# PROGRAMMING ASSIGNMENT 2: $k$NN & MODEL SELECTION

10-301/10-601 Introduction to Machine Learning (Summer 2023)
https://www.cs.cmu.edu/~hchai2/courses/10601/
OUT: Thursday, May 25th
DUE: Thursday, June 1st
TAs: Alex, Andrew, Sofia, Tara, Markov, Neural the Narwhal

**Summary** In this assignment, you will build a $k$NN classifier and apply it to several binary classification problems, with the goal finding the best value of $k$ using both validation and cross-validation. You will implement $k$NN models using Euclidean distance and Manhattan distance, plot training and (cross-)validation error rates as a function of $k$, and finally compare the effect of "re-training" your models on both the training *and* validation datasets as opposed to just using the training dataset.

## START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: https://www.cs.cmu.edu/~hchai2/courses/10601/
- **Late Submission Policy:** See the late submission policy here: https://www.cs.cmu.edu/~hchai2/courses/10601/
- **Submitting your work:**
  - **Programming:** You will submit your code for programming questions on the homework to Gradescope (https://gradescope.com). After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). When you are developing, check that the version number of the programming language environment (e.g. Python 3.9.12) and versions of permitted libraries (e.g. `numpy` 1.23.0) match those used on Gradescope. You have a **total of 10 Gradescope programming submissions.** Use them wisely. In order to not waste code submissions, we recommend debugging your implementation on your local machine (or the linux servers) and making sure your code is running correctly first before any Gradescope coding submission.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please use the provided template. You must typeset your submission using LaTeX. If your submission is misaligned with the template, there will be a **2% penalty** (e.g., if the homework is out of 100 points, 2 points will be deducted from your final score). Each derivation/proof should be completed in the boxes provided. Do not move or resize any of the answer boxes. If you do not follow the template, your assignment may not be graded correctly by our AI assisted grader.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeX users, replace `\choice` with `\CorrectChoice` to obtain a shaded box/circle, and don't change anything else.

## Instructions for Specific Problem Types

For “Select One” questions, please fill in the appropriate bubble completely:

**Select One:** Who taught this course?

- ● Henry Chai
- ○ Marie Curie
- ○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

**Select One:** Who taught this course?

- ● Henry Chai
- ○ Marie Curie
- ⊗ Noam Chomsky

For “Select all that apply” questions, please fill in all appropriate squares completely:

**Select all that apply:** Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- □ I don’t know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

**Select all that apply:** Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- ☒ I don’t know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

**Fill in the blank:** What is the course number?

10-601

10-~~6~~301

# Written Problems (10 points)

## 1 Empirical Questions (10 points)

The following questions should be completed as you work through the programming portion of this assignment.

1. (3 points) For the `iris` dataset, create a *computer-generated* plot showing error rate on the y-axis against $k$ on the x-axis for $k$ going from $1$ to $90$ (the number of training data points). Use the Euclidean distance metric for this question.

   On a single set of axes, include *both* training error rate and validation error rate, clearly labeling which is which. That is, for each possible value of $k$ from $1$ to $90$, you should report the error rate of a $k$NN classifier's predictions on both the training dataset and validation datasets given to you. You should include an image file below using the provided, commented out code in LaTeX, switching out `iris.png` to your file name as needed.

Plot

2. (4 points) Using the `iris` dataset, compare the effect of "training" your $k$NN model on just the training dataset vs. using both the training and validation datasets by computing the test error rate in both settings. Populate the table below for the pre-specified values of $k$. (Please round each number to the fourth decimal place, e.g. 0.1234)

   **NOTE:** you should *absolutely* not do something like this in practice; comparing different values of $k$ should only be done using the validation dataset. If you were to use a table like this to inform your selection of $k$, your reported test error would no longer be a reliable estimate of your $k$NN classifier's true error; a good rule of thumb is that a test dataset should only be used *once* in the entire pipeline. That being said, we are asking you to complete this task because it illustrates the fact that test error rate tends to decrease when models are trained on more data, regardless of the value of $k$.

| $k$ | **Test Error Rate (Train Only)** | **Test Error Rate (Train + Validation)** |
|---|---|---|
| 1 | | |
| 20 | | |
| 40 | | |
| 60 | | |

3. (3 points) For the iris dataset, create a *computer-generated* plot showing 10-fold cross-validation error on the y-axis against $k$ on the x-axis for $k$ going from 1 to 90 (the number of training data points). For this question, you will use both the Euclidean distance *and* the Manhattan distance to compute the cross-validation errors.

   On a single set of axes, include the 10-fold cross-validation computed using *both* the Euclidean distance and the Manhattan distance, clearly labeling which is which. That is, for each possible value of $k$ from 1 to 90, you should split the training dataset into 10 equal-sized chunks or "folds". Then compute the average error rate when each fold is used as the validation dataset for a $k$NN model "trained" on the remaining 9 folds with the Euclidean distance metric. Repeat this procedure with the Manhattan distance metric. You should include an image file below using the provided, commented out code in LaTeX, switching out crossval.png to your file name as needed.

Plot

## 2 Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details.
3. Did you find or come across code that implements any part of this assignment? If so, include full details.

Your Answer

# 3 Programming (88 points)

Your goal in this assignment is to implement a classifier based on distance between points - specifically a $k$NN model. In addition, we will ask you to run some end-to-end experiments selecting the best value for $k$ using the classic Fisher iris dataset. You will write one program: `knn.py`. The program you write will be automatically graded using Gradescope.

## 3.1 The Tasks and Datasets

**Materials** Download the zip file from the course website. The zip file will have a handout folder that contains all the data that you will need in order to complete this assignment.

**Starter Code** The handout will contain a preexisting `knn.py` file that itself contains some starter code for the assignment. While we do not require that you use the starter code in your final submission, we *heavily* recommend building upon the structure laid out in the starter code.

**Datasets** The handout contains one dataset, which has already been split into training, validation and test datasets. The first line of each `.csv` file contains the name of each attribute, and *the class label is always the last column.*

The task associated with this dataset is to predict the species of an iris, based on physical measurements. Note that in this dataset, there are three possible labels: *iris setosa* $(0)$, *iris versicolor* $(1)$, and *iris virginica* $(2)$. The attributes (aka. features) are:

1. `sepal length`: The length of the sepal in cm.
2. `sepal width`: The width of the sepal in cm.
3. `petal length`: The length of the petals in cm.
4. `petal width`: The width of the petals in cm.

The training data is in `iris_train.csv`, the validation data is in `iris_val.csv`, and the test data is in `iris_test.csv`.

## 3.2 Program #1: $k$NN 88 points)

In `knn.py`, you will implement a $k$ nearest neighbors classifier.

**Your implementation must satisfy the following requirements:**

- Use Euclidian or Manhattan distance (as specified) to calculate distances between points.
- Return predictions for a range of $k$ values as specified on the command line.
- Compute the error of a set of predictions given the true labels.
- Perform validation and 10-fold cross-validation; **NOTE:** the number of folds is not a command line argument, you may hard code in a value of 10 in your implementations.
- Break ties in distance between training data points in favor of data points that appear *earlier* in the training dataset i.e., ones with lower row indices. For example, if a data point is equally close to the 4th and the 10th training data point, then its nearest neighbor would be the 4th training data point.
- Break ties in labels of the nearest neighbors in favor of lower label values. For example, if the $5$ nearest neighbors to some data point have labels $0$, $0$, $1$, $2$, and $2$, then the predicted label would be $0$.
- Do not hard-code any aspects of the datasets into your code. We may autograde your programs on hidden datasets that include different attributes and output labels.

Careful planning will help you to correctly and concisely implement your $k$NN model. Here are a few *hints* to get you started:

- Write and test helper functions to calculate the two distance metrics.
- Be sure to correctly handle the case where the specified number of neighbors is greater than the total number of points.
- When calling the cross-validation method, you should combine the training and validation datasets.
- Also be sure to correctly handle the case where the number of folds does not divide evenly into the number of data points.
- Look under the FAQ post on Piazza for more useful clarifications about the assignment.

### 3.3 Command Line Arguments

The autograder runs and evaluates the output from the files generated, using the following command:

```
$ python knn.py [args...]
```

Where above `[args...]` is a placeholder for eleven command-line arguments, described in detail below:

1. `<train input>`: path to the training input `.csv` file (see Section 3.1)
2. `<val input>`: path to the validation input `.csv` file (see Section 3.1)
3. `<test input>`: path to the test input `.csv` file (see Section 3.1)
4. `<val type>`: whether we use "normal" validation (0) or cross-validation (1).
5. `<dist metric>`: whether we use Euclidean (0) or Manhattan (1) distance.
6. `<min k>`: The smallest value of $k$ to consider.
7. `<max k>`: The largest value of $k$ to consider.
8. `<train out>`: path of output `.txt` file to which the predictions on the *training* data should be written (see Section 3.4)
9. `<val out>`: path of output `.txt` file to which the predictions on the *validation* data should be written (see Section 3.4)
10. `<test out>`: path of output `.txt` file to which the predictions on the *test* data should be written (see Section 3.4)
11. `<metrics out>`: path of the output `.txt` file to which metrics such as train and test error should be written (see Section 3.5)

As an example, the following command line would run your program on the iris dataset and learn a $k$NN classifier with the Manhattan distance metric, using 10-fold cross-validation to find the best of $k$ from 10 to 20. The train predictions would be written to `iris_10to20_train.txt`, the validation predictions to `iris_10to20_val.txt`, the test predictions to `iris_10to20_test.txt`, and the metrics to `iris_10to20_metrics.txt`.

```
$ python knn.py iris_train.csv iris_val.csv iris_test.csv 1 1 10 20 \
iris_10to20_train.txt iris_10to20_val.txt iris_10to20_test.txt \
iris_10to20_metrics.txt
```

### 3.4 Output: Labels Files

Your program should write *either two or three* output `.txt` files depending on the `<val type>` argument.

1. If `<val type>` is 0 (“normal” validation), then you should output the predictions of your model on the training dataset (`<train out>`), validation dataset (`<val out>`), and test dataset (`<test out>`). `<train out>` and `<val out>` should contain an array where the rows correspond to the input values of $k$ and the columns correspond to the predicted labels for each data point. `<test out>` should contain a single list of predictions on the test dataset corresponding to the best value of $k$ i.e., the one with the lowest validation error rate. Columns in a row should be comma separated. Use ‘`\n`’ to create a new line.

   For example, if you set `<min k>` to 10 and `<max k>` to 20, then `<train out>` would have 11 rows and 90 columns (as there are 90 training data points); the 3rd entry in the 2nd row would be the prediction of an 11NN model on the 3rd training data point.

   The first few rows and columns of a sample `<train out>` output file are given below for `<val type>` 0, where $k$ ranges from 1 to 5 and the Euclidean distance metric is used:

   ```
   2,1,2,0,...
   2,1,1,0,...
   2,1,2,0,...
   ...
   ```

2. If `<val type>` is 1 (10-fold cross-validation), then you should output the predictions of your model on each fold of the combined training and validation datasets when that fold is held out (`<val out>`) as well as the predictions on the test dataset (`<test out>`). `<val out>` should contain an array where the rows correspond to the input values of $k$ and the columns correspond to the predicted labels for each data point when that data point is not used to train the model. `<test out>` should contain a single list of predictions on the test dataset corresponding to the best value of $k$ i.e., the one with the lowest cross-validation error rate. Columns in a row should be comma separated. Use ‘`\n`’ to create a new line.

   For example, if you set `<min k>` to 10 and `<max k>` to 20, then `<val out>` would have 11 rows and 120 columns (as there are 120 total training and validataion data points); the 3rd entry in the 2nd row would be the prediction of an 11NN model on the 3rd data point when the last 108 data points (the 2nd through 10th folds or all but the 1st fold) are used as the training dataset.

   The first few rows of a sample `<test out>` output file are given below for `<val type>` 1, where $k$ ranges from 1 to 10 and the Manhattan distance metric is used:

   ```
   1
   1
   0
   1
   ...
   ```

Your labels should exactly match those of a reference $k$NN implementation — this will be checked by the autograder by running your program and evaluating your output file against the reference solution.

### 3.5 Output: Metrics File

Generate another file where you report the error rates for the various $k$NN classifiers your program implements. This file should be written to the path specified by the command line argument `<metrics out>`. Again, the contents of this file will depend on `<val type>`.

1. If `<val type>` is 0 (“normal” validation), then you should output the following error rates in this order:

   - the training error rates for each value of $k$ (each on their own line),
   - the validation error rates for each value of $k$ (again, each on their own line),
   - the test error rate corresponding to just the best value of $k$ (the one that minimizes the validation error rate) *where only the training dataset is used to make predictions*, and
   - the test error rate corresponding to just the best value of $k$ (the one that minimizes the validation error rate) *where the training and validation datasets are first combined and then used to make predictions*.

   Use ‘`\n`’ to create a new line.

   A few lines of a sample `<metrics out>` output file are given below for `<val type>` 0, where $k$ ranges from 1 to 5 and the Euclidean distance metric is used:

   ```
   k=1 training error rate: 0.0
   k=2 training error rate: 0.044444444444444446
   k=3 training error rate: 0.044444444444444446
   ...
   k=1 validation error rate: 0.06666666666666667
   k=2 validation error rate: 0.06666666666666667
   k=3 validation error rate: 0.03333333333333333
   ...
   test error rate (train): 0.0
   test error rate (train + validation): 0.0
   ```

2. If `<val type>` is 1 (10-fold cross-validation), then you should output the 10-fold cross-validation error rates for each value of $k$ (each on their own line) followed by the error rate for just the best value of $k$ (the one that minimizes the cross-validation error rate). Use ‘`\n`’ to create a new line.

   A few lines of a sample `<metrics out>` output file are given below for `<val type>` 1, where $k$ ranges from 1 to 10 and the Manhattan distance metric is used:

   ```
   k=1 cross-validation error rate: 0.06666666666666667
   k=2 cross-validation error rate: 0.09166666666666666
   k=3 cross-validation error rate: 0.05
   ...
   test error rate: 0.0
   ```

Again, your reported error rates should be within 0.0001 of the reference solution. You do not need to round your reported numbers! The autograder will automatically incorporate the right tolerance for float comparisons.

At this point, you should be able to go back and answer questions 1-3 in the “Empirical Questions” section of this handout. Write your solutions in the template provided.

### 3.6 Submission Instructions

**Programming** Please ensure you have completed the following file for submission.

```
knn.py
```

When submitting your solution, make sure to select and upload the correct file. **Any other files will be deleted.** Ensure the file has the exact same spelling and letter casing as above. You can either directly zip the file (by selecting the file and compressing it – do not compress the folder containing the file) or directly drag it to Gradescope for submission.

**Written Questions** Make sure you have completed all questions from Written component (including the collaboration policy questions) in the template provided. When you have done so, please submit your document in **PDF format** to the corresponding assignment slot on Gradescope.