HW1 Report: Bird Whistle Recognition Using Machine Learning

110550205

Dataset Web Link

https://github.com/Kennethii2i/NYCU-AI-Capstone-Spr2025-Project1/tree/main
The dataset is sharing with 111550182

Research Question

How well can machine learning models recognize bird species based on their whistle recordings, and how do different data processing techniques affect classification and clustering performance?

Dataset Documentation

Data Type:

- Audio recordings (converted from MP3 to WAV format)
- Feature extraction using MFCC

External Source:

- The recordings were sourced from [Xeno-canto](), a global community-driven database of wildlife sounds.

Data Collection:

- Xeno-canto API was used to download 300 files per species.
- Some files were corrupt or failed to download.

Data Preprocessing:

1. **Format Conversion**: MP3 files were converted to WAV.
2. **Segmentation**: Each downloaded recording was separated into multiple 10-second audio recording, resulting in:
    - **Passer montanus**: 1605 files
    - **Corvus macrorhynchos**: 1218 files
    - **Larus canus**: 1207 files
    - **Anas platyrhynchos**: 1342 files
    - **Total**: 5372 files
3. **Feature Extraction**: 40-dimensional MFCC features were extracted.
4. **Outlier Removal**: Entries where the 10th MFCC feature was `0` were removed to clean the dataset, since I noticed that when the column is `0` has a bad sample.
5. **Label Encoding**: The categorical labels were transformed into numerical representations using `LabelEncoder`, stored in `label_encoder.pkl`.
6. **Data Splitting**: The dataset was split into training (80%) and validation (20%) sets using `train_test_split` with a `random_state` of 42 for reproducibility.

Machine Learning Methods

Supervised Learning Models:

- **Random Forest**: Used as a robust baseline classifier.
    - Used `RandomForestClassifier` with `n_estimators=100` and `random_state=42`.
    - Model trained on extracted MFCC features.

- **Support Vector Machine (SVM)**: Applied with different kernels to test classification performance.
    - Used `SVC` with a linear kernel.
    - Model trained on MFCC features.

- **Neural Network (NN)**: A deep learning approach for recognizing bird species based on extracted MFCC features.
    - Three-layer dense network using ReLU activations and softmax output.
    - Dropout layers (0.1) used for regularization.
    - Trained using `Adam` optimizer and `sparse_categorical_crossentropy` loss.
    - Best model saved as `best_bird_nn_model.keras` using `ModelCheckpoint`.

Unsupervised Learning Model:

- **K-Means Clustering**: Used to group bird sounds into clusters without predefined labels.
    - Used `KMeans` with the number of clusters set to the number of unique bird species.
    - Silhouette Score and Adjusted Rand Index used for evaluation.
    - Cluster visualization performed(see figure below).

Public Libraries Used: see references.

Analysis

1. Performance of Supervised Models:

Model	Accuracy	Confusion Matrix
Random Forest	0.9451	[[275 6 6 1] [5 219 4 3] [7 6 217 12] [1 5 3 305]]
SVM	0.8521	[[267 8 8 5] [13 192 21 5] [16 34 181 11] [1 12 25 276]]

Neural Network	0.9349	[[279 8 1 0] [4 214 12 1] [4 8 225 5] [7 4 16 287]]

- **Random Forest performed the best (94.51%)**, suggesting that the dataset benefits from an ensemble learning approach.
- **Neural Network came close (93.49%)**, showing that deep learning captures useful patterns.
- **SVM had the lowest accuracy (85.21%)**, likely due to the multiclass nature of the problem.

Confusion Matrix Observations

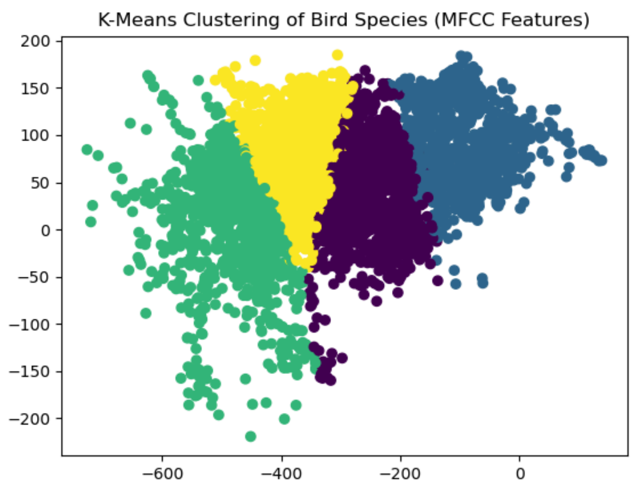
- **RF and NN show very low misclassification rates**, meaning they handle class separation well.
- **SVM struggles with misclassification**, especially in off-diagonal terms (e.g., classes getting confused with others).

2. Unsupervised Learning (K-Means):

Clustering Method	Silhouette Score	Adjusted Rand Index
K-Means	0.2510	0.1970

- **Silhouette Score (0.2510)**: Indicates poor separation between clusters.
- **Adjusted Rand Score (0.1970)**: Suggests weak correlation between clustering and ground truth labels.
- **Possible Reasons**:
    - Your dataset may not have clear, well-separated clusters.
    - Feature scaling and preprocessing might need improvement.
    - The number of clusters may not align well with the inherent structure of the data.

Observations: The clustering results show that while some grouping aligns with the actual labels, K-Means struggles due to the inherent complexity of bird vocalizations.



[**Figure 1.** K-means Clustering of bird Species]

Experiments

Training Size	RF Accuracy	SVM Accuracy	NN Accuracy	K-Means ARI
20%	88.19%	81.30%	85.12%	0.1722
40%	90.51%	82.88%	88.65%	0.1920
60%	92.19%	84.37%	90.98%	0.1918
80%	93.67%	85.02%	93.30%	0.1915
100%	**94.51%**	**85.21%**	**94.42%**	**0.1970**

[**Table 1.** The result of first experiment (effect of training data size)]

Model	Original Accuracy/ARI	With SMOTE	With Augmentation	With PCA
RF	**94.51%**	**95.26%**	**94.70%**	**92.56%**
NN	**93.49%**	**95.07%**	**94.60%**	**93.12%**
SVM	**85.21%**	**85.12%**	**89.67%**	**80.84%**
K-Means	**0.1970 (**ARI)	**0.1938**	**0.1984**	**0.1914**

[**Table 2.** Results of the second, third, and fourth experiments, respectively, showcasing the effects of data balance, data augmentation, and dimensionality reduction.]

Experiment 1: Effect of Training Data Size

- **Objective:**
  To analyze how the amount of training data impacts the performance of different models (Random Forest, SVM, Neural Network, and K-Means). This helps determine if increasing data size improves accuracy and whether there is a saturation point beyond which additional data has minimal impact.

- **Description:**
  - The dataset was split into different training sizes: **20%, 40%, 60%, 80%, and 100%**.
  - Each model was trained and validated on these different subsets.
  - Validation accuracy (for supervised models) and Adjusted Rand Index (for K-Means) were recorded.
  - The goal was to find the relationship between training data size and model performance.

Experiment 2: Effect of Data Balance (SMOTE)

- **Objective:**
  To determine whether applying **Synthetic Minority Over-sampling Technique (SMOTE)** to balance the dataset improves classification performance, particularly for models that may be sensitive to class imbalances.

- **Description:**
  - The dataset was imbalanced, with different numbers of samples per bird species.
  - **SMOTE was applied** to generate synthetic samples for underrepresented classes, making the dataset more balanced.
  - Models were trained on both the original and SMOTE-balanced datasets.
  - Performance was compared using validation accuracy, confusion matrices, and ARI (for K-Means).
  - The goal was to observe whether balancing data leads to improved accuracy and whether certain models benefit more from SMOTE.

Experiment 3: Effect of Data Augmentation

- **Objective:**
  To test whether **data augmentation** techniques can improve model generalization by increasing data diversity.

- **Description:**
  - **Time-stretching and pitch-shifting** were used as augmentation techniques:
  - **Time-stretching:** Speeds up audio by 10% (rate=1.1).
  - **Pitch-shifting:** Raises the pitch by 2 semitones (n_steps=2).
  - Each original audio sample was augmented, effectively **doubling the dataset size**.
  - Models were trained with and without augmented data, and performance was compared.
  - The experiment aimed to determine whether augmentation improves accuracy, particularly for models like SVM and Neural Networks that rely on robust feature learning.

Experiment 4: Effect of Dimensionality Reduction (PCA)

- **Objective:**
  To analyze how reducing the feature space using **Principal Component Analysis (PCA)** affects model performance.

- **Description:**
  - PCA was applied to **reduce the dimensionality** of MFCC features while retaining **95% variance**.
  - Models were trained on both the **original feature set** and the **PCA-transformed feature set**.
  - Performance was evaluated using accuracy (for RF, SVM, and NN) and ARI (for K-Means).
  - The goal was to see whether reducing feature dimensions improves efficiency and accuracy or leads to loss of critical information.

Discussion

Expected vs. Observed Results

The results **largely aligned with expectations**:
- **Increasing training data improved accuracy**, but gains plateaued after **80%**.
- **SMOTE enhanced RF and NN performance**, but **SVM showed little improvement** due to altered decision boundaries.
- **Data augmentation significantly boosted SVM (+4.46%)**, confirming its reliance on diverse training data.

- **PCA reduced accuracy for all models**, indicating that all **40 MFCC features were valuable**.

Factors Affecting Performance

- **Noise in audio recordings** affected classification, but RF and NN handled it better than SVM.
- **Class imbalance** impacted results; SMOTE helped RF and NN but not SVM or K-Means.
- **Feature distribution influenced model effectiveness**: RF was most robust, SVM struggled with non-linearity, and K-Means failed to form clear clusters.

Future Experiments

1. **Advanced augmentation** (background noise, reverb) for better generalization.
2. **CNNs on spectrograms** instead of MFCCs for improved classification.
3. **Hyperparameter tuning** for SVM and NN optimization.
4. **Exploring alternative dimensionality reduction** (t-SNE, autoencoders).

Key Takeaways & Open Questions

- **RF and NN are most effective**, while **K-Means is unsuitable**.
- **Augmentation helps SVM significantly**, but **PCA is not beneficial**.
- **Open questions:** Could CNNs outperform MFCC-based models? Would alternative augmentations further improve results?

References

Public Libraries Used

1. **Scikit-Learn** (for Random Forest, SVM, K-Means, and metrics)
    - Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825-2830. https://scikit-learn.org/
2. **TensorFlow/Keras** (for Neural Network)
    - Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2016). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/
3. **Joblib** (for model saving/loading)
    - Joblib Development Team. (2022). *Joblib: running Python functions as pipeline jobs*. https://joblib.readthedocs.io/
4. **Matplotlib** (for visualization)
    - Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90-95. https://matplotlib.org/
5. **NumPy & Pandas** (for data processing)
    - McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51-56. https://pandas.pydata.org/
    - Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). *Array programming with NumPy*. Nature, 585(7825), 357–362. https://numpy.org/
6. **Imbalanced-learn (SMOTE for oversampling)**
    - Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine*

Learning. Journal of Machine Learning Research, 18(17), 1-5. https://imbalanced-learn.org/

7. **Scikit-Learn PCA (Dimensionality Reduction)**
   - Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer. https://scikit-learn.org/stable/modules/decomposition.html#pca

Appendix: Code Implementation

- Repo