### **Guideline**

Preprocessing: MaldiDataset class

Models using: Naive Mean Distance and KNN

Accuracy: correctly predicted labels over the total number of labels

Experiment:

* Tune hyperparameters
* Try different preprocessing combinations and use cross-validation

Things to add:

* Feature Extraction/Selection (PCA, LASSO, mRMR)
* Imbalance (SMOTE, VAEs/GANs)
* Hyperparameter Tuning: Use GridSearchCV/RandomizedSearchCV to optimize
* Different Models: neural networks, decision trees, clustering
* Use SHAP or LIME to interpret complex models
* ROC and AUC to evaluate

Questions:

* Should we add new classifiers? How complex should the classifier be? Or just improve the given ones?
* We’re going to use explainability tools to understand the results, do you recommend anyone? We have SHAP or LIME

Features part more than models

Binings

Gaussian, mvl

DONE TO DO

### \*\*1. Preprocessing Focus\*\*

1. \*\*Binning\*\*

- Experiment with different \*\*bin widths\*\* (`step` parameter in `Binner`):

- Smaller steps → \*\*higher resolution\*\* features but larger dimensionality.

- Larger steps → \*\*lower resolution\*\* but faster computation.

- Test steps like \*\*3, 6, 10\*\* (already `6` is used).

2. \*\*Gaussian Smoothing\*\*

- Add Gaussian smoothing in addition to the Savitzky-Golay filter. This can smooth noise more effectively.

3. \*\*mvl (Moving Variance/Local)\*\*:

- Calculate local variance or smoothness of features over a small window size.

- Use this to \*\*derive new features\*\*:

---

### \*\*2. Feature Extraction and Selection\*\*

1. \*\*Principal Component Analysis (PCA)\*\*

- Reduce the \*\*dimensionality\*\* of the spectra while retaining variance:

2. \*\*LASSO for Feature Selection\*\*

- Use LASSO (L1-regularized regression) to identify the most \*\*important peaks\*\*:

---

### \*\*3. Handling Class Imbalance\*\*

1. \*\*SMOTE (Synthetic Minority Oversampling Technique)\*\*

- Apply SMOTE to balance the dataset:

```python

from imblearn.over\_sampling import SMOTE

smote = SMOTE()

X\_resampled, y\_resampled = smote.fit\_resample(X\_train, y\_train)

```

2. \*\*Advanced Option\*\*: Test generative models like \*\*VAEs/GANs\*\* to generate synthetic spectra for underrepresented fungi species.

---

### \*\*4. Model Tuning\*\*

1. \*\*Hyperparameter Tuning with Cross-Validation\*\*

- Use `GridSearchCV` or `RandomizedSearchCV` to tune `k` for KNN and other parameters:

```python

from sklearn.model\_selection import GridSearchCV

params = {'n\_neighbors': [3, 5, 7, 9]}

grid\_search = GridSearchCV(KNeighborsClassifier(), param\_grid=params, cv=5)

grid\_search.fit(X\_train, y\_train)

print(grid\_search.best\_params\_)

```

2. \*\*Retain Simplicity\*\*: Improve Naive Classifier and KNN rather than adding complex models like neural networks.

---

### \*\*5. Explainability Tools\*\*

1. \*\*SHAP\*\* (recommended)

- SHAP can explain which features (spectra peaks) contribute most to predictions:

```python

import shap

explainer = shap.KernelExplainer(knn.predict, X\_train\_sampled)

shap\_values = explainer.shap\_values(X\_test\_sampled)

shap.summary\_plot(shap\_values, X\_test\_sampled)

```

2. \*\*LIME\*\* (alternative):

- Use LIME to interpret predictions for individual spectra.

---

### \*\*6. Metrics for Evaluation\*\*

1. \*\*Global Accuracy\*\* (baseline comparison).

2. \*\*Accuracy per Class\*\*: Use bar charts.

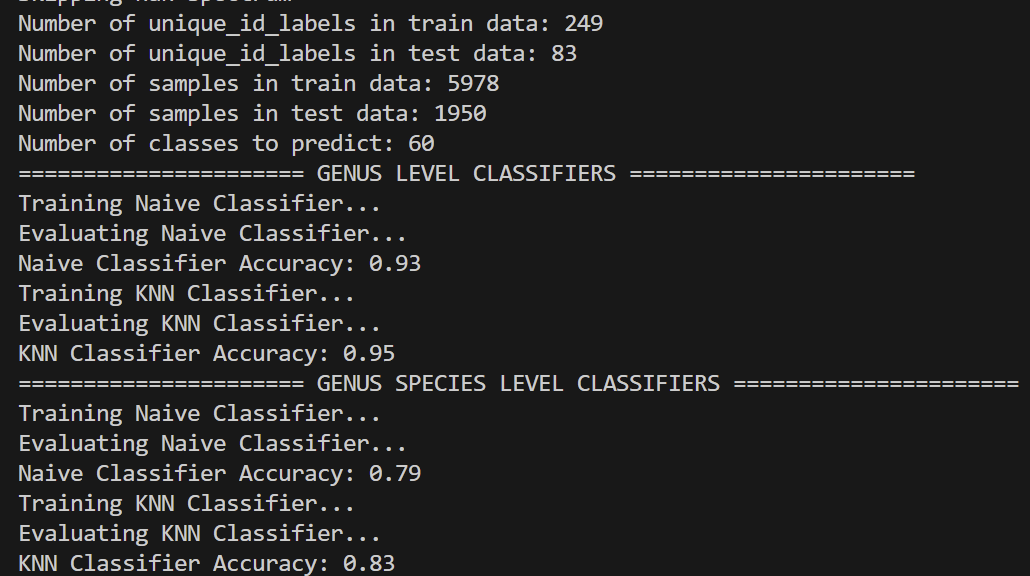
3. \*\*ROC-AUC Curves\*\*: Evaluate classifier performance for each species.

We tried this but we didn’t save the code, but some outputs are in the images file

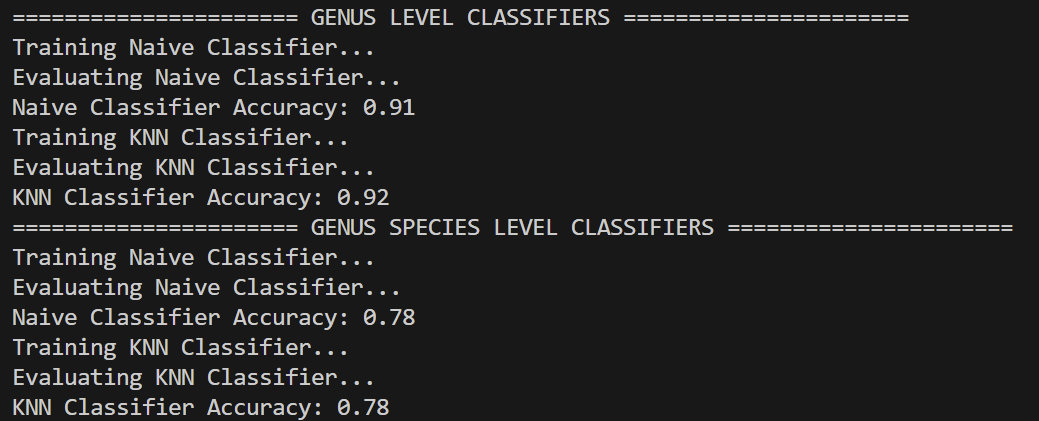
4. \*\*Confusion Matrix\*\*: Visualize misclassifications.

## Step 1. Preprocessing Improvements

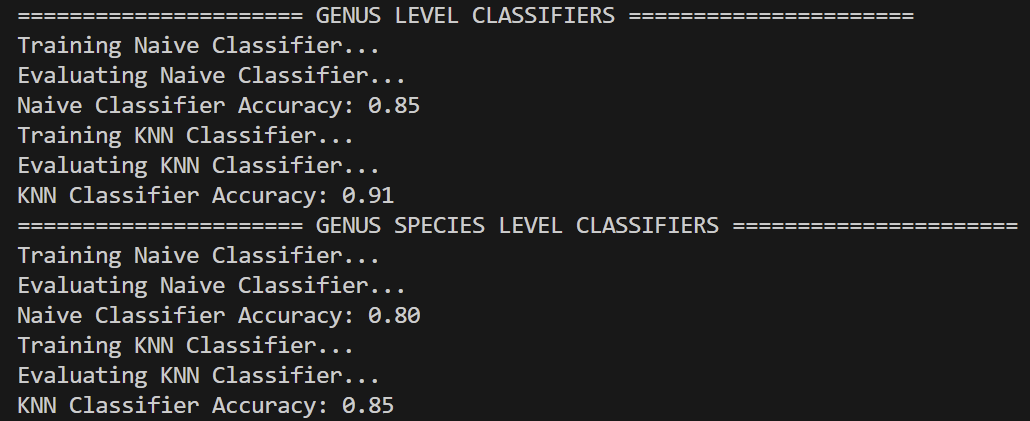
Baseline (bins=6)



Bins = 3

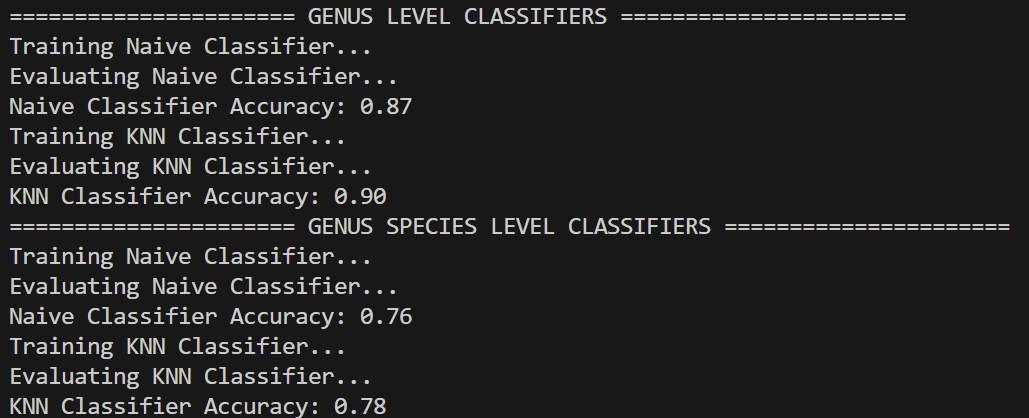


Bins = 10

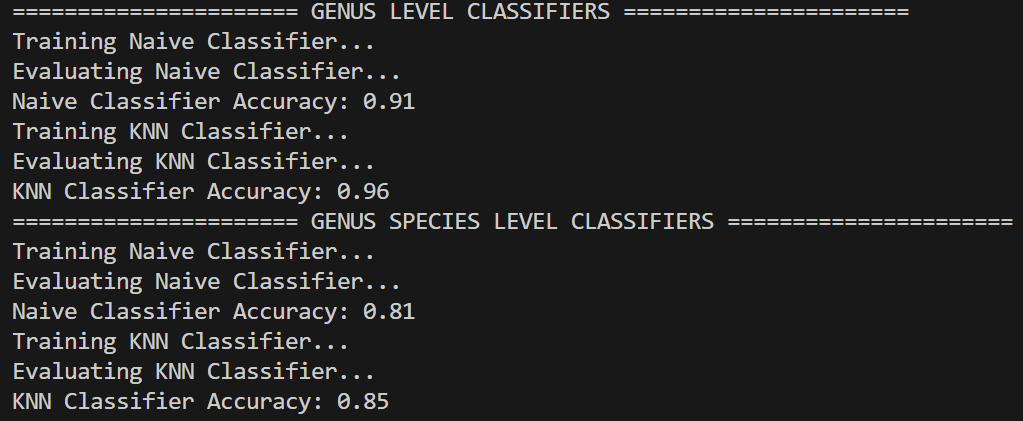


**Best bins n\_step=6**

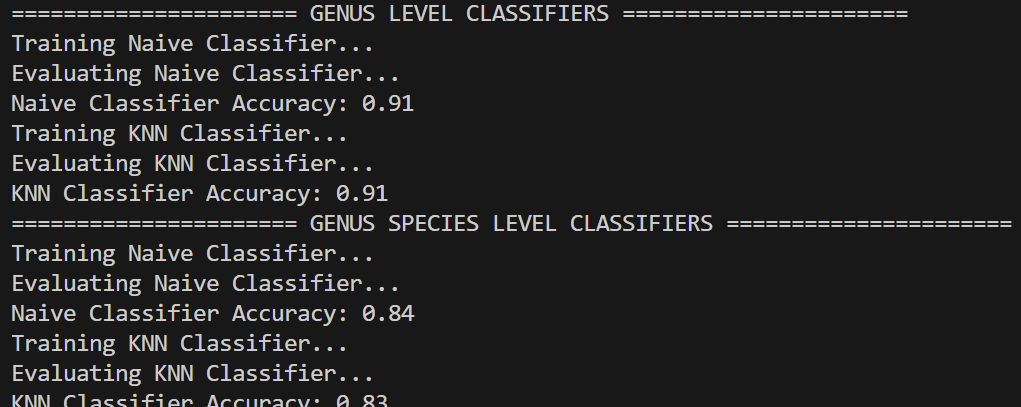
Gaussian smoothing 1.5



Gaussian smoothing 1

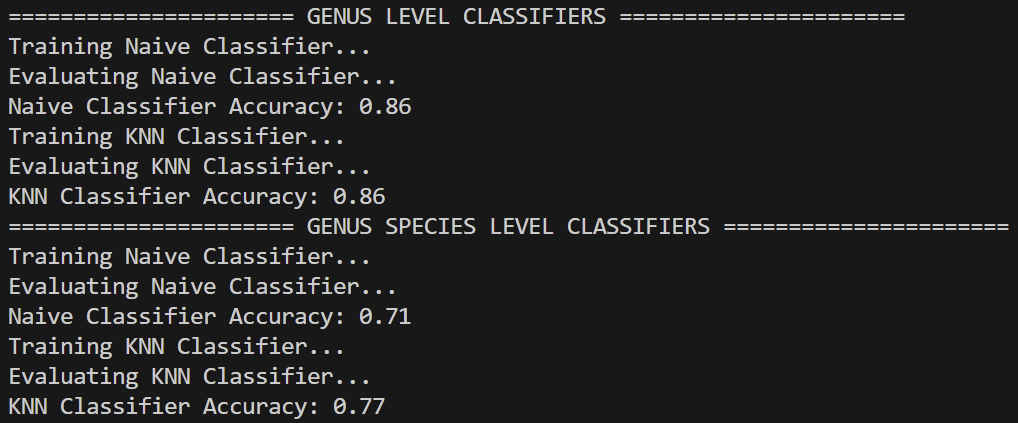


Gaussian smoothing 2

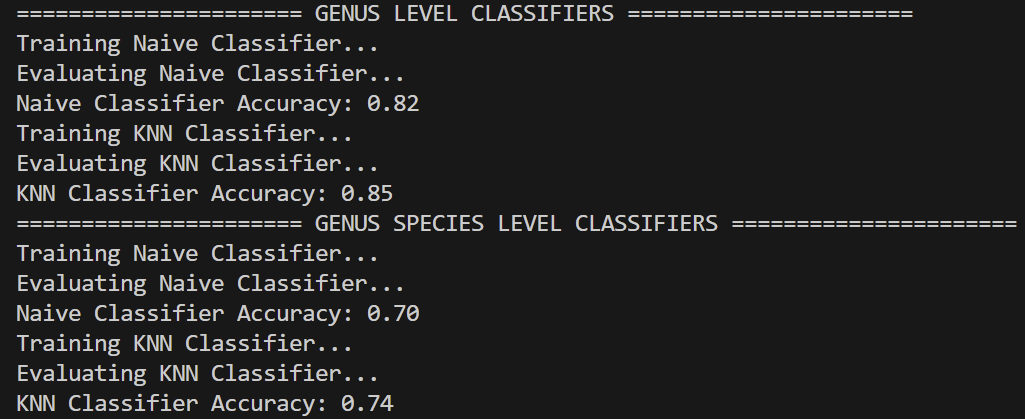


**Best sigma = 1**

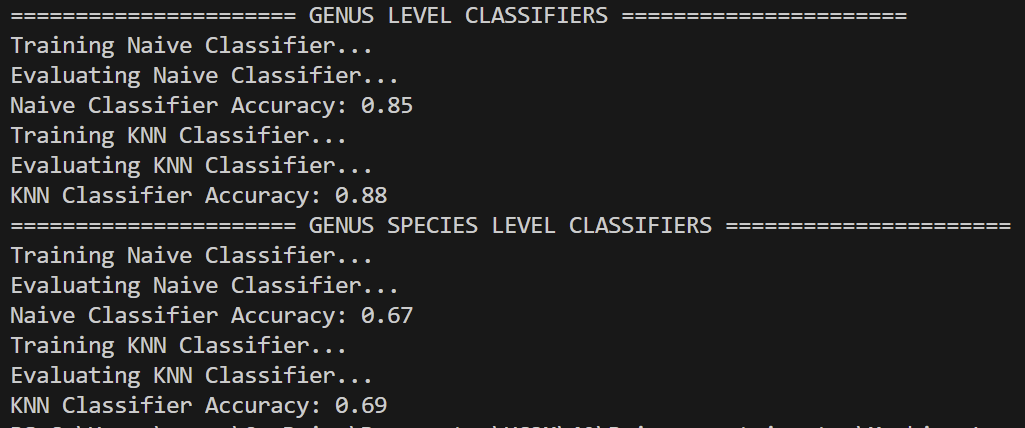
Moving Variance window = 20



Window = 10



Window = 30



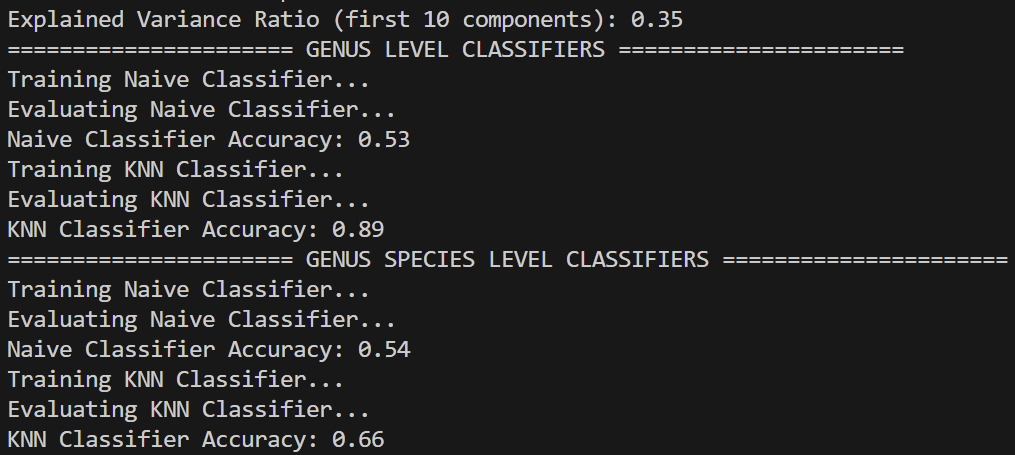
**Best without moving variance** (add complexity without meaningful gains)

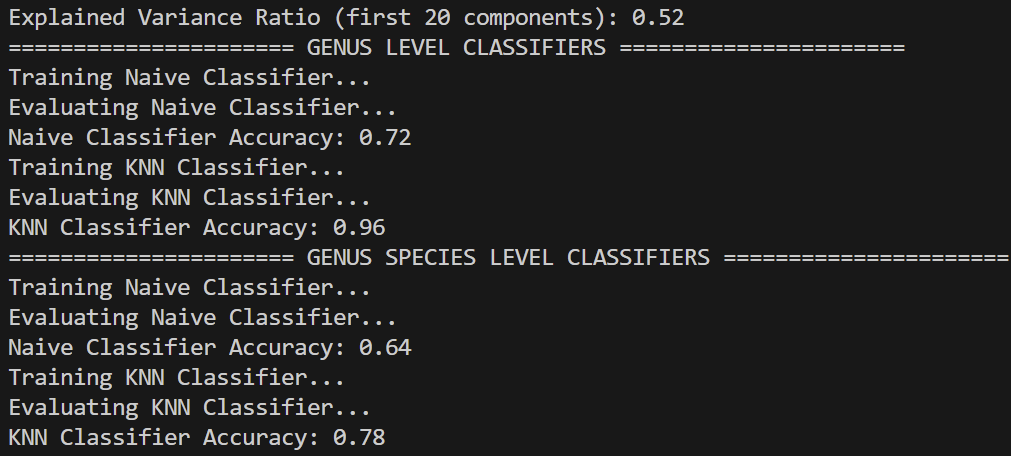
**Best bin n\_step = 3, gaussian sigma = 1**

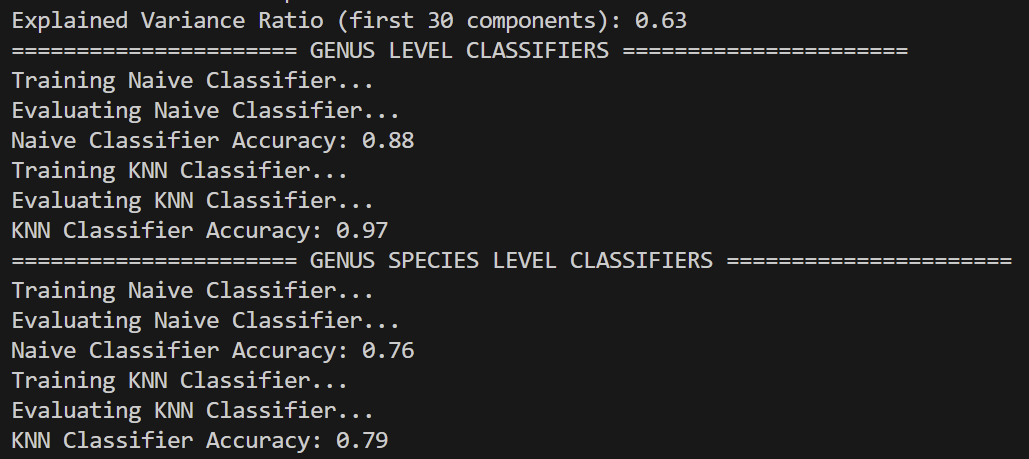
## Step 2: Feature Extraction and Selection

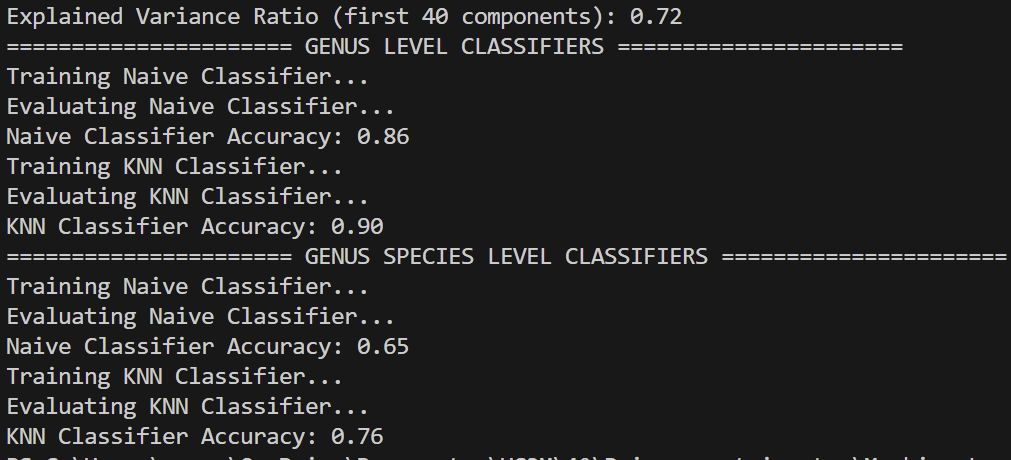
PCA

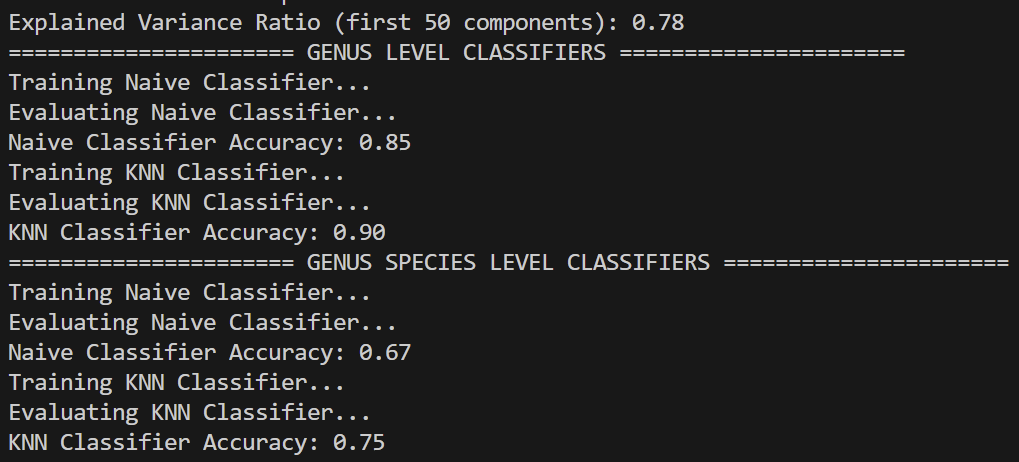
N\_components (10, 20, 30, 40, 50, 70, 100, 150)

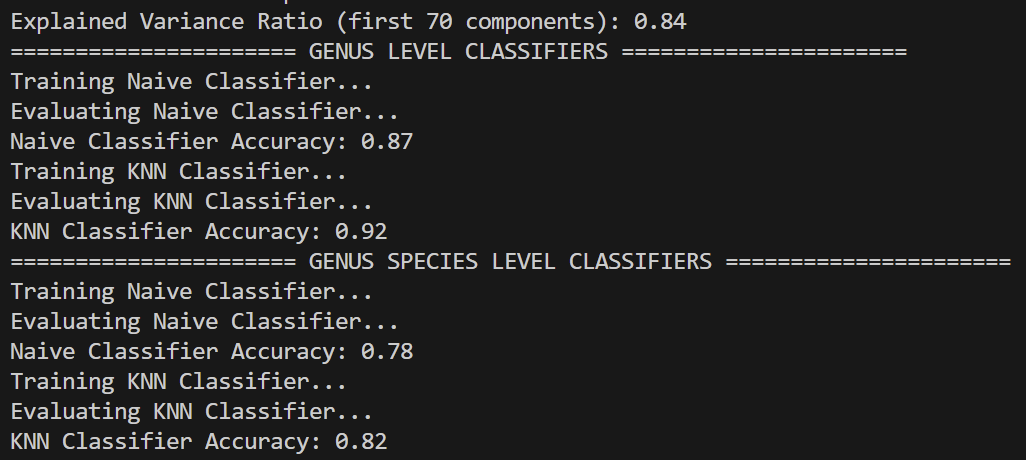


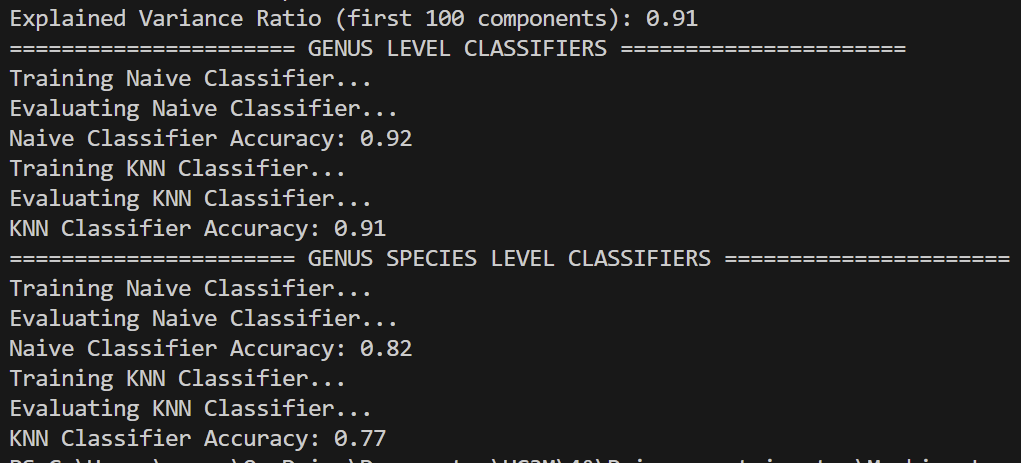


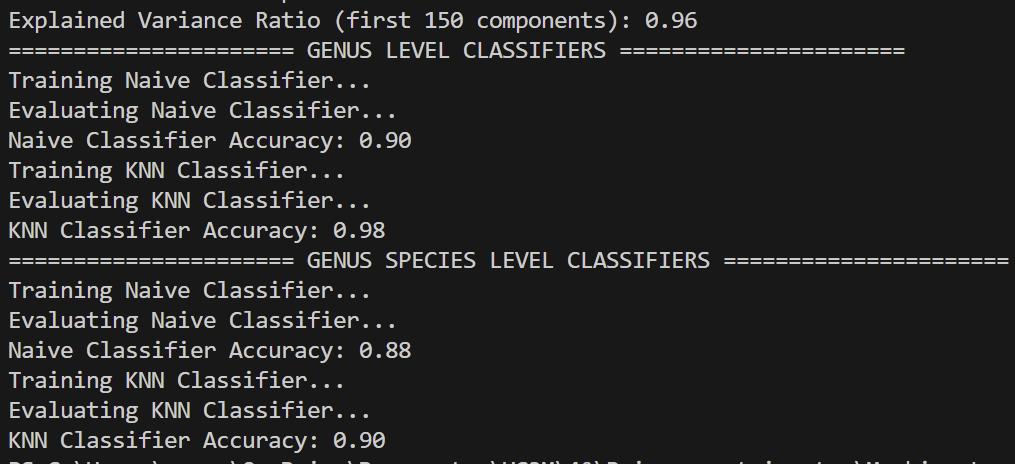












150 components achieves high accuracy but at the cost of:

* Increased dimensionality.
* Higher computational burden.
* Potential overfitting to test data.

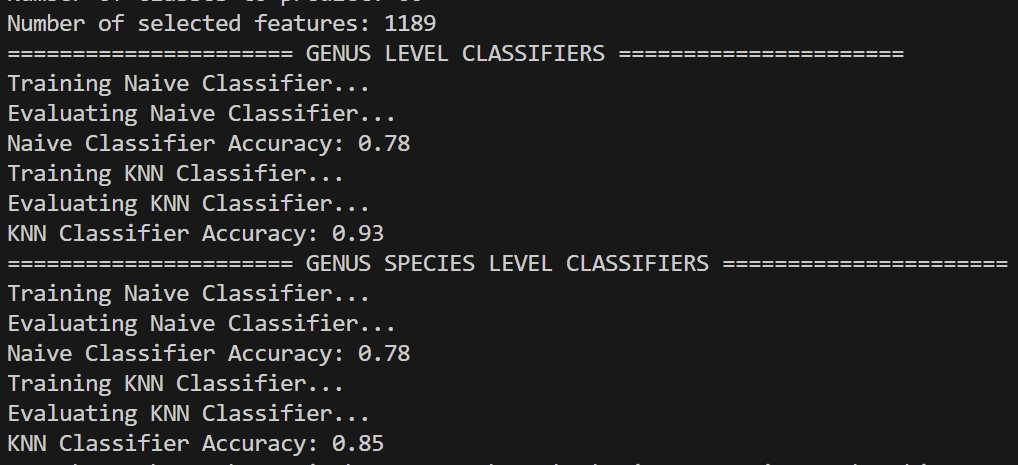
A mid-range value (70 components) provides a balance:

* High accuracy (~0.92-0.94 for genus).
* Lower dimensions = simpler, faster models.
* Generalizes better with less risk of overfitting.

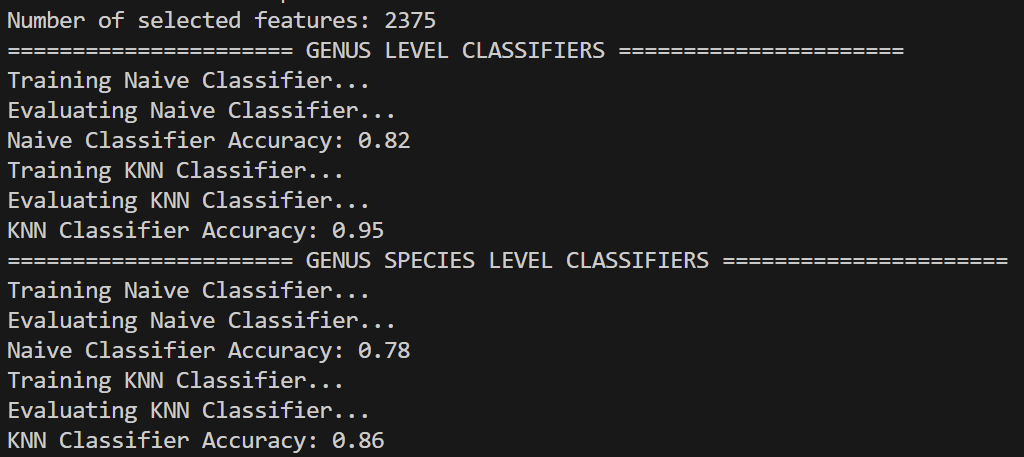
Best accuracy without caring about computational cost 150, otherwise 70

LASSO

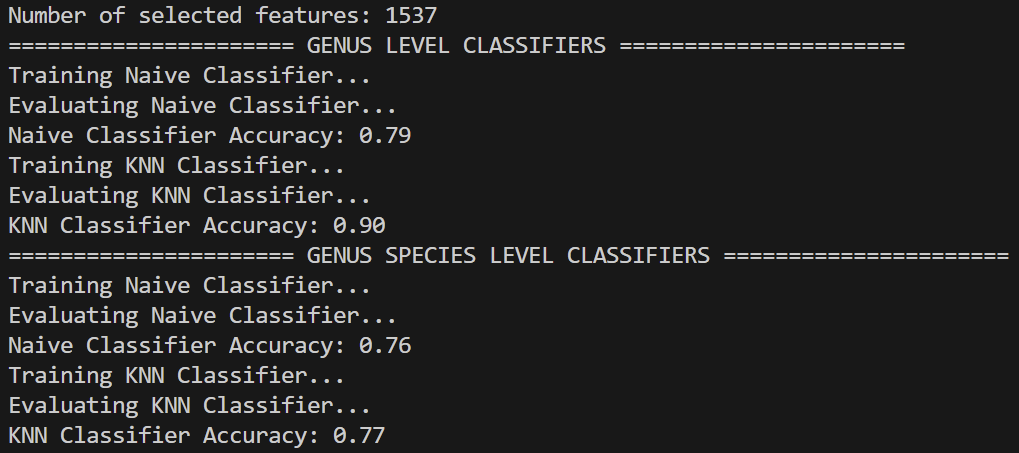
alpha=0.01



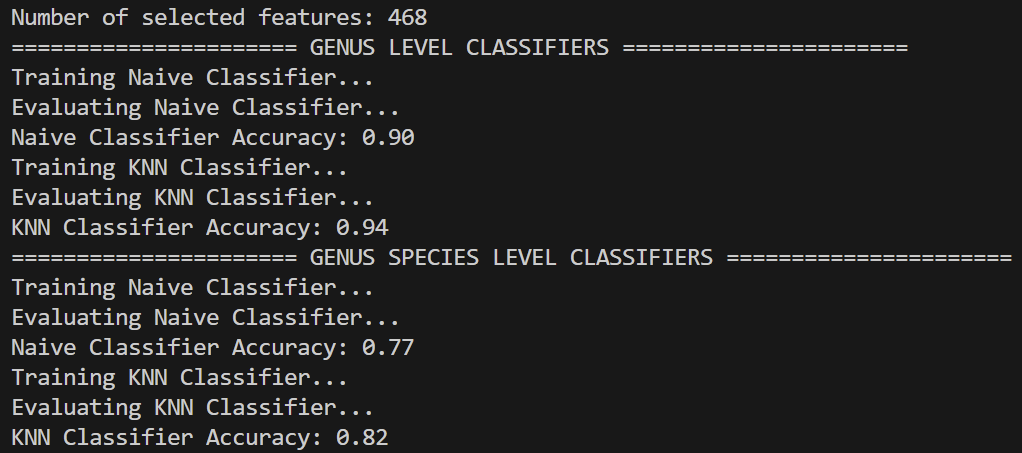
alpha=0.001



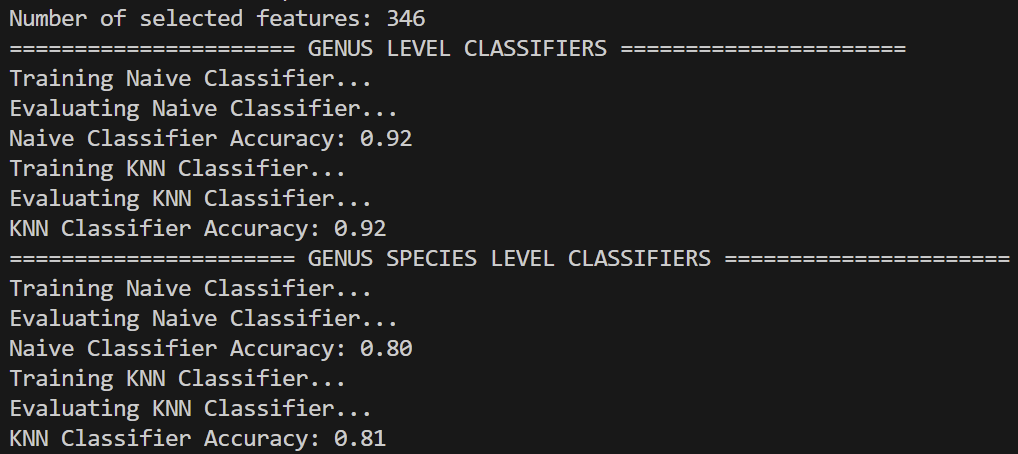
alpha=0.005



alpha=0.05



alpha=0.1



**Alpha = 0.05 is the best**, but without it is still better, so **baseline** (no feature selection) for maximum **performance** and **LASSO** for **interpretability**, it reduces dimensionality while keeping accuracy close to the baseline.

PCA + LASSO

Testing n\_components=10, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.46

KNN Classifier Accuracy: 0.68

Testing n\_components=10, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.46

KNN Classifier Accuracy: 0.68

Testing n\_components=10, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.45

KNN Classifier Accuracy: 0.68

Testing n\_components=10, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.46

KNN Classifier Accuracy: 0.68

Testing n\_components=10, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.46

KNN Classifier Accuracy: 0.68

Testing n\_components=20, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.57

KNN Classifier Accuracy: 0.78

Testing n\_components=20, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.55

KNN Classifier Accuracy: 0.77

Testing n\_components=20, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.57

KNN Classifier Accuracy: 0.77

Testing n\_components=20, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.57

KNN Classifier Accuracy: 0.77

Testing n\_components=20, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.57

KNN Classifier Accuracy: 0.77

Testing n\_components=30, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.63

KNN Classifier Accuracy: 0.79

Testing n\_components=30, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.63

KNN Classifier Accuracy: 0.79

Testing n\_components=30, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.63

KNN Classifier Accuracy: 0.79

Testing n\_components=30, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.63

KNN Classifier Accuracy: 0.79

Testing n\_components=30, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.63

KNN Classifier Accuracy: 0.79

Testing n\_components=40, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.64

KNN Classifier Accuracy: 0.80

Testing n\_components=40, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.64

KNN Classifier Accuracy: 0.80

Testing n\_components=40, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.64

KNN Classifier Accuracy: 0.80

Testing n\_components=40, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.65

KNN Classifier Accuracy: 0.81

Testing n\_components=40, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.64

KNN Classifier Accuracy: 0.80

Testing n\_components=50, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.67

KNN Classifier Accuracy: 0.79

Testing n\_components=50, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.67

KNN Classifier Accuracy: 0.79

Testing n\_components=50, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.68

KNN Classifier Accuracy: 0.79

Testing n\_components=50, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.67

KNN Classifier Accuracy: 0.79

Testing n\_components=50, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.67

KNN Classifier Accuracy: 0.79

Testing n\_components=70, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.68

KNN Classifier Accuracy: 0.81

Testing n\_components=70, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.68

KNN Classifier Accuracy: 0.81

Testing n\_components=70, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.68

KNN Classifier Accuracy: 0.81

Testing n\_components=70, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.68

KNN Classifier Accuracy: 0.81

Testing n\_components=70, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.68

KNN Classifier Accuracy: 0.81

Testing n\_components=100, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.70

KNN Classifier Accuracy: 0.81

Testing n\_components=100, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.71

KNN Classifier Accuracy: 0.81

Testing n\_components=100, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.71

KNN Classifier Accuracy: 0.81

Testing n\_components=100, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.71

KNN Classifier Accuracy: 0.81

Testing n\_components=100, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.71

KNN Classifier Accuracy: 0.81

Testing n\_components=150, alpha=0.1...

Training classifiers...

Naive Classifier Accuracy: 0.73

KNN Classifier Accuracy: 0.83

Testing n\_components=150, alpha=0.05...

Training classifiers...

Naive Classifier Accuracy: 0.73

KNN Classifier Accuracy: 0.83

Testing n\_components=150, alpha=0.01...

Training classifiers...

Naive Classifier Accuracy: 0.73

KNN Classifier Accuracy: 0.83

Testing n\_components=150, alpha=0.005...

Training classifiers...

Naive Classifier Accuracy: 0.73

KNN Classifier Accuracy: 0.82

Testing n\_components=150, alpha=0.001...

Training classifiers...

Naive Classifier Accuracy: 0.73

KNN Classifier Accuracy: 0.83

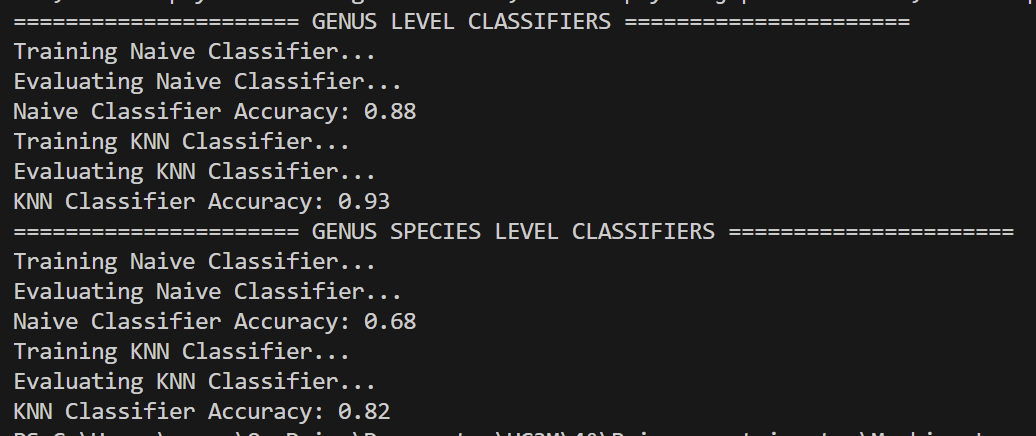
Conclusion: Within each n\_components value, changing alpha does not significantly affect the classifier accuracy. This indicates that LASSO is not effectively contributing to feature selection post-PCA, or the selected features are not influencing the classifiers differently. After PCA, features are already uncorrelated and reduced in dimensionality, so LASSO might not be zeroing out many coefficients.

**LASSO has no impact, so we’ll not use it**

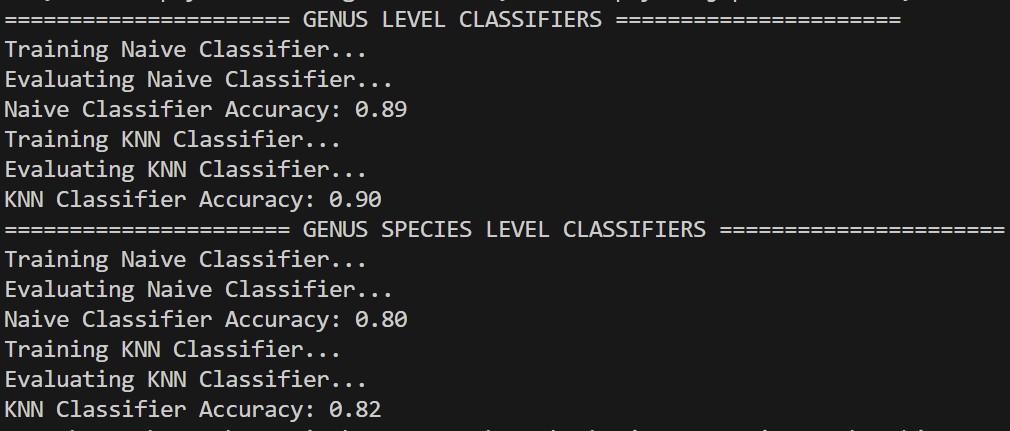
## Step 3: Handling Class Imbalance

SMOTE needs at least 2 samples to generate more

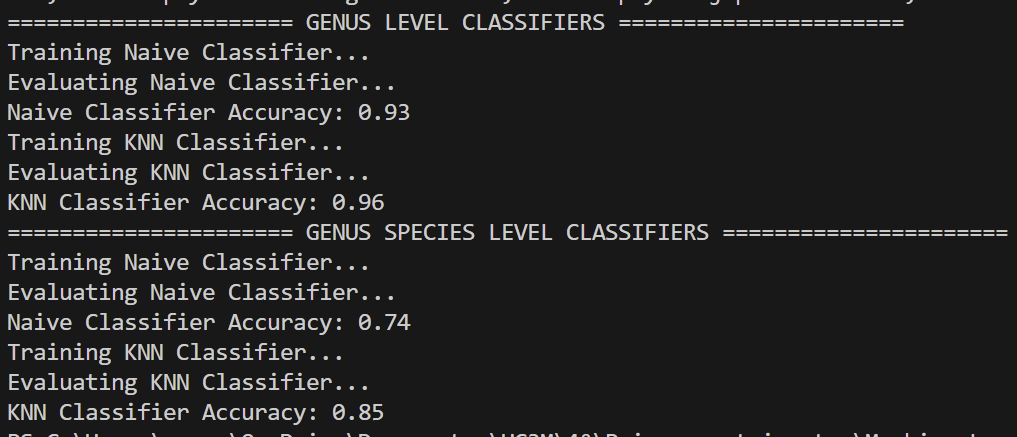
Components 70 + SMOTE



Components 150 + SMOTE



SMOTE in minority classes

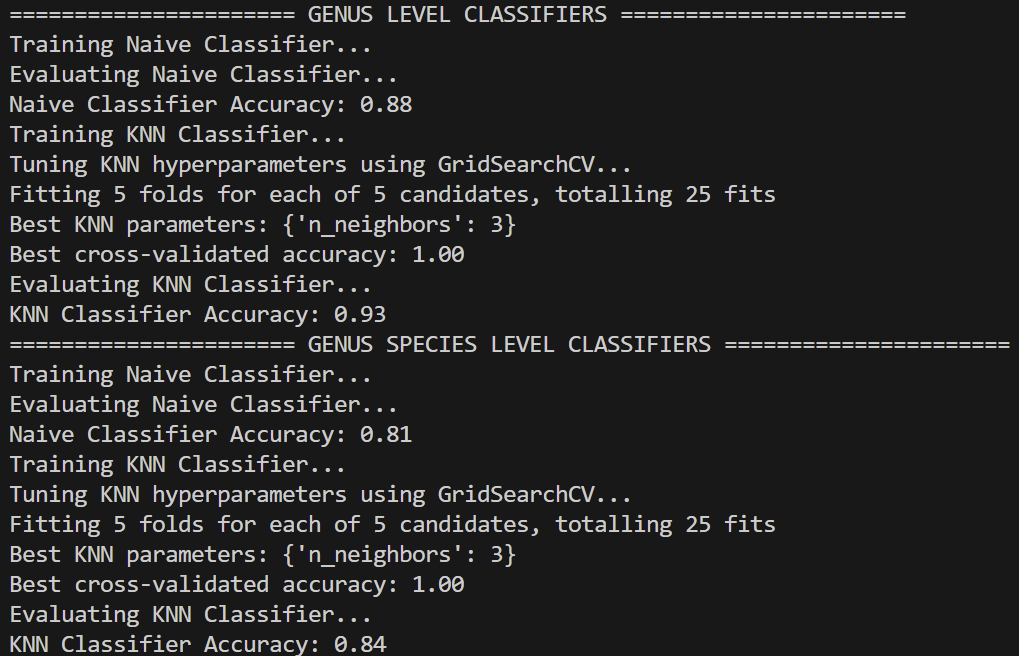


VAEs or GANs

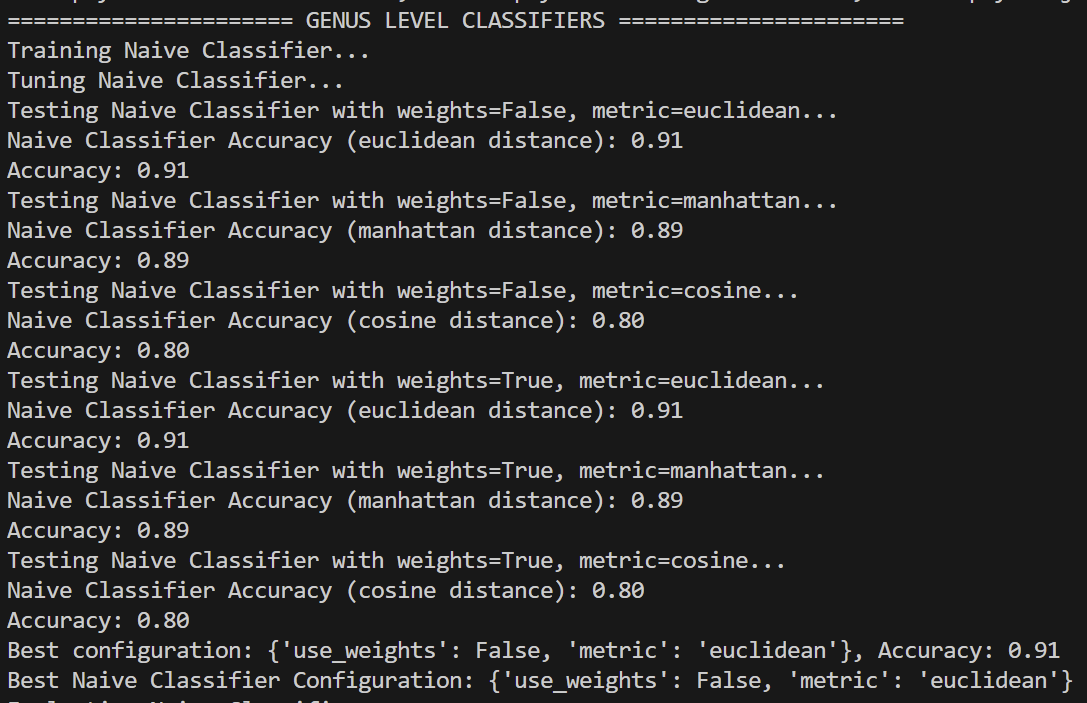
Try to generate samples for minority classes, the problem is that there are not enough samples for training and testing, so we can try to generate samples, use those for training and then the originals for testing.

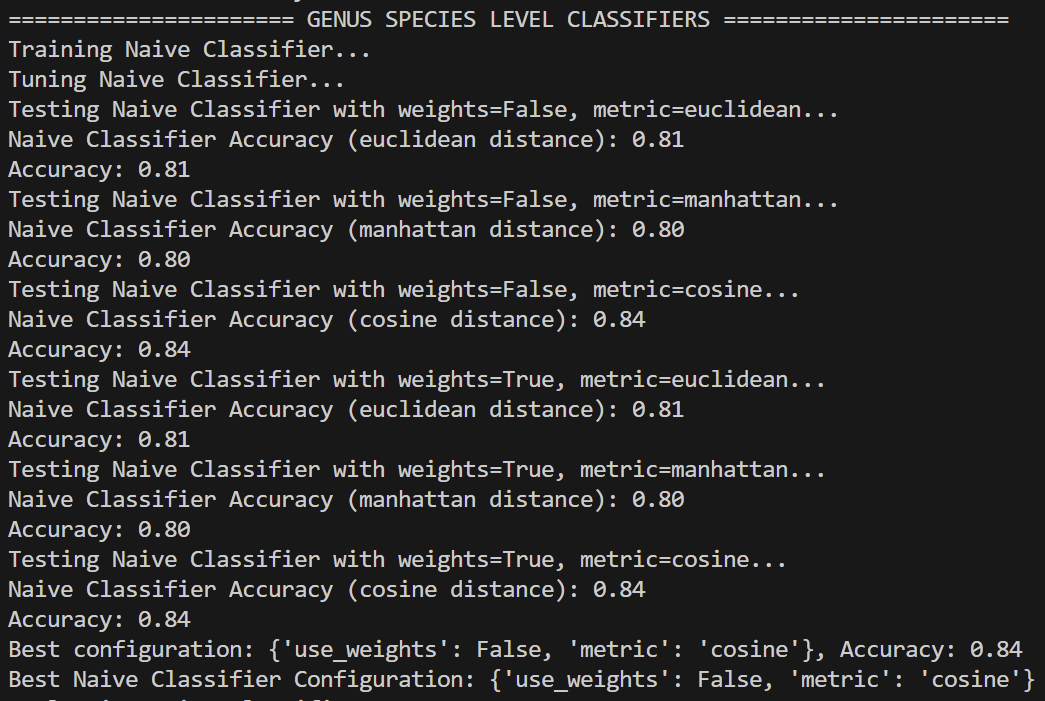
## Step 4: Model Tuning

Tuning k for KNN



Tuning for naive



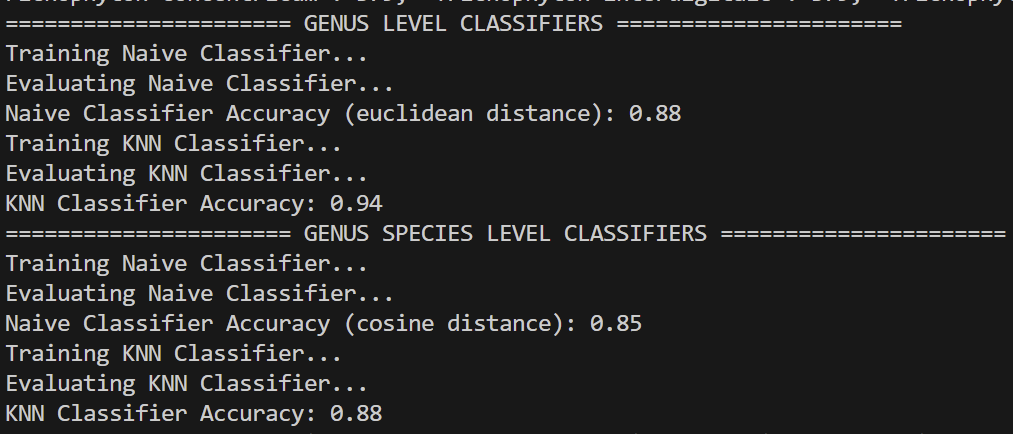


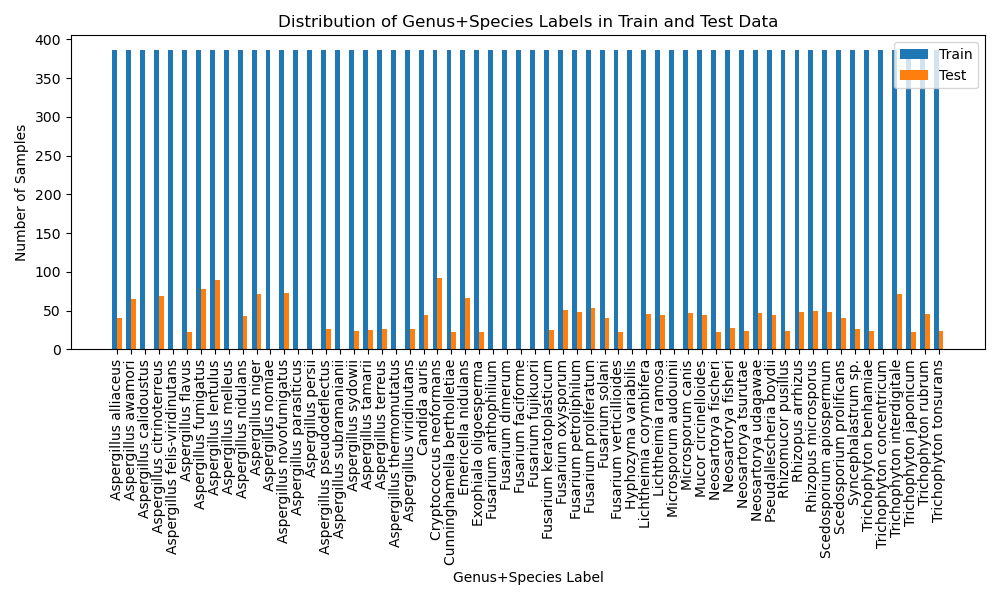
KNN n\_neighbors=3

Naive

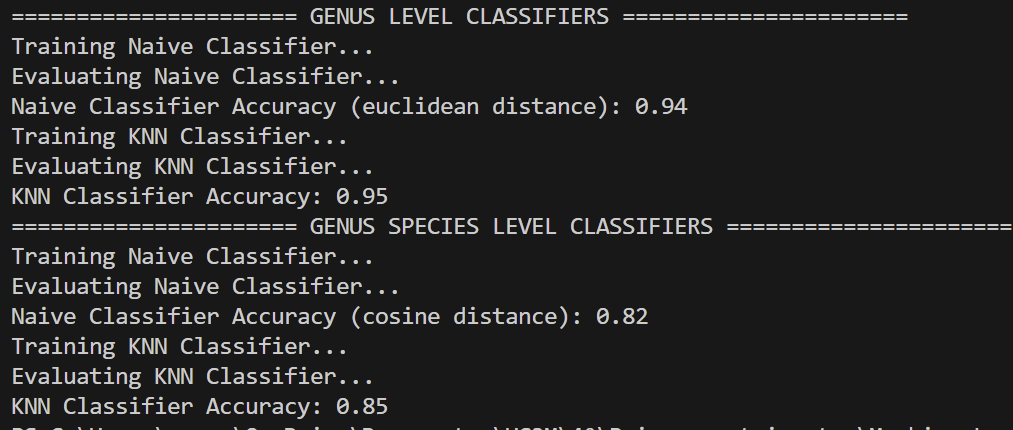
* genus: Euclidean distance, no weights
* species: Cosine distance, no weights

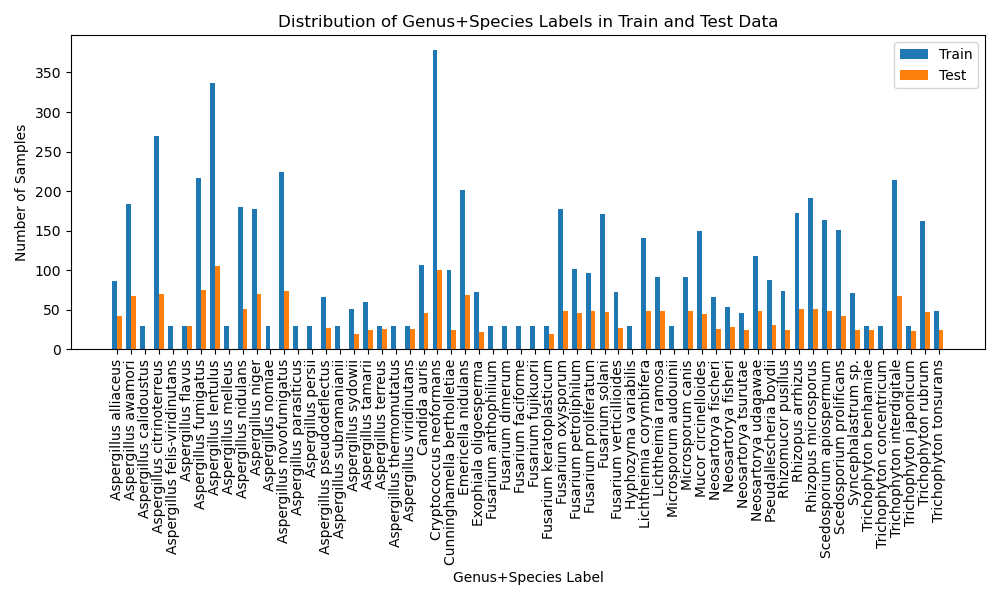
SMOTE all species up to max (~380)



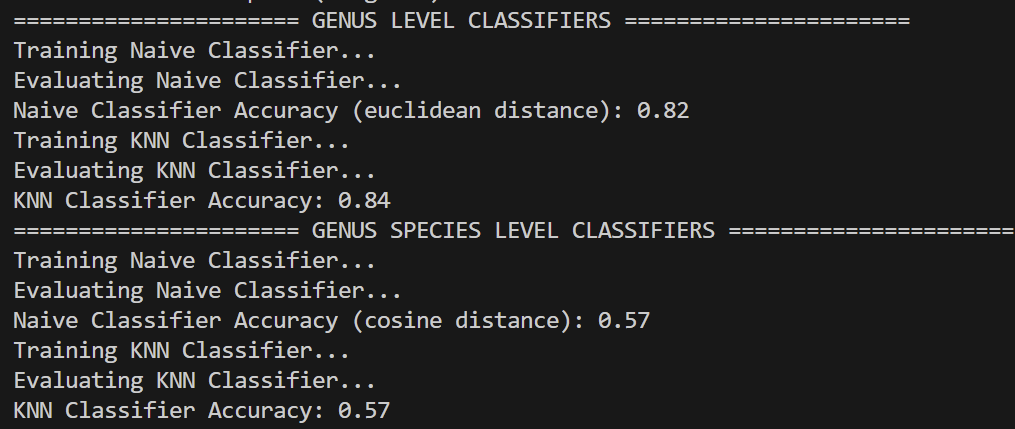


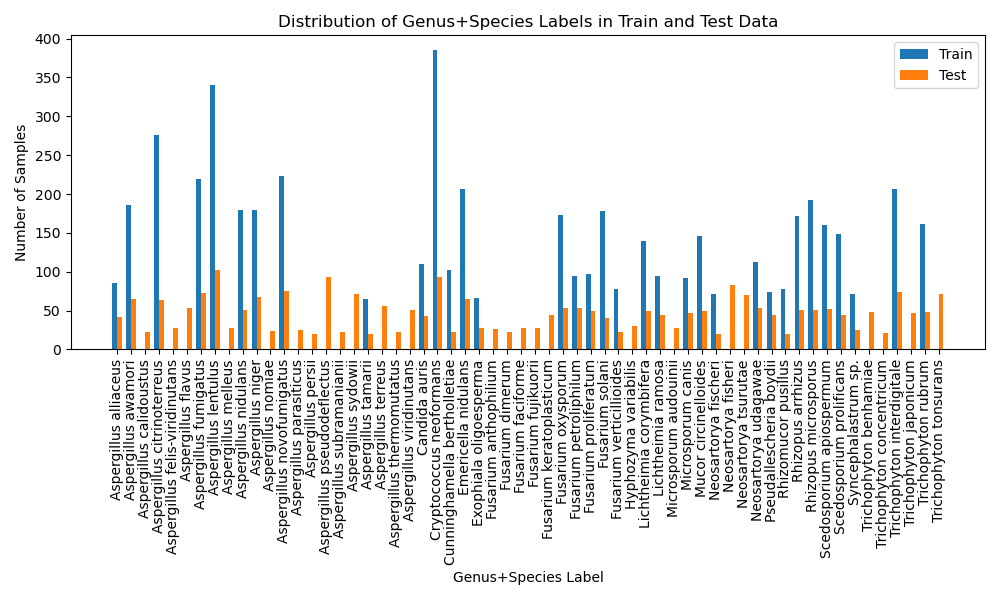
SMOTE minority generate samples up to ~30





Use SMOTE generated for training and original samples in test

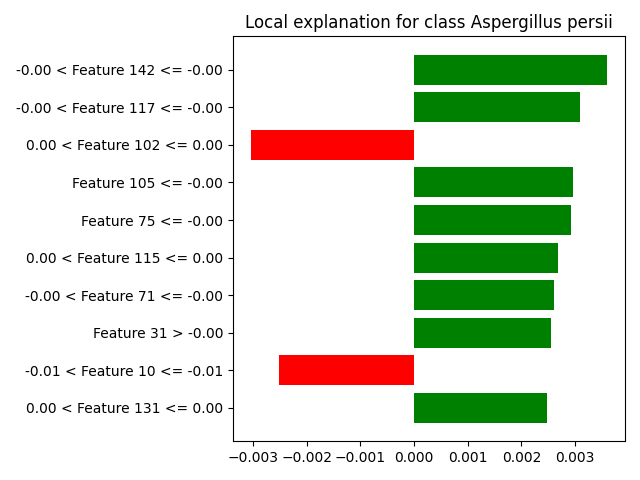




Train with SMOTE generated data not good, not enough spectrums to generate samples, just adds overfitting

## Step 5: Explainability Tools

LIME Explainability Tool with KNN result



## Step 6: Metrics for Evaluation

ROC curves

