```
from cvxopt import solvers
from cvxopt import matrix
from functools import partial


import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from scipy.spatial.distance import cdist


data = pd.read_csv("/content/titanic_processed.csv")
x_data = data.drop('Survived',axis=1).values
y_data = data['Survived'].values
y_data[y_data==0] = -1
x_train,x_test,y_train,y_test = train_test_split(x_data,y_data,test_size=0.2,random_state=0)


scaler = StandardScaler()
x = scaler.fit_transform(x_train)
y = y_train.reshape(-1,1)


sigma = 2.5
c = .35
def rbf_kernel(x1,x2,sigma):
  return np.exp(-(cdist(x1,x2,'sqeuclidean'))/(2*sigma**2))

kernel = partial(rbf_kernel,sigma=sigma)


n = x.shape[0]
I_n=np.eye(n)
P = (y@y.T)*kernel(x,x)
q = np.full(n,-1)
G = np.vstack((-1*I_n,I_n))
h = np.hstack((np.zeros(n),np.full(n,c)))
A = y_train.reshape(1,-1)
b = np.zeros(1)


P,q,G,h,A,b = map(lambda x:matrix(x,tc="d"),(P,q,G,h,A,b))


solutions = solvers.qp(P,q,G,h,A,b)

           pcost       dcost       gap    pres   dres
      0: -2.1166e+02 -5.8558e+02  6e+03  8e+00  4e-15
      1: -8.8848e+01 -4.9225e+02  6e+02  4e-01  3e-15
      2: -8.4608e+01 -1.3669e+02  5e+01  2e-03  1e-15
      3: -9.2566e+01 -1.1007e+02  2e+01  5e-04  1e-15
```

```
 4: -9.5567e+01 -1.0239e+02  7e+00  1e-04  1e-15
 5: -9.6721e+01 -9.9922e+01  3e+00  6e-05  1e-15
 6: -9.7365e+01 -9.8738e+01  1e+00  8e-06  1e-15
 7: -9.7606e+01 -9.8266e+01  7e-01  2e-06  1e-15
 8: -9.7781e+01 -9.7964e+01  2e-01  2e-07  1e-15
 9: -9.7826e+01 -9.7894e+01  7e-02  4e-08  1e-15
10: -9.7850e+01 -9.7861e+01  1e-02  5e-09  1e-15
11: -9.7852e+01 -9.7857e+01  4e-03  7e-10  1e-15
12: -9.7854e+01 -9.7855e+01  1e-03  1e-10  1e-15
13: -9.7854e+01 -9.7855e+01  2e-04  2e-11  1e-15
14: -9.7854e+01 -9.7854e+01  4e-05  3e-12  1e-15
Optimal solution found.
```

```
a = np.asarray(solutions['x']).squeeze()
```

```
support_index = np.logical_and(a>=1e-10,a<c)
x_s = x[support_index]
b = np.mean(y-a*y.T@kernel(x,x_s))
```

```
x_test = scaler.fit_transform(x_test)
predict = np.sign(a*y.T@kernel(x,x_test)+b)
print(f"Accuracy of the classifer:{(y_test==predict).mean()*100:.2f}%")
```

```
Accuracy of the classifer:75.28%
```