INF8245E - KAGGLE COMPETITION REPORT

Determine how much enjoyment a concert goer will have at a particular concert

Team name on Kaggle : TEAM UTC

César Bertrand
Matricule : 2232077
cesar.bertrand@{polymtl.ca,etu.utc.fr}

Yoann Betton
Matricule : 2232112
yoann.betton@{polymtl.ca,etu.utc.fr}

Stéphane Branly
Matricule : 2232279
stephane.branly@{polymtl.ca,etu.utc.fr}

ABSTRACT

The goal of this project is to design a machine learning algorithm that, given information on a particular concertgoer experience, can automatically classify the enjoyment of that concertgoer to that concert. In this classification problem, there are 4 classes. The training dataset consists of 170,000 training examples and the testing dataset contains 30,000 test examples. Each training rows contains a unique ID, 18 attributes and 1 target containing the class that needs to be predicted. The metric used to measure model performance is F1 score. The idea is both to use classification methods presented in the INF8245e course or otherwise, as well as to think about the design of new features that could increase the performance of the model for this classification problem.

1 FEATURE DESIGN

1.1 DATA VISUALIZATION

The very first step of our work was to understand the data we had to work with. We therefore took time to do an exploratory analysis of the data. We plotted the histograms of each numerical feature with the objective of understanding they were distributed. The plots obtained and visible in the appendix (4) show interesting results:

- $PersonalityTrait$ features appear to follow Gaussian distributions. However, we notice that the histogram of the $PersonalityTrait3$ feature is extremely recentered in 0. This highlights the presence of aberrant values that will have to considered during the preprocessing step.
- We also note the presence of extreme values (outliers) for the $ConcertAttendance$ feature. Indeed, some values reach 10^9 and lead to a large difference between the average of 3.10^5 and the median of 3.10^3.

The $Rain$ and $Seated$ features caught our attention because they would seem to have an influence on the enjoyment of a concertgoer. We have desired to confirm or deny using *countplot* (cf. 6). We

notice that people appreciate a concert more when the weather is nice than when it rains, but also when they are seated (although this is much less obvious). These features will therefore be useful for our model.
We also checked that the initial numerical data was not correlated two by two (5). It is important not to have redundant features as this could degrade the performance of the models tested.
Moreover, we noticed that about 800 values were missing in each feature. How to deal with these missing values is an important part of the pre-processing phase described below.

This data visualization step is therefore particularly important for understanding our data and choosing the relevant pre-processing methods to apply to set up our prediction model.

1.2 PRE-PROCESSING

An essential step before implementing learning algorithms is to pre-process the data. Indeed, as we have seen in the data visualization part, some data is missing, aberrant or even erroneous.

So we took care to properly clean the data for each feature. Here are the different processes used to clean the data:

- **Filling of missing data by inference**
  Some data are missing or "$Insert...$" but yet the analysis of the data allowed us to see that we could infer them. For example, knowing $ConcertID$ and some non-missing entries for $ConcertAttendance$, $InsideVenue$, $Rain$, and $BandName$, we can infer $ConcertAttendance$, $InsideVenue$, $Rain$, and $BandName$ by the most represented value (expected to be unique). This inference process was also used to infer $BandDebut$ as well as $BandGenre$ from $BandName$.
  If ever a $ConcertID$ was assigned different values for $BandName$, $ConcertAttendance$, $InsideVenue$ or $Rain$, the most frequent value was used to homogenize.
- **Filling of other missing values**
  The other values that may be missing for features have been replaced either by the average if it is a numeric feature, or by the most frequent value if it is a categorical feature. Note that we could have set up a learning algorithm to determine the missing values, but we chose to focus our resources on other aspects of the project.
- **Creating categorical variables from numeric variables**
  In order to group different entries for a feature, we decided to create categories. In this case for the variable $BandDebut$ which allowed us to create the groups $1950s, 1960s, 1970s...$ as well as for the variable $ConcertGoerAge$ to create the groups $10, 20, 30...$ in order to group music groups and spectators by decade.
- **Detection of outliers and standardization**
  We have noticed some outliers present in our dataset for the numerical features. As for example for the feature $PersonalityTrait2$. We were able manage them by applying a Robust Scaler1 in order to standardize the data.

$$X_{RobustScaled} = \frac{x_i - x_{med}}{x_{75} - x_{25}} \tag{1}$$

- **Encoding**
  Finally we applied one-hot encoding to transform all the categorical features into a set of boolean features each of which represents a possible value. We used this +3000 features representation with a NN.
  At the beginning of our work, we had tried another encoding: we had applied OrdinalEncoding for categorical features and OneHotEncoding for boolean features. It led to a 23 features representation that we used with Random Forest Classifier.

1.3 FEATURE SELECTION

In order to choose the most relevant variables for our prediction, a feature selection method can be used. Feature selection removes irrelevant, redundant, and noisy data; thereby, it can reduce storage, computational costs and avoid a decrease in learning algorithm performance. Besides removing

excessive features, feature selection also aims to improve prediction accuracy and reduce analysis time.
Feature selection works well with linear models.
Nevertheless, it is not the case with neural networks, since neural networks (NN) are capable of computing tons of input and adjusting the weights in order to get the best prediction. NN can adjust the weights in an autonomous way. Furthermore, it is very hard to determine the impact of a single input on the final prediction, because of all the hidden layers and the calculations that go with it. That is why we will not use feature selection for NN since it is irrelevant.

We will use two different ways of pre-processing and apply feature selection on those. Please note that the model accuracy is computed on a validation set.

- **Pre-processing 1: Little number of features (23)**
  With the first pre-processing we have 23 features. Using sklearn model.feature_importances_ on a random forest classifier, we can see which features are more important. The importance of a feature is basically: how much this feature is used in each tree of the forest. Formally, it is computed as the (normalized) total reduction of the criterion brought by that feature. We have sorted the features by importance and we train a random forest classifier with the top 5, 10, 15, 20 features (out of 23) to see if some features are useless regardless to the model accuracy.

# selected features	5	10	15	20	23/23
Model accuracy	0.63248	0.63426	0.664493	0.66923	0.673384

Table 1: Model accuracy depending on the number of features taken (Small amount of features, Random Forest Classifier)

  It seems that some features can be removed. Indeed the model accuracy loss is negligible (0.6733 to 0.66923 when removing 3 features).

- **Pre-processing 2: Large number of features (3270)**

  With the second pre-processing, the one described in section 1.2, we have 3270 features. We have sorted the features by importance and we train a random forest classifier with the top 100, 1000, 1500,.. features (out of 3270) to see if we some features are useless regardless to the model accuracy.

# selected features	100	1000	1500	2000	2500	3000	3270/3270
Model accuracy	0.395882	0.487059	0.497059	0.508824	0.506471	0.516471	0.655294

Table 2: Model accuracy depending on the number of features taken (Large amount of features, Random Forest Classifier)

  Here, we have quite the opposite effect than with the pre-processing 1: removing features has a significant impact on the model accuracy. Therefore we should not remove features with this model.

- **Partial discussions**
  At first sight, we might have thought that feature selection would have worked better on training set with large number of features. Whereas it is not the case, when comparing table 1 and 2. The reason for this, is that there are a lot of one hot encoding with pre-processing 2 therefore removing certain columns can affect significantly the model accuracy. That is why, we preferred not to used feature selection for the random forest classifier, as for the neural network.

1.4 External data

When we have a dataset, it is always better to do literary research in order to multiply the information and increase the expertise we can have on it.

Music genres
Here, we have mainly sought to add information concerning musical genres. We found different resources that allowed us to create new features relevant to the subject and other features already presented. We mainly used the work of students (Gandhi) who used a dataset on music according to genre. Thanks to this dataset, they were able to establish similarities between the different musical styles depending on the year. These similarities were then added for each entry taking into account the date of creation of the band as a temporal reference. With this dataset, we were also able to indicate how widespread the musical style was at the time the group was created.

Bands names
We noticed that different bands had quite similar names. Thus we performed an NLP analysis in order to indicate whether certain tokens appeared in several group names. We have for example the $Crazy$, $Joystick$, $Frogs$ tokens.

Geography
In order to group different musical cultures, we have added in one hot encoding the continent of origin of the music groups.

2 ALGORITHMS

The first machine learning algorithm we chose to use was a Logistic Regression Classifier. We used it as a starting point and as a baseline for the 2 other models that we have retained and will present now.

2.1 RANDOM FOREST

The Random Forest model is a popular and powerful ensemble learning method because it exploits the strong expressiveness of decision trees while reducing variance and therefore limiting overfitting. This model uses the Boostrap Aggregating procedure which makes the decision trees less sensitive to the noise of the training set and also gives each tree a random subset of features. Each sub-tree therefore works with a sub-set of individuals and a sub-set of features.
We chose to work with this model because it gave us a better performance than other ensemble methods as Boosting that we tested. We used this model with both pre processing approaches mentioned above. We will discuss how we tuned our Random Forest classifiers in the 3rd section of the report.

2.2 NEURAL NETWORK

We finally used deep neural networks. The input layer is the size of a dataset record (in the order of 3000) while the output layer is of size 4. Each neuron of the output layer corresponds to a possible prediction among $WorstConcertEver$, $DidNotEnjoy$, $Enjoyed$, $BestConcertEver$. The final prediction chosen is the one associated with the neuron having the best score with the softmax function.
The optimizer used for this neural network is adma (see paper from Diederik P. Kingma). We also tested with a gradient descent optimizer (Stochastic gradient descent) but the latter required much more epochs to converge towards a solution close to optimality unlike adma which converges very quickly.
We will then discuss how we chose the configuration of the neural network and the associated issues.

3 METHODOLOGY

3.1 TRAINING/VALIDATION SPLIT

In order to train our different algorithms, we divided the dataset into 2 parts: the training part and the test part.
The training part could be used if necessary with cross validation in folds in order to tune the hyper-parameters.

The final models could be compared using the local test dataset but also sometimes with that associated with Kaggle (taking into account the limit of 2 sent per day).

3.2 Setting hyper-parameters

3.2.1 Random Forest

To perform an hyper-parameter tuning for our Random Forest classifier, we considered the following hyper-parameter : the number of decision trees in the forest, the number of features considered by each tree and the maximum depth of each tree.
We used the GridSeachCV class of Scikit-Learn library in order to try specified values and pick those that give the best F1-score. Hyper-parameter values are optimized through a cross-validation process, resulting in a model that is unbiased by the training data. We took care to use a Stratified KFold cross-validation as the classes to be predicted are not in equal proportions in the training set (40%, 40%, 10%, 10%). This process ensures that each fold of dataset has the same proportion of observations with a given label.
We noticed that the hyper-parameter which had the most influence on performance was the maximum tree depth.

3.2.2 Neural Network

The configuration possibilities of neural network are endless. We still try to try a few that we can put into categories.

We distinguished a few sets of parameters that could be modified in our neural networks.

1. The shape of the network:
   Indeed there are different shapes of networks (1) concerning the configuration of the hidden layers, we can notably distinguish a constant form (A), increasing pyramidal (B), decreasing pyramidal (C), hourglass (D) or inverted hourglass (E).

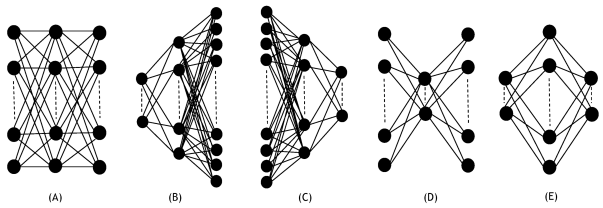

Figure 1: Neural Network shapes

2. The number of epochs and the size of the batch size:
   In order to determine the number of epochs, we can trace the evolution of the training and validation score according to the epoch, we do the same for the errors (2). We then take the epoch with the lowest validation error as a model.
3. The number of neurons per layer:
   Having too many neurons could lead the NN to overfitting the training dataset while having too few neurons could lead to underfitting. We decided to put a rather large number of neurons and to add layers of dropout in order to deactivate certain neurons. The layers of neurons are activated with a ReLU function.

4 Results

To compare the different models, we are interested in the score on the test set which was not used for the selection of parameters and hyperparameters. Finally we also look at the confusion matrix put in % by predicted value.
This makes it possible to know for each target value, how many % have been correctly predicted and also the percentage of bad prediction on the other labels (like the example on the fig 3).

Figure 2: Accuracy and Loss as a function of the epoch

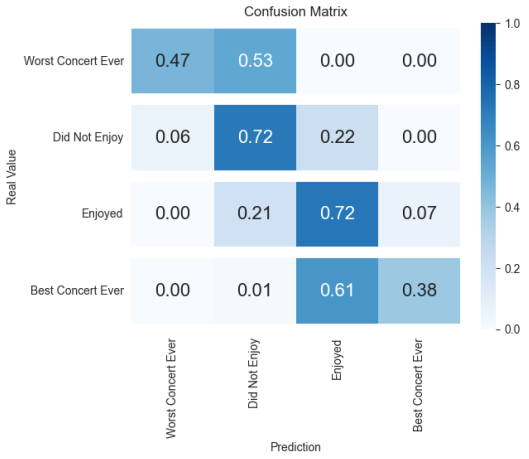

Figure 3: Confusion matrix in %

We started with logistic regression classifier, that we considered as a baseline later on. For random forest, we had similar results for different types of pre-processing whereas with neural network, it worked better with a large amount of features. As we can see in the table below, the strongest model is the neural network with the pre-processing with 3270 features. This model was used for the final prediction.

Model	Pre-processing	Kaggle Submission score
Logistic Regression	23 features	0.42620
Random Forest	3270 features	0.63619
Random Forest	23 features	0.65885
Neural Network	3270 features	0.68714

Table 3: Kaggle submissions score for different models

5 Discussion

During the addition of external data, we wanted to put data concerning the music groups. We tried to find information on the music groups such as their number of followers on the platforms,

number of hours of listening... We could not find information on the music groups while some had performed concerts with more than 10,000 spectators. We then noticed strange similarities between the names of music groups and the names of real music groups ($RunningattheDisco$, $Crazyplay$, $JoystickAttack\ldots$). We decided not to exploit the music groups more for fear of allocating too much time on a task that would have no impact on our prediction.

6 Statement of Contributions

At first, we all began on our owns with conventional pre-processing and classifiers. Yoann and I (César) worked with ensemble classifier such as random forest while Stéphane tried deep learning with a different way of pre-processing the data. Afterwards, we chose the method with the best score and used it for the final prediction. We then divided in three the writing of the report.
We hereby state that all the work presented in this report is that of the authors.

References

Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization. https://arxiv.org/pdf/1412.6980.pdf, 2017.

Anurag Gandhi. Understanding origins and fusion of music genres. https://shouvikmani.github.io/Million-Song-Dataset-Visualization/index.html, 2017.

APPENDIX

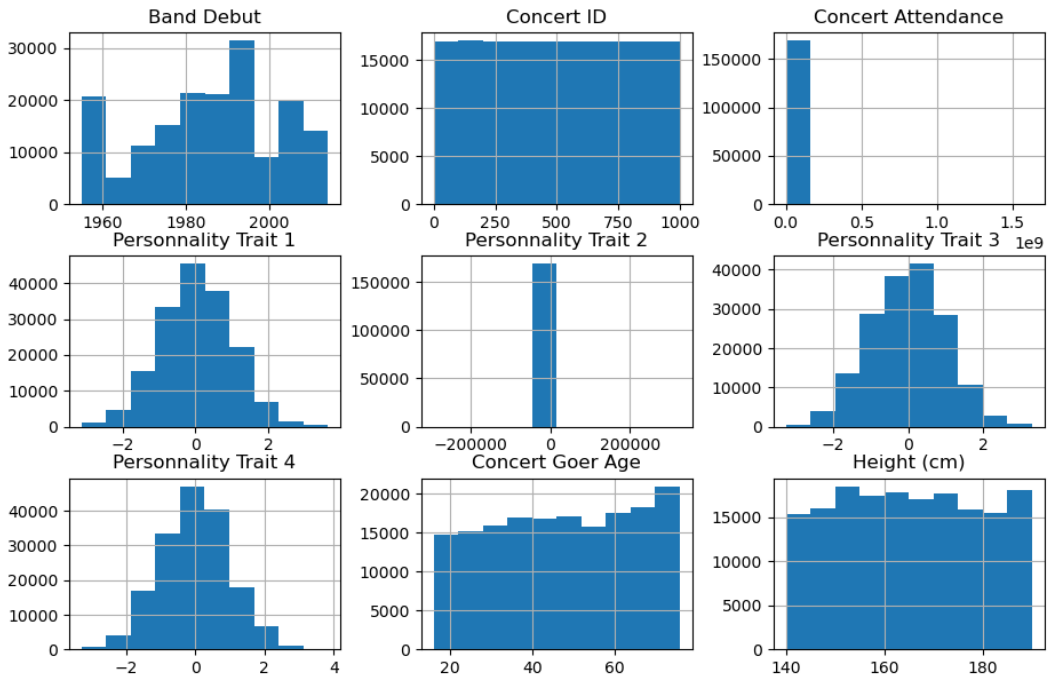

Figure 4: Histograms of numerical features

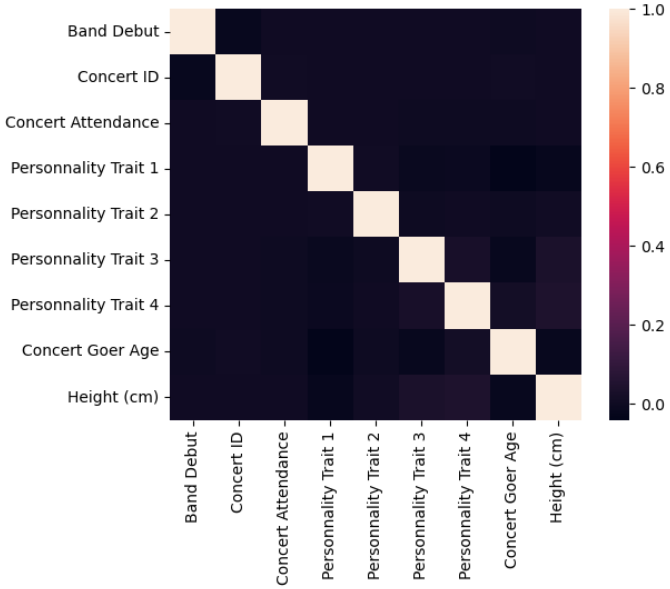

Figure 5: Correlation heatmap between original numerical features

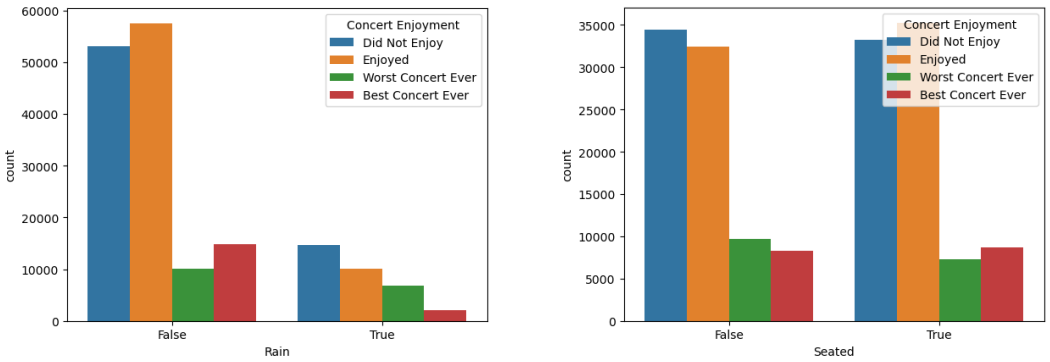

Figure 6: Rain & Seated influence on Concert Enjoyment