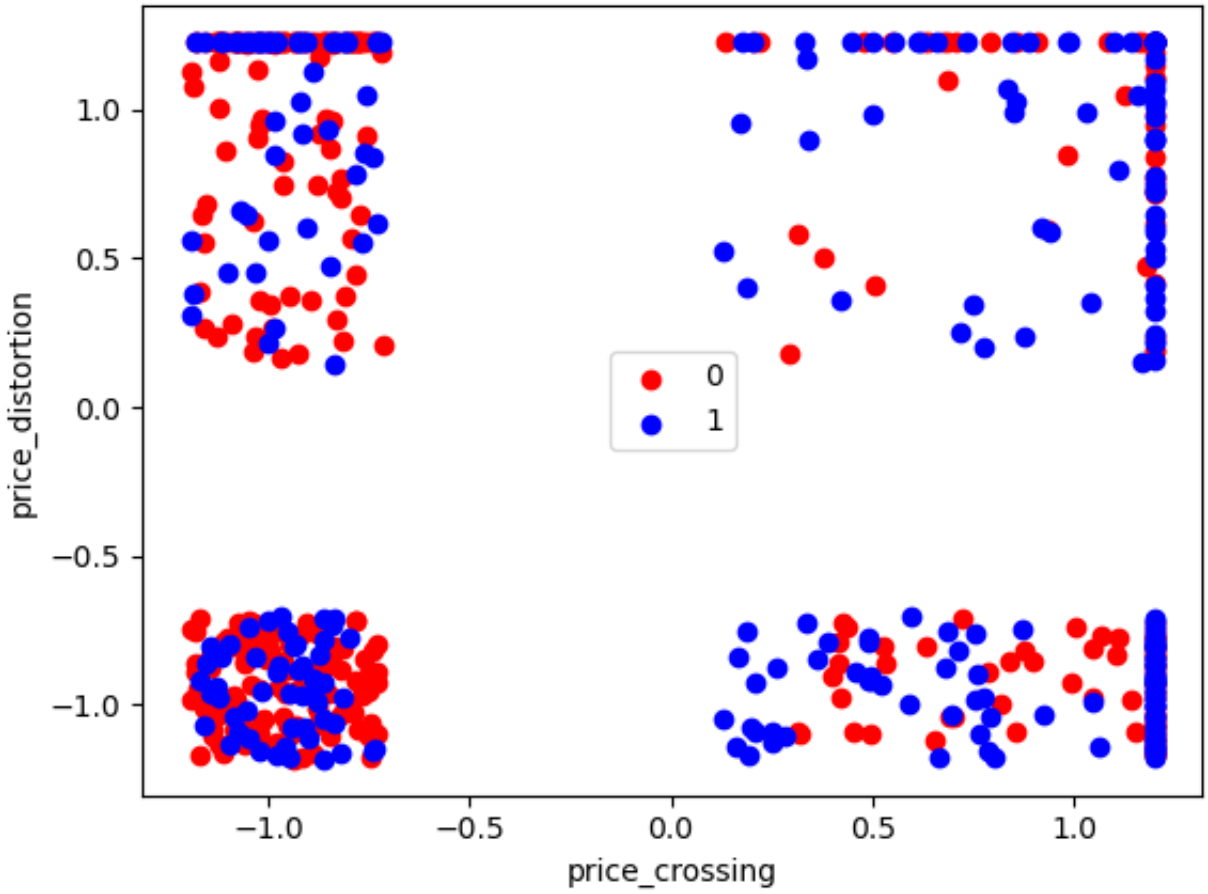
executed in 4ms, finished 14:35:30 2023-01-27

In [363]:
```
colors = ['red','blue']

for i in range(len(colors)):
    xs = X_train[:, 8][y_train == i]
    ys = X_train[:, 2][y_train == i]
    plt.scatter(xs, ys, c =colors[i], label = i)
plt.legend()
plt.xlabel(X_l[0])
plt.ylabel(X_l[1])
```

executed in 159ms, finished 15:11:44 2023-01-27

Out[363]: Text(0, 0.5, 'price_distortion')



In [362]:
```
y_train=y_train.astype('int')
clf = SGDClassifier()
clf.fit(X_train, y_train)

print(clf.coef_)
print(clf.intercept_)
```

executed in 6ms, finished 15:11:33 2023-01-27

```
[[ 0.43223878  1.1026829  -0.53984998 -0.0856848   0.60355118 -0.0271
3109
   1.3742536   0.17439454  1.80487549]]
[-1.33012375]
```

In [364]:
```
x_min, x_max = X_train[:, 0].min() - .5, X_train[:, 0].max() + .5
y_min, y_max = X_train[:, 1].min() - .5, X_train[:, 1].max() + .5
```

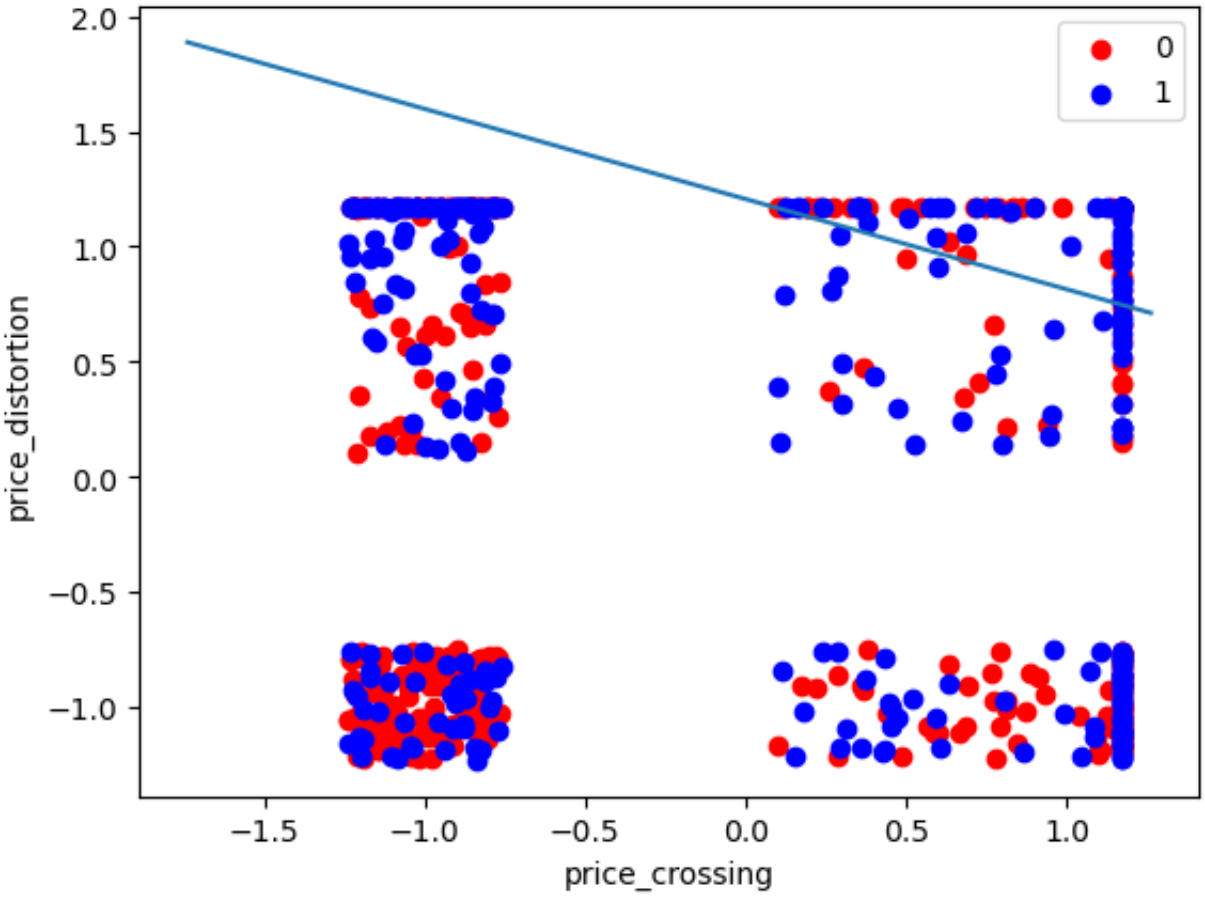
executed in 3ms, finished 15:11:52 2023-01-27

In [365]:
```
colors = ['red','blue']
for i in range(len(colors)):
    xs = X_train[:, 0][y_train == i]
    ys = X_train[:, 1][y_train == i]
    plt.scatter(xs, ys, c =colors[i], label = i)
plt.legend()
plt.xlabel(X_l[0])
plt.ylabel(X_l[1])


xs = np.arange(x_min, x_max, 0.5)
ys = (-clf.intercept_[0] - xs * clf.coef_[0, 0]) / clf.coef_[0, 1]
plt.plot(xs, ys)
```

executed in 149ms, finished 15:11:54 2023-01-27

Out[365]: [<matplotlib.lines.Line2D at 0x7fb0bdabc370>]



In [368]:
```
y_test=y_test.astype('int')

y_train_pred = clf.predict(X_train)
print( metrics.accuracy_score(y_train, y_train_pred) )
y_pred = clf.predict(X_test)
print( metrics.accuracy_score(y_test, y_pred) )

print( metrics.classification_report(y_test, y_pred) )
print( metrics.confusion_matrix(y_test, y_pred) )
```

executed in 11ms, finished 15:12:56 2023-01-27

```
0.6148148148148148
0.6977777777777778
              precision    recall  f1-score   support

           0       0.71      0.85      0.77       136
           1       0.67      0.46      0.55        89

    accuracy                           0.70       225
   macro avg       0.69      0.66      0.66       225
weighted avg       0.69      0.70      0.68       225

[[116  20]
 [ 48  41]]
```

5 Signing

In [15]:
```
print("My name is Yu-Ching Liao")
print("My NetID is: 656724372")
print("I hereby certify that I have read the University policy on Aca
```

executed in 3ms, finished 09:43:48 2023-01-27

```
My name is Yu-Ching Liao
My NetID is: 656724372
I hereby certify that I have read the University policy on Academic I
ntegrity and that I am not in violation.
```

In [ ]:
```

```

In [369]:
```
#print( clf.predict(scaler.transform([[4.7, 3.1]])) )
#print( clf.decision_function(scaler.transform([[4.7, 3.1]])) )
```

executed in 2ms, finished 15:17:54 2023-01-27

In [ ]: