Assignment 3 – Report

Part 1 : Core

1. Correlation heatmap (in jupyter notebook for clearer image)

- There is strong correlation between the two features *loudness* and *energy*, with average coefficient value of **0.728**. This indicates strong positive linear relationship. All the 10 dataset shows strong coefficient values, with the smallest value being **0.66** for *hiphop dataset* and highest value being **0.82** for *childrens_music dataset.*
- There is moderate correlation between the two features *acousticness* and *loudness*, and another two features acousticness and *energy*, with average coefficient value of **-0.48** and **-0.61** respectively in all dataset except *comedy* dataset. This indicates moderate negative linear relationship. In these 9 datasets, the correlation value for the two features is colored in black box, which indicates a high negative value from the scale bar.
- *Hiphop* dataset has the most negative correlation between variables, followed by *comedy* dataset.
- There is weak correlation between the two features *danceability* and *valence,* with average coefficient value of **0.43**. This indicates weak positive linear relationship. *Movie dataset* has the highest correlation value of **0.71**, and *hiphop dataset* has the lowest correlation value of **0.18**. between the two features.

The performance of the model can be affected by multicollinearity, which is caused by variables with high correlation. We can either eliminate one of the highly correlated features or use decision tree, gradient boosting and other algorithms, which are not affected by multicollinearity.

2. Boxplots (in jupyter notebook)

Excluded *loudness* because it has no obvious patterns compare to other.

- For *acousticness,* the *comedy* genre has the highest median value, and *ska* has the lowest median value. *Childrens_music* genre has the largest interquartile range. *Comedy* and *ska* have the most outliers
- For *danceability*, *hiphop* genre has the highest minimum value compare to other genres.
- For *energy*, *ska* genre has the most outliers and it has the highest median value compare to other genres.
- For *instrumentalness*, *electronic* is the only genre that has a wide range of value (it has wide interquartile range). All the other genre has significant outliers present.
- For *liveness*, comedy genre has the highest median value and the largest minimum and maximum values.

- For *speechiness*, *comedy* genre has significantly high median value compare to the rest. *Hiphop* genre has the largest interquartile range. All genre has significant outliers present.
- For *valence*, there is no outlier present. All genre has median value in the range 0.4 – 0.7. *Ska* genre has the highest median value.

Outliers in data can distort the data distribution, affect predictions and affect the overall accuracy of estimates if they are not detected and handled. Outliers can have a disproportionate effect on statistical results, such as the mean, which can result in misleading interpretations.

3. Bar graphs (refer to jupyter notebook for image)

From the bar graphs, I notice that *alternative*, *blues* and *ska* genre have similar distribution. *Childrens_music*, *folk* and *soul* genre have similar distribution. Having similar distribution might provide a false impression and fail to reveal key patterns.

4. Feature interaction plots

- Feature interaction plot between *loudness* and *energy*.

- We can derive from the correlation heatmap above that *loudness* and *energy* have strong correlation, which means they are associated with each other but this does not necessarily mean one causes another. The feature interaction plot allows us to understand better if one of them predicts the other.
- *loudness* is on the x-axis and *energy* is on the y-axis because the louder the music, the more energetic a person gets.
- The prediction values (red line) as seen in the plot, closely matches the actual value. This suggests that loudness can be used to predict energy.
- Thus, we can potentially remove either one as keeping both might be redundant.

- Feature interaction plot between *danceability* and *valence*.

- We can derive from the correlation heatmap above that *danceability* and *valence* have weak correlation, which means they are weakly associated with each other but this does not necessarily mean one does not cause another. The feature interaction plot allows us to understand better if one of them predicts the other or not.
- *danceability* is on the x-axis and *valence* is on the y-axis because the person is more likely to dance better with high valence sound.
- The prediction values (red line) as seen in the plot, does not matches the actual value as the actual values are scattered widely. This suggests that danceabiliy cannot be used to predict valence vice versa.

Part 2: Completion

Initial design

- Firstly, the 10 training datasets are combined into 1 training dataset. I have kept all variables except for *duration_ms*, *key*, *mode* and *tempo*. (This is because *key* and *mode* are not numerical values and *duration_ms* and *tempo* contains out of range and missing values respectively.)
- In the core part, we would prefer classifiers that are robust to multicollinearity. Thus, I have chosen to use **decision tree**, **random forest** and **gaussian naïve bayes**.

I have taken the idea from Assignment 1 where I had plotted the scores for each classifier.

I had used 20 simulations and train_test_split with an 80/20 split for good training size and hopefully reduce the risk of overfitting. Boxplots were produced.

Below is the table for the best scores

	Score	Paramerter
Decision Tree	0.44179	10
Random Forest	0.488655	10
GNB	0.401655	1e-09

KAGGLE

From the initial design, I had produced 3 csv file for the three classifiers. In Kaggle, I have got 0.31 for decision tree, 0.27 for random forest, 0.32 for Naïve Bayes. This indicates that models are overfitted.

Intermediary system

- In the intermediary system, I have added more features from the original dataset. The additional features I have added are the ones that I have excluded in the initial stage (*duration_ms*, *key*, *mode* and *tempo*).
- Some pre-processing steps need to be taken.
  - Missing values in *duration_ms* (“-1”) are being replaced with mean values for the column.
  - Missing values for *tempo* (“?”) are being replaced with mean values of the respective column too.
  - *Key* is in categorical from, so it is being replaced with numerical values. (C = 0 , C# = 1 ….. up to B=11).
  - *Mode* has only two labels (“Major” and “Minor), so it can be replaced in binary form (0 is “Minor”).

This updated new training dataset is then used for the steps later.

First stage

In this stage, I have used the same classifiers in the initial stage but with the new training set to check for any improvement. This can highlight how each classifier perform with extra features.

I have written up a function to test the 3 classifiers to find the average scores. I have used a 50/50 split instead to prevent overfitting.

The result is:

```
Decision Trr average score: 0.5396133333333333
Random Forest average score: 0.6625933333333334
Gaussain NB average score: 0.5187866666666666
```

From the result above, it looks like Random Forest performs the best with added new features (0.66).

Second stage

In this stage, I have used RFE (Recursive Feature Elimination) to identify irrelevant features. I have chosen **n_features_to_select = 9** , which means to remove 4 redundant features. This is because at the initial stage, I have excluded 4 features as well.

Out[94]:

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	valence
0	True	True	True	True	True	True	False	False	False	False	True	True	True

From above, we can remove *key*, *liveness*, *loudness* and *mode* (This are *false* in the table above)

In the core section, we found that there is strong correlation between the two features *loudness* and *energy*. This means that we can drop either one. The result above thus show that we can drop *loudness* and keep *energy* instead.

GridSearch CV is used for the classifiers to check for best parameters.

In [85]:
```
simplefilter(action='ignore', category=FutureWarning)

X_gv = train_data1.iloc[:,:-1]
y_gv = train_data1.iloc[:, -1]
X_gv = StandardScaler().fit_transform(X_gv)

DT_grid = { 'max_depth': [1, 3, 5, 8, 10]}
RF_grid = {'n_estimators': [50, 100, 150, 180], 'max_depth': [1, 3, 5, 8, 10], 'max_features': [2, 4, 6, 8, 9]}
NB_grid = {'var_smoothing': [1e-9, 1e-5,1e-1]}

DT_search = GridSearchCV(DecisionTreeClassifier(), DT_grid, cv=3).fit(X_gv, y_gv)
RF_search = GridSearchCV(RandomForestClassifier(), RF_grid, cv=3, n_jobs=-1).fit(X_gv, y_gv)
NB_search = GridSearchCV(GaussianNB(), NB_grid, cv=3, n_jobs=-1).fit(X_gv, y_gv)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=10, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best') Decision Tree best score : 0.5945798820733054

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=10, max_features=2,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=150,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False) Random forest best score : 0.6617799061275464

GaussianNB(priors=None, var_smoothing=1e-09) NB best score : 0.5236598916167603
```

Again, Random Forest performs the best with score of (0.66) using **max_depth** =10**,** **max_features** = 2 and **n_estimators**= 150.

Best Model

I have decided to choose my best model from the Intermediary system – 2nd stage model. This model has got a decent score of (0.67) while we remove the irrelevant features.

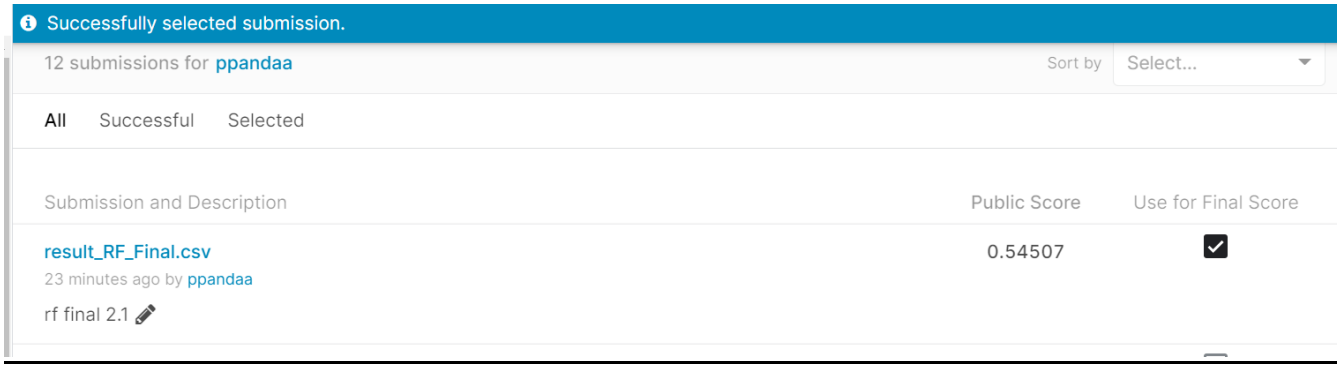
- Random Forest works great with high dimensional data. Compare to Decision Tree, Random Forest is faster to train as we work with a subset of features. Likewise, it would be rather easy to work with complex datasets like this music datasets.
- I have removed 4 redundant features in total.
- Classifiers are well optimised with its parameter from doing GridSearch CV.
- In fact, In the 2nd stage, the model performs almost similar as the model in stage 1 (very close score), but using lesser features in the 2nd stage.

Feature importance table:

	0	1
0	0.270334	popularity
1	0.117864	acousticness
2	0.074746	danceability
3	0.097821	duration_ms
4	0.079088	energy
5	0.078682	instrumentalness
6	0.199667	speechiness
7	0.044202	tempo
8	0.037596	valence

There are not in order but to sum up, *speechiness* has the highest value and *valence* has the lowest value.

Kaggle



Part 2 – Challenge

It is not easy to interpret my chosen machine learning model, which is Random Forest. Random forest combines multiple decision trees, and it becomes more difficult to interpret, especially with a large dataset. Random forest also has higher training time, compare to models such as decision tree.

Since I do not have good knowledge to understand how Random Forest does its job with large complex dataset, it does make it quite challenging for me to make any potential changes that could increase the accuracy score.

Compare to simpler model such as K-Nearest Neighbour, random forest has some advantages such as not sensitive to outliers, works well on large dataset like the one we use. Some disadvantage of random forest would be it is prone to overfitting, which is proven in the initial design, and it can produce low prediction accuracy compare to other machine learning algorithms, which suggest why the final score is below 0.70.

While using a more complex model, the variance increases and bias decrease. In order to build more accurate model, it is important to find the balance between bias and variance so that the model minimizes total error.