

MATH60629A. Homework.

Due date: November 3, 2026

Instructions:

- Please include your name and HEC ID with submission.
- The homework is due by 11:59pm on the due date. Please upload a PDF version of your assignment on ZoneCours.
- The homework is worth 15% of the course's final grade.
- Assignments are to be done individually.

1 Machine Learning principles (3pt)

Your answers to these questions should be short (max. 5 lines).

1. (1pt) Explain the difference between the training error and the generalization error. Make sure to describe how to estimate the generalization error of a model in practice including pitfalls of this approach (in other words, describe conditions under which this approach is reliable).
2. (1pt) Explain the effect of varying the regularization strength (for example when using Ridge regularization) on the bias-variance trade off.
3. (1pt) A colleague trained a model to predict the median price of houses using Montreal-housing data. The resulting model works well according to their validation data. However, this model performs poorly when used to predict the prices of houses in the Toronto housing market. Explain why that is and suggest a way of obtaining a better model for the Toronto data.

2 Classification (16pt)

- We will use a synthetic dataset that I created. It is available [here](#).
Once the data are accessible from your current working directory, you can load them using the following code:

```
data = np.load("a24_devoir_q2-classification.npz")
X = data["X"]
y = data["y"]
```

The data are encoded in a *numpy array*.

The task at hand is to predict the class of each datum from its two features.

1. (4pt) Following an initial data exploration, what do you notice? What can you say about the approximate test performance (in terms of accuracy) if using a linear model?
2. (2pt) Divide your dataset into a training and a test set. The test set must make up 20% of the total original dataset. Make sure to use this parameter upon calling the appropriate sklearn function: `random_state=123`. We ask that you provide the few lines of code you used to divide the data.
3. (1pt) What does the fact that we do not keep a validation set tell you?

You will now explore the Naive Bayes model.

4. (1pt) Which Naive Bayes classifier should you use? In other words, what is a reasonable distribution for the class conditional densities? Justify your answer.
5. (2pt) Train the model on the training data and obtain its accuracy for both train and test. Please provide the few lines of code to do these and provide the resulting train and test accuracies.
6. (4pt) Re-train the model using uniform class priors (provide the code for doing so). What do you notice in terms of the training and test accuracies? Explain this result.
7. (2pt) Do you think this observation is likely to hold in general?

3 Regression (32pt)

You will now train a k-NN and a neural network model for the task of predicting the rating of a text review.

The data to download are [here](#). Each datum is a review in text format of an Amazon product.¹

In the data file, each line corresponds to a datum. Each datum contains a target (y) followed by a short text (x). The target variable is the rating given by a user to a product. It is an integer value between 1 and 5. The text is the

¹For your information, the complete datasets are available [here](#). We use a subset of the *Instruments* category.

review. To pre-process the data you will first have to separate the targets from the features.

Hint: you can use the csv library to load this data. For example:

```
import csv

with open('instruments.csv', mode='r') as file:
    reader = csv.reader(file)
    for i, row in enumerate(reader):
        print(i, row)
```

1. (2pt) We will model this task as a regression problem (and so you can use mean squared error to measure performance). List one advantage and one disadvantage of instead modelling the problem as a classification problem with 5 classes.

Data pre-processing (5pt)

2. (2pt) First divide the dataset into training (80% of the data), validation (10%), and test sets (10%). For this set the random seed to 1234 (`random_state=1234`)

Provide the few lines of code you used to divide the data.

3. (3pt) Now you must obtain a bag-of-words representation of the features. sklearn provides several [functions](#) for doing so.

To limit the required training time, please use a maximum of 2,000 words in your vocabulary (`max_features=2000`) and the list of English stop words from sklearn (`stop_words="english"`). Words on this list will be automatically removed from the data since they are, a priori, less predictive for the task at hand. Use the default value for all other function parameters.

Provide the few lines of sklearn code you used to encode (and only those) the training, validation, and test data.

K-nearest neighbours (13pt)

4. (3pt) Which of the following three distance functions 'cosine', 'euclidean', and 'manhattan' do you deem more appropriate for this problem? Please justify.
5. (5pt) Given your previous answer, train an adequate k-NN model for this task. We ask that you train models with 1, 10, 50, 100, and 1000 neighbours.

What is the performance of each model on the training and validation sets?

Provide the few lines of code that you used to train this model and evaluate its performance

6. (5pt) What value of the hyperparameter provides the best results? Explain.

Neural networks (10pt)

Upon instantiating your neural networks, fix the random seed to 1234 (that is `random_state=1234`).

7. (5pt) You will now train a series of neural networks using different hyperparameters. Use the option `early_stopping=True` and find the hyperparameters that obtain the best results on the validation dataset (to give you an idea, training around 20 different models is sufficient). I suggest that you explore the following three hyperparameters: learning rate, size of the network, and the strength of the L2 regularization term.

In your answer, please include the code used to train these networks and to obtain the performance on the validation set (and only that part of the code).

What is the best combination of hyperparameters and what is their performance on the validation set?

8. (5pt) What did you find out about the importance of the various hyperparameters?

Comparison (2pt)

9. (2pt) What is the final performance of each model (k-NN and neural network)? *Please provide the few lines of code you used for this.*

4 Pre-trained representations (6pt)

The bag-of-words representation of Section 3 ignores word order and treats words as unrelated symbols. Modern ML systems instead often reuse representations learned by models pre-trained on large corpora (recall the capsule on learning representations).

We provide a pre-computed *sentence embedding* for each review: [instruments_embeddings.npz](#). Row i of the matrix is a 384-dimensional embedding of the i -th review of `instruments.csv`, computed with the pre-trained model `all-MiniLM-L6-v2`.² You can load them with:

```
E = np.load("instruments_embeddings.npz")["E"]
```

Use the same training/validation/test split as in Section 3 (`random_state=1234`) so that the rows remain aligned with your targets.

²The embeddings were generated with: `model = SentenceTransformer("all-MiniLM-L6-v2"); E = model.encode(texts); np.savez("instruments_embeddings.npz", E=E)`.

1. (3pt) Re-train your k-NN model (you may re-select k over the same grid as before, and briefly revisit your choice of distance function) and your best neural-network configuration from Section 3, this time using the embeddings as features. Report the validation and test MSEs. *Provide the few lines of code you used.*
2. (3pt) Compare these results with the bag-of-words results and explain the difference (or its absence). In particular: how can a 384-dimensional pre-trained representation compete with, or even beat, a 2,000-dimensional bag-of-words representation? Connect your answer to the idea of learning representations seen in class.

5 Auditing a generated solution (6pt)

A student asked an LLM to solve a task similar to that of Section 3 (predicting review ratings from text) and received the solution excerpted below. The code runs without error and the explanation reads fluently. It nonetheless contains **three conceptual mistakes**.

```
texts, y = load_reviews("reviews.csv")

vectorizer = CountVectorizer(max_features=2000, stop_words="english")
X = vectorizer.fit_transform(texts)    # encode the full dataset

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1234)

best_k, best_mse = None, np.inf
for k in [1, 10, 50, 100]:
    model = KNeighborsRegressor(n_neighbors=k).fit(X_train, y_train)
    mse = mean_squared_error(y_test, model.predict(X_test))
    if mse < best_mse:
        best_k, best_mse = k, mse

print("Final model: k =", best_k, "with MSE =", best_mse)
```

The accompanying explanation reads:

“We select the value of k with the lowest error on the test set and report that error as the final performance of the model. Among the values tried, $k = 1$ often performs best, since using a single neighbour minimizes the variance of the predictor.”

For each of the three mistakes: (a) identify it, (b) explain its consequence on the reported results or on the conclusions, and (c) explain how to fix it. (2pt per mistake)

6 Probing an online system (12pt)

You are tasked with studying an online system that could be making use of machine learning for making predictions (of course, you might not know for sure whether it *does* or does not). Rather than describing the system in general terms, you will *probe* it: form hypotheses about where it should fail, and test them with a small, systematic set of interactions.

Choose a system with a public interface that you can interact with directly. Pre-approved choices: [DeepL](#) (translation); an OCR tool (e.g., Google Lens); a dictation / speech-to-text tool; Shazam (music recognition); an image-tagging demo (e.g., on Hugging Face Spaces); the [Perspective API](#) demo (toxicity scoring); the next-word prediction of your phone's keyboard. You may propose another system, provided you can interact with it directly (when in doubt, ask us). **Excluded:** [Google Translate](#) (the course's running example), general-purpose chatbots (ChatGPT, Claude, Gemini, ...), and systems you cannot probe directly (e.g., recommendation feeds such as Netflix's or TikTok's).

1. (3pt) Describe the ML task the system is trying to solve (inputs and outputs), the data (or its type) that would be needed to train this model, and a suitable performance measure for it (you don't necessarily need to use a measure that exists).
2. (3pt) State two hypothesized failure modes, each grounded in a concept from the course (e.g., distribution shift, rare inputs / data sparsity, spurious correlations, mismatch between the training objective and user needs). For each, briefly justify why you expect it for this system.
3. (4pt) Design and run 6 to 10 interactions that test your hypotheses. Report them in a table: input, system output, hypothesis being tested, and whether the outcome supports or contradicts it. If the system restricts access in a way that blocks one of your tests, document the restriction: it is itself a finding about deployed ML systems.
4. (2pt) For each hypothesis, conclude (confirmed, rejected, or inconclusive). For the confirmed failure modes, given your machine learning knowledge, how would you try to correct them?