Project 2 Part B

● Graded

Group

Autumn Xu
Samantha Zheng
Yueping Gu
View or edit group

Total Points

18 / 18 pts

Question 1

Model descriptions

2 / 2 pts

✓ **+ 2 pts** Correct

+ 0 pts Incorrect / Blank

+ 1 pt Number of models less than requirements

+ 1 pt No description of your models

Question 2

Model strengths

2 / 2 pts

✓ **+ 2 pts** Correct

+ 1 pt Partial credit

+ 0 pts Incorrect / Blank

Question 3

Model weaknesses

2 / 2 pts

✓ **+ 2 pts** Correct

+ 1 pt Partial credit

+ 0 pts Incorrect / Blank

Question 4

Hyperparameters

2 / 2 pts

✓ **+ 2 pts** Correct

+ 1 pt Hyperparameters/Models are not enough

+ 0 pts Incorrect/Not provided

Question 5

Function calls **2** / 2 pts

✓ **+ 2 pts** Correct

+ 1 pt Partial Credit

+ 0 pts Incorrect/Not provided

Question 6

Cross validation demonstration **8** / 8 pts

6.1 **Set-up** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect or missing

6.2 **Performance metric** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts missing or incorrect

6.3 **Code** **3** / 3 pts

✓ **+ 3 pts** Correct

+ 0 pts Incorrect or missing.

6.4 **Graph** **3** / 3 pts

✓ **+ 3 pts** Correct

+ 0 pts missing or incorrect

Q1 Model descriptions
2 Points

More details can be found on Canvas.

Briefly define/describe your classification models.

Reminder:
If you're working by yourself, you only need to pick 2 models.
If you're working with a partner, you'll need to pick 4 models.
If you're working in a group of 6, you'll need to pick 6 models.

Ex. Linear regression is a model that predicts a numerical response value from a linear combination of numerical input values

XGBoost is a gradient boosting model that builds an ensemble of shallow decision trees sequentially, where each new tree corrects errors made by previous ones to minimize classification loss.

CatBoost is a gradient boosting model that builds symmetrical decision trees and natively handles categorical features using ordered boosting, improving performance on datasets with many discrete variables.

Random Forest is an ensemble learning model that constructs multiple decision trees using randomly sampled data and features, combining their predictions through majority voting to improve classification accuracy and reduce overfitting.

Decision Tree is a predictive model that learns hierarchical decision rules from data by recursively splitting features into branches, forming a tree structure that maps inputs to class labels.

KNN is a model that finds the k neighbors (with k determined based on performance or theoretical considerations) that are closest to the test sample according to a specified distance metric, and then classifies the test sample based on the majority label among those k nearest neighbors.

Naive Bayes Model is a generative probabilistic classification model that predicts the class of a test sample by applying Baye's theorem under the assumption that features are conditionally independent given the class label. The model estimates the posterior probability of each class based on the likelihood of the observed features and assigns the test sample to the class with the highest posterior probability.

OR

Upload your answers here:

No files uploaded

Q2 Model strengths

2 Points

Give two strengths for each of your models by comparing it to an alternative.

Ex. Linear regression is better than ANN because it's easier to interpret the parameters

ANN can model nonlinear feature interactions better than logistic regression or decision trees, so it would be more accurate for complicated data patterns. It supports multi-class classification, so it would be more flexible than simpler models like Naïve Bayes.

CatBoost is better than Random Forest when working with categorical features, as it natively supports categorical inputs without the need for manual encoding. CatBoost is more resistant to overfitting than traditional gradient boosting models because it uses ordered boosting.

Random forest performs well with large numbers of input features and doesn’t suffer from the curse of dimensionality as much as k- nearest neighbors, which tends to degrade in high-dimensional spaces. Random forest naturally reduces overfitting through bagging and random feature selection across trees, while support vector machines can over fit with insufficient tuning of kernel and regularization.

Decision trees provide a clear visual and logical explanation of how predictions are made, unlike artificial neural networks, that are more or less black-box models. Decision trees don’t require standardized or normalized features to perform well, unlike logistic regression which can be sensitive to the scale of numeric inputs.

KNN has an advantage over decision trees that it is very intuitive in terms of characterizing the similarity between samples, and it can model more flexible, non-linear decision boundaries without being limited/according to axis-aligned splits(as decision trees does), allowing it to capture complex patterns in the data intuitively and naturally. As KNN does not build an explicit model during training, this makes it simpler to implement than quicker to set up, where decision trees require recursive partitioning and pruning, which takes more time and computing power.

Naive Bayes Classifier, compared to CatBoost, is a much faster method with minimal computational resources, making it a good approach for sparse datasets or tasks like text classification. Also, it provides intuitive probabilistic outputs for how each feature contributes to classification decisions, while CatBoost is not that intuitive as it is a complex ensemble model.

OR

Upload your answers here:

No files uploaded

Q3 Model weaknesses

2 Points

Give two weaknesses for each of your models by comparing it to an alternative.

Ex. Linear regression is unable to explain non-linear relationships, unlike ANN

Random forests can struggle with extremely sparse datasets, where Naive Bayes models are often faster and more effective due to their probabilistic nature. Because random forests aggregate results from many decision trees, they can be computationally expensive at prediction time, unlike logistic regression which is much faster and lightweight.

Decision trees can be highly sensitive to small changes in the dataset — a small change can drastically alter the structure of the tree. In contrast, k-nearest neighbors tends to be more stable under such fluctuation. Decision trees may struggle to capture complex, non-linear relationships in the data, especially when interactions between features are subtle, more suitable for artificial neural network to model intricate patterns and hierarchical feature representations.

KNN is considered inferior to decision trees in terms of 1. KNN requires more pre-processing, for example, characterizing the distances for categorical features, but decision trees only requires minimal pre-processing. 2. KNN does not define the real relationship between features and response variables, as it only characterizes the distances between samples (not features), but decision trees does.

Naive Bayes Classifier is considered inferior to ANN in terms of 1. it assumes conditional independence between features given the class label, which is actually rare in real cases. ANN, in contrast, can capture these inter-dependencies and interactions between features. 2. NB Classifier is a linear classifier and is not good at modeling non-linear relationships, while ANN is highly flexible and can learn non-linear and hierarchical structures.

CatBoost is less interpretable than a decision tree because it is an ensemble of many decision trees. A decision tree gives human-readable decision flows. It is slower than probabalistic models like Naive Bayes or linear models like linear regression because it requires iterative training across multiple rounds.

ANN requires more tuning than random forest because it depends heavily on the tuning architecture, including the number of layers and units. Random Forests work with fewer hyperparameters. It is also harder to interpret than logistic regression because it does not expose the relationships in a human-readable form.

OR

Upload your answers here:

No files uploaded

Q4 Hyperparameters
2 Points

Pick a hyperparameter for each of your models. Make sure to include what values you'll be testing.

Ex. I'll use an l2-norm regularization penalty for linear regression, with a lambda weight on the penalty as the hyperparameter. I'll test values lambda = [0.1, 0.3, 0.5, 0.7, 0.9]

Decision tree relies on hyperparameters cost_complexity (for the the cost/complexity parameter), tree_depth (for maximum depth of the tree) and min_n (for the minimum number of data points in a node that are required for the node to be split further). We tested the following values cost_complexity = [0.001, 0.005, 0.01, 0.05, 0.1], tree_depth = [3, 5, 7, 9, 11], min_n = [5, 10, 15, 20].

Random forest depends on the hyperparameter mtry, the minimum number of predictors sampled for splitting at each node. We tested the values mtry = [1, 2, 4, 6, 8, 10, 12, 15, 20]

KNN depends on the hyperparameter k, the number of nearest neighbors. For our first response variable "edibility", I tested the values from 1-50 with interval 2. For our second response variable "odor", I tested the values from 1-150 with interval 5.

For Naive Bayes Classifier, I chose to tune the hyperparameter alpha, which controls smoothing to avoid zero-probability issues and acts like a form of regularization which helps prevent the predictions towards extremes. I tested the values [0.001, 0.01, 0.1, 0.5, 1.0, 2.0, 5.0] for both response variables.

CatBoost depends on the depth of the trees as a hyperparameter. It represents the maximum depth of each tree in the boosting process. I tested the values [4,6,8,10,12].

The ANN depends on units in the first hidden layer as the hyperparameter. It adjusts the model capacity in the initial layer. I tested the values [8,16,32, 64, 128]. More nodes allow the model to learn more complex patterns, but require more training data and regularization.

OR

Upload your answers here:

No files uploaded

Q5 Function calls

2 Points

Give the function call for each of your models in the programming language of your choice.

Ex. L2 regularized linear regression in Mathematica can be called by:
p = Predict[dataInput -> dataResponse,
  Method -> {"LinearRegression", "L2Regularization" -> 1}]

1. Decision tree in R:
```
tree_spec <- decision_tree(
  cost_complexity = cp,
  tree_depth = depth,
  min_n = minn
 ) %>%
  set_engine("rpart") %>%
  set_mode("classification")
tree_fit <- tree_spec %>%
  fit(odor ~ ., data = train2)
pred_class <- predict(tree_fit, test2, type = "class")
pred_prob  <- predict(tree_fit, test2, type = "prob")
```

2. Random forest in R:
```
rf_model <- randomForest(poisonous ~ ., data = train1, importance = TRUE,
proximity = TRUE, ntree = 500)
predictions <- predict(rf_model, newdata = test1)
```

3. CatBoost in Python:
```
model = CatBoostClassifier(
    depth=6,
    iterations=100,
    learning_rate=0.1,
    loss_function='Logloss',
    verbose=0
)
model.fit(X1_train, y1_train, cat_features=cat_features_1)
```

4. ANN in Python:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential([
    Dense(64, activation='relu', input_shape=(X1_train_enc.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=
['accuracy'])
model.fit(X1_train_enc, y1_train_enc, epochs=10, batch_size=32,
validation_split=0.2)
```

5. KNN in Python (sklearn package):

```
X1_knn = KNeighborsClassifier(n_neighbors=k, metric='hamming')
X1_knn.fit(X1_train_encoded, y1_train)
y1_test_pred = X1_knn.predict(X1_test_encoded)
```

6. Naive Bayes Classifier (sklearn package, features were encoded using ordinal encoding, and the response variable was encoded using label encoding):

```
NB1 = CategoricalNB(alpha=0.001,
min_categories=X1_n_categories_per_column)
NB1.fit(X1_train_encoded, y1_train_encoded)
y_pred_NB1 = NB1.predict(X1_test_encoded)
```

OR

Upload your answers here:

No files uploaded

Q6 Cross validation demonstration
8 Points

Q6.1 Set-up
1 Point

Which model and dataset are you using?

*Per the project description file, our understanding is that this part aims to test our ability to run cross-validation (with actual code), so we are giving one example here for one specific response variable(we have 2 in total) and one specific model(we have 6 in total).

Model: KNN
Dataset: Our data set has 5644 instances with 23 variables. We predict Odor(response variable) with the remaining 22 variables(features).

Q6.2 Performance metric
1 Point

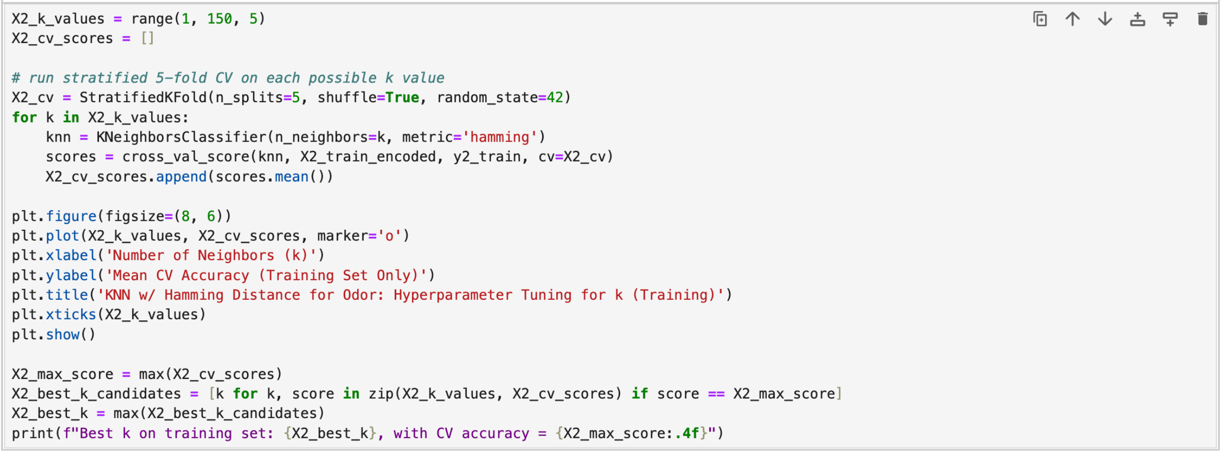
What metric will you be using to evaluate your chosen model?

We used use accuracy as the primary evaluation metric.

Q6.3 Code
3 Points

Upload your code that runs 5-fold cross validation to test hyperparameter values for your chosen model on the chosen dataset:

Project2_BQ6-3.png Download

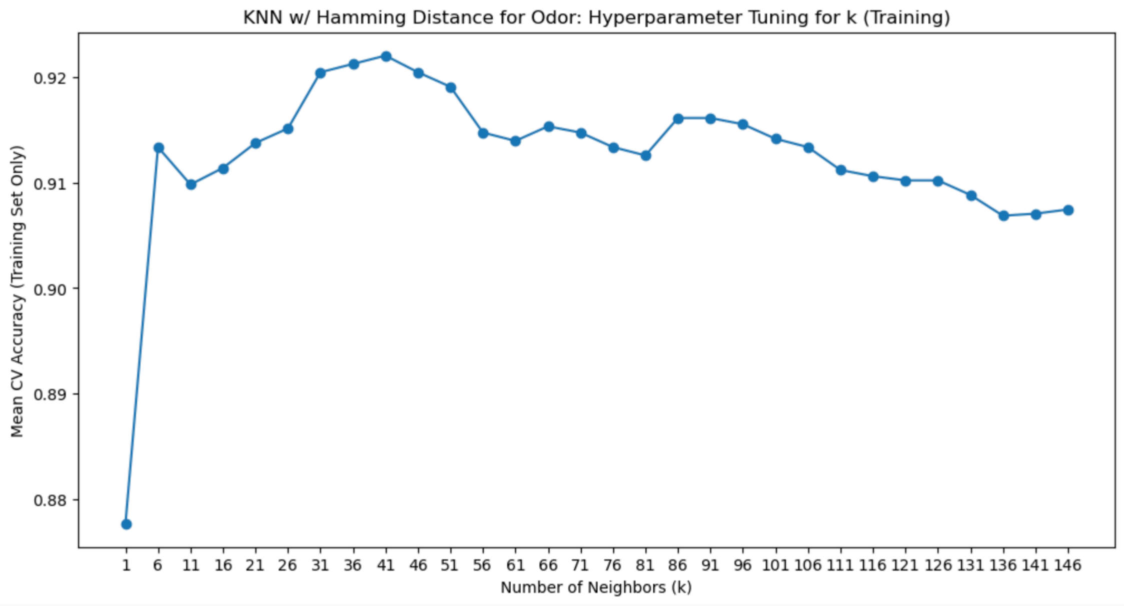


```
X2_k_values = range(1, 150, 5)
X2_cv_scores = []

# run stratified 5-fold CV on each possible k value
X2_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
for k in X2_k_values:
    knn = KNeighborsClassifier(n_neighbors=k, metric='hamming')
    scores = cross_val_score(knn, X2_train_encoded, y2_train, cv=X2_cv)
    X2_cv_scores.append(scores.mean())

plt.figure(figsize=(8, 6))
plt.plot(X2_k_values, X2_cv_scores, marker='o')
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Mean CV Accuracy (Training Set Only)')
plt.title('KNN w/ Hamming Distance for Odor: Hyperparameter Tuning for k (Training)')
plt.xticks(X2_k_values)
plt.show()

X2_max_score = max(X2_cv_scores)
X2_best_k_candidates = [k for k, score in zip(X2_k_values, X2_cv_scores) if score == X2_max_score]
X2_best_k = max(X2_best_k_candidates)
print(f"Best k on training set: {X2_best_k}, with CV accuracy = {X2_max_score:.4f}")
```

Q6.4 Graph
3 Points

Upload a graph that illustrates the results of your 5-fold cross validation:

Project2_BQ6-4.png Download



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