In [75]:
```
comparison = pd.DataFrame({'prediction' : pred , 'Actual' : y_test})
comparison
```
executed in 11ms, finished 14:24:18 2023-11-06

Out[75]:

	prediction	Actual
26	SG	SG
86	SG	SG
2	C	C
55	SG	SG
75	C	C
93	C	C
16	C	C
73	SG	SG
54	SG	C
95	C	C
53	C	C
92	C	C
78	SG	SG
13	SG	SG
7	SG	SG
30	C	C
22	SG	SG
24	C	C
33	C	C
8	SG	SG

6 확인학습

- iris 붓꽃데이터 중 setosa와 versicolor만 선택하여 해당 데이터셋을 이용한 SVM 선형 분류

6.1 데이터셋 로딩

In [76]:
```
from sklearn.datasets import load_iris

iris = load_iris()
```
executed in 20ms, finished 14:27:47 2023-11-06

In [83]:
```
x = iris.data[iris.target != 2]
y = iris.target[iris.target != 2]
```
executed in 15ms, finished 14:31:57 2023-11-06

In [84]:
```
x.shape , y.shape
```
executed in 15ms, finished 14:32:01 2023-11-06

Out[84]: ((100, 4), (100,))

In [88]:
```
data = pd.DataFrame(x , columns = iris.feature_names)
```
executed in 6ms, finished 14:34:05 2023-11-06

In [90]:
```
data['target'] = y
```
executed in 17ms, finished 14:34:16 2023-11-06

In [91]:
```
data
```
executed in 22ms, finished 14:34:18 2023-11-06

Out[91]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
95	5.7	3.0	4.2	1.2	1
96	5.7	2.9	4.2	1.3	1
97	6.2	2.9	4.3	1.3	1
98	5.1	2.5	3.0	1.1	1
99	5.7	2.8	4.1	1.3	1

100 rows × 5 columns

In [98]:
```
data.columns
```
executed in 17ms, finished 14:36:17 2023-11-06

Out[98]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'target'], dtype='object')
```

In [94]:
```
data.columns = ['sepal_length', 'sepal_width', 'petal_length',
       'petal_width', 'target']
```
executed in 8ms, finished 14:36:02 2023-11-06

In [97]:
```
data
```
executed in 28ms, finished 14:36:10 2023-11-06

Out[97]:

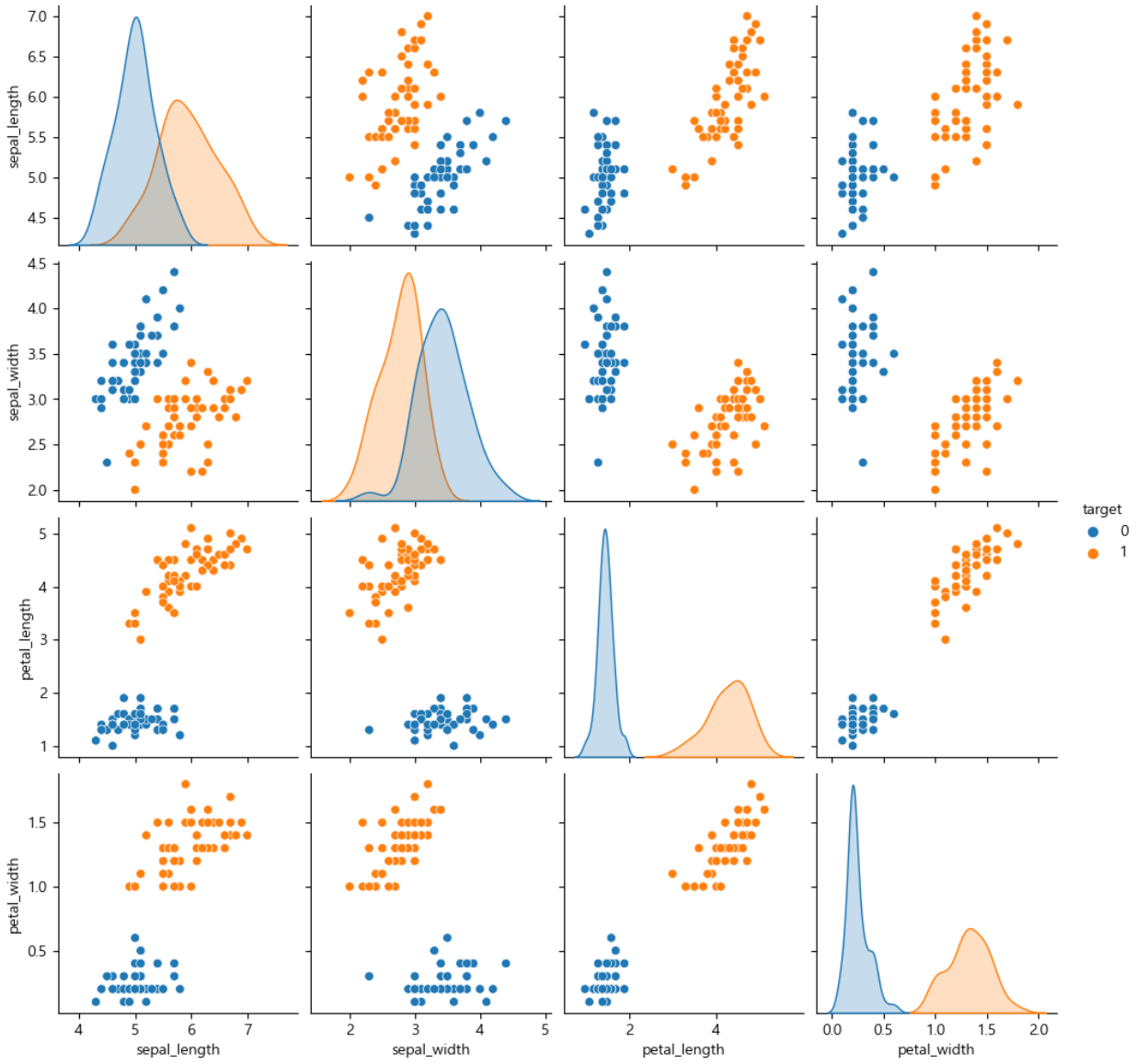
	sepal_length	sepal_width	petal_length	petal_width	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
95	5.7	3.0	4.2	1.2	1
96	5.7	2.9	4.2	1.3	1
97	6.2	2.9	4.3	1.3	1
98	5.1	2.5	3.0	1.1	1
99	5.7	2.8	4.1	1.3	1

100 rows × 5 columns

In [149]:
```
sns.pairplot(data , hue = 'target')
```

executed in 3.45s, finished 15:13:12 2023-11-06

Out[149]: <seaborn.axisgrid.PairGrid at 0x2aad471dbb0>

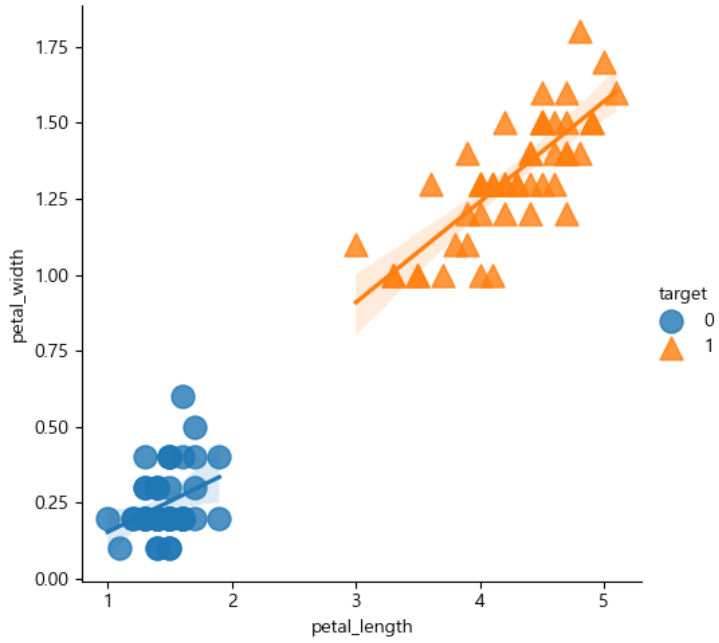


In [ ]:

In [102]:
```
sns.lmplot(x = 'petal_length',y = 'petal_width' , data = data , scatter_kws = {'s' : 150} , hue = 'target' , markers = ['o','^'])
```

executed in 432ms, finished 14:38:58 2023-11-06

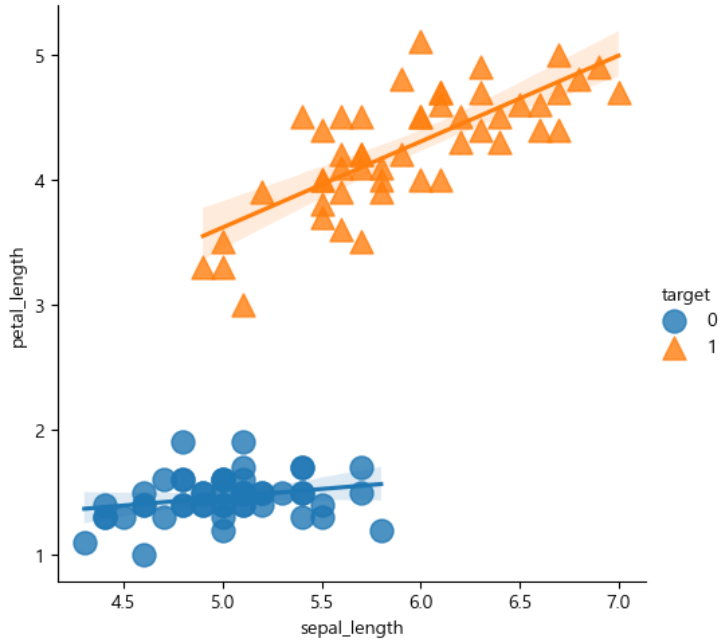
Out[102]: <seaborn.axisgrid.FacetGrid at 0x2aaccd21a30>



In [103]:
```
sns.lmplot(x = 'sepal_length',y = 'petal_length' , data = data , scatter_kws = {'s' : 150} , hue = 'target' , markers = ['o','^'])
```

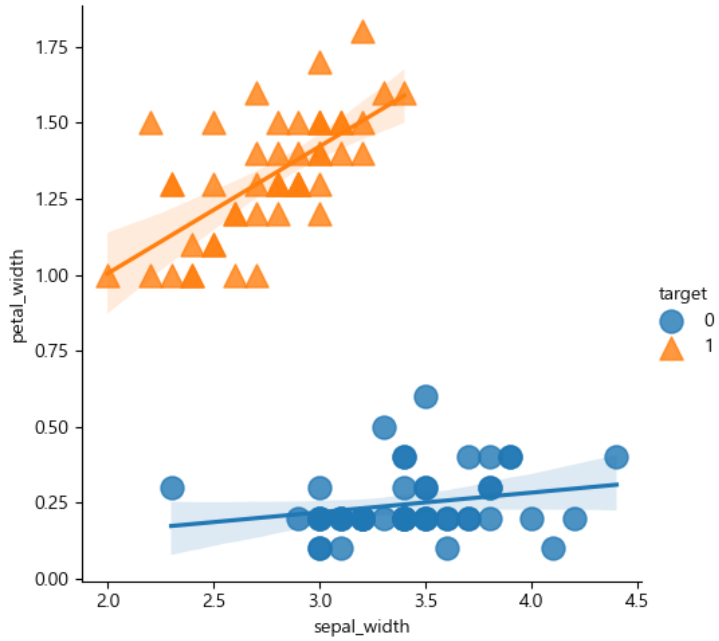
executed in 575ms, finished 14:38:59 2023-11-06

Out[103]: <seaborn.axisgrid.FacetGrid at 0x2aaccaca3a0>



In [104]:
```
sns.lmplot(x = 'sepal_width',y = 'petal_width' , data = data , scatter_kws = {'s' : 150} , hue = 'target' , markers = ['o','^'])
```
executed in 513ms, finished 14:38:59 2023-11-06

Out[104]: <seaborn.axisgrid.FacetGrid at 0x2aac94226d0>



- 모두 분류가 잘 되어 있다. sepal_width , petal_width를 이용하자

In [106]:
```
x = data[['sepal_width','petal_width']]

y = data.target
```
executed in 11ms, finished 14:39:36 2023-11-06

In [107]:
```
from sklearn.model_selection import train_test_split

train_input , test_input , train_target , test_target = train_test_split(x , y , test_size = 0.2 , random_state = 0)
```
executed in 19ms, finished 14:41:00 2023-11-06

In [111]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report , accuracy_score
```
executed in 12ms, finished 14:42:06 2023-11-06

In [150]:
```
params = {'C' : [0.0001 , 0.001 , 0.01 , 0.1 , 1 , 10],
          'kernel' : ['linear','poly','rbf']}
```
executed in 14ms, finished 15:15:53 2023-11-06

In [151]:
```
gs = GridSearchCV(SVC() , params , cv = 10)
gs.fit(train_input , train_target)

print(gs.best_params_)
```
executed in 507ms, finished 15:15:55 2023-11-06

```
{'C': 0.1, 'kernel': 'linear'}
```

In [152]:
```
svc = SVC(C = 0.1 , kernel = 'linear')
```
executed in 12ms, finished 15:16:00 2023-11-06

In [153]:
```
svc.fit(train_input , train_target)

print(classification_report(test_target , svc.predict(test_input)))
print('Accuracy : ' , accuracy_score(test_target , svc.predict(test_input)))
```
executed in 33ms, finished 15:16:02 2023-11-06

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00        10

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Accuracy :  1.0
```

6.2 보고서

In [154]:
```
prediction = pd.DataFrame({'prediction' : svc.predict(test_input) , 'Actual' : test_target})
prediction
```

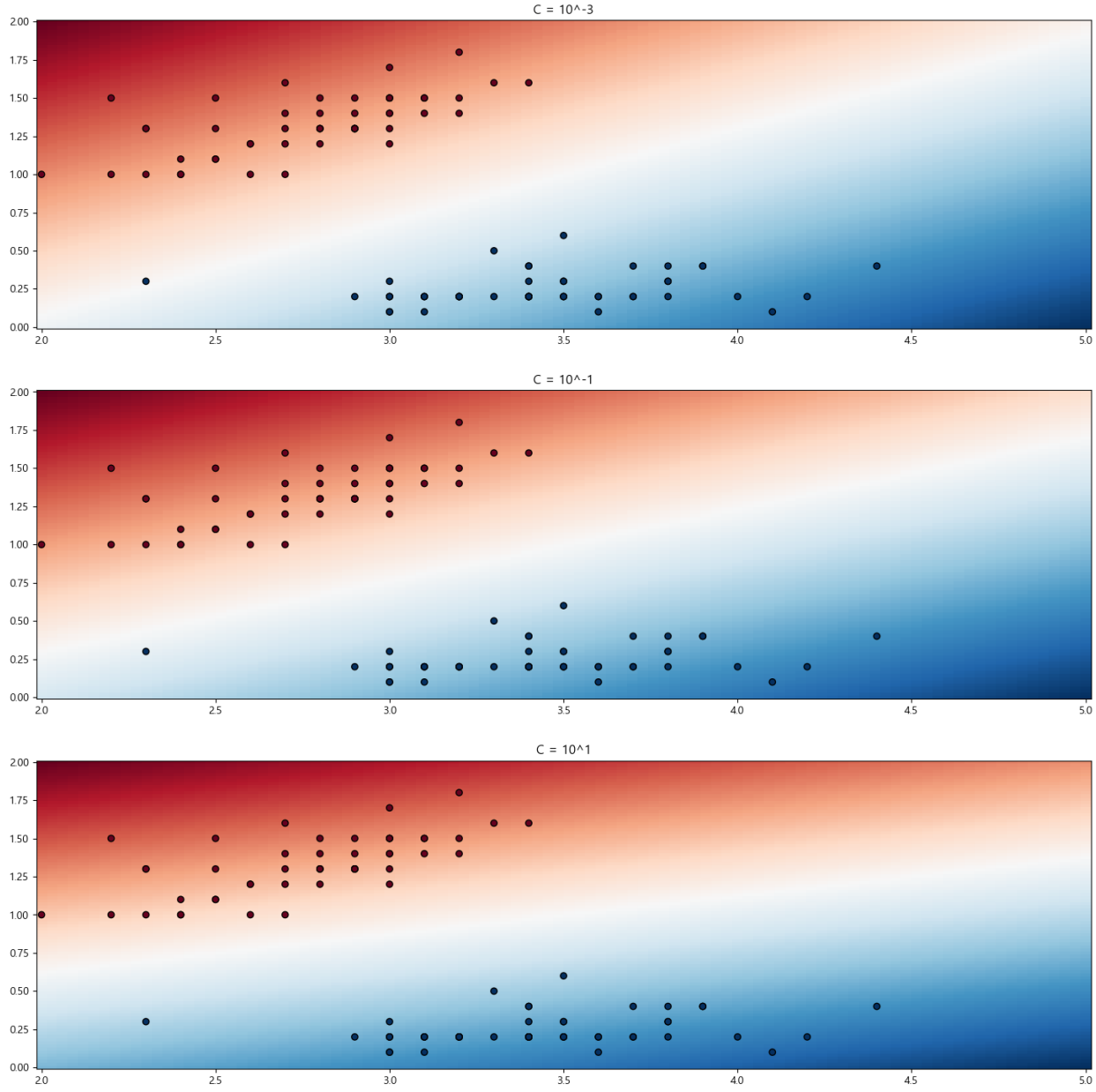
executed in 35ms, finished 15:16:06 2023-11-06

Out[154]:

	prediction	Actual
26	0	0
86	1	1
2	0	0
55	1	1
75	1	1
93	1	1
16	0	0
73	1	1
54	1	1
95	1	1
53	1	1
92	1	1
78	1	1
13	0	0
7	0	0
30	0	0
22	0	0
24	0	0
33	0	0
8	0	0

In [159]:
```
c_list = []
c_list.append(gs.best_params_['C']*0.01)
c_list.append(gs.best_params_['C'])
c_list.append(gs.best_params_['C']*100)

position = data.target
classifiers = []

for c in c_list:
    clf = SVC(C = c , kernel = 'linear')
    clf.fit(train_input , train_target)
    classifiers.append((c , clf))

plt.figure(figsize = (18,18))

xx , yy = np.meshgrid(np.linspace(2,5,100) , np.linspace(0,2,100))

for (k , (c ,clf)) in enumerate(classifiers):
    Z = clf.decision_function(np.c_[xx.ravel() , yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.subplot(3,1,k+1)
    plt.title('C = 10^%d' % np.log10(c))

    plt.pcolormesh(xx , yy , -Z , cmap = plt.cm.RdBu)
    plt.scatter(data['sepal_width'] , data['petal_width'] , c = position , cmap = plt.cm.RdBu_r , edgecolors = 'k')
```

executed in 546ms, finished 15:17:50 2023-11-06



- 위 그림에서 1행 2열이 best_params_에 있는 위치이다.