Assignment 2: Regression/Probability

Due October 29 at 11:59pm

This assignment is to be done individually.

Important Note: The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

DO NOT:

- Give/receive code or proofs to/from other students
- Use Google to find solutions for assignment

DO:

- Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
- Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment

---

Submitting Your Assignment

The assignment must be submitted online on Canvas. In order to simplify grading, you must adhere to the following structure.

You must submit two files:

1. You must create an assignment report in **PDF format**, called `report.pdf`. This report must contain the solutions to questions 1, 3 as well as the figures / explanations requested for 2.(please take screenshots from your entire screen for the figures requested for question 2.)

2. You must submit a .zip file of all your code, called `code.zip`. This must contain a single directory called `code` (no sub-directories, no leading path names), in which all of your files must appear[1]. There must be the 3 scripts with the specific names referred to in Question 2, as well as a common codebase you create and name.

   As a check, if one runs

   ```
   unzip code.zip
   cd code
   ./polynomial_regression_1d.py
   ```

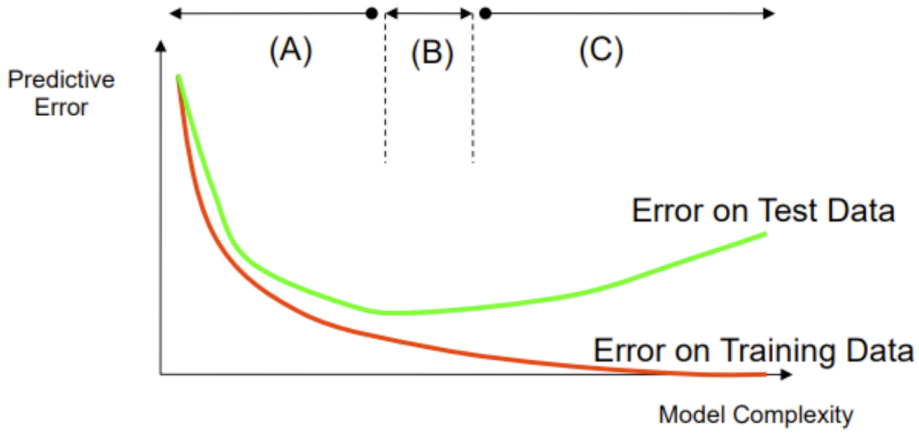
   the script produces the plots in your report from the relevant question.

[1] This includes the data files and others which are provided as part of the assignment.

1 Cross Validation

a) In order to get an unbiased estimate of how well our ML models and algorithms perform, it is typical to do a 60/20/20 split for our data: 60% train, 20% test and 20% validation. Several years ago this was widely considered best practice in machine learning. Do you still agree such ratios in the modern big data era? Why or why not? (Hint: consider different scenarios)

b) Consider the hypothetical graph below of predictive error (y-axis) vs. model complexity (x-axis), and how test/training error varies as model complexity increases.



i) Which part of the plot means that the model is Overfitting on the data? (Choose A, B, or C)

ii) Which part of the plot means that the model is Underfitting on the data? (Choose A, B, or C)

iii) Which part of the plot representing the ideal model complexity? (Choose A, B, or C)

c) For a decision tree model, whose train/test errors are in region A, which of the following is most likely to improve the model performance on real data? (Choose one)

i) Acquire more training data.

ii) Reduce the depth of the decision tree.

iii) Increase the depth of the decision tree.

Part A

I will start by stating the "need" for each of the phases (training, validation and testing) and what would happen if not enough data is dedicated to each of the phases:

- Sufficient amount of Training data is important as it ensures that our model, out of many models (validation), finds the best solution. In other words, if very little portion of the data is dedicated to the training phase, the model will just not learn the underlying trends that we want it to learn. To abstract it out, the variance would be very high if the training set is small.
- Validation phase picks out the best model from a bunch of models (We are essentially creating a new model every time we tune the hyperparameter). If insufficient amount of data is in the validation set, there will be a lot of noise in the estimated "best" model.
- Testing Phase is used to measure how our "best" model will perform while working with unseen datapoints. If not enough datapoints are present in the testing set, our assessment of the model's "generalization" might not be accurate.

The typical data split used to be 60/20/20 a few years back. This was because the data requirement was not as colossal as it is now. 60/20/20 is a perfect split for scenarios where the dataset is not huge, like in Natural Sciences (Medicine, Physics etc). The split 60/20/20 is still valid for such fields because data collection is very difficult (It is difficult to get actual data - not synthesized) and if we go any lower than 20 for testing and validation, we are actually not dedicating enough data points to the validation and testing phase, for these specific datasets. Usually the data points in such datasets are in the order of thousands, unlike modern Computer Vision dataset (where the datapoints are in the order of millions). Now to talk about a field like Computer Vision, even 10% of a dataset with 1 million samples is equal to 100,000 which is a lot (sufficient, to say the least). For current datasets (1 million datapoints) and models (consider UNET, which has 23 convolutional layers) it is better, in my opinion, to have "a lot" (enough) data in the training phase. A 23 Conv layer model is complex enough and the underlying trends are difficult for the model to learn, so *we should dedicate just enough datapoints to the validation and testing phases and then everything else should be included in the training set*. The actual splitting is done based on **Intuition**, **Experimentation**, **Tried and tested split** (from research papers tackling a similar problem/same dataset), **Searching the space** (Grid Search etc). [Comic]

Part B

i) Which part of the plot means that the model is Overfitting on the data? **C**

ii) Which part of the plot means that the model is Underfitting on the data? **A**

iii) Which part of the plot representing the ideal model complexity? **B**

Part C

For a decision tree model, whose train/test errors are in region A, which of the following is most likely to improve the model performance on real data? (Choose one)
Increase the depth of the decision tree.

2 Regression

In this question you will train models for regression and analyze a dataset. Start by downloading the code and dataset from the website.

The dataset is created from data provided by UNICEF's State of the World's Children 2013 report: `http://www.unicef.org/sowc2013/statistics.html`

Child mortality rates (number of children who die before age 5, per 1000 live births) for 195 countries, and a set of other indicators are included.

2.1 Getting started

Run the provided script `polynomial_regression.py` to load the dataset and names of countries / features.

Answer the following questions about the data. Include these answers in your report.

1. Which country had the highest child mortality rate in 1990? What was the rate?
2. Which country had the highest child mortality rate in 2011? What was the rate?
3. Some countries are missing some features (see original .xlsx/.csv spreadsheet). How is this handled in the function `assignment2.load_unicef_data()`?

For the rest of this question use the following data and splits for train/test and cross-validation.

- **Target value**: column 2 (Under-5 mortality rate (U5MR) 2011)[2].
- **Input features**: columns 8-40.
- **Training data**: countries 1-100 (Afghanistan to Luxembourg).
- **Testing data**: countries 101-195 (Madagascar to Zimbabwe).
- **Cross-validation**: subdivide training data into folds with countries 1-10 (Afghanistan to Austria), 11-20 (Azerbaijan to Bhutan), ... I.e. train on countries 11-100, validate on 1-10; train on 1-10 and 21-100, validate on 11-20, ...

2.2 Polynomial Regression

Implement linear basis function regression with polynomial basis functions. Use only monomials of a single variable (eg. x_1, x_1^2, x_2^2) and no cross-terms (eg. $x_1 x_2$).

Perform the following experiments:

a) Create a python script `polynomial_regression.py` for the following.

[2]Zero-indexing, hence `values[:,1]`.

Fit a polynomial basis function regression (unregularized) for degree 1 to degree 8 polynomials. Plot training error and test error (in RMS error) versus polynomial degree.

Put this plot in your report, along with a brief comment about what is "wrong" in your report.

Normalize the input features before using them (not the targets, just the inputs $\boldsymbol{x}$). Use `assignment2.normalize_data()`.

Run the code again, and put this new plot in your report.

b) Create a python script `polynomial_regression_1d.py` for the following.

Perform regression using just a single input feature.

Try features 8-15 (Total population - Low birthweight). For each (un-normalized) feature fit a degree 3 polynomial (unregularized).

Plot training error and test error (in RMS error) for each of the 8 features. This should be a bar chart (e.g. use `matplotlib.pyplot.bar()`).

Put this bar chart in your report.

The testing error for feature 11 (GNI per capita) is very high. To see what happened, produce plots of the training data points, learned polynomial, and test data points. The code `visualize_1d.py` may be useful.

In your report, include plots of the fits for degree 3 polynomials for features 11 (GNI), 12 (Life expectancy), 13 (literacy).

2.3 Regularized Polynomial Regression

Create a python script `polynomial_regression_reg.py` for the following.

Implement L_2-regularized regression. Fit a degree 2 polynomial using $\lambda = \{0, .01, .1, 1, 10, 10^2, 10^3, 10^4, 10^5\}$. Use normalized features as input. Use 10-fold cross-validation to decide on the best value for λ. Produce a plot of average validation set error versus λ. Use a `matplotlib.pyplot.semilogx` plot, putting λ on a log scale[3].

Put this plot in your report, and note which λ value you would choose from the cross-validation.

Getting Started

1. Which country had the highest child mortality rate in 1990? What was the rate?
   Niger: 313.7 (for only `x train` it's Angola: 243.2).

2. Which country had the highest child mortality rate in 2011? What was the rate?
   Sierra Leone: 185.3 (for only `x train` it's Chad: 169.0).

[3]The unregularized result will not appear on this scale. You can either add it as a separate horizontal line as a baseline, or report this number separately.

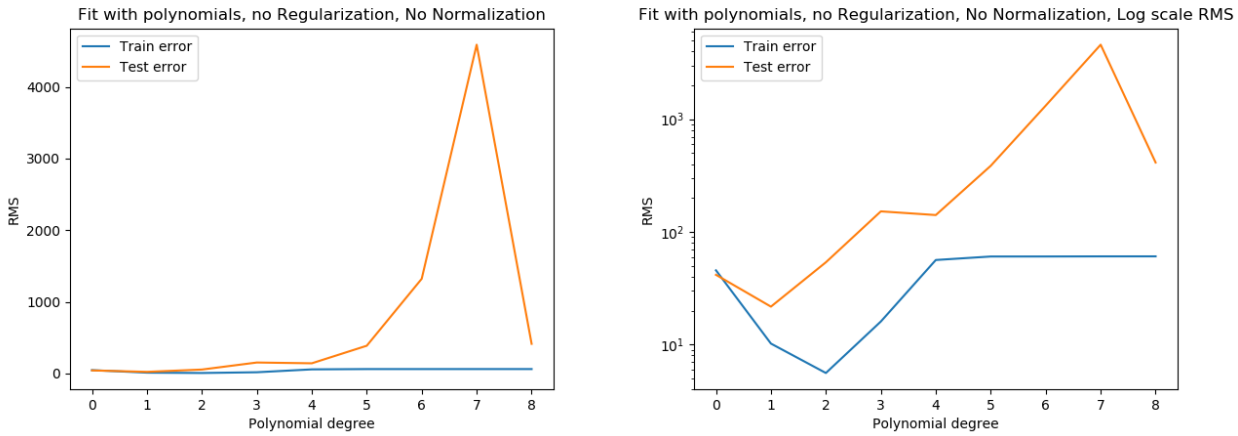
3. Some countries are missing some features (see original .xlsx/.csv spreadsheet). How is this handled in the function `assignment2.load unicef data()`?
   NaN cells are filled with the column's mean

Polynomial Regression

Part A

I will be using the following formula to calculate the loss

$$L(\overrightarrow{w}) = \sqrt{\frac{\|\overrightarrow{y} - X\overrightarrow{w}\|_2^2}{N}}$$

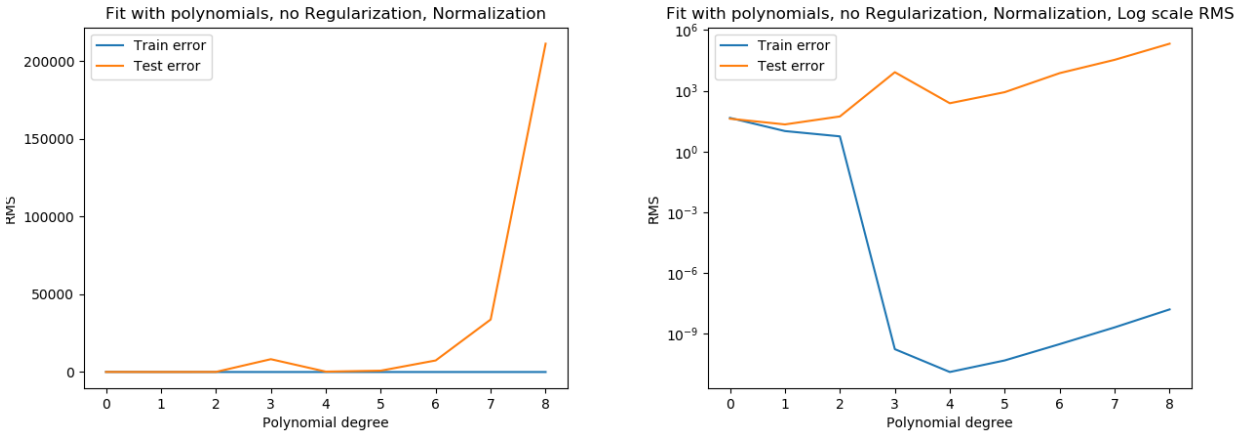


(a) Linear Scale

(b) Log Scale

Figure 1: Training and testing errors vs polynomial degree (Un-normalized)

The Problem with this approach is the training loss, which keeps on increasing after the 3rd degree polynomial (which is more visible if you look at the log scale of the losses), this problem goes away when we normalize the data.



(a) Linear Scale (b) Log Scale

Figure 2: Training and testing errors vs polynomial degree (Normalized)

The Normalized version of the code shows us that both of the losses are of much lower order (check out the log scale version of the graphs).

Part B

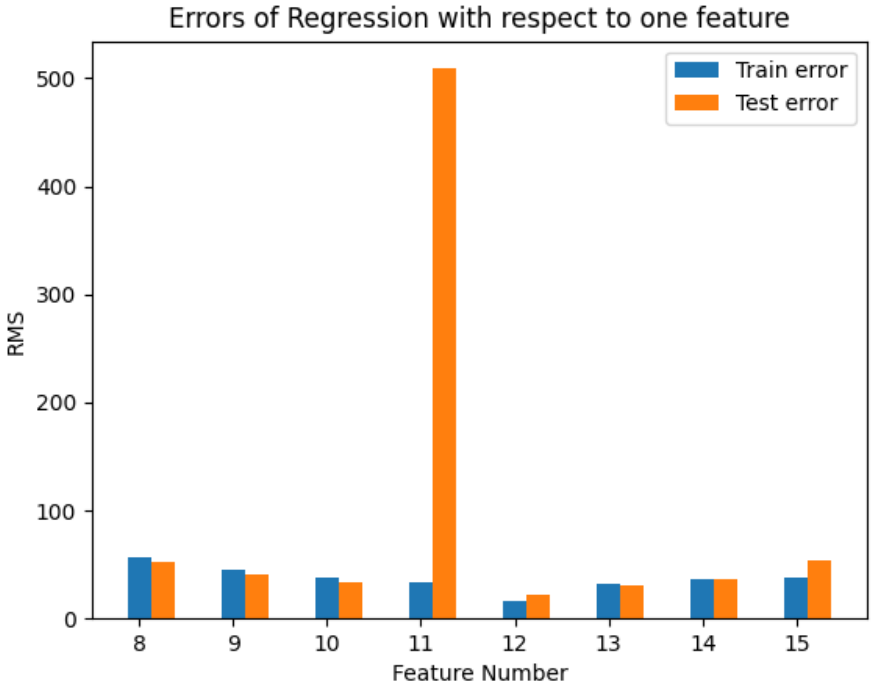
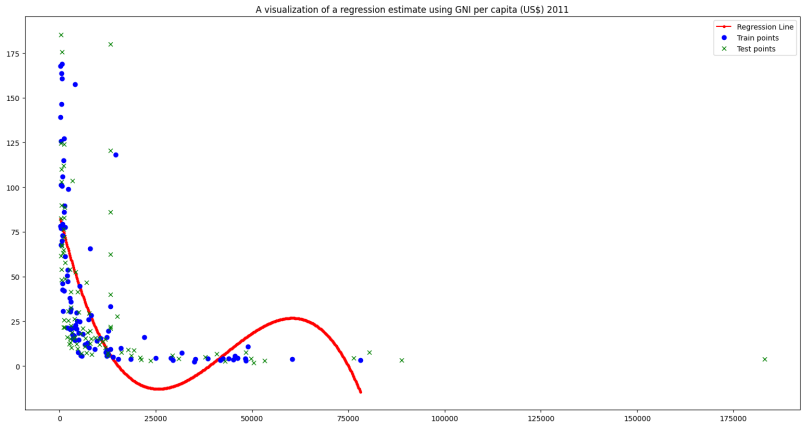
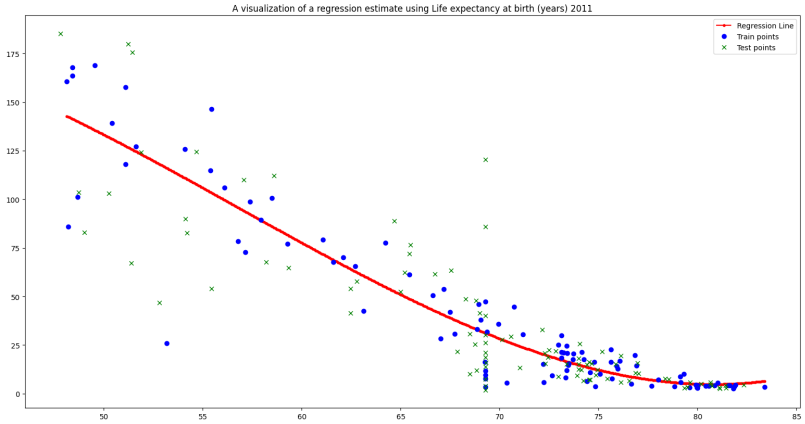


Figure 3: Training and testing errors for each feature column (8-15)

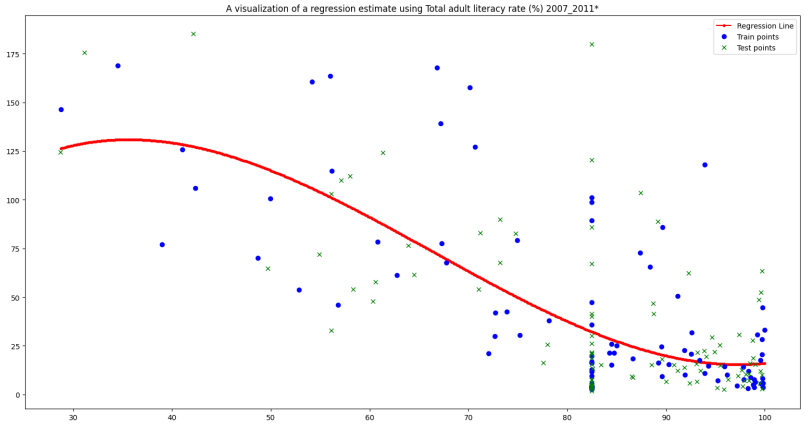
This Bar Chart shows that the testing error for feature 11 (GNI) is indeed very high. Now I will be plotting graphs for the three columns: 11, 12 and 13. It can be seen why the feature 11 gives a huge testing loss, there's one testing point which is miles away from all of the other points which does not fare well while calculating the losses. It can be solved if data is pre-processed and maybe is removed before using (pre-processing).



(a) Feature 11 GNI



(b) Feature 12 Life Expectancy



(c) Feature 13 Literacy

Figure 4: 3 degree polynomial fit for feature 11-13

Regularized Polynomial Regression

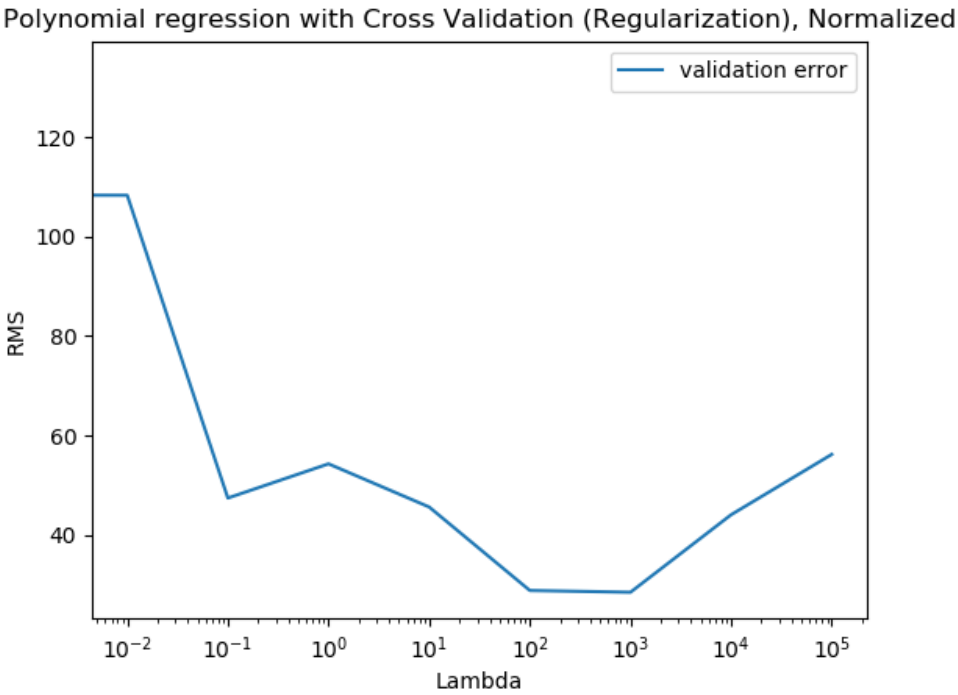


Figure 5: Regularized Polynomial Regression to find the optimal λ (10-fold cross validation)

The graph shows which value of λ is most suited, gives the least value of Root Mean Squared Error, for the given data. Which in this case is $\lambda = 1000$.

3 Probabilistic Modeling and Bayes Rule

a) Assume the probability of being infected with Malaria disease is 0.01. The probability of test positive given that a person is infected with Malaria is 0.95 and the probability of test positive given the person is not infected with Malaria is 0.05.
(a) Calculate the probability of test positive.
(b) Use Bayes Rule to calculate the probability of being infected with Malaria given that the test is positive.

b) Suppose P(rain today) = 0.30, P(rain tomorrow) = 0.60, P(rain today and tomorrow) = 0.25. Given that it rains today, what is the probability it will rain tomorrow?

c) A biased die has the following probabilities of landing on each face:

face	1	2	3	4	5	6
P(face)	0.2	0.1	0.1	0.2	0.1	0.3

i) I win if the die shows odd. What is the probability that I win? Compare this to a fair die (i.e., a die with equal probabilities for each face).

ii) What is the entropy of this die? Compare this to a fair die.

Part A

Firstly, mentioning all of the probabilities in mathematical form:

$$P(Malaria) = 0.01$$
$$P(Positive|Malaria = 1) = 0.95$$
$$P(Positive|Malaria = 0) = 0.05$$

As it clear that there is a decomposition in the given probability space, and $P(Malaria) + P(!Malaria) = 1$ (There can not be people which do not belong to one of the two categories) so it can be inferred that

$$P(!Malaria) = 0.99$$

Subpart a

And for such probability distributions we can use the following formula

$$P(A) = \Sigma_i P(A|B_i)P(B_i)$$

In this case we have

$$P(Positive) = P(Positive|Malaria = 1)P(Malaria)+P(Positive|Malaria = 0)P(!Malaria)$$

$$P(Positive) = (0.95)(0.01) + (0.05)(0.99)$$

$$P(Positive) = 0.059$$

So, the probability of test positive is 0.059.

Subpart b

Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

In our case it becomes

$$P(Malaria|Positive) = \frac{P(Positive|Malaria)P(Malaria)}{P(Positive)}$$

$$P(Malaria|Positive) = \frac{(0.95)(0.01)}{0.059}$$

$$P(Malaria|Positive) = 0.1610$$

Part B

The given data is

$$P(Rain_{today}) = 0.3$$
$$P(Rain_{tomorrow}) = 0.60$$
$$P(Rain_{today} \cap Rain_{tomorrow}) = 0.25$$

This question requires the formula of conditional probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(Rain_{tomorrow}|Rain_{today}) = \frac{P(Rain_{today} \cap Rain_{tomorrow})}{P(Rain_{today})}$$

$$P(Rain_{tomorrow}|Rain_{today}) = \frac{0.25}{0.3}$$

So if it rains today the chance of it raining tomorrow is:

$$P(Rain_{tomorrow}|Rain_{today}) = 0.833$$

Part C

Subpart i

The probability that I win is equal to the probability that an odd number appears. We just sum the probabilities of the odd numbers , i.e., 1, 3 and 5. So summing them up.

$$P(odd) = 0.2 + 0.1 + 0.1$$
$$P(odd) = 0.4$$

For a fair dice all sides appear equal number of times so the Probability of any number (1-6) appearing is $\frac{1}{6}$ or 0.1666. But intuitively it can be said that odd numbers appear half of the time or the probability of an odd number appearing is 0.5. So just to verify the probability of odd numbers appearing:

$$P(odd) = 0.1666 * 3 \qquad or \qquad P(odd) = 0.1666 + 0.1666 + 0.1666$$
$$P(odd) = 0.5$$

Subpart ii

Entropy is given by the formula:

$$H(x) = -\Sigma_{i=1}^{n} P(x_i) log(P(x_i))$$

So to calculate the entropy of this biased dice:

$$H(biasedDice) = -P(1)ln(P(1)) - P(2)ln(P(2)) - P(3)ln(P(3)) - P(4)ln(P(4))$$
$$-P(5)ln(P(5)) - P(6)ln(P(6))$$

$$H(biasedDice) = -(0.2)(-1.609) - (0.1)(-2.302) - (0.1)(-2.302) - (0.2)(-1.609) \\ -(0.1)(-2.302) - (0.3)(-1.203)$$

$$H(biasedDice) = 0.3218 + 0.2302 + 0.2302 + 0.3218 + 0.2302 + 0.3609$$

This is the entropy of the biased dice

$$H(biasedDice) = 1.6951$$

For a fair dice we use the exact same formula or we can just multiply 6 with the $-P(x)log(x)$.

$$H(fairDice) = -P(1)ln(P(1)) - P(2)ln(P(2)) - P(3)ln(P(3)) - P(4)ln(P(4)) \\ -P(5)ln(P(5)) - P(6)ln(P(6))$$

$$H(fairDice) = -(0.1666)(-1.791) - (0.1666)(-1.791) - (0.1666)(-1.791) \\ -(0.1666)(-1.791) - (0.1666)(-1.791) - (0.1666)(-1.791)$$

$$H(fairDice) = 0.2984 + 0.2984 + 0.2984 + 0.2984 + 0.2984 + 0.2984$$

This is the Entropy of a Fair Dice

$$H(fairDice) = 1.7909$$