IMPORT LIBRARIES

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_sc
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
```

LOAD AND PREPARE THE DATASET

In [2]:
```
digits = datasets.load_digits()
mask = (digits.target == 1) | (digits.target == 7)
X = digits.data[mask]
y = digits.target[mask]

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_s
```

TRAIN SVM WITH LINEAR AND RBF KERNELS

In [3]:
```
# Linear Kernel
svm_linear = SVC(kernel='linear', C=1.0)
svm_linear.fit(X_train, y_train)
# RBF Kernel
svm_rbf = SVC(kernel='rbf', C=1.0, gamma='scale')
svm_rbf.fit(X_train, y_train)
```

Out[3]:
```
SVC()
```

EVALUATE THE PERFORMANCE

In [5]:
```
#Linear Kernel Performance
print("Linear Kernel Evaluation:")
y_pred_linear = svm_linear.predict(X_test)
print(confusion_matrix(y_test, y_pred_linear))
print(classification_report(y_test, y_pred_linear))

#RBF Kernel Performance
print("RBF Kernel Evaluation:")
y_pred_rbf = svm_rbf.predict(X_test)
print(confusion_matrix(y_test, y_pred_rbf))
print(classification_report(y_test, y_pred_rbf))
```

```
Linear Kernel Evaluation:
[[59  0]
 [ 0 50]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        59
           7       1.00      1.00      1.00        50

    accuracy                           1.00       109
   macro avg       1.00      1.00      1.00       109
weighted avg       1.00      1.00      1.00       109

RBF Kernel Evaluation:
[[59  0]
 [ 0 50]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        59
           7       1.00      1.00      1.00        50

    accuracy                           1.00       109
   macro avg       1.00      1.00      1.00       109
weighted avg       1.00      1.00      1.00       109
```

HYPERPARAMETER TUNING

In [8]:
```
param_grid = {
    'C' : [0.1, 1, 10, 100],
    'gamma': [0.001, 0.01, 0.1, 1]
}
grid = GridSearchCV(SVC(kernel='rbf'),param_grid, cv=5)
grid.fit(X_train, y_train)
print("Best Hyperparameters:", grid.best_params_)
```

```
Best Hyperparameters: {'C': 0.1, 'gamma': 0.001}
```

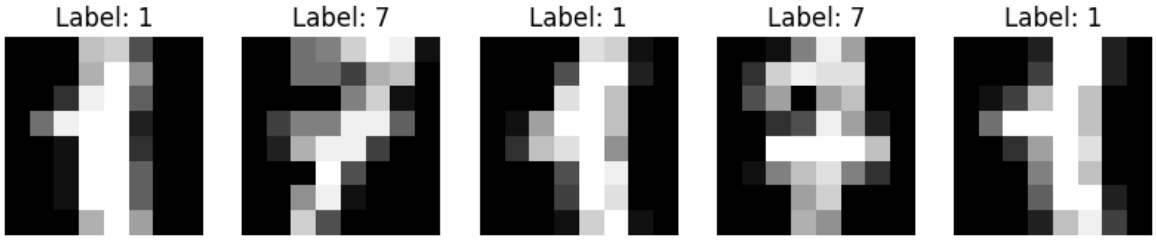
CROSS VALIDATION

In [9]:
```
# cross validation with best estimator
best_svm = grid.best_estimator_

cv_scores = cross_val_score(best_svm, X, y, cv=5)
print("Cross Validation Accuracy Scores:", cv_scores)
print("Mean CV Accuracy:", np.mean(cv_scores))
```

```
Cross Validation Accuracy Scores: [1. 1. 1. 1. 1.]
Mean CV Accuracy: 1.0
```

In [10]:
```
# Visualizing some digi samples
fig, axes = plt.subplots(1,5, figsize=(10,3))
for i, ax in enumerate(axes):
    ax.imshow(X[i].reshape(8,8), cmap='gray')
    ax.set_title(f"Label: {y[i]}")
    ax.axis('off')
plt.show()
```



In [ ]: