**CS703 4.0 Data Modeling**

*Madeline Lin*

**Task 4.1: Selecting Modeling Techniques**

So far, I have completed the data understanding and preparation phases. These prep work helps ensure that I start the modeling phase exactly where I need to be with minimal omissions and ‘do-overs’.

Prior to discussing modeling techniques, I would like to provide an overview of the definitions, descriptions, and assumptions of the different models I intend to utilize. This will help readers gain an understanding of the models, ultimately enhancing the readability of the main part of the document.

**Deliverable 1: Modeling Definitions and Descriptions**

* Content-based filtering method
  + Prediction Regression Models
  + Simple Linear Regression Model
    - Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine: 1) whether a set of predictor variables do a good job in predicting an outcome variable; 2) which variables in particular are significant predictors of the outcome variable and in what way they indicated by the magnitude and sign of the beta estimates impact the outcome variable.
    - The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b\*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.
  + Linear Regression Model with Interactions
    - Apart from what have been described above for linear regression model, there is an interaction between the independent variables. This means that the effect of one independent variable on the dependent variable depends on the level of another independent variable.
  + Stepwise Model
    - Stepwise regression is the step-by-step iterative construction of a regression model that involves the selection of independent variables to be used in a final model. It involves adding or removing potential explanatory variables in succession and testing for statistical significance after each iteration.
    - There are three types of stepwise regression models. They are forward selection, backward selection and both selection. Forward selection starts from zero variable and adds up the variables until the results are optimal. Backward selection starts from adding all variables and deletes each variable until the results are optimal. Both selection is a combination of the forward selection and backward selection as it starts somewhere in the middle.
  + Decision Tree Model
    - Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.
    - In terms of numerical data, it works by recursively partitioning the input variables into smaller intervals, which can then be treated as categorical variables. It selects a feature and a threshold value for that feature and splits the data into two subsets based on whether the value of that feature is above or below the threshold. In practice, however, decision trees may not perform as well on numerical data as on categorical data. The splitting process can be more complex with numerical data, as there are many possible ways to partition a continuous variable into intervals.
  + Random Forest Model
    - Random forest is an extension of the decision tree algorithm, where instead of using a single decision tree to make predictions, multiple decision trees are used to improve the accuracy and stability of the model. In a random forest, each tree is trained on a subset of the training data, and at each split of the decision tree, a random subset of features is considered. This randomness is intended to introduce diversity in the model, which helps to reduce the chance of overfitting and improve generalization performance. Once all the trees are trained, the predictions of the individual trees are combined to produce the final prediction.
    - Random Forest can work well with categorical and numerical data. In terms of numerical data, it partitions the feature space into rectangular regions, making it effective for handling non-linear relationships between the input features and the target variable. Additionally, random forest can handle missing values and is relatively robust to outliers, which can be common issues in numerical data.
  + Neural Network Model
    - Neural network model is a type of machine learning model which is inspired by the structure and function of a human brain. It consists of interconnected nodes, or neurons that are organized into layers. Each neuron receives input from other neurons and produces an output based on that input, which is then passed on to other neurons in the network. Through a training process, a neural network can learn to recognize patterns and relationships in data, and can be used for classification, regression, and prediction.
    - Neural network model can handle numerical data. It can predict a continuous numerical value based on a set of input features. Compared to regression model, the main advantage of neural network model is its ability to learn complex nonlinear relationships in data, thus can capture more intricate patterns and interactions. However, neural networks can be more computationally intensive to train than regression models, and can require more data to avoid overfitting.
* Collaborative filtering method
  + Similarity Metrics
    - Euclidean Distance
      * The Euclidean distance is a popular similarity metric used in machine learning field. It is a measure of the distance between two points in Euclidean space. The formula of calculating the Euclidean space is:

Point A (a1, a2, a3, …, an)

Point B (b1, b2, b3, …, bn)

d (A, B) =

* + - Pearson Correlation Coefficient
      * The Pearson correlation coefficient is a statistical measure that indicates the strength and direction of the linear relationship between two variables. The Pearson correlation coefficient ranges from -1 to 1, with

-1 indicating a perfect negative correlation, 0 indicating no correlation, and 1 indicating a perfect positive correlation. The formula of calculating Pearson Correlation coefficient is:

where r is the Pearson correlation coefficient,

x and y are the two variables being correlated,

x̄ and ȳ are the mean of x and y

* + - Cosine Similarity
      * Cosine similarity is a measure of similarity between two non-zero vectors in a high-dimensional space. It measures the cosine of the angle between the two vectors and ranges from -1 to 1, with 1 indicating that the two vectors are pointing in the same direction (i.e. identical),

0 indicating that the vector are orthogonal (i.e. completely dissimilar), and -1 indicating that the two vectors are pointing in opposite directions (i.e. negatively correlated). The formula of calculating cosine similarity is:

where (x.y) represents the dot product of the two vectors, which are

x1y1+x2y2+…+xn\*yn

||x||, ||y|| represent their Euclidean norms (i.e. magnitudes), which are and

* + - Jaccard Similarity
      * Jaccard similarity is a measure of similarity between two sets. It is defined as the size of the intersection of the sets divided by the size of the union of the sets. The formula of calculating jaccard similarity is:

J(X,Y) = |X ∩ Y| / |X ∪ Y|

**Deliverable 2: Modeling Assumptions**

* Content-based filtering method
  + Prediction Regression Models
  + Simple Linear Regression Model
    - The model assumes that the relationship between dependent variable and independent variables is linear, meaning that a change in the value of one variable is associated with a proportional change in the value of the other variable; Specifically, assumptions are:
    - Linearity: The relationship between the dependent variable and the independent variables is linear.
    - Independence: The observations in the dataset are independent of each other.
    - Homoscedasticity: The variance of the errors (the difference between the predicted values and the actual values) is constant across all levels of the independent variables.
    - Normality: The errors are normally distributed.
    - No multicollinearity: There is no perfect linear relationship between any of the independent variables.
  + Linear Regression Model with Interactions
    - Apart from what have been described above for linear regression model, there is an interaction between the independent variables. This means that the effect of one independent variable on the dependent variable depends on the level of another independent variable.
    - Stepwise Model
      * The stepwise model selects the most important attributes to include in a regression model. The basic idea is to start with a full model containing all potential attributes, then to iteratively remove or add attributes based on their significance or contribution to the model. Stepwise model still includes the assumptions which were covered in the above for the simple regression model.
      * Additionally, stepwise model can be sensitive to the sample size. Consequently, it is important to have a large enough sample size to ensure reliable estimates of the regression coefficients and to avoid overfitting the model.
    - Decision Tree Model
      * Feature independence: the features used in the decision tree should be independent of each other. That is, they should not be correlated with each other, as this can lead to problems with overfitting and bias in the resulting model.
      * Data quality: the quality of data used to train the decision tree should be good, with minimal missing or erroneous data. Decision trees can be sensitive to noisy or outlier data, so it is important to preprocess the data appropriately.
      * Splitting criterion: the criterion used to select the best feature to split the data should be appropriate for the problem at hand.
      * Three depth: the depth of the decision tree should be appropriate for the complexity of the problem. If the tree is too deep, it may overfit the data and fail to generalize well to new data. If the tree is too shallow, it may underfit the data and miss important patterns or relationships.
      * Sampling: when working with large datasets, it can be useful to use a subset of the data to train each node of the decision tree. This can help to reduce overfitting and improve the accuracy of the resulting model.
    - Random Forest Model
      * As mentioned above, random forest is an extension of the decision tree algorithm by using multiple decision trees to improve the accuracy and stability of the model.
      * Apart from the assumptions of decision tree model, it has an additional assumption of the number of trees. The number of trees should be sufficient to achieve good performance, but not so large as to be computationally expensive or to overfit the data. There is often a trade-off between model complexity and accuracy, which can be addressed through cross-validation and other methods.
    - Neural Network Model
      * Large data sets: neural networks can require a large amount of data to train effectively. This is especially true for deep neural networks, which have many layers of neurons and parameters. It is important to have enough data to avoid overfitting and to achieve good generalization performance.
      * Quality data: the data should be representative of the problem being solved, with minimal missing or erroneous data.
      * Activation functions: neural networks rely on activation functions to introduce nonlinearity into the model. Since my problem involves predicting the streaming time based on audio features, I can use a neural network model with a single output neuron that has a linear activation function in the output layer.
      * Network architecture: this includes the number of layers, the number of neurons in each layer, and the type of connections between the layers. Choosing the right architecture can be a complex process that requires experimentation and tuning.
      * Initialization: the initial values of the weights and biases in the neural network can have a significant impact on the performance of the model. It is important to choose appropriate initialization values to avoid getting stuck in local optima during training.
      * Regularization: neural networks can be prone to overfitting, especially for large and complex models. Regularization techniques, such as dropout, weight decay, or early stopping, can be used to mitigate this problem and improve generalization performance.
* Collaborative filtering method
* Similarity Metrics
  + - Euclidean Distance
      * Euclidean space: the data points lie in a Euclidean space, where the distance between two points is the straight-line distance between them. This is a fundamental assumption of the metric, and if the data points do not lie in a Euclidean space, the metric may not be appropriate.
      * Continuous variables: the variables used to compute the distance are continuous. If the variables are categorical or ordinal, or if they are measured on different scales, the metric may not be appropriate. In my case, audio features are continuous.
      * Independence: the variables used to compute the distance are independent of each other. If the variables are highly correlated or dependent, the metric may not be appropriate.
      * Linear relationships: the Euclidean distance metric assumes that the relationships between the variables are linear. If the relationships are nonlinear, the metric may not be appropriate.
      * Scale: the Euclidean distance metric is sensitive to the scale of the variables, and if the variables are measured on different scales, the metric may be biased towards the variables with larger scales. Therefore, it is often necessary to standardize the variables before using the Euclidean distance metric.
    - Pearson Correlation Coefficient
      * Linearity: the Pearson correlation coefficient assumes that the relationship between the two variables is linear. If the relationship is nonlinear, the correlation coefficient may not be appropriate.
      * Normality: the Pearson correlation coefficient assumes that the two variables are normally distributed. If the variables are not normally distributed, the correlation coefficient may not be appropriate.
      * Homoscedasticity: the Pearson correlation coefficient assumes that the variances of the two variables are equal. If the variances are unequal, the correlation coefficient may be biased.
      * Independence: the Pearson correlation coefficient assumes that the two variables are independent of each other. If the variables are dependent or have a cause-and-effect relationship, the correlation coefficient may not be appropriate.
      * Outliers: the Pearson correlation coefficient is sensitive to outliers, which can have a strong influence on the correlation coefficient. Therefore, it is important to check for outliers and to consider their impact on the correlation coefficient.
    - Cosine Similarity
      * Vector representation: two objects being compared are represented as vectors in a high-dimensional space. This is a fundamental assumption of the similarity measure, and if the objects cannot be represented as vectors, the similarity measure may not be appropriate.
      * Positive values: the values in the vectors are non-negative. If the values can be negative, the similarity measure may be biased or inappropriate.
      * No zero-norm vectors: the vectors being compared are not zero-norm vectors, that is, they are not vectors with all entries equal to zero. If the vectors are zero-norm vectors, the cosine similarity may not be well-defined.
      * Scale invariance: the cosine similarity is scale-invariant, meaning that it is insensitive to the magnitude or scale of the vectors. This can be an advantage in some applications, but it can also be a disadvantage if the magnitude or scale of the vectors is important for the problem being addressed.

**Deliverable 3: Defined Modeling Techniques**

There are many different modeling techniques, from which I need to determine which ones are the best for my project business goal – exploring the behind algorithm of Spotify recommendation system and optimizing the recommendation system to benefit Spotify users.

* Content-based filtering method
* For this method, my data mining goal is to find the optimal prediction model based on the audio features of the tracks.
* This should be based on supervised learning algorithm. All the 12 audio features (i.e. danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo) are the independent variables (i.e. x variables) and msPlayed is the dependent variable (i.e. y variable). I will build several prediction regression models and find the one with the least Root-mean-square-deviation (“RMSE”), which is a metric for evaluating the performance of regression models by measuring the difference between predicted values by a model and the actual observed values.
* The following is a summarized table indicating my intentions to use the modeling technique based on each model’s definitions, descriptions and assumptions.

|  |  |
| --- | --- |
| *Content-based filtering method* |  |
| Simple Linear Regression Model | Simple Linear Regression Model is a fundamental and widely used model.  The assumptions of this model align with my intuition, as there should be a linear relationship between the streaming time of a song and its audio features.  I plan using this modeling technique. It directly correlates to my data mining goal. |
| Linear Regression Model with Interactions | Apart from the points made in the simple linear regression model, some of the audio features will be correlated. For instance, in most of the cases, a song with higher energy will have higher liveness.  I plan using this modeling technique. |
| Stepwise Model | I plan using this modeling technique as I think it may be beneficial to use machine learning algorithms to determine which audio features and which interactions of audio features I should include instead of me manually figuring it out.  The assumptions of this model also align with my above point. |
| Decision Tree Model | I plan using this modeling technique to solve the numerical data. I would like to see how the decision tree model can help me predict the streams by creating a tree-like flowchart based on the audio features data. However, I will keep in mind that this model technique may not be the best option compared to regression model. For example, I think some of the assumptions may not be aligned. The audio features are not independent of each other. |
| Random Forest Model | As random forest model is an extension of the decision tree models, this may also not be the best choice.  I still plan using this modeling technique to solve the numerical data. While it is true that traditional decision tree models may not perform as well on numerical data compared to categorical data, random forest can handle both types of data effectively. |
| Neural Network Model | I plan using this modeling technique. My intuition is that there probably will not be a linear relationship between the audio features and my preference/how much I like a song. I use the stream as the metric to measure my preference of a song as generally I will stream more if I like a song more. While it might be a subjunctive matter to tell what kinds of music I like, I agree that using machine learning algorithm of predicting streams through audio features is a good approach.  Based on the Neural Network Model assumptions, it includes a number of layers where a number of neurons embedded within. Neural network model may be a good technique as it is a type of machine learning model that mimics the structure and function of a human brain.  I have never had an opportunity to know more about this modeling technique in my past data science projects. I would love to have a try this time. |
| *Collaborative filtering method* |  |
| Euclidean Distance | Based on my understanding, in my case, first, audio features of a track can are identified by a unique set of coordinates; thus they are in the Euclidean space. Second, audio features (apart from mode is binary) are continuous. Third, the target track and all tracks in Spotify ­are not highly correlated. Fourth, the relationships between variables should be linear. Audio features are quantified between 0 to 1 in general, but there are a few audio features (i.e. key, loudness, tempo) that are not the same scale as others, I might need to adjust. Overall, Euclidean space is an appropriate metric to use. |
| Pearson Correlation Coefficient | I plan using this modeling technique. Based on my understanding, in my case, first, the relationships between two variables is linear. Second, in terms of the homoscedasticity, the variances of audio features should be similar as they are quantified between 0 to 1 after I adjusted some of the audio features. Third, the target track and all tracks in Spotify ­are not highly correlated. Fourth, there should not be any outliers in my dataset as I’ve conducted the data preparation phase. Overall, Pearson correlation coefficient is an appropriate metric to use. |
| Cosine Similarity | I plan using this modeling technique. Based on my understanding, the first assumption works perfectly for my case. Each track has several audio features, which can be regarded as a vector in a high-dimensional space. As for the second and third assumptions, while the values of the vectors are generally non-negative, the attribute representing loudness is negative. To address this, I may consider adjusting the loudness attribute by adding a positive value to ensure that all values are non-negative. Finally, the fact that Cosine similarity is scale-invariant is a great advantage, as it means there is no need for me to normalize audio features that do not have a scale between 0 and 1. Overall, Cosine similarity coefficient is an appropriate metric to use. |
| Jaccard Similarity | I suppose this modeling technique does not apply for my dataset. Spotify launched a playlist initiative, Blend, where any user can invite any user to generate a playlist wherein the two users’ tastes are combined into one shared playlist[[1]](#footnote-1). Spotify may use the Jaccard Similarity to find the user B song set which is the most similar to the user A as Spotify is able to retrieve all sets from subscribers. It would be great if I have the datasets of other users; however, since I do not, I won’t be able to utilize Jaccard similarity technique. |

Overall, apart from the Jaccard Similarity, I plan using all the above modeling techniques for my following analysis.

**Task 4.2: Designing Tests**

**Deliverable: Test Design Document**

When designing tests, it is crucial to carefully consider my training, testing, and validation strategy in order to avoid introducing bias into the data.

Training data enables the model to learn from the patterns and relationships in the data, and to make accurate predictions on new, unseen data. The testing data is used to evaluate the performance of the model after it has been trained and tuned on the training data. The testing data is typically held out from the training process, and the model is evaluated on this dataset to see how well it generalizes to new, unseen data. Apart from training and testing split, in order to fine-tune the model’s hyperparameters during training and ensure the model is not overfitting to the training data, we usually have a validation data.

I will split 80% of my data for training and the remaining 20% for testing. Of my 80% training data, I will further split 20% of my training data into validation data and kept the 80% as training data.

So, 64% training data, 16% validation data and 20% testing data.

In summary,

First, I will train the model on the training dataset.

Then, I will fine-tune the model and its hyperparameters using the validation dataset.

Finally, I will evaluate the final model on the testing set to estimate its performance on new, unseen data.

Overall, this can help avoid introducing bias into the data as much as possible.

**Task 4.3: Building Models**

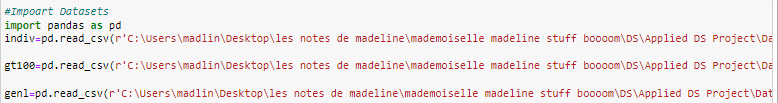
**Deliverable 1: Parameter Definitions/Settings**

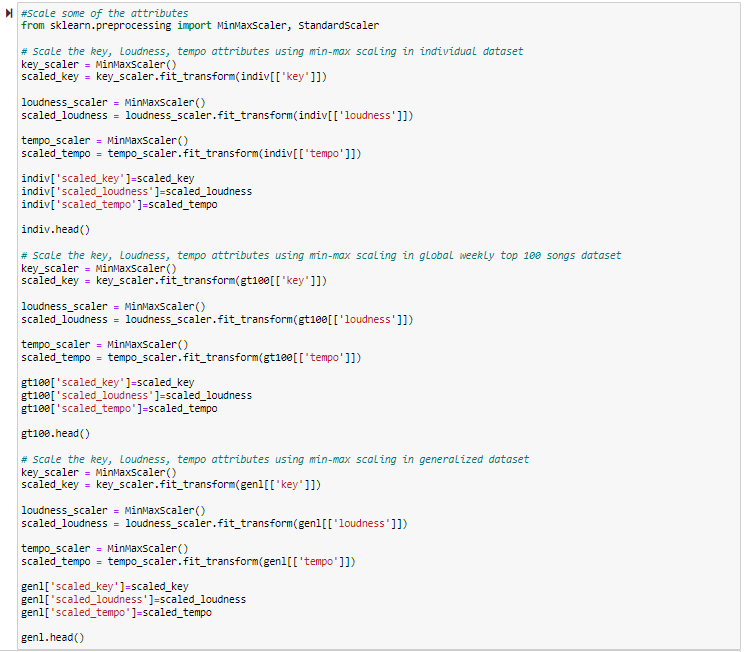
Regression models use a numerical optimization algorithm to find the best coefficients that minimize the prediction error, and these algorithms can be sensitive to the scale of the input features. Therefore, it is generally a good idea to scale the variables in a regression model to the same range. Moreover, as mentioned in modeling assumptions section, the Euclidean distance metric is sensitive to the scale of variables, Pearson correlation coefficient requires that the variables are normally distributed and is sensitive to outliers, and Cosine similarity metric requires positive values of the variables, I decided to perform scaling the some of the input features for across all three datasets. Based on my research, there are several methods with regard to rescaling data. They are min-max scaling, standardization, robust scaling, log transformation, power transformation and so forth. Min-max scaling method scales the data to a fixed range (usually 0 to 1), which satisfies the assumptions of all similarity metrics.

The below is a summary of how I adjusted the parameters settings. I mostly adjusted three audio features, which are key, loudness and tempo using the min-max scaling method.

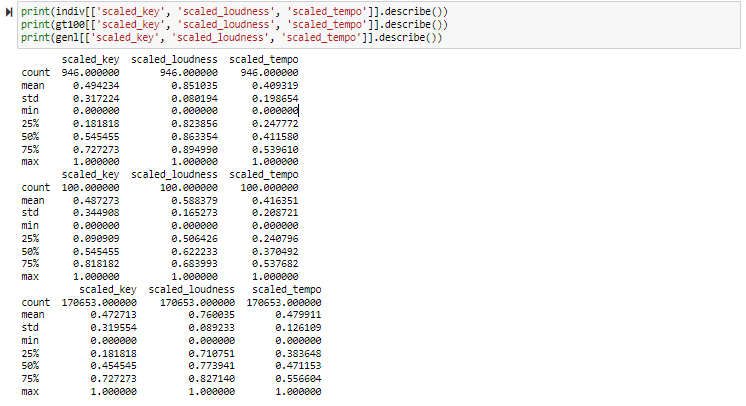
|  |  |  |
| --- | --- | --- |
|  | Description of the Fields | Scaling the data |
| *Audio Features* | | |
| danceability | Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable. | This attribute ranges from 0 to 1.  No further scaling action needed. |
| energy | Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy. | This attribute ranges from 0 to 1.  No further scaling action needed. |
| acousticness | A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic. | This attribute ranges from 0 to 1.  No further scaling action needed. |
| valence | A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry). | This attribute ranges from 0 to 1.  No further scaling action needed. |
| instrumentalness | Instrumentalness predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks. | This attribute ranges from 0 to 1.  No further scaling action needed. |
| key | The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C/D, 2 = D, and so on. | I used min-max scaling to scale the values to the range [0,1] by subtracting the min value and dividing by the range. |
| liveness | This value describes the probability that the song was recorded with a live audience. A value above 0.8 provides strong likelihood that the track is live. | This attribute ranges from 0 to 1.  No further scaling action needed. |
| loudness | The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db. | I used min-max scaling to scale the values to the range [0,1] by subtracting the min value and dividing by the range. |
| mode | Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0. | No further scaling action needed. |
| speechiness | Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks. | This attribute ranges from 0 to 1.  No further scaling action needed. |
| tempo | The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration. | I used min-max scaling to scale the values to the range [0,1] by subtracting the min value and dividing by the range. |
| duration\_ms | Duration\_ms is the duration of a track in milliseconds. | Duration of a track is irrelevant for predicting the streams. Thus, I won’t include this attribute in my model. |
| popularity | Spotify calculated the popularity of a track based on: 1) total streams of a track, 2) how recently a track has been played, 3) the frequency that a track has been played. | As msPlayed (i.e. streaming time) has been determined as the dependent variable, I excluded the popularity. |
| *General Features* | | |
| artist | Artist represents the artist name of the track. | N/A |
| track | Track represents the track name of the song. | N/A |
| album | Album represents the name of the album that a track belongs to. | N/A |
| msPlayed | MsPlayed represents the streaming time of a track in milliseconds. | N/A |
| uri | The Spotify URI for the track. Uniform resource indicator is a link that you can find in the Share menu of any track, album, or Artist Profile on Spotify. | N/A |
| *Other Features* | | |
| endTime | EndTime is the time when I end listening to the track. For example, 2020-01-16 23:31 means that I ended listening to a certain song at 23:31 on January 16, 2020. | N/A |
| rank | Rank represents the ranking of the tracks by a certain week. | N/A |
| source | Source is the music label of the track. | N/A |
| peak\_rank | Peak\_rank represents the highest rank of a track. | N/A |
| previous\_rank | Previous\_rank represents the previous rank of a track. | N/A |
| weeks\_on\_chart | Weeks\_on\_chart means how many weeks a track stays on chart. | N/A |
| year | Year indicates which year the track is released on Spotify. | N/A |
| released\_date | Released\_date indicates which date or year the track is released on Spotify. | N/A |

*Code*





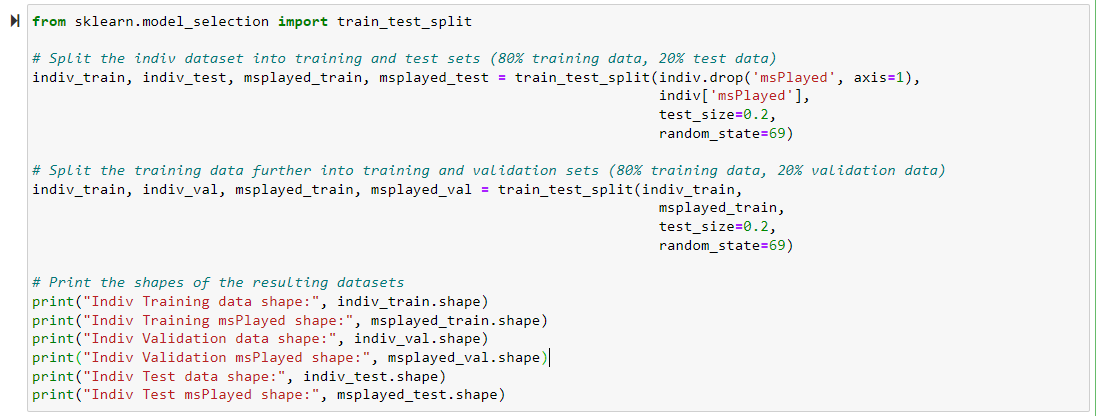
*Output*



As demonstrated in the above output screenshot, the key, loudness, and tempo attributes have been rescaled from 0 to 1.

Additionally, I performed the train/validation/test data split before modeling for Individual Dataset. Note that there is no need for train/validation/test data split for Global Weekly Top 100 Songs Dataset as I will use this dataset purely as the test dataset for my optimal stream prediction model. Also, since Generalized Dataset is used to apply unsupervised learning method, I can use the entire dataset to find the similar songs without the split.

*Code*



\*Random seed is used to ensure that the same samples are selected each time the algorithm is run, which in turn ensures the results are reproducible. The value of

random seed is random but it should stay consistent among different models.

*Output*



Based on the output screenshot, we can see that the Individual Dataset has been split into three sub datasets, which are individual training dataset, individual validation dataset, and individual test dataset. For each sub dataset, I separated the msPlayed variable as a target variable array because this is the dependent variable which I need to predict based on the audio feature variables. The individual training data shape (604, 19) means that this data frame has 604 observations (i.e. rows) and 19 attributes (i.e. columns). The individual training msPlayed shape (604, ) means that there are 604 values in the msPlayed array, one of each observation in the training dataset. Same interpretations for the rest four data shapes.

At this point, the data should be ready for modeling.

**Deliverable 2: Model Descriptions**

To my understanding, I have documented the type of model will be used and why I choose it in the Section Selecting Modeling Techniques, in this section, I would like to elaborate the explanations of inclusion of variables and why I choose these variables to be included in my model, together with how the model should be interpreted, as well as the difficulties encountered in the modeling process, or any important items of note if applicable.

* Content-based filtering method
* Prediction Regression Models
  + - Simple Linear Regression Model
      * As this is a simple linear regression and my first model, I decided to include all 11 audio features. They are danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo. MsPlayed is the dependent variable.
      * In terms of how to interpret the model, let’s use a simple linear regression model as an illustration.

y = b0 + b1 \* x1 + b2 \* x2

where:

y is the MsPlayed

x is one of the audio features (e.g. energy, speechiness)

b0 is the intercept (the value of MsPlayed when energy=0 and speechiness = 0)

b1, b2 are the slopes (the change in MsPlayed for a one-unit increase in energy/speechiness)

In my case, it means that I will stream around b0 miliseconds even the tracks with 0 value of energy and speechiness. Holding the speechiness feature constant, one-unit increase in energy of a track will lead me to stream b1 milliseconds more. Same for energy feature.

* + - Linear Regression Model with Interactions
      * Some audio features will have correlation with each other. Generally, songs with high danceability would have high energy, liveness, tempo. Songs with high instrumentalness tend to have fewer or no vocals while songs with high speechiness typically feature more vocals or spoken words. Therefore, instrumentalness and speechiness are often negatively correlated. Songs with major modes (i.e. mode = 1) have brighter and more upbeat sounds, which may have high valence and high energy; while songs with minor modes (i.e. mode = 0) have darker and more melancholic sounds, which may have low valence and low energy.
      * It is hard for me to figure out all interactions for all audio features. I decided to include a few interactions based on my intuition.
        + Danceability ~ Energy ~ Liveness ~ Tempo
        + Instrumentalness ~ Speechiness
        + Mode ~ Valence ~ Energy
      * Apart from interpretations in simple linear regression model, the interpretation with variables with interactions is as below.

y = b0 +b1 \* x1 + b2 \* x2 + b3 \* (x1 \* x2)

where b3 is the coefficient for the interaction term (x1 \* x2)

In my case, it means that the change in the effect of energy on streaming time for a one-unit increase in speechiness will be b3 milliseconds.

* + - Stepwise Model
      * I do not need to decide which variables to be included, as the stepwise model will either start from zero variable and adds up the variables or start from adding all variables and delete each variable or will start somewhere in the middle, until the results are optimal.
      * The interpretation of a stepwise model is similar to that of a standard linear regression model apart from that the variables included in the model are selected automatically by the algorithm rather than being chosen on my own.
    - Decision Tree Model
      * I do not need to decide which variables to be included, as the decision tree model can automatically select the most informative variables to create a tree-like structure that can be used to make predictions. I did some research for how to select variables for the decision tree. Generally, I can use feature selection techniques, such as information gain or Gini impurity to help identify the most informative audio features to include in the model.
      * In terms of how to interpret the model, I will start looking at the root node of the tree, which represents the first decision based on one of the input variables. From there, I will follow the path down the tree based on the values of the input variables until I reach a leaf node, which represents a predicted value of the target variable.
    - Random Forest Model
      * As random forest model is an ensemble learning method that combines multiple decision tree models to make predictions. When making a prediction with a random forest model, each decision tree in the forest independently predicts the target variable based on the input variables. The final prediction is then made by aggregating the predictions from all the decision trees.
      * The interpretations of a random forest model is similar to that of a decision tree model apart from that it additionally involves understanding how the individual decision trees combined to make the final prediction. This will involve examining the importance of each input variable for making predictions, as well as visualizing the structure of the decision trees in the forest.
    - Neural Network Model
      * The neural network model collects interconnected artificial neurons that can learn from data and make predictions. It consists of several layers (i.e. input layer, hidden layer and output layer) of neurons that are connected by weights. As this is my first time to try neural network model, I would like to keep it simple – I will include all audio features without interactions for this model.
      * In terms of how to interpret the model, let’s use a simple neural network model as an illustration.

h = f (Wx + e1)

y = Uh + e2

where:

x is a vector of input audio feature (i.e. danceability)

h is a vector of hidden units

y is a vector of output unit (i.e. MsPlayed)

f is an activation function applied to the weighted sum of inputs to each

hidden unit

W and U are weight matrices for the connections between the input layer

and the hidden layer, and the hidden layer and the output layer,

respectively

e1 and e2 are bias vectors for the hidden layer and the output layer,

respectively

In my case, first, the neural network model will try to learn how the energy of a song relates to my streaming by adjusting the weights (i.e. W) of the connections between the energy and the hidden layer, and the weights (i.e. U) of the connections between the hidden layer and the output layer, as well as the biases, to minimize the difference between the predicted MsPlayed value and the actual MsPlayed value for a given input value of energy.

* Collaborative filtering method
* Similarity Metrics
  + - Euclidean Distance
    - Pearson Correlation Coefficient
    - Cosine Similarity

I do not need to decide which variables to be included as these are all unsupervised learning methods.

The above three metrics are all similarity metrics that are used to measure similarity between two vectors. They differ in calculation methods, range of values and interpretation. In general, the interpretation is that the smaller distances between two tracks, more similarities are shared between two tracks. As for Euclidean Distance, it represents the straight-line distance between two points in a multidimensional space. A smaller Euclidean distance indicates that two tracks are closer to each other, which a larger distance indicates that they are further apart. As for Pearson correlation coefficient, a positive correlation closer to 1 indicates that two tracks tend to be more similar. As for Cosine Similarity, it measures the angle between two vectors; a positive cosine similarity closer to 1 indicates that two tracks tend to be more similar.

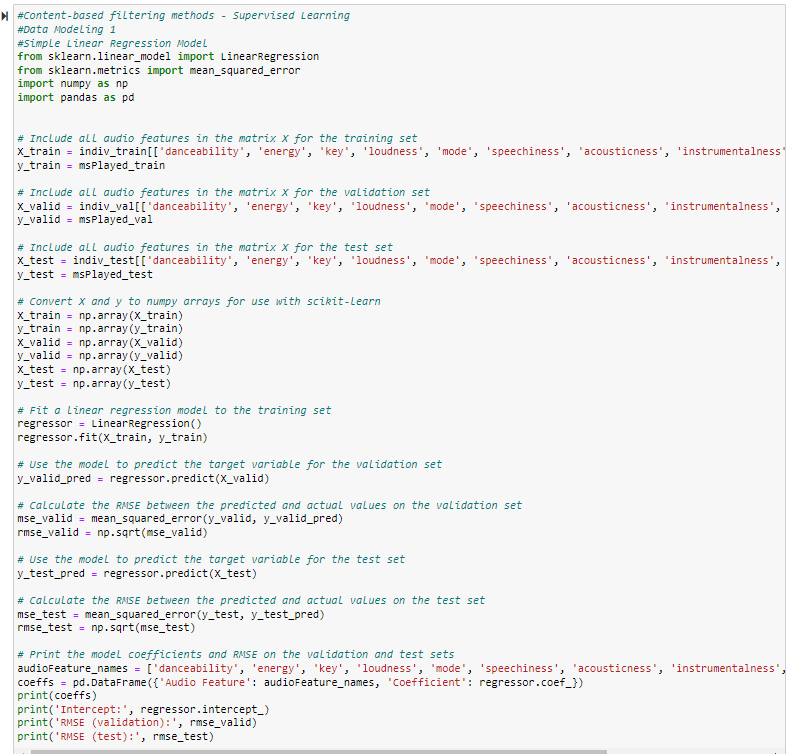
As far as I am concerned, I encountered some difficulties about the interpretations of the random forest model and neural network model. However, based on my research, I put down my understandings of how to interpret these two models as shown above. Additionally, there are not any important items need to be addressed for now.

**Deliverable 3: Data Models**

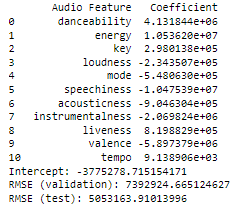
In this data models deliverable, I will present the code and the output with explanations. I will provide the further assessments and interpretations in the Assessing Models section.

* Content-based filtering method
* Prediction Regression Models
  + - Simple Linear Regression Model

*Code*

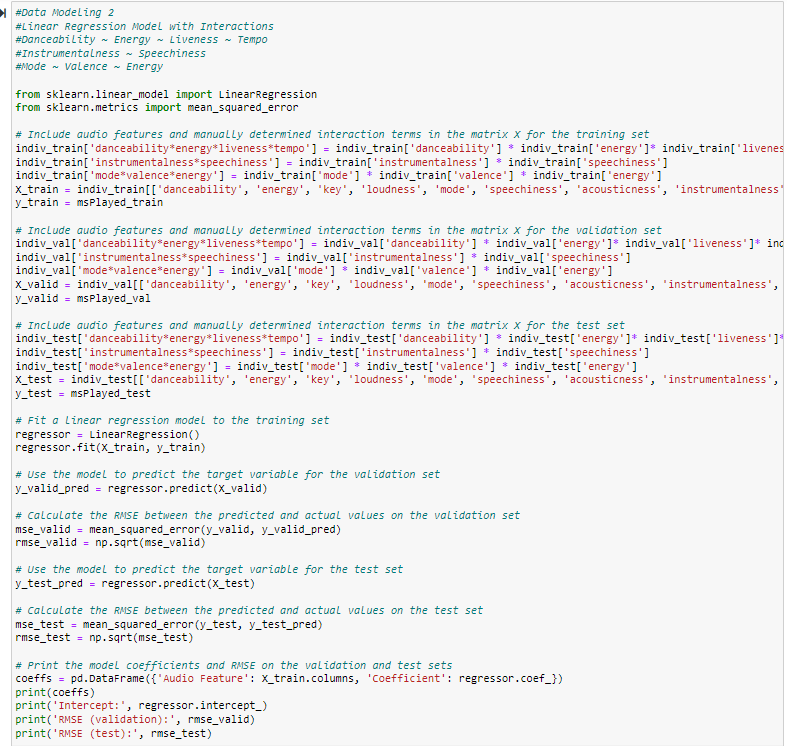


*Output*

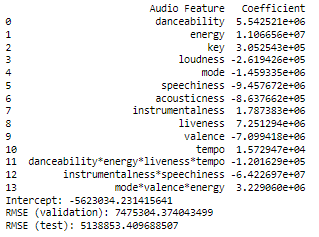


* + - * As shown in the above two screenshots, I firstly imported some Python libraries that are needed for my modeling. Sklearn (i.e. Scikit-learn) is a library used for machine learning. It provides a range of tools for building predictive models. It supports algorithms including linear regression, decision trees and random forests. Numpy and Pandas are also Python libraries that are used for numerical operations and data analysis.
      * After that, I included all audio features to fit the linear regression model into my train dataset. Then I performed a RMSE calculation of both validation dataset and test dataset. Finally, I print the coefficient of each audio feature and the RMSEs for validation and test datasets.
    - Linear Regression Model with Interactions

*Code*

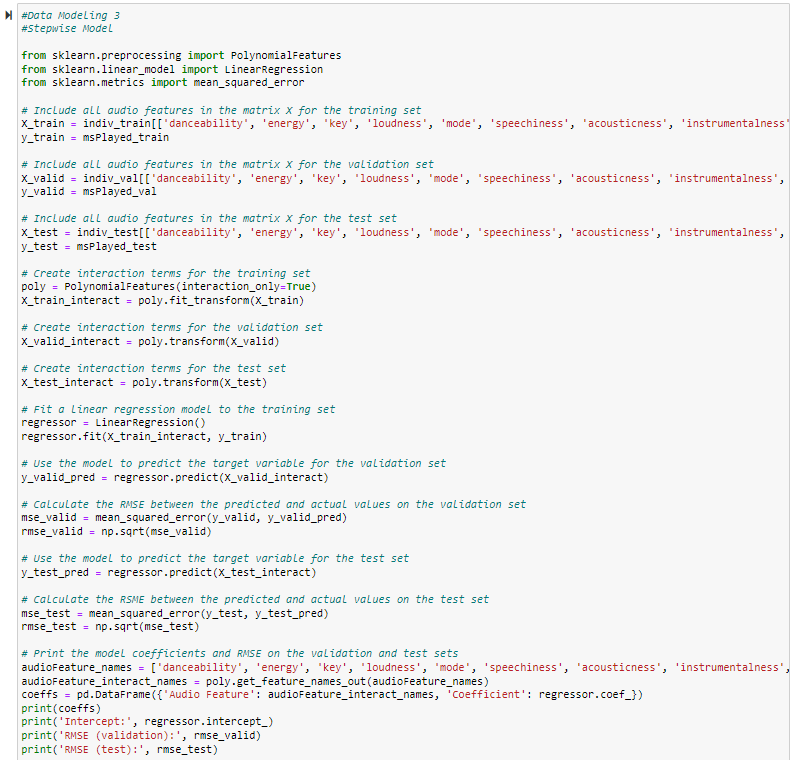


*Output*

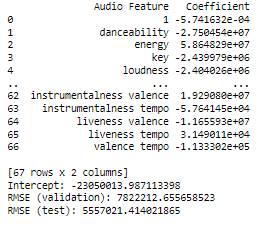


* + - * As shown in the above two screenshots, apart from including all audio features, I added some interactions among audio features based on my intuition. Other steps are the same as the simple linear regression model.
    - Stepwise Model

*Code*

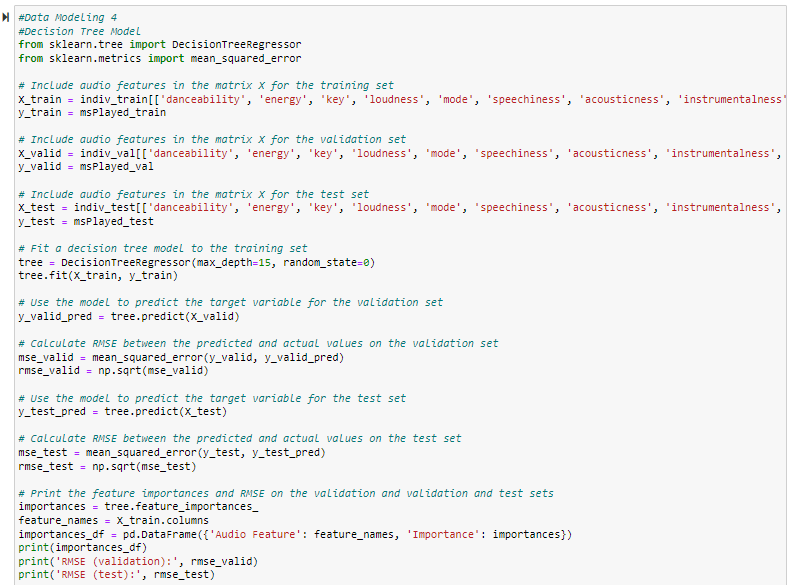


*Output*



* + - * As shown in the above two screenshots, I firstly imported a Polynomial Features library. As the default stepwise model only include audio features without interactions, I used this library so that stepwise model will have the capability to include higher-order interactions without myself having to manually add them. After that, I included all audio features in the matrix as a pool for stepwise model to select. Other steps are the same as the simple linear regression model.
    - Decision Tree Model

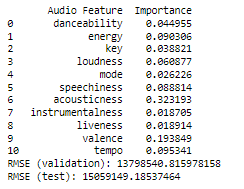
*Code*



\*Random state is used to ensure that the random number generator produces the same sequence of numbers each

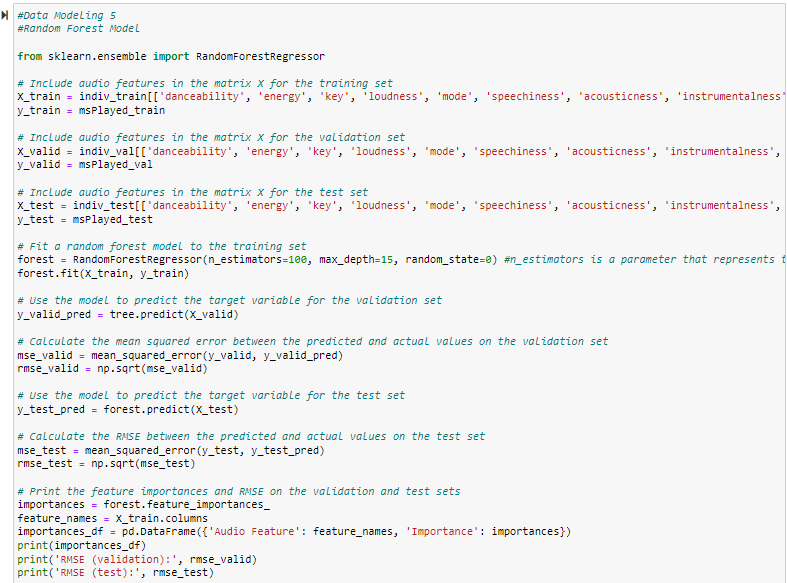
time the code is run. The value of random seed is random but it should stay consistent among different models.

*Output*



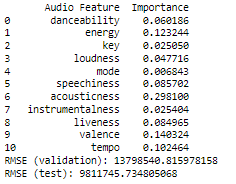
* + - * As shown in the above two screenshots, I firstly included all audio features to fit the tree model. I set the max depth to be 15 as there are 11 audio features of each track and I leave some space for the model to add some interaction terms. In tree model, we do not have the coefficient parameters like linear regression models. Instead, decision trees use a set of rules to recursively split the data into smaller and more homogeneous groups based on the features in the data. Therefore, I used the feature importance to indicate the significance of audio features for streaming determined by the tree model. Other steps are the same as the simple linear regression model.
    - Random Forest Model

*Code*



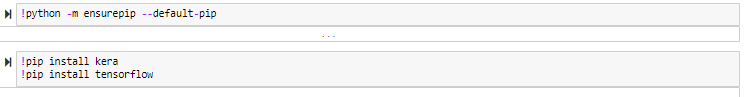
\*n estimators is a parameter that represents the number of decision trees

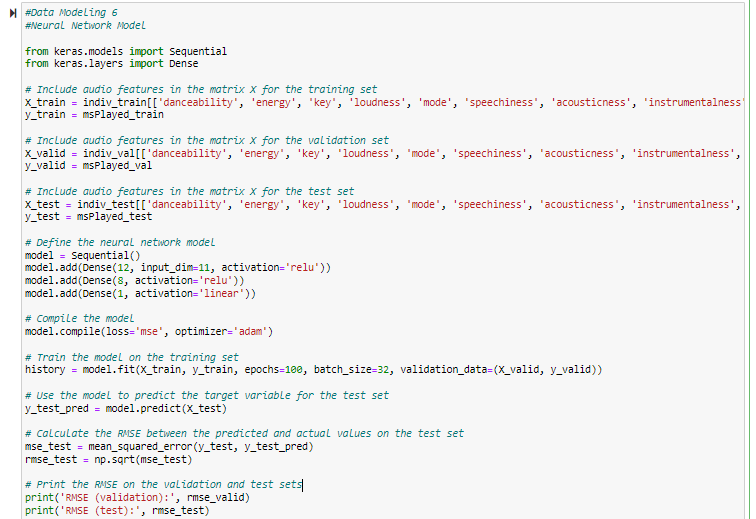
*Output*



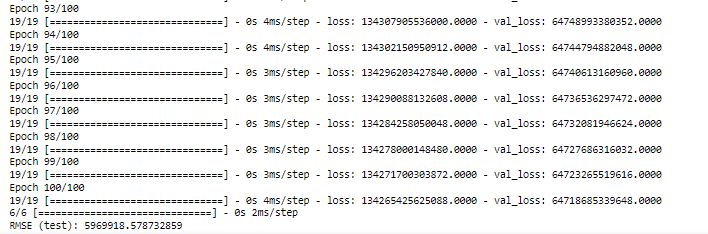
* + - * As shown in the above two screenshots, apart from the same steps as decision tree model, I built 100 decision trees with the same depth and combined them to improve the accuracy and reduce overfitting for this random forest model. The feature importance calculation is based on the average of all 100 decision trees in the forest.
    - Neural Network Model

*Code*





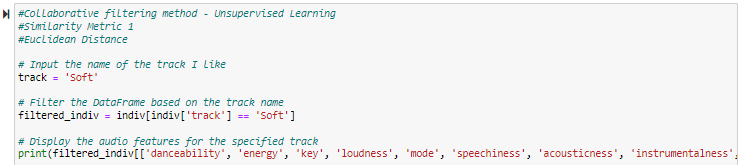
*Output*

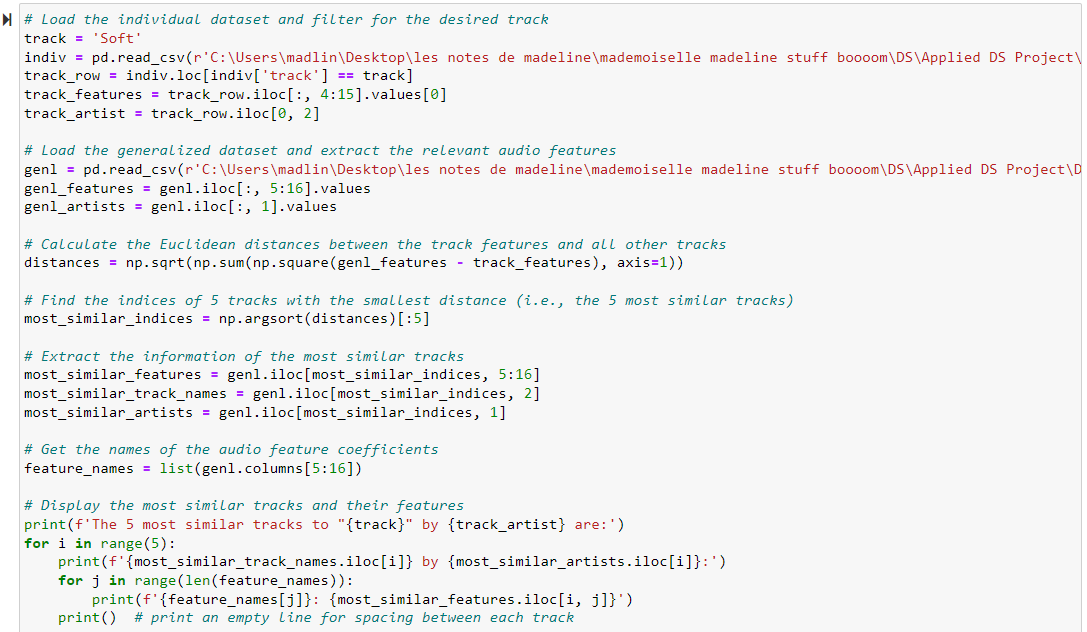


\*Due to the length of the complete output, only results from Epoch 93 to 100 are shown in this screenshot.

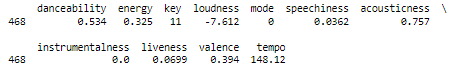
* + - * As shown in the above two screenshots, I firstly installed the Kera and Tensorflow packages using pip for neural network models. I included the single 11 audio features to fit the neural network model. After that, I defined the neural network model. The model consists of three fully connected layers (i.e. dense layers) stacked on top of each other, with the first layer having 12 neurons, the second layer having 8 neurons, and the output layer having single neuron. The input layer has 11 audio features that are passed as inputs to the first layer. The activation function used for the first and second layer is relu (i.e. rectified linear unit) which is commonly used in neural networks to introduce nonlinearity. The output layer uses a linear activation function, which means that the output of the model is a continuous value, in this case representing msPlayed.
      * The output shown is the training process of the neural network model. The loss is a measure of how well the model is performing at minimizing the difference between its predictions and the actual values. The output shows the loss for each epoch from 1 to 100. The loss should decrease over time as the model learns to make better predictions. Finally, it shows the RMSE of the test dataset.
* Collaborative filtering method
* Similarity Metrics
  + - Euclidean Distance

*Code*

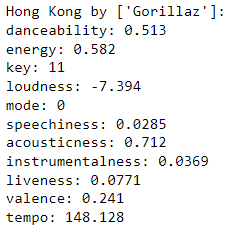
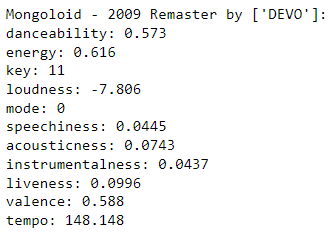
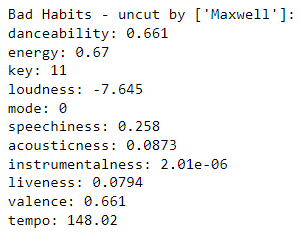
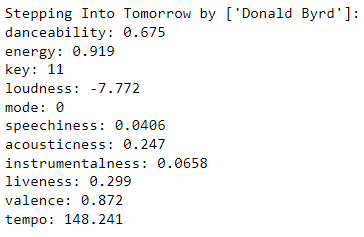
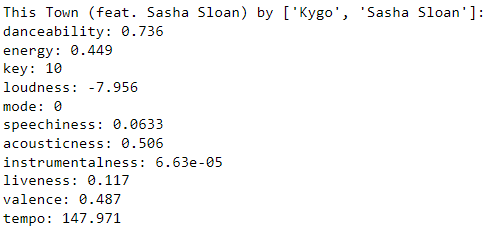




*Output*

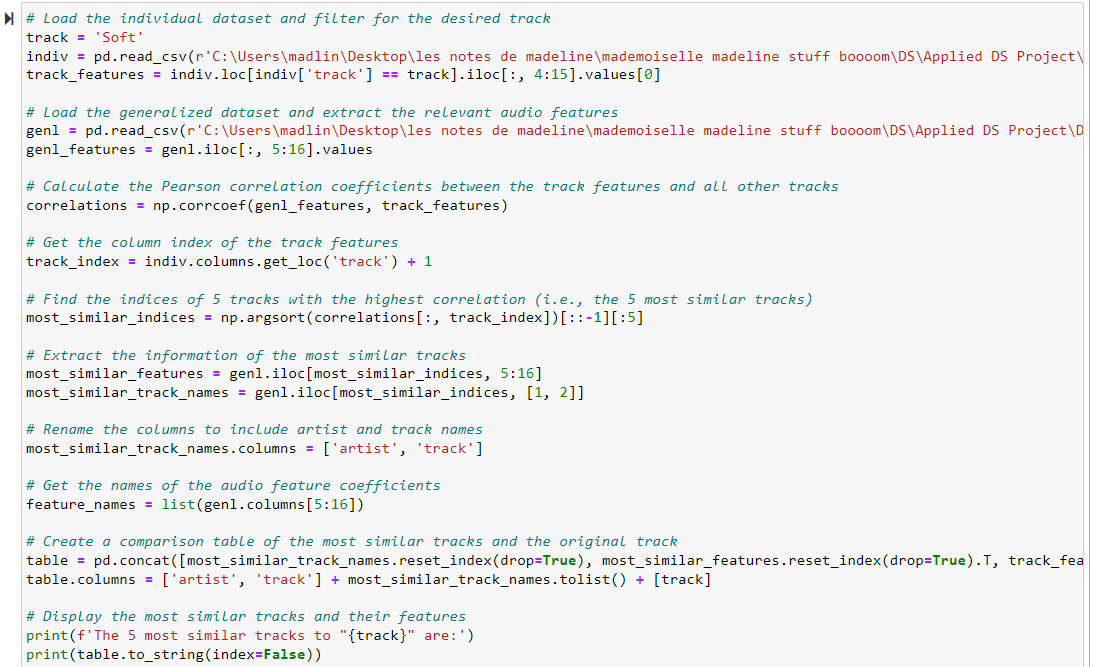




* + - * As shown in the above screenshots, I picked one of my favorite track *Soft* from my individual dataset and pulled its audio features. After that, I loaded the generalized dataset and extract the audio features columns. Then, I calculated the Euclidean distances between *Soft* are all other tracks. I found the 5 tracks with the smallest distances of *Soft* and pulled their audio features. They are *Hong Kong*, *Mongoloid – 2009 Remaster*, *Bad Habits – uncut*, *Stepping Into Tomorrow*, and *This Town (feat. Sasha Sloan)*.
    - Pearson Correlation Coefficient

*Code*

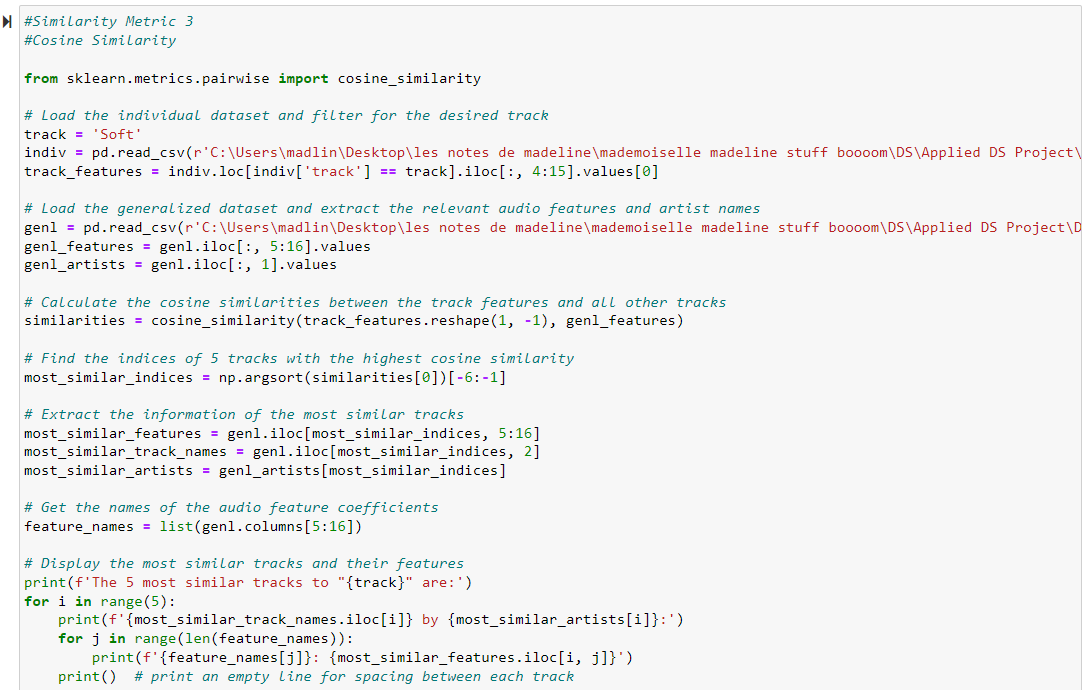


*Output*



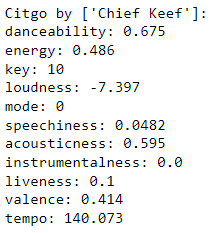
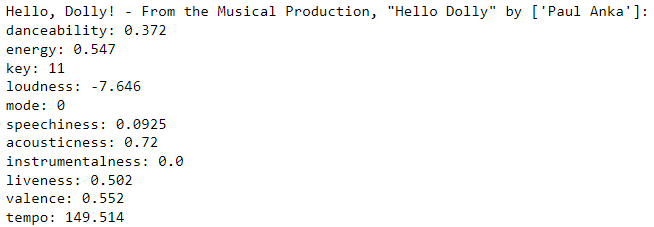
* + - * As shown in the above screenshots, I was unable to use Pearson correlation coefficient metric to calculate the distance between *Soft* and all other tracks. There is not enough memory available to perform such a request, which is to create a distance matrix of size (170654, 170654) as there are 170,654 rows in generalized dataset. The matrix requires a very large amount of memory to store, and may not be feasible to calculate on a single machine with limited memory. We may be able to use Hadoop Framework to solve this problem. I do not have this framework installed in my computer. Also, during my Big Data course last semester, we were unable to complete Hadoop assignment due to hardware issues. As a result, we ended up submitting documentation instead of competing practical exercises with Hadoop. Considering the practical limitations, I decided to pause at this point as continuing further would be too time-consuming.
    - Cosine Similarity

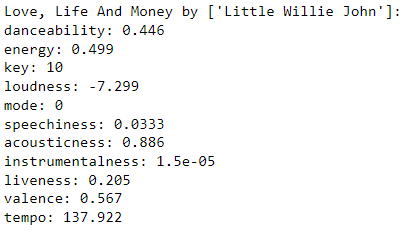
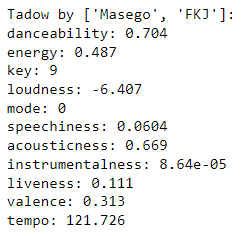
*Code*

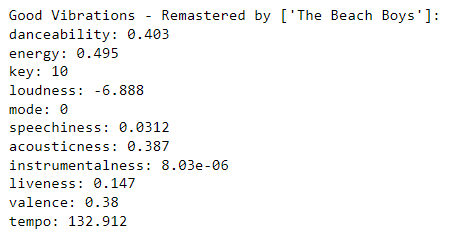


*Output*





* + - * As shown in the above screenshots, I imported the Cosine similarity package. Similarly as what I did for Euclidean Distance metric, I picked one of my favorite track *Soft* and loaded the generalized dataset. Then, I calculated the Cosine similarity between *Soft* are all other tracks. I found the 5 tracks with the smallest distances of *Soft* and pulled their audio features. Note that the np.argsort function returns an array of indices that would sort the Cosine similarity values in ascending order. [-6:-1] means the 5 highest Cosine similarity values from the sorted array were selected. So, the 5 most similar tracks are *Citgo*, *Hello, Dolly!*, *Love, Life and Money*, *Tadow*, *Good Vibrations – Remastered*. I was pretty excited to see that *Tadow* was selected because I love this song and it is already in my library actually.

**Task 4.4: Assessing Models**

**Deliverable 1: Model Assessment**

So far, I have finished building all the models. In this section, I would like to give a summary of the information developed in the models and provide technical assessments of the models.

* Content-based filtering method

*Summary of information developed in the models*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | SLR | LR with Interactions | Stepwise | Decision Tree | Random Forest | Neural Network |
| RMSE (test) | **5,053,164** | **5,138,853** | **5,557,021** | **15,059,149** | **9,811,746** | **5,890,962** |
| RMSE (validation) | 7,392,925 | 7,475,304 | 7,822,213 | 13,798,541 | 13,798,541 | 7,392,925 |
| Intercept | -3,775,279 | -5,623,034 | -23,050,014 | N/A | N/A | N/A |
| *Coefficients/Importance* | | | | | | |
| Danceability | 4,131,844 | 5,542,521 | -27,504,540 | 0.045 | 0.060 | N/A |
| Energy | 105,326,200 | 11,066,560 | 58,648,290 | 0.090 | 0.123 | N/A |
| Key | 298,014 | 305,254 | -2,439,979 | 0.039 | 0.025 | N/A |
| Loudness | -2,343,501 | -261,943 | -2,404,026 | 0.061 | 0.048 | N/A |
| Mode | -548,063 | -1,459,335 | N/A | 0.026 | 0.007 | N/A |
| Speechiness | -10,475,390 | -9,457,672 | N/A | 0.089 | 0.086 | N/A |
| Acousticness | -904,630 | -863,766 | N/A | 0.323 | 0.298 | N/A |
| Instrumentalness | -2,069,824 | 1,787,383 | N/A | 0.019 | 0.025 | N/A |
| Liveness | 819,883 | 7,251,294 | N/A | 0.019 | 0.085 | N/A |
| Valence | -5,897,379 | -7,099,418 | N/A | 0.194 | 0.140 | N/A |
| Tempo | 9,139 | 15,729 | N/A | 0.095 | 0.102 | N/A |
| Danceability\*  Energy\*  Liveness\*  Tempo | N/A | -120,163 | N/A | N/A | N/A | N/A |
| Instrumentalness\*  Speechiness | N/A | -64,226,970 | N/A | N/A | N/A | N/A |
| Mode\*  Valence\*  Energy | N/A | 3,229,060 | N/A | N/A | N/A | N/A |
| Instrumental\*  Valence | N/A | N/A | 19,290,800 | N/A | N/A | N/A |
| Instrumentalness\*  Tempo | N/A | N/A | -57,641 | N/A | N/A | N/A |
| Liveness\*  Valence | N/A | N/A | -11,655,930 | N/A | N/A | N/A |
| Liveness\*  Tempo | N/A | N/A | 31,490 | N/A | N/A | N/A |
| Valence\*  Tempo | N/A | N/A | 113330 | N/A | N/A | N/A |

*Technical assessments of the models*

* Based on the RMSE on the test dataset, the simple linear regression model has the relatively lowest RMSE among all six prediction regression models, followed by linear regression model with interactions, stepwise model, neural network model. The decision tree model and random forest model do not perform well compared to others in my case.
* In my opinion, due to the small sample size of my individual dataset which contains only 946 rows with 11 audio feature attributes, a simple regression model is adequate for developing a prediction model. This is also why a simple regression model is considered as the most commonly used and popular method. The decision tree and random forest models may be overfitting to the training data, which means that they capture the noise in the data and do not generalize well to the test data. Moreover, it appears that adding interactions between the audio features did not improve the performance of the linear regression and stepwise models. Therefore, it suggests that the relationships between the audio features are not complex and can be adequately captured by a simple linear regression model.
* Additionally, note that the table above illustrates that the RMSE values on the validation dataset are generally in agreement with those on the test dataset, except for some differences observed in the neural network model. When the RMSE is lower on the test dataset, it tends to be lower on the validation dataset as well.
* Based on the simple linear regression, we can see that energy plays the most significant positive role in my streaming of a track, followed by liveness, and danceability. However, speechiness plays a significantly negative role in my streaming of a track, followed by valence, loudness, and instrumentalness. Key and tempo play positive roles in my streaming time, but the impacts are pretty slight compared to other audio features. Mode plays negative role in my streaming time, but the impact is pretty slight compared to other audio features.
* Electronic dance music (“EDM”) consists of the majority of songs in my library. They tend to have high energy, liveness and danceability. Most EDMs do not have much vocals and I believe I do not listen to Raps much so it makes sense that speechiness plays a negative role. I do not listen to instrumental music much so it makes sense that instrumentalness plays a negative role.
* I am surprised to find that valence and loudness play a negative role in my streaming. I like cheerful songs which should have high valence and sound more positive. If I like EDM, it should be the case that loudness play a positive role in my streaming. It might be some noise or randomness in the data that is causing the negative relationship between valence or loudness and my streaming preferences.
* As the linear regression model has a slightly higher RMSE compared to the simple linear regression model. It is still worthwhile for me to interpret some interaction terms.
* As shown in the table, the interaction between instrumentalness and speechiness is negative, which aligns my initial intuition. Usually, a track with a higher instrumentalness has less vocals/spoken words/raps. The interactions among mode, valence and energy are positive, which also align my initial intuition. Most of the case, a track with high energy should have more positiveness, meaning a higher valence; and have a brighter and more upbeat sounds, meaning having a major key. However, the interactions among danceability, energy, liveness and tempo are negative. I could not be able to explain this, unfortunately.
* Collaborative filtering method

*Summary of information developed in the models*

|  |  |  |
| --- | --- | --- |
|  | Euclidean Distance | Cosine Similarity |
| *The 5 most similar tracks to Soft* | | |
| 1 | Hong Kong | Citgo |
| 2 | Mongoloid – 2009 Remaster | Hello, Dolly! |
| 3 | Bad Habits – uncut | Love, Life and Money |
| 4 | Stepping Into Tomorrow | Tadow |
| 5 | This Town (feat. Sasha Sloan) | Good Vibrations – Remastered |

*Technical assessments of the models*

Unsupervised learning models cannot be evaluated using a single metric, unlike supervised learning models that use RMSE as a measure of performance. This is because the goal of unsupervised learning is to uncover patterns, structures, or relationships within the data.

In my case, I am using the similarity metrics to uncover the relationships within the data. From my perspective, the approach to measure the performance should be just manually inspect the results. In other words, I will listen to those 10 tracks and determine whether I like them or not.

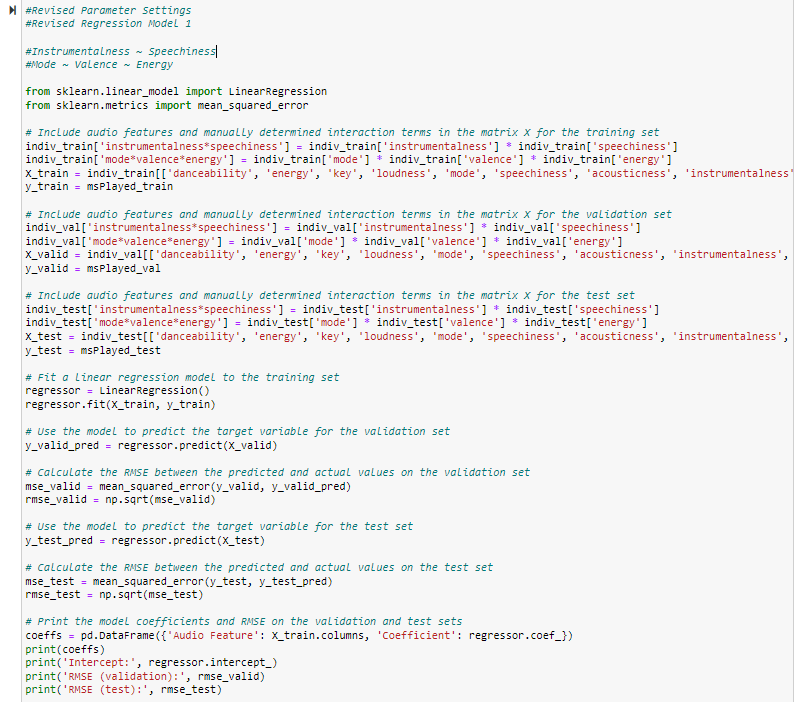
**Deliverable 2: Revised Parameter Settings**

So far, I haven’t performed the step of manually inspecting the results. This manual inspection will apply to both content-based filtering method and collaborative filtering method. For content-based filtering method, I will fit the optimal prediction model to the global weekly top 100 songs dataset and add one new column pred\_msPlayed. I will choose the top 20 songs with the largest predicted streaming time and listen to them and determine how many songs I want to add to my library. For collaborative filtering method, as discussed above, I will listen to those 10 songs which are attained based on two similarity metrics and determine how many songs I want to add to my library.

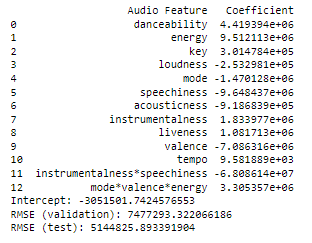
Based on my review of the CRISP-PM instructions, we will have the Phase 5.0 – Evaluation where it looks more broadly at which model best meets the business needs and this phase will include evaluating results, reviewing the process and determining the next steps. In contrast to the technical focus of the data modeling phase, I would like to reserve the manual inspection for the subsequent phase.

* Content-based filtering method
* Prediction Regression Models
  + - Simple Linear Regression Model
      * As for revising parameter settings, my understanding is that there should not be many adjustments in terms of the parameter settings for the simple linear regression model. Most of the time, we just leave the default parameter settings as it is for a simple linear regression model. Nevertheless, I tried two models with revise on parameter settings by added a few interaction terms on the top of the simple linear regression model to see if there is any space for lowering the RMSE.
      * For the first model, I removed the interaction terms of danceability\*energy\*liveness\*tempo for our original linear regression model with interactions as I was unable to interpret the negative coefficient. I just kept the interactions between instrumentalness and speechiness and the interactions among mode, valence and energy.

*Code*

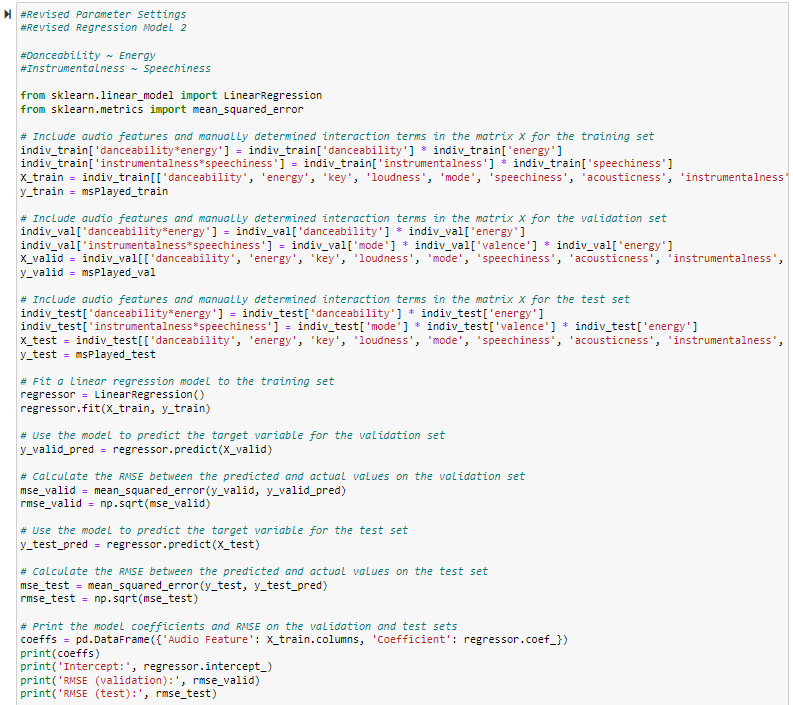


*Output*

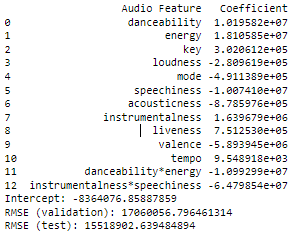


* + - * For the second model, I included the interactions between danceability and energy, and interactions between instrumentalness and speechiness.

*Code*

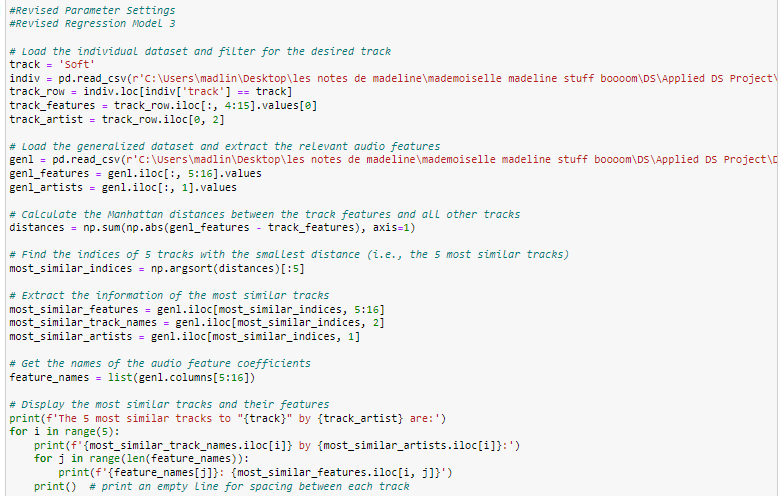


*Output*



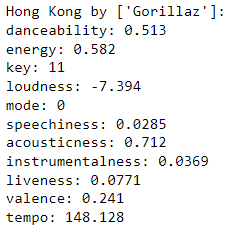
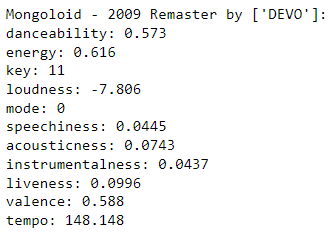
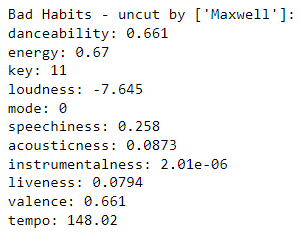
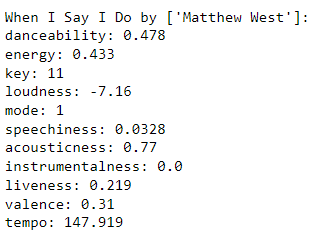
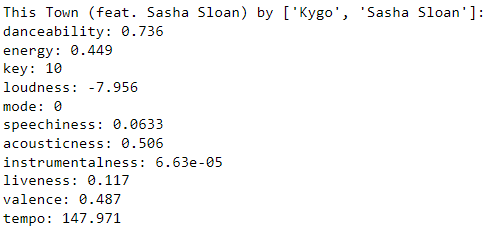
* + - * Unfortunately, as presented in the above two output screenshots, neither of the revised model have a lower RMSE compared to the simple linear regression model. I will stick to the simple linear regression model.
* Collaborative filtering method
  + Similarity Metrics
    - Euclidean Distance
      * In terms of the similarity metrics, for Euclidean Distance, the parameter that can be adjusted is the "p" value. By default, p=2, which means that the standard Euclidean Distance is used. However, we can set p=1 to use Manhattan Distance. I revised my code and produced the following output.

*Code*



*Output*



* + - * *Stepping Into Tomorrow* has been replaced by *When I say I do* when I used Manhattan distance. Apart from that, all other four songs remain the same. However, I am hesitant to revise the parameter settings in this way, as it seems to involve changing the metric itself rather than adjusting the parameters of the current metric.
    - Cosine Similarity
      * For Cosine Similarity, there are no specific parameters to adjust. However, the similarity measure can be affected by preprocessing steps such as scaling or normalization of the features. I believe I have already scaled the audio features so there should be no more adjustment in my case.

As this assignment involves a significant amount of data modeling, I included the code script for your reference in submission.

1. https://engineering.atspotify.com/2021/12/a-look-behind-blend-the-personalized-playlist-for-youand-you/ [↑](#footnote-ref-1)