STAT 587 - Winter 2026: HW 2

Due: Wednesday, Febrary 4th

Name: Esther Toobian
Instructor: Dr. Dorcas Ofori-Boateng

1. ISLP College Data (Predict `Apps`)

1(a) Least Squares Linear Regression

A least squares linear regression model was fit using all available predictors to estimate the number of applications received by each institution. The model was trained on 80% of the data, with performance evaluated on the held-out test set.

Predictor	Coefficient
Intercept	-58.9802
num__Accept	1.6656
num__Enroll	-1.0306
num__Top10perc	51.7066
num__Top25perc	-14.9924
num__F.Undergrad	0.0565
num__P.Undergrad	-0.0263
num__Outstate	-0.0712
num__Room.Board	0.1577
num__Books	0.1129
num__Personal	0.0388
num__PhD	-10.4225
num__Terminal	-1.4063
num__S.F.Ratio	4.9444
num__perc.alumni	-0.5175
num__Expend	0.0411
num__Grad.Rate	8.5633
cat__Private_Yes	-651.6070

Table 1: Estimated coefficients for least squares linear regression model predicting number of applications.

Estimated regression coefficients are reported in Table 1. Interpreting these as fitted associations rather than inferential statements, several predictors have relatively large coefficients or operate on large scales. For example, `Private` shows a large negative shift for private institutions relative to the baseline (public institutions) category. The coefficients for `Top10perc` and `Top25perc` have opposite signs; since these predictors are closely related, coefficient magnitudes and signs should be interpreted cautiously in the presence of multicollinearity. The coefficients for `Accept` and `Enroll` are also practically influential due to the large magnitude of these variables.

Metric	Value
Train MSE	990105.9662
Test MSE	1492443.3790

Table 2: Train and test mean squared error (MSE) for the least squares linear regression model.

Train and test mean squared errors are reported in Table 2; the test-set MSE was 1,492,443.38. As expected, the training MSE is lower than the test MSE, reflecting some degree of generalization error when the model is applied to unseen data. While the test error value is numerically large, it reflects the wide range and scale of application counts across institutions. On the original response scale, this corresponds to a root mean squared error (RMSE) of approximately 1,200 applications.

Given the magnitude of the test error and the potential for nonlinear relationships and interactions among predictors, these results motivate consideration of more flexible modeling approaches in subsequent parts.

1(b) Regression Tree

A regression tree was fit to the training data using all available predictors, without any pruning or cross-validation, to serve as a baseline tree model. The same 80% / 20% train–test split used in part (a) was retained.

Quantity	Value
Tree Depth	21
Number of Terminal Nodes (Leaves)	614

Table 3: Summary of the unpruned regression tree fit for predicting applications.

The fitted tree is very large, with depth 21 and 614 terminal nodes (Table 3). As a result, the full tree is too complex to display in a meaningful way, and no visualization of the complete tree is shown.

Metric	Value
Train MSE	0.0000
Test MSE	1851205.9103

Table 4: Train and test mean squared error (MSE) for the unpruned regression tree.

The unpruned tree achieves essentially zero training error (Train MSE ≈ 0), indicating that it fits the training data almost perfectly. However, it generalizes poorly, with a test-set mean squared error of 1,851,205.91 (Table 4). On the original scale of the response, this corresponds to a test RMSE of approximately 1,360 applications, providing clear evidence of substantial overfitting.

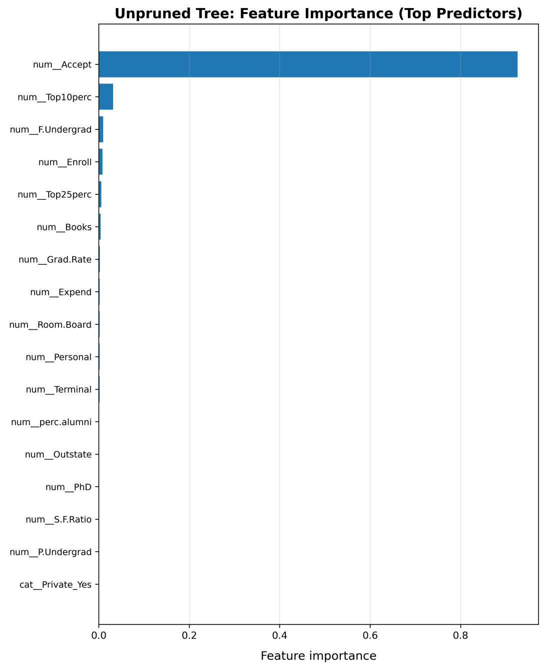

Figure 1: Feature importance for the unpruned regression tree.

Predictor	Importance
num__Accept	0.9260
num__Top10perc	0.0318
num__F.Undergrad	0.0096
num__Enroll	0.0081
num__Top25perc	0.0057
num__Books	0.0041
num__Grad.Rate	0.0027
num__Expend	0.0022
num__Room.Board	0.0021
num__Personal	0.0020
num__Terminal	0.0017
num__perc.alumni	0.0011
num__Outstate	0.0009
num__PhD	0.0008
num__S.F.Ratio	0.0007
num__P.Undergrad	0.0007
cat__Private_Yes	0.0000

Table 5: Top predictors by feature importance for the unpruned regression tree.

Figure 1 summarizes feature importance for the unpruned regression tree. Importance is overwhelmingly dominated by `Accept` (the number of accepted students), with all other predictors contributing relatively little. This concentration of importance, together with the large train–test error gap, further highlights the instability of the unpruned tree and motivates the use of cross-validation and pruning in part (c).

1(c) Cost-Complexity Pruning with Cross-Validation

In part (b), the unpruned regression tree achieved essentially zero training error but performed poorly on the test set, indicating substantial overfitting. In this section, cost–complexity pruning was used to control tree complexity, with the pruning level selected via cross-validation.

Pruning path and interpretation. Cost–complexity pruning produces a finite sequence of nested subtrees obtained by progressively increasing the complexity penalty parameter α. The pruning path determines which subtree sizes are structurally meaningful (i.e., where the optimal subtree changes as α increases), and cross-validation is then used to select among these candidate subtrees the one that best generalizes to unseen data.

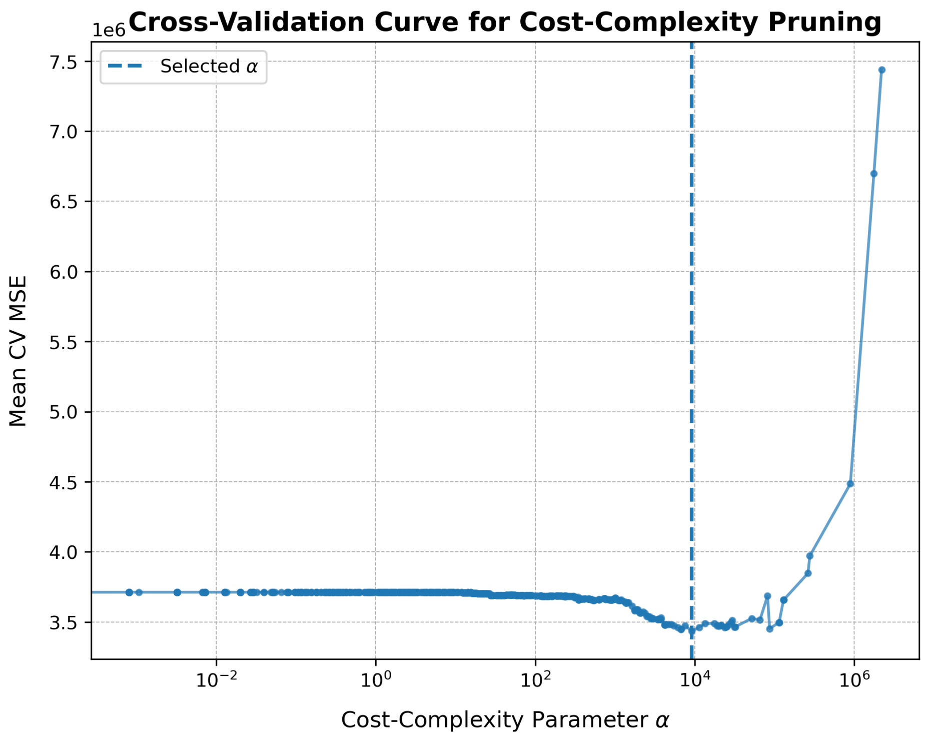

Figure 2: Cross-validation curve for cost–complexity pruning. The selected α is the value minimizing mean CV MSE.

Selecting α via cross-validation. Figure 2 shows the cross-validation curve of mean CV MSE versus the pruning parameter α (log scale). The curve exhibits the expected U-shaped behavior: very small α values correspond to overly complex trees that overfit, while very large α values correspond to overly pruned trees that underfit. The minimum CV error occurs near $\alpha \approx 9.2 \times 10^3$, which is the selected pruning level. At the selected value of α, pruning introduces regularization that increases training error while improving generalization. For the pruned tree, Table 6 reports a training MSE of 203,374.61 (Train RMSE $\approx$ 451) and a test MSE of 1,603,134.21 (Test RMSE $\approx$ 1,266).

Metric	Value
Train MSE	203374.6143
Test MSE	1603134.2147

Table 6: Train and test mean squared error (MSE) for the pruned regression tree.

Pruned model performance and comparison. The pruned tree selected by cross-validation is substantially smaller than the unpruned tree. Table 7 shows that pruning reduces the tree depth from 21 to 9 and the number of terminal nodes from 614 to 30, while also improving test performance.

Quantity	Unpruned Tree	Pruned Tree
Tree depth	21.0000	7.0000
Number of terminal nodes (leaves)	614.0000	30.0000
Train MSE	0.0000	203374.6143
Test MSE	1851205.9103	1603134.2147

Table 7: Comparison of unpruned and pruned regression trees for predicting applications.

The unpruned tree fits the training data nearly perfectly (Train MSE ≈ 0) but has a large test error, consistent with overfitting. After pruning, the training error increases while the test error decreases, consistent with the bias–variance tradeoff. Thus, pruning improves generalization relative to the unpruned tree, though the overall test error remains large on this response scale.

A simplified visualization of the selected pruned tree is shown in Figure 3. While still nontrivial, it is small enough to display and interpret (in contrast to the full unpruned tree from part (b)).

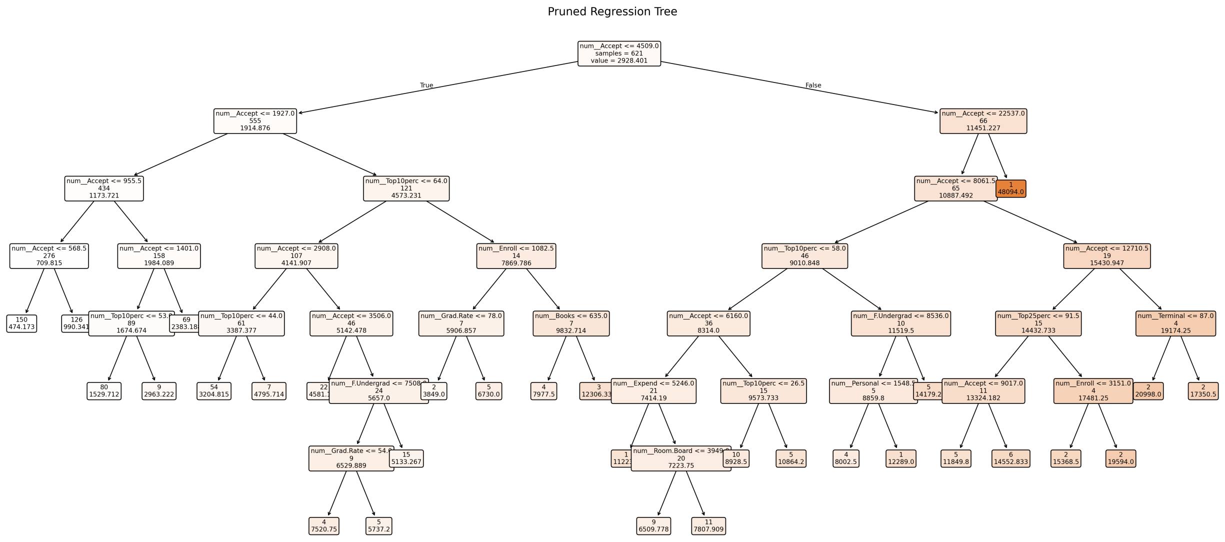

Figure 3: Selected pruned regression tree (minimal labels) using the cross-validated cost–complexity parameter.

Variable importance. Figure 4 and Table 8 summarize feature importance for the pruned tree. The importance distribution is highly concentrated: `Accept` dominates (importance ≈ 0.94), with substantially smaller contributions from `Top10perc`, `F.Undergrad`, and `Enroll`. This aligns with the structure of the plotted tree, where early splits are driven primarily by `Accept` (and a small set of additional enrollment/quality-related variables). Both the pruned and unpruned trees maintain the same 17 predictors, yet the order varies among those with the least importance. The distribution of importance is highly concentrated, with the majority of predictive signal concentrated in a small subset of variables. This concentration is reflected visually in Figure 4 and numerically in Table 8.

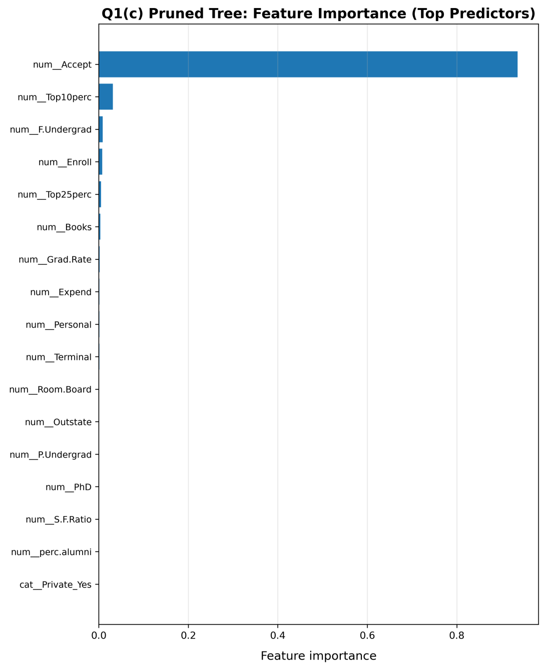

Figure 4: Feature importance for the pruned regression tree.

Predictor	Importance
num__Accept	0.9358
num__Top10perc	0.0316
num__F.Undergrad	0.0087
num__Enroll	0.0076
num__Top25perc	0.0054
num__Books	0.0034
num__Grad.Rate	0.0020
num__Expend	0.0016
num__Personal	0.0016
num__Terminal	0.0014
num__Room.Board	0.0009
num__Outstate	0.0000
num__P.Undergrad	0.0000
num__PhD	0.0000
num__S.F.Ratio	0.0000
num__perc.alumni	0.0000
cat__Private_Yes	0.0000

Table 8: Variable importances from the pruned regression tree. Predictors with zero importance were not used in any split.

In summary, cost–complexity pruning (selected by cross-validation) meaningfully reduces tree complexity and modestly improves test-set performance relative to the unpruned tree. The fitted pruned tree places most of its importance on `Accept` and a small set of related predictors.

1(d) Bagging Regression Trees

In this section, we apply bagging (bootstrap aggregating) to regression trees in order to reduce variance and improve predictive performance. Bagging fits a large number of trees on bootstrap resamples of the training data and averages their predictions. Unlike cost–complexity pruning, which restricts tree complexity directly, bagging retains fully grown trees and relies on averaging to stabilize predictions.

Bagging was implemented using regression trees grown on bootstrap samples, with all predictors available at each split. Two ensemble sizes were considered, $B = 500$ and $B = 1000$, to assess whether increasing the number of trees yields additional performance gains.

Table 9 reports the training and test mean squared error (MSE) for both ensemble sizes. Relative to the single-tree models in parts (b) and (c), bagging substantially improves test-set performance. The test MSE decreases to approximately 9.9×10^5, corresponding to test RMSE values of about 996 for $B = 500$ and 997 for $B = 1000$. Increasing the ensemble size from 500 to 1000 results in only a negligible change (slight increase) in both training and test error, indicating that the bagged estimator has largely stabilized by $B = 500$.

Metric	B=500	B=1000
Train MSE	324906.5196	331497.4660
Test MSE	991954.0863	994984.3157

Table 9: Train and test mean squared error (MSE) for bagging regression trees with $B = 500$ and $B = 1000$.

Unlike the unpruned regression tree, which nearly interpolated the training data and generalized poorly, the bagged models exhibit higher training error but markedly improved test performance. This behavior is consistent with the variance-reduction effect of bagging, where averaging across many high-variance trees leads to improved generalization.

Variable importance for bagging was computed using impurity-based importance measures aggregated across the ensemble. For each tree, importance is defined as the total reduction in squared error attributable to a predictor over all splits, and these values are averaged across the B trees and normalized to sum to one.

Figure 5 and Table 10 summarize feature importance for the bagged model with $B = 500$ (results for $B = 1000$ were qualitatively similar). All 17 predictors produced by the preprocessing step appear in the importance summary, though their contributions vary substantially in magnitude. The distribution of importance is highly concentrated: `Accept` dominates, accounting for approximately 92% of the total importance. The remaining variables have modest to very small importance values. Compared to the single-tree models, the ordering among lower-importance predictors varies more noticeably, reflecting the averaging effect across many trees.

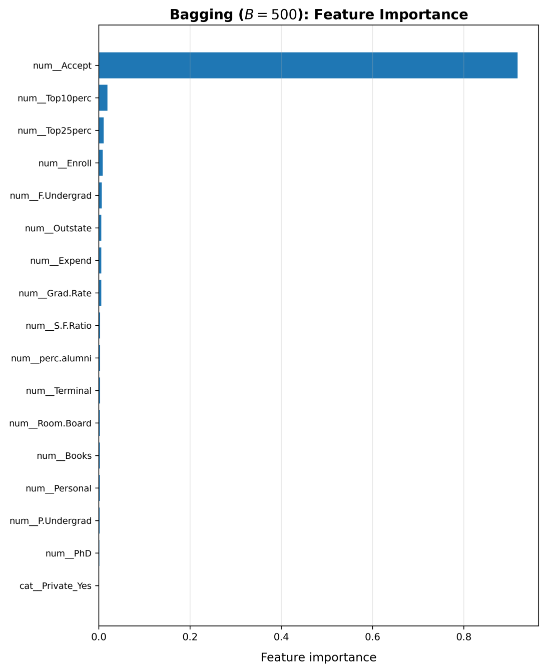

Figure 5: Feature importance for the bagging ensemble ($B = 500$).

Predictor	Importance
num__Accept	0.9178
num__Top10perc	0.0191
num__Top25perc	0.0107
num__Enroll	0.0086
num__F.Undergrad	0.0066
num__Outstate	0.0057
num__Expend	0.0055
num__Grad.Rate	0.0054
num__S.F.Ratio	0.0031
num__perc.alumni	0.0031
num__Terminal	0.0030
num__Room.Board	0.0027
num__Books	0.0026
num__Personal	0.0022
num__P.Undergrad	0.0021
num__PhD	0.0015
cat__Private_Yes	0.0002

Table 10: Top predictors by feature importance for bagging regression trees ($B = 500$).

Overall, bagging provides a substantial improvement in predictive performance over single regression trees by reducing variance through ensemble averaging. The results indicate that a moderate ensemble size is sufficient for stable performance, and that the predictive signal remains highly concentrated in `Accept`, with slight contributions from a small subset of variables.

1(e) Random Forest ($m = 3$)

A random forest regression model was fit using the same training data as in the previous parts, with the number of predictors considered at each split fixed to $m = 3$. Forests were trained with $B = 500$ and $B = 1000$ trees, and performance was evaluated using mean squared error (MSE) on both the training and test sets.

Table 11 summarizes the resulting errors. Increasing the number of trees from $B = 500$ to $B = 1000$ slightly improves test performance (Test MSE decreases from 1,077,786.05 to 1,039,993.61), which corresponds to a decrease in test RMSE from approximately 1038.16 to 1019.80 applications. Training error is nonzero in both cases (Train MSEs $\approx 3.8 \times 10^5,\ 3.9 \times 10^5$), reflecting the averaging/regularization effect induced by ensembling.

Metric	B=500	B=1000
Train MSE	382965.1815	391169.5215
Test MSE	1077786.0495	1039993.6076

Table 11: Train and test mean squared error (MSE) for random forests with $m = 3$ predictors considered at each split.

For variable importance, the standard impurity-based feature importance from the fitted random forest is reported. This represents the total reduction in squared-error impurity attributed to each predictor, averaged over all trees and normalized to sum to 1. Figure 6 and Table 12 show the ranked importances for the $B = 1000$ forest.

Importance is much less concentrated than in the single-tree models: while `Accept` remains the most influential predictor, additional variables such as `Enroll` and `F.Undergrad` also contribute substantially, and the remaining predictors have smaller but non-negligible importance. This broader distribution is consistent with random forests reducing reliance on a single dominant split variable by restricting the candidate predictors at each split ($m = 3$) and averaging across many decorrelated trees.

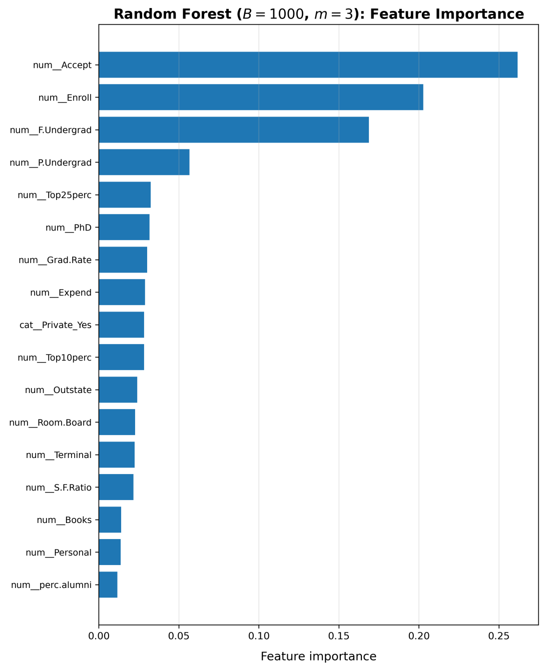

Figure 6: Impurity-based feature importance for the random forest ($B = 1000$, $m = 3$).

Predictor	Importance
num__Accept	0.2617
num__Enroll	0.2027
num__F.Undergrad	0.1687
num__P.Undergrad	0.0567
num__Top25perc	0.0325
num__PhD	0.0317
num__Grad.Rate	0.0302
num__Expend	0.0289
cat__Private_Yes	0.0283
num__Top10perc	0.0283
num__Outstate	0.0241
num__Room.Board	0.0226
num__Terminal	0.0224
num__S.F.Ratio	0.0217
num__Books	0.0140
num__Personal	0.0137
num__perc.alumni	0.0116

Table 12: Predictors ranked by impurity-based feature importance for the random forest ($B = 1000$, $m = 3$).

Relative to bagging (Table 9), the random forest exhibits slightly higher test error, with test RMSE increasing from approximately 996 under bagging to about 1020–1038 depending on B. Despite this, the random forest produces a more diffuse distribution of variable importance due to predictor subsampling at each split.

1(f) Comparisons and Recommendation

Table 13 compares train and test mean squared error (MSE) across all methods considered in parts (a)–(e), ordered by lowest test error. The unpruned regression tree achieves essentially zero training error (Train MSE ≈ 0) but has the worst test performance (Test MSE $\approx 1.85 \times 10^6$), which is consistent with severe overfitting. Cost–complexity pruning substantially reduces the tree size and increases training error, but in this run it does not improve predictive performance: the pruned tree has Test MSE $\approx 1.60 \times 10^6$, which is higher than both linear regression and bagging/random forests.

Method	Train MSE	Test MSE
Bagging (B=500)	324906.5196	991954.0863
Random forest (B=1000, m=3)	391169.5215	1039993.6076
Linear regression	990105.9662	1492443.3790
Pruned tree (CV)	203374.6143	1603134.2147
Unpruned tree	0.0000	1851205.9103

Table 13: Train and test mean squared error (MSE) for all methods.

Among the remaining methods, the ensemble approaches perform best on the test set. Bagging with $B = 500$ yields the lowest test error (Test MSE $\approx 9.92 \times 10^5$), with random forests ($B = 1000$, $m = 3$) slightly higher (Test MSE $\approx 1.04 \times 10^6$). Linear regression is substantially worse than these ensembles on the test set (Test MSE $\approx 1.49 \times 10^6$), suggesting that additive linear structure is not sufficient to capture the relevant nonlinearities and interactions that the tree-based ensembles can exploit.

Overall, based on the held-out test MSE, the recommended method is **Bagging with** $B = 500$, since it achieves the lowest test error among the methods evaluated here. Random forests are a close second, but with $m = 3$ they restrict the candidate split variables at each split. In this dataset (where predictive signal appears to be concentrated in a small subset of predictors), that restriction can slightly reduce performance relative to bagging. These conclusions are based on the fixed train–test split used throughout Question 1.

2. Business School Admissions

2(a) Exploratory Analysis (Training Data)

An exploratory analysis was conducted using the training data to assess the relationship between the predictors (GPA and GMAT) and the admission outcome. Figure 7 shows a scatter plot of GPA versus GMAT, colored by admission group. Applicants in the *Admit* group tend to have both higher GPAs and higher GMAT scores, while the *Not Admit* group clusters at lower values. The *Border* group lies between these two extremes and overlaps with both, indicating that perfect separation is unlikely.

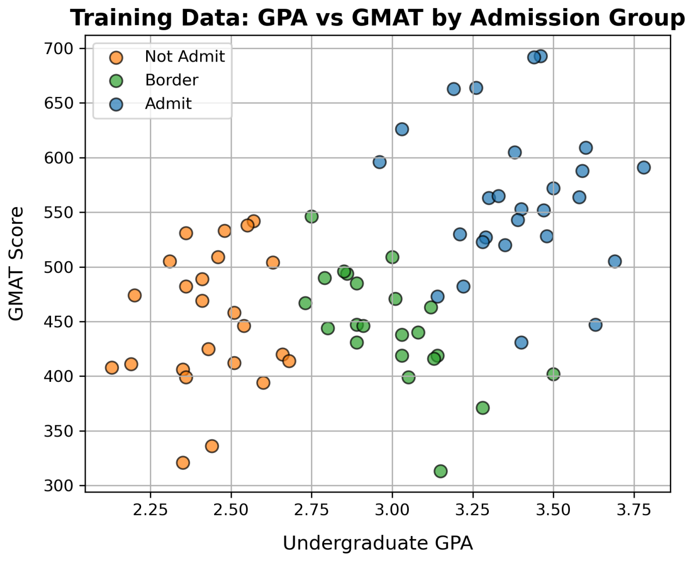

Figure 7: Training data: GPA vs. GMAT colored by admission group.

Marginal boxplots of GPA and GMAT by group are shown in Figure 8. GPA exhibits a clear increase from *Not Admit* to *Border* to *Admit*, suggesting it is a strong individual predictor. GMAT separates the *Admit* group well but shows substantial overlap between the *Not Admit* and *Border* groups. Overall, these plots suggest that GPA and GMAT together provide useful predictive information, with most classification ambiguity occurring for borderline applicants.

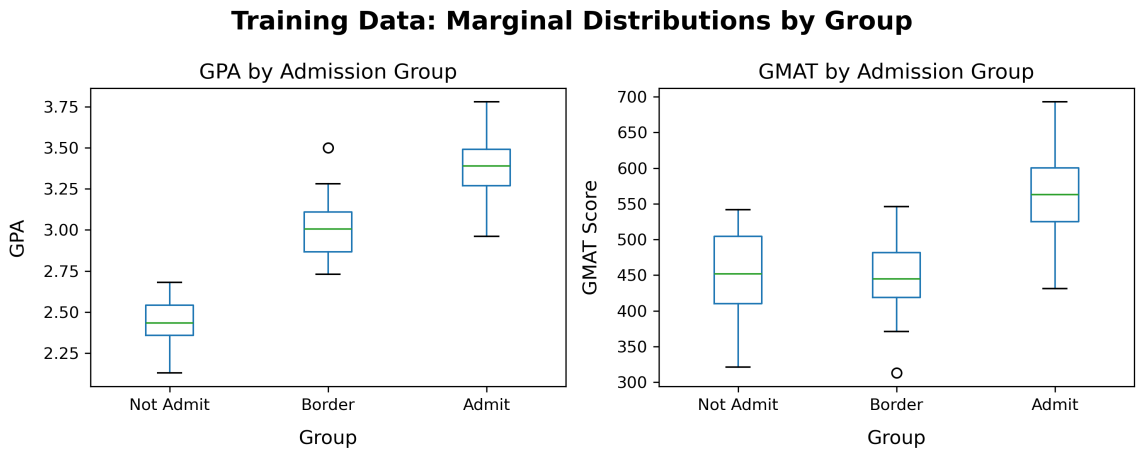

Figure 8: Training data: marginal distributions of GPA and GMAT by admission group.

2(b) Linear Discriminant Analysis

A linear discriminant analysis (LDA) model was fit on the training data using GPA and GMAT as predictors. Figure 9 shows the resulting LDA decision regions overlaid on the training observations.

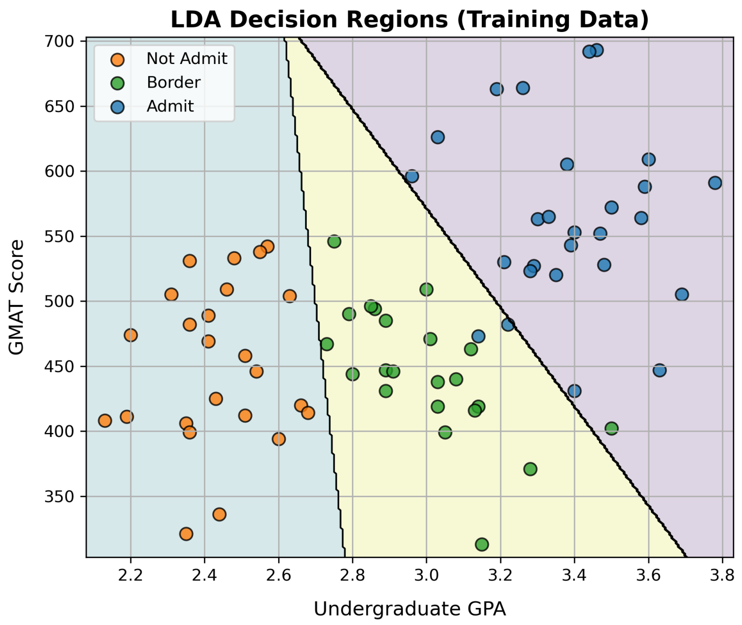

Figure 9: LDA decision regions on GPA/GMAT (training data).

The fitted decision regions are linear and broadly consistent with the exploratory patterns from part (a). The *Not Admit* region spans a wide range of GMAT values but is concentrated at lower GPA values, indicating that GPA drives much of the separation for non-admission in this LDA fit. The *Admit* region occupies the upper-right portion of the GPA–GMAT space. The *Border* region lies between these extremes. Overlap between the *Border* region and both neighboring groups suggests that borderline applicants will be the primary source of classification ambiguity.

	Not Admit	Border	Admit
Not Admit	24	0	0
Border	0	21	1
Admit	0	2	25

Table 14: LDA confusion matrix (training data). Rows are true classes and columns are predicted classes.

	Not Admit	Border	Admit
Not Admit	2	2	0
Border	0	4	0
Admit	0	1	3

Table 15: LDA confusion matrix (test data). Rows are true classes and columns are predicted classes.

The confusion matrices for the training and test sets are shown in Tables 14 and 15, with overall performance summarized in Table 18. On the training set, the misclassification rate is low (0.0411), with all errors occurring between the *Admit* and *Border* classes. On the test set, the misclassification rate increases to 0.25, with most errors again involving predictions of *Border*. Notably, there is no direct confusion between *Admit* and *Not Admit* in either split.

Class	Sensitivity (Train, OvR)	Specificity (Train, OvR)
Not Admit	1.0000	1.0000
Border	0.9545	0.9608
Admit	0.9259	0.9783

Table 16: LDA per-class sensitivity and specificity (OvR) on the training set.

Class	Sensitivity (Test, OvR)	Specificity (Test, OvR)
Not Admit	0.5000	1.0000
Border	1.0000	0.6250
Admit	0.7500	1.0000

Table 17: LDA per-class sensitivity and specificity (OvR) on the test set.

Per-class one-vs-rest sensitivity and specificity are reported in Tables 16 and 17. On the test set, the *Border* class has perfect sensitivity (all borderline applicants are correctly identified), but reduced specificity because several *Not Admit* and *Admit* observations are classified as *Border*. The high multiclass AUC values (OvR macro AUC = 0.9966 on training and 0.9792 on test) shown in Table 18 indicate strong ranking performance overall, even though some hard class assignments remain ambiguous near the borderline region.

Metric	Train	Test
Misclassification rate	0.0411	0.2500
Macro sensitivity	0.9602	0.7500
Macro specificity	0.9797	0.8750
AUC (OvR macro)	0.9965	0.9792

Table 18: LDA overall performance metrics on training and test sets. AUC is computed using multiclass OvR macro averaging.

Overall, the results suggest that LDA separates the extreme classes well, but borderline cases are the primary source of ambiguity, and generalization performance on the small test set is noticeably weaker than training performance.

2(c) Quadratic Discriminant Analysis

A quadratic discriminant analysis (QDA) model was fit on the training data using GPA and GMAT as predictors. Figure 10 shows the resulting QDA decision regions overlaid on the training observations.

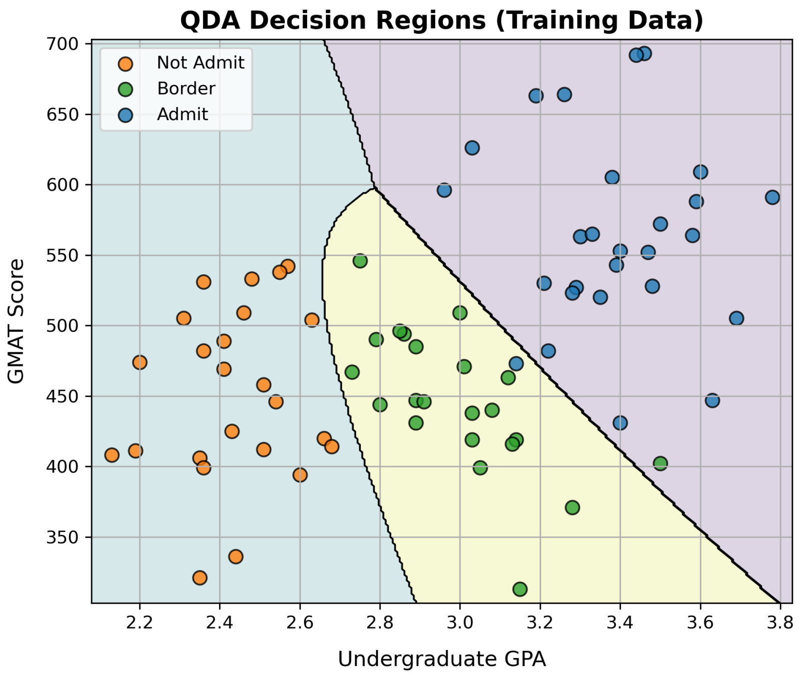

Figure 10: QDA decision regions on GPA/GMAT (training data).

Unlike LDA, the QDA decision boundaries are nonlinear, providing greater flexibility in separating the admission groups. The *Admit* region remains concentrated in the upper-right portion of the GPA–GMAT space, while the *Not Admit* region is primarily associated with lower GPA values. The *Border* region occupies an intermediate area, and the curved boundaries follow the shape of the training data more closely than the linear LDA boundaries. A small amount of overlap between the *Border* and *Admit* groups is still present, while the *Not Admit* group remains well separated in the training data.

	Not Admit	Border	Admit
Not Admit	24	0	0
Border	0	21	1
Admit	0	1	26

Table 19: QDA confusion matrix (training data). Rows are true classes and columns are predicted classes.

	Not Admit	Border	Admit
Not Admit	2	2	0
Border	0	4	0
Admit	0	0	4

Table 20: QDA confusion matrix (test data). Rows are true classes and columns are predicted classes.

The confusion matrices for the training and test sets are shown in Tables 19 and 20, with overall performance summarized in Table 23. On the training set, the misclassification rate is low (0.0274), with only two errors: one *Border* observation classified as *Admit* and one *Admit* observation classified as *Border*. On the test set, the misclassification rate is 0.1667, with the only errors corresponding to true *Not Admit* observations classified as *Border*. As in the LDA results, there is no direct confusion between the *Admit* and *Not Admit* classes.

Class	Sensitivity (Train, OvR)	Specificity (Train, OvR)
Not Admit	1.0000	1.0000
Border	0.9545	0.9804
Admit	0.9630	0.9783

Table 21: QDA per-class sensitivity and specificity (OvR) on the training set.

Class	Sensitivity (Test, OvR)	Specificity (Test, OvR)
Not Admit	0.5000	1.0000
Border	1.0000	0.7500
Admit	1.0000	1.0000

Table 22: QDA per-class sensitivity and specificity (OvR) on the test set.

Per-class one-vs-rest sensitivity and specificity are reported in Tables 21 and 22. On the test set, the *Border* class has perfect sensitivity but reduced specificity, reflecting that borderline predictions absorb some observations from the neighboring *Not Admit* class. The multiclass AUC values (OvR macro AUC = 0.9988 on training and 0.9583 on test) shown in Table 23 indicate strong ranking performance overall, despite some ambiguity in hard class assignments.

Metric	Train	Test
Misclassification rate	0.0274	0.1667
Macro sensitivity	0.9725	0.8333
Macro specificity	0.9862	0.9167
AUC (OvR macro)	0.9988	0.9583

Table 23: QDA overall performance metrics on training and test sets. AUC is computed using multiclass OvR macro averaging.

Overall, QDA fits the training data closely and maintains strong performance on the small test set, with misclassifications confined to adjacent admission categories. Relative to LDA, QDA reduces the test misclassification rate (0.1667 vs. 0.25), suggesting that the additional flexibility of nonlinear boundaries is beneficial for this dataset.

2(d) K-Nearest Neighbors

A K-nearest neighbors (KNN) classifier was fit using GPA and GMAT as predictors. Because KNN is distance-based and GPA and GMAT are measured on different scales, both predictors were standardized (centralized and scaled) using training-set means and standard deviations prior to model fitting. This prevents Euclidean distances from being dominated by GMAT and ensures that both predictors contribute comparably to neighbor selection.

Selection of K via Cross-Validation. The tuning parameter K was selected using 5-fold cross-validation on the training data. Candidate odd values $K \in \{1, 3, 5, \dots, 25\}$ were considered to reduce the frequency of ties. For each value of K, the mean cross-validated misclassification rate was computed and used as the model selection criterion.

Figure 11 shows the resulting cross-validated error curve. The minimum occurs at $K^* = 5$, which was therefore selected as the optimal value. The corresponding numerical cross-validation results are reported in Table 24.

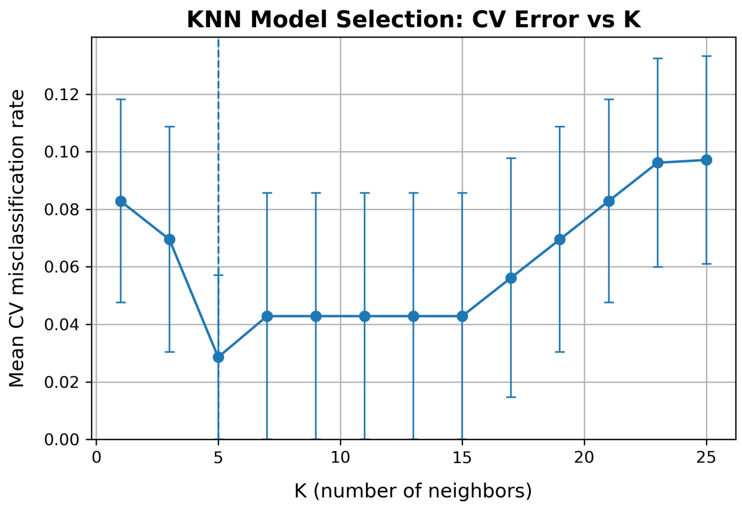

Figure 11: Five-fold cross-validated misclassification rate as a function of K for KNN. The dashed line indicates the selected value $K^* = 5$.

K	CV accuracy (mean)	CV error (mean)	CV accuracy (sd)
5	0.9714	0.0286	0.0639
7	0.9571	0.0429	0.0958
9	0.9571	0.0429	0.0958
11	0.9571	0.0429	0.0958
13	0.9571	0.0429	0.0958
15	0.9571	0.0429	0.0958
17	0.9438	0.0562	0.0930
3	0.9305	0.0695	0.0875
19	0.9305	0.0695	0.0875
1	0.9171	0.0829	0.0789
21	0.9171	0.0829	0.0789
23	0.9038	0.0962	0.0811
25	0.9029	0.0971	0.0807

Table 24: KNN model selection using training-only 5-fold CV. Reported values are mean CV accuracy/error across folds (and the SD of accuracy).

Decision Regions. Using the selected value $K^* = 5$, the KNN model was refit on the full training set. Figure 12 displays the resulting decision regions overlaid on the training observations. In contrast to the linear (LDA) and quadratic (QDA) boundaries, the KNN decision boundary is highly irregular, reflecting the local and nonparametric nature of the method. As in previous models, most ambiguity is concentrated in the *Border* region, while the *Admit* and *Not Admit* groups remain well separated.

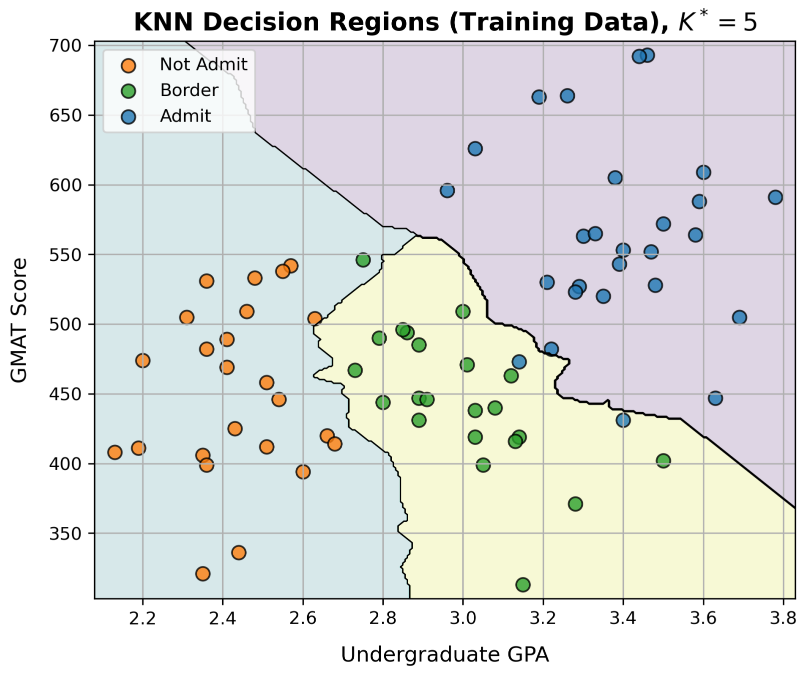

Figure 12: KNN decision regions on GPA/GMAT using $K^* = 5$ (training data).

Classification Performance. Three-class confusion matrices for the training and test sets are shown in Tables 25 and 26. On the training set, the KNN classifier at $K^* = 5$ achieves a low misclassification rate, with nearly all errors occurring between the *Border* group and one of the two extreme admission categories. This behavior is consistent with both the exploratory analysis and the decision-region plot, which indicate overlap involving borderline applicants.

On the test set, performance decreases modestly, as expected given the small test sample size. Misclassifications again primarily involve the *Border* group, while there is no direct confusion between the *Admit* and *Not Admit* classes. The cross-validated misclassification rate used to select K^* is reported in the K-sweep table (Table 24) and the CV curve (Figure 11).

	Not Admit	Border	Admit
Not Admit	24	0	0
Border	1	21	0
Admit	0	2	25

Table 25: KNN confusion matrix at K^* (training data). Rows are true classes and columns are predicted classes.

	Not Admit	Border	Admit
Not Admit	2	2	0
Border	0	4	0
Admit	0	1	3

Table 26: KNN confusion matrix at K^* (test data). Rows are true classes and columns are predicted classes.

Multiclass OvR Metrics. The performance measures sensitivity, specificity, and AUC are natively defined for binary classification, whereas the admission outcome consists of three classes. Therefore, these metrics were computed using a one-vs-rest (OvR) strategy. For each class, that class was treated as the positive class and the remaining two classes were treated as negative, yielding class-specific OvR confusion counts. Per-class sensitivity and specificity were computed from these counts for both the training and test sets and then macro-averaged across classes. Macro averaging assigns equal weight to each class, ensuring that performance on the smaller *Border* group is not overshadowed by the more prevalent classes.

Multiclass AUC was computed using OvR AUC with macro averaging, based on predicted class probabilities. This approach preserves the full multiclass structure of the problem and avoids collapsing the *Border* group into an artificial binary outcome. Per-class OvR metrics are reported in Tables 27 and 28, with macro-averaged results summarized in Table 29.

Class	Sensitivity (Train, OvR)	Specificity (Train, OvR)
Not Admit	1.0000	0.9796
Border	0.9545	0.9608
Admit	0.9259	1.0000

Table 27: KNN per-class sensitivity and specificity (OvR) on the training set at K^*.

Class	Sensitivity (Test, OvR)	Specificity (Test, OvR)
Not Admit	0.5000	1.0000
Border	1.0000	0.6250
Admit	0.7500	1.0000

Table 28: KNN per-class sensitivity and specificity (OvR) on the test set at K^*.

Metric	Train	Test
Misclassification rate	0.0411	0.2500
Macro sensitivity	0.9602	0.7500
Macro specificity	0.9801	0.8750
AUC (OvR macro)	0.9968	1.0000

Table 29: KNN overall performance metrics on training and test sets at K^*. AUC is computed using multiclass OvR macro averaging.

These results indicate strong overall ranking performance, as reflected by high macro-averaged AUC values on both the training and test sets. Sensitivity is highest for the *Admit* and *Not Admit* groups, while reduced specificity for the *Border* class reflects the same ambiguity observed in the confusion matrices and decision-region plot.

Overall, KNN with $K^* = 5$ provides strong predictive performance after appropriate standardization. As with LDA and QDA, remaining errors are largely confined to the *Border* group, with no direct confusion between the *Admit* and *Not Admit* classes.

2(e) Model Comparison and Recommendation

Table 30 compares the test-set performance of LDA, QDA, and KNN using overall misclassification rate, macro-averaged sensitivity, macro-averaged specificity, and macro-averaged one-vs-rest AUC.

Method	Test Error	Sensitivity	Specificity	AUC
LDA	0.2500	0.7500	0.8750	0.9792
QDA	0.1667	0.8333	0.9167	0.9583
KNN	0.2500	0.7500	0.8750	1.0000

Table 30: Comparison of LDA, QDA, and KNN classifiers based on test-set performance. Reported values include misclassification rate and macro-averaged one-vs-rest sensitivity, specificity, and AUC.

Linear Discriminant Analysis (LDA). LDA provides the most interpretable model, yielding linear decision boundaries that are easy to visualize and explain. While LDA separates the extreme admission categories well, its test misclassification rate is relatively high and both sensitivity and specificity are lower than those of QDA. Errors are concentrated among borderline applicants, and the reduction in performance from training to test data suggests limited flexibility in capturing more complex class structure.

Quadratic Discriminant Analysis (QDA). QDA introduces nonlinear decision boundaries, allowing for greater flexibility than LDA. Among the three methods, QDA achieves the *lowest test misclassification rate*, indicating the best overall classification accuracy on unseen data. It also exhibits strong sensitivity and specificity across classes, suggesting balanced performance. However, the increased flexibility of QDA reduces interpretability and raises mild concerns about overfitting, particularly given the small size of the training and test sets.

K-Nearest Neighbors (KNN). After appropriate standardization and selection of K via cross-validation, KNN demonstrates strong ranking performance, achieving the highest test-set AUC. This indicates strong separation between classes in terms of predicted probabilities. However, its test misclassification rate is higher than that of QDA and comparable to LDA, reflecting persistent ambiguity near class boundaries. Additionally, KNN is the least interpretable of the three methods and is sensitive to local data structure.

Conclusion. Considering all metrics jointly, QDA provides the best balance of predictive accuracy and class-level performance on this dataset, achieving the lowest test error while maintaining strong sensitivity, specificity, and AUC. Therefore, QDA is recommended as the preferred classifier for this problem. That said, the test set is very small, and differences in performance should be interpreted cautiously. In practice, additional validation data would be necessary before making strong conclusions about generalization performance.

CODE

All code for this assignment is available at: GitHub Repository: STAT587_DS1_HW2