Support Vector Machine (SVM) - Home Exercise:

The **Boston Housing Dataset** is a popular dataset used in regression tasks within the field of machine learning. This dataset contains information collected by the U.S. Census Service concerning housing in the area of Boston, Massachusetts. The dataset is often used to predict housing prices based on various features.
Below is an explanation of typical columns you might find in this dataset:

- **CRIM** (per capita crime rate by town) → Represents the crime rate per capita in the town. High crime rates can negatively impact property values.
- **ZN** (proportion of residential land zoned for lots over 25,000 sq. ft.) → Indicates the proportion of land within the town that is zoned for large residential properties. Higher values typically signify more affluent areas.
- **INDUS** (proportion of non-retail business acres per town) → Reflects the proportion of the town's land that is dedicated to non-retail business. Higher values suggest more industrial activity.
- **CHAS** (1 if tract bounds river; 0 otherwise) → A binary variable indicating whether the residential area is near the Charles River. Properties near scenic areas often have higher values.
- **AGE** (proportion of owner-occupied units built prior to 1940) → Indicates the age of the houses. Older homes might be cheaper but also might require more maintenance.
- **RAD** (index of accessibility to radial highways) → Measures how easily accessible highways are from the residential area. Better road access usually enhances property values.
- **TAX** (full-value property tax rate per $10,000) → Represents the property tax rate. High taxes may reduce the attractiveness of an area to buyers.
- **MEDV** (Median value of owner-occupied homes in $1000s) → The target variable representing the median value of homes in the area. This is the variable typically predicted in regression models.

These features collectively represent a variety of socio-economic, demographic, and geographic aspects influencing housing prices in the Boston area. The goal often is to use these independent variables to predict the dependent variable, `MEDV`.

So in the following exercise the label will be the house pricing (‘MEDV’ column) and the features will be all the other columns in the dataset.

Exercise instructions:

Use the **BostonHousing.csv** file for this exercise.

Data preparation:

1. Check for missing values in the dataset.
2. Check for duplicate rows in the dataset.
3. In case you found any of those, remove them from the df.

Simple SVM Machine Learning:

1. Use your preprocessing dataset.
2. Apply train / test split with test size of 30% and random seed of 42.
3. Apply feature scaling on the entire features of type Standardization.
4. Apply Simple SVM for a regression problem to predict the ‘MEDV’ value.
   Use kernel function of the RBF kernel and low penalty on misclassification points.
5. Predict the house pricing of the test set.
6. Evaluate your model using MAE, MSE and RMSE.
7. Plot the true vs. predicted house prices.

Optimal SVM Machine Learning:

1. Use your preprocessing dataset.
2. Apply train / test split with test size of 30% and random seed of 42.
3. Apply feature scaling on the entire features of type Standardization.
4. Apply Grid Search to find the optimal kernel function and the optimal C value from the following:
   For the kernel function → Check rbf / linear / poly / sigmoid.
   For the C value → Check C between [0.1, 1, 10, 100, 1000]
5. Use the MSE as the score metric to evaluate the optimal parameters.
6. Print the Grid Search optimal parameters.
7. Train your SVM model with the optimal parameters you found from the Grid Search
8. Evaluate your model using MAE, MSE and RMSE.

9. Check if there is any improvement from the previous exercise with the simple SVM model.
10. Plot the true vs. predicted house prices.

Model Deployment:

1. Train your SVM model on the entire dataset.
2. Export your final model into a joblib file.
   Make sure you also export other relevant preprocessing instances such as the standard scaler.
3. Import your final model and the preprocessing instances from the joblib files and load them back to your working area.