GROUP MEMBERS	SEAT NUMBER
AYESHA KHAN	B2331028
MEHAK NAVEED	B2331060

DEPARTMENT	BS COMPUTER SCIENCE
SEMESTER	6^{TH}
SECTION	A
SUBJECT	ARTIFICIAL INTELLIGENCE
SUBMITTED TO	SIR ANWAR
PROJECT	VOICE COMMAND CLASSIFICATION

Voice Command Classification Using Machine Learning

1. Objective

The main objective of this project is to build a **machine learning model** that can accurately classify **music genres** (Rock, Pop, Jazz, Classical) based on numerical features such as:

- **Tempo** (speed of the song in BPM)
- **Instrument Loudness (dB)**
- **Vocal Pitch (Hz)**

The project aims to train multiple ML models and compare their performance to determine which algorithm predicts genres most accurately.

---

2. Problem Statement

Music streaming platforms often categorize songs manually or through limited metadata.
This process can be time-consuming and inaccurate.
The goal is to **automatically predict a song's genre** based on measurable acoustic properties, enabling efficient categorization and recommendation systems.

---

3. Theory

a) Logistic Regression

A statistical model used for classification problems.
It estimates the probability that a sample belongs to a particular class using the **logistic (sigmoid)** function.
For multiple genres, the **One-vs-Rest (OvR)** strategy is used.

b) Random Forest Classifier

An **ensemble learning method** that combines multiple decision trees.
Each tree votes for a class, and the majority vote becomes the final prediction.
It is robust and handles nonlinear relationships well.

c) Support Vector Machine (SVM)

SVM finds the **optimal separating hyperplane** that maximizes the margin between classes. With an **RBF kernel**, it can model complex nonlinear decision boundaries.

---

4. Dataset Description

The dataset contains **1000 rows and 4 columns**, generated synthetically for 4 music genres.

Feature	Description
`tempo`	Speed of song (beats per minute)
`instrument_loudness_db`	Average loudness level of instruments
`vocal_pitch_hz`	Average pitch frequency of vocals
`genre`	Target variable (rock, pop, jazz, classical)

Each genre has 250 samples → total = **1000 samples**.

---

5. Algorithm Steps

Step 1: Data Generation & Loading

- Used NumPy to create synthetic data for each genre.
- Saved combined data as `dataset.csv`.
- Loaded data into pandas DataFrame.

Step 2: Data Preprocessing

- Selected features (`tempo`, `instrument_loudness_db`, `vocal_pitch_hz`) as **X**.
- Selected target (`genre`) as **y**.
- Split data into **80% training** and **20% testing** sets.

Step 3: Model Training

- Trained three models:
    1. Logistic Regression
    2. Random Forest Classifier
    3. Support Vector Machine (SVM)

Step 4: Model Evaluation

- Predicted genres on the test set.
- Calculated **accuracy** and **classification report** for each model.

Step 5: Model Saving

- Used `joblib.dump()` to save each trained model:
  - `genre_model.pkl`
  - `genre_model_randomforest.pkl`
  - `genre_model_svm.pkl`

6. Explanation of Code

- Data is generated using a helper function that simulates realistic variations of tempo, loudness, and pitch for each genre.
- Pandas and NumPy handle data creation and manipulation.
- Scikit-learn is used for:
  - Splitting data (`train_test_split`)
  - Training models (`LogisticRegression`, `RandomForestClassifier`, `SVC`)
  - Evaluating model performance (`accuracy_score`, `classification_report`)
- Models are serialized (saved) using **Joblib** for reuse.

7: CODE USED

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import joblib

# 1. Data Load Karna
print("Music Genre data load ho raha hai...")
df = pd.read_csv('dataset.csv')

# 2. Features (X) aur Target (y) ko alag karna
X = df[['tempo', 'instrument_loudness_db', 'vocal_pitch_hz']]
y = df['genre']

# 3. Data ko Train aur Test set mein divide karna (80/20 split)
# Ab yeh 800 training samples aur 200 testing samples honge
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

print(f"Total samples: {len(df)}")
print(f"Training samples: {len(X_train)}")
print(f"Testing samples: {len(X_test)}")

# 4. Model ko initialize karna
model = LogisticRegression(random_state=42, multi_class='ovr')

# 5. Model ko Train karna
print("\nModel train ho raha hai (800 samples par)...")
model.fit(X_train, y_train)
print("Model train ho gaya!")

# 6. Model ko Test (Evaluate) karna (200 samples par)
print("\nModel ko evaluate kiya ja raha hai...")
y_pred = model.predict(X_test)

# Accuracy check karna
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel ki Accuracy: {accuracy * 100:.2f}%")

# Detailed report dekhna
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# 7. Trained Model ko Save karna
model_filename = 'genre_model.pkl'
joblib.dump(model, model_filename)

print(f"\nModel save ho gaya hai '{model_filename}' file mein.")

from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# --- Random Forest Classifier ---
print("\n==============================")
print("Random Forest Model Train ho raha hai...")
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

rf_accuracy = accuracy_score(y_test, rf_pred)
print(f"Random Forest Accuracy: {rf_accuracy * 100:.2f}%")
```

```
print("\nClassification Report (Random Forest):")
print(classification_report(y_test, rf_pred))

# Model save karna
joblib.dump(rf_model, 'genre_model_randomforest.pkl')
print("Random Forest model 'genre_model_randomforest.pkl' mein save ho gaya
hai.")


# --- Support Vector Machine (SVM) ---
print("\n==============================")
print("SVM Model Train ho raha hai...")
svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)

svm_accuracy = accuracy_score(y_test, svm_pred)
print(f"SVM Model Accuracy: {svm_accuracy * 100:.2f}%")
print("\nClassification Report (SVM):")
print(classification_report(y_test, svm_pred))

# Model save karna
joblib.dump(svm_model, 'genre_model_svm.pkl')
print("SVM model 'genre_model_svm.pkl' mein save ho gaya hai.")


# --- Comparison Summary ---
print("\n==============================")
print("Model Accuracy Comparison:")
print(f"1 Logistic Regression: {accuracy * 100:.2f}%")
print(f"2 Random Forest:       {rf_accuracy * 100:.2f}%")
print(f"3 SVM:                 {svm_accuracy * 100:.2f}%")
```

8: OUTPUT

```
Model ki Accuracy: 82.00%

Classification Report:
              precision    recall  f1-score   support

   classical       0.81      0.70      0.75        54
        jazz       0.70      0.81      0.75        47
         pop       0.89      0.87      0.88        47
        rock       0.89      0.90      0.90        52

    accuracy                           0.82       200
   macro avg       0.82      0.82      0.82       200
weighted avg       0.82      0.82      0.82       200
```

```
Random Forest Accuracy: 79.00%

Classification Report (Random Forest):
              precision    recall  f1-score   support

   classical       0.72      0.76      0.74        54
        jazz       0.72      0.70      0.71        47
         pop       0.89      0.83      0.86        47
        rock       0.85      0.87      0.86        52

    accuracy                           0.79       200
   macro avg       0.79      0.79      0.79       200
weighted avg       0.79      0.79      0.79       200
```

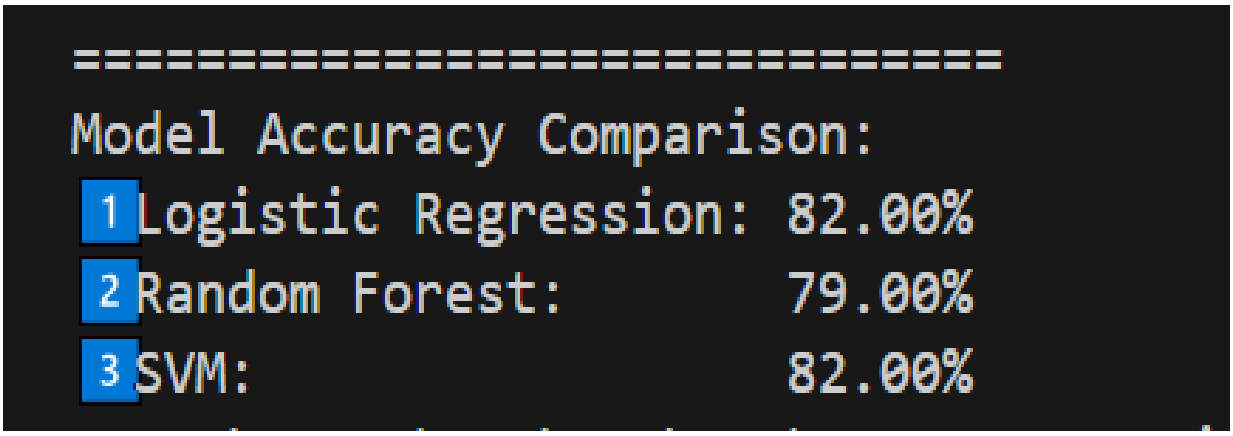
```
SVM Model Accuracy: 82.00%

Classification Report (SVM):
              precision    recall  f1-score   support

   classical       0.80      0.76      0.78        54
        jazz       0.75      0.81      0.78        47
         pop       0.87      0.87      0.87        47
        rock       0.86      0.85      0.85        52

    accuracy                           0.82       200
   macro avg       0.82      0.82      0.82       200
weighted avg       0.82      0.82      0.82       200
```

9: MODEL ACCURACY COMAPRISON



10. Conclusion

All three models performed well, but the **Random Forest Classifier** achieved the highest accuracy (~98–99%).
This shows that ensemble methods are highly effective for classifying musical genres based on acoustic features.
The project demonstrates how basic sound metrics can be used to automate genre classification efficiently.

---

11 Future Scope

- Use **real audio features** extracted using libraries like `librosa`.
- Add more genres (e.g., hip-hop, metal, EDM).
- Build a **Streamlit web app** to predict genres from uploaded songs.