PROGRAMMING MACHINE LEARNING APPLICATIONS

FINAL PROJECT

Team Members:

- Mohammad Anas Ilyas
- Sena Bui
- Mitanshi Kapadiya
- Yash Patel
- Charlotte Anderson

Guided by : Prof. David Hubbard

Description of Data Set:

Dataset link -
https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre

We have selected the dataset “Prediction of Music Genre" from kaggle. This dataset contains a sample size of 50,006 and describes the genre, artist, and goes over the creation of the music track through 18 different variables/columns. With these variables we can create a classification system to find the genre of each type of song as a form of prediction.

Variable Name	Variable Type	Description	Range/ Unique Values
instance_id	double	Unique ID for each music track	Unique
artist_name	character	Artist name	Unique
track_name	character	Track name	Unique
popularity	double	Popularity of the music track	0 - 99.0
acousticness	double	measure of whether the track is acoustic (instruments that produce sound through	0 - 0.996

		natural resonance, rather than electronics)	
danceability	double	measures how well a song is suited for dancing	0 - 0.9860
duration_ms	double	Duration of the music track in milliseconds (ms)	-1.0 ms - 4.83 ms
energy	double	Energy: measure of song's intensity and activity	0 - 1.0
instrumentalness	double	measurement of how likely a song is to contain vocals. Higher value indicates greater likelihood of no vocals	0 - 1.0
key	character	Music key	MusicKey (A,B,C…)
liveness	double	detects the presence of an audience in the recording.	0 - 1.0
loudness	double	loudness of a track in decibels (dB)	-47.0 - 3.744
mode	character	Mode	Minor/Major
speechiness	double	detects the presence of spoken words in a track	0 - 0.9420
tempo	character	speed at which a piece of music is played	0 - 200
obtained_date	character	Date	2001-03-31 to 2005-03-31
valence	double	the measure of how positive or negative a song sounds	0 - 0.9920
music_genre	character	Genre of the song	Electronic, Anime, Jazz, Alternative, Country, Rap, Blues, Rock, Classical, Hip-Hop

Problem Statement:

With the rapid growth of online music platforms and an ever-exploding global music library, it has become increasingly challenging to categorize and recommend songs to users. Traditional methods of classification often rely on manual tagging and predefined genre labels, which can be inconsistent, subjective, and struggle to keep up with the dynamic nature of modern music. The inconsistency can hinder the user's discovery of new music that aligns with their taste and limit the reach of some artists. One of the most prominent challenges faced by such methods is the rise of cross-genre songs, which seamlessly blend elements from multiple genres to create new, hybrid sounds.

Cross-genre songs have become more popular due to the creative freedom provided by digital production tools, evolving listener preferences, and the fluidity of modern music culture. For example, genres such as "rap-rock," "country-pop," and "EDM-hip hop" defy simple categorization and frequently blur the boundaries of traditional genre definitions. This presents a unique challenge for music recommendation algorithms that rely solely on rigid genre definitions, as they risk misclassifying or poorly recommending such songs. As a result, users may be deprived of discovering music that resonates with their evolving tastes, while artists creating innovative sounds may struggle to reach new audiences due to poor classification.

A robust and scalable system capable of leveraging key intrinsic features such as tempo, rhythm, acoustic profiles, and lyrical themes to accurately classify songs into genres—and even recognize cross-genre influences—is essential. By embracing a machine learning approach, we can provide a dynamic and adaptive method for categorizing songs, ensuring better music recommendations, enhancing user discovery experiences, and supporting artists who push the boundaries of genre norms.

Preprocessing:

To begin our preprocessing steps we started with the data loading of the music_genre.csv file that we loaded into a pandas data frame. An initial inspection of the data was performed with .head() and .shape to understand the variables and structure of the data frame. We handled missing values ensuring that the Tempo column did not contain “?” further replacing them with NaN values and converting them to a numeric type. All NaN values were then removed and replaced with the mean of the overall data set or dropped the row completely. There was further data type conversion to ensure columns like Tempo were converted to numeric types for further analysis enabling us to create our models.

Exploratory Analysis:

The exploratory data analysis steps included visualizations and further handling of missing data. We used .describe(include=”all”) to generate summary statistics which provided insights into unique variables, counts, and distributions of the variables. We examined the distribution of Tempo using a histogram with a density line using sns.histplot to visualize the shape. From this histogram we visualized a roughly Gaussian, although it was slightly skewed and multi-modal. This is where we replaced the NaN values with the mean based on the visualization. We then checked the distribution plot to verify the effectiveness and imputation.

All float-type variables were visualized through a box and whiskers plot to access their ranges, outliers, and means. Upon further investigation of the artist names, we decoded the names to transform non-english symbols to become readable. The dates columns were dropped as it only contained the year and the month the song was added to the data file, not the year the song was created. We created a bar chart to show the distribution on Key used for each song and converted them to dummy variables to be used in our models. Finally, the mode of the song was converted to dummy variables as well to complete the conversion of categorical variables to numerical variables.

We created boxplots of the remaining features to further check for outliers. With these plots, we noticed the length of the song had many outliers which are below 0 and above 10 minutes. Because the duration cannot be negative and 80 minutes for a music album doesn't look good, we chose a range of 0-10 minutes, which makes most sense, and removed all other points. Also, as per distribution plots, 'tempo', 'duration_min', 'loudness' and 'popularity' need to be transformed. From this, all other features are already scaled from visualizing the distribution plots.

Finally, a correlation plot was used to visualize which variables had strong and weak correlations. Through this plot, it seemed as though energy is strongly and positively correlated with loudness and strongly and negatively correlated with acousticness. We decided to either choose or not choose to remove these columns based on these results, but we will keep the columns for now and see if results are influenced by them. The reason we decided to keep them is that these features play an important role in predicting the genre in our models.

For each model, testing and training data sets were created with the encoding of the music genre variable as the predictive target.

Models:

1. SVC

To evaluate the Support Vector Machine's (SVM) efficacy in genre categorization, the dataset was subjected to both polynomial and radial basis function (RBF) kernels. The regularization parameter C was varied from 0.01 to 1000 in order to hyperparameter tune the RBF kernel. Based on the test data, the results showed that C=1 and C=10 produced the optimal balance between generalization and performance. Because of its capacity to represent intricate, non-linear relationships in the data, the RBF kernel outperformed the polynomial kernel in the tests. Comparing SVM to more straightforward models like Logistic Regression, classification reports revealed that SVM obtained excellent accuracy and a discernible improvement in correctly predicting genres.

SVM's strength is its competence in managing high-dimensional feature spaces, which is especially helpful for our dataset. However, with more complicated kernels and larger datasets, SVM'ization parameter C was varied from 0.01 to 1000 in order to hyperparameter tune the RBF kernel. Based on the test data, the results showed that C=1 and C=10 produced the optimal balance between generalization and performance. Because of its capacity to represent intricate, non-linear relationships in the data, the RBF kernel outperformed the polynomial kernel in the tests. Comparing SVM to more straightforward models like Logistic Regression, classification reports revealed that SVM obtained excellent accuracy and a discernible improvement in correctly predicting genres.s processing cost can rise dramatically. Further optimization of the speed could be achieved by investigating sophisticated kernels and improving feature selection. SVM proved to be a useful tool for this predicting task overall, exhibiting a high degree of accuracy in classifying music genres.

2. Logistics Regression

While other features already showed well-distributed values, min-max scaling was used to alter variables like tempo, popularity, and duration_min. Strong correlations, including the positive correlation between energy and loudness, were found using outlier analysis using correlation heatmaps and boxplots. Preprocessing prioritized data integrity, preparing the dataset for efficient modeling and trustworthy findings, even though highly correlated features were kept.

To categorize music genres according to different dataset properties, logistic regression was used. Following GridSearchCV's hyperparameter tuning, C=100 and penalty='l2' were found to be the optimal settings. In order to avoid overfitting and maintain a healthy balance between bias and variance, this design was chosen. The classification report shows that the model performed satisfactorily when tested on the test data. Although the existence of highly correlated characteristics in the dataset probably affects its performance, the results demonstrate that

Logistic Regression can successfully differentiate across genres. The confusion matrix offered more information about the model's predictions by pointing out both genres where misclassifications happened and places where the algorithm was able to produce good predictions.

Logistic regression was used to classify music genres based on several dataset features. The best settings were determined to be C=100 and penalty='l2' after GridSearchCV's hyperparameter adjustment. This design was used to prevent overfitting and preserve a sound ratio of variance to bias. The classification report demonstrates that the model's performance on the test data was satisfactory. The results show that Logistic Regression can effectively distinguish across genres, despite the fact that the dataset's performance is likely impacted by the presence of strongly correlated factors. By highlighting both genres where misclassifications occurred and locations where the algorithm was successful in producing accurate predictions, the confusion matrix provided further insight into the model's predictions.

3. Random Forest Classifier (RFC)

The Random Forest Classifier was used because of the performance in classification tasks, particularly when datasets contain feature importance and reducing overfitting. The RFC handles multiple decision trees which reduce the variance predictions and enhances stability. It is effective in handling categorical variables (although converted to dummy variables) and non-linear relationships. We could see the importance of the variables on predicting music genres. In this model, besides the data column, no other variables were dropped. It would be possible to run the best decision trees with hyperparameter training but our computers could not handle the 1300+ possible RFC outputs. This could be used in the future to specify which variables and parameters are unique for each individual genre and gross genre recognition.

The model performance achieved an accuracy of roughly 57% and a confusion matrix was made showing a breakdown of true positives, false positives, and other metrics from our dataset. While not perfect with the accuracy of 57%, it was the highest among the models tested. This indicates the model's ability to correctly predict the music genre of a song for a majority of instances. In the confusion matrix, we can visualize how well the model predicted across other key variable attributes. The diagonal values represent the true positive values where higher diagonals indicate a stronger performance. Off-diagonals show which areas the model was confused with providing insight in distinguishing between specific genres. Below, we can see the precision, recall, and f1-score for each genre prediction. The best predictions across all of the scores were for anime, classical and electronic music genres.

```
Accuracy: 0.5628698224852071
Classification Report:
              precision    recall  f1-score   support

 alternative       0.39      0.34      0.36       797
       anime       0.81      0.79      0.80       827
       blues       0.62      0.55      0.59       792
   classical       0.87      0.87      0.87       821
     country       0.58      0.56      0.57       804
  electronic       0.66      0.61      0.63       843
     hip-hop       0.37      0.38      0.38       824
        jazz       0.54      0.53      0.54       782
         rap       0.32      0.32      0.32       782
        rock       0.49      0.64      0.55       840

    accuracy                           0.56      8112
   macro avg       0.56      0.56      0.56      8112
weighted avg       0.57      0.56      0.56      8112
```

If this model is selected for the ensemble, further feature engineering could be useful to explore additional or derived features that may be better for separate genres.

4. K-Nearest Neighbours (KNN)

The approach for our KNN involved comprehensive data preparation, including numerical feature cleaning and normalization, TF-IDF vectorization of track and artist names, and careful feature combination with combined numerical and text features. By testing different k-values, it was identified k=12 as the optimal parameter with Euclidean distance, achieving a mean cross-validation accuracy of 0.4505 ± 0.0048 . We limited the features in the TF-IDF vectorizer to reduce the dimensionality of the feature space and to help with computational efficiency, speeding up model training and prediction times. We processed numerical data (including tempo and mode) with proper scaling and outlier handling, while text features (track and artist names) are handled using TF-IDF vectorization with different maximum features (1000 for tracks, 500 for artists). The implementation uses memory-efficient sparse matrices and chunk processing for handling potentially large datasets. Within our final combined notebook, different k-values of 1, 5,10, 50, and 100 were used to lessen the amount of run time for the KNN model and entire notebook file.

The cross-validation approach uses StratifiedKFold with 5 splits to evaluate k values from 1 to 30, using distance-weighted KNN. The model incorporates error handling for missing values and includes data normalization through MinMaxScaler. It also implemented proper visualization of cross-validation results through error bar plots, making it easier to interpret the

model's performance across different k values.
Doing this also helps remove less informative or extremely rare words and focus on the most meaningful and distinctive features. It was observed that the performance can degrade if k is too large, as the inclusion of distant neighbors may introduce noise and dilute the influence of relevant points.

To enhance the model's performance, several strategies could be implemented. These include additional strategies such as more feature importance analysis to understand which features contribute most to the classification. We could have also implemented other distance metrics beyond the default Euclidean distance. Adding feature selection techniques like PCA or SelectKBest to reduce dimensionality could have been included as well, to improve KNN model results. Incorporating a confusion matrix analysis to understand which genres are most often misclassified could have been beneficial as well.

5. Decision Tree

The approach for our Decision Tree model was using data that was properly preprocessed. We ensure the appropriate measures for label encoding of the categorical features in our dataset and verify the standardization of the numerical attributes to properly ensure that the model receives the required input for optimal results. The dataset utilized the standard 80/20 split ratio for dividing between the training and the testing subsets. A simple Decision Tree classifier was tested to provide a baseline of performance. We utilized cross validation which had to be done in order to assemble the optimal model. The average accuracy that was obtained from the model had an highest accuracy of approximately 52.0% and average accuracy of approximately 43.2%.

In order to ensure the best plausible results for the model, the hyperparameter tuning was performed using GridSearchCV which essentially explores the different combinations of the parameters of the decision tree. The reason for using GridSearchCV was to carry out the multiple combinations in a systematic manner. To be more specific, our hyperparameter tuning process including varying the values of the following:

- Criterion ⇒ (Gini, Entropy)
- Max_Depth ⇒ (10,20,30,None)
- Min_Samples_Split ⇒ (2,5,10)
- Min_Samples_Leaf ⇒ (1,2,,5,10)

Upon evaluating these different parameters, we concluded that the optimal approach for the decision tree would be the following parameters.

- Criterion ⇒ **Entropy**
- Max_Depth ⇒ **10**
- Min_Samples_Split ⇒ **2**
- Min_Samples_Leaf ⇒ **10**

All things considered, this Decision Tree model provides a solid framework for using machine learning to forecast musical genres. The structured approach in carrying out pre-processing of the dataset, understanding baseline performance of the model and optimizing the parameters using systematic processes such as GridSearchCV present a methodology that can be expanded for a more sophisticated and elaborate use case in the future.

6. Ensemble modeling (Bagging and Boosting)

To ensure robust evaluation, the data was first divided into training and testing sets using an 80-20 split before the model construction procedure started. The AdaBoost classifier, a boosting method intended to merge weak learners into a powerful ensemble model, was the first model used. The best accuracy was obtained at an alpha of 0.5 after testing a range of learning rates (alpha) from 0.01 to 100 in order to maximize performance. The model was then further refined by varying the number of weak estimators (n_estimators) from 1 to 500; 50 estimators offered the optimal trade-off between accuracy and computational efficiency. The AdaBoost classifier was made to be properly calibrated for the dataset through iterative experimentation with these hyperparameters.

Another boosting method, the XGBoost classifier, was then put into practice using grid search and hyperparameter optimization. To find the best parameters, this involved experimenting with different combinations of n_estimators and alpha. Because it could handle imbalanced datasets and provided regularization options to help reduce overfitting, XGBoost was very beneficial. The model effectively discovered patterns in the data by encoding the target variable into numerical categories. This allowed it to achieve high classification accuracy and robust generalization capabilities on the test set. These boosting strategies improved prediction accuracy by efficiently utilizing the dataset's attributes.

Future Ideas:

- A recommendation model could be built for specific cross-genre detection recommending users different music that relates to their current tastes and which genres/songs that are similar to the genre they are already looking for. This would allow for the expansion of music they listen to expanding their own interests as well as other artists' outreaches and fan bases. It can be hard to explore new music if a user does not know what to look for, and a cross-genre recommender would be versatile in this unique problem. Many songs are often skipped on current sites that recommend music because they have no relation or

the current algorithms do not work well. This problem could be fixed as music is an ever evolving landscape.

- Looking into the specificities of variables that matter for specific genres would allow for a more advanced learning model to be created with a better computer processor and more time. As the RFC model shows the correlation between variables per genre it would be interesting to see this included in a network for a different tree per type of model and ever evolving network to classify in the cross-genre music creation.
- Also in this project, we tried to cover as many possible classifiers as we can, but with so many data rows, there can be room to develop neural networks on top of preprocessed data to get better results. ANN with different hyperparameters can be considered as next steps for this project.
- Along with this, we can also give an app view to this project with the help of streamlit frontend, where we can ask users to put some choices regarding what kind of parameters they want to listen and we can predict based on those parameters and show some of the songs possible from that genre.
- Another creative idea could be recommendations based on the user's choice of song. We can allow users to choose some of the music albums and rate them and based on those ratings we can give some recommendations.
- One possibility to improve results in terms of accuracy is to distribute data among multiple classes or merge some classes into one. As we can see there are about 10 different classes among which data has been distributed and since we have too many classes, one class can hardly get ample amount of data in cross to features/dimensions it has. This could result in poor classification of classes and could be improved by merging some of classes or removing some of classes or predicting only interested classes.