Assignment 3 Jesper Ahlman

Core

To start off analysing the data I looked at the 4 largest correlations between features (appendix 1.1). Some of these make sense as the more energy in a song the louder you would expect the song to be e.g. in Rock music. Also acousticness has a large negative correlation with loudness and energy which also makes sense as we know loudness and energy are correlated and then them both having a negative correlation with acousticness makes sense because the more acoustic a song is the more quiet and calming the song is going to be. For example you don't see many Rock songs with large amounts of acoustics. These correlations may play a part in overfitting the models seeing as these features may be too similar so it may be useful to drop the one of the highly correlated features e.g. loudness or energy. On the other hand duration_ms had some of the lowest correlations on average (shown on appendix 1.2) which shows that it is a very independent feature which has little relation to other features. However these correlations only show the correlations between numerical features but it may be worth removing this feature as it may not be useful. With a feature like instance id the correlation values are not high enough to show there is any relation between instance id and genre so I will definitely be dropping this feature also. These correlations can also be visualised using the correlogram (appendix 1.3) which helps to show the scatter plots across features. From the graph the energy loudness correlation is evident as shown from the positive linear correlation (which is highlighted). Other key linear correlations are also present.

Whilst there are no NaN missing values in the datasets there are still plenty of missing values. Notably some entries in the Artist Name feature have “Empty Field” as their input which I would consider a missing value. Other features with missing values include tempo with a lot of inputs with “?” which is a missing value undetected and also duration_ms with a lot of inputs as -1 which is a missing value. The counts of each of these missing values for these features are found in appendix 2.1. It shows that 20.1% of artist names, 20.0% of duration_ms and 14.9% of tempo values are missing values. These are quite high percentages and so this will have an effect on results as they would act as major outliers and with the tempo feature it will treat the values as strings instead of floats making it hard to perform EDA and preprocessing on it. It will definitely be worth imputing these missing values in the preprocessing with the mean/mode values to improve accuracy.

The next findings I discovered in the dataset is in terms of the number of unique values. The list of features and the numbers to the right (appendix 3.1) represents the number of unique instances for the training set and the other image (appendix 3.2) is the test set. By looking into the number of unique values I see that instance id has 50000 unique values in the training set which makes sense as instance id is a unique value assigned to each instance however this can mean that it can be unhelpful for ML and we’ve seen before that there is a low correlation so it would be worth dropping. Track name and track id also has a large number of unique values with over 84% of values being unique for training and over 89% for the testing set so it wouldn't be very helpful to use these features in the classification process. Surprisingly the artist name has very low numbers of unique values especially for the training set which means there must

be a lot of repeat artists in the dataset so it may be worth encoding possibly or dropping the feature all together.

I then visualised the 3 categorical features against genres using heat maps to show frequencies for the training set (appendix 4). Some notable finding from this include that the key D# is not frequently used across all genres besides movie with 274 occurrences meaning it could help with the models to predict the movie music genre using D#. Similarly C# is very commonly used for Hip-Hop genre with the most occurrences with 958 so will be useful in classification. For the mode feature it appears that Major is more frequently used than Minor for all genres. Electronic music tends to use minor of the most over all genres however major is still predominantly used this may be a useful finding which could help for the classification. With the time signatures feature it seems to be that the majority of the data is under “4-Apr” so this will make it hard for the ML tool to be able to use this feature to provide benefits and the model may cause curse of dimensionality. However time signature would be able to provide some benefit since Comedy and Movie genres have lower recordings of “4-Apr” than the other genres with only 2419 and 3773 compared to the 4500 odd recording for other genres this means that if the ML model sees “3-Apr” as a value then it would probably be able to predict that its Comedy or Movie genre which would be beneficial.

I further visualised the remaining numerical features using a violin plot showing the values across genre’s (appendix 5). Most features had very similar distribution across genres which may not be useful for the classifier to use to help distinguish between genres but some features had interesting results. For example, speechness, liveliness and acousticness had a full spread of values but generally all the data is confined to the lower values across genres except for comedy genre which the data was skewed more towards the higher end as comedy music is not really music and is quite different which resulted in a higher mean. This is useful because it may make the classifier more effective at predicting comedy genre since if the results for these 3 features was high then the model would be able to learn that and predict comedy genre. Movie genre also had quite high acousticness results similar to comedy. Tempo’s results were very normally distributed which may result in it not being a very useful feature to help with the classification and may be worth dropping. In terms of loudness music, movie genre had much quieter results on average as movie music can be quite quiet so this is an interesting finding which may be useful for classification and will be shown when scaling is done. Popularity had very varying results across genres where each genre had very different averages. This will be very helpful in the prediction of music genres.

Completion

With my first submission I achieved an accuracy of 0.37 on the public leaderboard. To get that initially my preprocessing consisted of dropping instance id, artist name, track name and track id as there were too many unique values that I felt were worth dropping to not overfit the model with useful features. Then I filled na values with none or 0 to test if leaving the missing values such as “?” would make a difference and used label encoder to encode the non numerical values as I knew not all values were numerical. Then I used MinMaxScaler to scale the data between 0-1 and this was my preprocessed training dataframe. I repeated the exact same

process for the test set for consistency. I then used this preprocessed data on a simple classifier being KNN as I knew this model worked well on the Banknotes dataset in A1 and I wanted to start off with a basic intermediate model which I knew would work relatively well to start. I didn't do any dimensionality reduction between this step as I felt removing the names and ids at the start would be enough and the dataset was relatively small with only 14 features after this and I felt for these features to be included they must be useful. Despite using some naive approaches on the first model the accuracy of 37% showed that there was some improvement needed. This matched my cross validation score which reassures that I am not over training to the public leaderboard as it performs just as well on the training set.

After my first submission I experimented with some other models including Gradient Boosting, Decision Tree, Random Forest and MLP and also changing hyperparameters of KNN and none of this saw any improvement. I also tried using RFE feature selection to test my hypothesis before and I proved it correct as there was no improvement. The next thing I wanted to test was my method of preprocessing. This is because previously I was using label encoder which is a very naive approach for encoding categorical data as it assumes that one value is greater than the other. For example in mode the encoder would assume that Minor was greater than Major when this is not correct as categories should not be assigned values. Consequently this made my models very inaccurate as it would learn that Minor is greater than Major and use this factor in its prediction which can result in major inaccuracies. To fix this problem I changed my encoder to use OHE which is a much better way to encode categorical data by expanding the dimensionality to allow more features one for each possibly category the entry could include and then using binary encoding it would input 1 if present and 0 if not. This allows the model to fit better and treat each category individually which allows for better prediction. On top of this I used a pipeline to do this and used Standard Scaler instead of MinMaxScaler to try a different approach. On top of this I used the insight gained in EDA to impute the valid non-null missing values such as "?" for tempo and "-1" for duration ms. I imputed this using simple imputer and imputed it with the mean to fill them in as there were so many that I didn't want to drop them. I also used my most accurate model being KNN for this and saw a huge improvement to 51%.

I then changed the model back to MLP as I knew that was a neural network and would work better with OHE as it looks at all possible interactions between data in the training so it would be able to link the occurrences to form a better model. This increased the accuracy also by 10% to 62% on the public leaderboard showing that using a Neural Network Classifier helped. I also changed back to MinMaxScaler and it increased the accuracy further which proves that with this dataset the MinMaxScaler is better. To reduce overfitting the model I followed my EDA decision to remove the loudness feature as it had large correlation to energy so thought they were not worth both having to reduce the dimensionality. This proved unsuccessful and the accuracy decreased showing that they were different enough that it was worth keeping. I later realised that I was never using the fitted preprocessing pipeline created for the training set to transform the test set, which is bad preprocessing as it can create inconsistencies that the ML algorithm will tailor to the training set whereas the test set preprocessing would differ. By changing this it showed some improvement and is also better practice. I also tested out shuffling the preprocessed data frames as I thought the model could be learning to predict based on the

order in the data. However, this didn’t work at all since the 2 datasets would be shuffled differently and so when predicting labels they would be out of order and the order doesn’t play much effect on some algorithms which I later found to be better. I also tried HistGradientBoosting since the sample size is greater than 10 thousand but saw no improvement.

Following this I then spent some time trying to encode the artist names. This is because from the EDA I realised that artist names didn't have that many unique instances as there were a lot of repeat artists in the dataset. So from this I thought if i could encode this feature it may improve accuracy seeing as these artist names may reappear in the test set so it would help the model predict the correct genre. Since in the training set there were 5679 unique instances for artist names I knew One Hot Encoder would increase the dimensionality by over 5000 times so this may cause a curse of dimensionality so instead I encoded the most common names. This saw a slight increase in score and then I thought that it may be beneficial to find the most frequently used words for each genre and encode those using word statistics. This saw a bigger improvement in accuracy and paired with Gradient Boosting Classifier this increased the accuracy further to 63% and so was successful. This then took me to research other better ways to encode large categorical features. I found a classifier called CatBoost which promised to be able to handle categorical variables without the need to encode them but when I implemented this model it predicted all of the same genre for the test set so was unsuccessful. I then found other encoders and tried out the frequency encoder which basically outputs only one column per feature so doesn’t increase dimensionality and it works by encoding each instance with the count that that value appears across the whole dataset. Then paired with the MinMaxScaler which proved to be the most successful scaler it increased the accuracy score further as it included more data for the model to train on which proved beneficial. I was able to encode all the categorical values using this including artist name and track name, however I still excluded track id and instance id as there were too many unique instances that it would overcomplicate the model. Furthermore after proving successful I then decided to keep encoding key,mode and time signature using One Hot Encoding as there were very few unique instances and through research and testing I still knew One Hot Encoding was a more effective encoder. By doing this and combining it with a more powerful ensemble classifier being XGBoosting which works better on One Hot Encoder this improved my accuracy to my highest being 66.6%. I also tried an Ordinal Encoder on key and time signature features as I felt that they could be considered as ordinal but it didn't improve the accuracy surprisingly. Throughout this whole process the cross validation scores continued to maintain consistent with the public leaderboard which shows that I am not over training to the public leaderboard.

After this I went back to my CatBoost solution I attempted earlier to see if I could get it working. I found that the reason for my problem was that I had left instance ID in the training set so the model learnt to predict based on the instance ID and this led the classifier to predict folk genre for all entries. I removed the instance ID along with track ID and kept in track and artist names and just did the imputing I had been doing earlier on the missing values. I researched and found that CatBoost Classifier did the encoding of categorical variables, scaling and normalising for you so I put the dataframe in and predicted the labels and it increased my accuracy by nearly

10%. To ensure this was not just a success on the leaderboard and the score is an accurate representation I started to perform cross validation which I later found was not possible using the sklearn function because with CatBoost you need to specify the indexes of the features that you want encoded. To compromise I used normal validation comparing predicted labels to actual labels on the training set. This resulted in 0.75 accuracy which is very similar to the Kaggle result so I was satisfied with my submission on Kaggle. I also wanted to do a more deep analysis of the hyperparameters to find the best one for CatBoost with the training set. To do this I used GridSearchCV which is a grid search method which finds the best hyperparameter (iterations,learning rate and depth) for the training set using cross validation. The results showed the best depth was 9, learning rate was 0.04 and iterations was 100. Then I did validation on this again and saw an improvement so submitted this as my final model and saw a slight increase to 78.1% on the public leaderboard.

The system I chose to submit was the last submission on Kaggle. This is because it proved accurate on the validation with very similar accuracies so I know it hasn't been overfitted just to the training dataset so I'm confident on the results when the private leaderboard is released based on my public results. On top of this it is my highest performing model so I feel most confident in this system that it is the best. I am also confident in this Machine Learning tool because it does the encoding, normalising and scaling for you so I know the methods its using is the most optimal for the classifier. I am also confident that the hyperparameters chosen are the best for the model because of the cross validation grid search. I have also reflected upon past renditions to fill missing values appropriately and using the fitted models on the test set. The use of the quicker and more powerful classifier being CatBoost also proved effective in predicting accuracies. The decision to also not do any feature selection was a thought out exclusion also and proved to be a necessary one.

Challenge

The Machine Learning model I chose to use in my final submission was CatBoost Classifier. This model is an enhanced version of Gradient Boosting Classifier that sklearn has. This model is very hard to interpret because it uses an ensemble methodology to predict labels and so it is quite hard to understand why certain predictions have been made. The model takes an easy to interpret model being a decision tree where the prediction follows a set path but has added optimization where it creates an ensemble of various estimators. These estimators are lots of weaker models which Gradient Boosting combines them to make a stronger more effective result. This makes the model very intricate and hard to follow as there are lots of smaller models being created to combine to make the stronger one. The problem with this is if Spotify were to use my system for example the users and developers may not know whether they can trust this model because they may not understand exactly how the predictions are made so they wouldn’t be able to understand the relationships between features. Unlike easier to interpret models such as K-Nearest Neighbour or Decision Tree where it is much easier to interpret as for the decision tree you know it follows a binary path and KNN where you can visualise the predictions using a scatter plot and understand what the complexity control is doing. Due to CatBoost being a hard to interpret model, users will find it hard to understand and so they may deploy the solution without understanding it fully which may cause problems if the model starts to underperform

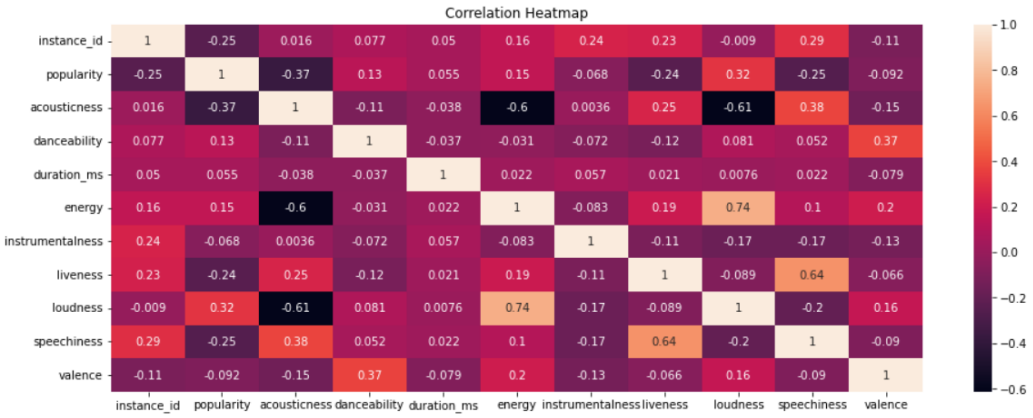
seeing as the users may find it hard to trace error to find the problem or feature causing the problem. Running the system doesn't take much time with results within 2 minutes which is a positive to this model compared to the normal Gradient Boosting Machine Learning model. CatBoost is unique in that it does the encoding, scaling and normalising itself which can make the preprocessing easier to follow. However, it can be very confusing to understand the inner workings of the model to understand how it encodes and scales because you cannot see the dataframe results after its preprocessing. Overall whilst CatBoost Classifier proved to be the most accurate model this doesn't mean it's the best model because interpretability also plays a big part in the real world use of the Machine Learning model.

Appendix

1.1

1	+0.743	energy	loudness
2	+0.638	liveness	speechiness
3	-0.612	acousticness	loudness
4	-0.601	acousticness	energy

1.2



1.3

2.1

```
instance_id            0
artist_name        10065
track_name             0
track_id               0
popularity             0
acousticness           0
danceability           0
duration_ms        10022
energy                 0
instrumentalness       0
key                    0
liveness               0
loudness               0
mode                   0
speechiness            0
tempo               7461
time_signature         0
valence                0
genre                  0
dtype: int64
```

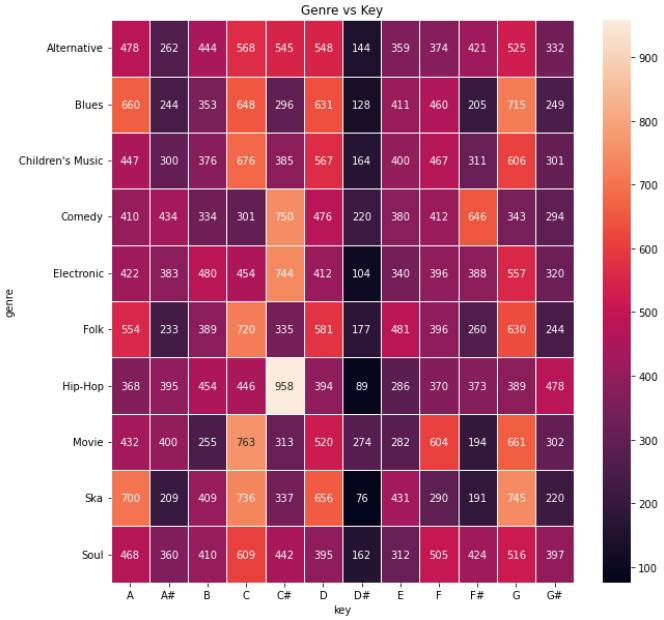
3.1

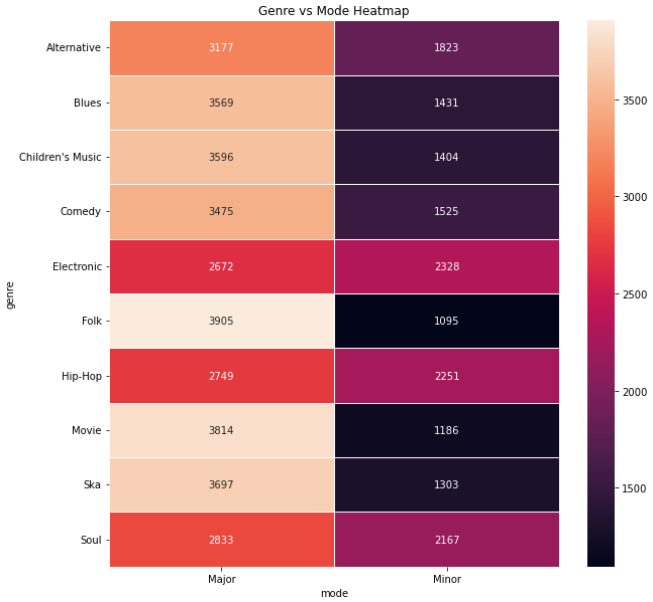
```
instance_id        50000
artist_name         5679
track_name         42472
track_id           46378
popularity            94
acousticness        4194
danceability         915
duration_ms        25159
energy              1511
instrumentalness    5163
key                   12
liveness            1673
loudness           15600
mode                   2
speechiness         1611
tempo              30461
time_signature         4
valence             1567
genre                 10
```

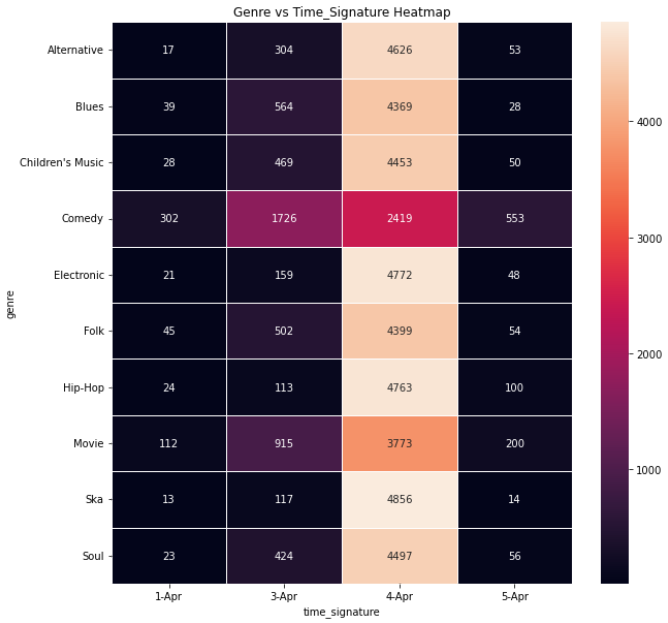
3.2

```
instance_id        30931
artist_name         4816
track_name         27550
track_id           29448
popularity            94
acousticness        3908
danceability         881
duration_ms        17827
energy              1331
instrumentalness    4834
key                   12
liveness            1630
loudness           13095
mode                   2
speechiness         1558
tempo              20923
time_signature         5
valence             1478
```

4

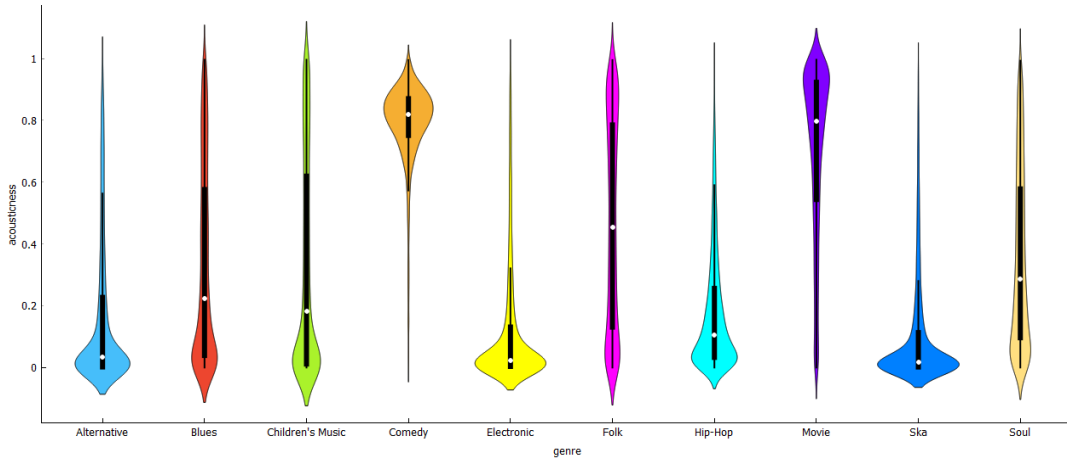




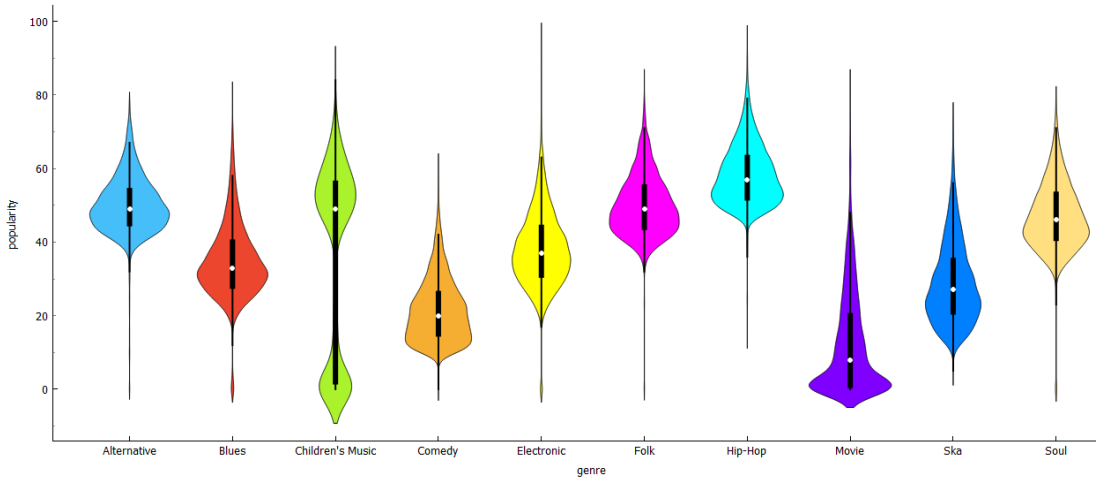


5

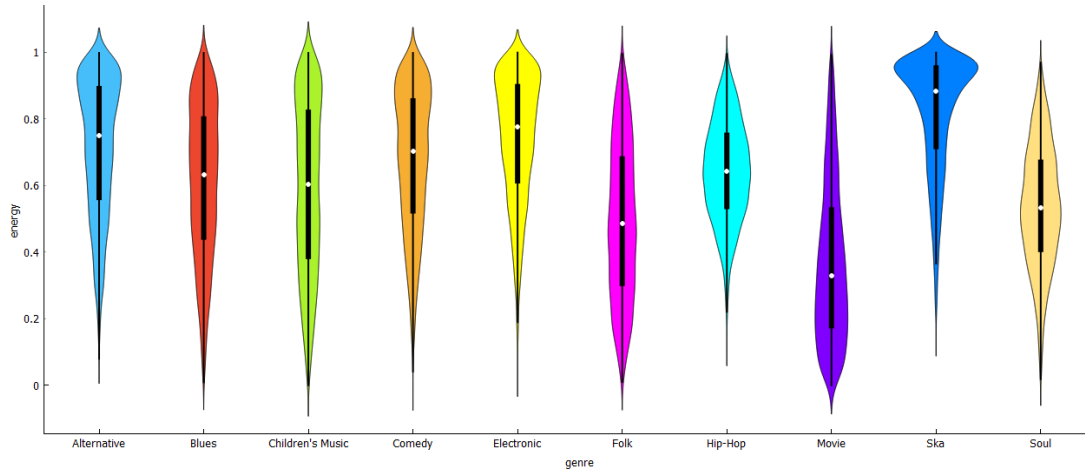
Acousticness vs Genre



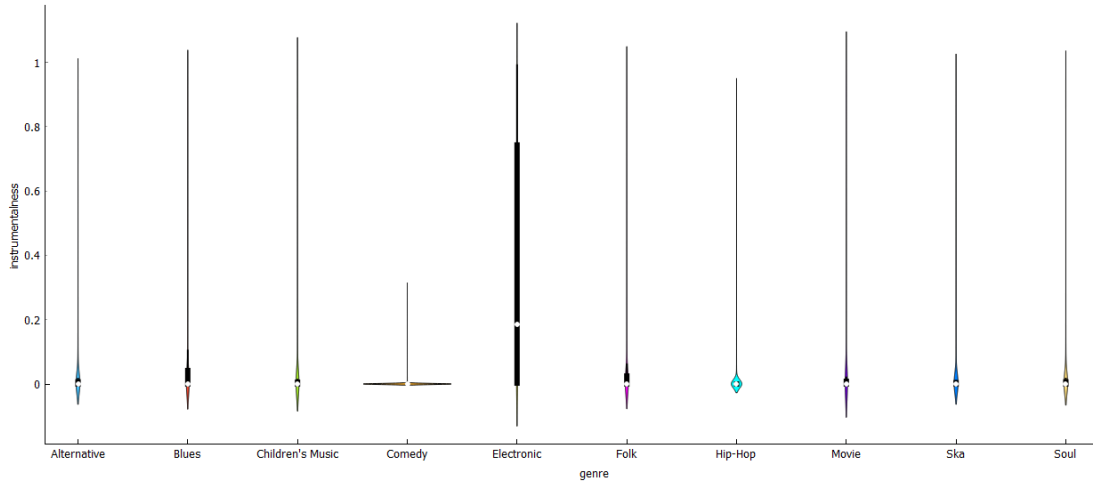
Popularity vs Genre



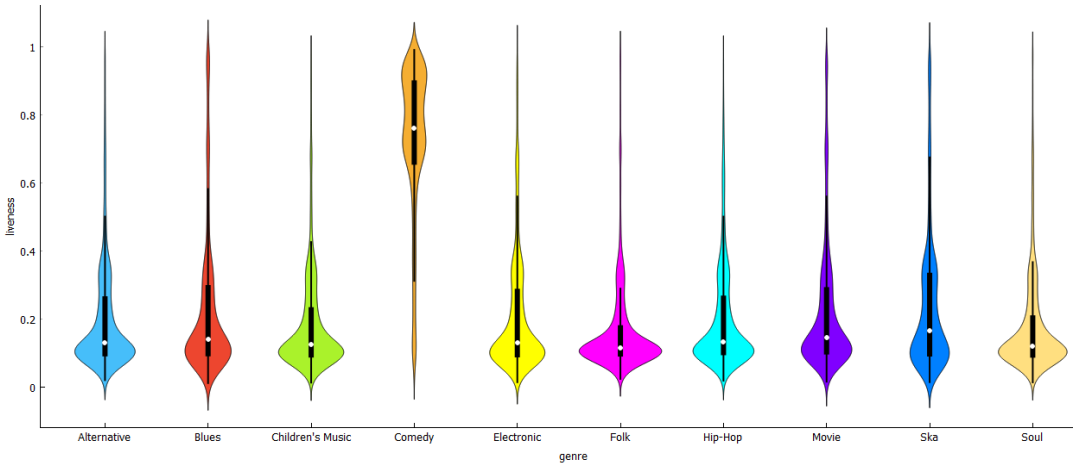
Energy vs Genre



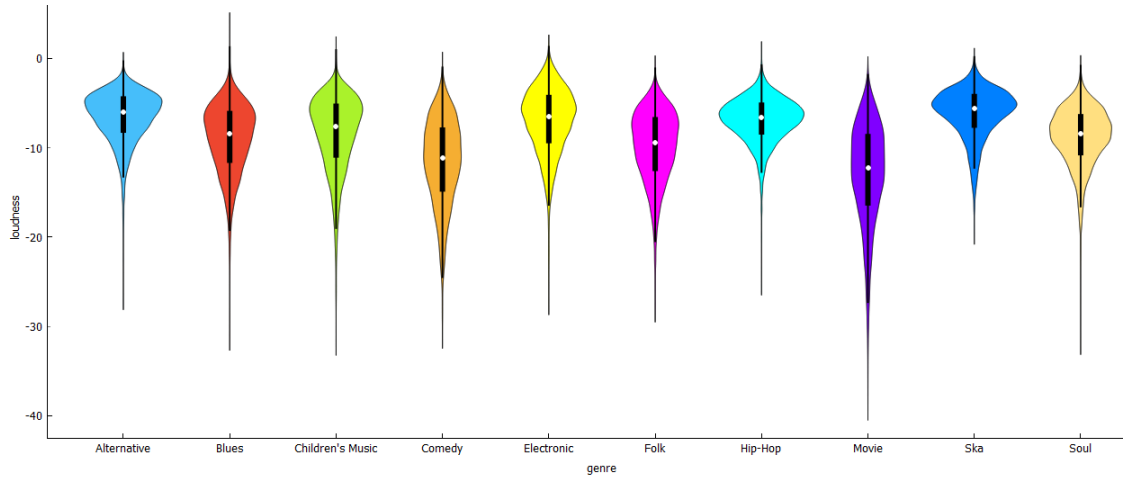
Instrumentalness vs Genre



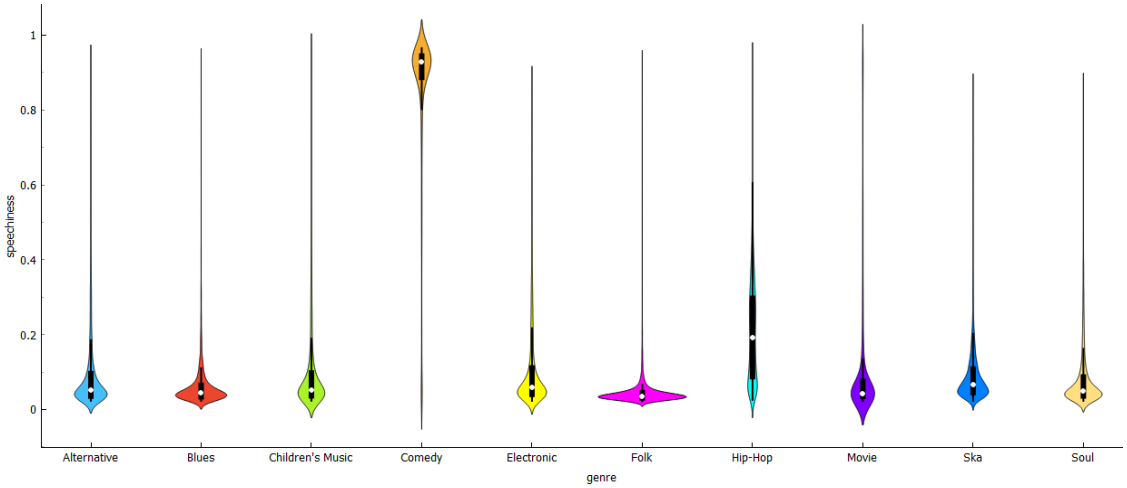
Liveliness vs Genre



Loudness vs Genre



Speechness vs Genre



Tempo vs Genre