# Week 6 – Support Vector Machines





Teaching Assistants sarasharish abaksy vighneshkamath

In this week's experiment, you will implement a Support Vector Machine classifier using one of the most popular machine learning frameworks in Python, called scikit-learn.

Scikit-learn is one of the most widely used and fully featured machine learning frameworks that, apart from offering a wide variety of machine learning models, also offers many classes for pre-processing data.

You are expected to use the pre-processing steps of your choice along with the SVM classifier to create a pipeline (explained later) which will be used to automate the entire process of training and evaluating the model you build.

#### You are provided with the following files:

- 1. week6.py
- 2. SampleTest.py
- 3. train.csv
- 4. test.csv

## **Dataset Format**

The dataset consists of 20 features (labelled as x1 to x20) and a target column that is named 'targets'.

The features are all numeric and continuous, hence no encoding is needed of any kind.

The target column 'targets' consists of the output class corresponding to the point X. It is an integer value You do not need to handle missing values in the dataset.

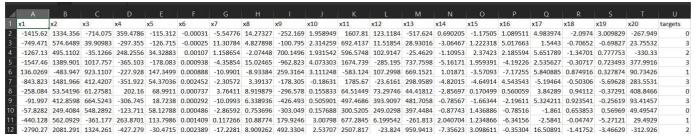


Figure 1: Dataset Format

The entire dataset is split into three parts. Two of these parts are supplied to you

- 1) train.csv is meant for you to train the model on
- 2) test.csv is meant for you to evaluate the accuracy of the trained model on

We will be measuring the performance of your model on a third split called eval.csv. You will be scored based on the accuracy of the model on this unseen validation split of the data.

The scoring will be done as follows:

- Score 10 : accuracy >= 85%
- Score 9: 75% <= accuracy < 85%
- Score 8: 70% <= accuracy < 75%
- Score 7: 65% <= accuracy < 70%
- Score 6: 60% <= accuracy < 65%
- Score 5: 55% <= accuracy < 60%
- Score 4: 50% <= accuracy < 55%
- Score 3: 45% <= accuracy < 50%
- Score 2: 40% <= accuracy < 45%
- Score 1: 35% <= accuracy < 40%
- Score 0: accuracy < 35%

## **Basics of scikit-learn**

The bedrock of scikit-learn is the estimator.

An estimator is any object that learns from data; it may be a classification, regression or clustering algorithm or a **transformer** that extracts/filters useful features from raw data.

Estimators meant for classifying data (such as the SVC class you will use here) implement the following methods, among many more:

- 1) The fit(X, y) method fits the model based on the training data X and y supplied. Since SVM is a supervised algorithm, it requires both the input X and the output labels y as input.
- 2) The predict(Xtest) method takes the test dataset X and returns a NumPy array of the predicted class labels for each point in the test
- 3) The score(Xtest, Ytest) method takes in the test dataset and the true labels and returns the model accuracy.

**Transformers** are used to transform the input dataset into a pre-processed form. They expose the transform() method for transforming the input data.

Transformers for pre-processing the input dataset can be found under the <u>sklearn.preprocessing</u> module. Use the appropriate pre-processing methods to improve the accuracy of your model.

# Scikit-Learn Pipeline

The Pipeline class in the sklearn pipeline library allows one to sequentially apply a list of transforms and a final estimator.

Intermediate steps of the pipeline must be 'transforms', that is, they must implement fit and transform methods. The final estimator only needs to implement the fit method.

The sequence of steps is given to the constructor of the Pipeline class in the form of a list of 2-tuples. The first element of the tuple is the name of the pipeline stage (a string) and the second one is the estimator or transform that is being applied in that stage. Note that is a class **object** and not just a class name.

The pipeline is run by calling the fit() method on the pipeline object.

Check out this tutorial on scikit-learn pipelines.

#### **Important Points:**

- 1. Please do not make changes to the function definitions that are provided to you, as well as the functions that have **already been implemented**. Use the skeleton as it has been given. Also do not make changes to the sample test file provided to you.
- 2. You are free to write any helper functions that can be called in any of these predefined functions given to you. Helper functions must be only in the file named 'YOUR SRN.py'.
- 3. Your code will be auto evaluated by our testing script and our dataset and test cases will not be revealed. Please ensure you take care of all edge cases!
- 4. The experiment is subject to zero tolerance for plagiarism. Your code will be tested for plagiarism against every code of all the sections and if found plagiarized both the receiver and provider will get zero marks without any scope for explanation.
- 5. Kindly do not change variable names or use any other techniques to escape from plagiarism, as the plagiarism checker is able to catch such plagiarism
- 6. Hidden test cases will not be revealed post evaluation.
- 7. Only the SVC model available in the sklearn.svm library is allowed for this experiment (documentation: <a href="here">here</a>). You must not use any other classifiers in the scikit-learn package.
- 8. You can use any kernel of your choice available in the scikit-learn package to improve the test accuracy of your model
- 9. You are only allowed to use the Pipeline available in the sklearn pipeline library and no other pipelines.
- 10. Make sure to use a Pipeline to stack all the pre-processing and estimator stages in the right order. Return the Pipeline object from the solve() method.

### week6.pv

- ✓ You are provided with structure of class SVM.
- ✓ The class SVM contains one constructor and one method.
- ✓ Your task is to write code for the solve() method.
- 1. You may write your own helper functions if needed
- 2. You can import libraries that come built-in with python 3.7
- 3. You cannot change the skeleton of the code
- 4. Note that the target value is an int

## **SampleTest.py**

- 1. This will help you check your code.
- 2. Passing the cases in this does not ensure full marks, you will need to take care of edge cases
- **3.** Name your code file as YOUR\_SRN.py
- 4. Run the command

python3 SampleTest.py --SRN YOUR\_SRN

if import error occurs due to any libraries that is mentioned in the skeleton code try: python3.7 SampleTest.py --SRN YOUR\_SRN