BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN
NHẬP MÔN HỌC MÁY

BÁO CÁO BÀI TẬP MÔN HỌC

NỘI DUNG: PRACTICE COMPETITION

PHASE 01- SCIKIT LEARN

Giảng viên hướng dẫn :Nguyễn Thanh Tình
Huỳnh Lâm Hải Đăng

Lớp :CQ2022/24

Sinh viên thực hiện :Đinh Viết Lợi-22120188
Nguyễn Trần Lợi- 22120190
Nguyễn Nhật Long-22120194

Hồ Chí Minh, ngày 23 tháng 5 năm 2025

MỤC LỤC

PHẦN 1: TỔNG QUAN ĐỒ ÁN

I. Tổng quan đồ án

1. Giới thiệu đồ án

- Đồ án Practice Competition là đồ án kết thúc môn học của bộ môn Nhập môn Học máy của lớp CQ2022/24 Trường Đại học Khoa học Tự nhiên học kỳ II năm học 2024-2025.
- Đồ án tập trung vào thực hành quy trình xây dựng các mô hình học máy phục với 3 giai đoạn chính: Exploratory Data, Model Development, and Model Evaluation. Trọng tâm quy trình sẽ nằm ở hai giai đoạn sau.
- Hình thức thực hiện đồ án được tổ chức dưới dạng một cuộc thi trên Kaggle với tư cách cá nhân hoặc theo nhóm. Các thông tin về cuộc thi được trình bày tại Kaggle competition.
- Bài toán được đặt ra bởi cuộc thi sẽ được yêu cầu giải quyết trong vòng 3 tuần với 3 giai đoạn (phase) khác nhau. Tuần đầu của cuộc thi được yêu cầu sử dụng Scikit-learn.

2. Yêu cầu đồ án

- Các sinh viên được mong đợi tuân thủ một quy trình phát triển cho dự án Học máy thực tế, được cấu trúc với ba bước chính: Phân tích Dữ liệu Khám phá (Exploratory Data Analysis), Phát triển Mô hình (Model Development) và Đánh giá Mô hình (Model Evaluation).
- Bài tập được thực hiện dưới hình thức một cuộc thi Kaggle, sinh viên cần tham gia theo thể thức cá nhân hoặc theo đội nhóm.
- Dự án được lên kế hoạch diễn ra trong tổng cộng 3 tuần, được cấu trúc thành 3 giai đoạn riêng biệt. Mỗi giai đoạn kéo dài một tuần và bao gồm toàn bộ chu trình của hai bước chính: Phát triển Mô hình, và Đánh giá và Phân tích Lỗi.
- Mỗi sinh viên/ nhóm được yêu cầu nộp quá trình thực hiện hàng tuần bao gồm mã nguồn Jupyter notebook và báo cáo cho từng Phase.
- Mã nguồn Jupyter Notebook cần có: thông tin sinh viên/ nhóm sinh viên, giải thích chi tiết từng bước bằng hình ảnh minh hoạ, biểu đồ, công thức, chú thích (comment) từng bước xử lý và trực quan, notebook được format tốt.
- Báo cáo bao gồm: thông tin sinh viên/ nhóm sinh viên, tự đánh giá kết quả, ưu và nhược điểm của các phương pháp xử lý, mô hình và frameworks khác nhau. Thông tin chi tiết mỗi giai đoạn cần đảm bảo:
    - **Model development**: Trình bày việc lựa chọn mô hình, thiết kế kiến trúc và thiết lập huấn luyện cùng với lý do đằng sau các lựa chọn của bạn.
    - **Đánh giá Mô hình:** Báo cáo về hiệu suất mô hình bằng cách sử dụng các chỉ số

phù hợp, đồng thời bao gồm các quan sát về thời gian chạy, mức sử dụng tài nguyên tính toán. Cung cấp phân tích chi tiết về bất kỳ loại lỗi nào mà sinh viên và/hoặc nhóm đã xác định là điểm yếu của mô hình.

II. Kết quả đạt được

- Quá trình xây dựng mô hình của nhóm được áp dụng mạnh mẽ và đa dạng các thư viện, framwork được hỗ trợ sẵn.
- Nhóm áp dụng nhiều các phương pháp xử lý khác nhau nhằm chuẩn bị được bộ dữ liệu tốt nhất, phân tích các đặc điểm của dữ liệu hỗ trợ, gây khó khăn cho quá trình huấn luyện.
- Áp dụng rộng rãi các các mô hình học máy khác nhau, đánh giá ưu nhược điểm từng mô hình, thiết lập bộ trọng số tốt nhất cho mô hình thông qua các phương pháp Grid Search, Random Search…
- Hầu hết mô hình được nhóm lựa chọn sau cùng đều cho kết quả ấn tượng, áp dụng tốt cho bộ dữ liệu test.
- Thông tin cụ thể về quá trình làm cũng như động lực được trình bày rõ ràng trong notebook và báo cáo.

PHẦN 2: BÁO CÁO NHÓM

I. Thông tin nhóm

Thành viên	Vai trò
Đinh Viết Lợi	Phương pháp SMOTE, tách frame, viết báo cáo tổng hợp.
Nguyễn Trần Lợi	Phương pháp mean, feature engineering, PCA, trực quan hoá.
Nguyễn Nhật Long	Lựa chọn mô hình, tìm kiếm bộ siêu tham số phù hợp, lựa chọn tham số đánh giá.

II. Quy trình thực hiện

- Tận dụng số lượng submit có hạn hằng ngày trong thời gian ngắn, nhóm liên tục cập nhật source code hằng ngày để kiểm tra mức độ cải thiện qua từng phiên bản:
  - Ngày 1: phương pháp mean lấy trung bình các vector audio_embedding, đánh dấu trực tiếp nhãn, mô hình Random Forest.
  - Ngày 2: Sử dụng Grid Search tìm kiếm siêu tham số tối ưu cho mô hình Random Forest.
  - Ngày 3: Thử nghiệm nhiều mô hình hơn cho bài toán, phát hiện SVM có hiệu suất tốt nhất trong các mô hình, tiến hành tìm bộ tham số tối ưu cho mô hình SVM.
  - Ngày 4: Áp dụng các phương pháp phổ biến trong lĩnh vực tiền xử lý dữ liệu, bao gồm các phương pháp SMOTE, trực quan hoá dữ liệu để xác định đặc trưng, tạo đặc trưng mới, PCA, LDA.
  - Ngày 5: Tạo thêm mẫu mới thông qua việc tách frame từ list các vector embedding.
  - Ngày 6: Tổng kết và kết hợp các phương pháp lại với nhau.

PHẦN 3: MODEL DEVELOPMENT

I. Phân tích bộ dữ liệu

- Bộ dữ liệu là tập các record được lưu dưới dạng bảng của loại dữ liệu supervised với nhãn is_turkey xác định xem liệu một audio âm thanh có phải là tiếng gà tây hay không.
- Các thuộc tính của dữ liệu bao gồm: các đặc trưng âm thanh của từng frame ~ 1s được embedding thành một features vector gồm 128 chiều thông thường đoạn audio n giây sẽ có ~n frame, các thuộc tính còn lại bao gồm thời gian bắt đầu và kết thúc của video.
- Quan sát đặc tính của bộ dữ liệu cụ thể là các thuộc tính cho thấy chỉ có thuộc tính audio_embedding sẽ là trọng tâm xác định nhãn của dữ liệu, các thuộc tính còn lại không đóng vai trò trong việc huấn luyện mô hình.

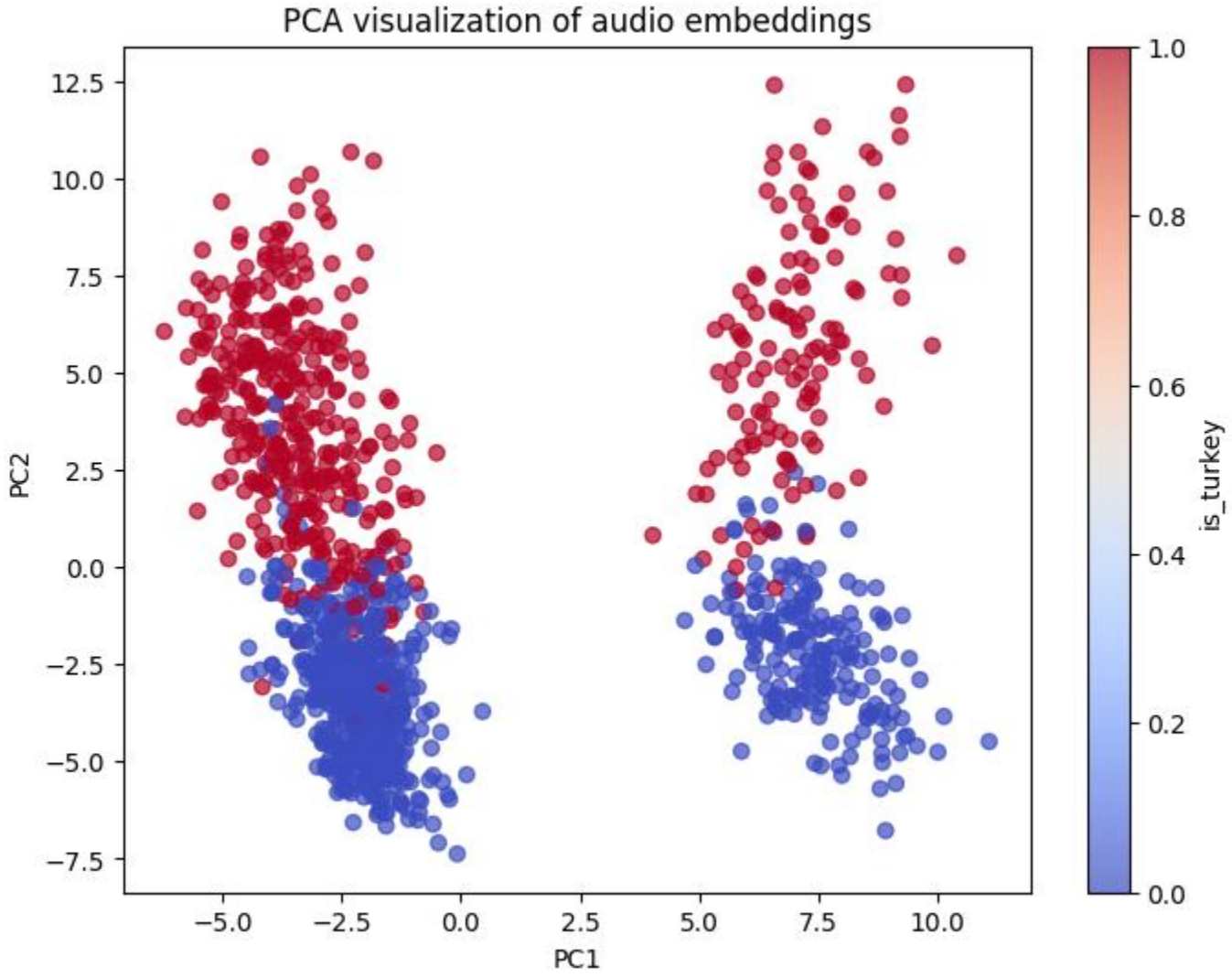
II. Chi tiết quá trình xây dựng mô hình

- **Trích xuất đặc trưng cố định từ audio_embedding:** Do mỗi audio_embedding có số lượng frame khác nhau (tức là chiều thứ 2 của embedding không cố định), để chuẩn hóa đầu vào cho mô hình học máy, ta thực hiện việc **lấy trung bình theo trục thời gian** cho từng audio_embedding. Cụ thể, với mỗi mẫu âm thanh, ta tính trung bình theo từng cột (tức là trung bình vector theo từng chiều đặc trưng), kết quả là mỗi mẫu sẽ được biểu diễn bởi một vector có chiều cố định.
- **Chuẩn hóa dữ liệu**: Sau khi rút trích đặc trưng cố định, ta sử dụng **StandardScaler** để chuẩn hóa dữ liệu đầu vào. Phương pháp này chuẩn hóa mỗi đặc trưng sao cho chúng có **trung bình bằng 0 và độ lệch chuẩn bằng 1**, giúp mô hình học máy hội tụ nhanh hơn và tránh bị lệ thuộc vào độ lớn của các đặc trưng.

```
scaler= StandardScaler()

# Chuyển dữ liệu thành dạng chuẩn
Z = scaler.fit_transform(train_X)
test_Z = scaler.transform(test_X)
```

- **Chia tập dữ liệu train và validation:** Dữ liệu sau chuẩn hóa được chia thành hai tập: **tập huấn luyện (training set)** và **tập xác thực (validation set)** để đánh giá khả năng tổng quát hóa của mô hình. Việc chia dữ liệu này đảm bảo mô hình không bị đánh giá trên chính dữ liệu nó đã học.
- Áp dụng mô hình Support Vector Classification (SVC) vào quá trình huấn luyện mô hình, tham số probability=True giúp trả về kết quả là khả năng nhãn thuộc về lớp nào.



- Quan sát biểu đồ trực quan cho thấy:
  - Cụm nhãn 1(màu đỏ) nằm phía trên.
  - Cụm nhãn 0(màu xanh) nằm phía dưới.
- Cho thấy việc sử dụng trung bình đủ để phân biệt giữa "1" và "0". Do đó việc sử dụng mô hình như Logistic Regression, SVM, RandomForest,... sẽ đạt được độ chính xác cao. Bộ tham số cụ thể cho từng mô hình ứng viên được đề cập tại "Phần 5- Đánh giá phương pháp".
- **Phương pháp K-folds**: **Chia dữ liệu thành K phần (folds)** bằng nhau và chạy mô hình **K lần**, mỗi lần chọn 1 fold làm **tập kiểm tra (test)**, và phần còn lại làm **tập huấn luyện (train)**. **Tính trung bình kết quả** của K lần để đánh giá tổng thể mô hình.

III. Các phương pháp khác

- SMOTE:
  - SMOTE (**Synthetic Minority Over-sampling Technique**) là một phương pháp tăng cường dữ liệu thiểu số trong các bài toán phân loại mất cân bằng (imbalanced

classification). SMOTE không sao chép dữ liệu thiểu số, mà thay vào đó tạo ra các điểm dữ liệu mới bằng **cách nội suy (interpolation)** giữa các điểm gần nhau trong không gian đặc trưng.

- Thư viện hỗ trợ:

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
```

- Tuy nhiên mô hình kết quả lại không cho kết quả cải tiến, điều này có thể được giải thích bởi vốn mô hình ban đầu không bị thiên lệch dữ liệu mạnh, việc nội suy ra những mẫu mới có thể không phù hợp.

```
X_train_mean = np.stack(X_train['audio_embedding'].apply(lambda x: np.mean(x, axis=0)))

adasyn_sampler = ADASYN(random_state=42)
X_train_resampled, y_train_resampled = adasyn_sampler.fit_resample(X_train_mean, y_train)


# smote_tomek_sampler = SMOTETomek(random_state=42)
# X_train_resampled, y_train_resampled = smote_tomek_sampler.fit_resample(X_train_mean, y_train)

print(f"Kích thước X_train_mean ban đầu: {X_train_mean.shape}")
print(f"Kích thước y_train ban đầu: {y_train.shape}")
print(f"Kích thước X_train_resampled sau ADASYN: {X_train_resampled.shape}")
print(f"Kích thước y_train_resampled sau ADASYN: {y_train_resampled.shape}")
print("\nSố lượng mẫu mỗi lớp sau ADASYN:")
print(y_train_resampled.value_counts())
```

```
Kích thước X_train_mean ban đầu: (956, 128)
Kích thước y_train ban đầu: (956,)
Kích thước X_train_resampled sau ADASYN: (1171, 128)
Kích thước y_train_resampled sau ADASYN: (1171,)
```

- PCA:
  - **PCA** là một phương pháp **giảm chiều dữ liệu tuyến tính** (linear dimensionality reduction) thường được sử dụng trong các bài toán tiền xử lý dữ liệu (preprocessing), phân tích đặc trưng và trực quan hóa dữ liệu. Nhóm áp dụng phương pháp này nhằm giảm 128 chiều dữ liệu sau embedding xuống còn 2 chiều để dễ quan sát dữ liệu hơn.
  - **Dùng PCA để trực quan hóa và nhận biết nhiễu**: Trước tiên, dữ liệu được biến đổi sang không gian mới bằng **PCA (Principal Component Analysis)** với 2 thành phần chính (PC1 và PC2). Việc này giúp đơn giản hóa dữ liệu và cho phép trực quan hóa các điểm dữ liệu theo hai trục chính. Qua biểu đồ phân tán, có thể quan sát thấy một số điểm dữ liệu có nhãn sai lệch rõ ràng so với cụm chính — ví dụ, các điểm màu đỏ (class 1) nằm lọt thỏm trong vùng màu xanh (class 0), hoặc ngược lại.
  - **Xóa nhiễu theo chiều PC2**: Vì sự phân tách giữa hai lớp dữ liệu được thể hiện rõ ràng theo chiều **PC2**, ta chọn cách **lọc nhiễu dựa vào ngưỡng của PC2**. Cụ thể: Với các điểm thuộc **class 0**, nếu giá trị PC2 quá cao (nằm trong vùng chủ yếu của

class 1), ta xem đó là nhiễu và loại bỏ. Ngược lại, với các điểm thuộc **class 1**, nếu giá trị PC2 quá thấp (nằm trong vùng chủ yếu của class 0), ta cũng loại bỏ.

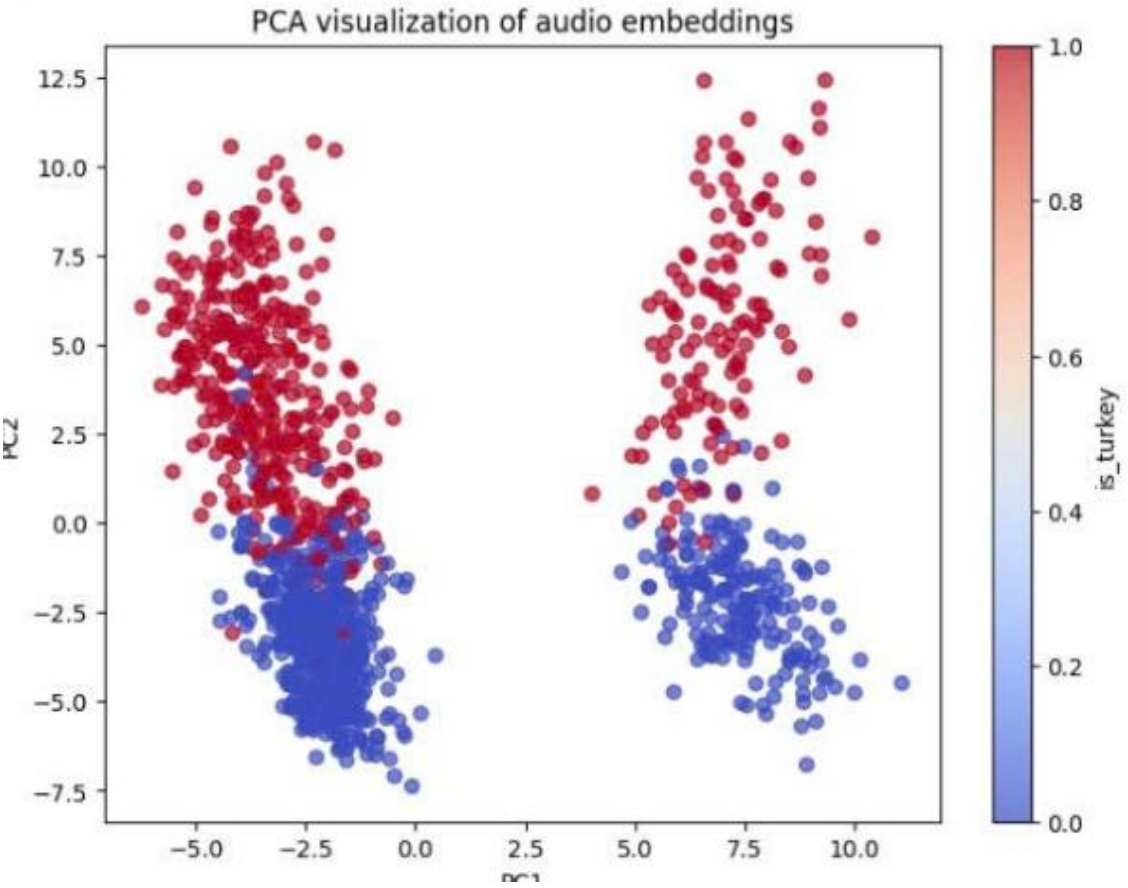
- **Kết quả sau khi lọc nhiễu**: Kết quả trên **tập validation** cho thấy **hiệu suất mô hình được cải thiện rõ rệt**, thể hiện qua các chỉ số như accuracy, precision hoặc recall đều tăng so với mô hình huấn luyện trên dữ liệu ban đầu. Tuy nhiên, khi **đánh giá mô hình trên tập test,** kết quả không tốt như trước khi lọc nhiễu. Điều này có thể giải thích bởi các mẫu dữ liệu này không phải nhiễu mà là các mẫu quan trọng đồng thời cũng chứng minh đây là bộ dữ liệu sạch.
- Thư viện hỗ trợ

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

```
from sklearn.decomposition import PCA

PCA_model = PCA(n_components=2)
X_pca = PCA_model.fit_transform(Z)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=train_Y, cmap='coolwarm', alpha=0.7)
plt.title("PCA visualization of audio embeddings")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.colorbar(label='is_turkey')
plt.show()
```



- Tách frames:

- Tách frames từ list các vector embedding dựa trên một ý tưởng và suy diễn rằng từng frame trong các video đều là một mẫu, có nhãn tương tự với nhãn của cả list vector. Việc tách các frames này ra thành một mẫu riêng biệt, giữ nguyên nhãn có thể giúp tạo thêm mẫu cho mô hình dữ liệu, giúp mô hình trở nên chính xác hơn đồng thời giữ lại thông tin thay vì chuẩn hoá list các vector này bằng mean.
- Phương pháp này được thực hiện bằng việc tách từng vector trong list vector thành các mẫu mới. Phương pháp này còn đồng thời kết hợp với việc loại bỏ các frame đầu cuối mỗi list để đảm bảo không có khoảng thông tin trắng hoặc nhiễu.

```python
def expand_audio_embeddings(data):
    expanded_rows = []
    for idx, row in data.iterrows():
        embeddings = row['audio_embedding']
        for emb in embeddings:
            new_row = row.copy()
            new_row['audio_embedding'] = emb
            expanded_rows.append(new_row)
    expanded_data = pd.DataFrame(expanded_rows)
    expanded_data.reset_index(drop=True, inplace=True)
    return expanded_data
```

PHẦN 4: MODEL EVALUATION

I. Các bộ chỉ số đánh giá

- AUC:
  - Ý nghĩa: AUC đo lường khả năng của mô hình trong việc phân biệt giữa hai lớp. AUC càng gần 1 thì mô hình phân biệt càng tốt. AUC = 0.5 tương đương với mô hình ngẫu nhiên.
  - Dữ liệu đầu vào: Nhãn thật (y_true) và xác suất dự đoán (y_prob).
  - AUC xét toàn bộ các giá trị ngưỡng từ 0 đến 1, không cố định tại một giá trị nào.
  - Thư viện hỗ trợ:

```
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_true, y_prob)
```

- Accuracy:
  - Ý nghĩa: Tỷ lệ dự đoán đúng trên tổng số mẫu. Phản ánh mức độ tổng quát mà mô hình dự đoán chính xác.
  - Dữ liệu đầu vào: Nhãn thật (y_true) và xác suất dự đoán (y_prob).
  - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y_prob) thành nhãn (y_pred).
  - Thư viện hỗ trợ:

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_true, y_pred)
```

- Precision:
  - Ý nghĩa: Tỷ lệ các mẫu được mô hình dự đoán là dương (positive) mà thực sự là dương. Phản ánh mức độ tin cậy khi mô hình dự đoán 1.
  - Dữ liệu đầu vào: Nhãn thật (y_true) và xác suất dự đoán (y_prob).
  - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y_prob) thành nhãn (y_pred).
  - Thư viện hỗ trợ:

```
from sklearn.metrics import precision_score
precision = precision_score(y_true, y_pred)
```

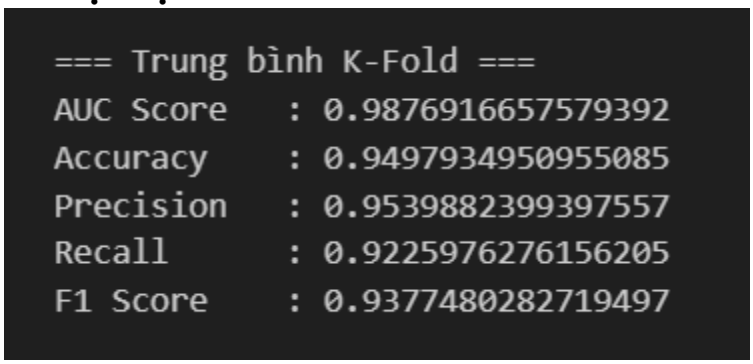
- Recall:
  - Ý nghĩa: Tỷ lệ các mẫu thực sự dương được mô hình dự đoán đúng là dương. Quan trọng trong các bài toán nhạy cảm như y tế, an ninh.
  - Dữ liệu đầu vào: Nhãn thật (y_true) và xác suất dự đoán (y_prob).
  - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y_prob) thành nhãn (y_pred).
  - Thư viện hỗ trợ:

```
from sklearn.metrics import recall_score
recall = recall_score(y_true, y_pred)
```

- F1-score:
  - Ý nghĩa: Là chỉ số cân bằng giữa Precision và Recall. F1-score cao nghĩa là mô hình vừa chính xác vừa phát hiện tốt lớp dương.
  - Dữ liệu đầu vào: Nhãn thật (y_true) và xác suất dự đoán (y_prob).
  - Cần chọn ngưỡng (ví dụ: 0.5) để chuyển xác suất (y_prob) thành nhãn (y_pred).
  - Thư viện hỗ trợ:

```
from sklearn.metrics import f1_score
f1 = f1_score(y_true, y_pred)
```

II. Đánh giá và cải thiện bộ chỉ số



- **AUC Score = 0.9877** → Rất cao, cho thấy mô hình **phân biệt rất tốt** giữa hai lớp (có tiếng gà tây vs không). Mô hình có khả năng **xếp hạng đúng xác suất** thuộc lớp dương (gà tây kêu).
- **Accuracy = 0.9498 (~95%** → Mô hình dự đoán đúng gần 95% tổng số mẫu. Tuy nhiên, **accuracy không phản ánh tốt hiệu suất nếu dữ liệu mất cân bằng**, cần xem cùng precision/recall.
- **Recall thấp hơn Precision** → Mô hình tránh được báo động giả (ít dương tính giả), nhưng có thể bỏ sót một số trường hợp thực sự dương tính (âm tính giả).
- F1 Score = 0.9377 →**Là điểm cân bằng giữa precision và recall. Giá trị > 0.93 cho thấy mô hình đạt hiệu quả cao cả trong** tính chính xác lẫn khả năng phát hiện**.**
- Mô hình đang thể hiện **rất tốt**, với **AUC gần tuyệt đối**, cùng **precision, recall, F1 đều vượt ngưỡng 0.92.** Tuy nhiên khi nhóm thử nhiều bộ tham số cho 5 chỉ số cao trên cao hơn, nhưng score submission lai thấp hơn → Khả năng cao mô hình đang bị overfitting.

III. Một số mô hình khác

- Trong quá trình thực nghiệm, nhóm đã tiến hành đánh giá hiệu suất của 20 mô hình học máy khác nhau nhằm giải quyết bài toán phân loại nhị phân:
  - **Mô hình tuyến tính & thống kê**: LogisticRegression, RidgeClassifier, GaussianNB, QuadraticDiscriminantAnalysis.
  - **Mô hình dựa trên cây & ensemble**: DecisionTree, RandomForest, ExtraTrees, Bagging, AdaBoost, GradientBoosting, HistGradientBoosting, XGBoost,

LightGBM, CatBoost.

- **Mô hình khoảng cách & mạng đơn giản**: KNeighborsClassifier, MLPClassifier, SVC.

- RandomForest, ExtraTrees đều cho chỉ số tốt hơn các mô hình khác khi dự đoán trên cùng tập dữ liệu.

PHẦN 5: ĐÁNH GIÁ PHƯƠNG PHÁP

I. Phương pháp tiền xử lý dữ liệu

- Các phương pháp tiền xử lý dữ liệu dựa trên hai trường phái chính bao gồm các phương pháp xử lý dựa theo nguyên tắc tiền xử lý tổng quát và tiền xử lý logic với thực tế.
- Trích xuất đặc trưng bằng mean:
  - **Ưu điểm:** Đơn giản, nhanh, dễ tính toán. Rút gọn được chiều dữ liệu → giảm kích thước đầu vào cho mô hình.
  - **Nhược điểm:** Thiếu cơ sở lý thuyết làm mất thông tin chi tiết theo từng đặc trưng. Có thể làm mất các mẫu âm thanh đặc trưng quan trọng.
- Chuẩn hoá dữ liệu:
  - **Ưu điểm:** Giúp đưa các đặc trưng về cùng một thang đo, rất phù hợp với: mô hình học máy tuyến tính (Logistic Regression, Linear Regression,...) và mô hình nhạy với khoảng cách (KNN, SVM, K-means...). Tăng hiệu quả huấn luyện.
  - **Nhược điểm:** Nhạy cảm với outliers nếu tồn tại nhiều outliers sẽ dẫn đến hiệu suất kém cho mô hình.
- Tách frames là một phương pháp logic với nhận định rằng từng frame của list vector embedding đều thể hiện được nội dung toàn video, việc cắt bỏ frame đầu và cuối hợp lý thực tế nhưng ứng dụng vào lại không thành công. Phương pháp này chỉ là nhận định khách quan, không dựa vào bằng chứng thực tế.
- PCA để trực quan, giúp ta nhìn sâu vào đặc tính của dữ liệu để hiểu hơn về đặc tính của dữ liệu từ đó có các ý tưởng tiền xử lý phù hợp khi cho rằng mẫu dữ liệu có thể chứ nhiễu. Đây là một phương pháp khả thi nếu được ứng dụng tốt hơn.

II. Phương pháp lựa chọn và xây dựng mô hình

- Bước 1: Chạy cùng lúc nhiều mô hình để xem mô hình nào đạt các chỉ số: AUC, Precision, Recall, Accuracy, F1 cao nhất
  - Kết quả: SVC, Extratrees, RandomForest đạt chỉ số cao. SVC đạt chỉ số cao nhất. Và trong thực tế, SVC cũng đạt score submission cao hơn 2 model còn lại.
- Bước 2: Chọn tham số
  - Dùng GridSearchCV để tìm bộ tham số đạt chỉ số cao nhất của model.
  - Dùng vòng lặp (tối ưu bằng joblib) để vét cạn, tìm ra bộ tham số (random_state, n_splits) đạt chỉ số cao nhất.
- Bước 3: Áp dụng thêm 1 số phương pháp kfolds, SMOTE,.... để xem mô hình có đạt hiệu quả tốt hơn không.
- Bước 4: Submit để kiểm tra bộ tham số nào đạt score trên leaderboard cao nhất.

Trong thực tế, Nhóm nhận thấy score trên leaderboard và 5 chỉ số đánh giá có mối liên hệ:

- Model với tham số phức tạp thì score thấp đi → Điều này chứng tỏ model dễ bị overfit trên tập dữ liệu này.
- Với AUC trong khoảng 0.987 đến 0.9904 thì model đạt kết quả cao hơn.
- Với AUC từ 0.983 trở xuống, và 0.9905 trở lên, model hầu như đạt kết quả thấp hơn.
- Khó khăn nhất là nhiều khi bộ tham số có 5 chỉ số đánh giá cao hơn, nhưng score submission lại thấp hơn bộ tham số có 5 chỉ số thấp hơn nó.
- Do đó điểm số đạt được do việc submit có yếu tố may rủi. Và phải thử ngẫu nhiên nhiều bộ tham số cho đến khi đạt quả cao hơn !

- Bước 5: Lặp lại bước 2
- K-fold thường cho kết quả tốt hơn vì:
  - **Tổng quát hơn:** Mô hình được đánh giá trên toàn bộ dữ liệu, không bị lệ thuộc vào một tập train-test duy nhất.
  - **Giảm overfitting/underfitting cục bộ**: Phát hiện nếu mô hình hoạt động tốt trên phần này nhưng kém trên phần khác.
  - **Ổn định hơn:** Trung bình nhiều lần → giảm nhiễu do chia dữ liệu ngẫu nhiên.

PHẦN 6: TÀI LIỆU THAM KHẢO

I. Tài liệu nhóm

- Github: Dzivilord/Machine_Learning_Project

II. Tài liệu tham khảo

- Khanh's Blog, 'Bài 24-Mất cân bằng dữ liệu (imbalanced dataset)' [Trực tuyến]. Đường dẫn: Khoa học dữ liệu
- Machine Learning cơ bản, 'Bài 27: Principal Component Analysis (phần 1/2)' [Trực tuyến]. Địa chỉ: Machine Learning cơ bản