CS 5805: Machine Learning I

Spring 2025

Homework 4

Date Assigned: 2/26/2025

Date Due: 3/7/2025

1. (5 points) Let us do k-NN classification for a given dataset where the data points P_1, P_2, …, P_n are in d-dimensional space, i.e., each point P_i is given by (x_1, x_2, …, x_d). Assume you are given a black-box function f() that takes as input a new, unlabeled point P_z and returns the nearest neighbor of P_z as well as that neighbor's corresponding class label. Thus the black-box function f() is essentially performing 1-nearest neighbor classification. Is it possible to construct a k-NN classifier (where k is greater than 1) using this black-box function f() alone? If yes, explain how. If not, explain why not.

2. (5 points) Same problem as above but instead of f() returning one single nearest neighbor, let us assume it returns "m" nearest neighbors. Again, the question is: is it possible to construct a k-NN classifier (where k is not equal to m) using this black-box function f alone? If yes, explain how. If not, explain why not.

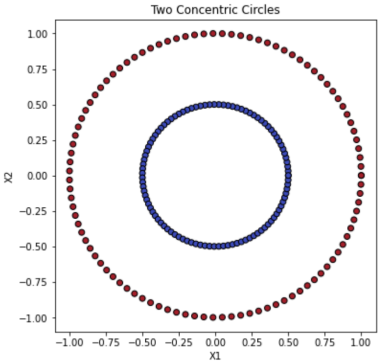
3. (10 points) Assume you are given the following dataset:

x	y	class
1	0.5	c_1
3	0.5	c_1
4	0.5	c_1
2	1.5	c_1
1	2.5	c_1
3	2	c_1
3	4.5	c_2
4	4.5	c_2
4.5	3	c_2
6	3	c_2
6	4.5	c_2

For this data, plot the given points on a 2D plane, and find the linear hyperplane that maximizes the margin of separation between the two classes. You can either calculate the hyperplane analytically (manually) or using Python libraries or some combination.

For full credit, i) identify the support vectors (how many are there? Which ones are they? Which class does each support vector support?) ii) write out the equation of the optimal hyperplane, and iii) compute the margin, and iv) suggest a way to impute confidence to the SVM's output, i.e., when the SVM makes a classification, how will you assign a confidence score to the classification? What approach will you take?

4. (20 points) Recall the two concentric circles dataset from HW2 (notebook to generate it was supplied with that assignment - 2concentriccircles.ipynb). It looked like this:



Three students are arguing about how to solve this problem using a linear support vector machine (SVM) by employing a suitable kernel that will transform the given features to a new space and find linear separators in that space.

a. Student X is saying she will use this feature mapping:

$$\phi(x,y) = \left[1,\ \sqrt{2}\,x,\ \sqrt{2}\,y,\ x^2,\ \sqrt{2}\,x\,y,\ y^2\right].$$

b. Student Y is saying he will use this feature mapping:

$$\phi(x,y) \ = \ \left[x^2, \sqrt{2}\,x\,y, y^2\right].$$

c. Student Z is saying no real mapping is needed, ie:

$$\phi(x,y) = [x,y].$$

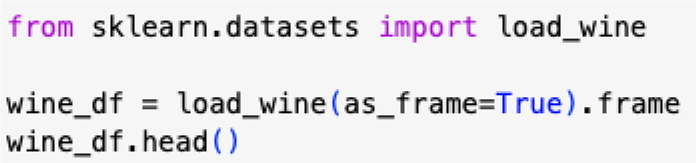
will work. Because the SVM implicitly does dot products, student Z argues that explicit product terms like "xy" are needed.

Who is right? Who is wrong? For each student explain whether they are right or wrong and give a 1-3 sentence explanation. You can write code if you would like to verify/confirm/test your answer (but you don't have to).

5. (30 points) Consider the Wine dataset at (https://archive.ics.uci.edu/dataset/109/wine). This dataset contains the results of chemical analysis of wines grown in the same region of Italy, originating from three different cultivars. The analysis measures the quantities of 13 chemical constituents found in each type of wine, with the "class" variable labeled from 0 to 2. In food science, collecting such data is often expensive, making it challenging to obtain large

datasets for training machine learning algorithms. In this task, you will build both Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) models to understand how they use the given data as prototypes (To understand what a “prototype” is, see lecture slides or video).

Step 1: Acquire the data using the following code. Choose a suitable number of folds for k-fold cross-validation. Explain why your choice of k is appropriate for the given dataset.



```
from sklearn.datasets import load_wine

wine_df = load_wine(as_frame=True).frame
wine_df.head()
```

Step 2: Select two features, **‘alcohol’** and **‘malic_acid’**, from the dataset. We will stick with just these two features for the rest of the assignment.

Step3: Perform basic visualizations with both marginal distributions and 2D joint distributions.

Step 4: Based on these visualizations, formulate assumptions about how these features can be used for classification. Explain your assumptions by interpreting patterns and separability observed in the data visualization.

Step 5: Construct the following three ML models and evaluate their classification performance using four metrics: **precision, recall, F1-score, and accuracy**. Additionally, plot the **decision boundaries** for each model (including support vectors for SVM).

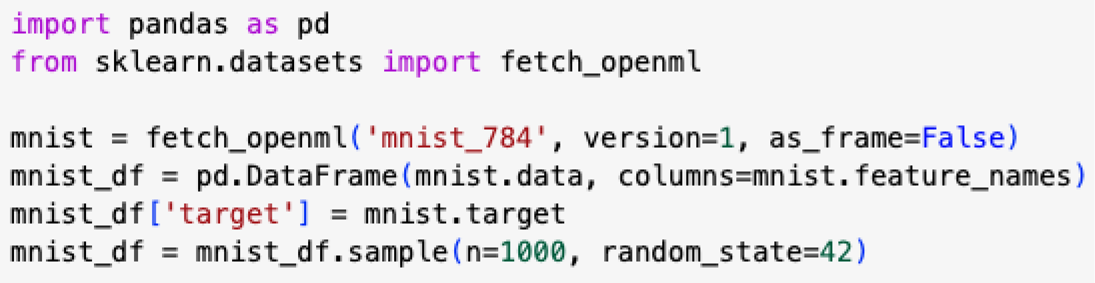
a) Train an SVM model with a linear kernel.
b) Train an SVM model with an RBF (Gaussian) kernel.
c) Train a K-NN model.

Step 6: Analyze whether the results from Step 5 support the assumptions made in Step 4. Explain how each model performs the classification task, discussing their strengths and limitations.

Step 7: If the models classify the data well, describe any equivalence between them in terms of decision boundaries and classification performance.

6. (30 points) Now, let’s move on to a high-dimensional classification task: the MNIST handwritten digit classification. This is a well-known dataset consisting of 70,000 handwritten digit images, each belonging to one of 10 class labels (0-9). Each image is represented as a 28 × 28 grayscale pixel array.
(https://archive.ics.uci.edu/dataset/683/mnist+database+of+handwritten+digits).

   Step 1: Acquire the data using the following code and sample 1,000 rows, which is sufficient for this assignment. Choose a suitable number of folds for k-fold cross-validation. Explain why your choice of k is appropriate for the given dataset.



```python
import pandas as pd
from sklearn.datasets import fetch_openml

mnist = fetch_openml('mnist_784', version=1, as_frame=False)
mnist_df = pd.DataFrame(mnist.data, columns=mnist.feature_names)
mnist_df['target'] = mnist.target
mnist_df = mnist_df.sample(n=1000, random_state=42)
```

Step 2: Perform basic visualizations and provide an analysis. For example, but not limited to, you can try plotting the pixel-wise distribution for different classes.

Step 3: Based on these visualizations, formulate assumptions about how these features can be used for classification. Explain your assumptions by interpreting the observed patterns and the degree of separability in the data.

Step 4: Construct the following three ML models and evaluate their classification performance using four metrics: **precision, recall, F1-score, and accuracy**. Additionally, plot the **decision boundaries** for each model (including support vectors for SVM).

a) Train an SVM model with a linear kernel.
b) Train an SVM model with an RBF (Gaussian) kernel.
c) Train a K-NN model.

Step 5: Analyze whether the results from Step 4 support the assumptions made in Step 3. Explain how each model performs the classification task, discussing their strengths and limitations.

Step 6: If the models classify the data well, discuss what each metric and decision boundaries measure, their relevance to classification in this domain, and any limitations they might have in this context.