Data 37711: Machine Learning & AI Part I **Fall 2025**

Homework 3

Student: Chenxi Peng Due date: November 20^{th}

1 Q1

1.1 (a)

i. Since $Y \in \mathbb{N}$, it is a positive integer used for counting. The Poisson distribution is a discrete probability distribution for non–negative integer counts and uses a single, positive mean parameter. So Poisson provides a simple, interpretable likelihood for count outcomes.

ii. The exponential is the canonical inverse link for the Poisson GLM, i.e.

$$\log \mu(x) = f(x) \quad \Longleftrightarrow \quad \mu(x) = \exp(f(x)),$$

which explains both the positivity and the canonical property.

1.2 (b) Loss function

For one observation (x, y) the likelihood under $\mu(x) = \exp(f(x))$ is

$$P(Y = y \mid x) = \frac{\exp(-\mu(x))\, \mu(x)^y}{y!}.$$

The negative log-likelihood is

$$\ell(f(x), y) = \exp(f(x)) - yf(x) + \log(y!).$$

Since $\log(y!)$ is constant with restrict to f, the loss function becomes

$$L(f) = \exp(f(x)) - yf(x).$$

1.3 (c) Gradient

$$\frac{\partial L(f)}{\partial f} = \exp(f(x)) - y$$

Thus, for observation i:

$$g_i := \frac{\partial L}{\partial f(x_i)} = \exp(f(x_i)) - y_i.$$

The negative gradient (pseudo-residual) is

$$r_i = y_i - \exp(f(x_i)).$$

1.4 (d) Choosing each h_j in first–order boosting

Initialization:

$$f^{(0)} = \log\left(\frac{1}{n}\sum_{i=1}^{n} y_i + 1\right),$$

so that $\exp(f^{(0)})$ matches the sample mean approximately.

Iteration at a regression tree j:

1. Compute pseudo-residuals (negative gradients)

$$r_i^{(j)} = y_i - \exp\left(f^{(j-1)}(x_i)\right).$$

2. Fit a regression tree h_j to $\{(x_i, r_i^{(j)})\}$ using least squares

$$h_j = \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} \left(r_i^{(j)} - h(x_i)\right)^2,$$

   where $\mathcal{H}$ is the class of regression trees.

3. Then the update should use the tree predictions:

$$f^{(j)}(x) = f^{(j-1)}(x) + \rho\, h_j(x),$$

   where $\rho \in (0, 1]$ is a learning rate.

2 Q2

2.1 (a) Generalization bound for hyperparameter selection

Suppose we have k candidate hyperparameter choices. For each choice $i \in \{1, \ldots, k\}$, training on the training set yields a hypothesis h_i. Let the test set consist of ℓ i.i.d. samples. Denote the empirical test risk by $\widehat{R}(h_i)$ and the true risk by $R(h_i)$.

Assume the loss is bounded in $[0, 1]$. By Hoeffding's inequality, for any fixed h_i and any $\varepsilon > 0$,

$$\Pr\left(\left|\widehat{R}(h_i) - R(h_i)\right| > \varepsilon\right) \le 2\exp(-2\ell\varepsilon^2).$$

Applying the union bound over the k hypotheses,

$$\Pr\left(\exists i\colon\ |\widehat{R}(h_i) - R(h_i)| > \varepsilon\right) \le 2k\exp(-2\ell\varepsilon^2).$$

Set the right side equal to δ and solve for ε:

$$\varepsilon = \sqrt{\frac{1}{2\ell}\ln\left(\frac{2k}{\delta}\right)}.$$

Thus, with probability at least $1 - \delta$, simultaneously for all i,

$$|\widehat{R}(h_i) - R(h_i)| \le \sqrt{\frac{1}{2\ell}\ln\left(\frac{2k}{\delta}\right)}.$$

Let

$$\hat{\imath} = \arg\min_i \widehat{R}(h_i)$$

be the hyperparameter chosen using the test set. The above uniform bound implies that, with probability at least $1 - \delta$,

$$\left| R(h_{\hat{\imath}}) - \widehat{R}(h_{\hat{\imath}}) \right| \leq \sqrt{\frac{1}{2\ell} \ln\left(\frac{2k}{\delta}\right)}.$$

This gives a non-trivial generalization bound for the model selected using the test set.

2.2 (b)

Using the same test set to select hyperparameters and to estimate generalization error leads to *selection bias*. Since the chosen model $h_{\hat{\imath}}$ minimizes the noisy empirical test risk among k options, its test performance is an optimistically biased estimate of its true risk.

When k is large relative to ℓ, the term

$$\sqrt{\frac{\ln k}{\ell}}$$

can be large, meaning that some model may appear to have extremely low test risk purely by chance. This is analogous to a multiple comparisons problem: trying many hyperparameters effectively "overfits" the test set itself.

2.3 (c)

To avoid this bias, the standard practice is to split the data into:

- a training set used to fit models,
- a validation set used to select hyperparameters,
- a test set used *only once* for final evaluation.

Alternatively, one may use cross-validation or nested cross-validation when data are limited. Separating the selection and evaluation procedures prevents overfitting to the test set and yields an unbiased estimate of generalization error.

2.4 (d)

I chose two model classes from the set {`RandomForestRegressor`, `KNeighborsRegressor`, `MLPRegressor`}: the Random Forest regressor and the k-Nearest Neighbors regressor. For each model class, I constructed a reasonably large hyperparameter grid, with the total number of hyperparameter combinations exceeding the number of points in the validation set, as required. The grids used are shown below:

```
# Random Forest hyperparameter grid
rf_param_grid = {
    "n_estimators": [20, 50, 100, 150, 200, 300, 400, 500, 600, 800],
    "max_depth": [None, 5, 10, 20, 30, 40, 50, 60, 70, 80],
    "min_samples_split": [2, 4, 6, 8, 10]
}

# KNN hyperparameter grid
knn_param_grid = {
    "n_neighbors": [i for i in range(1, 51)],
```

```
    "algorithm": ["auto", "ball_tree", "kd_tree"],
    "leaf_size": [30, 60],
    "weights": ["uniform", "distance"]
}
```

Using grid search over these hyperparameters, I trained each candidate model on the training set and selected the best hyperparameter configuration according to the mean squared error (MSE) on the validation set. The corresponding test-set performance for each selected model is:

```
=== Random Forest Results ===
Best hyperparameters: {'n_estimators': 150, 'max_depth': None, 'min_samples_split': 2}
Validation MSE: 0.3482961451247165
Test MSE: 0.40581505668934237

=== KNN Results ===
Best hyperparameters: {'n_neighbors': 15, 'algorithm': 'auto',
'leaf_size': 30, 'weights': 'distance'}
Validation MSE: 0.5004747598408696
Test MSE: 0.49142810852384455
```

To study the relationship between the empirical risk on the validation set and the empirical risk on the test set, I plotted a scatter plot of validation MSE (x-axis) versus test MSE (y-axis) across all hyperparameter combinations for both models, together with the identity line $y = x$ spanning the full range of both models' errors.

From the Figure 1, the Random Forest model shows a consistent pattern where the test MSE is slightly higher than the validation MSE for nearly all hyperparameter choices. In contrast, for the KNN model, the validation and test MSE values tend to lie close to the identity line, and in many cases the test MSE is slightly lower than the validation MSE. This indicates that, under this dataset and hyperparameter grid, KNN exhibits more stable generalization behavior across train–validation–test splits.

In addition, there is an obvious positive correlation between validation MSE and test MSE.

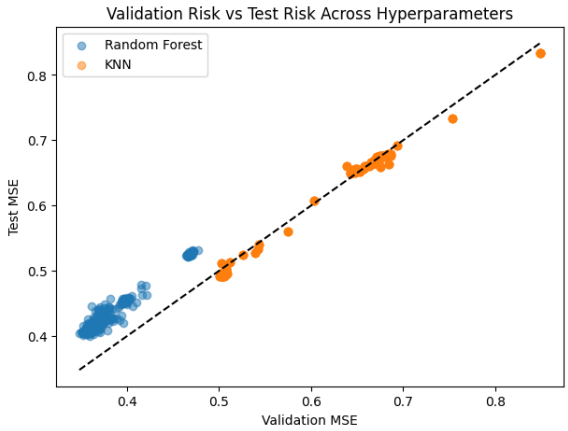

Figure 1: The relationship between the empirical risk on the validation set and the empirical risk on the test set for Random Forest and KNN models

The code can be found here

3 Q3

By plotting the empirical risk on both the training and test sets as the number of trees increases, we can compare the generalization behaviour of boosting on `dset0` and `dset1`. The result is shown in Figure 2.

For `dset0`, the behaviour is consistent with boosting applied to a relatively clean dataset. As the number of trees increases from 1 to 300, both the training error and the test error decrease. The test error reaches a value close to zero around 300 trees, while the training error also becomes very small. Beyond 300 trees, the test error rises slightly but does not exhibit severe overfitting. Overall, the generalization curve remains stable even at 1000 trees, illustrating the well-known "benign overfitting" behaviour of boosting on clean data.

For `dset1`, the behaviour differs. Although the training error decreases rapidly as more trees are added, the test error fluctuates around values between 0.21 and 0.23. The lowest test error occurs at 100 trees (around 0.20), but beyond that point the curve becomes noticeably unstable, with the test error sometimes increasing and sometimes decreasing again. This instability suggests that the model begins to fit noise as it grows more complex.

The contrasting behaviour between the two datasets is most naturally explained by the *noise level* in the data. The results indicate that `dset0` is relatively clean: the labels are largely consistent, allowing boosting to add many trees without sharply degrading test performance. In contrast, `dset1` likely contains label noise or ambiguous decision boundaries. Since boosting is a highly flexible and powerful model, it eventually starts fitting the noise in the training data, which leads to unstable generalization and larger fluctuations in the test error.

In summary, the cleaner dataset (`dset0`) supports stable generalization even as model complexity grows, whereas the noisier dataset (`dset1`) shows less stable behaviour due to the model's tendency to fit noise when it is present.

The code can be found here.

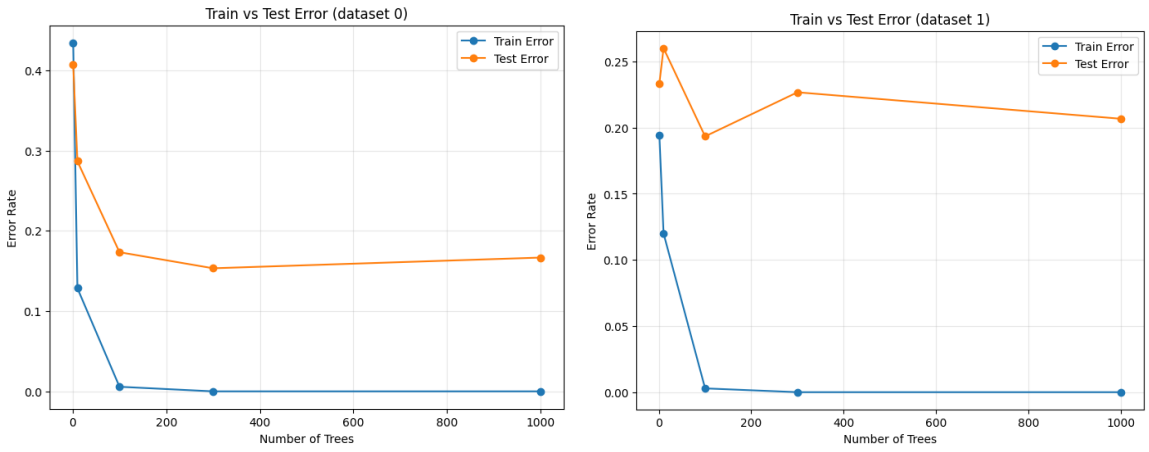

Figure 2: The generalization behaviour of the gradient boosting model with the increasing number of trees in the model fitting