## Assignment 1 (20% of total course weight) – Machine Learning

**California State University San Bernardino, School of Computer Science and Engineering (CSE)**

**Date of Issue**: November 20, 2023, **Date of submission**: **December 1, 2023 – 11:59 pm (PST)**

**Course**: CSE 5120 Introduction to Artificial Intelligence

**Assessment brief:** In the early 2000s, the support vector machine (SVM) model class was the most popular approach for “off-the-shelf” supervised learning, for the cases when no specialized prior knowledge about a domain is known. An SVM aims at devising a computationally efficient way of segregating different data classes with a hyper-plane that best differentiates the classes. SVM selects this hyper-plane (i.e., the maximum margin separator) from the hypothesis space that maximizes the distance to data points of either class. The support vectors are the data points that are closest to the separator constructed by the SVM model (three circled points in **Figure** below).



***Figure:*** *SVM classification of two class problem (data points on the left and right of h0 represent two classes, class 1 and 2, respectively) with three candidate linear separators (h0, h1 and h2). The maximum margin separator (h0) is the middle point of the margin (area between h1 and h2). In the dual representation equation (right),* *is the Lagrange dual variable associated with each data point, is the class variable which can be +1 and -1 for samples with positive and negative examples, respectively. are the data samples that represent a sample class, is the weight vector, is the intercept, and circled data points are the support vectors that touch the boundary of the margin.*

SVMs have three attractive properties:

1. SVMs construct a maximum margin separator—a decision boundary with the largest possible distance to example points. This helps them generalize well.
2. SVMs can not only create a linear separating hyperplane, but they also have the ability to embed the data into a higher-dimensional space, using the kernel trick. Often, data that are not linearly separable in the original input space are easily separable in the higher-dimensional space.
3. SVMs are nonparametric—the separating hyperplane is defined by a set of example points, not by a collection of parameter values. An SVM model keeps only the examples that are closest to the separating plane—usually only a small constant time the number of dimensions. Thus, SVMs combine the advantages of nonparametric and parametric models: they have the flexibility to represent complex functions, but they are resistant to overfitting.

**Scikit-Learn libraries for SVM implementation.**

Scikit-Learn provides useful libraries to implement SVM algorithms. As discussed in the class, we will use the built-in libraries for SVM to implement the solution to a problem which will be a classification problem. First, we will use a LinearSVC() classifier. As its name suggests, LinearSVC() uses a linear kernel which can either be passed as kernel = ‘linear’ in the input argument to the above function or leaving the kernel undeclared.

Scikit-Learn provides two other classifiers - SVC() and NuSVC() which are used for classification purposes. Compared with SVC(), NuSVC() uses a parameter to control the number of support vectors. By default, the kernel parameter is set to RBF along with other values such as poly, linear, sigmoid or a callable function.

**Dataset description – Predicting a pulsar star**

Pulsar is a rare type of Neutron star that produces radio emission detectable on Earth. Pulsars are helpful probes for space-time, the inter-stellar medium, and states of matter. In this dataset, class imbalance exists where samples for pulsar examples are of minority positive class and spurious examples are of the majority negative class.

There are a total of 16,259 and 1,639 samples of spurious examples of negative class and real pulsars, respectively. There are 8 predictor variables in the dataset with the final column representing the binary class, represented as 0 (negative) and 1 (positive).

**Attribute Information:**

Each sample in the dataset is described by 8 continuous variables, and one class variable. The first four variables are the statistics obtained from the integrated pulse profile. The remaining four variables are similarly obtained from the SNR curve. The variables as defined in the dataset are listed below:

1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.
9. Class

**Assignment description:**

The code for this assignment consists of 2 Python files (pulsar\_classification.py and evaluation.py) which you will need to read and understand in order to complete the assignment. You can download all the supporting code and files as a **zip folder** from Assignment 1 link given on Canvas (**pulsarClassification.zip**).

Your assignment is based on two parts as given below:

1. Code implemented for pulsar classification via support vector machines in given pulsar\_classification.py and evaluation.py files (in specific sections as indicated in detail below)
2. A report on what you did for your classifier (i.e., how you implemented with screenshots from evaluation.py script given in the folder)

|  |  |
| --- | --- |
| **File Name** | **Description** |
| Pulsar\_classification.py | Where all of your support vector classifier code for pulsar dataset will reside. In this file, you only need to implement the initialization and training of your classifier as given in the assignment details section. You should print confusion matrix, precision, recall, specificity, and accuracy score for this dataset and copy it in your assignment report. |
| evaluation.py | The evaluation file where you will implement your code to load your classifier, run it on test cases for the selected dataset, and print the precision, recall, specificity, confusion matrix and accuracy score. |
| pulsarClassifier.sav | This will be the trained model (SVM) on ***pulsar*** dataset that you will produce in your pulsar\_classification.py file using Pickle library (see description in the following section) |
| Assignment Report.doc | Report explaining your steps for all the above work with description and screenshots. See below for more details |

You can use Spyder (installed through Anaconda from week 1 lecture) or other IDE.

**Files to Edit and Submit:** You will need to edit and submit (pulsar\_classification.py, evaluation.py) files to implement your model for the pulsar dataset. You can copy and paste all the necessary pieces of code that we wrote in the class. Once you have completed the implementation of your classifier, you should save your classifier for the pulsar dataset with the name “pulsarClassifier.sav” and upload it in Zipped folder in your assignment. To save and load models, you can use “pickle” library from scikit learn, as Pickle is the standard way of serializing objects in Python. You are welcome to search for “save and load machine learning models using pickle” via Google or your favorite search engine for help in implementing this part. Once completed your pulsar\_classification.py file, you will need to implement the code in evaluation.py file to load your classifier (i.e., pulsarClassifier.sav) via Pickle, **predict the classes for the test cases using 80%, 20% split in** evaluation.py and print {precision, recall, specificity, confusion matrix, accuracy score} as given in the pulsar\_classification.py file for help. For confusion matrix and accuracy scores, please refer to the code we covered in Week 12 for implementing Support Vector Machines and Decision Tree Algorithms. You will need to test your classifier with evaluation.py to copy screenshots in your report.

**Note:** You are welcome to use any existing code from the class to generate any other interesting visualizations of either the data itself or the model predictions (similar to how we visualized the results in the class)

**Academic Dishonesty:** Your code will be checked against other submissions in the class for logical redundancy. If you copy someone else’s code and submit it with minor changes, they will be detected, so please do not try that, and submit your own work only. In case of cheating, the University’s academic policies on cheating and dishonesty will strictly apply which may result from the deduction in your grade to expulsion.

**Getting Help:** If you are having difficulty in implementing the classifier, contact course staff for help and refer to the content recorded and uploaded in Week 12 part 2 on Machine Learning). Office hours and Slack are there for your support. If you are not able to attend office hours, then please inform your instructor to arrange for additional time. The intent is to make this assignment rewarding and instructional, not frustrating and demoralizing. You can either complete this assignment on your own or discuss the problem and collaborate with another member of the class (or different section). Please clearly acknowledge and mention your group member in your assignment report submission who you will collaborate with in this assignment. Your report and program (classifier.py, evaluation.py files and pulsarClassifier.sav model) will be separately submitted by yourself via Canvas irrespective of your collaboration with your group member. Group discussions are encouraged but copying of programs is NOT recommended. Programming based on your own skills is encouraged.

**Assignment Details**

1. **Classifier pulsar\_classification.py file - 12%**

This dataset is created to implement your code for a test classifier. The following steps will be followed to build a support vector machine classifier (pulsar\_classification.py) for Pulsar dataset:

1. Load dataset
2. Train and test split (80%, 20%)
3. Data preprocessing: Standardization and any other data cleaning needed.
4. Training and testing support vector machine classifier
5. **Evaluation evaluation.py file**

You will copy the code in your evaluation.py file to only load your saved model from section 1, use the functions created in pulsar\_classification.py files to generate and load test dataset, make model predictions for this dataset and then finally print precision, recall, specificity, confusion matrix and accuracy score. Use these metrics to copy in your assignment report section 3.

**Note:** For this work, please write a description of your work in the report (2-page maximum) that may include the use of SVC model instead of LinearSVC and your observations for this work.

1. **Assignment Report – 8%**

Your report will explain your observations and work for the Pulsar dataset. Please use screenshots (or manually edited versions) of your model performance metrics (precision, recall, specificity, confusion matrices, accuracy score) to support your explanation.

**Assignment 1** (20%)

CSE 5120 (Section - 01) – Introduction to Artificial Intelligence – Fall 2023

*Submitted to*

Department of Computer Science and Engineering  
California State University, San Bernardino, California

*by*

Student name (CSUSB ID)

(Your collaborator in this homework (if any))

Date: *Month Day, Year*

*Email:*

* *Your email*
* *Your collaborator’s email (if you collaborated with any)*

**Assignment Report**

A brief description of your work here acknowledging your collaboration with your class fellow (or a friend from other CSE 5120 sections), and the capacity at which he/she collaborated with you, followed by the algorithms you implemented.

1. **Pulsar\_classification.py for Pulsar dataset**

Your explanation of the dataset, your code solution, and any documentation with screenshots of your code Evaluation (results from **pulsar\_classification**.py)

1. **Evaluation (evaluation.py) for your model performance evaluation - optional**

You can also provide the description of your code written in evaluation.py to load the saved model that can be readily used on test dataset for the staff. This section is optional, and you can skip it