CSC311 A1

Benson Li

February 2, 2023

1 Nearest Neighbours and the Curse of Dimensionality.

1.1 (a)

We need at least 100 data points.
Reason: In order to guarantee that any new test point is within 0.01 of an old point, we need a ”cutoff” data-point per 0.01 distance.
Therefore, we need at least $\frac{1-0}{0.01} = 100$ data-points.

1.2 (b)

In the 10 dimension space, we need use closed balls with $r = 0.01$ to fill the space.
Since closed balls with $r = 0.01$ in 10 dimension space, has the volume $\frac{\pi^5}{120}(0.01)^{10}$, we need about $5 \cdot 10^{19}$ balls to fill the space of $[0, 1]^{10}$

This number is very large, which is hard to maintain. The reason behind this is that the number of possible data-point increases exponentially as we increase the number of features. This led to the difficulties in maintain any new test point is within 0.01 of an old point?

1.3 (c)

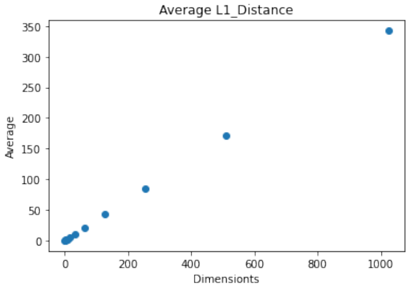


Figure 1: The Average Graph for L1 Distance

1.4 (d)

$$E[R] = E[\sum_{i=1}^{d} Z_i] = \sum_{i=1}^{d} E[Z_i] = \sum_{i=1}^{d} \frac{1}{6} = \frac{1}{6}d$$

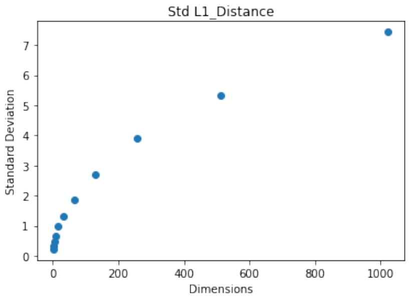


Figure 2: The Standard Deviation Graph for L1 Distance

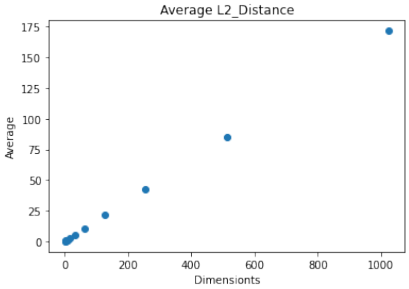


Figure 3: The Average Graph for L2 Distance

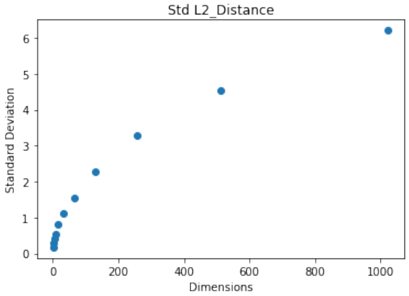


Figure 4: The Standard Deviation Graph for L1 Distance

$$\begin{aligned} Var[R] =& Var[\sum_{i=1}^{d} Z_i] \\ &= \sum_{i=1}^{d} Var[Z_i] \quad \textbf{Since each Z are independent} \\ &= \sum_{i=1}^{d} \frac{7}{180} = \frac{7}{180}d \end{aligned}$$

1.5 (e)

1.5.1 (i)

$$E : \left|R - E[R]\right| \geq k$$

1.5.2 (ii)

By Markov's Inequality, $P(E) = P(\left|R - E[R]\right| \geq k) \leq \frac{Var(R)}{k^2}$

1.5.3 (iii)

$$\lim_{k\to\infty} P(E) \leq \lim_{k\to\infty} \frac{Var(R)}{k^2} = 0 \quad \text{for Var(R)} < \infty$$

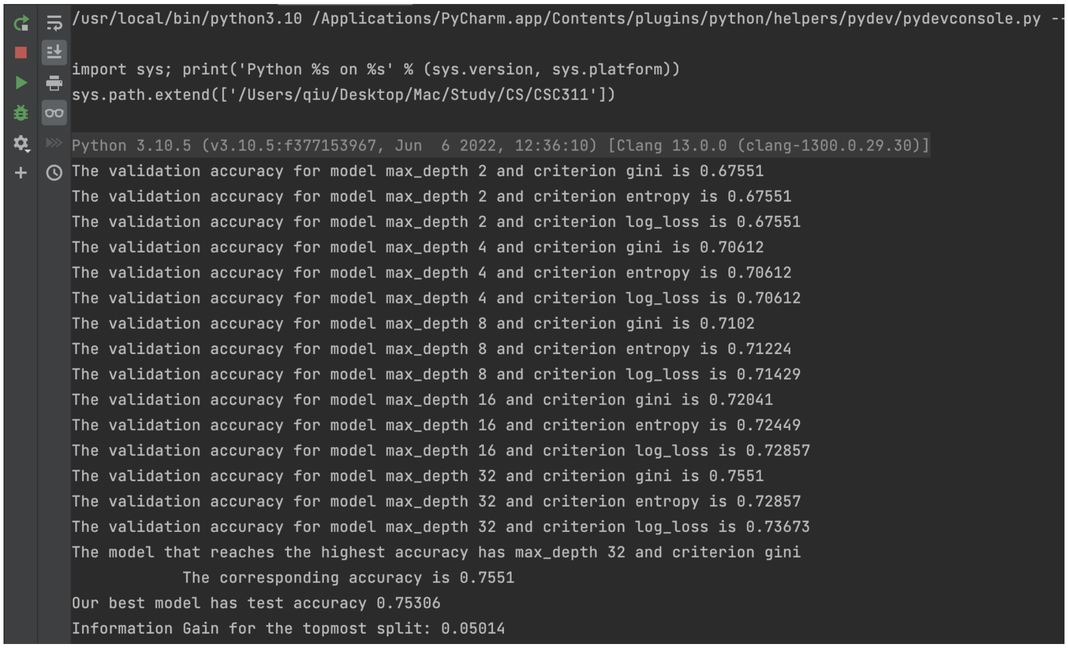
.

2 Decision Trees

2.1 (a)

Please refer to the file hw1_code.py

2.2 (b)

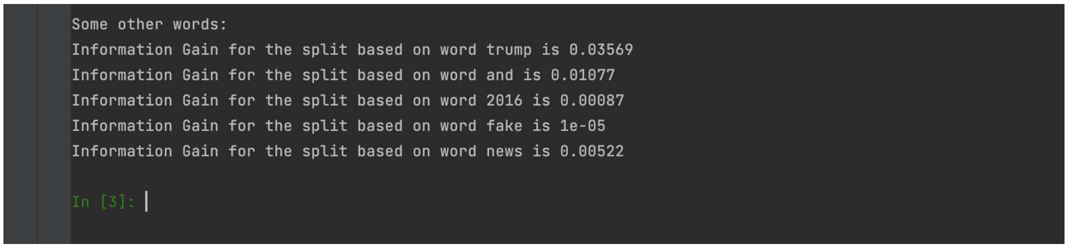


```
/usr/local/bin/python3.10 /Applications/PyCharm.app/Contents/plugins/python/helpers/pydev/pydevconsole.py --

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/qiu/Desktop/Mac/Study/CS/CSC311'])

Python 3.10.5 (v3.10.5:f377153967, Jun  6 2022, 12:36:10) [Clang 13.0.0 (clang-1300.0.29.30)]
The validation accuracy for model max_depth 2 and criterion gini is 0.67551
The validation accuracy for model max_depth 2 and criterion entropy is 0.67551
The validation accuracy for model max_depth 2 and criterion log_loss is 0.67551
The validation accuracy for model max_depth 4 and criterion gini is 0.70612
The validation accuracy for model max_depth 4 and criterion entropy is 0.70612
The validation accuracy for model max_depth 4 and criterion log_loss is 0.70612
The validation accuracy for model max_depth 8 and criterion gini is 0.7102
The validation accuracy for model max_depth 8 and criterion entropy is 0.71224
The validation accuracy for model max_depth 8 and criterion log_loss is 0.71429
The validation accuracy for model max_depth 16 and criterion gini is 0.72041
The validation accuracy for model max_depth 16 and criterion entropy is 0.72449
The validation accuracy for model max_depth 16 and criterion log_loss is 0.72857
The validation accuracy for model max_depth 32 and criterion gini is 0.7551
The validation accuracy for model max_depth 32 and criterion entropy is 0.72857
The validation accuracy for model max_depth 32 and criterion log_loss is 0.73673
The model that reaches the highest accuracy has max_depth 32 and criterion gini
            The corresponding accuracy is 0.7551
Our best model has test accuracy 0.75306
Information Gain for the topmost split: 0.05014
```

Figure 5: The output of the function 'select_model'



```
Some other words:
Information Gain for the split based on word trump is 0.03569
Information Gain for the split based on word and is 0.01077
Information Gain for the split based on word 2016 is 0.00087
Information Gain for the split based on word fake is 1e-05
Information Gain for the split based on word news is 0.00522

In [3]: |
```

Figure 6: The output for computing IG for some other words

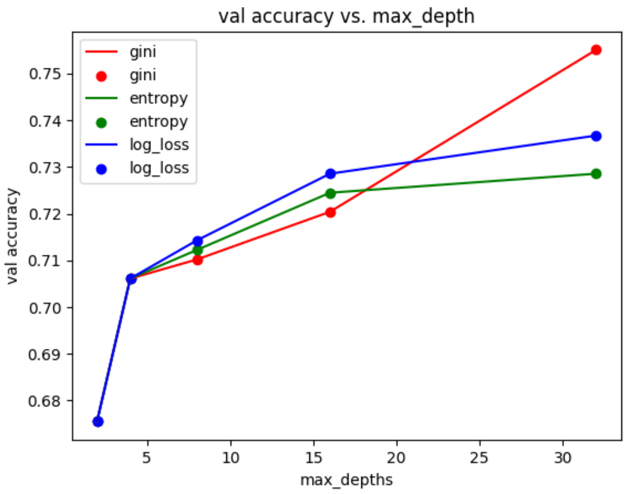


Figure 7: The plot for val accuracy vs. max_depth

2.3 (c)

Please refer to the Figure 8 below

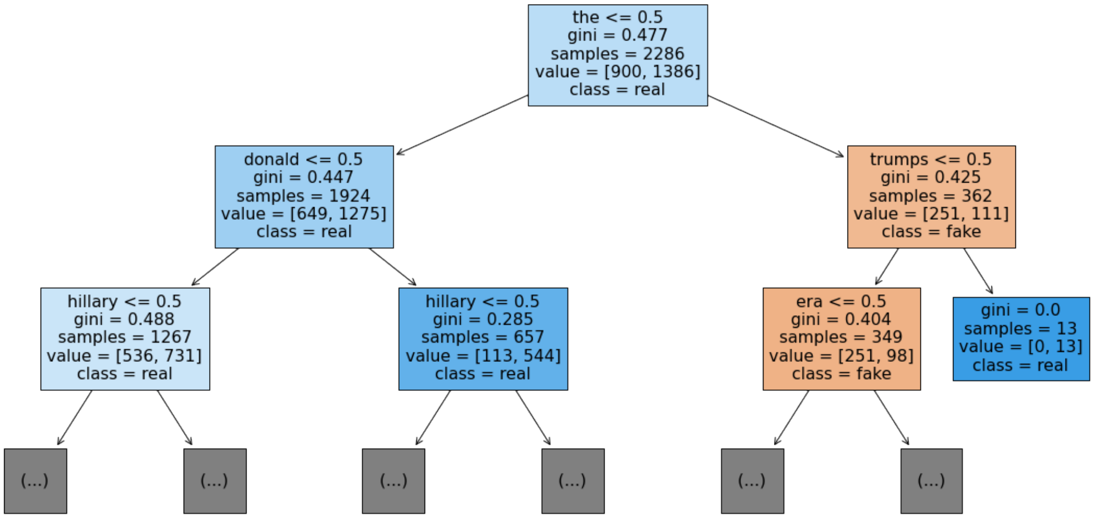


Figure 8: The visualization of the best tree model

2.4 (d)

Please refer to the file hw1_code.py

3 Regularized Linear Regression.

3.1 (a)

Denote k as the learning rate.
If $w_j > 0$:

$$w_j \leftarrow (1-k\beta_j)w_j - k(\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)}-t^{(i)}) + \alpha_j)$$

$$b \leftarrow b - k(\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}-t^{(i)}))$$

If $w_j = 0$:

$$w_j \leftarrow (1-k\beta_j)w_j - k(\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)}-t^{(i)}))$$

$$b \leftarrow b - k(\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}-t^{(i)}))$$

If $w_j < 0$:

$$w_j \leftarrow (1-k\beta_j)w_j - k(\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)}-t^{(i)}) - \alpha_j)$$

$$b \leftarrow b - k(\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}-t^{(i)}))$$

Answer: notice that we have the term $(1-k\beta_j)w_j$ for each expression in updating w_j. When $k\beta_j > 0$, the relative contribution of the original w_j is decreasing for each time we update it.

3.2 (b)

$$\begin{aligned}
\frac{\partial J_{reg}^{\beta}}{\partial w_j} =& \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)}-t^{(i)}) + \beta_j w_j \\
=& \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}y^{(i)} - (\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}t^{(i)}) + \beta_j w_j \\
=& \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}\sum_{j'=1}^{D} x_{j'}^{(i)}w_{j'} - (\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}t^{(i)}) + \beta_j w_j \\
=& \frac{1}{N}\sum_{i=1}^{N}\sum_{j'=1}^{D} x_j^{(i)}x_{j'}^{(i)}w_{j'} - (\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}t^{(i)}) + \beta_j w_j \\
=& \sum_{j'=1}^{D}\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}x_{j'}^{(i)}w_{j'} - (\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}t^{(i)}) + \beta_j w_j
\end{aligned}$$

Therefore,

$$A_{jj'} = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}x_{j'}^{(i)}$$

$$c_j = (\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}t^{(i)}) - \beta_j w_j$$

3.3 (c)

Notice that we have

$$\sum_{j'=1}^{D} A_{jj'} w_{j'} = \frac{1}{N} \sum_{j'=1}^{D} (x_j)^T x_{j'} w_{j'}$$
$$= \frac{1}{N} (x_j)^T X w$$

$$c_j = \frac{1}{N} (x_j)^T \bar{t} - \beta_j w_j$$

Then, we have equation:

$$\frac{1}{N} (x_j)^T X w = \frac{1}{N} (x_j)^T \bar{t} - \beta_j w_j$$

Then, collapsing the equations for all partials:

$$\frac{1}{N} X^T X w = \frac{1}{N} X^T \bar{t} - \beta^T w$$

$$(\frac{1}{N} X^T X + \beta^T I) w = \frac{1}{N} X^T \bar{t}$$

The closed-form solution is:

$$w = (\frac{1}{N} X^T X + \beta^T I)^{-1} \frac{1}{N} X^T \bar{t}$$