Project 2 Part C

● Graded

Group

Autumn Xu
Yueping Gu
Samantha Zheng
View or edit group

Total Points

25 / 25 pts

Question 1

Dimension reduction method(s) **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect or missing

Question 2

Classification problem 1 **10** / 10 pts

2.1 **Input data** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts No dataset uploaded

2.2 **Visualizations of cross validation** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts missing or incorrect

2.3 **Pick your best hyperparameter values** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts missing or incorrect

2.4 **Validation results** **2** / 2 pts

✓ **+ 2 pts** Correct

+ 1 pt missing performance or runtime or models

+ 0 pts missing or incorrect

2.5 **Repeat 2.1-2.4** **5** / 5 pts

✓ **+ 5 pts** Correct

+ 3 pts Partially correct

+ 0 pts Missing or Incorrect

Question 3

Classification problem 2 **10** / 10 pts

3.1 **Without dimension reduction** **5** / 5 pts

✓ **+ 5 pts** Correct

+ 3 pts Partial

+ 0 pts Missing/Incorrect

3.2 **With dimension reduction** **5** / 5 pts

✓ **+ 5 pts** Correct

+ 3 pts Partial

+ 0 pts Missing/Incorrect

Question 4

Review **4** / 4 pts

✓ **+ 4 pts** Credit

+ 2 pts Partial credit

+ 0 pts Missing/Incorrect

Q1 Dimension reduction method(s)

1 Point

How many features do you currently have? How will you cut that number in half? Describe/define the dimension reduction method(s) and explain why you chose them:

Before any processing, we read in the dataset as 22 features. After applying the dummy encoding to transform the categorical features into dummy/indicator variables (excluding the target variable poisonous), we end up with a high-dimensional numeric feature space. To manage the curse of dimensionality and reduce computational complexity, we used principal component analysis. Principal component analysis is a linear dimensionality reduction technique that transforms the original correlated features into a smaller number (11 features) of uncorrelated variables, principal components, which capture the directions of much variance in the data. In total, each of our training/testing datasets has 11 transformed features and 1 target feature (either odor or poisonous). We chose principal component analysis due to its unsupervised process: it doesn't rely on the target variable and is suitable when many of the original features are sparse, as is common after one-hot encoding. Principal component analysis captures the majority of the variance with a smaller number of components, which is helpful in preserving most of the information in a more compact form. Principal component analysis improves computational efficiency for modeling, and help void overfitting by eliminating noise and redundancy from high-dimensional data.

Q2 Classification problem 1

10 Points

Q2.1 Input data
1 Point

Split your dataset for the first classification problem into a training set and a validation set. Upload them here:

y1_train_dropped.csv Download

```
class
p
e
p
e
e
p
e
e
p
e
e
p
e
p
e
p
e
p
e
e
p
e
e
e
e
e
e
p
e
e
p
e
e
e
e
e
e
p
e
e
e
e
e
p
e
p
p
e
```

```
e
p
e
e
p
p
e
e
e
e
e
e
e
e
p
p
e
p
e
e
e
e
p
p
e
e
e
e
e
p
e
e
e
e
p
e
e
e
p
e
e
p
p
p
e
p
p
e
p
e
p
e
```

```
e
e
e
e
e
e
p
e
e
e
p
e
e
p
p
e
e
e
p
e
p
e
e
p
e
e
e
e
e
e
p
e
e
e
p
p
e
e
e
e
p
p
e
e
e
e
e
e
p

```

X1_train_dropped.csv Download

```
1  Large file hidden. You can download it using the button above.
```

X1_test_dropped.csv Download

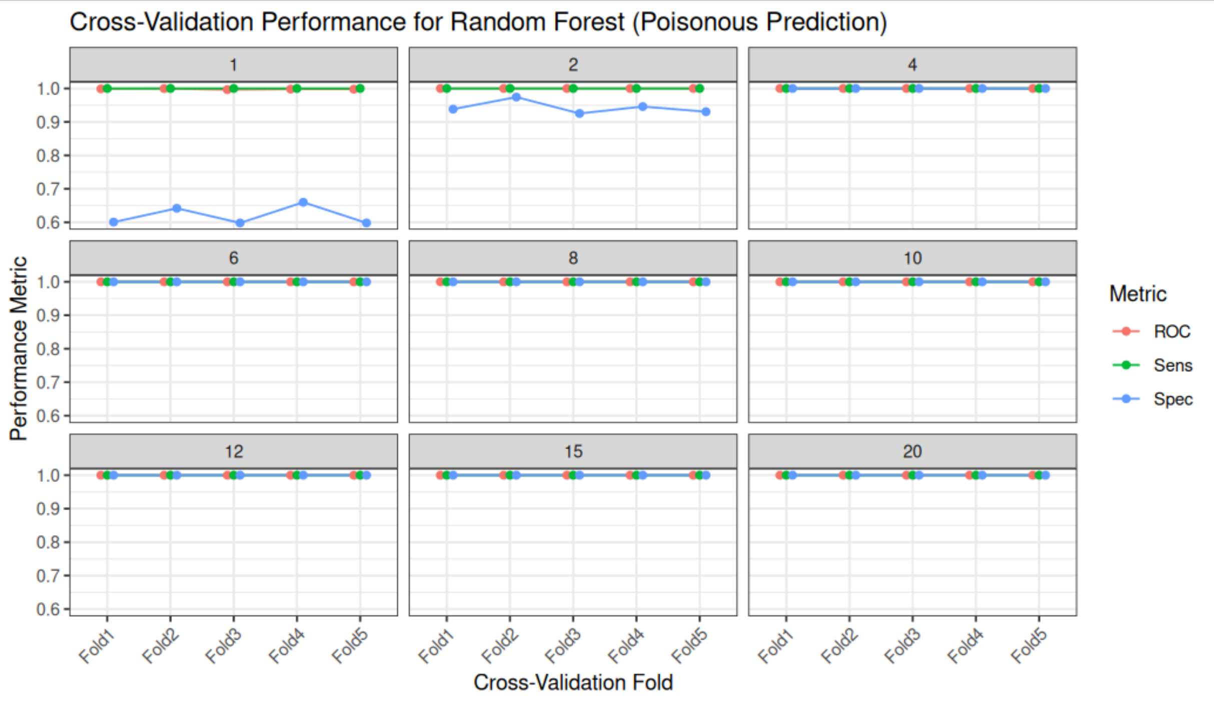
```
cap-shape,cap-surface,cap-color,bruises,odor,gill-attachment,gill-spacing,gill-size,gill-color,stalk-shape,stalk-root,stalk-surface-above-ring,stalk-surface-below-ring,stalk-color-above-ring,stalk-color-below-ring,veil-type,veil-color,ring-number,ring-type,spore-print-color,population,habitat
f,f,n,t,n,f,c,b,p,t,b,s,s,w,g,p,w,o,p,k,v,d
f,f,g,f,f,f,c,b,g,e,b,k,k,n,n,p,w,o,l,h,y,p
f,f,g,f,f,f,c,b,g,e,b,k,k,p,p,p,w,o,l,h,y,p
f,y,g,t,n,f,c,b,p,t,b,s,s,w,p,p,w,o,p,k,v,d
x,y,e,t,n,f,c,b,n,t,b,s,s,g,p,p,w,o,p,k,v,d
x,s,g,t,f,f,c,b,p,t,b,s,s,w,w,p,w,o,p,h,s,u
f,f,y,f,f,f,c,b,h,e,b,k,k,p,b,p,w,o,l,h,v,d
x,f,g,f,n,f,w,b,n,t,e,f,s,w,w,p,w,o,e,k,s,g
x,f,e,t,n,f,c,b,w,t,b,s,s,w,w,p,w,o,p,n,v,d
f,f,g,f,f,f,c,b,p,e,b,k,k,b,n,p,w,o,l,h,v,d
f,y,g,f,f,f,c,b,g,e,b,k,k,b,n,p,w,o,l,h,y,d
x,y,n,t,n,f,c,b,u,t,b,s,s,g,w,p,w,o,p,n,v,d
f,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,v,g
f,y,y,f,f,f,c,b,p,e,b,k,k,b,p,p,w,o,l,h,y,g
x,y,g,t,n,f,c,b,p,t,b,s,s,w,w,p,w,o,p,k,y,d
f,y,y,f,f,f,c,b,p,e,b,k,k,n,b,p,w,o,l,h,y,d
f,y,g,f,f,f,c,b,p,e,b,k,k,b,n,p,w,o,l,h,v,g
f,y,b,t,n,f,c,b,g,e,b,s,s,w,w,p,w,t,p,r,v,g
x,y,n,t,n,f,c,b,n,t,b,s,s,p,g,p,w,o,p,k,v,d
x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,v,u
x,y,y,f,f,f,c,b,p,e,b,k,k,n,n,p,w,o,l,h,y,d
x,s,b,t,f,f,c,b,p,t,b,s,f,w,w,p,w,o,p,h,s,g
x,y,y,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,s,g
f,f,w,f,n,f,w,b,n,t,e,s,s,w,w,p,w,o,e,k,s,g
x,f,y,f,f,f,c,b,p,e,b,k,k,p,n,p,w,o,l,h,v,g
f,s,w,t,f,f,c,b,w,t,b,s,s,w,w,p,w,o,p,h,s,g
f,f,g,f,n,f,w,b,k,t,e,f,f,w,w,p,w,o,e,k,a,g
x,f,y,f,f,f,c,b,p,e,b,k,k,p,b,p,w,o,l,h,y,p
x,s,w,f,n,f,w,b,k,t,e,f,f,w,w,p,w,o,e,n,a,g
x,s,w,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,m
f,s,n,f,n,f,w,b,p,t,e,s,s,w,w,p,w,o,e,k,s,g
x,y,g,t,n,f,c,b,p,t,b,s,s,g,g,p,w,o,p,n,v,d
x,f,g,f,f,f,c,b,p,e,b,k,k,b,b,p,w,o,l,h,y,d
x,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,g
x,s,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,g
f,s,g,t,f,f,c,b,w,t,b,f,f,w,w,p,w,o,p,h,s,g
f,y,n,t,n,f,c,b,w,t,b,s,s,w,p,p,w,o,p,k,y,d
x,y,n,t,n,f,c,b,n,t,b,s,s,w,w,p,w,o,p,k,y,d
f,y,e,t,n,f,c,b,n,t,b,s,s,w,g,p,w,o,p,n,y,d
x,y,g,t,n,f,c,b,n,t,b,s,s,w,p,p,w,o,p,k,y,d
x,f,g,f,f,f,c,b,p,e,b,k,k,b,p,p,w,o,l,h,v,g
x,s,n,f,n,f,w,b,n,t,e,s,s,w,w,p,w,o,e,k,s,g
x,f,n,f,n,f,w,b,h,t,e,s,f,w,w,p,w,o,e,k,a,g
f,f,g,f,n,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,y,u
f,s,n,f,n,f,w,b,k,t,e,s,f,w,w,p,w,o,e,k,a,g
```

```
x,y,n,t,n,f,c,b,u,t,b,s,s,p,g,p,w,o,p,n,y,d
f,y,y,f,f,f,c,b,p,e,b,k,k,b,b,p,w,o,l,h,y,d
f,y,g,f,f,f,c,b,g,e,b,k,k,p,p,p,w,o,l,h,y,d
f,f,e,t,n,f,c,b,p,t,b,s,s,p,g,p,w,o,p,n,y,d
f,f,g,t,n,f,c,b,p,t,b,s,s,w,p,p,w,o,p,k,v,d
x,y,n,t,n,f,c,b,w,t,b,s,s,g,w,p,w,o,p,n,y,d
f,y,g,t,n,f,c,b,w,t,b,s,s,g,p,p,w,o,p,k,y,d
x,y,y,t,a,f,c,b,n,e,r,s,y,w,w,p,w,o,p,k,s,p
x,y,e,t,n,f,c,b,u,t,b,s,s,w,w,p,w,o,p,n,y,d
x,s,p,f,c,f,c,n,p,e,b,s,s,w,w,p,w,o,p,n,s,d
f,f,n,t,n,f,c,b,n,t,b,s,s,g,g,p,w,o,p,k,y,d
f,y,e,t,n,f,c,b,n,t,b,s,s,p,g,p,w,o,p,k,y,d
f,f,n,t,n,f,c,b,u,t,b,s,s,w,p,p,w,o,p,k,v,d
x,s,w,f,c,f,c,n,u,e,b,s,s,w,w,p,w,o,p,n,s,d
s,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,y,u
x,y,n,t,n,f,c,b,p,t,b,s,s,g,w,p,w,o,p,k,v,d
f,y,y,f,f,f,c,b,p,e,b,k,k,p,n,p,w,o,l,h,y,d
x,s,w,t,f,f,c,b,p,t,b,s,f,w,w,p,w,o,p,h,v,u
f,f,e,t,n,f,c,b,w,t,b,s,s,w,w,p,w,o,p,k,y,d
x,f,w,f,n,f,w,b,n,t,e,s,f,w,w,p,w,o,e,k,a,g
f,y,e,t,n,f,c,b,u,t,b,s,s,p,p,p,w,o,p,n,v,d
x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,g
f,y,n,t,n,f,c,b,w,t,b,s,s,g,w,p,w,o,p,n,y,d
f,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,s,g
x,y,e,t,n,f,c,b,u,t,b,s,s,g,g,p,w,o,p,k,v,d
f,y,e,t,n,f,c,b,p,t,b,s,s,p,w,p,w,o,p,k,v,d
f,f,y,f,f,f,c,b,g,e,b,k,k,n,b,p,w,o,l,h,y,d
b,y,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m
x,f,w,f,n,f,w,b,h,t,e,f,s,w,w,p,w,o,e,k,a,g
x,y,y,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,n,s,g
x,f,n,t,n,f,c,b,n,t,b,s,s,p,w,p,w,o,p,k,y,d
x,f,e,t,n,f,c,b,p,t,b,s,s,g,w,p,w,o,p,k,y,d
s,f,g,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p,n,y,u
x,s,p,f,c,f,c,n,u,e,b,s,s,w,w,p,w,o,p,k,s,d
f,f,e,t,n,f,c,b,p,t,b,s,s,p,w,p,w,o,p,k,v,d
f,f,e,t,n,f,c,b,p,t,b,s,s,p,p,p,w,o,p,n,y,d
f,s,w,f,n,f,w,b,n,t,e,s,s,w,w,p,w,o,e,n,a,g
x,s,w,f,c,f,c,n,u,e,b,s,s,w,w,p,w,o,p,k,s,d
x,f,w,f,c,f,c,n,p,e,b,s,s,w,w,p,w,o,p,n,v,d
f,f,g,t,n,f,c,b,p,t,b,s,s,w,w,p,w,o,p,k,y,d
b,y,w,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,g
x,y,e,t,n,f,c,b,p,t,b,s,s,g,p,p,w,o,p,n,y,d
x,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,u
f,f,g,f,f,f,c,b,h,e,b,k,k,p,n,p,w,o,l,h,y,g
f,y,g,f,f,f,c,b,p,e,b,k,k,p,b,p,w,o,l,h,v,g
x,y,c,f,n,f,w,n,w,e,b,s,f,w,n,p,w,o,e,w,v,l
f,f,e,t,n,f,c,b,w,t,b,s,s,w,w,p,w,o,p,n,y,d
x,y,e,t,n,f,c,b,p,t,b,s,s,p,g,p,w,o,p,n,v,d
x,y,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,m
f,f,w,f,n,f,w,b,n,t,e,f,s,w,w,p,w,o,e,n,a,g
f,y,n,t,n,f,c,b,p,t,b,s,s,p,w,p,w,o,p,n,y,d
f,y,y,f,f,f,c,b,p,e,b,k,k,n,p,p,w,o,l,h,v,g
```

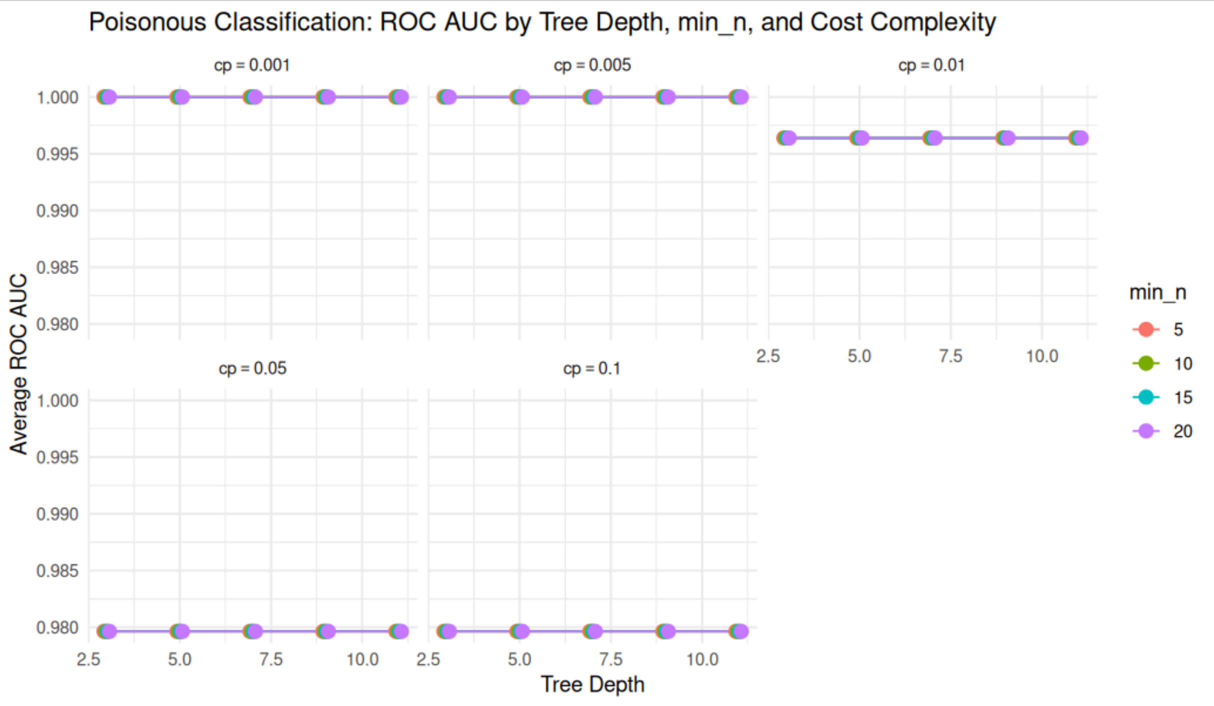
Q2.2 Visualizations of cross validation
1 Point

Upload visualizations for each model that shows the k-fold cross-validation results from testing different hyperparameter models

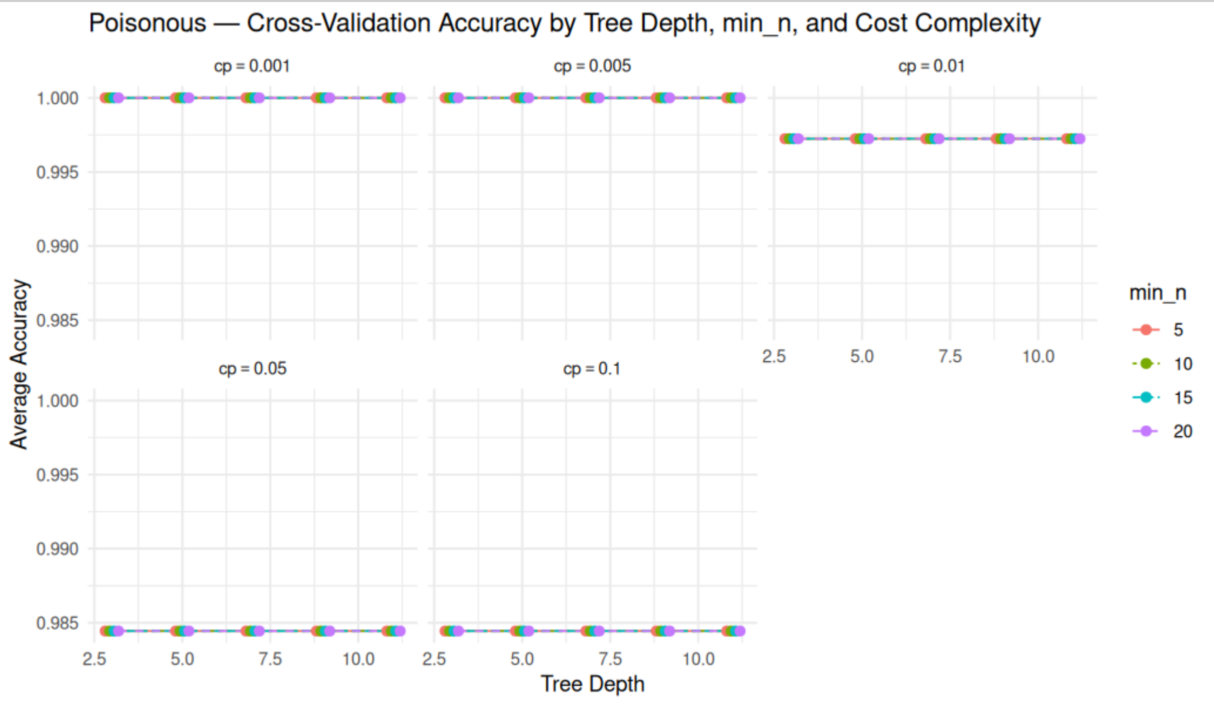
random_forest_poisonous_q22.png Download



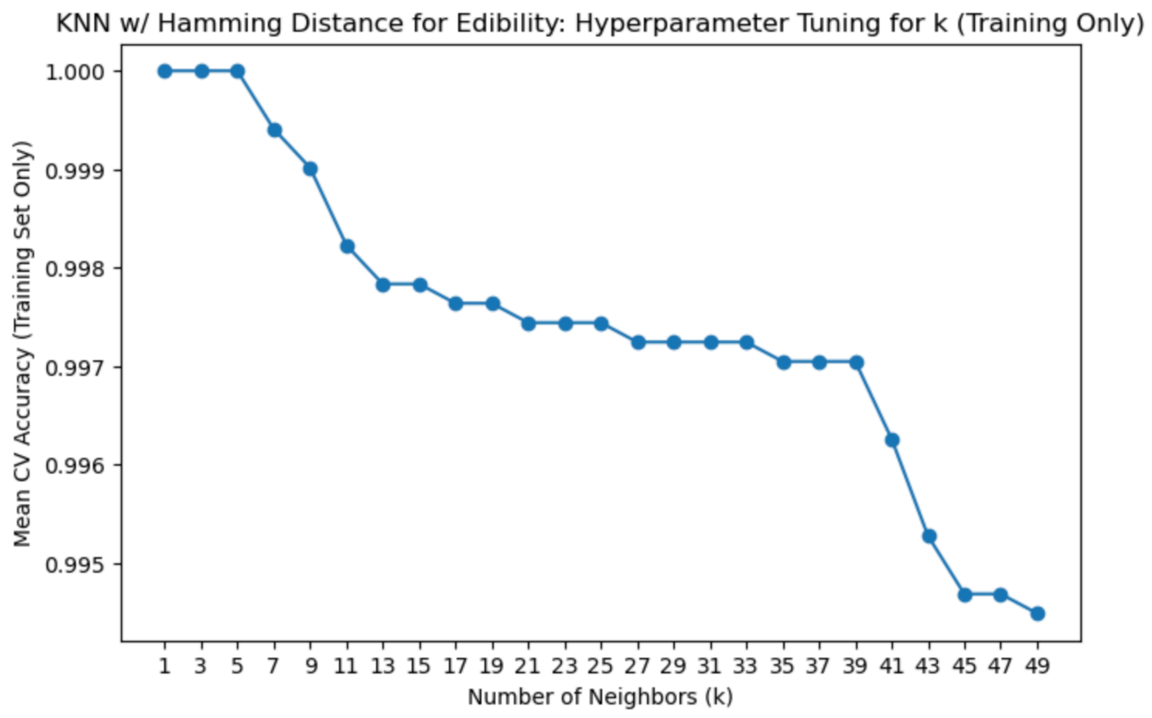
decision_tree_poisonous_roc_auc_q22.png Download



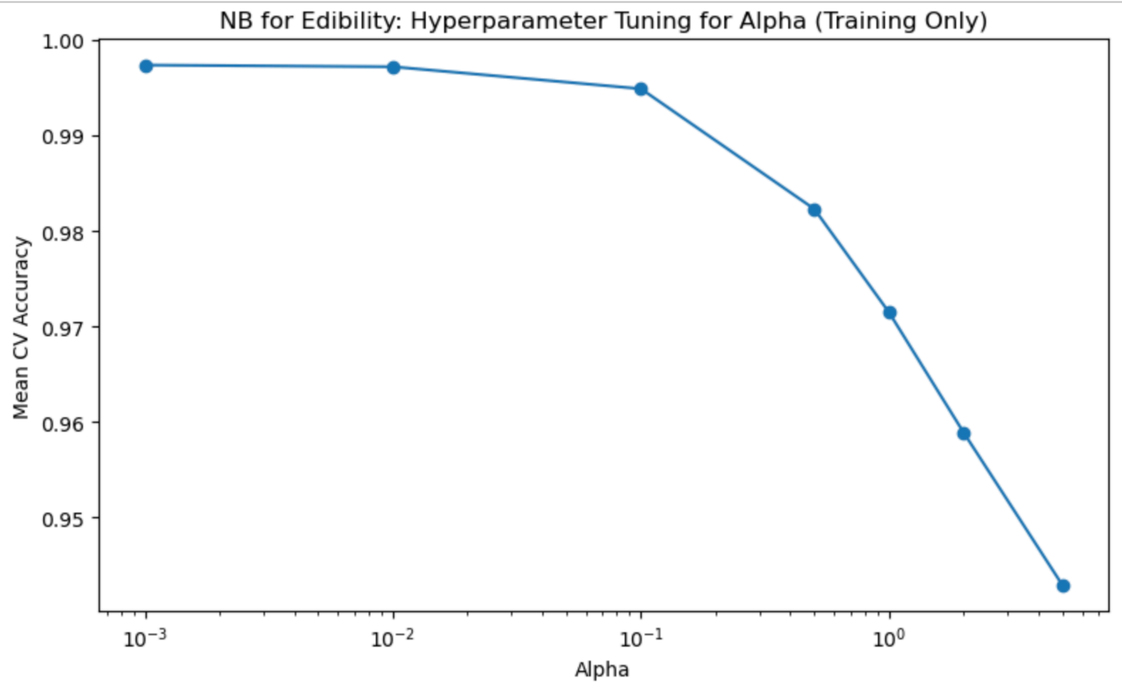
decision_tree_poisonous_accuracy_q22.png Download



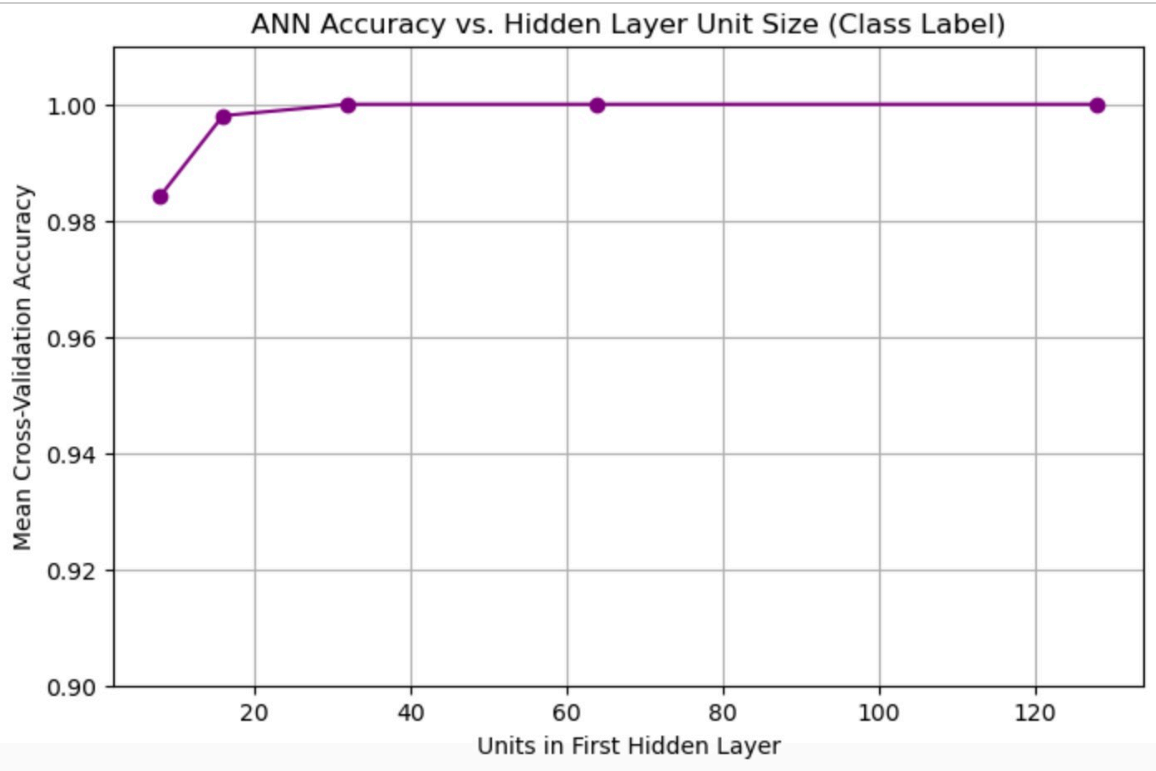
KNN_hyper_edibility.png Download



▾ NB_hyper_edibility.png ⭳ Download

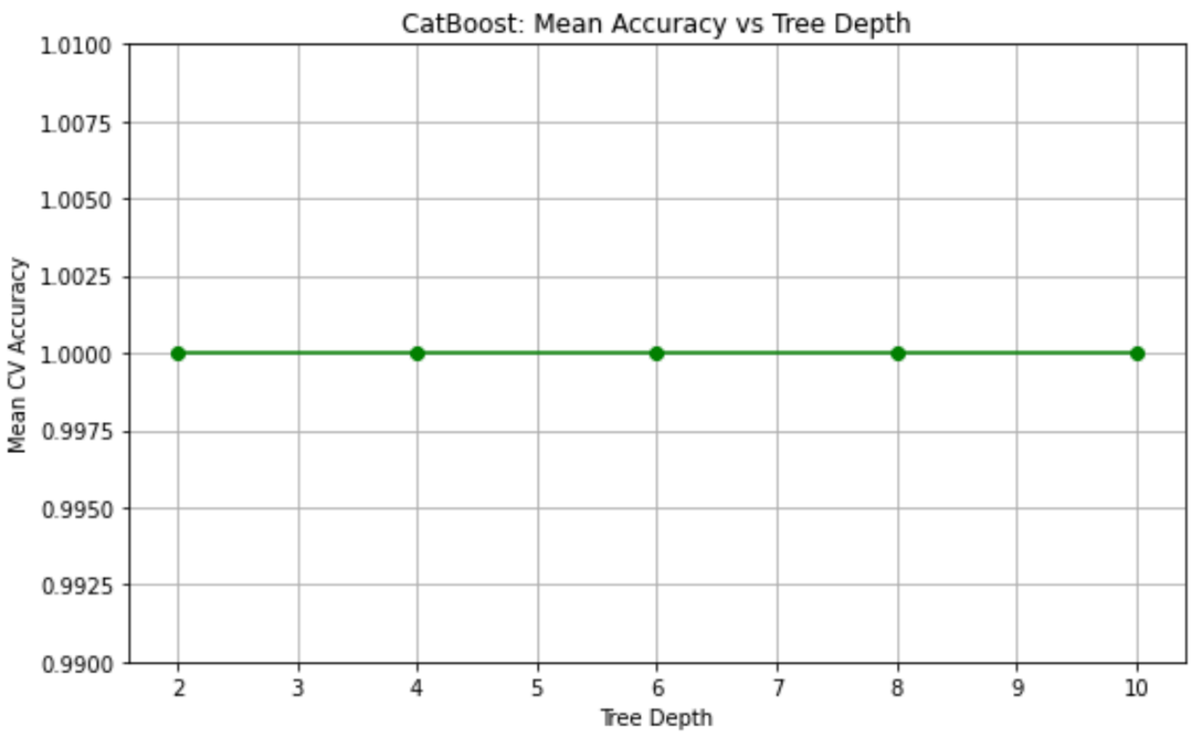


▾ 221181745190997_.pic.jpg ⭳ Download



Screenshot 2025-04-20 at 20.40.57.png

Download



Q2.3 Pick your best hyperparameter values
1 Point

Which hyperparameter values did you end up picking for each model? Explain why you picked each value:

For the classification problem to predict poisonousness (poisonous or not), we chose the following hyperparameters for each model:

1. For the decision tree, we chose the set of cost_complexity, tree depth, and minimum number of observations that must exist in a terminal leaf node as hyperparameters. Cost complexity controls the penalty for adding additional splits to the tree. A higher value makes the model more conservative (i.e., simpler trees with fewer splits), while a lower value allows for more complex trees. We tested [0.001, 0.005, 0.01, 0.05, 0.1] to cover low and high regularization. Tree depth sets the maximum depth a tree can grow, controlling how many splits the tree is allowed to make from the root to a leaf. We tested [3, 5, 7, 9, 11] from shallow to fairly deep trees. For the minimum number of observations in a leaf node, larger values prevent the tree from growing too deep with small terminal nodes, reducing overfitting; smaller values allow more detailed trees with small leaves. We ended up picking cost complexity = 0.01, tree depth = 3, min number of observations at leaf node = 5, since such hyperparameters yielded an averaged accuracy of 0.9972439 and an averaged area under the receiver operating curve = 0.9963918 on the cross-validated training data.

2. For random forest, we chose mtry, the number of features randomly selected at each split in the random forest as the hyperparameter. This hypreparameter controls the randomness and diversity of the trees in the forest. Lower values means more randomness, more diverse trees. Higher values mean less randomness, each tree sees more features, thus potentially stronger individual trees but more correlation between them. We tested values 4, 6, 8, 10, 12, 14, 16, the square root of the number of predictors (rounded down), half of the number of predictors (rounded down) to compare performance across a broad spectrum of model complexity. We ended up picking mtry = 2 due to its high accuracy and area under receiver operating curve.

3. For CatBoost, I tested depth = [2, 4, 6, 8, 10] using 5-fold stratified cross-validation.
All depths give perfect accuracy (1.0000), so we chose the smallest depth (2) to minimize complexity and reduce the risk of overfitting. A shallower tree also trains and predicts faster while giving the same performance.

4. With ANN, we selected the number of units in the first hidden layer as the hyperparameter to tune. This parameter represents the model’s capacity to learn complex patterns from the input features. We tested the values [8, 16, 32, 64, 128]. We chose 64 units in the first hidden layer as the final value for this task. It achieved perfect accuracy while maintaining a lower training time compared to larger ones.

5. For KNN, we selected k, the number of nearest neighbors, as the hyperparameter to tune (with Hamming distance as the distance metric as all features are categorical). This hyperparameter determines how many neighbors the algorithm considers when assigning a class label to a new data point. A smaller k makes the model more sensitive to local patterns, and a larger k averages over a broader neighborhood. We specifically tested the values from 1-50 with 2 interval in between, and we got k = 5, k = 3, k = 1 reached the highest, perfect mean CV accuracy = 1.0. We finally picked k = 5, as it has a smaller probability for overfitting than k = 3 or k = 1, thus achieving a better bias-variance balance.

6. For Naive Bayes Classifier, we selected alpha, the Laplace smoothing parameter, as the hyperparameter to tune. This parameter helps to prevent zero-probability errors. We specifically tested alpha = [0.001, 0.01, 0.1, 0.5, 1.0, 2.0, 5.0], and alpha = 0.001 achieved the highest accuracy = 0.9973. This very small (nearly 0 but not equal to 0) value prevents zero probability that causes error, and at the same time prevents overly flattening the class-conditional probability distributions.

Q2.4 Validation results
2 Points

Train your models on all the training data, using the chosen hyperparameter values. What was your performance and runtime for training?

Training

1. For decision tree, hyperparameter combination cost complexity = 0.01, tree depth = 3, min number of observations at leaf node = 5 yielded averaged accuracy of 0.9972439 and averaged area under receiver operating curve = 0.9963918 on the cross validated training data. It took ~0.047 seconds to train model on the classification problem that classifies poisonous.

2. For random forest tree, hyperparameter (number of predictors sampled for spliting at each node) mtry = 2 yielded area under receiving operating curve of 1, sensitivity of 1 and specificity of 0.9427835. It took around ~5.14 seconds to train model on the classification problem that classifies poisonous.

3. For CatBoost, the chosen hyperparameter value tree depth = 2 gives an average training accuracy of 1.0000. It took approximately 0.2932 seconds to train the model.

4. For ANN, we selected 64 units in the first hidden layer as the final hyperparameter value. The model was trained in approximately 0.90 seconds with training accuracy 1.00.

5. For KNN, hyperparameter k = 5 reached accuracy of 1.000 in 0.008392 seconds for all training data.

6. For Naive Bayes Classifier, hyperparameter alpha = 0.001 reached accuracy of 0.9974 in 0.0155 seconds for all training data.

Predict the validation data with your trained model.
What was your performance and runtime for testing?

Testing

1. Decision tree predicted on the test data with accuracy of 1 and area under receiver operating curve of 1. It takes decision tree model ~0.0125 seconds to predict poisonous classification problem

2. Random forest predicted on the test data with accuracy of 0.9628, with

195 true positives, 349 true negatives, and 21 false positives from confusion matrix. It takes random forest ~0.0307 seconds to predict poisonous classification problem.

3. CatBoost evaluated the validation data with the trained model with depth = 2. It achieved a testing accuracy of 1.0000 in approximately 0.0045 seconds.

4. ANN achieved a testing accuracy of 1.0000 and testing took 0.0378 seconds.

5. KNN predicted on the test data with accuracy of 1.000, with 216 true positives, 349 true negatives, 0 false positive and 0 false negative from confusion matrix. It takes 0.087 seconds to make the prediction.

6. Naive Bayes Classifier predicted on the test data with accuracy of 0.9965, with 216 true positives, 347 true negatives, 2 false positives and 0 false negative from confusion matrix. It takes 0.0007 seconds to make the prediction.

Q2.5 Repeat 2.1-2.4
5 Points

Reduce the dimensions of your dataset in half. Split your dataset for the first classification problem into a training set and a validation set. Upload them here:

▾ pca1_train.csv Download

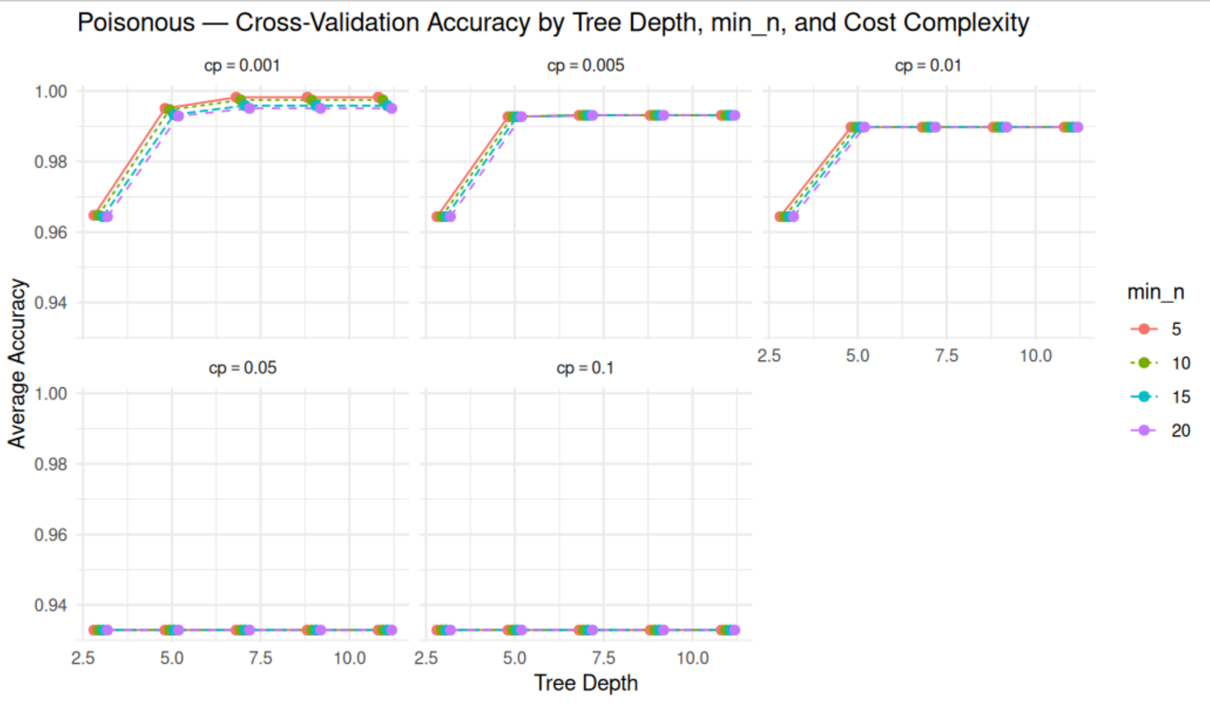
1 Large file hidden. You can download it using the button above.

▾ pca1_test.csv Download

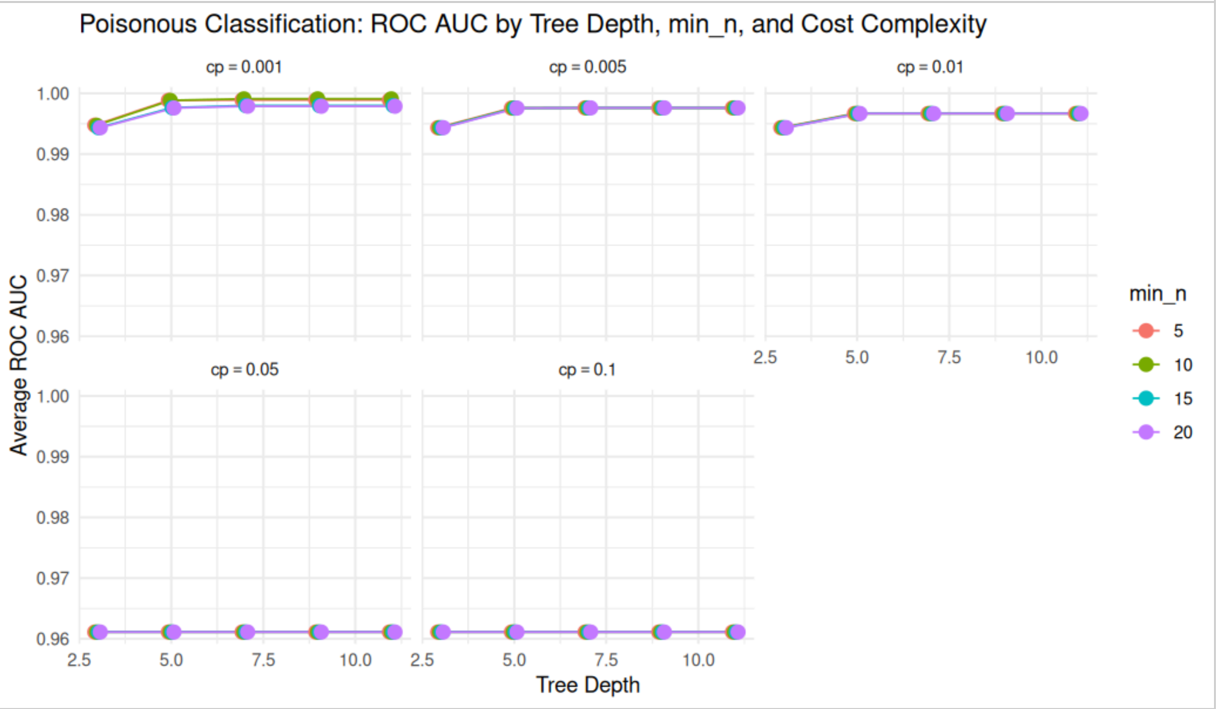
1 Large file hidden. You can download it using the button above.

Upload visualizations for each model that shows the k-fold cross-validation results from testing different hyperparameter models

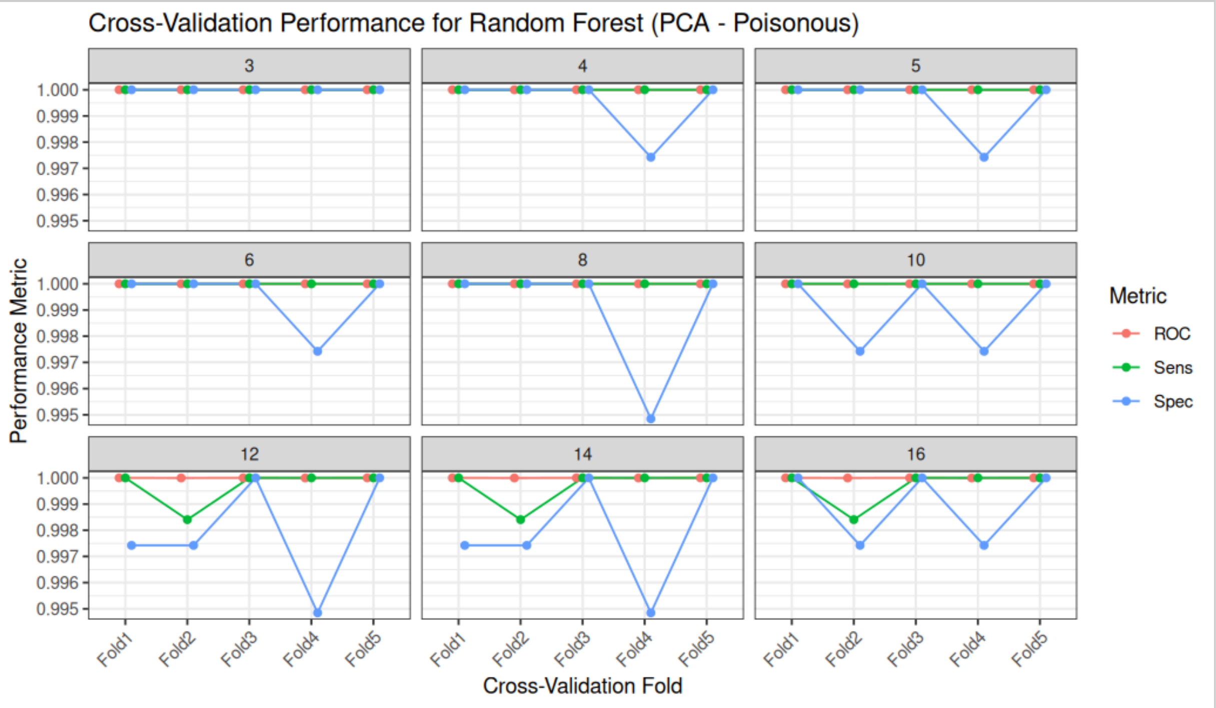
▾ decision_tree_crossed_accuracy_poisonous_pca_q25.png Download



decision_tree_crossed_roc_auc_pca_poisonous_q25.png Download

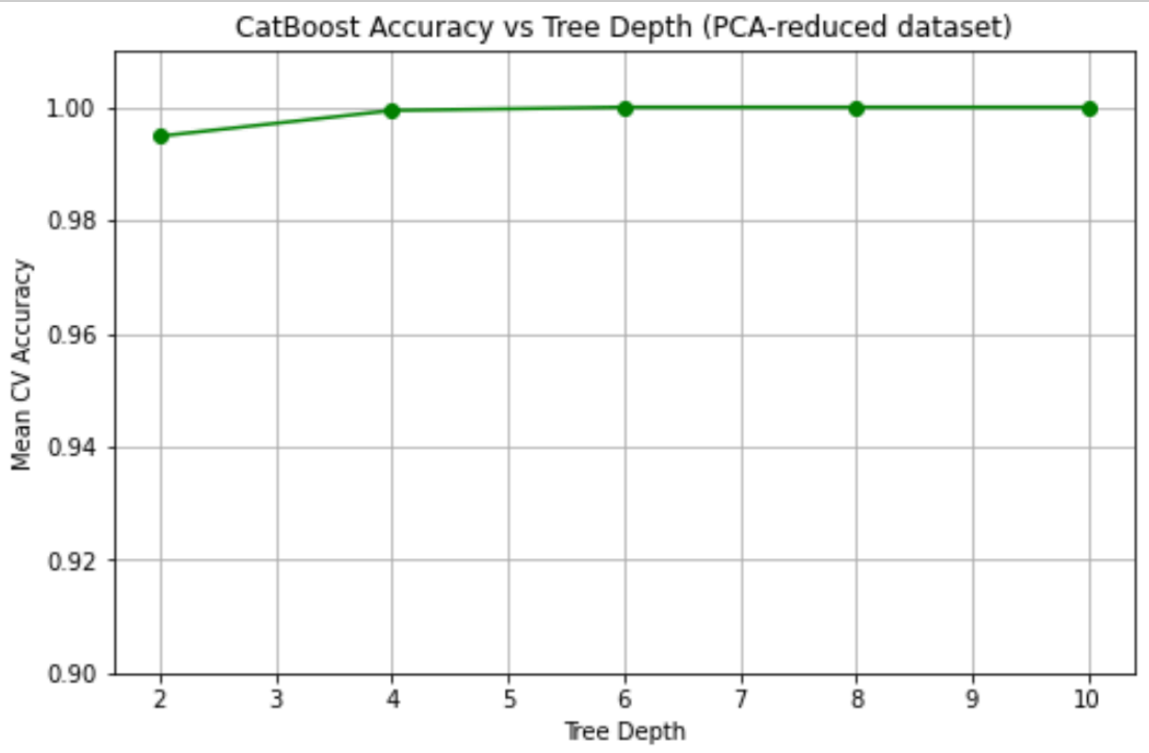


random_forest_crossed_pca_poisonous_q25.png Download



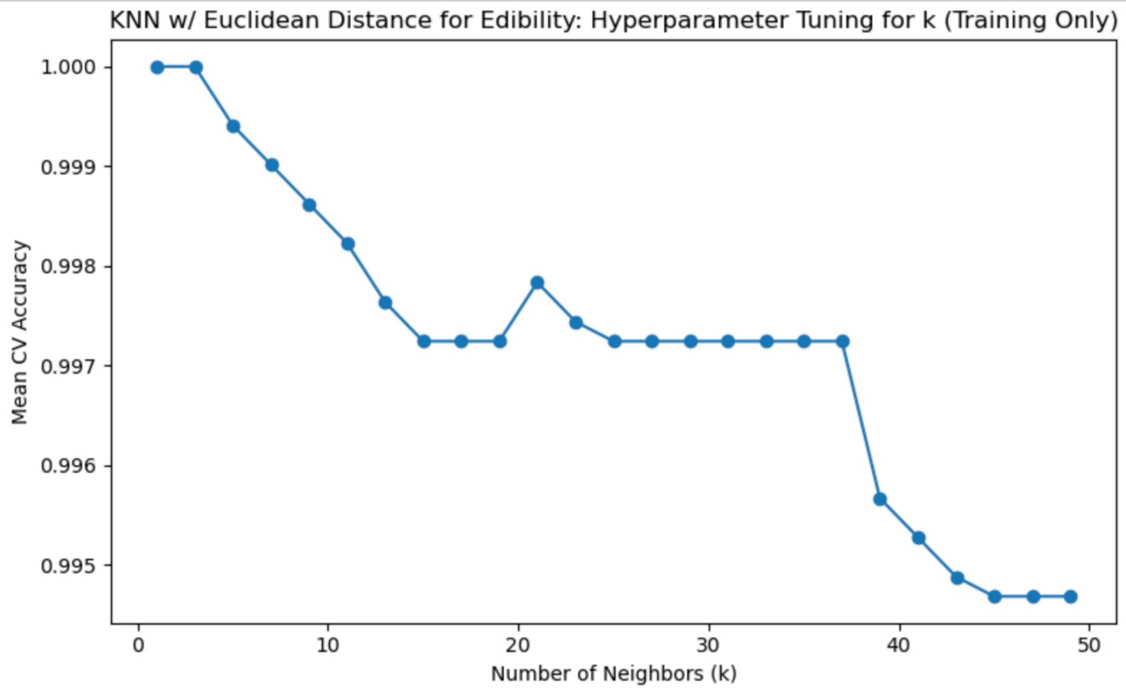
Screenshot 2025-04-20 at 16.24.19.png

Download

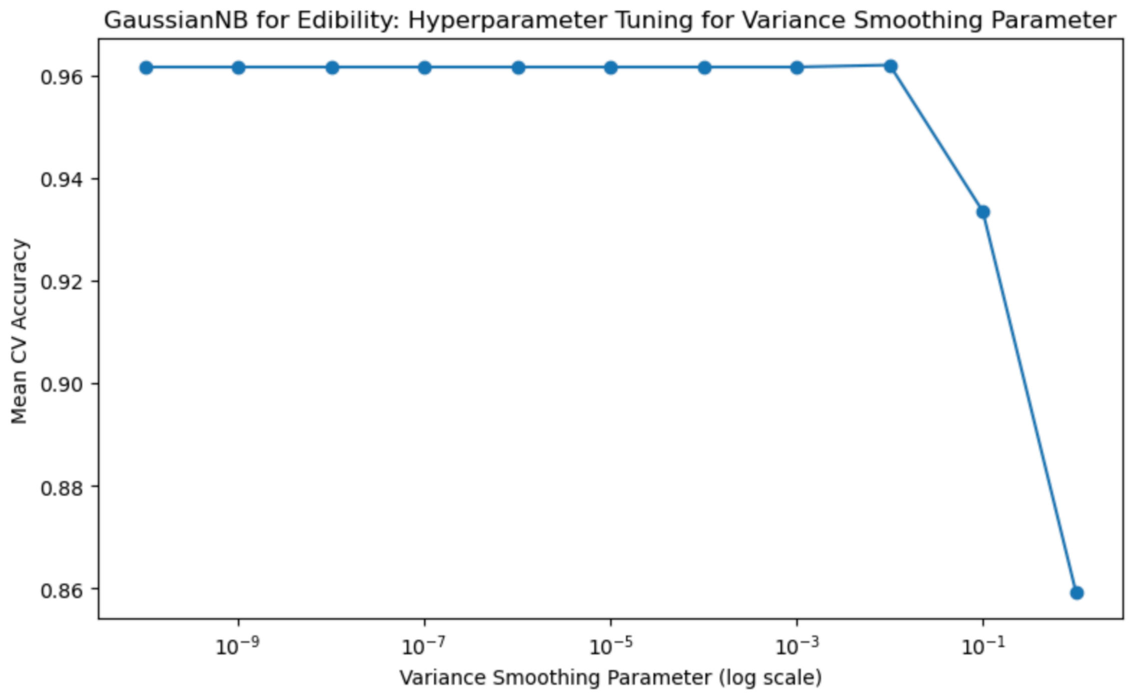


KNNPCA_hyper_edibility_.png

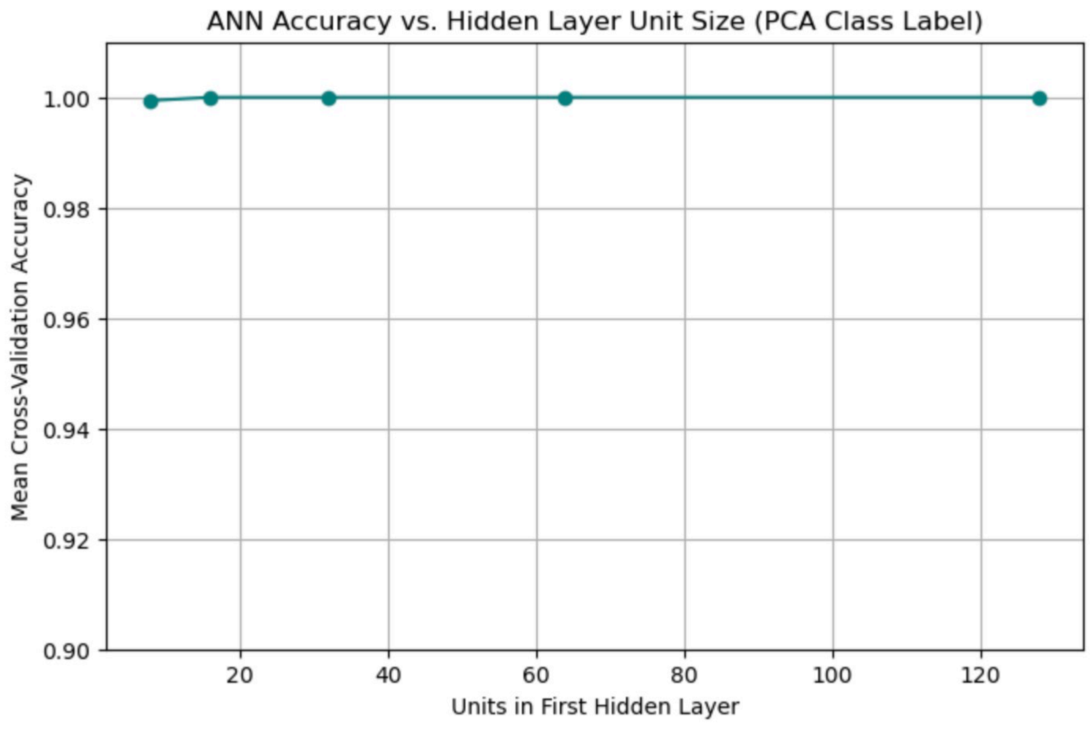
Download



▾ NBPCA_hyper_edibility.png Download



▾ 221211745191026_.pic.jpg Download



Which hyperparameter values did you end up picking for each model? Explain why you picked each value:

For the classification problem to predict poisonousness (PCA used for dimensionality reduction) we chose the following hyperparameters for each model:

1. Decision tree: After dimension reduction for the poisonous classification

problem, we chose the hyperparameter set (cost complexity = 0.001, tree depth = 7, min_n the minimum number of observations that must exist in a terminal leaf node = 5) for the decision tree model. This hyperparameter set is chosen due to high accuracy across k = 5 cross validation.

2. Random forest: After dimension reduction for the poisonous classification problem, we chose the hyperparameter mtry = 3 (number of predictors sampled for spliting at each node) for the random forest model for high area under receiver operating curve.

3. CatBoost: we selected a tree depth of 6. This value is chosen because it achieved the highest mean accuracy (1.0000) while maintaining a low training time (0.17 seconds). Deeper trees like depth 8 and 10 also achieved perfect accuracy, but with higher training costs.

4. ANN: we selected 16 units in the first hidden layer as the optimal hyperparameter. The model achieved a mean accuracy of 1.0000, matching the performance of larger architectures but with a slightly lower training time (0.78s) and comparable testing time (0.04s). This shows that the choice provides an ideal performance and computational efficiency, avoiding unnecessary model complexity but still giving perfect accuracy.

5. KNN: after dimension reduction for the poisonous classification problem, we still tuned the hyperparameter k, the number of nearest neighbors. But this time, we chose Euclidean distance as the distance metric instead of Hamming distance as the features after PCA all became continuous. Specifically, we tested k values from 1-50 with interval = 2. k = 1 and k = 3 ended up both achieving the highest and perfect accuracy = 1, and we picked k = 3 as the best value to reduce the risk of overfitting (compared to k = 1).

6. Naive Bayes Classifier: after dimension reduction for the poisonous classification problem, we needed to run Gaussian Naive Bayes instead of categorical Naive Bayes as all features were converted to continuous variables. The hyperparameter to tune in this context is the var_smoothing variable, which controls the amount of variance added to each feature to prevent numerical instability due to division by near-zero values. We tested var_smoothing from 1e^-10 to 1e^0, with 11 logarithmically spaced intervals. We found that 1e^-2 reached the highest accuracy of 0.9620. Therefore, we ended up picking the best value to be var_smoothing = 1e^-2.

Train your models on all the training data, using the chosen hyperparameter values. What was your performance and runtime for training?

Training

1. Decision tree: after dimension reduction, it takes decision tree model ~0.0364 seconds to train on the poisonous classification problem. Under the hyperparameter set (cost complexity = 0.001, tree depth = 7, minimum number of observations that must exist in a terminal leaf node min_n = 5), the decision tree achieved averaged accuracy of 0.998, and averaged area under receiver operating curve of 0.999 across the k = 5 validation folds.

2. Random forest: after dimension reduction, it takes random forest model ~1.03 seconds to train on the poisonous classification problem. mtry the number of predictors sampled for spliting at each node mtry = 3 for the random forest model yields area under receiver operating curve, sensitivity and specificity all equal to 1.

3. CatBoost: with a tree depth of 6, the model achieved a training accuracy of 1.0000, in 0.2135 seconds.

4. ANN: we achieved a final training accuracy of 0.9980, with a total training time of 0.72 seconds.

5. KNN: after dimension reduction, with the best k = 3, it takes KNN 0.012279 seconds to train the whole training data, reaching an accuracy of 1.000.

6. Naive Bayes Classifier: after dimension reduction, with the best var_smoothing value 1e^-2, it takes NB Classifier 0.002205 seconds to train the whole training data, reaching an accuracy of 0.9624.

Predict the validation data with your trained model.
What was your performance and runtime for testing?

Testing

1. Decision tree: after dimension reduction, it takes decision tree model ~0.0103 seconds to predict on the poisonous classification problem. The model achieved perfect accuracy and area under receiver operating curve (both equal to 1) on the testing data.

2. Random forest: after dimension reduction, it takes random forest model ~0.0176 seconds to predict on the poisonous classification problem. The decision tree predicted 216 true positives, 349 true negatives with accuracy of 1, 95% confidence interval of (0.9935, 1), no information rate of 0.6177, p value [Acc > NIR] < 2.2e-16, kappa of 1. The prevalence, detection rate, and

detection prevalence were each 0.6177.

3. CatBoost: using a tree depth of 6, the model achieved a testing accuracy of 1.0000. The runtime for testing is approximately 0.0027 seconds.

4. ANN: we achieved a final test accuracy of 0.9965, with a total test time of 0.0206 seconds.

5. KNN: after dimension reduction, with the best k = 3, it takes KNN 0.014465 seconds to predict the testing data, reaching an accuracy of 1.000. We got 216 true positives, 349 true negatives, 0 false positives and 0 false negatives.

6. Naive Bayes Classifier: after dimension reduction, with the best var_smoothing value 1e^-2, it takes NB Classifier 0.000399 seconds to predict the testing data, reaching an accuracy of 0.9540. Specifically, the testing results contain 210 true positives, 329 true negatives, 20 false positives, and 6 false negatives.

Q3 Classification problem 2

10 Points

Repeat the work from problem 1 on the second problem

Q3.1 Without dimension reduction
5 Points

Split your dataset for the second classification problem into a training set and a validation set. Upload them here:

y2_train_dropped.csv

Download

```
odor
n
a
n
n
n
a
n
f
n
l
c
l
f
f
n
f
a
n
n
n
n
n
n
n
f
l
n
f
f
n
m
f
p
a
n
f
n
f
n
n
f
a
a
n
f
n
n
f
```

```
n
f
n
n
n
n
f
n
n
n
l
n
n
n
f
n
c
f
a
n
n
n
f
n
n
n
n
f
n
n
n
n
n
f
n
n
n
n
a

```

y2_test_dropped.csv

Download

```
odor
n
f
n
n
a
f
n
n
c
n
n
p
c
f
a
a
n
f
n
n
n
n
n
f
f
n
n
n
f
l
a
c
f
f
l
n
c
n
n
n
n
n
f
n
n
n
a
f
```

```
c
n
f
n
f
n
f
f
n
n
n
n
n
l
n
l
l
n
n
f
n
l
n
f
n
f
n
l
n
l
n
n
n
n
f
a
f
f
n
f
f
n
a
f
n
n
p
f
l

```

X2_train_dropped.csv Download

1	Large file hidden. You can download it using the button above.

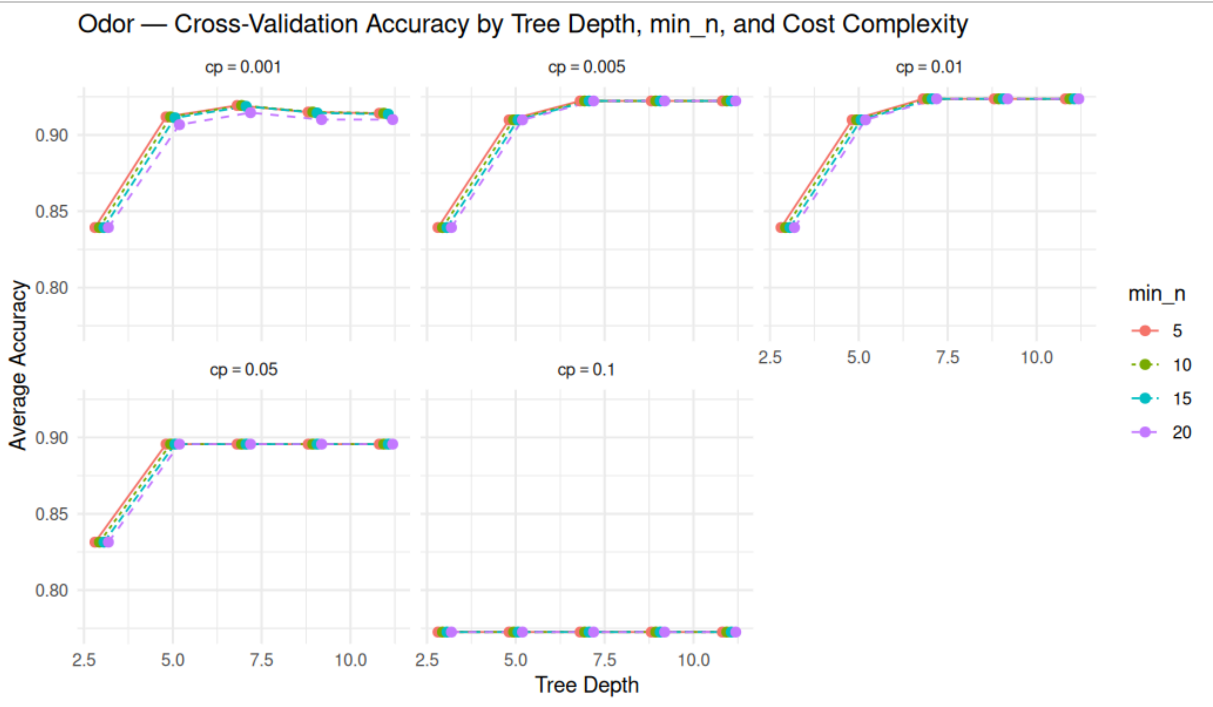
X2_test_dropped.csv Download

```
class,cap-shape,cap-surface,cap-color,bruises,gill-attachment,gill-spacing,gill-size,gill-color,stalk-shape,stalk-root,stalk-surface-above-ring,stalk-surface-below-ring,stalk-color-above-ring,stalk-color-below-ring,veil-type,veil-color,ring-number,ring-type,spore-print-color,population,habitat
e,x,y,c,f,f,w,n,w,e,b,f,s,w,n,p,w,o,e,w,v,l
p,f,y,g,f,f,c,b,g,e,b,k,k,b,p,p,w,o,l,h,y,d
e,f,s,n,f,f,w,b,h,t,e,s,s,w,w,p,w,o,e,n,a,g
e,x,f,n,f,f,w,b,k,t,e,s,f,w,w,p,w,o,e,k,a,g
e,f,y,n,t,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,s,p
p,f,y,g,f,f,c,b,g,e,b,k,k,b,p,p,w,o,l,h,y,p
e,x,f,g,t,f,c,b,p,t,b,s,s,g,g,p,w,o,p,n,y,d
e,x,y,g,t,f,c,b,p,t,b,s,s,p,w,p,w,o,p,n,y,d
p,x,f,p,f,f,w,n,p,e,b,s,s,w,w,p,w,o,p,n,s,d
e,x,f,e,t,f,c,b,u,t,b,s,s,p,w,p,w,o,p,n,y,d
e,f,f,e,t,f,c,b,n,t,b,s,s,p,p,p,w,o,p,n,y,d
p,x,s,w,t,f,c,n,w,e,e,s,s,w,w,p,w,o,p,k,v,u
p,x,s,w,f,f,w,n,u,e,b,s,s,w,w,p,w,o,p,n,v,d
p,x,f,y,f,f,c,b,g,e,b,k,k,b,p,p,w,o,l,h,v,d
e,x,y,n,t,f,c,b,n,e,r,s,y,w,w,p,w,o,p,k,y,g
e,x,s,w,t,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,n,m
e,x,y,g,t,f,c,b,u,t,b,s,s,p,g,p,w,o,p,n,v,d
p,x,f,y,f,f,c,b,g,e,b,k,k,n,b,p,w,o,l,h,y,d
e,f,f,g,t,f,c,b,n,t,b,s,s,w,p,p,w,o,p,k,y,d
e,f,y,e,t,f,c,b,p,t,b,s,s,p,w,p,w,o,p,n,y,d
e,x,y,n,t,f,c,b,w,t,b,s,s,w,w,p,w,o,p,n,y,d
e,x,y,e,t,f,c,b,n,t,b,s,s,p,p,p,w,o,p,n,y,d
e,x,f,n,t,f,c,b,w,t,b,s,s,g,g,p,w,o,p,n,v,d
p,x,y,y,f,f,c,b,p,e,b,k,k,n,b,p,w,o,l,h,y,g
p,f,s,w,t,f,c,b,w,t,b,f,f,w,w,p,w,o,p,h,s,g
e,x,f,e,t,f,c,b,p,t,b,s,s,w,w,p,w,o,p,k,v,d
e,x,f,n,t,f,c,b,n,t,b,s,s,p,g,p,w,o,p,k,v,d
e,x,y,n,t,f,c,b,p,t,b,s,s,p,g,p,w,o,p,k,v,d
p,f,f,y,f,f,c,b,p,e,b,k,k,p,n,p,w,o,l,h,v,g
e,x,y,w,t,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,g
e,x,y,y,t,f,c,b,p,e,r,s,y,w,w,p,w,o,p,k,y,g
p,x,f,g,f,f,w,n,u,e,b,s,s,w,w,p,w,o,p,n,s,d
p,x,s,g,t,f,c,b,p,t,b,s,f,w,w,p,w,o,p,h,s,u
p,x,y,g,f,f,c,b,p,e,b,k,k,p,b,p,w,o,l,h,v,p
e,f,s,y,t,f,w,n,p,t,b,s,s,w,w,p,w,o,p,u,v,d
e,x,f,n,f,f,w,b,p,t,e,f,s,w,w,p,w,o,e,k,a,g
p,x,f,w,f,f,w,n,p,e,b,s,s,w,w,p,w,o,p,k,v,d
e,x,f,n,t,f,c,b,w,t,b,s,s,g,w,p,w,o,p,k,y,d
e,f,y,e,t,f,c,b,p,t,b,s,s,g,w,p,w,o,p,k,y,d
e,f,y,g,t,f,c,b,u,t,b,s,s,w,w,p,w,o,p,k,v,d
e,x,s,g,f,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,s,g
e,f,y,g,t,f,c,b,w,t,b,s,s,w,p,p,w,o,p,k,v,d
p,x,y,y,f,f,c,b,g,e,b,k,k,n,b,p,w,o,l,h,y,g
p,f,s,p,t,f,c,b,r,e,b,s,s,w,w,p,w,t,p,r,v,m
e,x,f,n,t,f,c,b,u,t,b,s,s,w,w,p,w,o,p,n,v,d
```

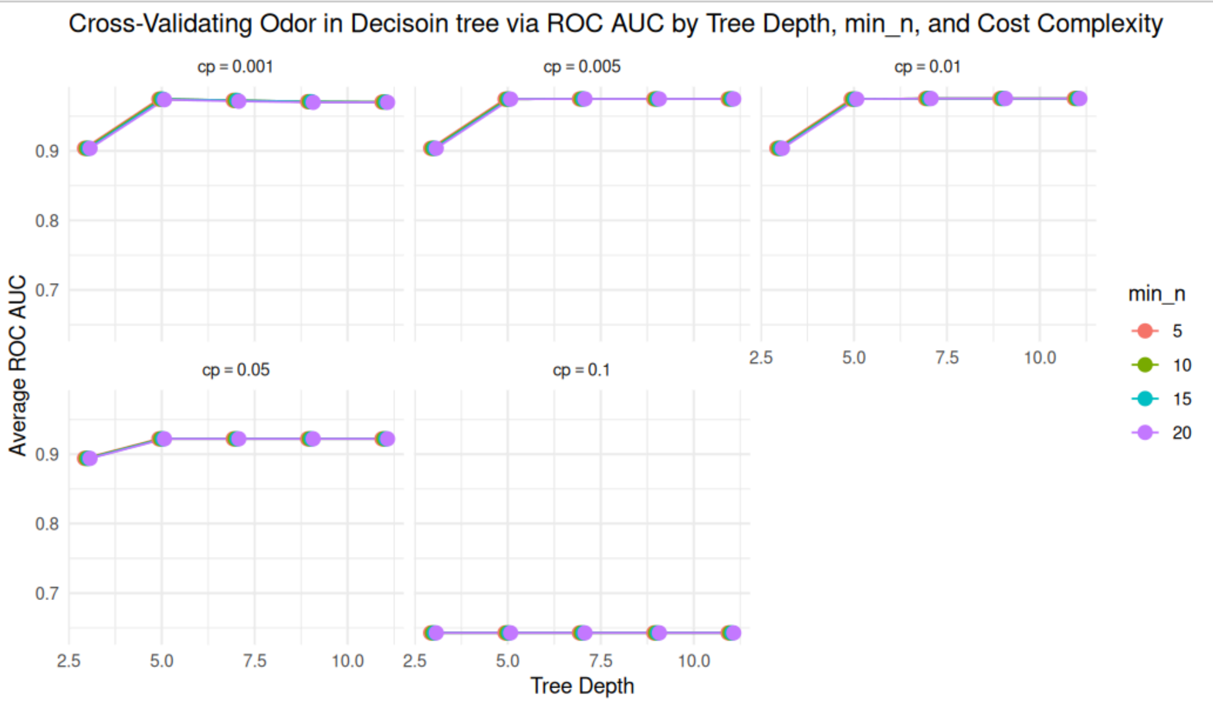
```
p,f,y,g,f,f,c,b,g,e,b,k,k,n,b,p,w,o,l,h,v,d
e,b,s,w,t,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,s,m
e,f,y,g,t,f,c,b,p,t,b,s,s,g,p,p,w,o,p,k,v,d
p,x,f,g,f,f,c,n,u,e,b,s,s,w,w,p,w,o,p,n,s,d
e,x,f,g,t,f,c,b,u,t,b,s,s,g,p,p,w,o,p,k,v,d
p,f,s,b,t,f,c,b,p,t,b,s,f,w,w,p,w,o,p,h,v,u
e,x,f,e,t,f,c,b,u,t,b,s,s,g,w,p,w,o,p,k,y,d
p,f,y,g,f,f,c,b,h,e,b,k,k,p,b,p,w,o,l,h,v,g
e,x,f,e,t,f,c,b,n,t,b,s,s,p,w,p,w,o,p,k,y,d
p,f,s,g,t,f,c,b,p,t,b,f,f,w,w,p,w,o,p,h,s,g
p,f,y,y,f,f,c,b,h,e,b,k,k,b,b,p,w,o,l,h,y,g
e,f,f,g,t,f,c,b,p,t,b,s,s,g,g,p,w,o,p,n,v,d
e,x,y,e,t,f,c,b,p,t,b,s,s,g,w,p,w,o,p,n,y,d
e,f,s,g,f,f,w,b,k,t,e,s,s,w,w,p,w,o,e,k,s,g
e,f,f,g,t,f,c,b,n,t,b,s,s,p,p,p,w,o,p,k,v,d
p,f,y,w,t,f,w,n,w,e,b,s,s,w,w,p,w,o,p,w,c,l
e,x,s,y,t,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,n,m
e,x,s,w,f,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,s,g
e,f,y,n,t,f,c,b,p,e,r,s,y,w,w,p,w,o,p,k,y,g
e,x,y,y,t,f,c,b,p,e,r,s,y,w,w,p,w,o,p,n,y,p
e,f,f,n,t,f,c,b,n,t,b,s,s,w,p,p,w,o,p,n,y,d
e,f,s,g,f,f,w,b,k,t,e,f,f,w,w,p,w,o,e,n,s,g
p,f,y,y,f,f,c,b,h,e,b,k,k,n,n,p,w,o,l,h,y,p
e,x,f,n,f,f,w,b,h,t,e,f,s,w,w,p,w,o,e,k,a,g
e,b,s,w,t,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g
e,x,f,g,t,f,c,b,w,t,b,s,s,p,g,p,w,o,p,k,y,d
p,f,y,g,f,f,c,b,h,e,b,k,k,b,b,p,w,o,l,h,v,d
e,x,y,g,t,f,c,b,p,t,b,s,s,p,p,p,w,o,p,k,v,d
p,f,s,g,t,f,c,b,p,t,b,s,f,w,w,p,w,o,p,h,s,g
e,f,y,e,t,f,c,b,p,t,b,s,s,p,p,p,w,o,p,k,y,d
e,b,s,w,t,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,s,m
p,b,s,p,t,f,c,b,w,e,b,s,s,w,w,p,w,t,p,r,v,g
e,f,f,w,t,f,w,n,p,t,b,s,s,w,w,p,w,o,p,u,v,d
e,x,f,w,f,f,w,b,k,t,e,f,s,w,w,p,w,o,e,k,s,g
e,x,f,n,t,f,c,b,n,t,b,s,s,w,p,p,w,o,p,k,y,d
e,x,y,g,t,f,c,b,w,e,b,s,s,w,w,p,w,t,p,w,v,p
e,f,f,e,t,f,c,b,n,t,b,s,s,p,g,p,w,o,p,k,y,d
p,f,f,g,f,f,c,b,g,e,b,k,k,b,p,p,w,o,l,h,v,p
e,x,s,w,t,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m
p,f,f,y,f,f,c,b,h,e,b,k,k,b,p,p,w,o,l,h,y,g
p,x,s,g,t,f,c,b,h,t,b,f,s,w,w,p,w,o,p,h,v,g
e,x,y,g,t,f,c,b,w,t,b,s,s,w,g,p,w,o,p,n,y,d
p,f,f,g,f,f,c,b,g,e,b,k,k,p,p,p,w,o,l,h,v,d
p,f,f,g,f,f,c,b,g,e,b,k,k,b,b,p,w,o,l,h,y,d
e,x,s,g,f,f,w,b,h,t,e,f,s,w,w,p,w,o,e,n,a,g
e,x,y,y,t,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g
p,f,f,y,f,f,c,b,h,e,b,k,k,n,b,p,w,o,l,h,y,g
e,x,f,e,t,f,c,b,u,t,b,s,s,w,w,p,w,o,p,k,v,d
e,x,s,g,f,f,w,b,p,t,e,s,f,w,w,p,w,o,e,n,s,g
p,x,s,w,t,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u
p,f,f,g,f,f,c,b,h,e,b,k,k,n,b,p,w,o,l,h,y,p
e,b,y,w,t,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,g
```

Upload visualizations for each model that shows the k-fold cross-validation results from testing different hyperparameter models

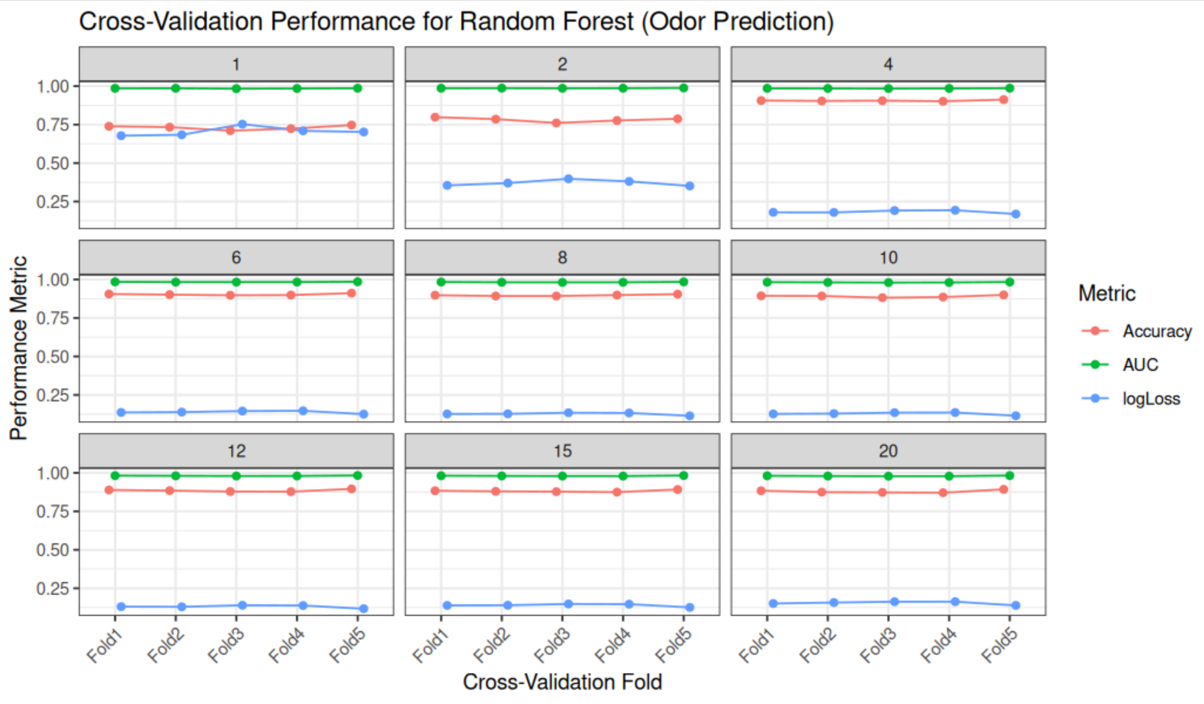
decision_tree_odor_accuracy_q31.png Download



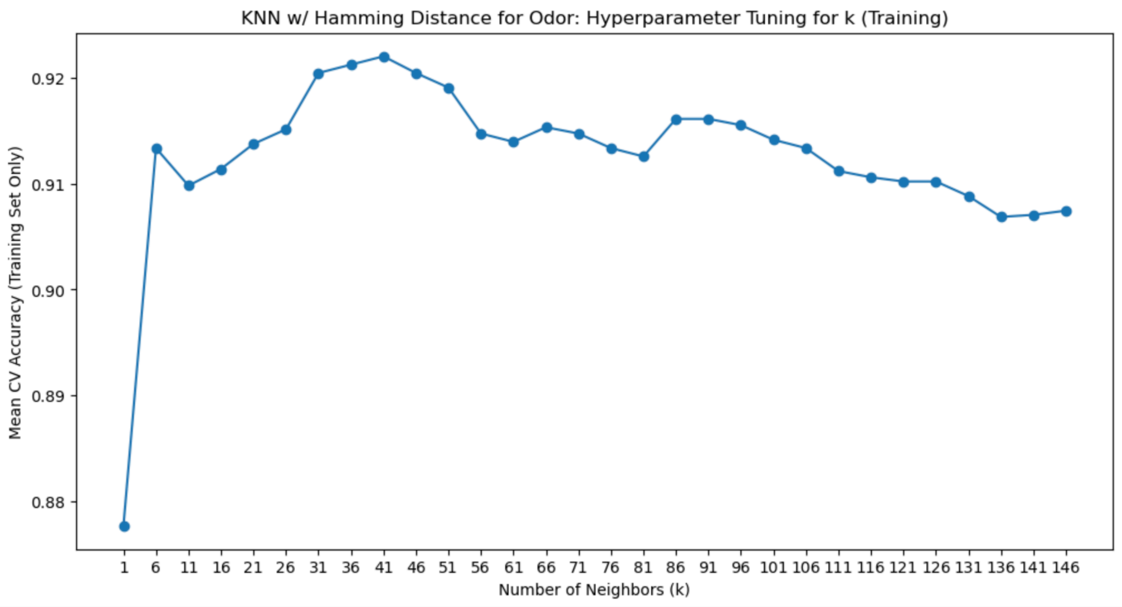
decision_tree_odor_roc_auc_q31.png Download



▾ random_forest_odor_crossed_q31.png Download

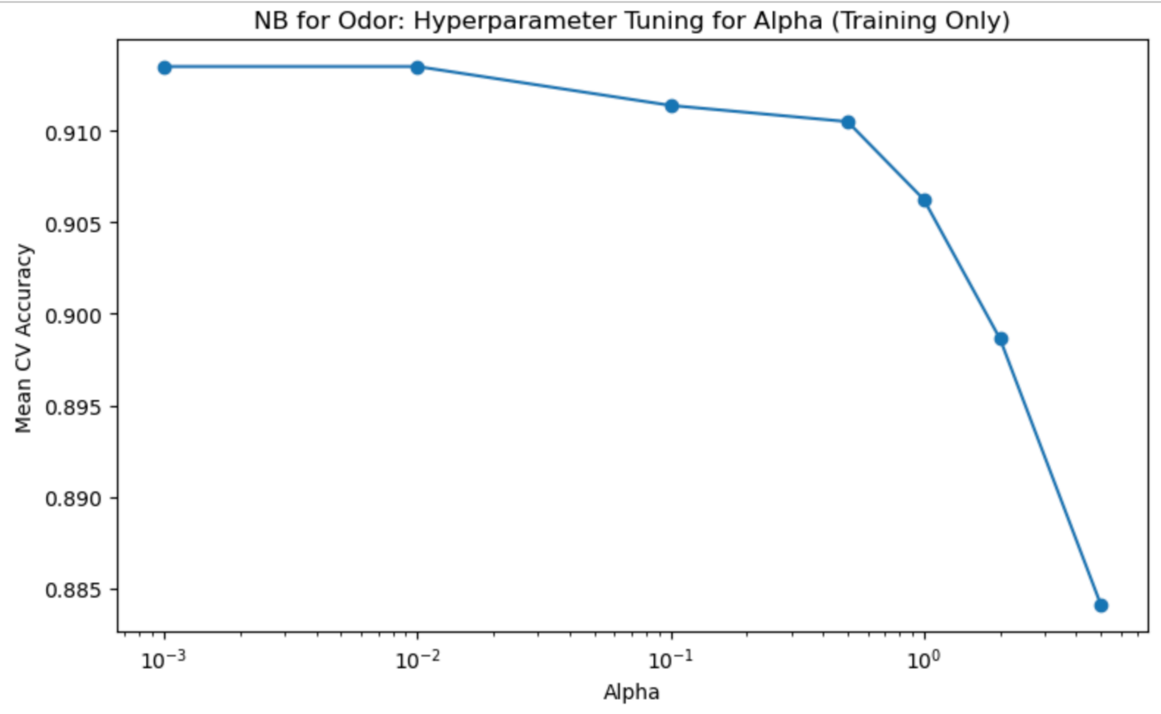


▾ KNN_hyper_odor.png Download

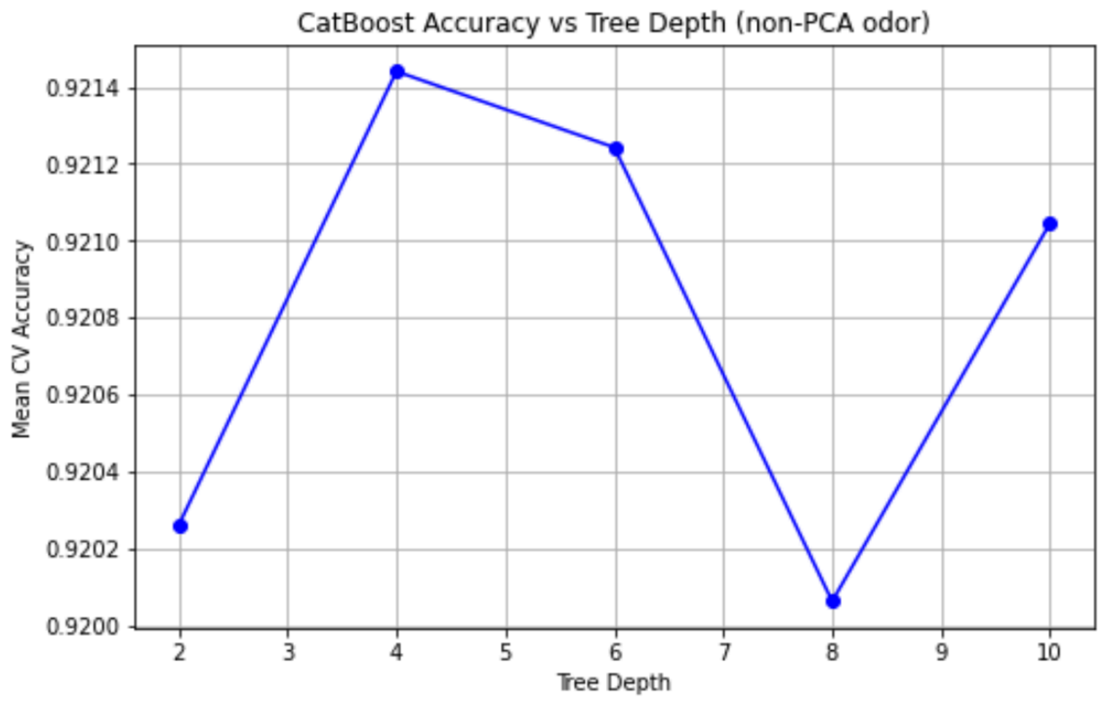


```
Best k on training set: 41, with CV accuracy = 0.9220
```

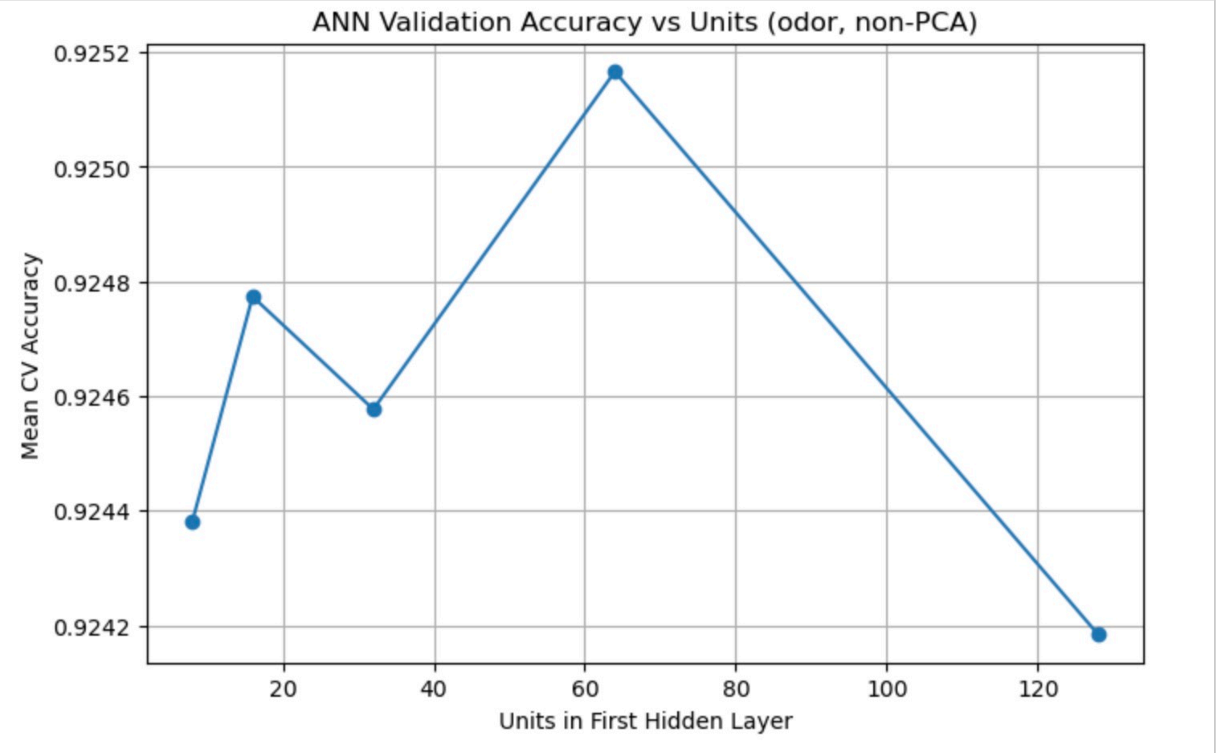
▾ **NB_hyper_odor.png** Download



▾ **Screenshot 2025-04-20 at 20.26.05.png** Download



▾ 221461745195579_.pic.jpg Download



Which hyperparameter values did you end up picking for each model? Explain why you picked each value:

For the classification problem to predict odor, we chose the following hyperparameters for each model:

1. For decision tree, we ended up picking cost complexity = 0.005 , tree depth = 7 , min number of observations at leaf node = 5, since such hyperparameter yielded averaged accuracy of 0.9972439 and averaged area under receiver operating curve = 0.9963918 on the cross validated training data.

2. For random forest, we tested the number of predictors sampled for splitting at each node mtry = 4, 6, 8, 10, 12, 14, 16, the square root of the number of predictors (rounded down), half of the number of predictors (rounded down) to compare performance across a broad spectrum of model complexity. We ended up picking mtry = 8, because it resulted in the lowest log loss. Accuracy at mtry = 8 is quite high: 0.8972230. Kappa, mean F1, sensitivity, and specificity are all strong at this setting, indicating balanced performance across classes.

3. For CatBoost, depth = 4 gives the highest mean validation accuracy of 0.9214, slightly outperforming other values while keeping training time moderate. Hence, we selected depth = 4 as the optimal hyperparameter, balancing both model performance and efficiency.

4. For ANN, the unit size 64 gives the highest mean validation accuracy of 0.9252, slightly outperforming the other configurations. We therefore selected 64 units as the final hyperparameter value, as it gives the best model performance.

5. For KNN, we tuned the number of nearest neighbor k (same reasoning as in Classification problem 1, w/ Hamming distance as the distance metric). We specifically tuned k from 1-150 with interval = 5. We picked k = 41 as the best value as it reached the highest accuracy of 0.9220.

6. For Naive Bayes Classifier, we tuned alpha, the parameter for Laplace smoothing(same reasoning as in Classification problem 1). We specifically tested alpha = [0.001, 0.01, 0.1, 0.5, 1.0, 2.0, 5.0], and we determined alpha = 0.001 as the best value as it achieved the highest accuracy = 0.9135.

Train your models on all the training data, using the chosen hyperparameter values. What was your performance and runtime for training?

Training

1. For decision tree, the chosen hyperparameter set (cost complexity = 0.005 , tree depth = 7 , min number of observations at leaf node = 5) yielded accuracy of 0.7395368 and area under receiver operating curve of 0.7803387, it took ~0.2402 seconds to train the model.

2. For random forest, mtry = 8 (hyperparameter: number of predictors sampled for spliting at each node) at the classification problem 2 to predict odor before dimension reduction, it took ~27.5 seconds to train the model. Its corresponding area under receiver operating curve is 0.9825, so it discriminates between classes well. The model's area under precision-recall curve is 0.4837, meaning moderate performance when focusing on the positive class detection. The model achieved an accuracy of 0.8996, kappa of 0.8492 (strong agreement beyond chance between predicted and actual classes), mean F1 score of 0.7943 (balanced precision and recall across all classes, mean sensitivity of 0.7993, mean specificity of 0.9846, mean balanced accuracy of 0.8919 (robust for imbalanced classes).

3. For CatBoost, the final CatBoost model with depth = 4 achieved a training accuracy of 0.9305. The training time was approximately 2.36 seconds.

4. For ANN, the training process achieved a final training accuracy of 0.9299 with a train time of 0.69 seconds.

5. For KNN, hyperparameter k = 41 reached accuracy of 0.9293 in 0.002363 seconds for all training data.

5. For Naive Bayes Classifier, hyperparameter alpha = 0.001 reached accuracy of 0.9267 in 0.0118 seconds for all training data.

Predict the validation data with your trained model.
What was your performance and runtime for testing?

Testing

1. Decision tree predicted on the test data with accuracy of 0.9221239 and area under receiver operating curve of 0.9743191. It took ~0.0117 seconds to predict odor.

2. Random forest predicted on the test data with accuracy : 0.8708, 95% confidence interval of (0.8403, 0.8973) no information rate of 0.492, p value [Acc > NIR] < 2.2e-16, and kappa of 0.8061. It took ~0.0182 seconds to predict odor.

3. CatBoost: The model achieved a test accuracy of 0.9168. The testing time is 0.0057 seconds.

4. ANN: It achieved a test accuracy of 0.9221 with a test time of 0.0189 seconds.

5. KNN predicted on the test data with accuracy of 0.9133, with 516 correctly predicted and 49 incorrectly predicted instances from the confusion matrix. It takes 0.085 seconds to make the prediction.

6. Naive Bayes Classifier predicted on the test data with accuracy of 0.9080, with 513 correctly predicted and 52 incorrectly predicted instances from confusion matrix. It takes 0.0009 seconds to make the prediction.

Q3.2 With dimension reduction
5 Points

Reduce the dimensions of your dataset in half. Split your dataset for the second classification problem into a training set and a validation set. Upload them here:

pca2_train.csv Download

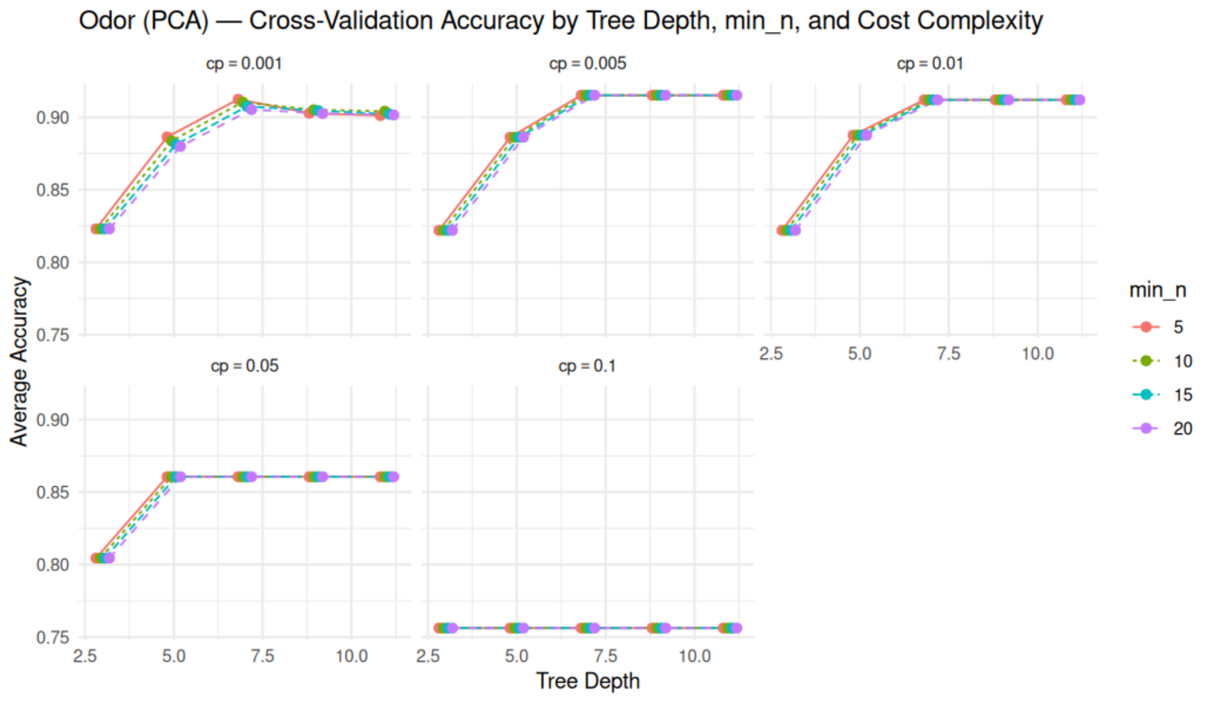
1 Large file hidden. You can download it using the button above.

pca2_test.csv Download

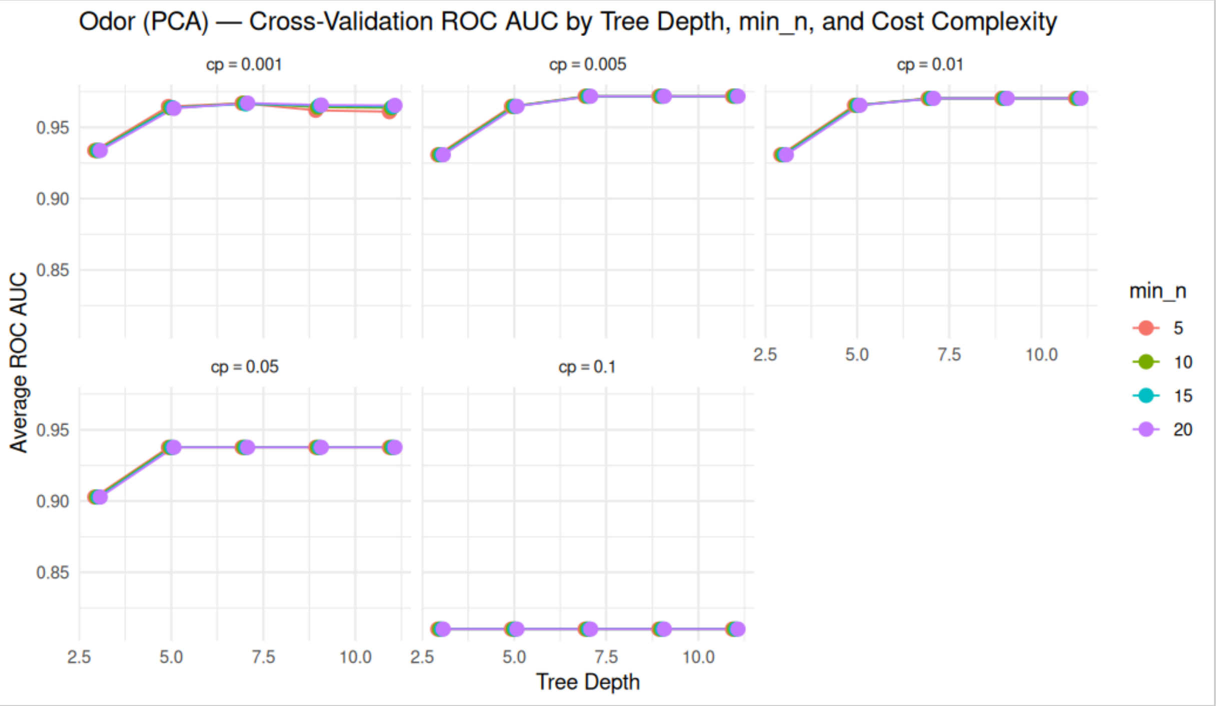
1 Large file hidden. You can download it using the button above.

Upload visualizations for each model that shows the k-fold cross-validation results from testing different hyperparameter models

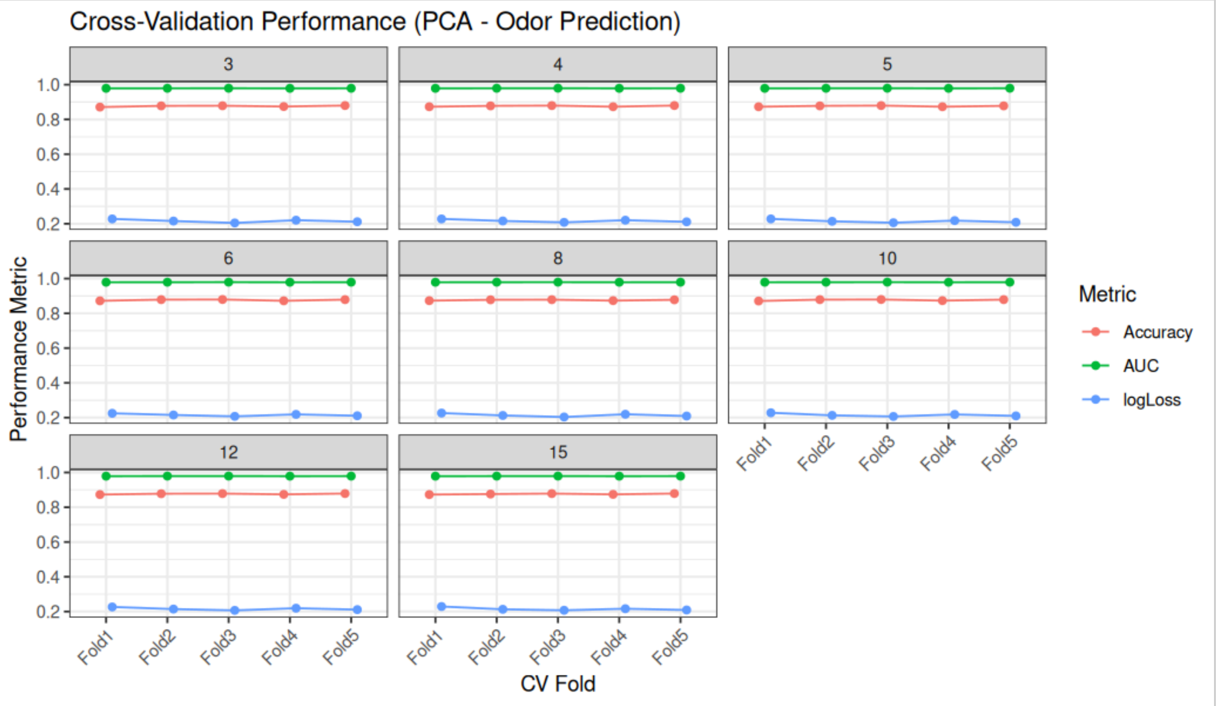
decision_tree_crossed_accuracy_pca_odor_q32.png Download



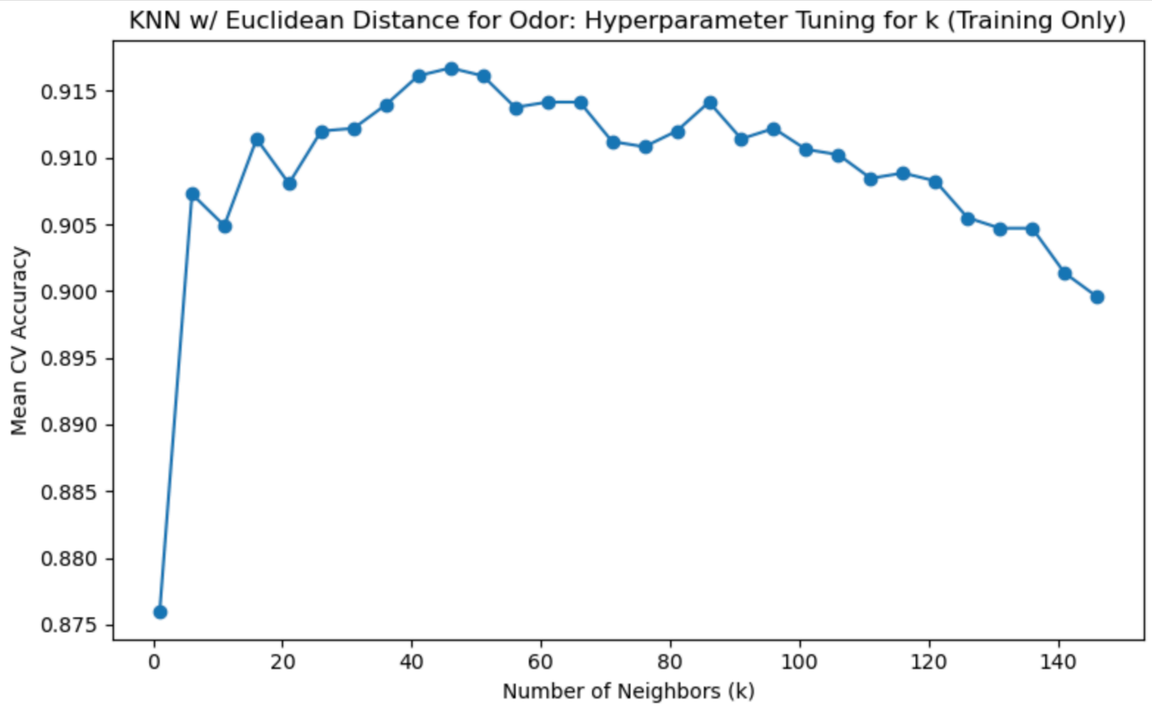
decision_tree_crossed_roc_auc_pca_odor_q32.png Download



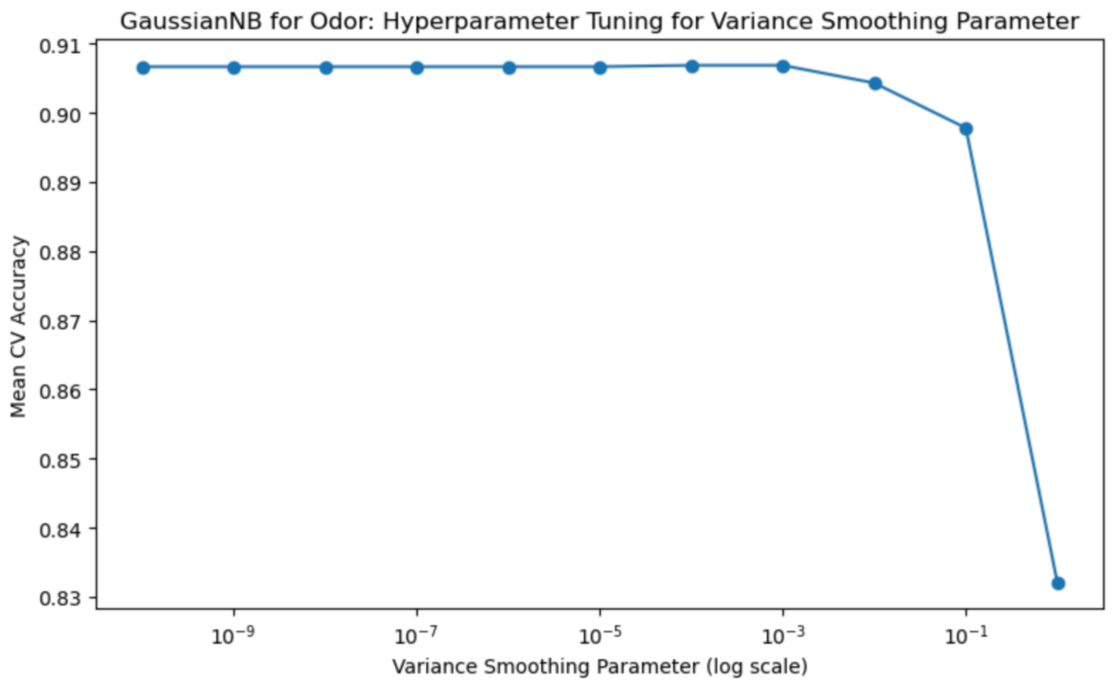
random_forest_crossed_pca_odor_q32.png Download



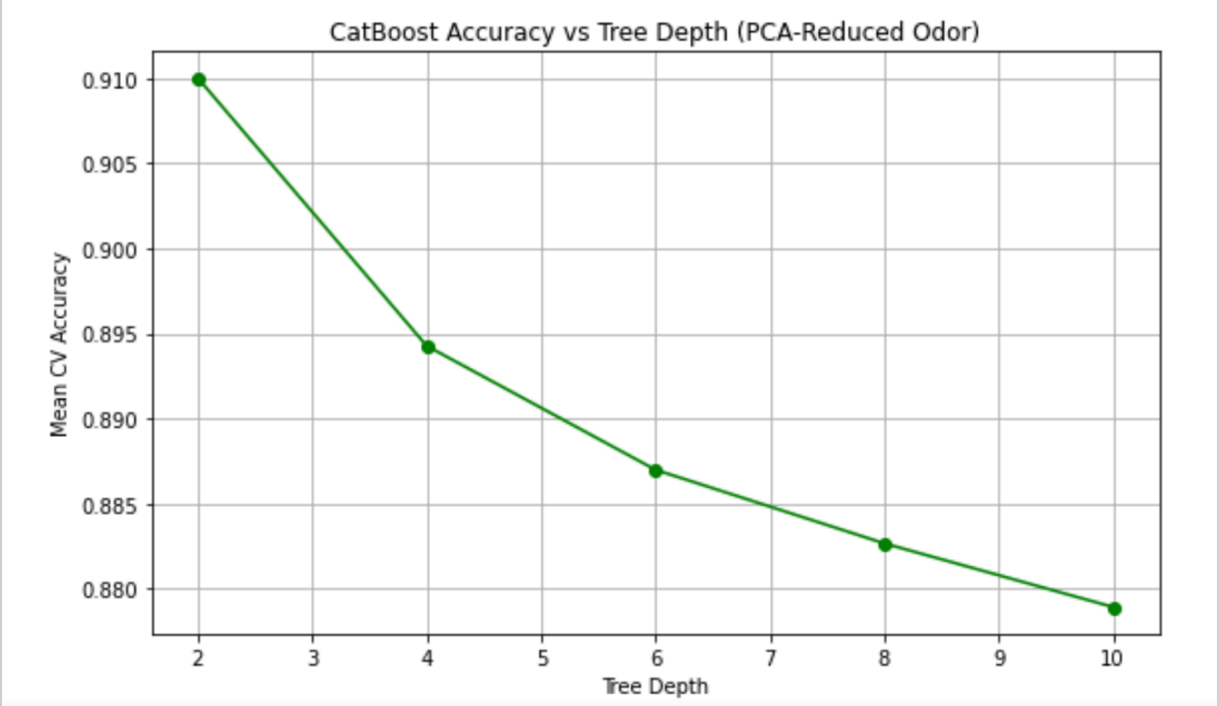
▾ KNNPCA_hyper_odor.png Download



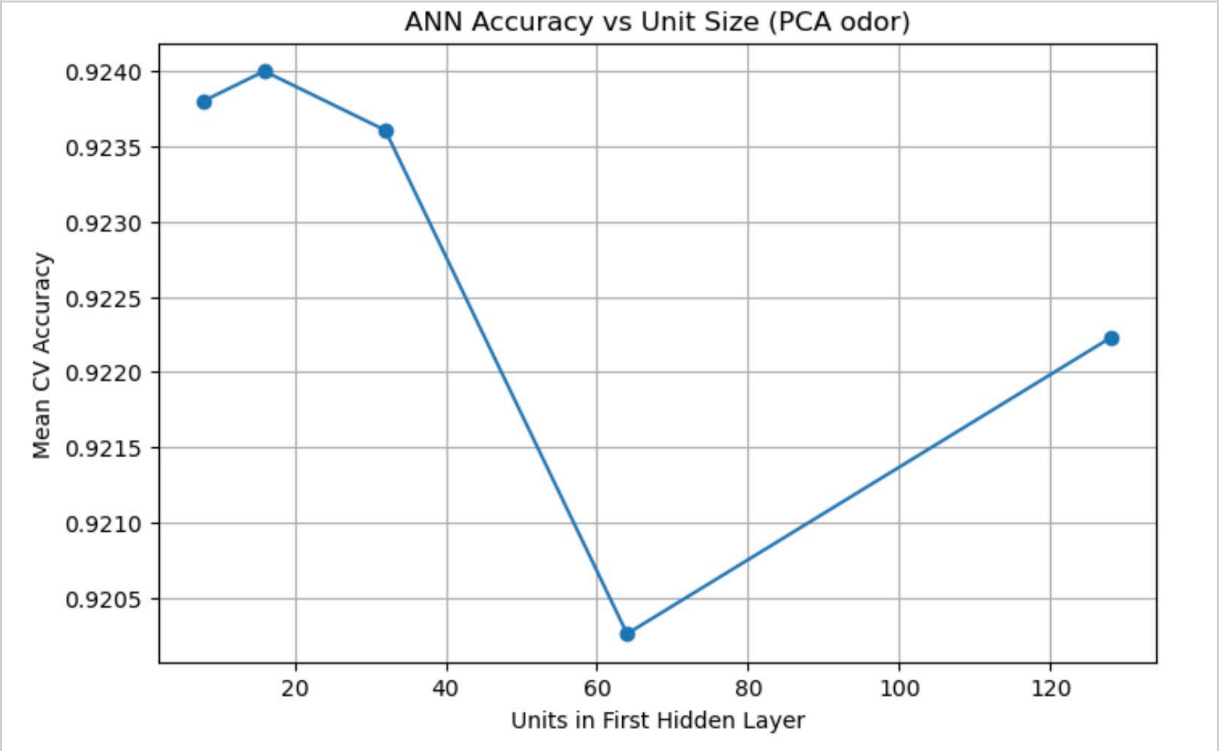
▾ NBPCA_hyper_odor.png Download



▾ **Screenshot 2025-04-20 at 20.28.44.png** Download



▾ **221551745196934_.pic.jpg** Download



Which hyperparameter values did you end up picking for each model? Explain why you picked each value:

For the classification problem to predict odor (PCA used for dimensionality reduction) we chose the following hyperparameters for each model:

1. Decision tree: after dimension reduction for the odor classification

problem, we chose the hyperparameter set (cost complexity = 0.005, tree depth = 7, minimum number of observations that must exist in a terminal leaf node min_n =5) for the decision tree model. This hyperparameter set is chosen due to high accuracy across k = 5 cross validation.

2. Random forest: after dimension reduction for the odor classification problem, we chose the hyperparameter mtry = 8 (number of predictors sampled for splitting at each node) for the random forest model due to its low value in cross entropy loss. The model also performs well under this hyperparameter for other metrics such as area under receiver operating curve of 0.9791736, accuracy of 0.8761583, mean F1-score of 0.7498050, and balanced accuracy of 0.8656778.

3. CatBoost: We selected a tree depth of 2 for the CatBoost model. This decision was based on the 5-fold cross-validation results, where depth = 2 achieved the highest mean accuracy (0.9100) among the tested depths [2, 4, 6, 8, 10]. Despite deeper trees having higher model capacity, they consistently performed worse in cross-validation, likely due to overfitting on the PCA-reduced features.

4. ANN: The model with 16 units achieved the highest mean cross-validation accuracy of 0.9240, so we selected 16 as the optimal number of units. This value is the best-performing one.

5. KNN: after dimension reduction for the poisonous classification problem, we still tuned the hyperparameter k, the number of nearest neighbors. But this time, we chose Euclidean distance as the distance metric instead of Hamming distance as the features after PCA all became continuous. Specifically, we tested k values from 1-150 with interval = 5. We picked k = 46 as the best value as it achieved the highest and perfect accuracy = 0.9167.

6. Naive Bayes Classifier: After dimension reduction for the odor classification problem, we needed to run Gaussian Naive Bayes instead of categorical Naive Bayes as all features were converted to continuous variables. The hyperparameter to tune in this context is the var_smoothing variable (w/ same reasoning as for Classification problem 1). We tested var_smoothing from 1e^-10 to 1e^0, with 11 logarithmically spaced intervals. We found that the best mean CV accuracy was reached at 1e^-4 and 1e^-3, which was 0.9069. We ended up picking the best value to be var_smoothing = 1e^-4 (the smaller value), which was the better choice to ensure numerical stability without overly flattening the class-conditional distributions.

Train your models on all the training data, using the chosen hyperparameter values. What was your performance and runtime for training?

Training

1. Decision tree: after dimension reduction, it takes ~0.343 seconds to train the decision tree model for odor classification problem. Under the hyperparameter set (cost complexity = 0.005, tree depth = 7, minimum number of observations that must exist in a terminal leaf node min_n =5), the decision tree model achieved averaged accuracy of 0.9151373, and averaged area under receiver operating curve of 0.9718212 across validation folds

2. Random forest: after dimension reduction, it takes ~1.76 seconds to train the random forest model for the odor classification problem. Main performance metric: cross entropy loss of 0.2141672. The model performs well under this hyperparameter for other metrics such as area under receiver operating curve of 0.9791736, accuracy of 0.8761583, mean F1-score of 0.7498050, and balanced accuracy of 0.8656778.

3. CatBoost: Using the selected depth = 2, we get training accuracy 0.9309 in 0.20 seconds.

4. ANN: Using the selected unit size of 16, we get the final training accuracy 0.9299 with training time 0.69 seconds.

5. KNN: after dimension reduction, with the best k = 46, it takes KNN 0.001948 seconds to train the whole training data, reaching an accuracy of 0.9305.

6. Naive Bayes Classifier: after dimension reduction, with the best var_smoothing value 1e^-4, it takes NB Classifier 0.001545 seconds to train the whole training data, reaching an accuracy of 0.9128.

Predict the validation data with your trained model.
What was your performance and runtime for testing?

Testing

1. Decision tree: after dimension reduction, it takes ~0.00548 seconds to predict on testing data for decision tree model for odor classification problem. On the testing data, the model achieved accuracy of 0.9168142 and area under receiver operating curve of 0.9681328.

2. Random forest: after dimension reduction, it takes ~0.0170 seconds to predict on testing data for random forest model on for odor classification problem. On the testing data, the model achieved accuracy of 0.8673, 95% confidence interval of (0.8365, 0.8941), no information rate of 0.492, p value [Acc > NIR] < 2.2e-16, kappa of 0.8008.

3. CatBoost: When evaluated on the PCA-reduced test set the model got test accuracy: 0.8991andtest time 0.0053 seconds. There's a slight drop from training accuracy (0.9309 to 0.8991), likely due to the limited expressiveness of a shallow tree depth and information loss from PCA.

4. ANN: The trained ANN achieved a final test accuracy 0.9221 with testing time: 0.0192 seconds.

5. KNN: after dimension reduction, with the best k = 46 it takes KNN 0.015358 seconds to predict the testing data, reaching an accuracy of 0.9168. Specifically, the testing results contain 518 correctly predicted instances and 47 incorrectly predicted instances.

6. Naive Bayes Classifier: after dimension reduction, with the best var_smoothing value 1e^-4, it takes NB Classifier 0.000438 seconds to predict the testing data, reaching an accuracy of 0.8938. Specifically, the testing results contain 505 correctly predicted instances and 60 incorrectly predicted instances.

Q4 Review

4 Points

Compare the two classification problems. Was either one of them "easier" to solve?

Predicting edibility is easier to solve compared to predicting odor. For all our six models, in both original and dimensionality-reduced datasets, the training and testing accuracy outputs are higher for edibility prediction, with most reaching at least 0.99 (except for Random forest testing accuracy before PCA, which is slightly lower(0.9628)). In contrast, odor prediction yielded lower performance, with models achieving training and testing accuracies generally between 0.90 and 0.96. This discrepancy is likely due to the fact that edibility is a binary classification problem (edible vs. poisonous), while odor classification involves nine distinct categories, making it a more complex, multiclass task. Moreover, the decision boundary between "edible" and "poisonous" may be more sharply defined in the feature space, while odor labels could overlap more, reducing separability for classifiers.

Compare the classification models. Was any one of them "easier" to fit?

To compare which model(s) are easier to fit, we targeted the training time with the best hyperparameter(s). We found that for both edibility and odor, out of the six models, KNN and Naive Bayes Classifier consistently have the shortest training time, which were around 0.00x seconds to 0.01 seconds(mostly at the scale of 0.00x seconds). This is possibly due to that KNN has no actual training step other than calculating and storing the training data points and NB Classifier relies on simple statistical estimation, while decision tree, random forest, CatBoost, and ANN involve iterative or ensemble-based procedures, which yield longer training time.

Consider the dimension reduction. What impact did it have?

1. Decision tree: dimensionality reduction caused an increase in its runtime of training and predicting for both classification problems. We did not observe significance change in performance metrics (specificity, sensitivity, kappa) before vs after dimensionality reduction for both classification problems for decision tree.

2. Random forest: it takes random forest less time to train and predict data on both classification problems after reduction. The time decrease is by some decent factor close to a magnitude. We did not see notable changes in performance metrics (accuracy, specificity, area under receiving operating curve) before and after dimensionality reduction.

3. CatBoost: Dimension reduction with PCA had negative effects on predictive performance. This is likely because PCA transforms categorical features into numerical principal components, stripping away the category-specific information that CatBoost excels at using in its split logic. However, PCA led to moderate improvements in training efficiency for both tasks by reducing the number of feature combinations evaluated during tree construction.

4. ANN: For ANN, the impact of PCA was task-dependent. On the simpler class classification task, PCA improved performance, likely by eliminating feature redundancy and noise. However, for the more complex odor classification task, PCA slightly reduced accuracy, possibly due to loss of nuanced input variation. In terms of runtime, PCA had no noticeable effect on training or testing efficiency for either task.

5. KNN: for both problems, dimensionality reduction slightly decreases the testing time, but there are no significant changes for training time and training and testing accuracy (i.e, the changes are very small).

6. Naive Bayes Classifier: for both problems, dimensionality reduction slightly decreases the training time, but there is no significant change for testing time(the magnitude of decrease was very small). However, notably, for both problems, dimensionality reduction decreases the accuracy for both training and testing for more than 0.01. This decrease is more significant for edibility prediction. These changes sugget that dimensionality reduction is not very ideal for NB Classifier, considering the reduction in accuracy (although there is small decrease in time).

In conclusion, there are slight changes in terms of training/testing time and accuracies after PCA dimensionality reduction, but the changes are not consistent across models, and the magnitudes of changes are not very significant.

Consider the hyperparameter tuning. Did the "best" hyperparameter value ever differ between the two classification problems?

1. Decision tree: Yes. The the optimal cost complexity pruning parameter differed. For the poisonous classification task tended to require a smaller cost complexity value , allowing the tree to grow with slightly more detail before being pruned. Meanwhile, the odor classification task benefited from a higher cost complexity pruning parameter; applying more regularization and resulting in simpler trees that were better at generalizing over a more complex, multiclass label set.

2. Random forest: Yes. Predicting whether a mushroom is poisonous required a relatively low value for mtry. This suggests that only a small number of features were consistently important in making accurate predictions for the binary classification task. In contrast, predicting odor, a multiclass problem with 7 classes, benefited from a higher mtry value for random forest. This indicates that more features were useful simultaneously to capture the subtle differences between multiple classes.

3. CatBoost: Yes. For CatBoost, the best hyperparameter for edibility (class) prediction was depth = 2, as it was the first value to achieve perfect accuracy across folds with minimal model complexity. For odor prediction, the best depth was 4, but the accuracies fluctuated irregularly across different depth values.

4. ANN: Yes/No. For ANN, the best hyperparameter for edibility (class) prediction was 64 units in the first hidden layer — the first value that achieved perfect accuracy. For odor prediction, the best unit size was 16 or 32, depending on the trial, but the accuracies varied across unit sizes.

5. KNN: Yes. The best k values(k = 3, k = 5) for edibility prediction were much smaller than for odor prediction (k = 41, k = 46).

6. Naive Bayes Classifier: No. Although the best smoothing values selected for edibility and odor prediction tasks differed, they were all on a very small scale (much smaller than 1). This suggests that for both problems only minimal smoothing was needed to ensure numerical stability, while still allowing the model to rely heavily on the true variance observed in the data. Excessive smoothing (e.g., values closer to 1) led to a notable drop in accuracy.

In conclusion, hyperparameter tuning results generally differed across the two classification problems. Simpler binary tasks like edibility often benefit from smaller models and fewer features, while multiclass problems like odor classification required greater model complexity or broader feature usage to achieve optimal performance.