NAME OF CANDIDATE: .....

STUDENT ID: .....

SIGNATURE: .....

THE UNIVERSITY OF NEW SOUTH WALES

Term 2, 2022

COMP9417 Machine Learning and Data Mining – Final Examination

1. TIME ALLOWED — 24 HOURS
2. THIS EXAMINATION PAPER HAS 12 PAGES
3. TOTAL NUMBER OF QUESTIONS — 4
4. ANSWER **ALL 4 QUESTIONS**
5. TOTAL MARKS AVAILABLE — 100
6. OPEN BOOK EXAM - LECTURE NOTES, TUTORIALS, AND ONLINE RESOURCES ARE PERMITTED. PLEASE 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 (2 + 4+ 4 + 3) + (3+3+6) = 25.

(a) (**Bias, Variance & MSE**) Let $X_1, \ldots, X_n$ be i.i.d. random variables with mean μ and variance σ^2. Define the estimator $T = \sum_{i=1}^n a_i X_i$ for some constants $a_1, \ldots, a_n$.

(i) What condition must the a_i's satisfy to ensure that T is an unbiased estimator of μ? Unbiased means that $\mathbb{E}T = \mu$.
What to submit: your working out, either typed or handwritten.

(ii) Under the condition identified in the previous part, which choice of the a_i's will minimize the MSE of T? Does this choice of a_i's surprise you? Provide some brief discussion. **Hint: this is a constrained minimization problem.**
What to submit: your working out, either typed or handwritten, and some commentary.

(iii) Suppose that instead, we let $a_i = \frac{1+b}{n}$ for $i = 1, \ldots, n$, where b is some constant. Find the value of b (in terms of μ and σ^2) which minimizes the MSE of T. How does the answer here compare to the estimator found in the previous part? How do the two compare as the sample size n increases? Are there any obvious issues with using the result of this question in practice?
What to submit: your working out, either typed or handwritten, and some commentary.

(iv) Suppose now that you are told that $\sigma^2 = \mu^2$. For the choice of b identified in the previous part, find the MSE of the estimator in this setting. How does the MSE compare to the MSE of the sample average $\overline{X}$? (Recall that $\text{MSE}(\overline{X}) = \text{var}(\overline{X}) = \sigma^2/n = \mu^2/n$). Further, explain whether or not you can use this choice of b in practice.
What to submit: your working out, either typed or handwritten, and some commentary.

(b) **(kNN Regression)** 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 $t_1, \ldots, t_k$.

(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). You may use any results from tutorials, labs or lectures without proof[1].
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.

[1] Please reference any results that you use, e.g. by stating that a particular result follows from Tutorial A, question B, part C.

(iii) Using the results so far, write down an expression for the MSE of $\hat{m}(x_0)$. 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 ($k \to \infty$). Similarly, what happens to the variance? What does this tell you about the relationship between bias and variance and choice of k?
What to submit: your working out, either typed or handwritten, and 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 (1+3+3+6) + (4+2+6) = 25.

(a) (**Kernel SVM**) Consider the following 2-dimensional data-set, where y denotes the class of each point.

index	x_1	x_2	y
1	1	0	-1
2	0	1	-1
3	0	-1	-1
4	-1	0	+1
5	0	2	+1
6	0	-2	+1
7	-2	0	+1

Throughout this question, you may use any desired packages to answer the questions.

(i) Use the transformation $x = (x_1, x_2) \mapsto (\phi_1(x), \phi_2(x))$ where $\phi_1(x) = 2x_2^2 - 4x_1 + 1$ and $\phi_2(x) = x_1^2 - 2x_2 - 3$. What is the equation of the best separating hyper-plane in the new feature space? Provide a plot with the data set and hyperplane clearly shown.
What to submit: a single plot, the equation of the separating hyperplane, a screen shot of your code, a copy of your code in your .py file for this question.

(ii) Fit a hard margin linear SVM to the **transformed** data-set in the previous part[2]. What are the estimated values of $(\alpha_1, \ldots, \alpha_7)$. Based on this, which points are the support vectors? What error does your computed SVM achieve?
What to submit: the indices of your identified support vectors, the train error of your SVM, the computed α's (rounded to 3 d.p.), a screen shot of your code, a copy of your code in your .py file for this question.

(iii) Consider now the kernel $k(x, z) = (2 + x^\top z)^2$. Run a hard-margin kernel SVM on the **original** (untransformed) data given in the table at the start of the question. What are the estimated values of $(\alpha_1, \ldots, \alpha_7)$. Based on this, which points are the support vectors? What error does your computed SVM achieve?
What to submit: the indices of your identified support vectors, the train error of your SVM, the computed α's (rounded to 3 d.p.), a screen shot of your code, a copy of your code in your .py file for this question.

(iv) Provide a detailed argument explaining your results in parts (i), (ii) and (iii). Your argument should explain the similarities and differences in the answers found. In particular, is your answer in (iii) worse than in (ii)? Why? To get full marks, be as detailed as possible, and use mathematical arguments or extra plots if necessary.
What to submit: some commentary and/or plots. If you use any code here, provide a screen shot of your code, and a copy of your code in your .py file for this question.

[2] If you are using the `SVC` class in `sklearn`, to get a hard-margin svm, you need to set the hyper parameter `C` to be very large.

(b) (**Test Set Linear Regression**) Recall that the rMSE (root-MSE) of a hypothesis[3] h on a **test** set $\{(x_i, y_i)\}_{i=1}^n$, where x_i is a p-dimensional vector and y_i is a real number, is defined as

$$\text{rMSE}(h) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - h(x_i))^2}.$$

For all parts, assume that the x_i's are known to you, but the y_i's are not. Suppose however that you are permitted to query rMSE(h). What this means is that you can query, for any hypothesis h, the rMSE of h on the test set. Suppose further that you know that rMSE(z) $= c_0$, where z is the hypothesis that returns zero for any input, i.e. $z(x_i) = 0$ for each $i = 1, \ldots, n$, and c_0 is some arbitrary positive number.

(i) Assume you have a set of T hypotheses $\mathcal{H} = \{h_1, h_2, \ldots, h_T\}$. You are told that for each i, there is a hypothesis h in $\mathcal{H}$, such that $h(x_i) = y_i$. In other words, for any point in the test set, there is at least one hypothesis in $\mathcal{H}$ that predicts that point correctly[4]. Suppose that you are permitted to blend predictions of different hypotheses in $\mathcal{H}$[5]. Design a naive, brute-force algorithm that constructs a hypothesis g from the elements of $\mathcal{H}$ such that rMSE(g) $= 0$. How many queries of rMSE do you need to make? How does your algorithm scale (in the worst case) with the test size n? Describe your algorithm in detail.
What to submit: a description of your algorithm and the number of queries required, either typed or handwritten.

(ii) We now consider a better approach than brute-force. For a given hypothesis h, define

$$h(X) = (h(x_1), h(x_2), \ldots, h(x_n))^\top$$
$$y = (y_1, y_2, \ldots, y_n)^\top.$$

We can always compute $h(X)$, but we do not know y. What is the smallest number of queries required to compute $y^\top h(X)$? Describe your approach in detail.
What to submit: a description of your approach and the number of queries required, either typed or handwritten.

(iii) Give a set of K hypotheses $\mathcal{H} = \{h_1, \ldots, h_K\}$, use your result in the previous part to solve

$$\min_{\alpha_1,\ldots,\alpha_K} \text{rMSE}\left(\sum_{k=1}^{K}\alpha_k h_k\right),$$

and obtain the optimal weights $\alpha_1, \ldots, \alpha_K$. Describe your approach in detail, and be sure to detail how many queries are needed and the exact values of the α's, in terms of X, y and the elements of $\mathcal{H}$.
What to submit: a description of your algorithm and the number of queries required, either typed or handwritten.

[3] The term hypothesis just means a function or model that takes as input x and returns as output a prediction $h(x) = \hat{y}$.
[4] Note that this does not imply that there is some hypothesis in $\mathcal{H}$ that predicts all points correctly.
[5] For example, you could construct a blended hypothesis g that returns the predictions of h_2 on test points 1 to 5, and the predictions of h_5 on points 6 to n.

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 if you wish. The parts are worth 4 + 3 + 8 + 1 + 3 + 2 + 3 + 1 = 25.

In the 9417 group project, many of you applied gradient boosting, which is sometimes regarded as the best out-of-the-box learning algorithm. In this question, we will derive and implement the gradient boosting algorithm from scratch, and apply it to a toy data set. The gradient boosting algorithm can be described as follows: Let $\mathcal{F}$ be a set of base learning algorithms[6], and let $\ell(y, \hat{y})$ be a loss function, where y is the target, and $\hat{y}$ is the predicted value. Note that this set-up is general enough to include both regression and classification algorithms. Suppose we have data (x_i, y_i) for $i = 1, \ldots, n$, which we collect into a single data set D_0. We then set the number of desired base learners to T, and proceed as follows:

(I) Initialize $f_0(x) = 0$ (i.e. f_0 is the zero function.)

(II) For $t = 1, 2, \ldots, T$:

(GB1) Compute:

$$r_{t,i} = -\frac{\partial}{\partial f(x_i)} \sum_{j=1}^{n} \ell(y_j, f(x_j)) \Big|_{f(x_j)=f_{t-1}(x_j),\ j=1,\ldots,n}$$

for $i = 1, \ldots, n$. We refer to $r_{t,i}$ as the i-th pseudo-residual at iteration t.

(GB2) Construct a new pseudo data set, D_t, consisting of pairs: $(x_i, r_{t,i})$ for $i = 1, \ldots, n$.

(GB3) Fit a model to D_t using our base class $\mathcal{F}$. That is, we solve

$$h_t = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(r_{t,i}, f(x_i))$$

(GB4) Choose a step-size. This can be done by **either** of the following methods:

(SS1) Pick a fixed step-size $\alpha_t = \alpha$

(SS2) Pick a step-size adaptively according to

$$\alpha_t = \arg\min_{\alpha} \sum_{i=1}^{n} \ell(y_i, f_{t-1}(x_i) + \alpha h_t(x_i)).$$

(GB5) Take the step

$$f_t(x) = f_{t-1}(x) + \alpha_t h_t(x).$$

(III) return f_T.

We can view this algorithm as either a generalization of AdaBoost covered in lectures/labs, or as performing (functional) gradient descent on the base class $\mathcal{F}$. Note that in (GB1), the notation means that after taking the derivative with respect to $f(x_i)$, set all occurences of $f(x_j)$ int he resulting expression with the prediction of the current model $f_{t-1}(x_j)$, for all j. For example,

$$\frac{\partial}{\partial x} \log(x+1) \Big|_{x=23} = \frac{1}{x+1} \Big|_{x=23} = \frac{1}{24}.$$

[6] For example, you could take $\mathcal{F}$ to be the set of all depth-1 decision trees, often referred to as decision stumps. Alternatively, $\mathcal{F}$ could be the set of all depth-2 trees, or the set of all 1-layer neural networks, etc.

(a) Consider the regression setting where we allow the y-values in our data set to be real numbers. Suppose that we use squared error loss $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$. For round t of the algorithm, show that $r_{t,i} = y_i - f_{t-1}(x_i)$. Then, write down an expression for the optimization problem in step (GB3) that is specific to this setting (you don't need to actually solve it).

What to submit: your working out, either typed or handwritten.

(b) Using the same setting as in the previous part, show that the adaptive approach (SS2) leads to a step-size:

$$\alpha_t = \frac{\sum_{i=1}^{n} h_t(x_i)(y_i - f_{t-1}(x_i))}{\sum_{i=1}^{n} (h_t(x_i))^2}.$$

What to submit: your working out, either typed or handwritten.

(c) We will now implement gradient boosting on a toy dataset from scratch, and we will use the class of decision stumps (depth 1 decision trees) as our base class ($\mathcal{F}$), and squared error loss as in the previous parts. In your implementation, you may make use of `sklearn.tree.DecisionTreeRegressor`, but all other code must be your own[7]. The following code generates the data and demonstrates plotting the predictions of a fitted decision tree (more details in `q3.py`):

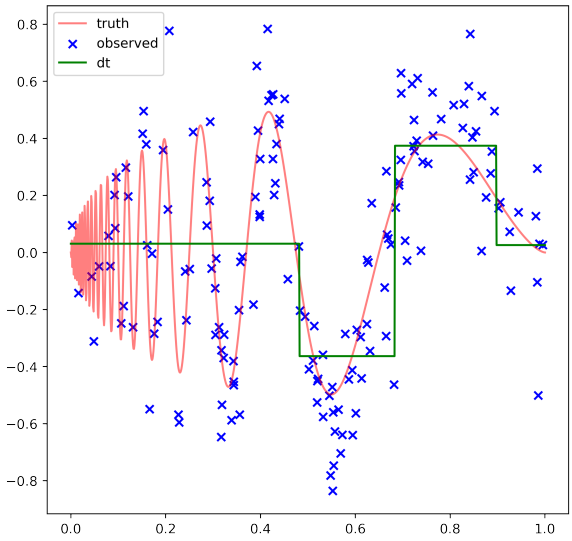
```
np.random.seed(123)
X, y = f_sampler(f, 160, sigma=0.2)
X = X.reshape(-1,1)

fig = plt.figure(figsize=(7,7))
dt = DecisionTreeRegressor(max_depth=2).fit(X,y)               # example model
xx = np.linspace(0,1,1000)
plt.plot(xx, f(xx), alpha=0.5, color='red', label='truth')
plt.scatter(X,y, marker='x', color='blue', label='observed')
plt.plot(xx, dt.predict(xx.reshape(-1,1)), color='green', label='dt')  # plotting
example model
plt.legend()
plt.show()

```

The figure generated is

[7] you can use `NumPy` and `matplotlib`, but do not use an existing implementation of gradient boosting.



Your task is to generate a 5 x 2 figure of subplots showing the predictions of your fitted gradient boosted model. There are 10 subplots in total, the first should show the model with 5 base learners, the second subplot should show it with 10 base learners, etc. The last subplot should be the gradient boosted model with 50 base learners. Each subplot should include the scatter of data, as well as a plot of the true model (basically, the same as the plot provided above but with your implemented gradient boosted model in place of `dt`). Comment on your results, what happens as the number of base learners is increased? You should do this two times (two 5x2 plots), once with the adaptive step size, and the other with the step-size taken to be $\alpha = 0.1$ fixed throughout. There is no need to split into train and test data here. Comment on the differences between your fixed and adaptive step-size implementations. How does your model perform on different parts of the data?

What to submit: two 5 x 2 plots, one for adaptive and one for fixed step size, some commentary, and a screen shot of your code and a copy of your code in your .py file.

(d) Repeat the analysis in the previous question but with depth 2 decision trees as base learners instead. Provide the same plots. What do you notice for the adaptive case? What about the non-adaptive case? *What to submit: two 5 x 2 plots, one for adaptive and one for fixed step size, some commentary, and a copy of your code in your .py file.*

(e) Now, consider the classification setting where y is taken to be an element of $\{-1, 1\}$. We consider the following classification loss: $\ell(y, \hat{y}) = \log(1 + e^{-y\hat{y}})$. For round t of the algorithm, what is $r_{t,i}$? Write down an expression for the optimization problem in step (GB3) that is specific to this setting (you don't need to actually solve it).

What to submit: your working out, either typed or handwritten.

(f) Using the same setting as in the previous part, write down an expression for α_t using the adaptive approach in (SS2). Can you solve for α_t in closed form? Explain.

What to submit: your working out, either typed or handwritten, and some commentary.

(g) Consider a different classification loss: $\ell(y, \hat{y}) = e^{-y\hat{y}}$. For round t of the algorithm, what is $r_{t,i}$? Write down an expression for the optimization problem in step (GB3) that is specific to this setting (you don't need to actually solve it). Further, can you find an expression for α_t in (SS2) for this setting? Explain.

What to submit: your working out, either typed or handwritten.

(h) In practice, if you cannot solve for α_t exactly, explain how you might implement gradient boosting. Assume that using a constant step-size is not a valid alternative. Be as specific as possible in your answer. What, if any, are the additional computational costs of your approach relative to using a constant step size ?

What to submit: some commentary.

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 if you wish. The parts are worth (2+3) + (3+3) + (2+3+3) + (1+1+4) = 5+6+8+6= 25.

Throughout this question, try to keep your answers as brief as possible. For discussion questions you need to provide 2-3 sentences of explanation at most. For calculation questions, please circle your final answers and show all working.

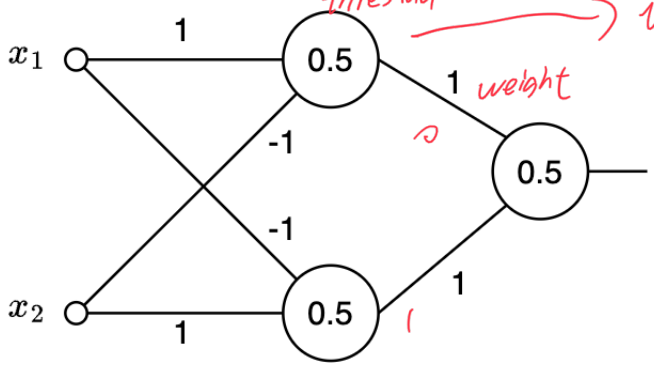
(a) (**Perceptron Power**)

(i) Consider the following Boolean functions $f_1, f_2, \ldots, f_5$ which each take two binary inputs x_1 and x_2, and return a binary output. Their outputs can be represented in the following table (each column represents the output of f_i on the corresponding inputs x_1 and x_2.)

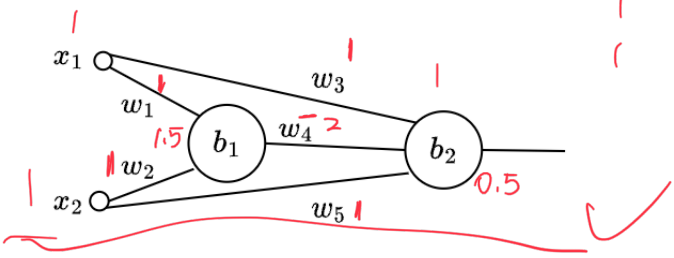
x_1	x_2	f_1	f_2	f_3	f_4	f_5
0	0	0	0	0	0	1
0	1	0	1	1	0	0
1	0	0	0	1	1	0
1	1	0	1	0	0	1

Which of the functions can be learned exactly by a perceptron? Explain why.
What to submit: some commentary.

(ii) Consider the following two layer network which implements the XOR function.



The numbers on the edges represent weights, and the numbers inside the nodes represent thresholds. The nodes return 1 if the inputis larger than the threshold value. For example, the top node in the middle layer takes as input $w_1x_1 + w_2x_2 = x_1 - x_2$, and if this is larger than the threshold $b_1 = 0.5$, outputs a value of 1, and 0 otherwise. You should check that this network implements the XOR function by testing it for all possible boolean inputs (x_1, x_2), and comparing it to the XOR truth table. It turns out that a more efficient representation of the XOR function can be constructed, which is of the following form:



Provide weights and thresholds $w_1, \ldots, w_5$ and b_1, b_2 that implement the XOR function using this new architecture. Demonstrate that the network works by computing its output on all possible inputs. Here, your weights $w_1, \ldots, w_5$ must be whole numbers, e.g. $\pm 1, \pm 2, \pm 3, \ldots$, and your thresholds b_1, b_2 should be chosen from $0.5, 1, 1.5, 2, 2.5, \ldots$. Try to find the smallest thresholds/weights that work.

What to submit: the weights and thresholds for the new network, along with a demonstration that the network does what it is meant to.

(b) (**K-means**) Note that the sub-questions in this question are independent of each other.

(i) Consider the following two-dimensional points: $\{(5,3),(4,4),(6,3),(5,4),(2,3)\}$. You run K-means (using Euclidean distance) on this dataset with $K = 2$ and initial cluster centers $C_0 = \{(5,2),(4,5)\}$, where the subscript 0 denotes that these are the cluster centers at time 0. Compute C_1 and C_2, the cluster centers after 1 and 2 iterations of the K-means algorithm. Be sure to detail your working.

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

(ii) Your friend, who studied accounting, is now promoted to the role of Data Scientist at his consulting company. They excitedly tell you that they're working on a clustering problem. 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) (**Learning Theory**) Note that the sub-questions in this question are independent of each other.

(i) Consider a hypothesis space $\mathcal{H}$ consisting of 1000 hypotheses. At least how many samples must you provide the learning algorithm in order to assure that with probability .98, the learner will output a hypothesis in $\mathcal{H}$ with true accuracy .95?

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

(ii) 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, the size of H must be finite.

Which of the statements are true? Which are false? Explain why or provide a counter-example.

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

(iii) Consider the hypothesis class $\mathcal{V}$ which contains all hypotheses of the form

$$v_{a,b}(x) = \begin{cases} 1 & \text{if } x \in [a,b] \\ 0 & \text{therwise} \end{cases}$$

So the classifier assigns a positive label to points inside the interval $[a, b]$, and zero to anything outside the interval. The class $\mathcal{V}$ contains all such classifiers (i.e. one classifier for each interval $[a, b]$ for every pair of real numbers a and b). What is the VC dimension of $\mathcal{V}$? Explain.
What to submit: your answer, and any working either typed or handwritten.

(d) (**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.

(i) What is the entropy of Y?
What to submit: your answer, and any working either typed or handwritten.

(ii) Suppose that feature X is chosen for the root of a decision tree. What is the information gain associated with this attribute?
What to submit: your answer, and any working either typed or handwritten.

(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.
What to submit: your answer, and any working either typed or handwritten.