1)

a) $\hat{w}_{RR} = \underset{w}{\operatorname{argmin}} \|\Phi w - y\|_2^2 + \lambda \|w\|_2^2$

$= \underset{w}{\operatorname{argmin}} (\Phi w - y)^T (\Phi w - y) + \lambda w^T w$

$= \underset{w}{\operatorname{argmin}} \; w^T \Phi^T \Phi w - 2 w^T \Phi^T y + y^T y + 2\lambda w$

Let $f(w) = w^T \Phi^T \Phi w - 2 w^T \Phi^T y + y^T y + 2\lambda w$

$\nabla_w f = 2\Phi^T \Phi w - 2\Phi^T y + 2\lambda w = 0$

$\Phi^T \Phi w + \lambda w = \Phi^T y$

$(\Phi^T \Phi + \lambda I) w = \Phi^T y$

$\hat{w}_{RR} = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$

Let $\Phi = U \Sigma V^T$

$\hat{w}_{RR} = (V \Sigma^T U^T U \Sigma V^T + \lambda I)^{-1} V \Sigma^T U^T y$

$= (V \Sigma^T \Sigma V^T + \lambda I)^{-1} V \Sigma^T U^T y$

$= (V \Sigma^T \Sigma V^T + \lambda V V^T)^{-1} V \Sigma^T U^T y$

$= (V(\Sigma^T \Sigma + \lambda I) V^T)^{-1} V \Sigma^T U^T y$

$= V(\Sigma^T \Sigma + \lambda I)^{-1} V^T V \Sigma^T U^T y$

$= V(\Sigma^T \Sigma + \lambda I)^{-1} \Sigma^T U^T y$

From class, we know that $(\Sigma^T \Sigma + \lambda I)^{-1} \Sigma^T = \Sigma^T (\Sigma \Sigma^T + \lambda I)^{-1}$

$\Rightarrow \hat{w}_{RR} = V \Sigma^T (\Sigma \Sigma^T + \lambda I)^{-1} U^T y$

$= V \Sigma^T U^T U (\Sigma \Sigma^T + \lambda I)^{-1} U^T y$

$= \Phi^T U (\Sigma \Sigma^T + \lambda I)^{-1} U^T y$

$= \Phi^T (U(\Sigma \Sigma^T + \lambda I) U^T)^{-1} y$

$= \Phi^T (U \Sigma \Sigma^T U^T + \lambda U U^T)^{-1} y$

$= \Phi^T (U \Sigma V^T V \Sigma^T U^T + \lambda I)^{-1} y$

$= \Phi^T (\Phi \Phi^T + \lambda I)^{-1} y$

Thus, $\boxed{\hat{w}_{RR} = \Phi^T (\Phi \Phi^T + \lambda I)^{-1} y}$

For this solution to be valid, $(\Sigma \Sigma^T + \lambda I)^{-1}$ must exist. While Σ may have some singular values $= 0$, causing $\Sigma \Sigma^T$ to have 0s on the diagonal, adding a $\lambda > 0$ to all diagonal entries of $\Sigma \Sigma^T$ mean all values on diagonal of $\Sigma \Sigma^T + \lambda I$ are > 0, since all $\sigma_i \geq 0$ so $\sigma_i + \lambda > 0$, meaning $(\Sigma \Sigma^T + \lambda I)^{-1}$ exists regardless of what the rank of Φ is.

b)

i) $\hat{w}_{RR} = \Phi^T \alpha = \Phi^T (\Phi \Phi^T + \lambda I)^{-1} y$

$\Phi^T \alpha = \Phi^T (K + \lambda I)^{-1} y$ since we define $K = \Phi \Phi^T$

$(\Phi^T)^+ \Phi^T \alpha = (\Phi^T)^+ \Phi^T (K + \lambda I)^{-1} y$

we know $\Phi^T \in \mathbb{R}^{d \times n}$ and $(\Phi^T)^+ \in \mathbb{R}^{n \times d}$, so $(\Phi^T)^+ \Phi^T \in \mathbb{R}^{n \times n}$. since Φ has n linearly independent rows, Φ^T has n linearly independent columns, so $(\Phi^T)^+ \Phi^T = I$.

Thus, $\boxed{\alpha = (K + \lambda I)^{-1} y}$

ii) $\hat{y} = w_{RR}^T \phi(z) = (\Phi^T \alpha)^T \phi(z) = \alpha^T \Phi \phi(z)$

we know $\Phi \phi(z) = \begin{pmatrix} \phi(x_1)^T \phi(z) \\ \vdots \\ \phi(x_n)^T \phi(z) \end{pmatrix} = \begin{pmatrix} K(x_1, z) \\ \vdots \\ k(x_n, z) \end{pmatrix} = K(X, z)$

$\Rightarrow \hat{y} = \alpha^T \Phi \phi(z) = \alpha^T K(X, z) = ((K + \lambda I)^{-1} y)^T K(X, z)$

$\Rightarrow \boxed{\hat{y} = ((K + \lambda I)^{-1} y)^T K(X, z)}$

c) we defined $K = \Phi \Phi^T$ where $K_{ij} = \kappa(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, or represents the similarity between the vectors. This is useful because we can find the weight vector α using K which requires a simple evaluation between vector pairs using the kernel function. This allows to **avoid computing and storing $\phi(x_i)$** which grows on the order of power such as requiring $O(p^d)$ to store a d-degree polynomial from our p-dimensional input.

d) code attached

2)

a)

i) $g(x) = g_1(x) + g_2(x)$

$g_1(x)$ convex: $g_1(w) \geq g_1(v) + \nabla g_1(v)^T (w-v) \quad \forall w, v \in \mathbb{R}^p$

$g_2(x)$ convex: $g_2(w) \geq g_2(v) + \nabla g_2(v)^T (w-v) \quad \forall w, v \in \mathbb{R}^p$

$g_1(w) + g_2(w) \geq g_1(v) + g_2(v) + \nabla g_1(v)^T (w-v) + \nabla g_2^T (w-v) \quad \forall w, v \in \mathbb{R}^p$

$g_1(w) + g_2(w) \geq g_1(v) + g_2(v) + \nabla (g_1(v) + g_2(v))^T (w-v) \quad \forall w, v \in \mathbb{R}^p$

$\Rightarrow g(w) \geq g(v) + \nabla g(v)^T (w-v) \quad \forall w, v \in \mathbb{R}^p$

Thus, **$g(x)$ is a convex function**

ii) $g(w) = w^T K w$

$g(w) = g(v) + \nabla v g^T (w-v) + \frac{1}{2} (w-v)^T \nabla v^2 g (w-v)$

$= v^T K v + 2(Kv)^T (w-v) + (w-v)^T K (w-v)$

1st-order Taylor expansion includes the 1st 2 terms:

$g(v) + \nabla v g^T (w-v) = v^T K v + 2(Kv)^T (w-v)$

Need to show $\forall w, v \in \mathbb{R}^p \quad g(w) \geq g(v) + \nabla_v g^T (w-v)$

since K is positive semi-definite, $(w-v)^T K (w-v) \geq 0$

$\Rightarrow g(w) = v^T K v + 2(Kv)^T (w-v) + (w-v)^T K (w-v)$

$\geq v^T K v + 2(Kv)^T (w-v)$

$= g(v) + \nabla v g^T (w-v)$

$\Rightarrow g(w) \geq g(v) + \nabla v g^T (w-v) \quad \forall w, v \in \mathbb{R}^p$ so **$g(w)$ is a convex function**

b) At t steps, $f(\alpha^{(t)}) = \sum_{i=1}^{n} (1 - y_i K_i^T \alpha^{(t)})_+ + \lambda \alpha^{(t)T} K \alpha^{(t)}$

since $y_i K_i^T \alpha^{(t)} > 1$, $1 - y_i K_i^T \alpha^{(t)} < 0 \Rightarrow (1 - y_i K_i^T \alpha^{(t)})_+ = 0$

Thus, $f(\alpha^{(t)}) = \sum_{i=1}^{n} (1 - y_i K_i^T \alpha^{(t)})_+ + \lambda \alpha^{(t)T} K \alpha^{(t)} \Rightarrow$ **$f(\alpha^{(t)}) = \lambda \alpha^{(t)T} K \alpha^{(t)}$**

$\alpha^{(t+1)} = \alpha^{(t)} - \tau \nabla_\alpha f|_{\alpha^{(t)}}$

$= \alpha^{(t)} - \tau (2\lambda K \alpha^{(t)})$

$\Rightarrow$ **$\alpha^{(t+1)} = \alpha^{(t)} - 2\tau\lambda K \alpha^{(t)}$**

c) $\nabla_\alpha f = \sum_{i=1}^{n} \mathbb{I}\{1 - y_i K_i^T \alpha > 0\} (-y_i K_i) + 2\lambda K \alpha$

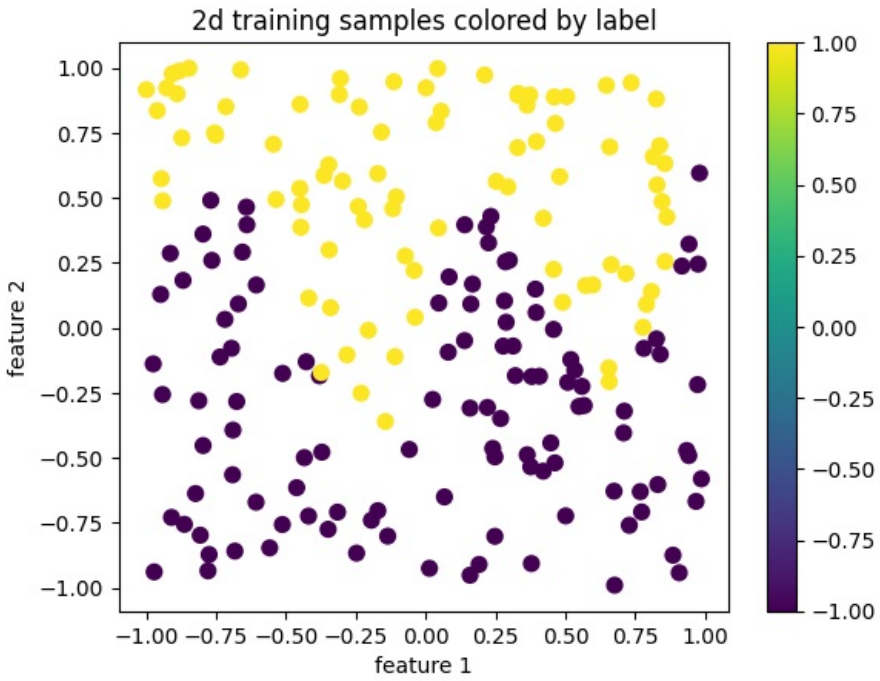
At optimal $\nabla_\alpha f|_{\alpha = \alpha^*} = 0 \Rightarrow \sum_{i=1}^{n} \mathbb{I}\{1 - y_i K_i^T \alpha > 0\} (-y_i K_i) + 2\lambda K \alpha = 0$

Suppose $\lambda K \alpha = 0$. Then either $\lambda = 0$ which is not the case, or $K\alpha = 0$, meaning $K_i^T \alpha = 0 \ \forall i$, so all n samples are classified the same way, also unlikely to be the case. Therefore, $2\lambda K \alpha \neq 0$, so $\sum_{i=1}^{n} \mathbb{I}\{1 - y_i x_i^T \alpha > 0\} (-y_i K_i) \neq 0$ so some samples have $1 - y_i K_i^T \alpha > 0 \Rightarrow y_i K_i^T \alpha < 1$, so they are classified correctly (as $\alpha^{(t)}$ classified all correctly) but are close to the decision boundary, so some loss is added, so **not all terms = 0 in loss** as we can obtain even perfect classification with loss > 0.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
import sys
import numpy.linalg as la
from tabulate import tabulate
import random
from PIL import Image
import math
```

1d

In [2]:
```
n = 200
p = 2
X = 2 * (np.random.rand(n, p) - 0.5)
y = np.sign(X[:, 1] - (X[:, 0] ** 2 / 2 + np.sin(X[:, 0] * 7) / 2))

plt.figure(1)
plt.scatter(X[:, 0], X[:, 1], 50, c=y)
plt.colorbar()
plt.xlabel("feature 1")
plt.ylabel("feature 2")
plt.title("2d training samples colored by label")
plt.show()
```

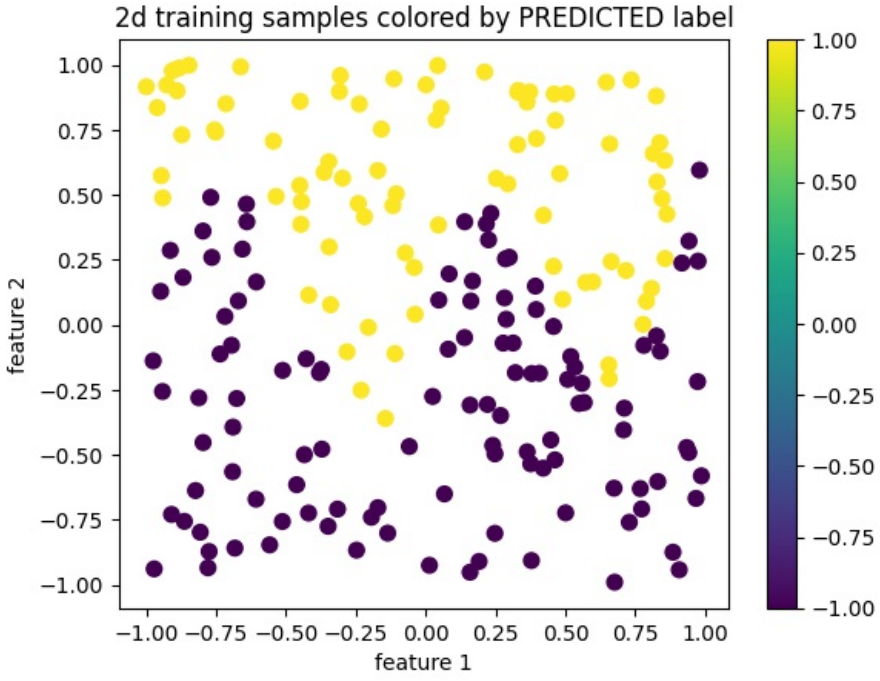


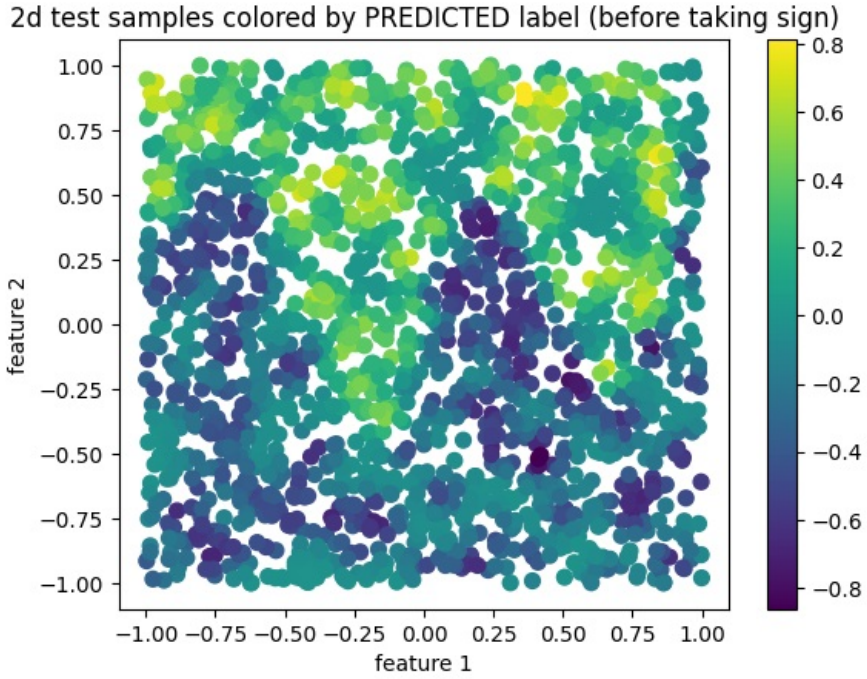
In [3]:
```
def krr(sigma, lam):
    ### YOUR CODE STARTS HERE ###
    K = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            # note: ||x_i - x_j||^2 = (x_i - x_j)^T (x_i - x_j) = (x_i - x_j) dot (x_i - x_j)
            K[i, j] = math.e ** (-(np.dot(X[i] - X[j], X[i] - X[j])) / (2 * sigma**2))
    alpha = la.inv(K + lam * np.eye(n)) @ y
    yhat = K @ alpha
    ### YOUR CODE ENDS HERE ###

    y2 = np.array(np.sign(yhat))
    plt.figure(2)
    plt.scatter(X[:, 0], X[:, 1], 50, c=y2)
    plt.colorbar()
    plt.xlabel("feature 1")
    plt.ylabel("feature 2")
    plt.title("2d training samples colored by PREDICTED label")
    plt.show()

    ntest = 2000
    Xtest = 2 * (np.random.rand(ntest, p) - 0.5)

    ### YOUR CODE STARTS HERE ###
    Ktest = np.zeros((ntest, n))
    for i in range(ntest):
        for j in range(n):
            # note: ||x_i - x_j||^2 = (x_i - x_j)^T (x_i - x_j) = (x_i - x_j) dot (x_i - x_j)
```

```
            Ktest[i, j] = math.e ** (-(np.dot(Xtest[i] - X[j], Xtest[i] - X[j])) / (2 * sigma**2))
    ytest = Ktest @ alpha
    ### YOUR CODE ENDS HERE ###

    plt.figure(3)
    plt.scatter(Xtest[:, 0], Xtest[:, 1], 50, c=np.array(ytest))
    plt.colorbar()
    plt.xlabel("feature 1")
    plt.ylabel("feature 2")
    plt.title("2d test samples colored by PREDICTED label (before taking sign)")
    plt.show()
```

In [4]:

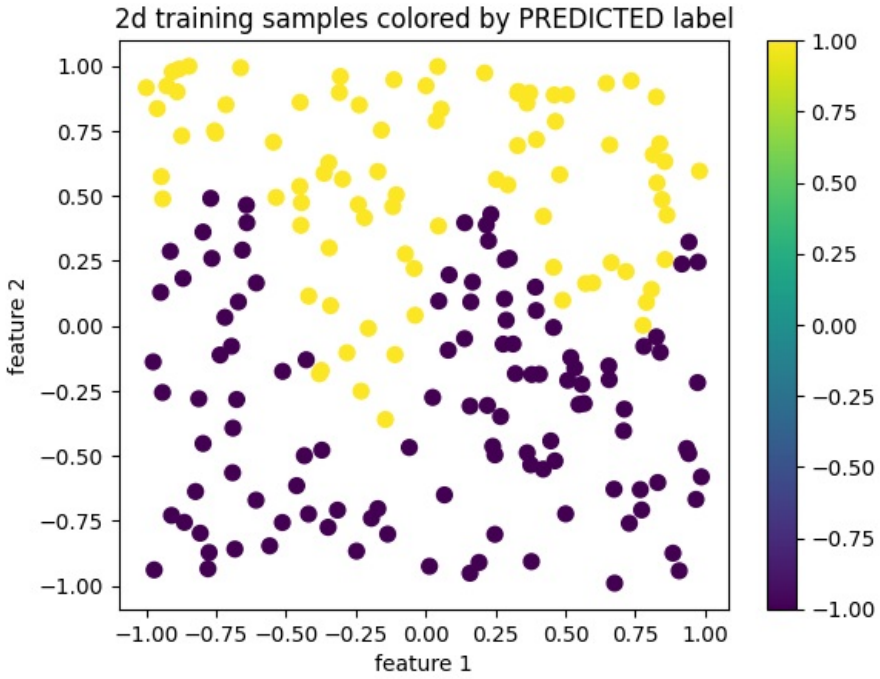
```
krr(sigma=0.05, lam=1)
```

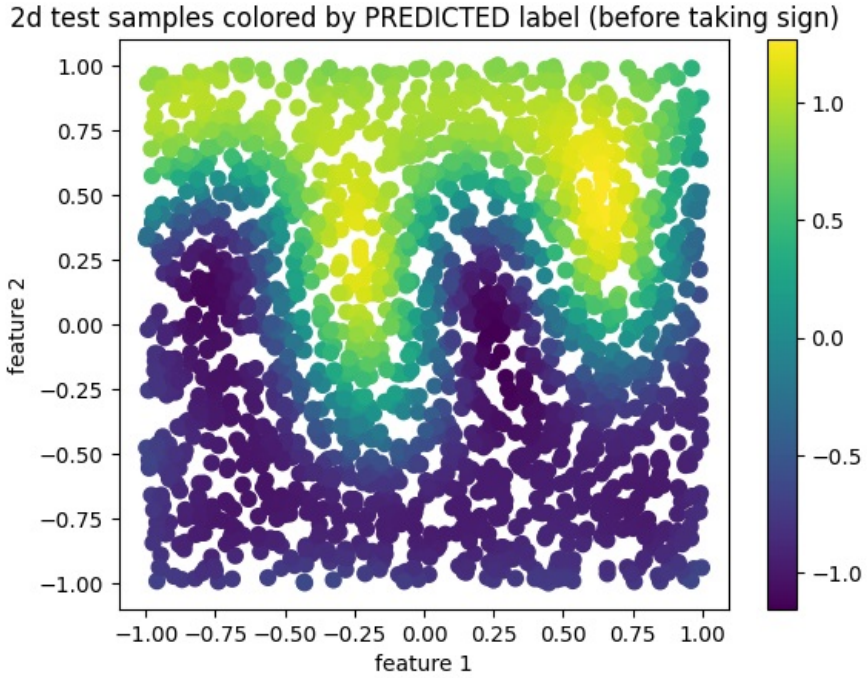




In [5]:

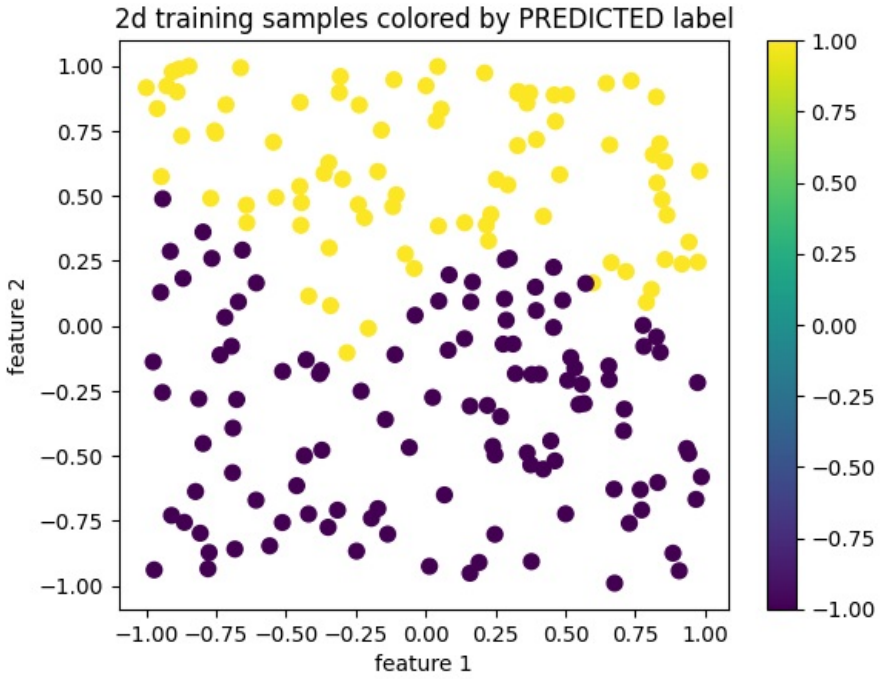
```
krr(sigma=0.25, lam=1)
```

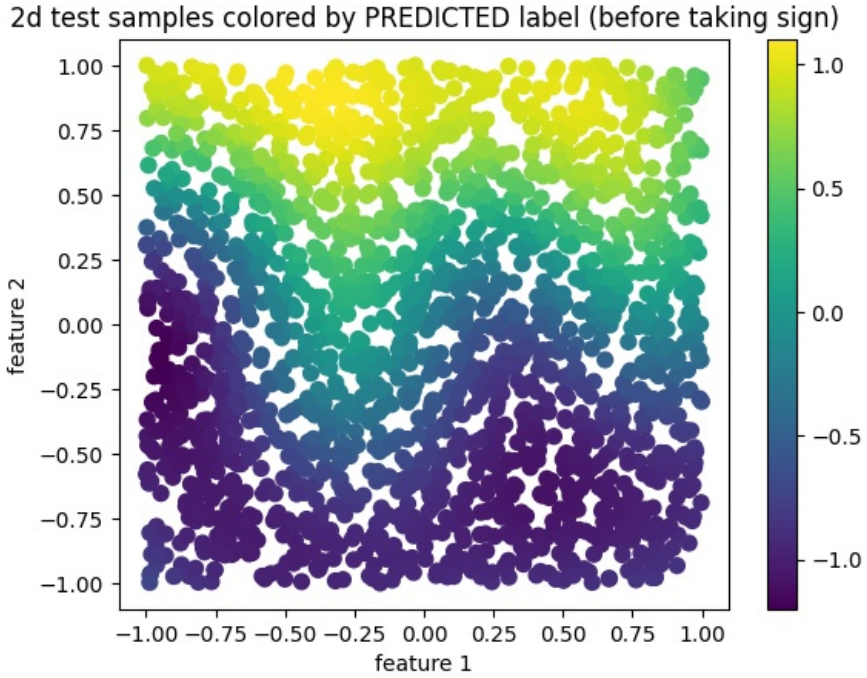




In [6]:

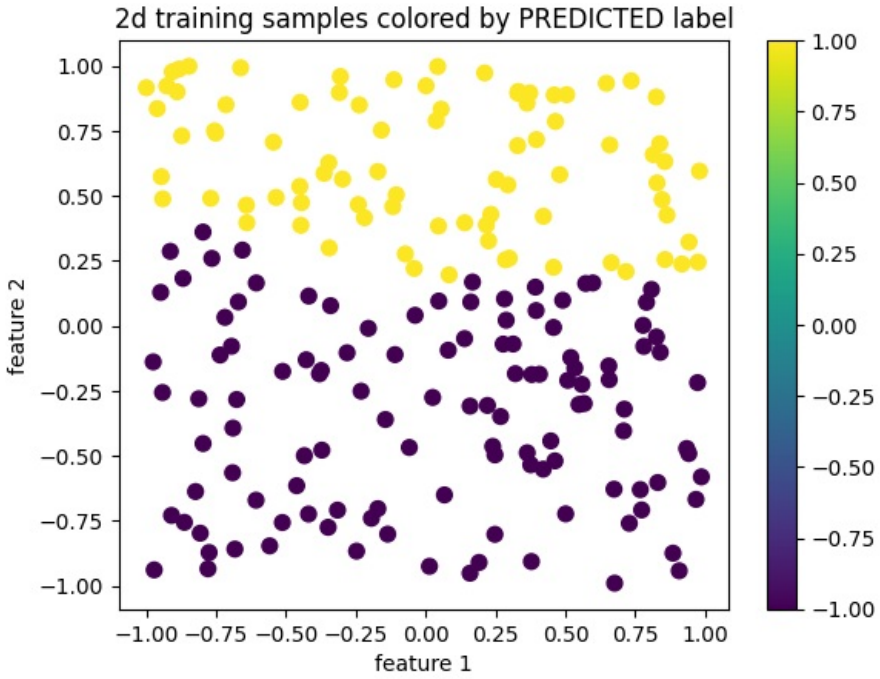
```
krr(sigma=0.5, lam=1)
```

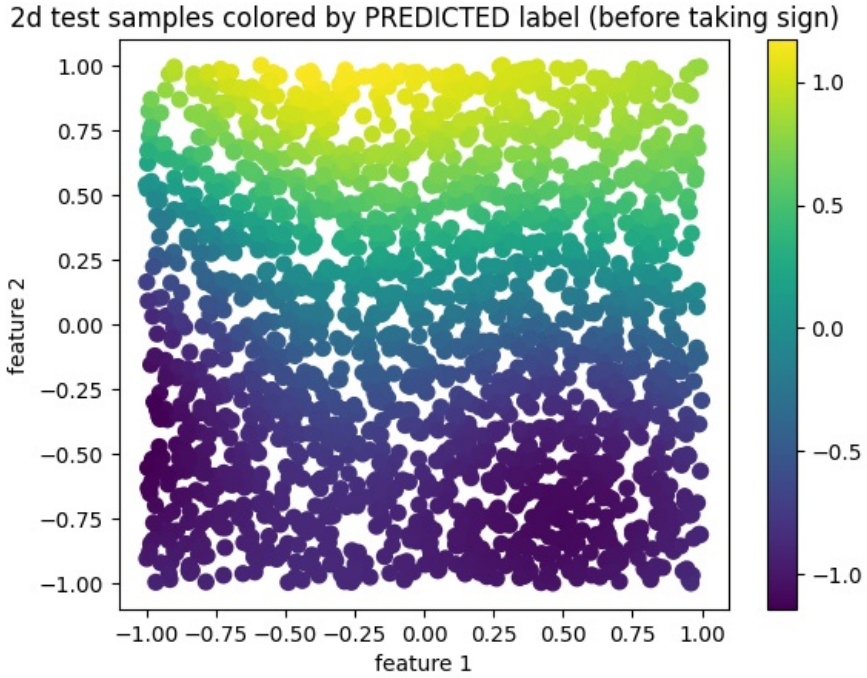




In [7]:

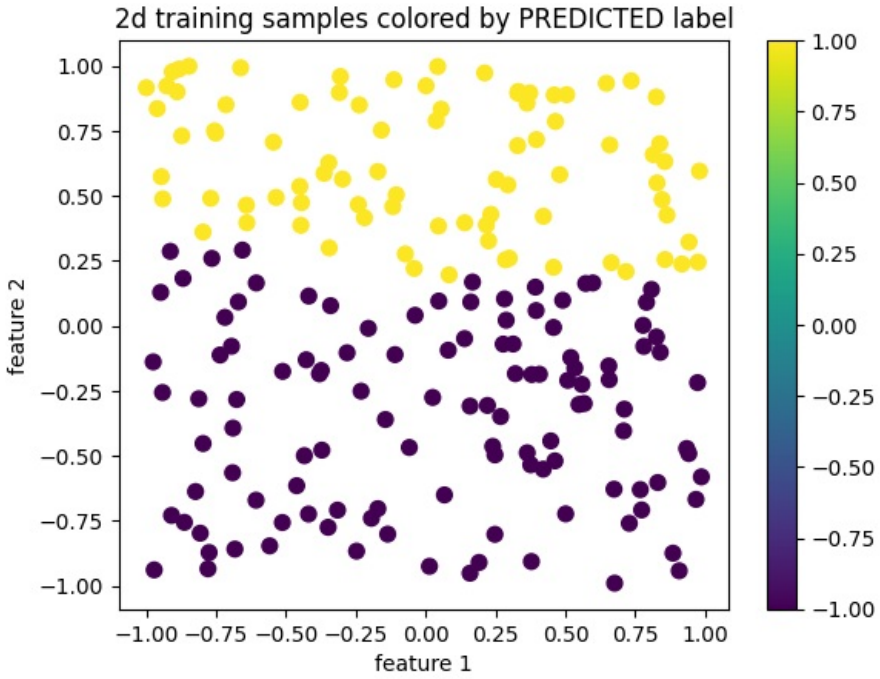
```
krr(sigma=0.75, lam=1)
```

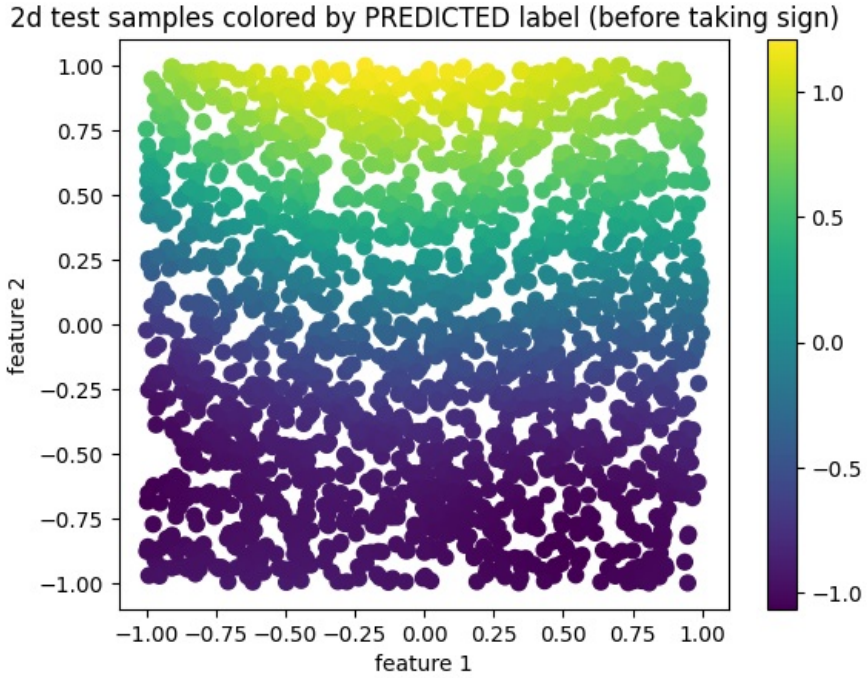




In [8]:

```
krr(sigma=1, lam=1)
```





We see that the model becomes more linear as σ increases. When $\sigma = 0.05$, we see that the decision boundary in the test samples is not very distinct. On the other hand, when $\sigma = 0.8$ or $\sigma = 1$, for example, the decision boundary is significantly more linear compared to $\sigma = 0.25$, having a highly nonlinear decision boundary.

If we think of $K(x_i, x_j)$ as providing a similarity score between x_i and x_j, we see that as $||x_i - x_j||$ decreases, $K(x_i, x_j)$ increases, which makes sense because if x_i and x_j are closer together, they should have a higher similarity score. Given this, we see that as σ increases, $K(x_i, x_j)$ increases, meaning a higher σ treats the same samples as more similar to each other. This means the decision boundary can incorporate information of both x_i and x_j without becoming too nonlinear since the $K(x_i, x_j)$ treats them as similar already. Therefore, we see that with higher σ, the decision boundary is more linear.