MSEM 695 Design Project

Patel Jeelkumar

Executive Summary

I am currently in the final semester of my master’s degree in engineering management. I have done my bachelor’s in electrical engineering, and I learned a lot about electrical power systems, control systems, power generation, AC DC motors, etc. After COVID-19, I got my first job as a business development executive which was not related to my bachelor’s degree at all but, I had nothing to do till I found a field job so, I accepted that job and started working as a business development executive in an IT company. I would say, this job was a major turning point in my career as I realized that I like the work of a business analyst. I realized that I must gain technical skills and some of the MBA skills to become a successful business analyst/data analyst. That day I decided to pursue a master’s degree in this field and here I am in the last semester of my master’s degree. This project is based on data analysis of musical data and covers all fundamentals of data analysis. By doing this project, my project portfolio will be stronger than ever.

PROJECT PLAN

**MSEM 695 DESIGN PROJECT –** Python based data classification using audio data.

Jeelkumar Patel

[Jpatel9@uab.edu](mailto:Jpatel9@uab.edu)

**Project Description –** This project aims to develop a machine learning-based system to classify songs into genres (Hip-Hop and Rock) accurately. The system will analyze audio features such as danceability, energy, acoustic Ness, tempo, etc. and metadata to categorize songs into predefined genres.

Using a dataset comprised of songs of two music genres, I will train a classifier to distinguish between the two genres based only on track information derived from Echonest (now part of Spotify). I will first make use of pandas and seaborn packages in Python for sub setting the data, aggregating information, and creating plots when exploring the data for obvious trends or factors I should be aware of when doing machine learning.

Next, I will use the scikit-learn package (It’s an open-source python library that that features various classification, regression, clustering, and dimensionality reduction algorithms.) to predict whether I can correctly classify a song's genre based on features. I will go over implementations of common algorithms such as PCA, logistic regression, decision trees, and so forth.

**Problem you are solving -** The challenge in the music industry is the accurate categorization of songs into genres, which is often subjective and inconsistent. This project seeks to create a more objective and reliable method for genre classification.

**Why it matters -** Accurate genre classification enhances user experience on streaming platforms, aids musicologists in their research, and improves the organization of music databases. It also assists artists and record labels in correctly positioning their music in the market.

Apart from this, another reason behind doing this project is, this project can really level up my portfolio of data analytics projects. I have done several projects on business data analysis excluding data classification-based project, so this project can showcase my data classification skills and, I have never worked with audio data which is also a challenging part for me about this project.

**Who does it impact -** This project impacts streaming services, artists, record labels, musicologists, and general music consumers.

**Steps to accomplish the final project -**

* Data Collection and Preprocessing

Collect a wide range of songs from various genres.

Extract audio features and metadata for analysis.

* Model Development

Design and develop a machine learning model suitable for classifying song genres.

Test and refine the model using a subset of the collected data.

* Model Training and Validation

Train the model with a large dataset to ensure accuracy and generalizability.

Validate the model's performance using standard metrics.

* Implementation and Feedback

Implement the model in a real-world environment.

Gather feedback for further refinement.

**Specific results to be obtained in the final project -**

• A machine learning model capable of accurately classifying songs into genres.

• A comprehensive report detailing the methodology, results, and effectiveness of the classification system.

**Deliverable 1 - Execution items**

**Data Loading and Inspection** [Import necessary libraries, Load the dataset, Inspect basic information about the dataset]

**Data Cleaning and Preprocessing** [Handle missing values, check for duplicates, and remove them, Convert categorical variables into numerical format, Exploratory Data Analysis]

**Generate summary statistics** [Create visualizations using seaborn to identify patterns and trends, Explore relationships between different features and genres]

**Correlation Analysis** [Calculate and visualize correlations between features, Identify highly correlated features, Insights and Documentation]

**Deliverables -** Summarize key findings from EDA, Document potential challenges and considerations for machine learning.

**Deliverable 2 - Execution items**

**Feature Selection -** Identify relevant features for classification, Consider the impact of highly correlated features on model performance,

**Data Scaling -** Normalize or standardize numerical features to ensure fair representation.

**Encoding -** Convert categorical variables into numerical format using appropriate encoding methods.

**Train-Test Split -** Divide the dataset into training and testing sets.

**Deliverables** **-** Document preprocessing steps and reasons behind decisions made.

**Deliverable 3 - Execution items**

**Model Selection -** Choose appropriate classification algorithms (e.g., PCA, logistic regression, decision trees), Implement baseline models.

**Hyperparameter Tuning -** Fine-tune hyperparameters using techniques like grid search or random search.

**Model Training -** Train models on the training dataset.

**Model Evaluation -** Evaluate models using the testing dataset, Measure performance metrics (accuracy, precision, recall, F1-score), Visualize and interpret results using confusion matrices.

**Comparison and Selection -** Compare performance across different algorithms, Select the best-performing model.

**Deliverables –** A Jupyter Notebook file including all the code for model selection, training, and evaluation with detailed comments/explanation for each code.

**Deliverable 4 - Execution items**

Discuss challenges encountered during the project, Provide recommendations for future improvements.

**Deliverables –**

1. Well-organized summary report including all required information of the different sections of the project, Python model used for the classification, accuracy analysis of the model, result analysis, future scope, challenges faced, resources used.
2. The final Jupyter Notebook file including the final code, data visualizations and explained comments for each code section.

**DELIVERABLE 1**

**Summary of EDA (Exploratory Data Analysis) Results:**

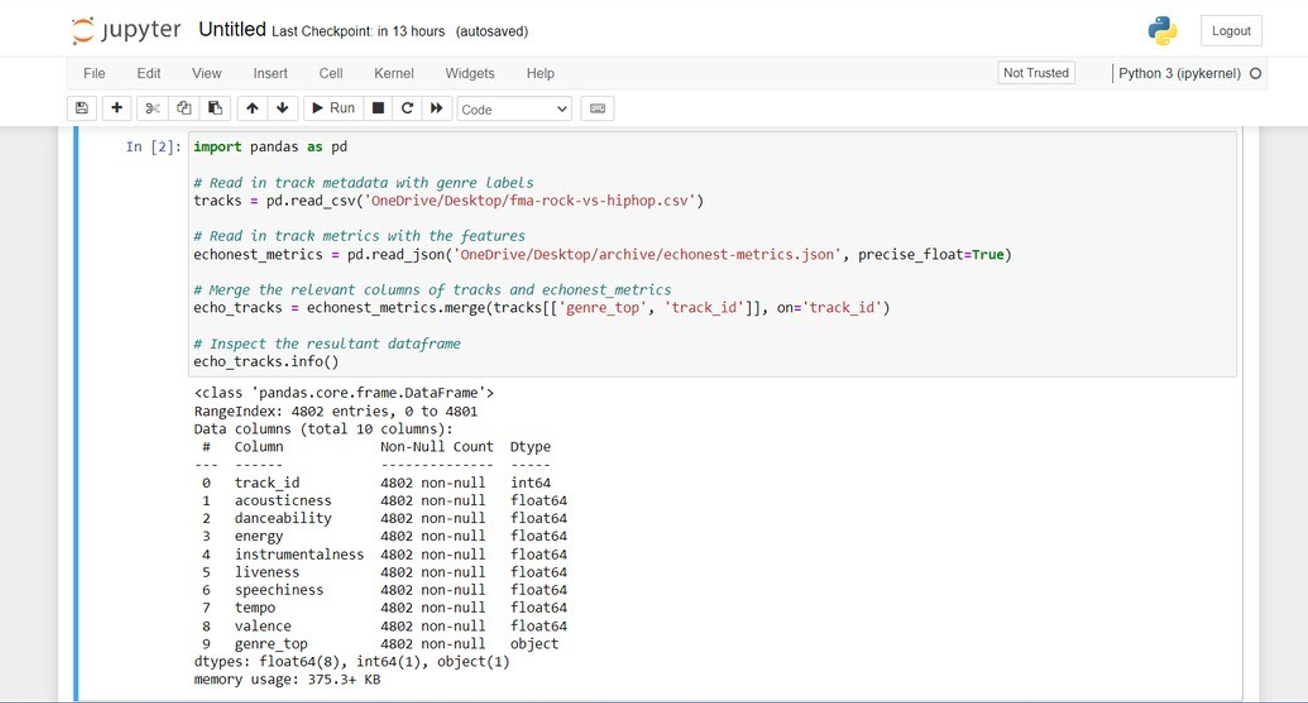
**Data Loading and Inspection** [Import necessary libraries, Load the dataset, Inspect basic information about the dataset]

To begin with, I have loaded the data about music tracks alongside the track metrics compiled by The Echo Nest. I have another dataset that has musical features of each track such as danceability and acoustic Ness on a sale from -1 to 1. These exist in two different file formats which are CSV (popular file format for tabular data) and JSON (common text-based file format used for data transmission in web applications). I have provided the link for both databases below from where one can download the data sets easily.

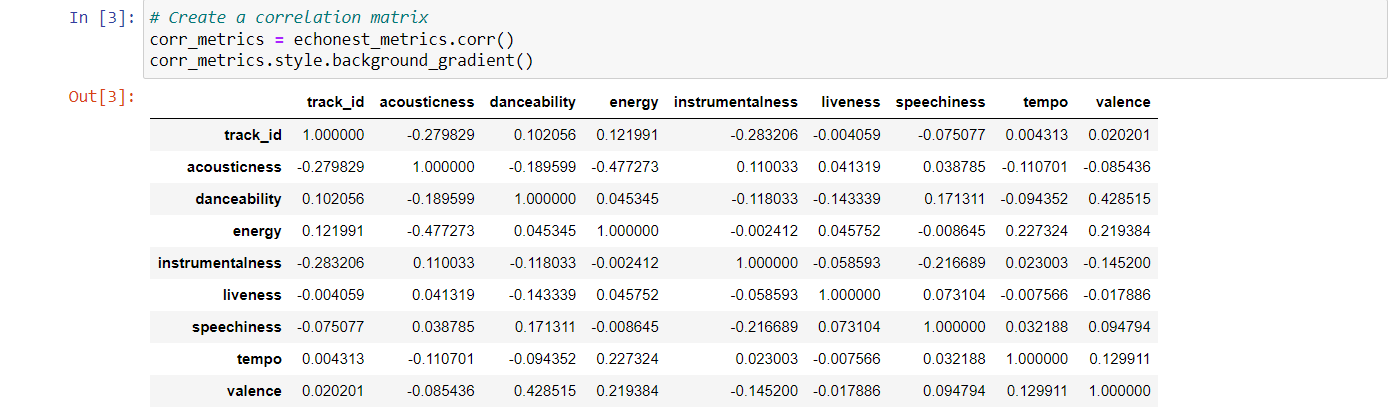
Fma-rock-vs-pop.csv - [link](https://www.kaggle.com/datasets/veronikafilippou/fmarockvshiphop?resource=download)

Echonest-metrics. json - [link](https://www.kaggle.com/datasets/veronikafilippou/echonestmetricsjson/discussion)

After loading the data, I have merged the both datasets using the matching “track\_id” values and kept “track\_id” and “genre\_top” columns from tracks data. The next step is to provide general information of the this merged data which you can in below output.



As we can see, there are total 4802 entries and total 10 columns/features in the new merged data named as “echo\_tracks”.   
**Feature Distribution:** The two genres (Rock and Hip-Hop) differ in how they distribute auditory elements including danceability, energy, and tempo. Hip-hop tracks, for example, may be faster and more danceable than rock tunes.  
**Correlation Analysis:** The performance of the classification models may be impacted by some features that show substantial correlations. For instance, a positive correlation between energy and tempo may suggest problems with multicollinearity. Typically, we avoid using features that have strong correlation with others because of these two reasons. 1. To keep model simple and improve interpretability 2. Using less features for data analysis reduces computation time drastically which will further speed up the analysis.

I have provided the results of correlation analysis below. 

As we can see, there is no significant correlation between any of the features of our data.   
**Outliers:** The feature values of some songs may contain outliers, which may have an impact on the performance and training of the model. It's critical to recognize and respond to these outliers correctly. As we can see in the information provided above, there zero null values in our data set.

**Potential challenges and considerations**  
**1**. **Feature engineering:** feature engineering means managing existing features(columns) and creating new features using existing features in order to make the data analysis more seamless. It presents a number of potential challenges and considerations for machine learning tasks. Even if the presented dataset has audio features, there may be ways to generate new features or extract existing ones to enhance classification performance.

**2. Handling Missing Values:** It is important to deal with the dataset's missing values in a proper manner. Strategies such as imputation or the exclusion of partial samples may be required, depending on the degree of missingness.  
**3. Model Selection and Evaluation:** To determine which model is most suited for the task, it is necessary to test out different categorization algorithms. But judging models only on their accuracy might not be enough, particularly when there are uneven classes.  
**4. Addressing Imbalanced Classes:** To effectively handle class imbalance, strategies like oversampling, under sampling, or employing ensemble methods like SMOTE (Synthetic Minority Over-sampling Technique) may be required.  
**5. Interpreting Model Outputs:** A key component of model interpretation and reliability is comprehending how the selected model generates predictions. Models explain ability approaches and feature importance analysis are two techniques that can shed light on the model's decision-making process.   
**6. Generalization and Robustness:** It's critical to make sure the trained model performs effectively when applied to new data. Hyperparameter tweaking and cross-validation are two strategies that can assist reduce overfitting and increase model robustness.  
**7. Ethical Considerations:** Throughout the project lifetime, it's critical to keep ethical issues like fairness, transparency, and bias mitigation in mind given the possible societal influence of genre classification algorithms.

**DELIVERABLE 2**

**Splitting the dataset**

**Why use train\_test\_split?**

The primary goal of train\_test\_split is to provide a reliable way to assess how well a machine learning model generalizes to unseen data.

Training Set: Used to fit the model (I am using two models for this project which are Decision Tree and Logistic Regression), allowing it to learn patterns from the provided data.

Test Set: Used to evaluate the model's performance on data it hasn't seen before. This simulates how the model would perform in the real world.

Preventing Overfitting: Overfitting occurs when your model becomes too closely tuned to the training data and can't generalize well to new examples. Splitting your data helps detect signs of overfitting:

If the model performs exceptionally well on the training set but poorly on the test set, it's likely overfit.

**How to use train\_test\_split: -**

Code 🡪 from sklearn. model\_selection import train\_test\_split

Sklearn is an abbreviation used for the very popular python library Scikit-Learn. It provides many classification, clustering, and regression algorithms which one uses to create complex models.

**Prepare your data:**

X: Represents your feature data (the input variables to your model).

y: Represents your target data (the output we are trying to predict).

**Splitting the data:**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

test\_size: Specifies the proportion of the dataset to be used for the test set (e.g., 0.25 means 25% for testing, 75% for training).

random\_state: Controls the randomness of the split. Setting a fixed random\_state ensures your results are reproducible.

**Key considerations:**

Dataset Size: For smaller datasets, consider a larger test\_size to get a more reliable evaluation. For very large datasets, a smaller test\_size may suffice.

Data representativeness: Ensure that both the training and test sets are representative of the overall distribution of your data.

A screenshot of a computer code

Description automatically generated

**Normalization data using Standardization method.**

Normalizing data in machine learning refers to the process of scaling individual features to a standard range. The goal is to ensure that all features contribute equally to the model training process and to prevent one feature from dominating due to its larger scale. Normalization typically involves transforming the data so that it has a mean of 0 and a standard deviation of 1, or scaling it to a specific range, such as [0, 1].

Here are two common methods for normalizing data:

Z-score normalization (Standardization) (I am using this method)

Min-Max scaling.

A close-up of a computer screen

Description automatically generated

Principle Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in machine learning and statistics. The primary goal of PCA is to transform the original features of a dataset into a new set of uncorrelated variables called principal components. These components are ordered by the amount of variance they capture, with the first component capturing the most variance and subsequent components capturing less.

A screenshot of a computer

Description automatically generated

If we look at the result, which showcases the variance of each feature with the target variable in terms of percentage in descending order. Let’s take the first number which is 0.23 means the first principal component (PC1) is highly impacting (about 23%) the target variable.

I have visualized this result to understand it better.

A screenshot of a graph

Description automatically generated

Jupyter Notebook file to access code - [Link](http://localhost:8888/notebooks/Design_project.ipynb)

PCA, You tube video - [link](https://www.youtube.com/watch?v=8klqIM9UvAc&t=649s)

**DELIVERABLE 3**

**Projecting on to our features**

Below provided image showcases the code and result of PCA.

A screenshot of a computer

Description automatically generated

Projecting on to our features: - This phrase refers to the process of applying the PCA transformation to the original dataset. In the context of PCA, "projecting" means transforming the data from its original high-dimensional space into a lower-dimensional space represented by principal components.

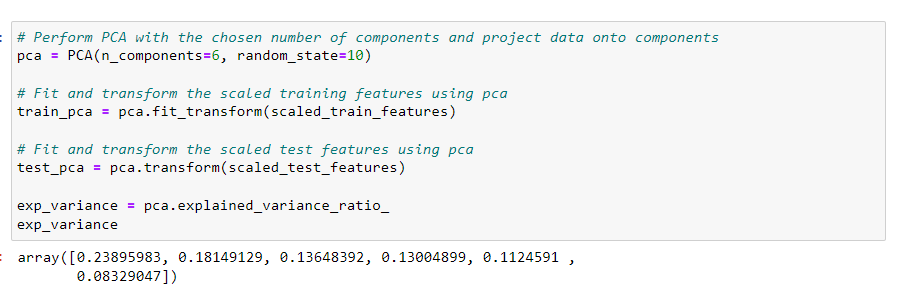
We saw from the image that 6 features can explain 85% of the variance.

Before applying PCA, it's common to analyze the variance explained by each principal component. The plot mentioned here likely represents the cumulative explained variance against the number of principal components. It suggests that by using only the first 6 principal components, approximately 85% of the total variance in the data can be retained. This implies that these 6 components capture the most significant information in the original dataset.

Indexing starts at 0: - This is a reminder that in programming, especially in languages like Python, indexing typically starts at 0. So, when referring to the "6 features," it means features with indices 0 to 5.

Use 6 components to perform PCA: - Given the insight from the plot, the decision is made to perform PCA with only the first 6 principal components. This involves calculating these components and transforming the original dataset to a new representation that retains the essential information captured by these 6 components.

Reduce the dimensionality of our train and test features: - Once the PCA is applied to the training dataset (features), the same transformation is then applied to the test dataset. This ensures consistency between the training and test datasets in terms of dimensionality reduction.

Now, I have performed PCA again but with number of components = 6.   


Here we can see that the number pf principal components are reduced to 6 which were 8 in the previous PCA. Below is the visualiztion of the principal components variance.

A graph of blue bars

Description automatically generated with medium confidence

**Data Modeling**

In this project, I am going to compare two different data models and use whichever is performing the best.

**Model 1: - Decision tree**

Train a decision tree to classify genre: - This step involves using a decision tree algorithm to build a model that can classify songs into different genres. Training a model refers to the process of providing it with a labeled dataset (input features along with corresponding genre labels) so that it can learn patterns and relationships between features and the target variable.

Lower-dimensional PCA projection of the data: - Before training the decision tree, the data is preprocessed using Principal Component Analysis (PCA), as mentioned in the previous section. The dataset is projected onto a lower-dimensional space using the first 6 principal components, as determined earlier. This reduced representation of the data is used as input for the decision tree model.

Decision tree algorithm: - Decision trees are a type of machine learning algorithm used for classification and regression tasks. In this case, it's employed for genre classification. A decision tree operates by recursively splitting the dataset into subsets based on the most significant features at each step. The goal is to create a tree-like structure where each leaf node represents a predicted genre.

Rule-based classification: - Decision trees are rule-based classifiers, meaning that they make decisions by applying a set of rules based on input features. In the context of music genre classification, these rules would be learned during the training phase. For example, the decision tree might learn that certain combinations of features (possibly derived from the PCA projection) are indicative of specific music genres.

Interpretability and Visualization: - One advantage of decision trees is their interpretability. The model's decision-making process can be visualized as a flowchart, showing the sequence of binary decisions leading to the final classification. This makes it easier for humans to understand and interpret how the model arrives at its predictions.

Below is the code for implementing Decision Tree.

A close-up of a computer screen

Description automatically generated

**Model 2: - logistic regression**

Assessment of Decision Tree Performance: - The section begins by acknowledging that while the decision tree's performance is deemed decent, it cautions against assuming it's the perfect tool for the job. Even if a model performs well, it's crucial to explore and test other algorithms to ensure the best model is selected for the given data.

Introduction of Logistic Regression: - Logistic regression is introduced as an alternative algorithm for classification. Unlike decision trees, which use a tree-like structure of binary decisions, logistic regression makes use of the logistic function (also known as the sigmoid function) to calculate the odds that a given data point belongs to a specific class. Logistic regression is a linear model suitable for binary classification tasks.

Testing Multiple Algorithms: - The project emphasizes the importance of testing multiple algorithms rather than sticking to a single model. This approach is common in machine learning, where the effectiveness of different algorithms is evaluated to find the one that performs best for the specific dataset and problem at hand.

Comparing Models Using Performance Metrics: - To compare the decision tree and logistic regression models, the project proposes using performance metrics such as false positive rate and false negative rate. These metrics are essential in assessing the accuracy of a classifier, particularly in binary classification tasks. A false positive occurs when the model predicts a positive class when it should be negative, and a false negative occurs when the model predicts a negative class when it should be positive.

A screenshot of a computer

Description automatically generated

Here we can see that from the above-mentioned classification report, both models do similarly well, boasting an average precision of 87% each. However, looking at our classification report, we can see that rock songs are well classified, but hip-hop songs are disproportionately misclassified as rock songs. To solve this problem, we must perform data balancing.

Jupyter Notebook file to access code - [Link](http://localhost:8888/notebooks/Design_project.ipynb)

**DELIVERABLE 4**

**Data Balancing**

Both our models do similarly well, boasting an average precision of 87% each. However, looking at our classification report, we can see that rock songs are well classified (see precision parameter in classification report), but hip-hop songs are disproportionately misclassified as rock songs.

Why might this be the case? Well, just by looking at the number of data points we have for each class, we see that we have far more data points for the rock classification than for hip-hop, potentially skewing our model's ability to distinguish between classes. This also tells us that most of our model's accuracy is driven by its ability to classify just rock songs, which is less than ideal.

To account for this, we can weigh the value of a correct classification in each class inversely to the occurrence of data points for each class. Since a correct classification for "Rock" is not more important than a correct classification for "Hip-Hop" (and vice versa), we only need to account for differences in sample size of our data points when weighting our classes here, and not relative importance of each class.

A screenshot of a computer code

Description automatically generated

**Effect of data balancing on the performance of both data models.**

We've now balanced our dataset, but in doing so, we've removed a lot of data points that might have been crucial to training our models. Let's test to see if balancing our data improves model bias towards the "Rock" classification while retaining overall classification performance.

Note that we have already reduced the size of our dataset and will go forward without applying any dimensionality reduction. In practice, we would consider dimensionality reduction more rigorously when dealing with vastly large datasets and when computation times become prohibitively large.

A screenshot of a computer code

Description automatically generated

From the above classification report, we can see the improved precision for Hi-hop (84%) and Rock (82%) labels.

**DELIVERABLE 5**

**Cross – Validation to evaluate model**

Balancing our data has removed bias towards the more prevalent class. To get a good sense of how well our models are performing, we can apply what's called cross-validation (CV). This step allows us to compare models in a more rigorous fashion.

Before we can perform cross-validation we will need to create pipelines to scale our data, perform PCA, and instantiate our model of choice - DecisionTreeClassifier or LogisticRegression.

Since the way our data is split into train and test sets can impact model performance, CV attempts to split the data multiple ways and test the model on each of the splits. Although there are many different CV methods, all with their own advantages and disadvantages, we will use what's known as K-fold CV here. K-fold first splits the data into K different, equally sized subsets. Then, it iteratively uses each subset as a test set while using the remainder of the data as train sets. Finally, we can then aggregate the results from each fold for a final model performance score.

Here I have taken k = 10

A screenshot of a computer program

Description automatically generated

Here we can see the results of cross validation for both models and their scores are 75% and 78% respectively.

Link for Jupyter Notebook file - <http://localhost:8888/notebooks/Design_project.ipynb>