CLASSIFICATION ON CONCERT GOER'S ENJOYMENT INF8245E MACHINE LEARNING FINAL REPORT

Team **The Overfitters**
Members Zhanwen XIN (zhanwen.xin@polymtl.ca, 2126463)
Benjamin ÅSTRAND (benjamin.astrand@polymtl.ca, 2231111)
Ka Hei CHU (ka-hei.chu@polymtl.ca, 2163284)

ABSTRACT

In this report, we show how we performed different classification models on a classification problem with different data preprocessing methods. Choice of classification model is specified in depth, especially on the reasoning for models' performance, followed by a summary of model's accuracy at the end.

1 PROBLEM BACKGROUND

This project focuses on a classification problem on concert experience based on an investigation of band information, concert venue and facilities and concert goer information. The concert experience is categorized into "Worst Concert Ever", "Did Not Enjoy", "Enjoyed", "Best Concert Ever". Two set of dataframes is included in this dataset, a training set of 170000 by 19 dataframe and a test set of 30000 by 18 dataframe without labels.

2 MODEL ARCHITECTURE

Figure 1 demonstrates the structure of the classification model from training set to output. Methods we have used are listed under each frame box. The best classification methods are XGBoost and Neural Network and will be discussed in details in the the corresponding section.

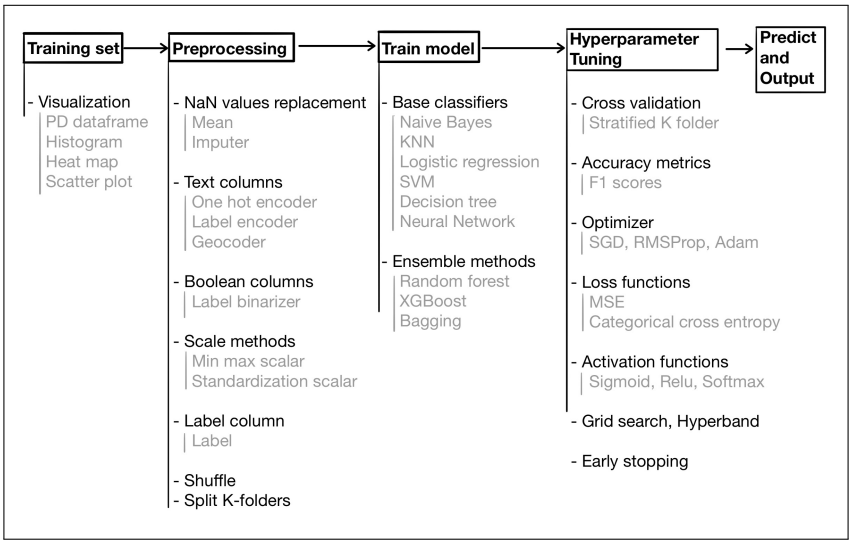
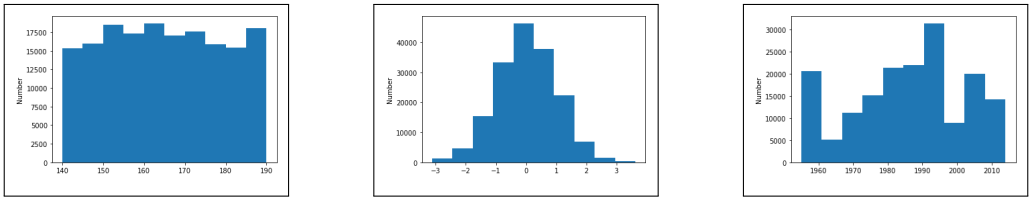


Figure 1: Model architecture

3 PREPROCESSING OF THE DATA

The original dataset comprises 170,000 rows (examples) and 18 columns (features). Each of the columns provides data about the band, the concert-goer, or about the concert/environment. The dataset includes numerical features, binary features, and categorical features, such as 'Concert Goer Age', 'Inside Venue', 'Band Name' respectively.

The distribution of the training data for each feature was studied to better determine suitable preprocessing methods. Some features seemed to have a uniform distribution (Figure 2a), others seemed to have a Gaussian distribution (Figure 2b) and for some features, the ground truth distribution was more difficult to distinguish (Figure 2c). A strong imbalance in the distribution of data was also observed in several features, such as concert attendance.



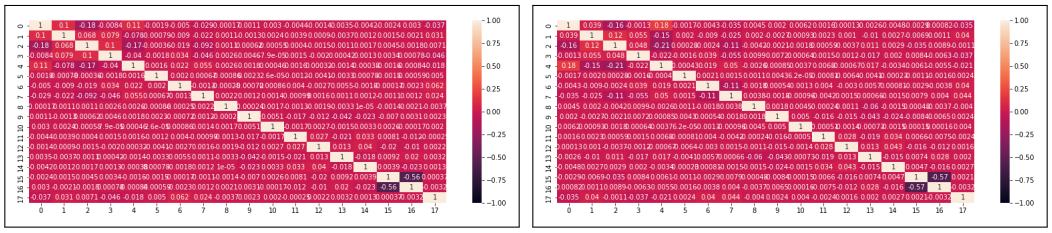
(a) Concert Goer Height (cm) (b) Personality Trait 1 (c) Band Debut

Figure 2: Typical data distributions (a) uniform, (b) Gaussian, (c) unclear

3.1 FEATURE DESIGN

All features were used except for 'Id', which is the identifier field of a concert review example. The value of this feature is assumed to have no real-world meaning. Even if 'Id' were sorted, for example in chronological order, this information is disregarded. This feature also has no duplicate values in the training set, nor in the 30% of the test that was given. It is therefore assumed not to appear in the rest of the test set and can be removed as a feature.

The covariance matrix was studied to determine if there is a linear correlation between any pair of features (Figure 3a). The strongest covariance found, the one with the largest absolute value, was -0.56, which was considered too low to conclude that the two features are linearly dependent. Resampling was used to eliminate the risk of low correlation caused by an imbalance in the data (Figure 3b). The attempt did not have a significant impact on the covariance matrix. The strongest covariance increased to -0.57, resulting in the same conclusion as before the resampling.



(a) Before resampling (b) After resampling

Figure 3: Heatmap of covariance between all features, except 'Band Genre', before and after resampling, the highest covariance (0.57) shows the correlation between longitude and latitude coordinnate of concert goer country of origin

3.2 PREPROCESS METHOD

As earlier mentioned, the dataset includes features with different data types, which can be preprocessed in different ways. Initially, a label encoder was used on binary features to convert the values 'True', 'False' to 1, 0 respectively, and a one-hot encoder was used on categorical features. As different models were evaluated, so were the preprocessing methods. In the neural network, it was found

that the accuracy from the one-hot encoded data and label encoded data was similar and at around 0.66. Theoretically, the artificial distance created by label encoder could have a negative effect on the model's performance. But for our dataset, results have shown that this distance is negligible. Label encoder allowed for a large reduction in the number of features compared to the one-hot encoder, which decreases the complexity of the model and can, in theory, prevent overfitting. As a result, the label encoder was used for all its assets and no accuracy loss.

Two of the features, 'Band Country of Origin' and 'Concert Goer Country of Origin', are geographical locations stored as strings. These strings were passed to an API (Geocoder) and mapped to their corresponding coordinates (latitude, longitude), with the goal that the model should learn based on the distances between different locations. An algorithm was developed to first save unique locations (152 unique locations) and their coordinates in a Python dictionary, then perform the look-up in the dictionary instead of calling the API for each of the 170,000 examples. This preprocess method not only took advantage of the true distance between countries, but also reduced the number of features significantly from 154 to 4 compared to the data preprocessed by one hot encoder, which in its turn reduced total runtime drastically from 2.5 hours to 24 minutes.

To handle missing values in the dataset, different methods were evaluated. For the label and one-hot encoder, NaN values were replaced by an extra category. For the other numerical columns, a univariate imputer that replace missing values mean and a multivariate data imputer that can learn the missing value from the existing data were tested. Theoretically, multivariate data imputer can find a better representation hence lead to higher accuracy. But results shown that these two methods made no big difference.

The column 'Concert Goer Id' was processed by slicing the string value and keeping the numerical part. This could potentially improve performance if the same concert goer appears in several examples, which is a common occurrence considering that there are only 2,000 unique concert goers in the training set.

For feature scaling, both standardization and normalization were attempted for different models. In a neural network, normalization is useful since the algorithm does not make assumptions about the distribution of the data (Lakshmanan, 2019). It is verified by the fact that our NN model did not learn with original dataset, but works well with standardized or normalized data.

The preprocessed data was shuffle before passing to model to avoid any unwanted patterns in the dataset.

4 MODELING CHOICE

In this assignment, 6 base classifiers and 2 ensemble methods were employed. The 6 base classifiers are Naïve Bayes, K-Nearest Neighbor (KNN), Decision Tree, Logistic Regression, Linear Support Vector Machine (SVM) and Neural Network. Two ensemble methods, bagging and boosting, were applied on top of Decision Tree (Random Forest and Extreme Gradient Boosting (XGBoost)) and Neural Network.

4.1 BASE CLASSIFIER

Classifiers such as Naïve Bayes, Logistic Regression, KNN are considered as weaker learners compared to the others for our dataset because they either rely on strong assumptions (conditionally independent features given class label in Naïve Bayes, linearly separable features in Logistic Regression), or high computational complexity (KNN).

As the state of the art, Linear SVM is considered as a weak learner for our dataset with an accuracy of 0.38. Theoretically, SVM is superior than logistic regression in term of handling features that are not linearly separable and it guaranteed that SVM can find the separating hyperplane for classes if it exists (Chandar, 2021). It does not work in our case because:

- Some features in the dataset is extremely unbalanced (Section 3). SVM is very sensitive to the balance of the data since it has a direct effect on the ratio between the positive and negative support vectors (Kumar, 2022).

- With a high dimensional dataset, it is very difficult to train SVM, the SVM model we have is not trained enough to find the optimal separating hyperplane.

Decision Tree classifier is the second best base learner with an accuracy of 0.6. Decision tree is prone to overfitting as it goes deeper. We firstly overfit our decision tree to the data then apply reduced error pruning to mitigate the overfitting issue. Notably, Decision Tree is ideal for a documentation classification like our problem because:

- It allows interpretation, i.e. when a feature value is bigger than a threshold, the model classifies it to a specific class.
- It can handle all types of attributes easily.
- It is easy to construct in terms of implementation and hyperparameter tuning, and easy to train with low computational complexity.

Neural Network classifier is the best classifier among all the base classifiers with an accuracy of 0.648. It is a complex yet flexible model which allows us to approximate almost any function. It is powerful in terms of:

- black box setting without need in basis function selection;
- regularization that is possible to applied to each hidden layer to prevent overfitting.

The limitations of our Neural Network model are:

- It consumed enormous computer resources, hence it is not guaranteed that our NN structure and hyperparameters are the optimal set.
- It showed a certain degree of high variance. To tackle this problem, we applied bagging on top of 10 NN models to mitigate the variance issue.

4.2 Ensemble methods

The ensemble methods we chose are bagging and boosting. Combined with different base classifiers, we obtained three classification models. They are Random Forest (bagging of decision trees), Bagging NN (bagging of neural network) and XGBoost (extreme gradient boosted decision tree).

4.2.1 Random Forest

Random Forest has an accuracy of 0.656. Compared to a single decision tree (0.60), the accuracy is increased by 9.3%. Usually, bagging on a base classifier brings 1 or 2 percent of accuracy improvement by reducing the the variance, which is observed in the bagging of neural network. The relatively big accuracy improvement comes from the fact that a single decision tree has high variance, by combining multiple decision tree in random forest, this high variance is effectively reduced. The limitations of our random forest model are:

- The existence of lots of irreducible noise in the dataset makes the classification model prone to overfit or biased. The nature that Random Forest algorithm uses fully grown decision trees prevent it from a more general and accurate model.
- Random Forest cannot handle extrapolation data (Mwiti, 2022). It predicts well in the range of data where it is trained. But for the unseen data, it is hard to give an accurate prediction.

4.2.2 XGBoost

Among all the classification models, XGBoost has the best accuracy to the authors knowledge. It is 1% higher than random forest and 0.4% higher than bagging Neural Network. Different from Random Forest, XGBoost combines multiple weaker models, which are generated in a process formalized as a gradient descent algorithm, to create a collectively stronger model (Mitchell, 2022). These weaker models are less deep decision trees compared to the fully grown decision tress in random forest. These trees have high bias but low variance. After gradient boosting, the bias term is

reduced. This is the reason that for the dataset with much noise, XGBoost with lower bias error can have better performance than Random Forest.

The limitations of XGBoost are:

- Its accuracy drops with sparse and unstructured data. This is verified when we used only one hot encoder to preprocess our categorical features. XGBoost was hard to learn a better model and to achieve a high accuracy with the one hot encoder dataset.
- Gradient boosting method is very sensitive to outliers since every classifier follows the path of fixing the errors in the predecessor learners. Since our dataset has a lot of noises and outliers, this limitation is believed to be a main factor that prevents us from a higher accuracy.

4.2.3 BAGGING OF NEURAL NETWORK

Compared to the accuracy of a single NN (0.648), by applying Bagging, it is increased to 0.662 with 10 NN models.Bagging is applied on Neural Network because:

- it works well with "reasonable" classifiers. In our problem, NN is one of the best base classifier we have;
- there is a variance in our single NN model and bagging can reduce the variance error and push one last step for the accuracy of NN model.

5 HYPERPARAMETER TUNING SCHEME

This section focuses on the hyperparameter tuning for the top 2 models: XGBoost and Neural Network. For each hyperparameter, why we choose it, how to tune it, and how it improve our model is explained.

5.1 XGBOOST

Stratified 3-folder cross validation was employed when tuning hyperparameters for XGBoost classifier. The parameters that we tuned, and why we tune it are listed below.
max_depth: If maximum depth is too high, the model will overfit to the noise in our data, while a too small maximum depth will lead to a bias tree that has underfitting issue. Choice of a maximum depth has big impact on the model accuracy.
When tuning the maximum depth, a big value of 100 was used as the first try to let the model overfit to our data. Then by decreasing it to get the optimal value of 15.
n_estimator: When number of estimator is 1, it means only one tree is used. With a too big number of estimator will increase the parameters to search and more computational time, while brings little difference. The optimal number of estimator in this problem is 100 out of 50, 100, 150.
min_child_weight: The optimal value for min_child weight is 5 out of 1, 5, 15, 200.
reg_lambda: A too high L2 regularization coefficient will have strong restriction on the weight and lead to an underfitting model with low variance but high bias. On the opposite, when the lambda value is too small, model could overfitting to the noise. The optimal value for it is 4.

5.2 NEURAL NETWORK

In the hyperparameter tuning of Neural Network model, we used categorical cross entropy as the loss function, and patience of 5 and 10 for early stopping. The parameters that we tuned and the reasons we tuned them are listed below.
number of unit: A too small number of hidden unit will lead to a too simple model that cannot capture the complexity of problem, and a too high number of unit will increase the computational time and overfit to the dataset. The optimal unit we found by hyperband search for a one layer neural network is 275 from 20 to 300.
number of hidden layers: Number of hidden layer decides the structure of NN and it is directly link to the complexity of NN model. We explored 1, 2 and 3-hidden-layer NN and found that a 3-hidden-layer neural network was capable to fit well on our dataset and provided the best accuracy.

Table 1: Model and accuracy conclusion

Classifier	Accuracy	Hyperparameter
Bernoulli Naive Bayes	0.40	/
Logistic regression	0.40	C=0.001, max_iter=15
KNN	0.59	n_neighbors=10
Linear SVM	0.38	C=0.1, max_iter=150, kernel=rbf
Decision trees	0.60	max_depth=15
Neural Network	0.648	L1 units: 100, L2 units: 20, number of epoch: 200
Random forest	0.656	n_estimators=50
XGBoost	0.666	n_estimator=50, max_depth=15, reg_lambda=16
Bagging Neural Network	0.662	number of model = 10

number of epoch: A small number of epoch does not allow the model to be trained enough, while a too big number of epoch will lead to overfitting problem. Epoch number between 100 and 150 is optimal for our case.
activation function: It is the key that allows NN to capture nonlinear problem. Relu and sigmoid function were tested on the hidden layers and Relu provided higher accuracy. On the output layer, softmax function was used because the problem has multiple categories.
loss function: Two loss functions defined by Mean Square Error (MSE) and Categorical Cross Entropy were tested. The latter provides a better performance because our prediction label is one-hot encoded.
optimizer: It is tuned with different methods, such as Stochastic Gradient Descent (SGD), Root Mean Square Prop (RMSProp), and Adaptive Moment Estimation (Adam). The previous two optimizers are tested before Adam. Adam has the highest accuracy because it updates the learning rate based on the running average of gradient and gradient square.

6 Results

Table 1 summarizes the F1 accuracy of each classifier, as well as the hyperparameter to achieve that accuracy. The ranking of the model performance is XGBoost $>$ Bagging Neural Network $>$ Random forest $>$ Neural Network $>$ Decision trees $>$ KNN $\gg$ Bernoulli Naive Bayes $\approx$ Logistic regression $\approx$ Linear SVM. This ranking reflects only a level of performance for each classifier because it is not guaranteed that we obtained the most optimal hyperparameter for each of them.

The discussion of how each classifier works, why it works or why it does not work is in Section 4. How to tune the hyperparameter is in Section 5. The potential improvement to the model performance is presented in Section 7.

7 Possible improvement

Preprocessing of country data could be further improved. Latitude and longitude can be viewed as the two angles that, together with the radius, make up the three coordinates in a spherical coordinate system. By assuming that the Earth is a perfect sphere (i.e. constant radius), the latitude and longitude can easily be transformed into a Cartesian coordinate system. This representation would better capture the actual distance between different locations and potentially improve performance.

8 Conclusions

This report focuses on the data preprocessing, modeling choice and hyper parameter tuning. The two best classification models are XGBoost and Bagging Neural Network. The report presents our thinking on why these two models work the best, what are their limitations and why the other models have poor performance based on our dataset, the preprocess methods, hyperparameter tuning, etc. Potential improvement to our code is proposed at the end.

AUTHOR CONTRIBUTIONS

Zhanwen Xin: Conceptualization, Building running environment, Data preprocessing (one-hot encoder, label encoder, replacing NaN with mean value, normalization scalar, etc), Classification models (all models except KNN), Hyperparameter tuning for all classifiers (NN alongside the other authors), Writing- Review & Editing.
Benjamin Åstrand: Focused on data preprocessing. Conceptualized and coded the functions: identifies unique countries, converting country names to coordinates and stores them in a dictionary (Section 3.2), data imputation and multi-variate imputer. Performed hyperparameter tuning of the neural network alongside the other authors. Wrote section 3.
Ka Hei Chu: Conceptualized and coded functions: resampling, heatmap, mapping countries to coordinates with the generated dictionary, KNN-model and Gradient Boost model. Performed cross-validation and hyperparameters tunning alongside the other authors. Wrote section 4.

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

REFERENCES

Sarath Chandar. *INF8245E Machine Learning Course Notes Chapter 8*. September 2021.

Dhairya Kumar. Demystifying support vector machines. https://towardsdatascience.com/demystifying-support-vector-machines-8453b39f7368, 2022.

Swetha Lakshmanan. How, when, and why should you normalize / standardize / rescale your data? https://pub.towardsai.net/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff, May 2019.

Rory Mitchell. Gradient boosting, decision trees and xgboost with cuda. https://developer.nvidia.com/blog/gradient-boosting-decision-trees-xgboost-cuda/, 2022.

Derrick Mwiti. Random forest regression: When does it fail and why? https://neptune.ai/blog/random-forest-regression-when-does-it-fail-and-why, 2022.