ML HW2

Maitar Asher

03/06/2023

1 Designing socially aware classifiers

(i) It is not enough to just remove the sensitive attribute A from the dataset to achieve fairness because it is very likely that the sensitive attribute bias could leak via other attributes.

Let's take for instance a classifier that determines if a student should be accepted into Columbia and have gender be our sensitive attribute, with male and female as its two possible values. If we remove the sensitive attribute from the dataset to prevent bias towards one gender, there may still be other attributes in the dataset that unintentionally favor one gender over the other (e.g extracurricular activities attributes: volunteering work, sports, etc, grades, recommendations). If we train our classifier on this set of data and remove the gender attribute, the resulting classifier may still exhibit gender-based discrimination.

(ii)

- Suppose $\mathbb{P}(\hat{y}=1) = \mathbb{P}_a(\hat{y}=1)\ \forall a \in \{0,1\}$
  - then $\mathbb{P}_0(\hat{y}=1) = \mathbb{P}_1(\hat{y}=1) = \mathbb{P}(\hat{y}=1)$
  - thus $\mathbb{P}_0(\hat{y}=1) = \mathbb{P}_1(\hat{y}=1)$
- Suppose $\mathbb{P}_0(\hat{y}=1) = \mathbb{P}_1(\hat{y}=1)$
  - $\mathbb{P}(\hat{y}=1) = \mathbb{P}_0(\hat{y}=1)\mathbb{P}(A=0) + \mathbb{P}_1(\hat{y}=1)\mathbb{P}(A=1)$ by total probability rule
  - We can treat $\mathbb{P}_0(\hat{y}=1)$ and $\mathbb{P}_1(\hat{y}=1)$ as a common factor because we assumed they are equal
  - then $\mathbb{P}(\hat{y}=1) = \mathbb{P}_a(\hat{y}=1)(\mathbb{P}(A=0)+\mathbb{P}(A=1)) = \mathbb{P}_a(\hat{y}=1)(1) = \mathbb{P}_a(\hat{y}=1)$
  - thus $\mathbb{P}(\hat{y}=1) = \mathbb{P}_a(\hat{y}=1)\ \forall a \in \{0,1\}$
- thus we have showed that $\mathbb{P}_0(\hat{y}=1) = \mathbb{P}_1(\hat{y}=1) \iff \mathbb{P}(\hat{y}=1) = \mathbb{P}_a(\hat{y}=1)\ \forall a \in \{0,1\}$

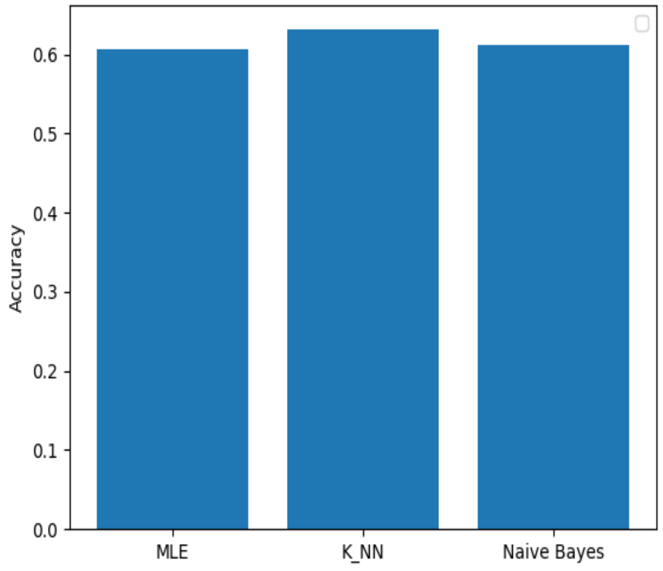
- Note: $\mathbb{P}_0(\hat{y}=1) = \mathbb{P}(\hat{y}=1|A=0)$, $\mathbb{P}_1(\hat{y}=1) = \mathbb{P}(\hat{y}=1|A=1)$

(iii) If we are to generalize the above equivalence for some fixed $a \in \mathbb{N}$ and some fixed $\hat{y} \in \mathbb{R}$:

$$\mathbb{P}(\hat{Y}=\hat{y}) = \mathbb{P}_a(\hat{Y}=\hat{y})$$

(iv) Code submitted in Coursework.

(v)

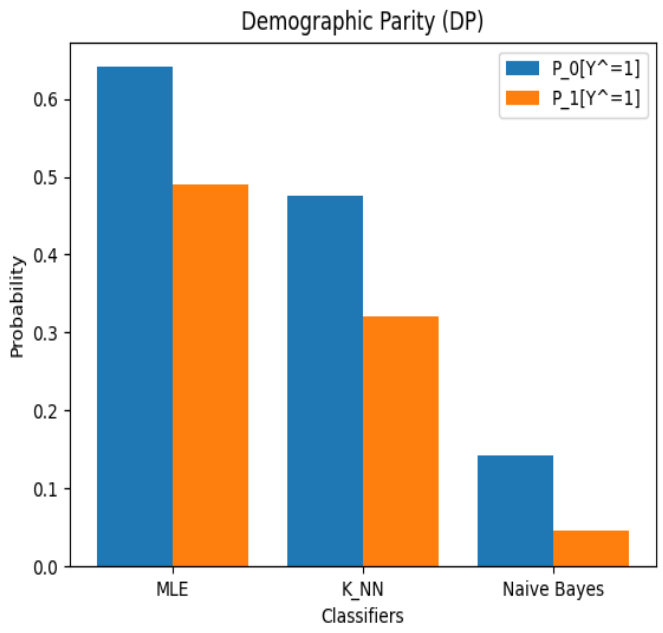


My accuracies are as followed:
MLE: 0.606
K NN: 0.6315 (doesn't change much with different k and distance metrics*)
Naive Bayes: 0.613

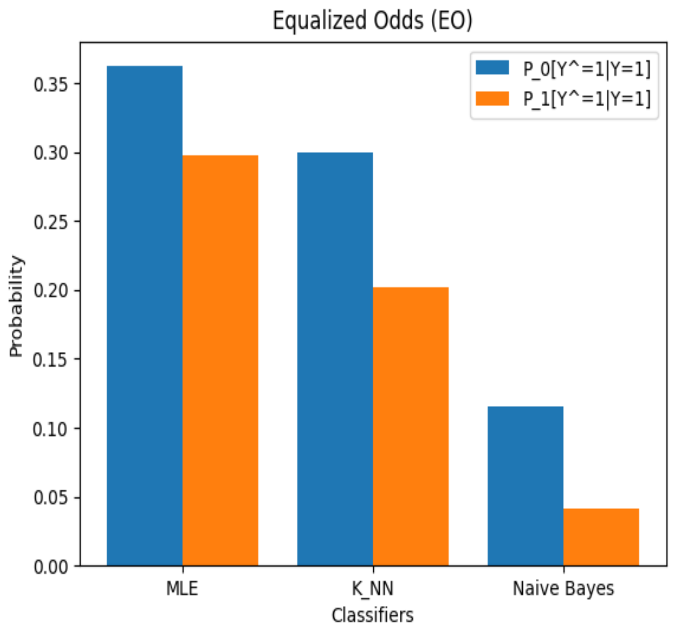
*
for $k = 3, d = 1$: 0.624
for $k = 3, d = 2$: 0.6235
for $k = 3, d = inf$: 0.613
for $k = 5, d = 1$: 0.636
for $k = 5, d = 2$: 0.633
for $k = 5, d = inf$: 0.633
for $k = 6, d = inf$: 0.6315

And so it seems that the K NN classifier performs slightly better than the others, but it is also worth mentioning that it is slower. I did not see that different training sample sizes affected the classification performance.

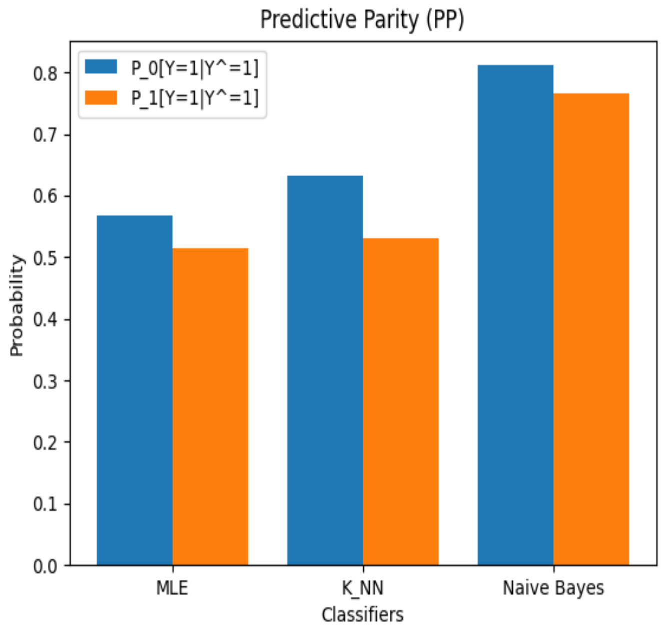
(vi)



For DP, Naive Bayes seems to be the fairest (although the gap is relatively the same)



For EO, MLE seems to be the fairest. [THIS IS THE GRAPH FOR TRUE POSITIVE]



For PP, MLE seems to be the fairest. [THIS IS THE GRAPH FOR EQUAL POSITIVE PREDICTIVE]

(vii) Take the Demographic Parity fairness definition which aims to ensure that selection rates are equal for different demographic groups. Such fairness definition is applicable in scenarios where there is a concern that an algorithm may unfairly disadvantage certain groups. For example, a bank using an algorithm to approve loans may aim for demographic parity to prevent discrimination against minorities group. However, potential disadvantages of this fairness definition may be a reduction in overall accuracy and perpetuating existing biases.

2 Data dependent perceptron mistake bound

(i)

Dataset: $\{((3,0), y=1), ((-3,0), y=-1)\}$

1. Initizlize $\vec{w}^(0) = (0,0)$
2. checking $((3,0), y = 1) \rightarrow sign((0,0) \cdot (3,0)) = 0 \neq 1$ so we update $\vec{w}^(1) = (0,0) + (3,0) = (3,0)$, $(t = 1)$
3. checking $((-3,0), y = -1) \rightarrow sign((3,0) \cdot (-3,0)) = -1 = -1$ No mistake
4. at this point there will be no more mistakes and the boundary will classify all correctly so $T = 1$, we made one mistake
5. At this point the decision boundary will be on the y-axis (w is orthogonal to the decision boundary), so $R = 3$ and $\gamma = 3$
6. $1 = (\frac{3}{3})^2$ Thus $T \leq (\frac{R}{\gamma})^2$ and we showed that it is tight – it can make exactly $(\frac{R}{\gamma})^2$ mistakes

(ii)

$||\vec{w}_T||^2 = ||\vec{w}_{(T-1)} + y_{i(T)}\vec{x}_{i(T)}||^2 = ||\vec{w}_{(T-1)}||^2 + 2y_{i(T)}(\vec{w}_{(T-1)} \cdot \vec{x}_{i(T)}) + ||y_{i(T)}\vec{x}_{i(T)}||^2$

$= ||\vec{w}_{(T-1)}||^2 + 2y_{i(T)}(\vec{w}_{(T-1)} \cdot \vec{x}_{i(T)}) + (y_{it}^2)||\vec{x}_{i(T)}||^2 = ||\vec{w}_{(T-1)}||^2 + 2y_{i(T)}(\vec{w}_{(T-1)} \cdot \vec{x}_{i(T)}) + (1)||\vec{x}_{i(T)}||^2$

Now let's similarity expand $||\vec{w}_{(T-1)}||^2$

$= ||\vec{w}_{(T-2)}||^2 + 2y_{i(T-1)}(\vec{w}_{(T-2)} \cdot \vec{x}_{i(T-1)}) + ||\vec{x}_{i(T-1)}||^2 + 2y_{i(T)}(\vec{w}_{(T-1)} \cdot \vec{x}_{i(T)}) + ||\vec{x}_{i(T)}||^2$

if we continue to expand $||\vec{w}_{(T-i)}||^2$ we will stop when $T = 1$ which uses $||\vec{w}_0||^2$

$$= ||\vec{w}_0||^2 + \sum_{t=1}^{T} 2y_{it}(\vec{w}_{(t-1)} \cdot \vec{x}_{i(t)}) + \sum_{t=1}^{T} ||\vec{x}_{it}||^2$$

We know that $||\vec{w}_0||^2 = 0$ and that for all t the expression $2y_{it}(\vec{w}_{(t-1)} \cdot \vec{x}_{i(t)})$ will be negative because at any time t, a mistake was made, meaning the sign of y_{it} will differ from the sign of $(\vec{w}_{(t-1)} \cdot \vec{x}_{i(t)})$. Thus we can conclude that:

$$||\vec{w}_T||^2 \leq \sum_{t=1}^{T} ||\vec{x}_{it}||^2$$

Now let's examine $\sum_{t=1}^{T} ||\vec{x}_{it}||^2$.

$$\sum_{t=1}^{T} ||\vec{x}_{it}||^2 = \sum_{t=1}^{T} ||\vec{x}_{it} - P\vec{x}_{it} + P\vec{x}_{it}||^2 = \sum_{t=1}^{T} ||(I - P)\vec{x}_{it} + P\vec{x}_{it}||^2$$

$$= \sum_{t=1}^{T} ||(I-P)\vec{x}_{it}||^2 + \sum_{t=1}^{T} 2((I-P)\vec{x}_{it} \cdot P\vec{x}_{it}) + \sum_{t=1}^{T} ||P\vec{x}_{it}||^2$$

Because we know that $\max_{xi} \in S||(I-P)x_i|| \leq \epsilon < R$,
we know that $\sum_{t=1}^{T} ||(I-P)\vec{x}_{it}||^2 \leq \sum_{t=1}^{T} \epsilon = T\epsilon \leq \epsilon^2 T$

Lets examine $2((I-P)\vec{x}_{it} \cdot P\vec{x}_{it})$:
Because $(I-P)$ is the projector onto the orthogonal complement space of w^* we can infer than $(I-P)\vec{x}_{it}$ yields a vector that is orthogonal to w^* (their dot product equals to 0).
$P\vec{x}_{it} = w^* {w^*}^T \vec{x}_{it} = w^* c$ where c is some scalar $c = {w^*}^T \vec{x}_{it}$ Thus the dot product $(I-P)\vec{x}_{it} \cdot P\vec{x}_{it} = 0$

Then
$$\sum_{t=1}^{T} ||\vec{x}_{it}||^2 \leq \epsilon^2 T + \sum_{t=1}^{T} ||P\vec{x}_{it}||^2$$

Thus
$$||\vec{w}_T||^2 \leq \epsilon^2 T + \sum_{t=1}^{T} ||P\vec{x}_{it}||^2$$

(iii)
$$\vec{w}_T \cdot \vec{w}^* = (\vec{w}_{(T-1)} + y_{iT}(\vec{x}_{iT})) \cdot \vec{w}^*$$

Now let's similarity expand $\vec{w}_{(T-1)}$
$$= (\vec{w}_{(T-2)} + y_{i(T-1)}(\vec{x}_{i(T-1)}) + y_{iT}(\vec{x}_{iT})) \cdot \vec{w}^*$$

if we continue to expand $\vec{w}_{(T-i)}$ we will stop when $T = 1$ which uses $\vec{w}_0 = 0$
$$= (\vec{w}_0 + \sum_{t=1}^{T} y_{it}\vec{x}_{i(t)}) \cdot \vec{w}^* = \left(\sum_{t=1}^{T} y_{it}\vec{x}_{i(t)} \cdot \vec{w}^* \right)$$

Squaring the expression yields
$$(\vec{w}_T \cdot \vec{w}^*)^2 = \left(\sum_{t=1}^{T} y_{it}\vec{x}_{i(t)} \cdot \vec{w}^* \right)^2 = \sum_{t=1}^{T} \left(y_{it}\vec{x}_{i(t)} \cdot \vec{w}^* \right)^2$$

Using the formula for the expansion of the square of the sum of N numbers (cited in README.txt) we get:
$$\sum_{t=1}^{T} \left(y_{it}\vec{x}_{i(t)} \cdot \vec{w}^* \right)^2 = \sum_{t=1}^{T} \left(y_{it}\vec{x}_{i(t)} \cdot \vec{w}^* \right)^2 + 2\sum_{t=1}^{T}\sum_{j=1}^{t-1} \left(y_{it}\vec{x}_{i(t)} \cdot \vec{w}^* \right) \left(y_{ij}\vec{x}_{i(j)} \cdot \vec{w}^* \right)$$

Because we are given that $\forall i \; y_i(w^* \cdot x_i) \geq \gamma$
$$2\sum_{t=1}^{T}\sum_{j=1}^{t-1} \left(y_{it}\vec{x}_{i(t)} \cdot \vec{w}^* \right) \left(y_{ij}\vec{x}_{i(j)} \cdot \vec{w}^* \right) \geq 2T(T-1)\gamma^2$$

Now let's examine the first term in the equation:
Let $\vec{m}$ denote $y_{it}\vec{x}_{i(t)}$
then $\left(y_{it}\vec{x}_{i(t)} \cdot \vec{w}^*\right)^2 = (\vec{m} \cdot \vec{w}^*)^2$ Let us now look at $P\vec{m} = \vec{w}^*\vec{w}^{*T}\vec{m}$
Then $||P\vec{m}|| = ||\vec{w}^*\vec{w}^{*T}\vec{m}||$ where note that $\vec{w}^{*T}\vec{m} = \vec{w}^* \cdot \vec{m} = \vec{m} \cdot \vec{w}^*$ is some scalar
Then $||P\vec{m}||^2 = (\vec{w}^*\vec{w}^{*T}\vec{m})^2 = ||\vec{w}^*||(\vec{w}^* \cdot \vec{m})^2 = (\vec{m} \cdot \vec{w}^*)^2$
So

$$\sum_{t=1}^{T} \left(y_{it}\vec{x}_{i(t)} \cdot \vec{w}^*\right)^2 = \sum_{t=1}^{T} ||P\vec{m}_t||^2 = \sum_{t=1}^{T} ||P(y_{it}\vec{x}_{i(t)})||^2 = \sum_{t=1}^{T} ||P\vec{x}_{i(t)}||^2$$

Thus

$$(\vec{w}_T \cdot \vec{w}^*)^2 \geq \sum_{t=1}^{T} ||P\vec{x}_{i(t)}||^2 + 2T(T-1)\gamma^2$$

And so

$$(\vec{w}_T \cdot \vec{w}^*)^2 \geq \sum_{t=1}^{T} ||P\vec{x}_{i(t)}||^2 + T(T-1)\gamma^2$$

(iv) We have showed that:

- $||\vec{w}_T||^2 \leq \epsilon^2 T + \sum_{t=1}^{T} ||P\vec{x}_{it}||^2$
- $(\vec{w}_T \cdot \vec{w}^*)^2 \geq \sum_{t=1}^{T} ||P\vec{x}_{it}||^2 + T(T-1)\gamma^2$

We also know that

- $\vec{w}_T \cdot \vec{w}^* = ||\vec{w}_T||||\vec{w}^*||cos\theta = ||\vec{w}_T||cos\theta$ because $\vec{w}^*$ is a unit vector
- then $(\vec{w}_T \cdot \vec{w}^*)^2 = ||\vec{w}_T||^2 cos^2\theta \leq ||\vec{w}_T||^2$ because $0 \leq cos^2(\theta) \leq 1$

  $(\vec{w}_T \cdot \vec{w}^*)^2 \leq ||\vec{w}_T||^2$

Thus

$$\sum_{t=1}^{T} ||P\vec{x}_{it}||^2 + T(T-1)\gamma^2 \leq \epsilon^2 T + \sum_{t=1}^{T} ||P\vec{x}_{it}||^2$$

$$T(T-1)\gamma^2 \leq \epsilon^2 T$$

$$(T-1)\gamma^2 \leq \epsilon^2$$

$$T\gamma^2 \leq \epsilon^2 + \gamma^2$$

$$T \leq \left(\frac{\epsilon}{\gamma}\right)^2 + 1$$

3 Constrained optimization

Note there is only one constraint $g(x) = w \cdot x + w_0 = 0$

$$L(x, \lambda) = ||x - x_a||^2 + \lambda(w \cdot x + w_0)$$

$$\frac{\partial}{\partial x} L(x, \lambda) = 2(x - x_a) + \lambda w = 0$$

$$2(x - x_a) = -\lambda w$$

$$x = -\frac{\lambda w}{2} + x_a$$

Now we can maximize for gamma

$$\frac{\partial}{\partial \lambda} L(x, \lambda) = (w \cdot x + w_0) = 0$$

Plugging x into the equation, we get:

$$-\frac{\lambda w^2}{2} + w x_a + w_0 = 0$$

$$\frac{\lambda w^2}{2} = w x_a + w_0$$

$$\lambda w^2 = 2 w x_a + 2 w_0$$

$$\lambda = \frac{2 w x_a + 2 w_0}{w^2} = \frac{2(w x_a + w_0)}{w^2}$$

Putting it all in again in min x

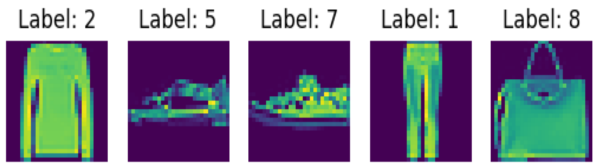
$$x = -\frac{\lambda w}{2} + x_a = -\frac{(\frac{2(w x_a + w_0)}{w^2}) w}{2} + x_a = -\frac{(w x_a + w_0)}{w} + x_a$$

Thus

$$||x - x_a|| = || - \frac{(w x_a + w_0)}{w} + x_a - x_a|| = ||\frac{(w x_a + w_0)}{w}||$$

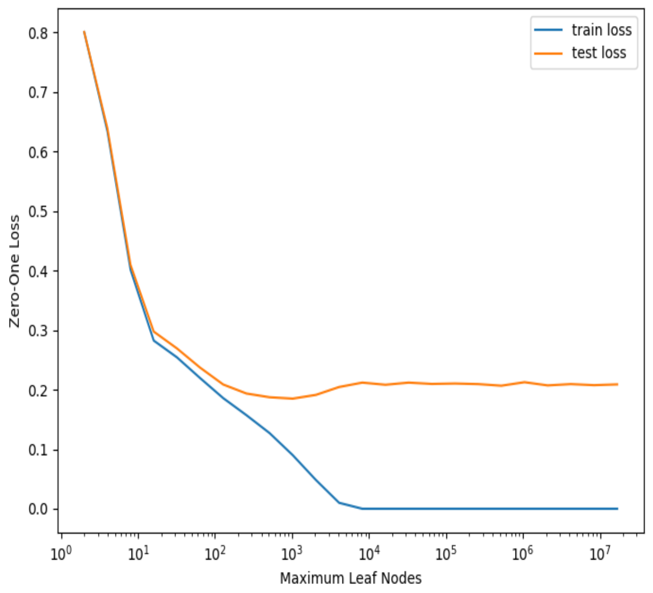
4 Decision Trees, Ensembling and Double Descent

(i)



FashionMNIST is a dataset that is similar in structure to the classical MNIST dataset but contains images of fashion items rather than handwritten digits. The FashionMNIST dataset consists of 10 different classes of fashion items, such as shoes, bags, dresses, and t-shirts, whereas the classical MNIST dataset consists of 10 classes of handwritten digits from 0 to 9. Each item in our training/testing data is a 28x28 matrix, a 2d array of pixel values representing some fashion item (instead of a digit item). Although both datasets are similar in structure, the data in FashionMNIST is more challenging to work with than the classical MNIST because it is more visually complicated–it is in colors (rather than in black and white) and the shapes they represent are more complicated.

A simple classifier like nearest neighbors is likely to do poorly on this Fashion-MNIST due to the high dimensionality and variability of the images. Nearest neighbors algorithm relies on finding the most similar samples in the feature space to a given test sample and predicting its label based on the majority label of the k closest samples. However, in high-dimensional spaces, the notion of similarity becomes less clear, and the curse of dimensionality makes it harder to find nearest neighbors that accurately represent the test sample. Therefore, more sophisticated methods, such as deep learning approaches, are needed to handle such high-dimensional datasets with high variability.



(ii)

Min loss for the test is ≈ 0.184 and for the train ≈ 0.09

As the number of allowed leaf nodes increases the tree becomes more complex. While the training error will decrease to zero, the test error will initially decrease but eventually, start to increase due to overfitting. Overfitting leads to excessive sensitivity to noise in the training data, resulting in poor generalization to new data. When the model is overly complex and overfitting occurs, the variance is high, and we need to balance it with the bias to achieve minimum loss. Based on the image, it is advisable to stop splitting at approximately 10000 leaf nodes.

From print statements in code:
max leaf nodes: $[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]$
num leaves: $[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]$
min test loss: 0.18489999999999995
train loss: 0.09001666666666663

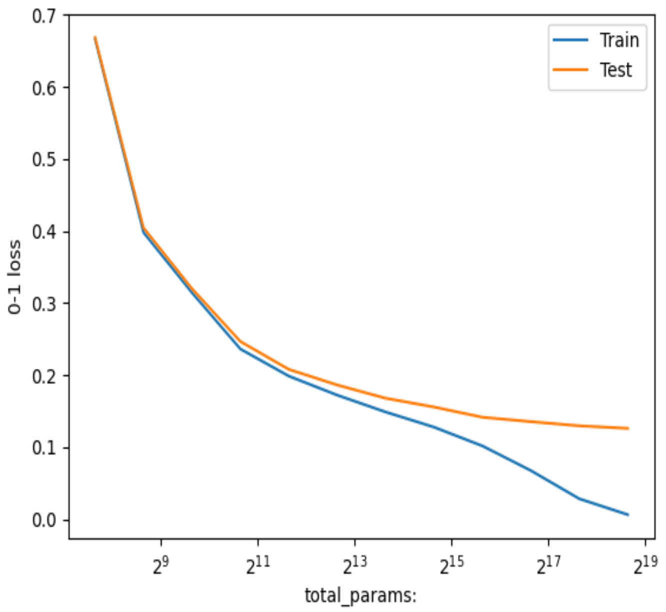
(iii) The decision tree classifiers didn't always utilize the complete capacity of allowed leaves, where the maximum number of leaves utilized by a trained classifier was approximately 5000. This is because the algorithm might terminate splitting early if it concludes that further splitting won't enhance the model's performance.

From print statements in code:
max leaf nodes: [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216]
num leaves actually used: [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 4912, 4905, 4912, 4910, 4907, 4912, 4911, 4908, 4915, 4911, 4905, 4910]

(iv) When each estimator in a random forest has access to only a subset of all features, it can help to reduce the correlation between the trees. If all the features were available to every tree, the trees would make similar splits and decisions, and the predictions would be highly correlated. In such cases, our random forest would perform well on the training set but poorly on the test set.

By using only a subset of features for each tree, the trees' splits and decisions are made on different subsets of the available features. This leads to more diverse trees and reduces the overall correlation between the trees. The lower the correlation between the trees, the less likely they are to make the same predictions and the better the random forest's generalization performance. Thus, utilizing a subset of features for each estimator in a random forest can help to reduce variance, leading to improved model performance.

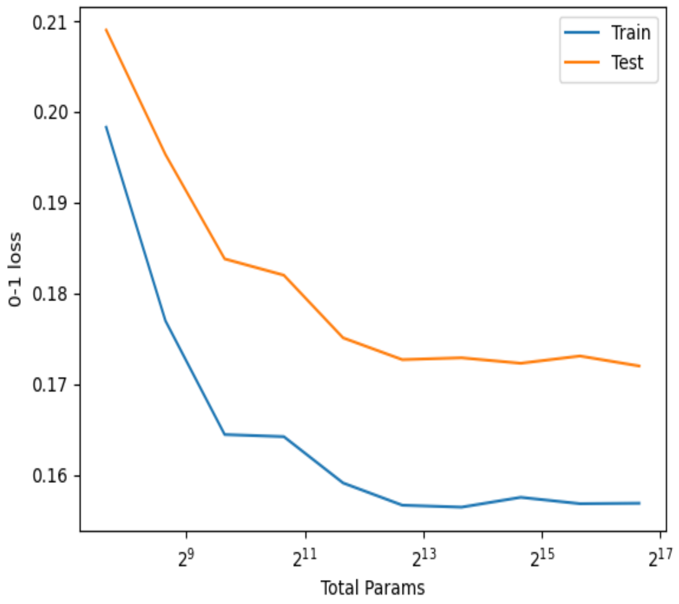
(v).a When observing the train and test 0-1 loss of a random forest classifier with a fixed number of estimators and a varying number of maximum allowed tree leaves for individual estimators, an overfit seems possible because the tree depth is not limited. As the complexity of the model can continue to grow, it



may result in overfitting.

From print statements in code:
'test loss:' 0.1261
min index: 11
train loss: 0.006266666666666643
total params: [200 400 800 1600 3200 6400 12800 25600 51200 102400 204800 409600]

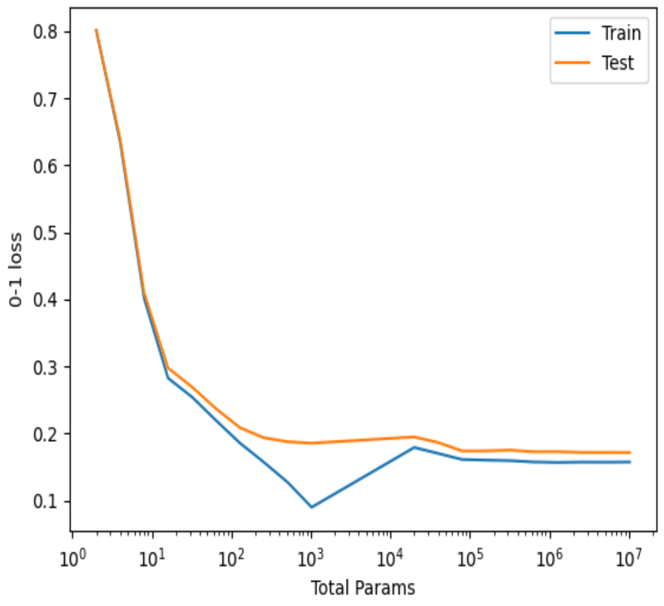
(v).b



My best loss is smaller compared to the best loss achieved with a single decision tree $\approx 0.1722 <\approx 0.184$ From print statements in code:
'test loss:', 0.17200000000000004
min index: 9
train loss: 0.15688333333333337
Total Params: [200 400 800 1600 3200 6400 12800 25600 51200 102400]

(vi)

The surprising result is is the sudden pick at $\approx 10^4 = 10000$ Given that there are 60,000 data points I don't think there is a relationship between the order of the number of parameters and the number of data points in the training set.


Train
Test
0-1 loss
Total Params
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7