Kaggle Competition Report

Team name: Widob

Members	**Matricules**	**Emails**
Wissal Hamhoum	2213908	wissal.hamhoum@polymtl.ca
Brando Tovar	1932052	grover-brando.tovar-oblitas@polymtl.ca
Doha Zrouki	1958859	doha.zrouki@polymtl.ca

1 Feature Design

The first step in data science is to understand the data we have, this process is called data exploration, this step is crucial to perform a logical and efficient data cleaning and preprossessing subsequently. The feature design can hence be split into a few steps, that are presented below.

1.1 Data Exploration

The data provided in this competition is divided into two csv files: training and testing. files having 170 000 and 30 000 entries, respectively. The information is organised into 19 columns, 18 of which are features and the last one (Concert Enjoyment) is the target. For the features, we can distinguish three categories:

- Features related to the band: (Band Name, Band Country of Origin, Genre, and Band Debut), have information about each band's name, its genre, its debut year, and its country of origin.
- Features related to the concert: (Concert ID, Inside Venue, Concert Attendance, and Rain); they describe the concert's characteristics, such as the number of attendees, whether it was in an inside a venue or not, and whether it rained at the time of the concert or not.
- Features related to the concert goer (Seated, Personnality Trait 1, Personnality Trait 2, Personnality Trait 3, Personnality Trait 4, Height, Concert Goer Country of Origin, Concert Goer ID and Concert Goer Age), these features, on the other hand, give information about the concert attendee's physical and personality traits as well as their age and ID.

As for the labels, we are dealing with 4 classes, describing the level of enjoyment experienced by the concert goer: Worst Concert Ever, Did Not Enjoy, Enjoyed and ”Best Concert Ever”.
To further understand the data we used the pandas.DataFrame functions info() and describe().
The first one gives us information about the missing data. As we can see in figure A in the appendix we have missing values in most of the training data apart from the ID and the target column. The describe function, gives us numerical information about the dataset such as the mean, the std and count. We noticed that most of the numerical features have significant standard deviation except the ”Personality Trait 1”, ”Personnality Trait 3” and ”Personnality Trait 4”. This will help us later when making decisions in the feature selection task.

1.2 Data Cleaning

After examining the data we are working with, the next step would be to correct the missing values in order to train a model with as little noise as possible. In the following subsections, we present the methods used to fill in these missing data points for each of the three groups of features mentioned before.

1.2.1 Features related to Concert Goers

For the Concert Goers we had access to the Concert Goer ID which was very helpful to fill the data. Indeed, we assumed that two concert goers with the same ID should share the same characteristics related to a person (Personnality Trait 1, Personnality Trait 2, Personnality Trait 3, Personnality Trait 4, Height, Concert Goer Country of Origin, Concert Goer ID and Concert Goer Age). Therefore we filled the data using the mode of each of these columns ensuring that all the data related to a concert goer is the same. If the ID was missing we simply filled with the most similar row of the data by using the euclidean distance.

1.2.2 Features related to the Bands

For the band related features, we distinguish two types of missing data: those having ”nan” as values and those having ”Insert name of the column” as values. We started by changing all the missing values to have nans so it will become easier to detect them. the actual number of missing data in this category is persented in the table 1

Table 1: The number of missing data per column in concert related data

Feature	**The number before**	**The actual number**
Band Genre in training data	883	964
Band Genre in testing data	138	157
Band Country of Origin in training data	790	970
Band Country of Origin in testing data	156	182
Band Debut in training data	857	857
Band Debut in testing data	157	157

After that, we used the band name column to deduce the missing values in the other band columns, and vice versa. After repeating the process multiple times, fixing one column using the others, we managed to correct all the missing data for this group of features.

1.2.3 Features related to the Concerts

For the Concert features, we have the columns: Concert ID, Concert Attendance, Inside Venue and Rain. We can say that for the same Concert ID, the other columns’ values should be the exact same. Therefore, to fill the missing values in these columns, we looped through all the unique Concert ID values and filled the missing values with the most frequent value. We also replaced the values that were not equal to the most frequent value. For example, for the same concert ID value, the concert attendance value should be the same, however, we noticed one value that is 1000 times bigger, which means that 3 zeros were accidentally added during the data entry process.

After filling missing values and wrongly entered values by mapping columns that are related, we still had a few missing values, in the ”independent” columns like Concert ID, Concert Goer ID and Band Name, there were filled with the mean values across their respective columns.

1.3 Outliers and Normalization

After fixing the problems in the testing and training datasets, we should now prepare the data to train the classification model. Depending on the feature’s type, different treatments were applied. In the case of numerical features such as ’Band Debut’, ’Concert Attendance’, ’Personnality Trait 1’, ’Personnality Trait 2’, ’Personnality Trait 3’, ’Personnality Trait 4’, ’Concert Goer Age’, ’Height (cm)’ we used normalization which is one of the most important steps in data pre-processing, that consist in transforming these features in a common range. The purpose of this step is to minimize

the bias introduced by the features having higher range in discriminating pattern classes and ensuring that all the numerical features have ”equal numerical contribution” Singh & Singh (2020). Numerous methods are usually used to normalize data such as: mean and standard deviation based, minimum-maximum based, decimal scaling normalization (DSN), median and median absolute deviation normalization (MMADN), tanh based normalization (TN) and sigmoidal normalizationSingh & Singh (2020). In our case we tried 2 different methods and compared their impact on the classification result by training the ”catboostclassifier”, that will be presented later.
The tested methods are:

• z-score : normalize the features by subtracting the mean and dividing by the standard deviation.

$$z = \frac{x - u}{s} \tag{1}$$

where u is the mean and s the standard deviation

• Min-max normalization: normalize by applying the equation 2

$$z = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

We noticed that the difference between the results given by the two methods, wasn’t significant, therefore, we kept z-score since its results were slightly better.

Because the normalisation can be strongly affected by the outliers, this step was preceded by outlier removal operation where we measured the z-score of the features and removed those having $\mid score \mid$ greater than 3, which represents 0.3 % of the dataset. This threshold was selected after examining the distribution of the values in the dataset.

1.4 Data Encoding

The preparation of categorical and boolean features is done by encoding, i.e transforming them into a numerical representation.

1.4.1 Categorical Features

Categorical data can be divided into two groups Dahouda & Joe (2021) :

- Nominal: we cannot attribute any kind of order between the values
- Ordinal: there is some kind of order in the values.

In the case of ordinal features, the label encoding or the ordinal encoding should be applied to reflect the ordinal meaning of the data. Dahouda & Joe (2021) The label encoding consists in assigning an integer to each categorical value. Inversely, the encoding of nominal group should reflect the non ordinal character of the features, the used techniques are: one-hot encoding and dummy encoding Dahouda & Joe (2021). In these techniques for each value, of the feature to encode, a new column is created and 1 is attributed if this feature exists and 0 otherwise.
In our data, the target variable is an ordinal categorical variable so we applied an ordinal encoding; Worst Concert Ever = 0, Did not Enjoy = 1, Enjoyed = 2 and Best Concert Ever = 3. As for the other categorical features, ’Band Name’, ’Band Genre’, ’Band Country of Origin’ and ’Concert Goer Country of Origin’, we used a one-hot encoder because they are nominal features.

1.4.2 Boolean Features

For the boolean features, we converted them to integer by attributing 1 to the ”True” values and 0 for the ”False” ones.

2 Algorithms

After preparing the data, multiple models were trained. First, to assess if the data pre-processing was successful and yields a satisfying performance score, and second, to compare the models performances, choose the best ones and tune their hyper-parameters.

In total, 8 models were trained : Linear discriminant analysis, suport vector machine (SVM), logistic regression, random forest, decision tree, XGB classifier, CatBoost and multilayer perceptron (MLP). The f1 scores and confusion matrices were computed for the training and validation data, before and after hyper-parameter tuning, table 2 in the result section compiles the f1 scores of the models.
The 8 models were chosen based on what we learned in class, what we used in the previous assignments, as well as after researching what were the most commonly used and performing machine learning algorithms on the internet, specifically in the scikit-learn documentation scikit-learn developers (2022a).
The main models we focused on in this assignment are Catboost, XGB Classifier and MLP. The Catboost classifier, which is a ”high-performance open source library for gradient boosting on decision trees” Yandex (2022), was initially chosen because in theory, the model does not need hyper-parameter tuning and should ”provide great results with default parameters” Yandex (2022). This model is also significantly faster compared to XGB for example, according to the developers. This model has also the advantage of performing the data transformation of categorical features to numerical values Yandex, however, this step was not used since we needed to pre-process the data to train other models too.
The XGB Classifier stands for Extreme Gradient Boosting and ”provides a parallel tree boosting”xgboost developers (2022). This model is known to be fast and have good classification performances, and is the go-to model for Kaggle competition winnersBrownlee (2016).
Finally, the multilayer perceptron model trains using gradient descent, with gradient calculations are performed using backpropagation. The loss function the model tries to minimize is the Cross-Entropy loss function scikit-learn developers (2022b).

3 Methodology

In this section we will present the steps we followed to train our models:

3.1 Feature selection

After preparing the data we wanted to study the importance of these features by using **sklearn** function **SelectKBest**. This function calculates the f_classif or the chi-squared between each feature and the target in order to measure the independence between the two.Bhatt (2019). Figure 2 in the appendix shows that some features have more importance than others, so we exploited this graph and tried different values of k to have different sets of features and we tested all the features sets.

3.2 Feature engineering

Since we used one hot encoding for our categorical features, our data resulted in very high dimensionality. The two feature most responsible for the high dimensionality were Concert Goer Country of Origin and Band Country of Origin that gave us more than a 100 additional columns. Since ordinal encoding categorical features is not a good idea for most models, we still decided to use one hot encoding since it also gave us better results in practice. Yet there was several drawbacks. There was no way that our model could have trained with most countries, so inevitably another dataset would have problems predicting in terms of new unseen countries. So instead of the countries as features we decided to replace them with a feature called Same Country that verifies if the Band Country of Origin is the same as the Concert Goer Country of Origin. Another intuition for this is that the concert enjoyment of a concert goer might be affected by the fact that the band is local to his country of origin. This new feature improved our accuracy.

3.3 Training and hyper-parameter tuning

Hyper parameter tuning is an important step in any machine learning project where we search for the best hyper-parameters for our model. One of the techniques used in tuning the models is random search. In this method an exhaustive list of hyper-parameters and their possible ranges is constructed, then at each iteration, a random combination is sampled from the hyper-parameters distribution. Due to its randomness, it is most likely to cover most of the action space in less time than the grid search and thus it gives faster and better results.
In our case, we performed random search for each tested model by preparing a list of the hyper-parameters to be tuned. This list, evidently depends on the type of the model.

4 Results

4.1 Performances of the tested models

All 8 models were tested with different feature combinations (using [10, 20, 50, 100, ... , k] best features). Table 2 shows the training and validation f1 scores with the final version of the feature matrix (with all the one-hot encoded features + a columns that maps the correspondence between the band's and the concert goer country of origin, as explained in the methodology section). This version achieved the highest scores, the previous scores are not presented in this report since they are too numerous.

Table 2: f1 scores of different models on the training and validation sets

Models	Train f1 score	Validation f1 score
Linear Discriminant Analysis	0.59659	0.59634
Support Vector Machine	0.60262	0.60302
Logistic Regression	0.61870	0.61792
Random Forest	**0.99429**	0.65050
Decision Tree	**0.99429**	0.54993
XGB Classifier	**0.62358**	**0.62130**
CatBoost Classifier	**0.70152**	**0.67285**
Multilayer Perceptron	**0.68389**	**0.66279**

From table 2, we observe that the top 3 models with the best validation scores are the XGB Classifier, the Catboost Classifier and the multilayer perceptron (MLP), with the best score achieved by the Catboost Classifier. We can also see that the Random Forest and Decision Tree models are overfitted and yield the same training result. The similarity between these 2 models is expected, since the Random Forest model "combines several decision trees" Vadapalli (2022). To resolve the overfitting, a hyper-parameter tuning was performed on these 2 models (as well as on the other ones) but the scores were not satisfying.

4.2 Hyper-parameter tuning

Hyper-parameter tuning was performed on all the tested models, For the sake of brevity and clarity, we will explain the tuning process of the top 3 models only.

For every model, we extracted the hyper-parameters from the corresponding library documentation and choose the values based on the default values, as well as most common values from online examples. Table 3 presents the hyper-parameters explored for each model.

Table 4 shows the performance scores of the top 3 models before and after the hyper-parameter tuning using the RandomizedSearchCV function.

We can conclude that the CatBoosst Classifier is the best model for the data we have. We trained it with the tuned hyper-parameters on the entire training set and the submitted prediction on the test set on Kaggle gave us a f1 score of 0.67466.

Table 3: Hyper-parameters used for the tuning step

XGB Classifier	CatBoost Classifier	Multilayer Perceptron Classifier
iterations = np.arange(100, 2000, 100)	iterations = np.arange(100, 2000, 100)	hidden_layer_sizes=[(2 ** i,) for i in range(1, 10)]
learning_rate=[0.1 , 0.01 , 0.001]	learning_rate = np.arange(0.05, 0.15, 0.01)	activation = [”identity”, ”logistic”, ”tanh”, ”relu”]
max_depth=[3 , 5 , 7]	depth = np.arange(8, 15, 1)	solver = [”lbfgs”, ”sgd”, ”adam”]
colsample_bytree = np.arange(0.2 , 1 , 0.4)	l2_leaf_reg = np.arange(1, 10, 1)	alpha = np.arange(0.0001, 0.1, 0.0001)
subsample = np.arange(0.7 , 1 , 0.1)	border_count ñp.arange(1, 255, 1)	learning_rate = [”constant”, ”invscaling”, ”adaptive”]
gamma = [0 , 1 , 5]		learning_rate_init = np.arange(0.0001, 0.1, 0.0001)
n_estimators = np.arange(50 , 150 , 25)		

Table 4: Scores of the top 3 models before and after hypertuning

	Before hyper-parameter tuning		**After hyper-parameter tuning**	
Models	**Train f1 score**	**Validation f1 score**	**Train f1 score**	**Validation f1 score**
XGB Classifier	0.62358	0.62130	0.65161	0.64632
CatBoost Classifier	**0.70152**	**0.67285**	**0.78304**	**0.67546**
Multilayer Perceptron	0.68389	0.66279	0.65324	0.65392

5 Discussion

Our approach was pretty intuitive and followed what is generally used in data science. We first inspected the data, then filled the missing and incoherent values. We processed our categorical features either using one hot encoding or feature engineering. We removed outliers and also normalized our numerical data. We then train multiple models and fine-tuned them while choosing the one that gave us the best validation score.

One of the pros of our approach is that it gave us a pretty decent result on the test set of 0.67466, which is a similar score of what we obtain on the validation set. We believe that our feature ’Same Country’ will also improve the generalization of our model. When our model will ’see’ new countries for example, it will still be able to know whether or not the concert goer is from the same country of the band. One other pro of our methodology is that we fine tuned our models manually before choosing the one that had the best score which improved the odds of choosing the correct one.

One of the cons of our methodology is that by removing countries as a feature we might have lost some useful information for the prediction. For example some concert from a particular country might enjoy more concerts overall than other countries. What we could have done is keep the countries as features along the Same Country feature and see if it gave a better result. Another con comes from the fact that we manually tried our models. This was a long process and it might have been a better option to use the LazyClassifier from the lazypredict library to automate the processus of finding the best model. Another con is that we didn’t explore the possibility that grouping the data by a particular feature might have given us a more predictable model. For example, concerts where inside venue is set to true might be better predicted with a different model than those with inside venue at false. We would then have two expert models that, combined, would have the potential to give us a better score on the whole dataset.

6 Statement of Contribution

The work on this project was equally distributed between all three team members. We all did the data exploration step prior to our first meeting. The data cleaning step was split according to the 3 categories of data (concert goer related, concert related and band related). We then each trained and tuned the hyper-parameters of 2-3 models. We also had collaborative sessions where we all coded together. As for the report, we initially split the sections, and then revised it and finalised it together.

”We hereby state that all the work presented in this report is that of the authors.”

References

Bhavesh Bhatt. Chi-squared for feature selection using selectkbest, 2019. URL `https://bhattbhavesh91.github.io/chi-square-feature-selection/#:~:text=We%20calculate%20Chi-square%20between,the%20independence%20of%20two%20events`.

Jason Brownlee. A gentle introduction to xgboost for applied machine learning, 2016. URL `https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/`.

Mwamba Kasongo Dahouda and Inwhee Joe. A deep-learned embedding technique for categorical features encoding. *IEEE Access*, 9:114381–114391, 2021. doi: 10.1109/ACCESS.2021.3104357.

scikit-learn developers. Supervised learning, 2022a. URL `https://scikit-learn.org/stable/supervised_learning.html#supervised-learning`.

scikit-learn developers. 1.17. neural network models (supervised), 2022b. URL `https://scikit-learn.org/stable/modules/neural_networks_supervised.html#neural-network-models-supervised`.

Dalwinder Singh and Birmohan Singh. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524, 2020. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2019.105524. URL `https://www.sciencedirect.com/science/article/pii/S1568494619302947`.

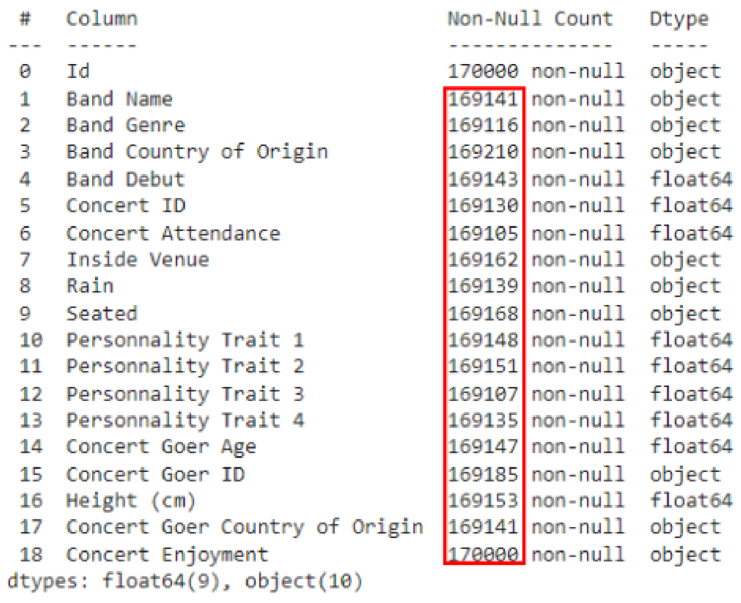
Pavan Vadapalli. Random forest vs decision tree: Difference between random forest and decision tree, 2022. URL `https://www.upgrad.com/blog/random-forest-vs-decision-tree/#:~:text=A%20decision%20tree%20combines%20some,forest%20model%20needs%20rigorous%20training`.

xgboost developers. Introduction to boosted trees — xgboost 1.7.1 documentation, 2022. URL `https://xgboost.readthedocs.io/en/stable/tutorials/model.html`.

Yandex. How training is performed. URL `https://catboost.ai/en/docs/concepts/algorithm-main-stages`.

Yandex. Catboost is a high-performance open source library for gradient boosting on decision trees, 2022. URL `https://catboost.ai/`.

A Appendix



```
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   Id                           170000 non-null  object
 1   Band Name                    169141 non-null  object
 2   Band Genre                   169116 non-null  object
 3   Band Country of Origin       169210 non-null  object
 4   Band Debut                   169143 non-null  float64
 5   Concert ID                   169130 non-null  float64
 6   Concert Attendance           169105 non-null  float64
 7   Inside Venue                 169162 non-null  object
 8   Rain                         169139 non-null  object
 9   Seated                       169168 non-null  object
 10  Personnality Trait 1         169148 non-null  float64
 11  Personnality Trait 2         169151 non-null  float64
 12  Personnality Trait 3         169107 non-null  float64
 13  Personnality Trait 4         169135 non-null  float64
 14  Concert Goer Age             169147 non-null  float64
 15  Concert Goer ID              169185 non-null  object
 16  Height (cm)                  169153 non-null  float64
 17  Concert Goer Country of Origin  169141 non-null  object
 18  Concert Enjoyment            170000 non-null  object
dtypes: float64(9), object(10)
```

Figure 1: Data info.

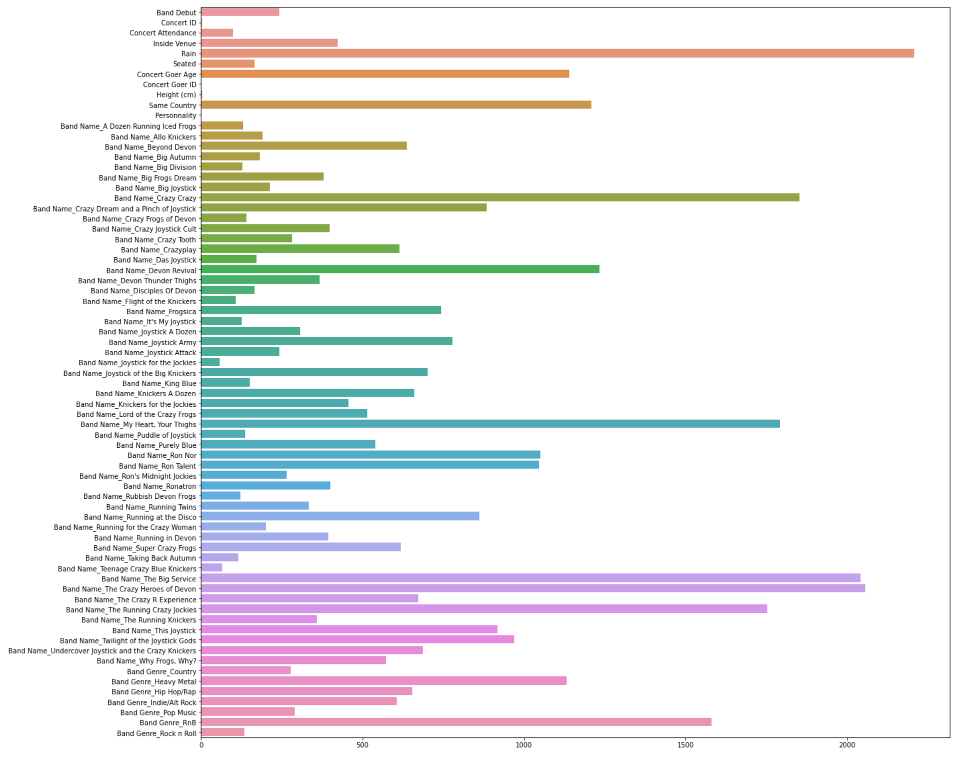


Figure 2: SelectKbest function plot