Homework 2
Estimators, Linear and Logistic Regression, and Naïve Bayes[1]

CMU 10-701: Introduction to Machine Learning (Fall 2023)

https://machinelearningcmu.github.io/F23-10701/
OUT: Wednesday, Sep 20th, 2023

DUE: Wednesday, Oct 4th, 2023, 11:59pm

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 in our course syllabus for more information: https://machinelearningcmu.github.io/F23-10701/#Syllabus.
- **Late Submission Policy:** See the late submission policy here: https://machinelearningcmu.github.io/F23-10701/#Syllabus.
- **Submitting your work:**
  - **Gradescope:** There will be two submission slots for this homework on Gradescope: Written and Programming.
    For the written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using the written submission slot. Please use the provided template. The best way to format your homework is by using the Latex template released in the handout and writing your solutions in Latex. However submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Each derivation/proof should be completed in the boxes provided below the question, **you should not move or change the sizes of these boxes** as Gradescope is expecting your solved homework PDF to match the template on Gradescope. If you find you need more space than the box provides you should consider cutting your solution down to its relevant parts, if you see no way to do this, please add an additional

[1]Compiled on Thursday 21st September, 2023 at 18:26

page at the end of the homework and guide us there with a 'See page xx for the rest of the solution'.

You are also required to upload your code, which you wrote to solve the final question of this homework, to the Programming submission slot.

Regrade requests can be made after the homework grades are released, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted.

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 LaTeXusers, use ■ and ●for shaded boxes and circles, and don't change anything else. If an answer box is included for showing work, **you must show your work**!

1 MLE and MAP in Linear and Ridge Regression [28 Points]

In this problem, we will motivate the use of OLS in linear regression. Consider the following linear regression model:

For each data point in $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$,

$$y^{(i)} = \boldsymbol{w}^T \boldsymbol{x}^{(i)} + \epsilon \text{ where } y^{(i)}, \epsilon \in \mathbb{R} \text{ and } \boldsymbol{w}, \boldsymbol{x}^{(i)} \in \mathbb{R}^{d+1}$$

In matrix notation, we can express this linear relationship for all data points as:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{y}, \boldsymbol{\epsilon} \in \mathbb{R}^n, \boldsymbol{X} \in \mathbb{R}^{n \times (d+1)}, \text{ and } \boldsymbol{w} \in \mathbb{R}^{d+1}$$

1. **[8 Points]** Under the assumption that the residuals are normal i.i.d. ($\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$), we can write:

   $$\boldsymbol{y} | \boldsymbol{X}, \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, \sigma^2 \boldsymbol{I})$$

   Show that the MLE for $\boldsymbol{w}$ is the same as the solution provided by OLS seen in class: $\boldsymbol{w}^*_{\text{MLE}} = \boldsymbol{w}^*_{\text{OLS}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. In other words, the MLE induces a mean squared error loss.

2. [**10 Points**] We will now provide some motivation for the use of ridge regression. In the same setting as above, now assume that we have a Gaussian prior on $\boldsymbol{w}$:

$$\boldsymbol{w} \sim \mathcal{N}\left(0, \frac{2\sigma^2}{\lambda}\boldsymbol{I}\right)$$

for some fixed $\lambda > 0$. Recall that in ridge regression, the optimal parameter vector is given by:

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2.$$

Show that the solution to the MAP estimate $\boldsymbol{w}^*_{\text{MAP}}$ is the same as the one obtained from ridge regression. In other words, the MAP induces L2 regularization.

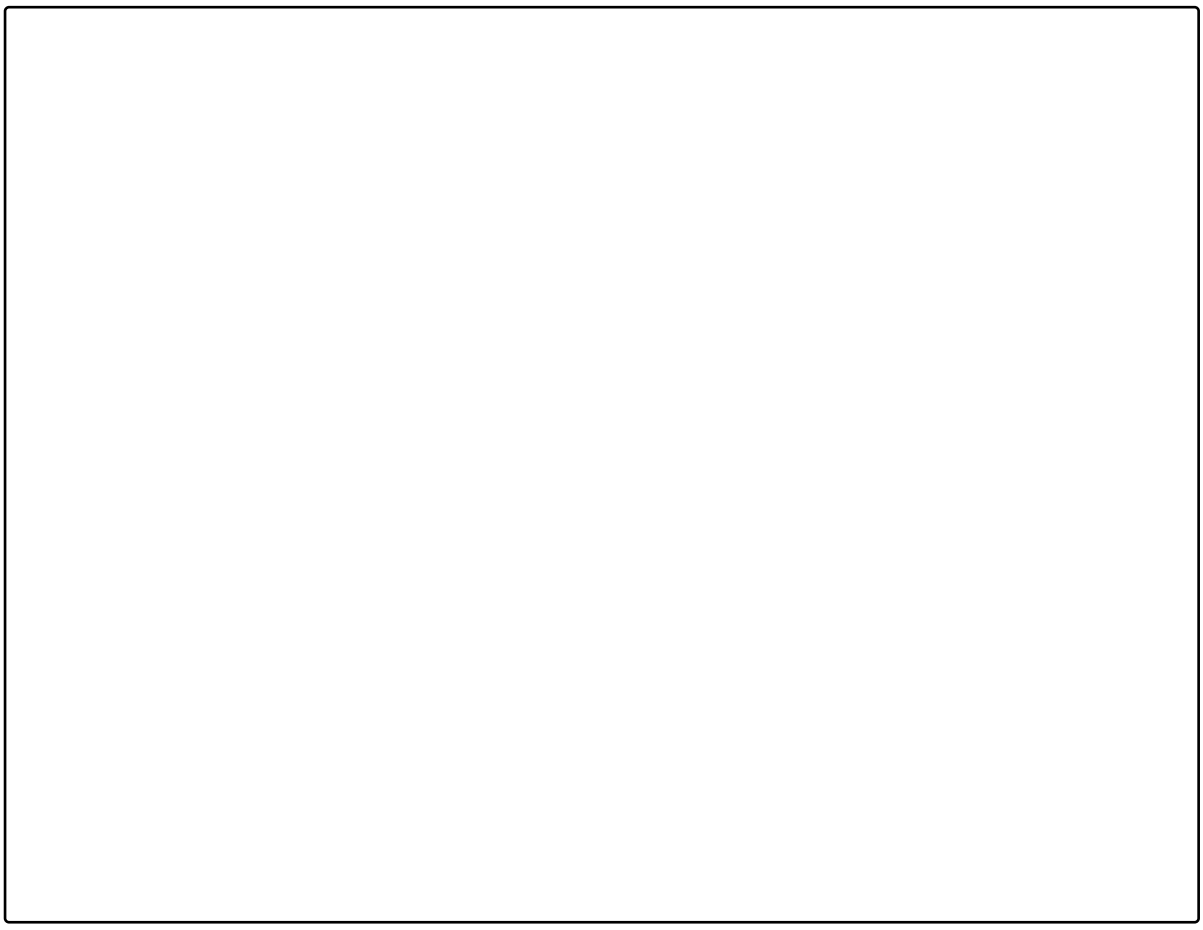
(Hint: Start by writing down the expression for the negative log posterior and show that minimizing it gives the same solution as minimizing the OLS with Ridge regression.)

3. **MLE estimator might be biased [10 points]**

In this question you will prove that the MLE estimate for the variance parameter of a Gaussian is a biased estimator. Given n independent observations drawn from a univariate Gaussian distribution $x^{(1)}, ..., x^{(n)} \sim \mathrm{N}(\mu, \sigma^2)$, recall that the MLE of the mean and variance parameters are

$$\widehat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}$$
$$\widehat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \widehat{\mu})^2.$$

Prove that $\widehat{\mu}$ is an unbiased estimator of μ (by showing $\mathbb{E}[\widehat{\mu}] = \mu$) and that $\widehat{\sigma}^2$ is a biased estimator of σ^2 (by showing $\mathbb{E}[\widehat{\sigma}^2] \neq \sigma^2$). Also, propose an unbiased estimator of σ^2 based on your new insights.



2 Naive Bayes [22 Points]

Let $X = (x_1, x_2, .., .x_d)$ denote a set of features and $y \in \{0, 1\}$ denote a binary label. Note that in any generative model approach, we model the conditional label distribution $P(y \mid X)$ via the conditional distribution of features given the label $P(X \mid y)$:

$$P(y \mid X) \propto P(X \mid y)P(y)$$

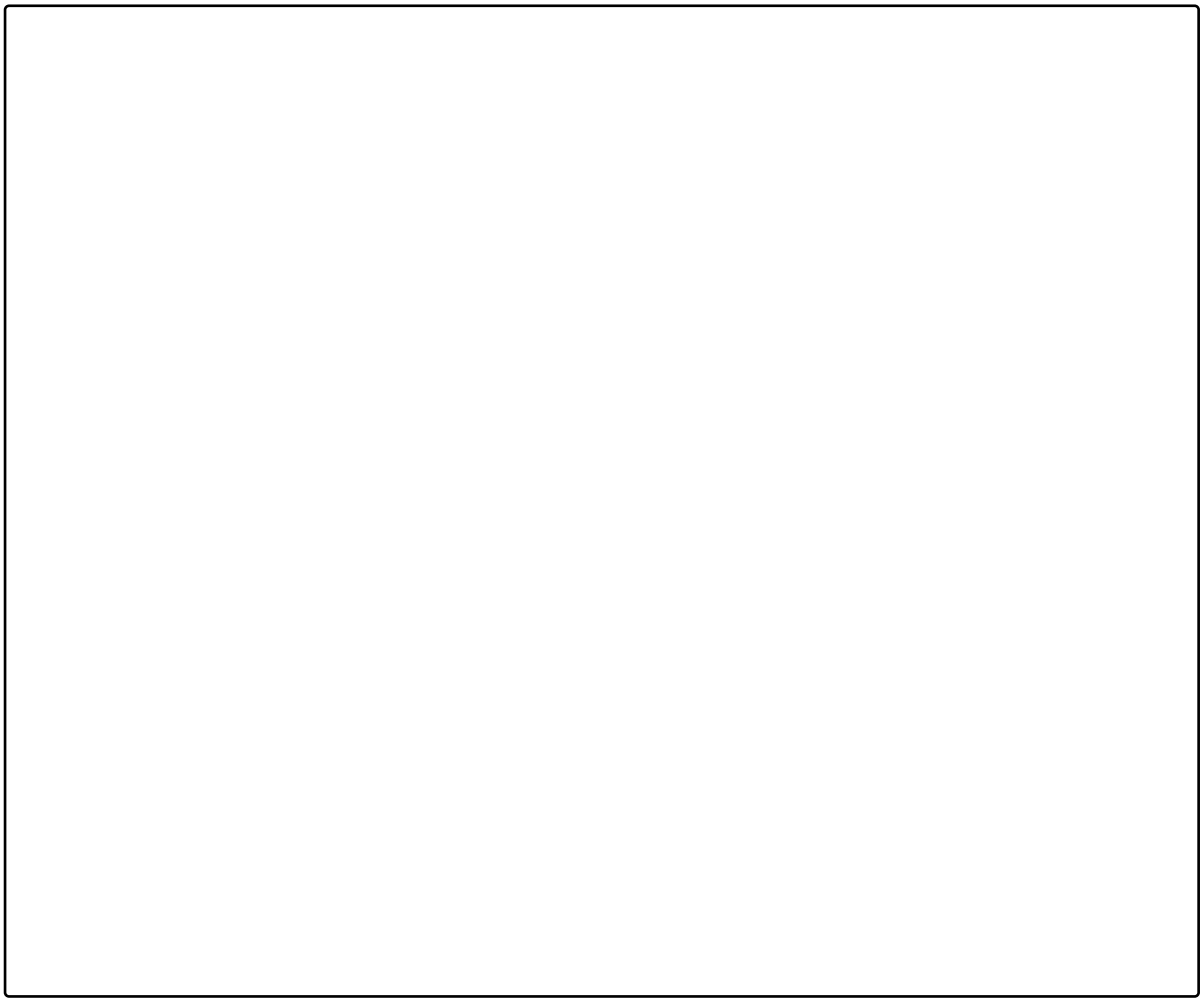
1. **Multinomial Naive Bayes** Suppose that each feature x_i takes values in the set $\{1, 2, ..., K\}$. Further, suppose that the label distribution is Bernoulli, and the feature distribution conditioned on the label is multinomial. Please give detailed step by step derivations for the following questions.

   (a) **[3 points]** What is the total number of parameters of the model under the Naive Bayes assumption?

   (b) **[3 points]** What is the total number of parameters of the model without the Naive Bayes assumption?

(c) [**6 points**] Suppose we change the set of values that y takes, so that $y \in \{0, 1, ..., M-1\}$. How would your answers change in both cases (with and without the Naive Bayes assumption)?

2. **Gaussian Naive Bayes [10 points]** Suppose each feature is real-valued, with $x_i \in \mathbb{R}$, and $P(x_i \mid y = c) \sim \mathcal{N}(\mu_{i,c}, 1)$ for $i = 1, 2, ..., d$ and $c = 0, 1$. Again, suppose that the label distribution is Bernoulli with $P(y = 1) = p$. Under the Naive Bayes assumption, show that the decision boundary $\{(x_1, x_2, \ldots, x_d) : P(y = 0 \mid x_1, x_2, \ldots, x_d) = P(y = 1 \mid x_1, x_2, \ldots, x_d)\}$ is linear in $x_1, x_2, \ldots, x_d$.



3 Implementing Naive Bayes [35 points]

In this question you will implement a Naive Bayes classifier for a text classification problem. You will be given a collection of text articles, each coming from either the serious European magazine The Economist, or the not-so-serious American magazine The Onion. The goal is to learn a classifier that can distinguish between articles from each magazine.

We have pre-processed the articles so that they are easier to use in your experiments. We extracted the set of all words that occur in any of the articles. This set is called the vocabulary and we let V be the number of words in the vocabulary. For each article, we produced a feature vector $X = \langle X_0, \ldots, X_{V-1} \rangle$, where X_i is equal to 1 if the i^{th} word appears in the article and 0 otherwise. Each article is also accompanied by a class label of either 0 for The Economist or 1 for The Onion.

When we apply the Naive Bayes classification algorithm, we make two assumptions about the data: first, we assume that our data is drawn iid from a joint probability distribution over the possible feature vectors X and the corresponding class labels Y; second, we assume for each pair of features X_i and X_j with $i \neq j$ that X_i is conditionally independent of X_j given the class label Y (this is the Naive Bayes assumption). Under these assumptions, a natural classification rule is as follows: Given a new input X, predict the most probable class label $\hat{Y}$ given X. Formally,

$$\hat{Y} = \underset{y}{\text{argmax}} P(Y = y \mid X)$$

Using Bayes Rule and the Naive Bayes assumption, we can rewrite this classification rule as follows:

$$\begin{aligned}
\hat{Y} &= \underset{y}{\text{argmax}} \frac{P(X \mid Y = y)P(Y = y)}{P(X)} && \text{(Bayes Rule)} \\
&= \underset{y}{\text{argmax}} P(X \mid Y = y)P(Y = y) && \text{(Denominator does not depend on } y\text{)} \\
&= \underset{y}{\text{argmax}} P\left(X_1, \ldots, X_V \mid Y = y\right) P(Y = y) \\
&= \underset{y}{\text{argmax}} \left(\prod_{w=1}^{V} P\left(X_w \mid Y = y\right) \right) P(Y = y) && \text{(Conditional independence).}
\end{aligned}$$

The advantage of the Naive Bayes assumption is that it allows us to represent the distribution $P(X \mid Y = y)$ using many fewer parameters than would otherwise be possible. Specifically, since all the random variables are binary, we only need one parameter to represent the distribution of X_w given Y for each $w \in \{1, \ldots, V\}$ and $y \in \{0, 1\}$. This gives a total of $2V$

parameters. On the other hand, without the Naive Bayes assumption, it is not possible to factor the probability as above, and therefore we need one parameter for all but one of the 2^V possible feature vectors X and each class label $y \in \{0, 1\}$. This gives a total of $2\left(2^V - 1\right)$ parameters. The vocabulary for our data has $V \approx 26,000$ words. Under the Naive Bayes assumption, we require on the order of 52,000 parameters, while without it we need more than $10^{7000!}$

Of course, since we don't know the true joint distribution over feature vectors X and class labels Y, we need to estimate the probabilities $P(X \mid Y = y)$ and $P(Y = y)$ from the training data. For each word index $w \in \{1, \ldots, V\}$ and class label $y \in \{0, 1\}$, the distribution of X_w given $Y = y$ is a Bernoulli distribution with parameter θ_{yw}. In other words, there is some unknown number θ_{yw} such that

$$P\left(X_w = 1 \mid Y = y\right) = \theta_{yw} \quad \text{and} \quad P\left(X_w = 0 \mid Y = y\right) = 1 - \theta_{yw}$$

We believe that there is a non-zero (but maybe very small) probability that any word in the vocabulary can appear in an article from either The Onion or The Economist. To make sure that our estimated probabilities are always non-zero, we will impose a Beta(2,2) prior on θ_{yw} and compute the MAP estimate from the training data.

Similarly, the distribution of Y (when we consider it alone) is a Bernoulli distribution with parameter ρ. In other words, there is some unknown number ρ such that

$$P(Y = 1) = p \quad \text{and} \quad P(Y = 0) = 1 - p.$$

In this case, since we have many examples of articles from both The Economist and The Onion, there is no risk of having zero-probability estimates, so we will instead use the MLE.

3.1 Program: Naive Bayes

Please confine all code you write to a single, self-contained file titled `naive_bayes.py`. Submit this file to Gradescope under Homework 2 Programming.

The file `hw2data.pkl` contains the following elements:

• `Vocabulary`: A text file containing the words occurring in the news articles. Each line in this file is a word (note that some of the words may look a little strange because we have run them through a stemming algorithm that tries to make words with common roots look the same. For example, "stemming" and "stemmed" would both become "stem".) The line number (zero indexed; i.e the first word maps to 0) indicates the id of that word. The file contains 26048 words.
• `XTrain`: is a n × V dimensional matrix describing the n documents used for training your Naive Bayes classifier. The entry `XTrain(i,j)` is 1 if word j appears in the i^{th} training document and 0 otherwise.

• `yTrain` is a n×1 dimensional matrix containing the class labels for the training documents. `yTrain(i,1)` is 0 if the i^{th} document belongs to The Economist and 1 if it belongs to The Onion.
• `XTest` and `yTest` are the same as `XTrain` and `yTrain`, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.

Note that `XTrain` and `XTest` are stored in a sparse array format.

You are free to implement your Naive Bayes classifier any way you wish. However, we have provided the following recommended structure in the handout:

1. Complete the function `D = NB_XGivenY(XTrain, yTrain)`. The output `D` is a $2 \times V$ matrix, where for any word index $w \in \{1, \ldots, V\}$ and class index $y \in \{0, 1\}$, the entry `D[y,w-1]` is the MAP estimate of $\theta_{yw} = P(X_w = 1|Y = y)$ with a Beta(2,2) prior distribution. **Note:** to help with numerical issues, you should clip D to be in $[10^{-5}, 1 - 10^{-5}]$.
2. Complete the function `p = NB_YPrior(yTrain)`. The output `p` is the MLE for $p = P(Y = 0)$.
3. Complete the function `yHat = NB_Classify(D, p, X)`. The input `X` is an $m \times V$ matrix containing m feature vectors (stored as its rows). The output `yHat` is a $m \times 1$ vector of predicted class labels, where `yHat[i]` is the predicted label for the i^{th} row of `X`. **Note:** in this function, you will want to use logspace arithmetic to avoid numerical problems (see the section below).
4. Complete the function `error = ClassificationError(yHat, yTruth)`.

3.2 Logspace Arithmetic

When working with very large or very small numbers (such as probabilities), it is often useful to work in *logspace* to avoid numerical precision issues. In logspace, we keep track of the logs of numbers, instead of the numbers themselves. For example, if $p(x)$ and $p(y)$ are probability values, instead of storing $p(x)$ and $p(y)$ and computing $p(x) * p(y)$, we work in log space by storing $\log(p(x))$, $\log(p(y))$, and we can compute the log of the product, $\log(p(x) * p(y))$ as $\log(p(x) * p(y)) = \log(p(x)) + \log(p(y))$.

3.3 Empirical Questions

1. [**10 Points**] In this question we explore how the size of the training data set affects the test and train error. For each value of m in $\{100, 130, 160, \ldots, 580\}$, train your Naive Bayes classifier on the first m training examples (that is, use the data given by `XTrain[0:m]` and `yTrain[0:m]`).

   Plot the training and testing error for each such value of m. The x-axis of your plot should be m, the y-axis should be error, and there should be one curve for training error and one curve for testing error; both curves should be on the same graph (same axes). Then explain the general trend of both the training and testing error curves.

2. [**10 Points**] When estimating the class prior and attribute distributions, we currently add 2 observations to each outcome because multiplying by zero can be fatal. In this question, we explore how estimates change when *stronger* priors are added to the parameters.

   To make sure that our estimated probabilities are always non-zero, we will impose a Beta(α, β) prior on θ_{yw}. Plot the training and testing error for $\alpha \in [2, 5, 10, 25, 50, 100]$. Keep β fixed at 2. The x-axis of your plot should be α, the y-axis should be error, and there should be one curve for training error and one curve for testing error; both curves should be on the same graph (same axes). Then, explain the intuition behind the observed changes regarding varying levels of α.

3. Next, we will try to interpret the learned parameters. For this question, revert back to the standard Beta(2,2) prior. Train your classifier on the data contained in `XTrain` and `yTrain`. For each of the following criteria, fill in the table for each label $y \in \{0, 1\}$:

   (Note that some of the words may look a little strange because we of the stemming algorithm. #1 should be the word that give the highest value and #5 the fifth highest):

   (a) **[5 Points]** Top five words that the model says are most likely to occur in a document from class y. That is, the top five words according to this metric:

   $$P(X_w = 1|Y = y)$$

	Word #1	Word #2	Word #3	Word #4	Word #5
$Y = 0$					
$Y = 1$					

   (b) **[5 Points]** Top five words w according to this metric:

   $$\frac{P(X_w = 1|Y = y)}{P(X_w = 1|Y \neq y)}.$$

	Word #1	Word #2	Word #3	Word #4	Word #5
$Y = 0$					
$Y = 1$					

   (c) **[5 Points]** Which list of words is more informative about the class y? Briefly explain your reasoning.

4 Time Series Prediction [15 points]

Time series data usually consists of a set of observations recorded over time with a regular frequency. Time series forecasting is the task of predicting future events by analyzing the trends of the past. In this question, you will use linear regression to perform *long* time series prediction. The goal is to train a simple linear model for predicting the upcoming entry in the time series based on the near history.

To predict the value x_i, you should use the past N data points in the series $X_{(i-N,i)} = \{x_{i-1}, x_{i-2}, \cdots, x_{i-N}\}$ as the input vector to your linear model, where $X_{(i-N,i)}$ means the vector formed by a subsequence of the time series data from index $i - N$ to $i - 1$. Given weight vector $\mathbf{w}$ and bias b, We predict the upcoming value $\tilde{x}_i$ via

$$\tilde{x}_i = X_{(i-N,i)}\mathbf{w} + b$$

Instructions

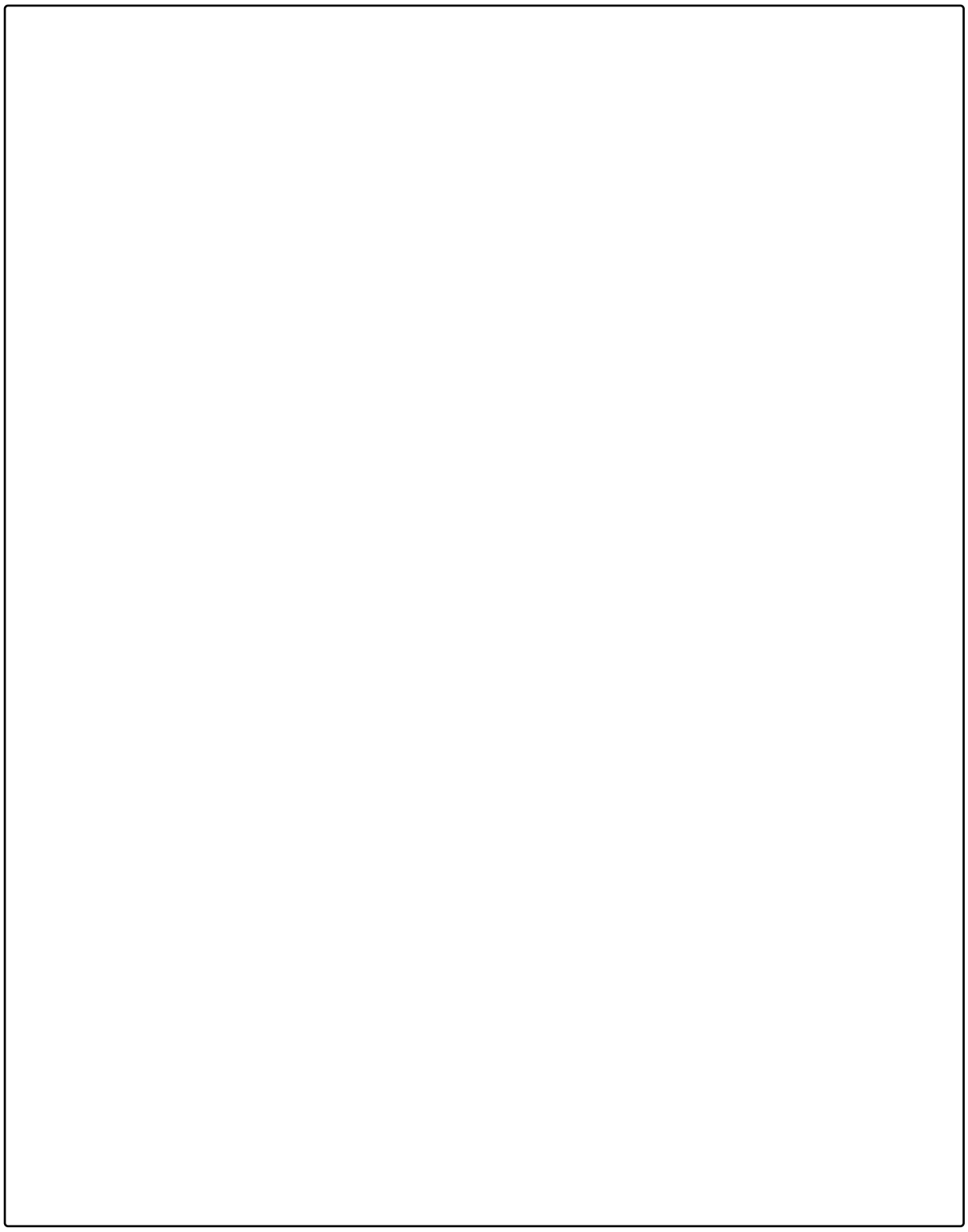
In this question, you'll perform time-series estimation on a climate sequence dataset. The dataset contains weather data from the Jena weather station in Germany from 2009 to 2016. We have pre-processed the dataset on your behalf to contain only the temperature data measured every 30 minutes and partitioned the dataset into training and test series.

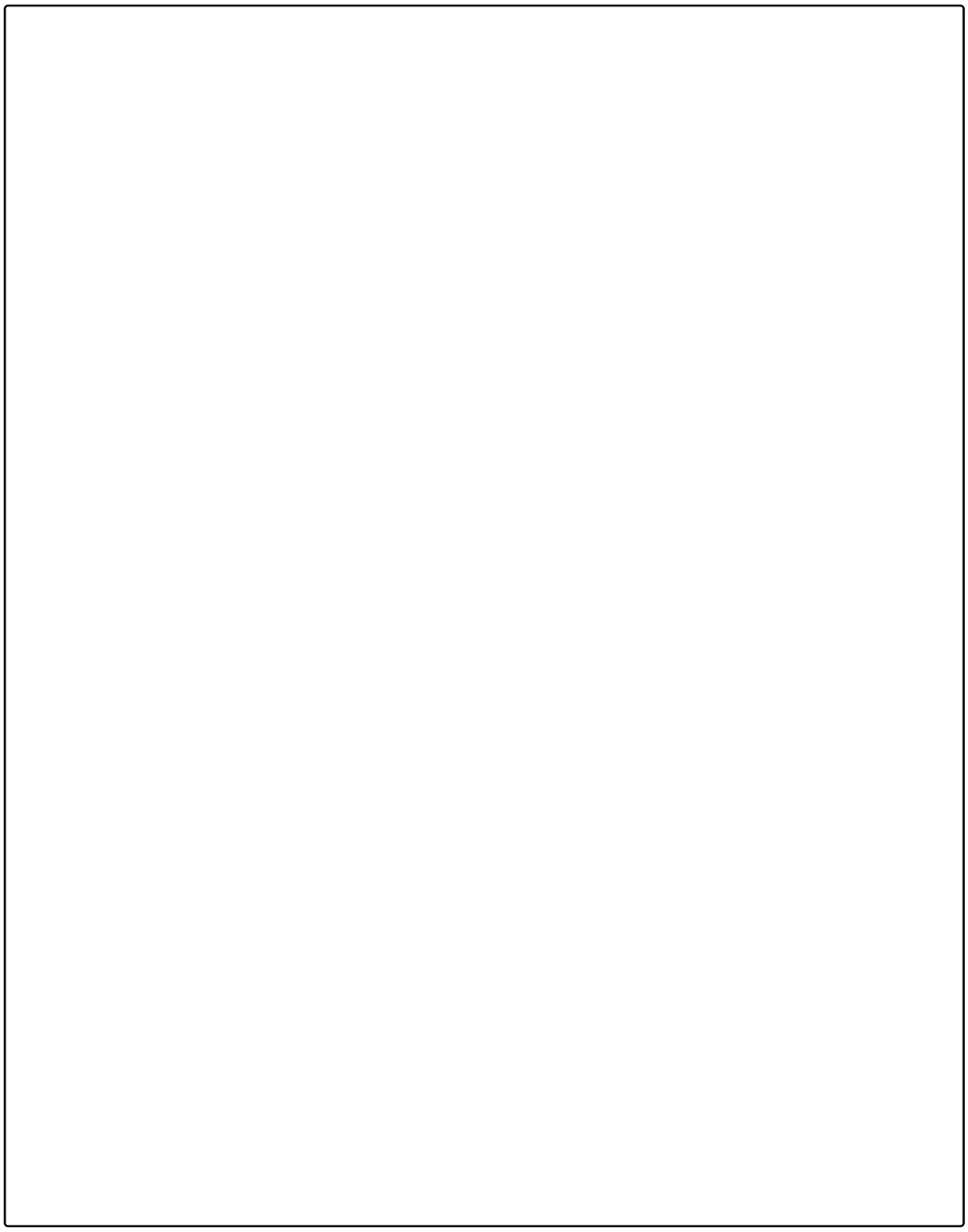
This is an open-ended, exploration question where you must research and implement *at least three different* methods for performing linear regression (but we encourage you to investigate more!). These can be approximate/stochastic for computing the ordinary least squares solutions, more advanced iterative methods or anything else you find; at most one of these can come from lecture (either ordinary least squares or gradient descent).

For each of these methods, you should train a linear model to predict the temperature at a certain time based on the past 2 years of data (so $2 * 365 * 24 * 2 = 35040$ observations) by minimizing the mean squared error subject to $L2$-regularization. You should describe the method in detail, along with any hyperparameters you needed to set and how you set them. Then, evaluate the performance of each model by

- plotting the predicted temperature series versus ground truth series using the test data,
- computing the mean squared error on the test data series, and
- reporting how long the method took you to train.

Feel free to explore and implement any linear regression techniques to solve the problem. However, keep in mind that you must implement these methods from scratch by yourselves and you are **only** allowed to use `numpy` library for your implementation. Other libraries such as `scikit-learn`, `pytorch`, `tensorflow` or `statsmodel` are **not** allowed. Please confine all code you write to a single, self-contained file titled `linear_regression.py`. Submit this file to Gradescope under Homework 2 Programming.





5 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment?

   (b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

2. (a) Did you give any help whatsoever to anyone in solving this assignment?

   (b) If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. (a) Did you find or come across code that implements any part of this assignment?

   (b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).