Group Assignment 1 - CS 986/CS 985

Group i - Peter Janik (201979128), Inno Mvula (201973944), Thom Reynard (201977555), Greig Govenlock (201516388)

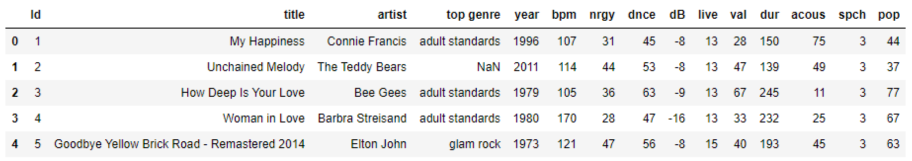
March 2nd 2020

I.I Hit Song Science

The *science* behind popularity perception has become of huge importance in particular for the music industry, estimated last year to be worth around 19.1 Billion (USD) [1]. There have been several studies conducted over the past 15 years trying to find the formula for the perfect song. These studies include considering social influence [2] as well as regression and classification techniques, [3] and there are even studies stating that it just can't be done [4].

I.II Introduction to Dataset

In this assignment, we were challenged to build and train a regression and classification model to predict the popularity and genre of each song in a test dataset [5, 6]. This dataset is already split into a test and train subset of 453 and 114 entries respectively each with 14 shared features comprising of object and integer values with the addition that the training set has a popularity value. In addition to the variables given within the dataset, an additional variable was



	Id	title	artist	top genre	year	bpm	nrgy	dnce	dB	live	val	dur	acous	spch	pop
0	1	My Happiness	Connie Francis	adult standards	1996	107	31	45	-8	13	28	150	75	3	44
1	2	Unchained Melody	The Teddy Bears	NaN	2011	114	44	53	-8	13	47	139	49	3	37
2	3	How Deep Is Your Love	Bee Gees	adult standards	1979	105	36	63	-9	13	67	245	11	3	77
3	4	Woman in Love	Barbra Streisand	adult standards	1980	170	28	47	-16	13	33	232	25	3	67
4	5	Goodbye Yellow Brick Road - Remastered 2014	Elton John	glam rock	1973	121	47	56	-8	15	40	193	45	3	63

Figure 1: This figure gives an example of the test data set and its 15 features; Id, Song Title, Artist, Genre, Release Year, Beats per Min., Energy, Danceability, Loudness(dB), Liveness, Valence, Duration, Acoustiness, Speechiness and Popularity.

created by combining the duration of the song with beats per minute to count the total number of beats in each song.

For the Regression analysis we have been asked to ignore the attributes Id and Popularity in the dataset. Similarly, for the Classification task we have asked to ignore the attributes Id and Genre. The nature of some of the more obscure song attributes is unknown. In total, after extracting the NaN values and converting and transforming the data, we have a total of 5694 [438 x 13] data points in which to build our models. The popularity in our dataset in an arbitrary integer value where the higher the value, the more popular the song has been.

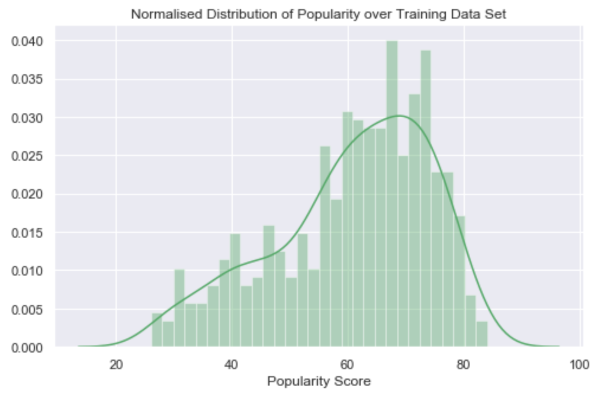


Figure 2: (Colour online) - Detailing the popularity distribution across the training dataset between values ranging from a minimum of 26 to a maximum of 84.

II Regression

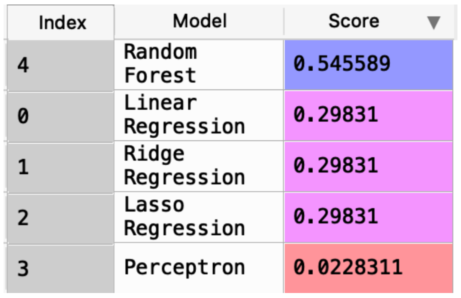
II.I Methods

Following the split of data into the training and test sets, several different regression methods were run to determine the effectiveness and accuracy of each in predicting the popularity of the songs in the test set. The regression models used include Linear, Logistic, Lasso, Ridge and Random Forest Regression in addition to a Perceptron and Gaussian Naive Bayes regression. The coefficients of determination (also known as R^2) values, found to be good predictors of model accuracy were compiled and compared across all models.

The Linear, Lasso and Ridge regressions all produced identical scores of 0.298310, similarly explaining 29.83 percent of the variance of the dependent variable in the model.

Random Forest regression is widely used in machine learning for its high accuracy and its ability to consider small subsets of features over an entire dataset. The Random Forest Regressor was found to produce the highest R^2 value of 0.545589 meaning that 54.56 percent of the variance of the dependent variable can be explained by the independent variables in the model.

Perceptron uses a method called neutral networks where it takes individual samples in steps to train its model. Although we found that it only produced a R^2 value of 0.022831. This indicates that the dataset has a lot of variance across its features and that we would need more information to improve this model.



Index	Model	Score ▼
4	Random Forest	0.545589
0	Linear Regression	0.29831
1	Ridge Regression	0.29831
2	Lasso Regression	0.29831
3	Perceptron	0.0228311

Figure 3: Summary of R^2 value for each regression method

1. **Linear Regression**
   We use a multivariate linear regression model to fit over several independant variables and calculate the accuracy of our model from the coefficient of determination (R^2) and the root mean square error (RMSE).

   Multiple Linear Regression Mathematical model:

   $$\hat{y} = \sum_{n=0}^{13} \Omega_n x_n \tag{1}$$

   Where $\hat{y}$ is the predicted value, n the number of features, x_i th term of the random error and Ω_j is the model parameter [7]. This implies our linear model would be of the form;

   $$Popularity = \Omega_0 * Title + \Omega_1 * Genre + \Omega_2 * Year ... + \Omega_{12} * Speechiness \tag{2}$$

   The accuracy of each regression model can be compared using the R^2 and the RSME value which are calculated by;

   $$R^2 = \frac{var(mean) - var(line)}{var(mean)}; \quad RSME = \sqrt{(\hat{y} - y)^2} \tag{3}$$

   Where var(mean) is variance from the average value of the model, var(line) is the variance from the line of best fit through the predictions and y is the known values [8].

2. **Logistic Regression**
   Logistic Regression analysis was used to predict the popularity of each song as it is an effective regression technique. We saw that the accuracy

for the logistical regression was relatively low which is due to the degree of the outcome being large. Meaning the variation of the popularity score ranges across almost 60 values whereas regression techniques are typically used for dichotomous outcomes [9].

Multiple Logistic Regression Mathematical Model (Normalised) [7];

$$p(\hat{y}) = \frac{1}{1 + \exp^{(-\sum \Omega_n y_n)}} \tag{4}$$

3. **Lasso and Ridge Regression**

   A common problem with the linear regression is that it over-fits the model due it not generalising well. To attempt to improve upon the linear models, we investigate Lasso and Ridge regression which have an additional error term to compensate for the high variance between the predicted and actual values of the test. Although Lasso regression is a better statistical model as it properly evaluates feature selection when you have multiple features that are correlated with each other.

   Lasso Regression Mathematical Model;

$$\hat{y_l} = \sum_{n=0}^{13} \Omega_n x_n + L_1 \tag{5}$$

   where $L_1 = \lambda_1 \sum x_n^2$ and λ is a hyper-parameter.

   Ridge Regression Mathematical Model;

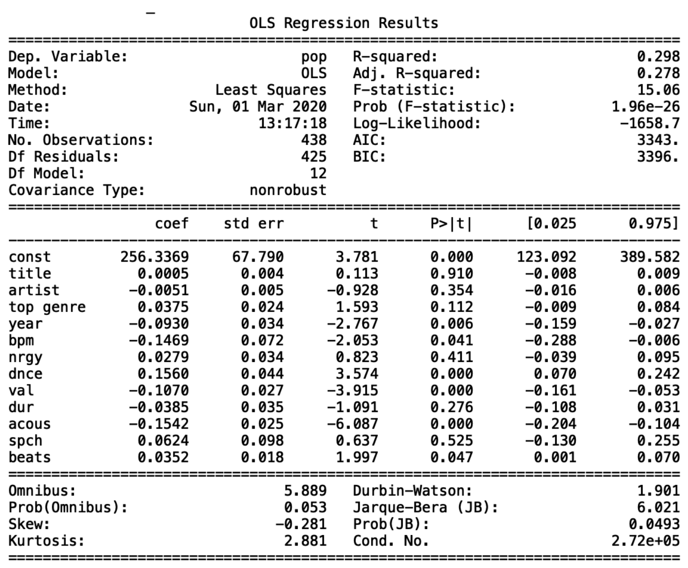
$$\hat{y_r} = \sum_{n=0}^{13} \Omega_n x_n + L_2 \tag{6}$$

   where $L_2 = \lambda_2 \sum |x_n|$ and λ is a hyper-parameter.

   Both models [10] here attempt to improve the fit of the linear regression but due to the large variety of the popularity, they do not improve upon the linear regression score. We were able to evaluate the regression models by using Statsmodel's OLS Regression Results as seen in Figure 4. The keys parameters are the Adj. R-Squared - relating to the fit of the model, $P > |t|$ - p-value where larger than 0.05 the parameter can deemed irrelevant and the coef - which describes how a one unit increase in each independant variable affects the dependant variable.

4. **Random Forest**

   Random Forest Regression uses the Ensemble Technique of Boosting by combining and running multiple Decision Trees in parallel. While for many it remains as "one of the most accurate learning algorithms available (...) [producing] highly accurate [models] (...), [it has] been observed to overfit some datasets with noisy classification/regression tasks, [also remaining] biased in favor of attributes with more levels" [11]. It is important to take this into consideration given our dataset where columns like 'top genre' contain a large amount of levels.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    pop   R-squared:                       0.298
Model:                            OLS   Adj. R-squared:                  0.278
Method:                 Least Squares   F-statistic:                     15.06
Date:                Sun, 01 Mar 2020   Prob (F-statistic):           1.96e-26
Time:                        13:17:18   Log-Likelihood:                -1658.7
No. Observations:                 438   AIC:                             3343.
Df Residuals:                     425   BIC:                             3396.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        256.3369     67.790      3.781      0.000     123.092     389.582
title          0.0005      0.004      0.113      0.910      -0.008       0.009
artist        -0.0051      0.005     -0.928      0.354      -0.016       0.006
top genre      0.0375      0.024      1.593      0.112      -0.009       0.084
year          -0.0930      0.034     -2.767      0.006      -0.159      -0.027
bpm           -0.1469      0.072     -2.053      0.041      -0.288      -0.006
nrgy           0.0279      0.034      0.823      0.411      -0.039       0.095
dnce           0.1560      0.044      3.574      0.000       0.070       0.242
val           -0.1070      0.027     -3.915      0.000      -0.161      -0.053
dur           -0.0385      0.035     -1.091      0.276      -0.108       0.031
acous         -0.1542      0.025     -6.087      0.000      -0.204      -0.104
spch           0.0624      0.098      0.637      0.525      -0.130       0.255
beats          0.0352      0.018      1.997      0.047       0.001       0.070
==============================================================================
Omnibus:                        5.889   Durbin-Watson:                   1.901
Prob(Omnibus):                  0.053   Jarque-Bera (JB):                6.021
Skew:                          -0.281   Prob(JB):                       0.0493
Kurtosis:                       2.881   Cond. No.                     2.72e+05
==============================================================================
```

Figure 4: (Colour online) OLS Regression Summary generated by the Statsmodel Python package

III Model Optimisation

Following the Regression Analysis in Section II, it is clear that based on the R^2 values produced by each regression model (see Figure 3), the Random Forest Regression is the optimal model of choice for data training, even when considering the negative aspects mentioned previously.

While the Random Forest Regression already performs significantly better than the other regression models, there are several available methods to further improve its prediction accuracy through model optimisation techniques like feature selection and hyper-parameter optimisation.

III.I Feature Selection

One of the most popular methods of model optimisation is feature (independent variable) selection. Increasing the number of variables typically results in a more accurate prediction, the inclusion of 'irrelevant variables' can lead to model over-fitting. This occurs when the model performs with great accuracy on the training set, but this performance does not translate to the test set. For that reason, it is key to find the right balance of maximising the number of predictors while avoiding the issue of over-fitting.

The two main methods used to find this balance: Step-wise and Best Subsets Selection, examined below.

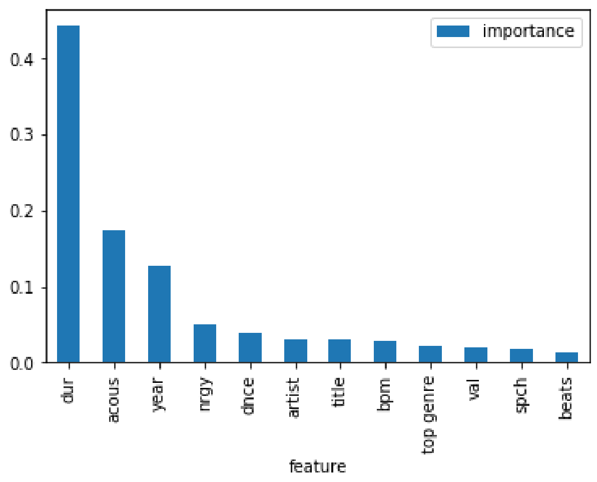


Figure 5: (Colour online) The importance of each feature calculated for the training dataset, thus making song duration along with acousticness and year the more important factors.

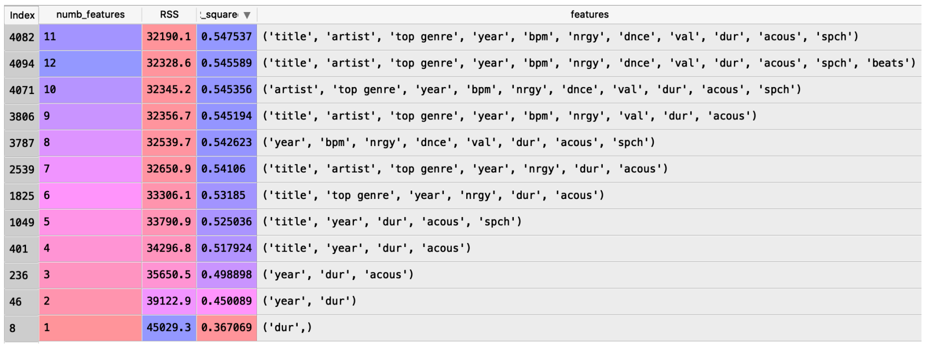
III.II Best Subsets Selection

This is a feature selection process by which every possible combination of independent variables within a model is tested to determine the 'best subset' leading to the highest statistical significance. This is typically determined by the lowest Residual Sum of Squares (RSS) value or alternatively the highest coefficient of determination (R^2).

As evident from Figures 6 and 7 after comparing every combination of the 12 independent variables (across 4094 models), the lowest RSS and highest R^2 scores are produced by models with 11 features (independent variables). These 11 features include all 11 initial independent variables within the dataset. Interestingly, adding the created variable BEATS improves accuracy to a greater extent than the removal of the variable and reducing the number of features to 10 (see Figure 6).

III.III Stepwise Feature Selection

Much like Best Subsets, Stepwise Feature Selection is a process used to estimate the optimal number of independent variables in a model to increase prediction accuracy. Unlike with Best Subset selection where the selection process is random, a forward and backward stepwise respectively adds or removes features based on their perceived significance to the outcome of the model. Similarly to the Best Subset Selection process, the initial analysis of RSS and R^2 provides indication that including 11 features remains optimal. However, the features included have changed with BEATS now found to be more significant to the



Index	numb_features	RSS	_square	features
4082	11	32190.1	0.547537	('title', 'artist', 'top genre', 'year', 'bpm', 'nrgy', 'dnce', 'val', 'dur', 'acous', 'spch')
4094	12	32328.6	0.545589	('title', 'artist', 'top genre', 'year', 'bpm', 'nrgy', 'dnce', 'val', 'dur', 'acous', 'spch', 'beats')
4071	10	32345.2	0.545356	('artist', 'top genre', 'year', 'bpm', 'nrgy', 'dnce', 'val', 'dur', 'acous', 'spch')
3806	9	32356.7	0.545194	('title', 'artist', 'top genre', 'year', 'bpm', 'nrgy', 'val', 'dur', 'acous')
3787	8	32539.7	0.542623	('year', 'bpm', 'nrgy', 'dnce', 'val', 'dur', 'acous', 'spch')
2539	7	32650.9	0.54106	('title', 'artist', 'top genre', 'year', 'nrgy', 'dur', 'acous')
1825	6	33306.1	0.53185	('title', 'top genre', 'year', 'nrgy', 'dur', 'acous')
1049	5	33790.9	0.525036	('title', 'year', 'dur', 'acous', 'spch')
401	4	34296.8	0.517924	('title', 'year', 'dur', 'acous')
236	3	35650.5	0.498898	('year', 'dur', 'acous')
46	2	39122.9	0.450089	('year', 'dur')
8	1	45029.3	0.367069	('dur',)

Figure 6: (Colour online) Summary of minimum RSS and maximum R^2 for each combination of features using Best Subsets Selection

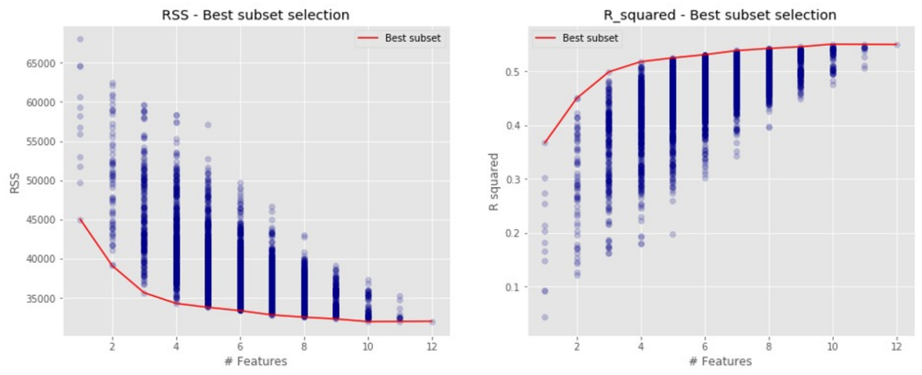
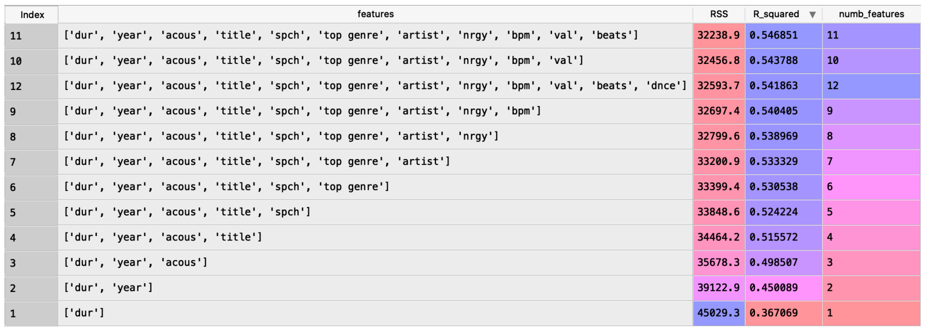


Figure 7: (Colour online) A Random Forest Regression model is run over 4094 iterations recording and appending the RSS and R^2 value generated with each iteration to determine the optimal number of features in the model

model then the Danceability variable.

It is often the case with machine learning algorithms where the test data Mean Squared Error (MSE) is higher than the training set MSE. This occurs when a model is "fit to the training data using least squares specifically [estimating] the regression coefficients such that the training RSS is minimized. In particular, the training RSS decreases as features [are added] to the model, [while] the test error may not. Therefore the training RSS and R^2 may not be used for selecting the best model unless we adjust for this underestimation." [12] This underestimation is taken into account when considering other key values such Mallow (C_p), Akaike Information Criterion (AIC), Bayes Information Criterion (BIC) and the adjusted R^2 variance across the model, which significantly punishes models with 'irrelevant variables'.

Following further analysis of the C_p, AIC, BIC and adjusted R^2 values, there is now indication to suggest that 10 variables may instead be optimal for this



Index	features	RSS	R_squared ▼	numb_features
11	['dur', 'year', 'acous', 'title', 'spch', 'top genre', 'artist', 'nrgy', 'bpm', 'val', 'beats']	32238.9	0.546851	11
10	['dur', 'year', 'acous', 'title', 'spch', 'top genre', 'artist', 'nrgy', 'bpm', 'val']	32456.8	0.543788	10
12	['dur', 'year', 'acous', 'title', 'spch', 'top genre', 'artist', 'nrgy', 'bpm', 'val', 'beats', 'dnce']	32593.7	0.541863	12
9	['dur', 'year', 'acous', 'title', 'spch', 'top genre', 'artist', 'nrgy', 'bpm']	32697.4	0.540405	9
8	['dur', 'year', 'acous', 'title', 'spch', 'top genre', 'artist', 'nrgy']	32799.6	0.538969	8
7	['dur', 'year', 'acous', 'title', 'spch', 'top genre', 'artist']	33200.9	0.533329	7
6	['dur', 'year', 'acous', 'title', 'spch', 'top genre']	33399.4	0.530538	6
5	['dur', 'year', 'acous', 'title', 'spch']	33848.6	0.524224	5
4	['dur', 'year', 'acous', 'title']	34464.2	0.515572	4
3	['dur', 'year', 'acous']	35678.3	0.498507	3
2	['dur', 'year']	39122.9	0.450089	2
1	['dur']	45029.3	0.367069	1

Figure 8: (Colour online) Summary of minimum RSS and maximum R^2 for each combination of features using Stepwise Selection

models prediction accuracy. Akaike's Information Criterion (AIC) is used to penalise the inclusion of additional unnecessary variables to a model, where a low AIC value indicates a strong model. The lowest AIC value was produced with a model containing 10 features. The Bayesian Information Criteria (BIC) much like AIC penalises unnecessary variables, however does so to a greater extent, indicating that 5 features may instead be optimal for model performance. Mallows C_p is again a variation of the AIC value used to measure model performance, and much like AIC indicates 10 features to be the optimal number.

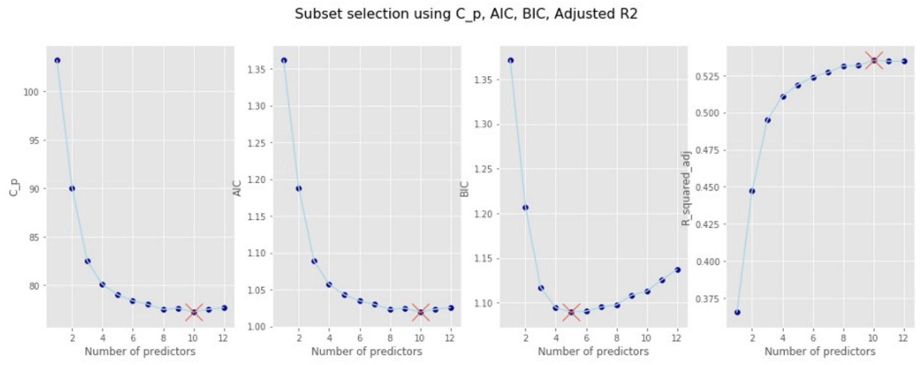
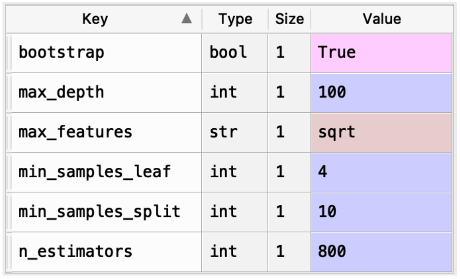


Figure 9: (Colour online) Displays the C_p, AIC, BIC and the adjusted R^2 variance across the model with different numbers of dependant variables.

III.IV Hyper-parameter Optimisation

Another form of model improvement is hyper-parameter optimisation whereby the parameters of the regression model used are adjusted to increase the R^2 value, thus overall increasing the accuracy of the models prediction capabilities. While each model has assigned default values to its own individual parameters, these can be further tuned to alter the models performance. Hyper-parameter tuning is heavily based on trial-and-error to determine the effect that a change may have on the prediction accuracy. This, however, similarly to feature selection can lead to the problem of over-fitting [13]. SKLearn has developed techniques to improve a user's ability to optimise model features through Random Parameter Search. Using a combination of K-Fold Cross Validation, a Randomised Search uses a predefined list of parameters specified and a random choice of values from the grid to determine the optimal choice of parameters for the model. The below figure indicates the optimal parameters chosen through RandomSearch:



Key	Type	Size	Value
bootstrap	bool	1	True
max_depth	int	1	100
max_features	str	1	sqrt
min_samples_leaf	int	1	4
min_samples_split	int	1	10
n_estimators	int	1	800

Figure 10: (Colour online) A summary of best parameters as selected by RandomSearchCV package by SKLearn

III.V Principal Component Analysis (PCA)

A prominent feature selection technique, PCA relies on "linear dimensionality reduction using an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components" [14]. These components can then be more easily analysed to determine relations within datasets.

While PCA is often used for data visualisation purposes to better understand the data being analysed, its main purpose remains improving the speed of the machine learning process, which given the small size of the dataset in this case is found to be unnecessary. Similarly, as found in the initial data analysis, there is minimal correlation between the variables, causing potential issues when creating each individual principal component. On this basis, PCA was omitted as a model optimisation technique from the assessment.

III.VI Findings and Conclusion

In reality, when training the model on the subsets of the 5, 10 and 11 variables indicated by the Best Subset and Stepwise Selection, the RMSE value as tested on Kaggle increased over the final model used with all parameters included. It is possible that this difference is due to the issues mentioned of the model favouring high-level categorical variables. This is later found not to be the case when the subsequent exclusion of each categorical variable also lead to an increase in the RMSE value within the test set. It is also possible the model was overfitted leading to a better performance on the training set than on the test set as is often the case.

This was similarly the case when optimising the models parameters. Using the optimal parameters generated using both the Random and Grid Search functions did not lead to an improvement in the overall RMSE of the model when comparing to the test dataset submitted on Kaggle.

IV Classification

IV.I Introduction

Following our initial exploratory analysis of the data we decided to tackle the problem using 3 methods using 3 classifiers. This included the Random Forest Classifier, Adaptive Boost in tandem with the Decision Tree Classifier and lastly the Ensemble Learning Classifier. Initial exploratory analysis of the distribution of genres revealed how imbalanced our dataset was. There were 86 genres, meaning 86 different classes to predict. Roughly 40 had a single instance and 3 genres were over-represented with over 60 instances each. Having discovered this we knew training the data as it is would produce less than desirable results. There are 10 audio features in the dataset as well. Because of the lack of significant correlation among the audio features we decided to use all of them. This is also why we overlooked Principal Component Analysis to reduce the dimensions of the dataset.

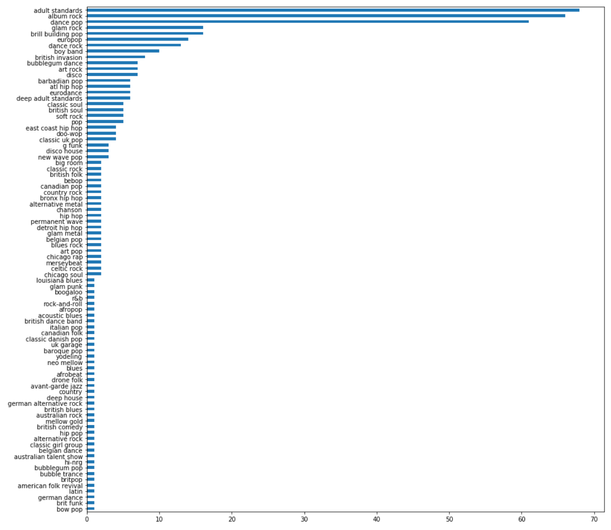


Figure 11: (Colour online) The distribution of the genres for the training data-set is shown to have 3 major categories with an abundance of others.

IV.II Class Imbalance

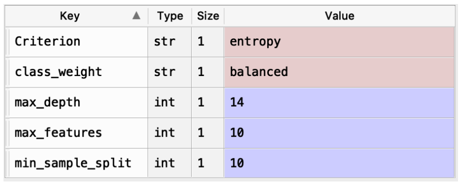
Imbalanced data sets are characterized by large skews in the distribution of classes where some classes are more represented than others. This can cause bias in our models as some machine learning algorithms end up ignoring minority classes and focusing more on majority classes. One approach to tackling this issue is to randomly re-sample the dataset. There are 2 main methods, Random Under Sampling and Random Over Sampling. To tackle this issue, we used a technique called ROS. We came to this decision after having observed the size of our dataset. RUS would involve deleting examples to match up to the number of instances in the minority class. Due to how small our data set was we didn't think it best to reduce it size any further. So instead we tackled the issue using ROS which involves duplicating examples of the minority class to match up to the number of observations in the majority class.

V Methods

V.I Adaptive Boost

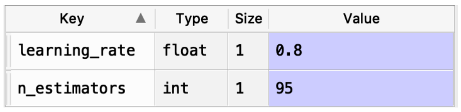
We started by building and training an adaptive boosted classifier, AdaBoost, with the Decision Tree classifier as its base. AdaBoost is useful in that it first trains a base classifier to make predictions on the training set. It then weighs training instances, increasing the relative weight of instances that have been mis-classified and the opposite for those that have been correctly classified. Since so many of our genres only have one instance, we expect many classifiers to have trouble making predictions and mis-classifying some instances based off the small amount of information. Hence the use of the AdaBoost. After training the base classifier it then trains the main classifier using the updated weights which allows the classifier to make better predictions.

To further enhance our model, we used ROS to balance our classes, in effect creating a larger dataset which reduces the bias towards the majority classes. In addition, we applied a technique called hyper-parameter tuning to both our base classifier and main classifier. This involves selecting some parameters from the classifier that we would like to adjust with various values. We used the function GridSearch to iterate through the various values to find the combination that produces the best possible result for each parameter. Figure 12 is the result of tuning the base decision tree classifier and Figure 13 is our main classifier.



Key ▲	Type	Size	Value
Criterion	str	1	entropy
class_weight	str	1	balanced
max_depth	int	1	14
max_features	int	1	10
min_sample_split	int	1	10

Figure 12: The best parameters determined by the base Decision Tree Classifier.

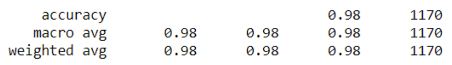


Key ▲	Type	Size	Value
learning_rate	float	1	0.8
n_estimators	int	1	95

Figure 13: The best parameters determined by Adaptive Boost Classifier with a prediction rate of 95%.

Having tuned our classifiers we trained and evaluated our model. A great way to do this is using a classification report. The report lists all our classes as well as their individual precision, recall, f1-score, and support scores. In addition, we get overall metrics for the model including accuracy. As you'll see in Figure 14, the overall model achieved high scores, with an average of 0.98 across all metrics. Interestingly, all of the individual genres that had the fewest cases achieved high scores across all metrics with high accuracy. This is the case due to the duplication of a significant number of instances.

Training the machine using this highly scored model over several iterated instances is much more likely to produce a prediction with a high accuracy. Also, of note is the scores of our frequent genres, adult standards (Precision: 0.73, Recall: 0.62, f1-score:0.67), album rock (Precision: 0.73, Recall: 0.57, f1-score: 0.64) and dance pop (Precision: 0.78, Recall: 0.54, f1-score: 0.64). This is due to the increased amount of information contained within these classes, producing more statistically significant results. The classifier also has more trouble making predictions when ROS is not used as scores were highest amongst most frequent genres and lowest for the least frequent genres.



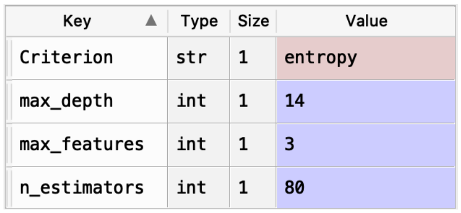
```
    accuracy                           0.98      1170
   macro avg       0.98      0.98      0.98      1170
weighted avg       0.98      0.98      0.98      1170
```

Figure 14: Classification Report for the Adaptive Boost Classification Model, which evaluates precision, recall and f1-score respectively.

V.II Random Forest

The Random Forest Classifier is known for its ability to produce high accuracy scores and to handle data-sets with large dimensionality. Just as with the previous model, it was evaluated using precision, recall and accuracy as metrics. At first, without using ROS, the model performed well, according to the results of the metrics, as expected. The model produced low precision, recall and accuracy scores for classes that were less frequent and high scores for the majority classes. The next step was to implement an over sampling method to see if by balancing the data the performance could be improved, as done when previously building the model with Adaptive Boost. A version of oversampling called Synthetic Minority Over-sampling Technique (SMOTE) was used, where unlike with ROS the SMOTE creates new training instances which are based on the original examples. After some experimentation, 4 new instances were chosen to be added to each minority class. This increased the size of our dataset to 513 observations, providing the model with more examples to learn from. Hyper-parameter optimisation was one again used to tweak several parameters belonging to the Random Forest Classifier. The results of the tuning are shown below:

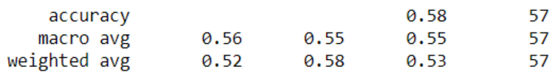
After tuning we trained and fitted our model. Upon evaluating the performance we observed a significant improvement in our metric scores for all the



Key	Type	Size	Value
Criterion	str	1	entropy
max_depth	int	1	14
max_features	int	1	3
n_estimators	int	1	80

Figure 15: The 'best parameters' determined by the Hyper-Optimisation method for the Random Forest Classification.

minority genres, however scores of the majority classes decreased. As for the overall model, the results were mediocre compared to the boosted model we trained earlier, as seen in the Figure below.



```
    accuracy                          0.58        57
   macro avg       0.56      0.55     0.55        57
weighted avg       0.52      0.58     0.53        57
```

Figure 16: Classification Report for the Random Forest Classification Model, which evaluates precision, recall and f1-score respectively.

V.III Ensemble Learning

Having constructed a number of classifiers to test from, the next step was to implement a voting classifier. Individually some of the classifiers were not as powerful as expected but produced enough variation so that a combination of multiple classifiers through the ensemble technique would produce statistically significant results. The nature of the dataset resulted in a range of performances for individual classifiers justifying the use of soft voting rather than hard voting. A wide variety of classifiers in the voting was included with examples shown below:

1. A decision tree with a max depth of 5 and using entropy as the criterion. A "for" loop was used to test a large number of different max depths and 5 gave the best performance.

2. A one vs one support vector machine is not the best suited classifier for this kind of problem but it was included for variety, with the impact of a relatively poor performance being minimised by the soft voting style.

3. Another random forest classifier was included but this time using the Gini Criterion yielded a better result for the model, whilst also having the less immediate benefit of making it more distinct from the decision tree already included.

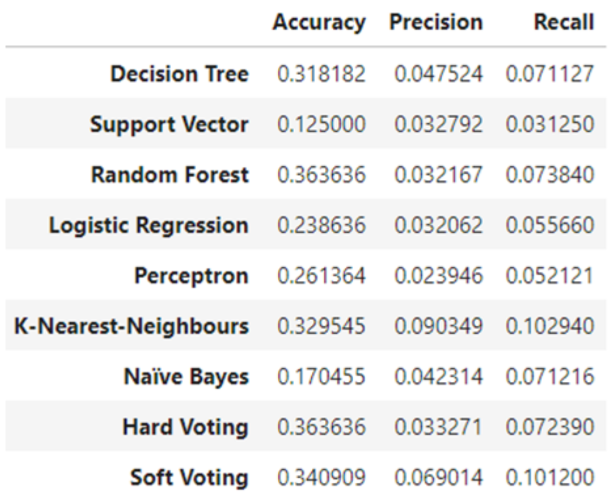
4. Logistic Regression without class weight. Balanced class weights were tested to compensate for the skewed dataset but this actually led to a worse performance.

5. A multi-layer perceptron, using “tanh” as activation.

6. A K-Nearest-Neighbours Classifier. As with the decision tree, a “for” loop was used to determine that setting the number of neighbours to 11 resulted in the highest prediction accuracy.

7. A Naïve Bayes classifier. Another relatively poor performing classifier when used alone, this was included for variety.

Both hard and soft voting strategies were implemented but soft voting performed better, as expected. This is because of the difference in performance where some classifiers coped better than others with the significant difference in genre sample sizes in the training set.



	Accuracy	Precision	Recall
Decision Tree	0.318182	0.047524	0.071127
Support Vector	0.125000	0.032792	0.031250
Random Forest	0.363636	0.032167	0.073840
Logistic Regression	0.238636	0.032062	0.055660
Perceptron	0.261364	0.023946	0.052121
K-Nearest-Neighbours	0.329545	0.090349	0.102940
Naïve Bayes	0.170455	0.042314	0.071216
Hard Voting	0.363636	0.033271	0.072390
Soft Voting	0.340909	0.069014	0.101200

Figure 17: Classification Report for the 7 classification models used in the Ensemble method as well as the Hard and Soft Voting methods.

The soft voting classifier was somewhat effective but did not produce the highest score. The inclusion of “weighted votes” within the ensemble perhaps dispersed the strength of weaker classifiers, namely the support vector machine and the Naïve Bayes classifier. These two classifiers are typically better suited to binary classification methods however the inclusion of different strategies account for weaknesses in the other classifiers within the ensemble, those better suited to the nature of the given dataset. Nonetheless, soft-voting resulted in a better precision score for each of its component classifiers and a better recall score than all classification model except K-Nearest-Neighbours.

Kaggle Competition Performance

Regression

The best score for our Random Forest regression predictions was from TEAM I(Peter) with 7.00519.

Classification

The best score for our Adaptive Boost regression predictions was from TEAM I with 0.39285.

References

[1] Music Business Association. IFPI Global Music Report 2019. https://musicbiz.org/news/ifpi-global-music-report-2019-global-music-market-grows-for-fourth-consecutive-year/, 2019. [Online; accessed 28-February-2020].

[2] Matthew J Salganik, Peter Sheridan Dodds, and Duncan J Watts. Experimental study of inequality and unpredictability in an artificial cultural market. *science*, 311(5762):854–856, 2006.

[3] Noam Koenigstein, Yuval Shavitt, and Noa Zilberman. Predicting billboard success using data-mining in p2p networks. In *2009 11th IEEE International Symposium on Multimedia*, pages 465–470. IEEE, 2009.

[4] François Pachet and Pierre Roy. Hit song science is not yet a science. In *ISMIR*, pages 355–360, 2008.

[5] Nicolas Carbone. CS985/6 Spotify Regression Problem. https://www.kaggle.com/c/cs98x-spotify-regression/overview, 2020. [Online; accessed 27-February-2020].

[6] Nicolas Carbone. CS985/6 Spotify Regression Problem. https://www.kaggle.com/c/cs98xspotifyclassification/overview, 2020. [Online; accessed 27-February-2020].

[7] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[8] William Mendenhall, Terry Sincich, and Nancy S Boudreau. *A second course in statistics: regression analysis*, volume 5. Prentice Hall Upper Saddle River, NJ, 1996.

[9] Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1):3–14, 2002.

[10] Saptashwa Bhattacharyya. Ridge and Lasso Regression: L1 and L2 Regularization. https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b, 2020. [Online; accessed 02-March-2020].

[11] Will Koehrsen. An Implementation and Explanation of the Random Forest in Python. https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76, 2020. [Online; accessed 02-March-2020].

[12] Xavier Bourret Sicotte. Choosing the optimal model: Subset selection. https://xavierbourretsicotte.github.io/subset$_{s}election.html, 2020. [Online; accessed 02 - March - 2020]$.

[13] Will Koehrsen. Hyperparameter Tuning the Random Forest in Python. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74, 2020. [Online; accessed 02-March-2020].

[14] Aditya Sharma. Principal Component Analysis (PCA) in Python. https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python, 2020. [Online; accessed 02-March-2020].