Week 6 – Support Vector Machines

MACHINE INTELLIGENCE LABORATORY- PESU UE19CS305

Teaching Assistants-

vighneshkamath43@gmail.com

sarasharish2000@gmail.com

abaksy@gmail.com

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.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	targets
2	-1415.62	1334.356	-714.075	359.4786	-115.312	-0.00031	-5.54776	14.27327	-252.169	1.958949	1607.81	123.1184	-517.624	0.690205	-1.17505	1.089511	4.983974	-2.0974	3.009829	-267.949	0
3	-749.471	574.6489	39.90983	-297.355	-126.715	-0.00025	11.30784	4.827898	-100.795	2.314259	692.4137	11.51854	28.93016	-3.06467	1.222318	5.017663	1.5443	-0.70652	-0.69827	23.75532	3
4	-1267.13	495.1102	-35.1266	248.2556	34.32883	0.00107	1.158654	-2.07448	700.1496	1.931542	596.5748	102.9147	-25.4629	-1.10953	2.37423	2.185594	5.651789	-1.34701	0.777753	-330.33	2
5	-1547.46	1389.901	1017.757	-365.103	-178.083	0.000938	-4.35854	15.02465	-962.823	4.073303	1674.739	-285.195	737.7598	-5.16171	1.959391	-4.19226	2.535627	-0.30717	0.723493	377.9916	3
6	136.0269	-483.947	923.1107	-227.928	147.3499	0.000888	-10.9901	-8.93384	259.3164	3.111248	-583.124	107.2998	669.1521	1.01871	-3.57093	-7.17255	5.840885	0.874916	0.327874	90.73426	1
7	-843.823	1481.966	412.4207	-351.922	54.37036	0.002452	-2.30572	3.39137	-178.305	-0.18631	1785.67	-23.6161	298.9589	-4.82015	-4.64914	4.543543	-5.19464	-0.50306	-5.69628	283.5531	3
8	-258.084	53.54196	61.27581	202.16	68.9911	0.000737	3.76411	8.919879	-296.578	0.155833	64.51449	73.29746	44.41812	-2.85697	0.170499	0.560059	3.84289	0.94112	-0.37291	408.8466	0
9	-91.997	412.8598	664.5243	-306.745	18.7238	0.000292	-10.0993	6.338936	-426.493	0.505901	497.4686	393.9097	481.7058	-0.78567	-1.66344	-2.19611	5.324211	0.923541	-0.25619	93.41457	3
10	-57.8282	249.4084	548.2892	-123.711	58.12788	0.000486	-2.86592	0.753696	-303.049	0.157688	300.5205	249.0298	397.4484	-0.87743	1.436886	-0.78516	-1.861	0.653853	0.56969	49.49547	0
11	-440.128	562.0929	-361.177	263.8701	113.7986	0.001409	0.117266	10.88774	179.9246	3.00798	677.2845	6.199542	-261.813	2.040704	1.234866	-6.34156	-2.5841	-0.04747	-5.27121	29.4929	1
12	-2790.27	2081.291	1324.261	-427.279	-30.4715	0.002389	-17.2281	8.909262	492.3304	2.53707	2507.817	-23.824	959.9413	-7.35623	3.098611	-0.35304	16.50891	-1.41752	-3.46629	-312.926	2

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 data
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 sklearn.preprocessing 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: here). 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.py

- ✓ 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

SUBMISSION FORMAT and DEADLINE:

You are required to complete code in week6.py and **rename it to SRN.py**

Failing to submit in the right format will lead to zero marks without a chance for correction

Example: If your SRN is PES1201801819 your file should be PES1201801819.py

- Delete all print statements if used for debugging before submission
- Ensure you run your file against sample test before submitting
- Check for syntax and indentation errors in your file, and make sure that tabs and spaces are used uniformly for indentation (i.e., if tabs are used then only tabs must be used throughout the file, and similarly for spaces)
- All the helper functions should be in the same file (SRN.py), make sure you run the sample test cases as indicated above before submitting.

Deadline: on or before 17/10/2021 10:00 PM

Where to submit: Edmodo