NAME OF CANDIDATE: .....

STUDENT ID: .....

SIGNATURE: .....

THE UNIVERSITY OF NEW SOUTH WALES

Term 2, 2023

COMP9417 Machine Learning and Data Mining – Final Examination

1. TIME ALLOWED — 8 HOURS
2. THIS EXAMINATION PAPER HAS 11 PAGES
3. TOTAL NUMBER OF QUESTIONS — 4
4. ANSWER **ALL QUESTIONS**
5. TOTAL MARKS AVAILABLE — 100
6. OPEN BOOK EXAM - LECTURE NOTES, TUTORIALS, AND ONLINE RESOURCES ARE PERMITTED. PLEASE REFERENCE ANY RESULTS THAT YOU USE, E.G., BY STATING THAT A PARTICULAR RESULT FOLLOWS FROM TUTORIAL A, QUESTION B, PART C. REFER TO EXAM INSTRUCTIONS ON MOODLE COURSE PAGE FOR SUBMISSION AND OTHER GUIDANCE.
7. DISCUSSION WITH OTHER STUDENTS IS STRICTLY PROHIBITED. EXAM SUBMISSIONS WILL BE CHECKED FOR PLAGIARISM. CHEATING WILL RESULT IN A FAILING GRADE FOR THE COURSE AND POTENTIAL FURTHER DISCIPLINARY ACTION.

Question 1

Please submit Question1.pdf on Moodle using the Final Exam - Question 1 object. You must submit a single PDF. You may submit multiple .py files (placed in a single zip file) if you wish. **Do not put your PDF in the zip file.** The parts are worth (1+4+3+2+2) + (1+1+1+5+5) = 12+13 = 25.

(a) **(Bias of Estimators)** Let $\gamma > 0$ and suppose that $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} N(\gamma, \gamma^2)$. We define:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i,$$

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

You may use the following two facts without proof:

(F1) $\overline{X}$ and S^2 are independent.

(F2) $\overline{X}$ and $c_* S$ are both unbiased estimators of γ.[1]

What to submit: for all parts (i)-(v), include your working out, either typed or handwritten. For all parts, you must show all working for full credit. Answers without working will receive a grade of zero.

(i) Consider the estimator:

$$T_1 = a\overline{X} + (1-a)c_* S.$$

Show that for any choice of constant a, T_1 is unbiased for γ.

(ii) What choice of a gives you the best (in the sense of MSE) possible estimator? Derive an explicit expression for this optimal choice of a. We refer to this estimator as T_1^*.

(iii) Consider now a different estimator:

$$T_2 = a_1\bar{X} + a_2(c_* S),$$

and we do not make any assumptions about the relationship between a_1 and a_2. Find the constants a_1, a_2 explicitly that make T_2 best (from the MSE perspective), i.e. choose a_1, a_2 to minimize $\text{MSE}(T_2) = \mathbb{E}(T_2 - \gamma)^2$. We refer to this estimator as T_2^*.

(iv) Show that T_2^* has MSE that is less than or equal to the MSE of T_1^*.

(v) Consider the estimator $V_+ = \max\{0, T_2^*\}$. Show that the MSE of V_+ is smaller than or equal to the MSE of T_2^*.

(b) **(kNN Regression) Note: Using an existing/online implementation of the algorithms described in this question will result in a grade of zero. You may use code from the course with reference.** Consider the usual data generating process $y = f(x) + \epsilon$, where f is some unknown function, and ϵ is a noise variable with mean zero and variance σ^2. Recall that in kNN regression, we look at the k nearest neighbours (in our dataset) of an input point x_0, we then consider their corresponding response values and average them to get a prediction for x_0. Given a dataset $D = \{(x_i, y_i)\}_{i=1}^n$ we can write down the kNN prediction as

$$\hat{m}(x_0) = \frac{1}{k}\sum_{i \in \mathcal{N}_k(x_0)} y_i,$$

where $\mathcal{N}_k(x_0)$ is the set of indices of the k nearest neighbours of x_0. Without loss of generality, label the k nearest neighbours of x_0 as $z_1, \ldots, z_k$ and their corresponding response values by $t_1, \ldots, t_k$.

[1] You do not need to worry about knowing or calculating c_* for this question, it is just some constant.

(i) Show that

$$[\text{bias}(\hat{m}(x_0))]^2 = \left(f(x_0) - \frac{1}{k} \sum_{i=1}^{k} f(z_i) \right)^2 .$$

Throughout, you should treat x_0 as a fixed point (not a random variable).
What to submit: your working out, either typed or handwritten.

(ii) Derive an expression for the variance $\text{var}(\hat{m}(x_0))$.
What to submit: your working out, either typed or handwritten.

(iii) Using the results so far, write down an expression for the MSE of $\hat{m}(x_0)$.
What to submit: your working out, either typed or handwritten, and some commentary.

(iv) Take $f(x) = 0.3 \sin(\frac{x}{3})$ and $\epsilon \sim N(0, \sigma^2)$ where $\sigma = 0.3$. Generate $n = 20$ samples from this model and fit a kNN-regression model with k ranging from 1 to 20. Provide a 5×4 grid of plots showing the performance of the kNN model for that choice of k. In each of your sub-plots, plot the samples as well as the true function. You may **not** use any existing implementations of kNN regression from online sources. *What to submit: a single plot and a screen shot of your code as well as a copy of your code in your .py file.*

(v) Describe what happens to the bias of the kNN estimator at x_0 when k is very small (1NN), and what happens when k is very large ($k \to n$). Similarly, what happens to the variance? What does this tell you about the relationship between bias and variance and choice of k? For full credit, your answer should use the results of the previous two questions as evidence for your discussion. *What to submit: Some commentary.*

Question 2

Please submit Question2.pdf on Moodle using the Final Exam - Question 2 object. You must submit a single PDF. You may submit multiple .py files (placed in a single zip file) if you wish. **Do not put your PDF in the zip file.** The parts are worth (2+1+4+ 2+4+ 6 + 6) = 25.

Note: Using an existing/online implementation of the algorithms described in this question will result in a grade of zero. You may use code from the course with reference.

Recall the K-means algorithm: Given data $X_1, \ldots, X_n$, the number of clusters K, and number of iterations T:

1. Initialization: start with initial set of K-means (cluster centers): $\mu_1^{(0)}, \mu_2^{(0)}, \ldots, \mu_K^{(0)}$.
2. For $t = 1, 2, 3, \ldots, T$:
   - For $i = 1, 2, \ldots, n$: Find nearest mean to X_i by solving

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \|X_i - \mu_k^{(t-1)}\|_2^2.$$

   - For $k = 1, \ldots, K$: set[2] [3]

$$C_k^{(t)} = \{X_i \text{ such that } k_i = k\}$$
$$\mu_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{X_i \in C_k^{(t)}} X_i.$$

(a) Consider the following data-set of $n = 5$ points in $\mathbb{R}^2$:

$$\left\{ \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \begin{pmatrix} 4 \\ 4 \end{pmatrix}, \begin{pmatrix} 6 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \end{pmatrix} \right\}.$$

You run K-means on this dataset with $K = 2$ and initial cluster centers $\mu_1^{(0)} = (5, 2)^T, \mu_2^{(0)} = (4, 5)^T$. Compute the cluster centers at the next two iterations: $\mu_1^{(1)}, \mu_2^{(1)}$ and $\mu_1^{(2)}, \mu_2^{(2)}$ by hand. Be sure to show your working.

What to submit: your cluster centers and any working, either typed or handwritten.

(b) Your friend tells you that they are working on a clustering problem at work. You ask for more details and they tell you they have an unlabelled dataset with $p = 10000$ features and they ran K-means clustering using Euclidean distance. They identified 52 clusters and managed to define labellings for these clusters based on their *expert* domain knowledge. What do you think about the usage of K-means here? Do you have any criticisms?

What to submit: some commentary.

(c) Consider the data and random clustering generated using the following code snippet[4]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets

X, y = datasets.make_circles(n_samples=200, factor=0.4, noise=0.04, random_state=13)
colors = np.array(['orange', 'blue'])
```

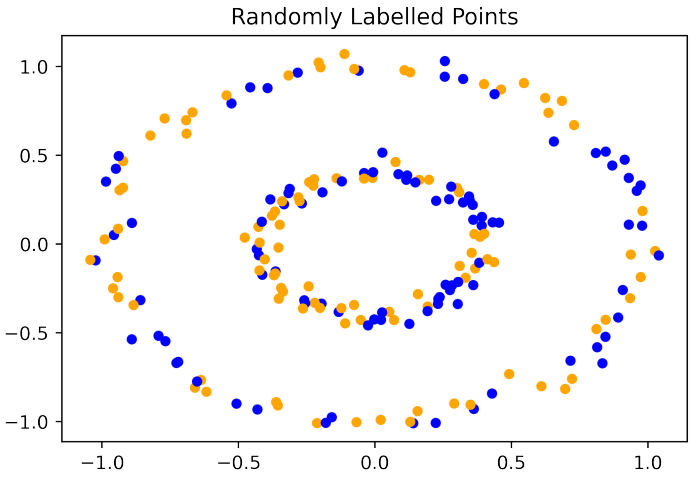
[2] Recall that for a set S, $|S|$ denotes its cardinality. For example, if $S = \{4, 9, 1\}$ then $|S| = 3$.

[3] The notation in the summation here means we are summing over all points belonging to the k-th cluster at iteration t, i.e. $C_k^{(t)}$.

[4] a copy of this code is provided in `code_student.py`

```
7
8  np.random.seed(123)
9  random_labeling = np.random.choice([0,1], size=X.shape[0], )
10 plt.scatter(X[:, 0], X[:, 1], s=20, color=colors[random_labeling])
11 plt.title("Randomly Labelled Points")
12 plt.savefig("Randomly_Labeled.png")
13 plt.show()
14
```

The random clustering plot is displayed here:



Implement K-means clustering from scratch on this dataset. Initialize the following two cluster centers:

$$\mu_1^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mu_2^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

and run for 10 iterations. In your answer, provide a plot of your final clustering (after 10 iterations) similar to the randomly labeled plot, except with your computed labels in place of `random_labelling`. Do you think K-means does a good job on this data? Provide some discussion on what you observe. *What to submit: some commentary, a single plot, a screen shot of your code and a copy of your code in your .py file.*

(d) You decide to extend your implementation by considering a feature transformation which maps 2-dimensional points (x_1, x_2) into 3-dimensional points of the form $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. Run your K-means algorithm (for 10 iterations) on the transformed data with cluster centers:

$$\mu_1^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \mu_2^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}.$$

Note for reference that the nearest mean step of the algorithm is now:

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\| \phi(X_i) - \frac{1}{|C_k^{(t-1)}|} \sum_{X_j \in C_k^{(t-1)}} \phi(X_j) \right\|_2^2, \tag{1}$$

where $\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$. In your answer, provide a plot of your final clustering using the code provided in (c) as a template. Provide some discussion on what you observe. *What to submit: a single plot, a screen shot of your code and a copy of your code in your .py file, some commentary.*

(e) You recall (from lectures perhaps) that directly applying a feature transformation to the data can be computationally intractable, and can be avoided if we instead write the algorithm in terms of a function h that satisfies: $h(x, x') = \langle \phi(x), \phi(x') \rangle$. Show that the nearest mean step in (1) can be re-written as:

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \big(h(X_i, X_i) + T_1 + T_2 \big),$$

where T_1 and T_2 are two separate terms that may depend on $C_k^{(t-1)}$, $h(X_i, X_j)$ and $h(X_j, X_\ell)$ for $X_j, X_\ell \in C_k^{(t-1)}$. The expressions should **not** depend on ϕ. *What to submit: your full working.*

(f) With your answer to the previous part, you design a new algorithm: Given data $X_1, \ldots, X_n$, the number of clusters K, and the number of iterations T:

1. Initialization: start with initial set of K clusters: $C_1^{(0)}, C_2^{(0)}, \ldots, C_K^{(0)}$.
2. For $t = 1, 2, 3, \ldots, T$:
   - For $i = 1, 2, \ldots, n$: Solve

$$k_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \big(h(X_i, X_i) + T_1 + T_2 \big).$$

   - For $k = 1, \ldots, K$, set

$$C_k^{(t)} = \{X_i \text{ such that } k_i = k\}.$$

The goal of this question is to implement this new algorithm from scratch using the same data generated in part (c). In your implementation, you will run the algorithm two times: first with the function:

$$h_1(x, x') = (1 + \langle x, x' \rangle),$$

and then with the function

$$h_2(x, x') = (1 + \langle x, x' \rangle)^2.$$

For your initialization (both times), use the provided `initial_clusters`, which can be loaded in by running `inital_clusters = np.load('init_clusters.npy')`. Run your code for at most 10 iterations, and provide two plots, one for h_1 and another for h_2. Discuss your results for the two functions. *What to submit: two plots, your discussion, a screen shot of your code and a copy of your code in your .py file.*

(g) The initializations of the algorithms above were chosen very specifically, both in part (d) and (f). Investigate different choices of intializations for your implemented algorithms. Do your results look similar, better or worse? Comment on the pros/cons of your algorithm relative to K-means, and more generally as a clustering algorithm. For full credit, you need to provide justification in the form of a rigorous mathematical argument and/or empirical demonstration. *What to submit: your commentary.*

Question 3

Please submit Question3.pdf on Moodle using the Final Exam - Question 3 object. You must submit a single PDF. You may submit multiple .py files (placed in a single zip file) if you wish. **Do not put your PDF in the zip file.** The parts are worth (2+2+8+7) +(2+2+2) = 19 + 6 = 25.

Note: Using an existing/online implementation of the algorithms described in this question will result in a grade of zero. You may use code from the course with reference.

(a) (**Regularized Sequence Model**) Throughout this problem, let $y, \beta \in \mathbb{R}^p$ and $\lambda > 0$. Further, we define the matrix $W \in \mathbb{R}^{(p-2)\times p}$ as

$$W = \begin{bmatrix} 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \end{bmatrix},$$

where blanks denote zero elements.[5] The following code allows you to load in the data[6]:

```
import numpy as np
import matplotlib.pyplot as plt
t_var = np.load("t_var.npy")
y_var = np.load("y_var.npy")
plt.plot(t_var, y_var)
plt.show()

```

Note, the t variable is purely for plotting purposes, it should not appear in any of your calculations.

(i) Consider the loss function:

$$L(\beta) = \frac{1}{2}\|y - \beta\|_2^2 + \lambda\|W\beta\|_2^2.$$

Find an explicit closed-form expression for the solution $\hat{\beta} = \arg\min_\beta L(\beta)$. Update the following code[7] so that it returns a plot of $\hat{\beta}$ and calculates $L(\hat{\beta})$. Only in your code implementation, set $\lambda = 0.3$.

```
def create_W(p):
    ## generate W which is a p-2 x p matrix as defined in the question
    # your code here
    return W

def loss(beta, y, W, L):
    ## compute loss for a given vector beta for data y, matrix W, regularization
    parameter L (lambda)
    # your code here
    return loss_val


## your code here
plt.plot(t_var, y_var, zorder=1, color='red', label='truth')
plt.plot(t_var, beta_hat, zorder=3, color='blue',
```

[5] If it is not already clear: for the first row of W: $W_{11} = 1, W_{12} = -2, W_{13} = 1$ and $W_{1j} = 0$ for any $j \geq 4$. For the second row of W: $W_{21} = 0, W_{22} = 1, W_{23} = -2, W_{24} = 1$ and $W_{2j} = 0$ for any $j \geq 5$ and so on.

[6] a copy of this code is provided in `code_student.py`

[7] a copy of this code is provided in `code_student.py`

```
            linewidth=2, linestyle='--', label='fit')
plt.legend(loc='best')
plt.title(f"L(beta_hat) = {loss(beta_hat, y, W, L)}")
plt.show()

```

What to submit: a closed form expression along with your working, a single plot and a screen shot of your code along with a copy of your code in your .py file.

(ii) Write out each of the two expressions that make up the loss function ($\frac{1}{2}\|y - \beta\|_2^2$ and $\lambda\|W\beta\|_2^2$) explicitly using summations. Explain the role played by each of the two terms (explain what W is doing). Be as specific as possible. *What to submit: your answer, and any working either typed or handwritten.*

(iii) Recall from Homework 1 the coordinate-level iterative scheme.[8] Derive closed-form expressions for $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ where for $j = 1, 2, \ldots, p$:

$$\hat{\beta}_j = \arg\min_{\beta_j} L(\beta_1, \ldots, \beta_{j-1}, \beta_j, \beta_{j+1}, \ldots, \beta_p).$$

What to submit: a closed form expression along with your working.

(iv) Implement both gradient descent and the coordinate scheme in code (from scratch) and apply it to the provided data. In your implementation:

- Use $\lambda = 0.3$ for the coordinate scheme, and step-size $\alpha = 0.001$ for your gradient descent scheme.
- Initialize both algorithms with $\beta = 1_p$, the p-dimensional vector of ones.
- For the coordinate scheme, be sure to update the β_j's in order (i.e. 1,2,3,...)
- For your coordinate scheme, terminate the algorithm after 3000 updates (each time you update a single coordinate, that counts as an update.)
- For your GD scheme, terminate the algoirthm after 3000 updates as well.
- Create a single plot of k vs $\Delta^{(k)} = L(\beta^{(k)}) - L(\hat{\beta})$, where $\hat{\beta}$ is the closed form expression derived earlier. Your plot should have both the coordinate scheme (blue) and GD (green) displayed and should start from $k = 0$. Your plot should have a legend.

What to submit: a single plot and a screen shot of your code along with a copy of your code in your .py file.

(b) (**Iterative Optimization**) Gradient-based optimization algorithms are a driving force in modern machine learning, and we have already seen multiple examples in this course, namely: gradient descent, stochastic gradient descent and Backpropagation. In this question we will consider a general approach to generating a family of gradient-based iterative algorithms. Assume that you have some (differentiable) function $f : \mathbb{R}^p \to \mathbb{R}$ that you wish to minimize.[9] Let ψ be some differentiable

[8] Initialize $\beta^{(0)}$, and then solve for $k = 1, 2, 3, \ldots, K$

$$\begin{aligned}
\beta_1^{(k)} &= \arg\min_{\beta_1} L(\beta_1, \beta_2^{(k-1)}, \beta_3^{(k-1)}, \ldots, \beta_p^{(k-1)}) \\
\beta_2^{(k)} &= \arg\min_{\beta_2} L(\beta_1^{(k)}, \beta_2, \beta_3^{(k-1)}, \ldots, \beta_p^{(k-1)}) \\
&\vdots \\
\beta_p^{(k)} &= \arg\min_{\beta_p} L(\beta_1^{(k)}, \beta_2^{(k)}, \beta_3^{(k)}, \ldots, \beta_p).
\end{aligned}$$

[9] The notation here just means that f is a function that takes inputs (vectors) from $\mathbb{R}^p$ and returns elements in $\mathbb{R}$ (real numbers).

function and consider the pseudo-distance:

$$H_\psi(x,z) = \psi(x) - \psi(z) - \langle \nabla\psi(z), x - z\rangle\,, \qquad x, z \in \mathbb{R}^p.$$

We will minimize f iteratively (like in gradient descent) using the following update rule at each step:

$$x^{(t+1)} = \arg\min_x \{\alpha \left\langle \nabla f(x^{(t)}), x \right\rangle + H_\psi(x, x^{(t)})\}, \tag{2}$$

where $\alpha > 0$ is a given learning rate (step size). Summarize your results for the next three questions using the following table:

	ψ	$H_\psi(x,z)$	$x^{(t+1)}$
(i)	$\frac{1}{2}\|x\|_2^2$	-	$x^{(t)} - \alpha\nabla f(x^{(t)})$
(ii)	$\frac{1}{2}x^TQx$	-	-
(iii)	$\sum_{i=1}^p x_i \log x_i$	-	-

What to submit: for all three parts, submit your working out, either typed or handwritten, and your filled in table of results. You cannot use matlab/sympy etc to solve any calculations, and all working must be shown.

(i) Show that if we take $\psi : \mathbb{R}^p \to \mathbb{R}$ as $\psi(x) = \frac{1}{2}\|x\|_2^2$, then solving the update step (2) recovers the gradient descent algorithm.[10]

(ii) Let Q be a $p \times p$ invertible matrix, and consider $\psi : \mathbb{R}^p \to \mathbb{R}$ defined as $\psi(x) = x^TQx$. Redo the previous question with this choice of ψ.

(iii) Let $\mathcal{S}^{p-1}$ denote the space of p dimensional vectors such that the coordinates of the vector are non-negative and sum to 1[11].Consider $\psi : \mathcal{S}^{p-1} \to \mathbb{R}$ defined by $\psi(x) = \sum_{i=1}^p x_i \ln x_i$. What updates does this choice of ψ generate?[12]

[10]Hint: find an expression for H_ψ, then plug your result into (2), differentiate with respect to x, set equal to zero and solve for x.
[11]For example, if $p = 3$, then $x = (1/3, 2/3, 0)$ belongs to $\mathcal{S}^2$, but $z = (1/3, -1/3, 1)$ doesn't belong to $\mathcal{S}^2$.
[12]Hint: it may be easier to work on the coordinate level rather than with vectors for this question.

Question 4

Please submit Question4.pdf on Moodle using the Final Exam - Question 4 object. You must submit a single PDF. You may submit multiple .py files (placed in a single zip file) if you wish. **Do not put your PDF in the zip file.** The parts are worth (1+2+2+2+2) + (1+2+2) + 6 + 5 = 9 + 5 + 6 + 5 = 25.

(a) (**MLE Limitations**) Consider independent observations $X_i \sim N(\mu_i, \sigma^2)$, $Y_i \sim N(\mu_i, \sigma^2)$ for $i = 1, \ldots, n$.[13]. *What to submit: for parts (i),(ii),(iii) submit your full working either typed or handwritten. For part (iv), provide commentary, a singe plot and your code (screen shot + a copy in your .py file(s).) For part(v), provide some commentary.*

(i) Write out an expression for the likelihood:

$$L(\mu_1, \mu_2, \ldots, \mu_n, \sigma^2) = \prod_{i=1}^{n} p(X_i; \mu_i, \sigma^2) p(Y_i; \mu_i, \sigma^2).$$

Simplify your expression as much as possible for full credit.

(ii) Find the MLE estimates $\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_n$ of $\mu_1, \mu_2, \ldots, \mu_n$.

(iii) Find the $\hat{\sigma}^2$ that maximizes the function $g(\sigma^2) = L(\hat{\mu}_1, \ldots, \hat{\mu}_n, \sigma^2)$.[14]

(iv) Let $\hat{\sigma}_n^2$ denote the estimator computed in the previous part. We are now using the subscript to make the dependence on the sample size n explicit. For any reasonable estimator, we would expect that the quantity: $\delta_n = |\hat{\sigma}_n^2 - \sigma^2|$ becomes smaller as n becomes larger. To check whether or not this happens, set up an experiment in Python to calculate $\hat{\sigma}_n^2$ for increasing sample sizes[15]. For each sample size n, simulate a new data-set from the model[16] and calculate the deviation δ_n. You may assume throughout that $\mu_i = 1$ for all i, and that $\sigma^2 = 0.5$. Provide a plot of δ_n against n. What do you observe in your plot?

(v) Usually, MLE estimators perform well, especially when the sample size is large. Discuss briefly why the MLE might be failing in this problem. What is different about this data generating procedure compared to other MLE problems we have seen in the course?

(b) (**Tree Construction**) Consider the data set below, with features: W, X and Z, and target Y.

index	Y	W	X	Z
1	0	A	0	0
2	0	B	1	1
3	0	B	1	1
4	1	B	2	0
5	1	A	2	0
6	0	A	0	1
7	1	A	2	1
8	1	B	1	0
9	1	B	1	0
10	0	A	0	1

Throughout this question, use $\log_2$ in your calculations. You must calculate everything by hand and not use an online implementation of decision trees. *What to submit: for (i)-(iii), submit your answer, and any working either typed or handwritten.*

[13] Note that this means X_i and Y_i are independent of each other, and also that X_i and Y_j are independent of each other for any choice of i and j

[14] Note, $\hat{\sigma}^2$ computed this way actually corresponds to the MLE estimate of σ^2. You do not need to prove this fact.

[15] You can use the following code for the choice of n: `N = 100; ns = np.linspace(10,10000, N, dtype=int)`

[16] at each n, you will sample a brand new set of X_i and Y_i's. There is no need to set a seed here.

(i) What is the entropy of Y?

(ii) Suppose that feature X is chosen for the root of a decision tree. What is the information gain associated with this attribute?

(iii) Draw the full decision tree (without pruning) and with X as the feature chosen for the root node. Label your tree clearly and outline the prediction made at each leaf node. What can you say about feature W? Show all working.

(c) (**Learning Theory**) Consider a hypothesis class $\mathcal{F}$ with VC dimension equal to 4. Consider the following statements:

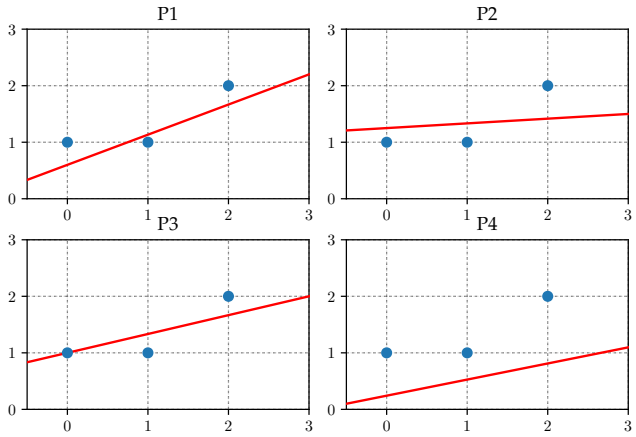
(I) Given a dataset of 4 samples, there is at least one classifier in $\mathcal{F}$ that can achieve an error of zero on these points.

(II) Classifiers in $\mathcal{F}$ will always achieve non-zero error on any dataset of size 5.

(III) Since the VC dimension of $\mathcal{F}$ is finite, $\mathcal{F}$ itself must also be finite.

Which of the statements are true? Which are false? Explain why or provide a counter-example. *What to submit: provide your answer with full working shown, either typed or handwritten.*

(d) (**Regularized Models**) Consider the four plots below labelled P1, P2, P3 and P4 which depict a fitted linear regression model built on 3 training data points:



Each of the four fits correspond to one of the following four regression schemes:

(I) $\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 + \lambda w_1^2$ where $\lambda = 1$

(II) $\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 + \lambda w_1^2$ where $\lambda = 10$

(III) $\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 + \lambda(w_0^2 + w_1^2)$ where $\lambda = 1$

(IV) $\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 + \lambda(w_0^2 + w_1^2)$ where $\lambda = 10$

Match each scheme to a single plot above, and be sure to explain in detail your choice. No explicit calculations are needed here. *What to submit: your commentary.*