Statistical Methods for Machine Learning

Francesco Torgano 16028A

A.Y. 2022-2023

Ridge Regression

0 Project guidelines

Download the Spotify Tracks Dataset and perform ridge regression to predict the tracks' popularity. Note that this dataset contains both numerical and categorical features. The student is thus required to follow these guidelines:

1. Train the model using only the numerical features
2. Appropriately handle the categorical features (for example, with one-hot encoding or other techniques) and use them together with the numerical ones to train the model
3. Use 5-fold cross validation to compute your risk estimates
4. Thoroughly discuss and compare the performance of the model

The student is required to implement from scratch (without using libraries, such as Scikit-learn) the code for the ridge regression, while it is not mandatory to do so for the implementation of the 5-fold cross-validation

0.1 Disclaimer

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offenses in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1 Introduction

The goal of this project is to provide an implementation from scratch of the Ridge Regressor, a linear predictor. This regressor uses a parameter α that controls the bias of the algorithm and that will need to be tuned to get the best performance.

This regressor will be applied to a dataset containing song data from Spotify along with a popularity value, the goal is predicting the value of this popularity attribute given the other information about the song.

The dataset contains both numerical and categorical data that will need to be handled properly to use it to train a predictor. To choose a proper encoder, there will be tests done to check which encoding gives more information.

The predictor will be trained first using only numerical data and then using also the categorical data properly encoded.

In each of those cases, the best value for the parameter α will be searched using 5-fold cross validation.

2 Dataset

2.1 Structure

The dataset contains information about tracks from Spotify, it contains 114000 entries with 20 attributes each.

2.2 Preprocessing

2.2.1 The track_genre problem

This dataset is balanced around the number of genres, for each genre there are 1000 songs.

The main problem is that some songs have multiple genres and whenever a song has more than one genre, its entry is simply replicated with only the track_genre attribute different. Using the track_id as guide we can see that there are 89741 actual songs in this dataset instead of 114000.

We can notice that the popularity of the song is tied with the genre, a song has a popularity value different for each different genre associated with it, that's why there are multiple entries for each song.

This could cause some data leakage, since a song could end up both in the train and test set, just with a different value for a single attribute.

This problem needs to be tackled in 2 different cases, if we are considering only numerical values and if we are considering all attributes.

In the first case, since we are not considering the track_genre attribute, there would be duplicated entries all with the same value and possibly different popularity value. To solve this, we will get rid of all the duplicates and use as popularity value the mean of the values for that track.

Table 1: Dataset attributes

Attribute	Type	Description
track_id	Categorical	Spotify ID for the track
artists	Categorical	Artists that worked on the song, separated by ;
album_name	Categorical	Album name that the song is part of
track_name	Categorical	Name of the song
duration_ms	Numerical	Track length in ms
explicit	Boolean	Indicates whether the track has explicit lyrics
danceability	Numerical	Indicates how danceable the track is
energy	Numerical	Indicates how intense the track is
key	Numerical	Key of the track
loudness	Numerical	Indicated track loudness in dB
mode	Boolean	Indicates major(1) or minor(0) modality of the track
speechiness	Numerical	Indicates how much spoken words are in the track
acousticness	Numerical	Confidence measure of being an acoustic track
instrumentalness	Numerical	Confidence measure of being an instrumental track
liveness	Numerical	Detects presence of audience in the recording
valence	Numerical	Indicates "positiveness" of the track
tempo	Numerical	Estimated tempo of the track in BPM
time_signature	Categorical(encoded)	Estimated time signature
track_genre	Categorical	Genre of the song
popularity	Numerical	Indicates the song popularity

In the second case, since we have the track_genre attribute, we will leave the "duplicates" as is since there is a link between track_genre and popularity that can be detected using all attributes.

The track_id attribute will be ignored from now on, since it doesn't provide actual info about the track.

2.2.2 The track_genre problem alternative solution

Another option to handle the track_genre problem could have been treating it like the artists attribute and group all genres of a song in a single attribute and averaging their popularity, doing so would allow having a single record for each song but would also ruin the balance of genres in the dataset and lose some information regarding the connection between popularity and track_genre.

The results of the experiment can be found in chapter 4.2.2

2.3 Encoding

Encoding categorical values back to numerical values is a necessary step to use Ridge Regression.

The explicit attribute is boolean and will be considered and converted to numerical simply by substituting 'True' with 1 and False with 0, similar to how

the mode attribute is already encoded in the dataset, so it won't undergo a proper encoding process like the 'string' attributes.

In this case, we have 4 main categorical attributes to take care of:

Table 2: Categorical Attributes

Attribute	#Unique Values
artists	31437
album_name	46589
track_name	73608
track_genre	114

There are various ways to handle the encoding of categorical attributes:

- Standard encodings, use only information from the features being encoded
  - One-Hot Encoding: create an array that for each unique value has an attribute set to 1 and all others to 0.
  - Ordinal Encoding: assign to each unique value an integer, introduces an ordering that might not be present in the attribute itself.
  - Binary Encoding: similar to One-Hot Encoding, but each unique value is associated to a binary number instead of a single attribute, creating less new columns but introducing some ordering information.
- Supervised encodings, also use information about the target variable to give a numerical value to the features being encoded
  - Target Encoding: group entries for each unique value of the attribute, calculate the average of the target attribute (y, popularity) and encode the value with the average of the target value [1].
  - CatBoost Encoding: similar to target encoding but uses 'ordered boosting' to prevent target leakage [2].

One-Hot encoding technique can be discarded from the start since it would change the number of features in our dataset from 20 to more than 130 thousands, which would make training the linear predictor much harder because of the much bigger memory requirements.

All the other options are feasible and will be compared in chapter 4.2.1 to find which provides the best results.

2.4 Scaling

All the features, both the numerical and the categorical encoded, will be subject to a standardization:

$$z = \frac{x - \mu}{\sigma}$$

This should help apply the regularization parameter α more fairly to each feature.

3 Ridge Regression

3.1 Theory

Ridge regression is a more stable version of the linear regression, this is achieved by including a stabilizer (α) in the model.

Using α, we can play with the tradeoff between train and test error.

Linear regression is defined as:

$$w_S = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} ||Sw - y||^2$$

Meanwhile, Ridge regression can be defined as:

$$w_{S,\alpha} = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} ||Sw - y||^2 + \alpha||w||^2$$

where d is the number of features and S is the matrix containing the data we want to learn.

The solution can be found in closed form:

$$w = w_{S,\alpha} = (\alpha I + S^T S)^{-1} S^T y$$

Varying the parameter α between 0 (equal to linear regression) and $+\infty$ we can control the regularization, increasing alpha will improve the test scores while making the train scores worse.

3.2 Implementation

The implementation of Ridge regression is done in Python using only the NumPy library to handle matrix/vector operations and the Pandas library to handle the data, since the function takes DataFrame(s) as input for the train/test data.

The Regressor is implemented in the file *ridgeReg.py* in the class RidgeReg.

This class has various methods:

- constructor(alpha): takes the parameter α as input, this parameter will be used as the regularization parameter
- fit(x,y): takes the training set, divided in x and y and calculates the vector w by using the closed form solution for it, this solution was chosen because it's faster and easier to implement than gradient descent
- predict(x): takes as input a dataframe containing the record for which to predict the target output, outputs an array of predictions, one for each record
- r2_score(y_true,y_pred): takes as input 2 arrays containing the true target values and the predicted target values and calculates the coefficient of determination R^2, a measure often used to check how good a predictor is

- mse(y_true,y_pred): same as r2_score but calculates MSE (mean squared error) instead

The main part of the code is the fit function, the code for it is shown in image and implements the closed form to solve the problem.

Figure 1: Code of the fit method of RidgeReg Class

```
# x,y: Dataframe
def fit(self,x,y):
    x_c = x.copy()
    # add feature with all 1
    x_c['1'] = 1

    transposed = x_c.transpose() # S^T
    sts = np.dot(transposed,x_c) # S^T.S
    alpha_I = self.alpha*np.identity(len(sts)) # AI
    alpha_I_sts = alpha_I+sts # AI+S^T.S
    inverted = np.linalg.inv(alpha_I_sts) # (AI+S^T.S)^-1
    w = inverted.dot(transposed).dot(y) # ((AI+S^T.S)^-1).S^T.y
    # save results
    self.w0 = w[len(w)-1]
    self.w = w[:len(w)-1]
```

4 Experiments

4.1 Prediction with numerical features

The experiment will use 5-repeated 5-fold to calculate an average of R^2 and MSE at train and test scenarios for various value of α to look for the best value of that parameter.

The experiment is run from *ridgeNumerical.ipynb* and the plots are also created from there.

Operations:

1. Select a value for α
2. Separate dataset in train and test partition according to the 5-Fold CV scheme
3. Standardize all features
4. Train Ridge Regression with α equal to the chosen value using train set

5. Test model using test set
6. Calculate R^2 and MSE for both training and test sets
7. Go back to 1 until all folds are used
8. Average R^2 and MSE results and compare
9. Repeat from 2 for 5 times, shuffling the dataset each time
10. Repeat from 1 until all α options are done

To find the proper alpha value we initially run the algorithm with general values (0,2,4,8,10,100,1000) and then fine tune the search.

Table 3: Results of search for optimal α value

Alpha	Test R^2
0	0.03356934653313325
2	0.033569361654920686
4	0.033569372601158005
8	0.033569381969885057
10	0.03356938039382448
100	0.03356500175557505
1000	0.033070678950767006

In the table 3 is reported only the R^2 results since even just these values can show that the difference is marginal compared to the linear predictor (α=0) that we can use as a baseline, the biggest change is 0.0000000354, which is pretty negligible.

Changing the α value doesn't significantly improve the predictor, its value for the coefficient of determination really close to 0 shows that this predictor doesn't explain the variance present in the independent variable and is akin to predicting each time the average popularity of the whole dataset.

For the sake of completion, in image 2, we can see that a value of α close to 8 seems to be the best option. By looking at this graph we can also see that α causes a tradeoff between train and test error, increasing α causes the training error to increase and the test error to decrease, at least initially.

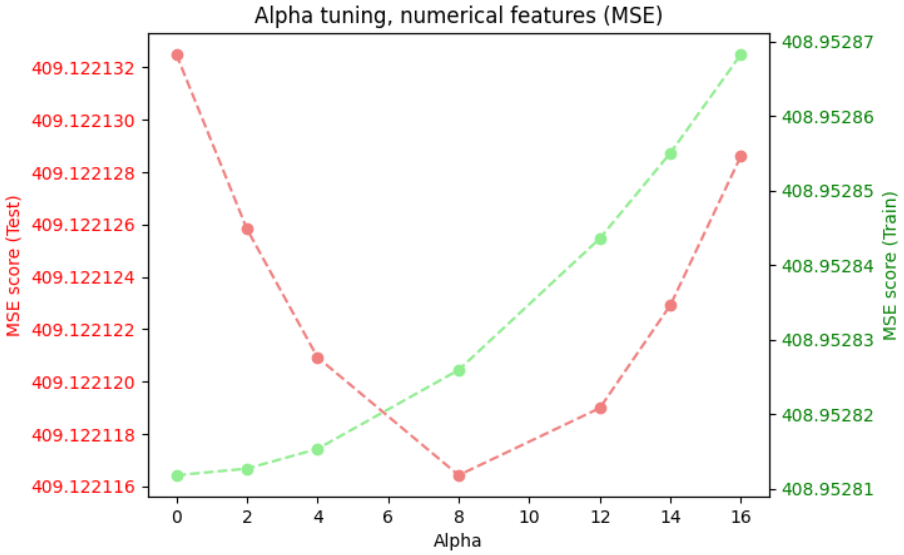
4.2 Encoding comparison

4.2.1 Comparison between encodings

We need to choose a way to encode the categorical features of the dataset, to properly choose we will compare them against each other using a Ridge Regressor with $\alpha = 0$ to mimic a linear regressor and pick the one with the best 5-CV performance. The random_state options are all set to 1 to allow reproducibility.

Method:

Figure 2: Results of Ridge Regression trained with the numerical dataset with various alpha values



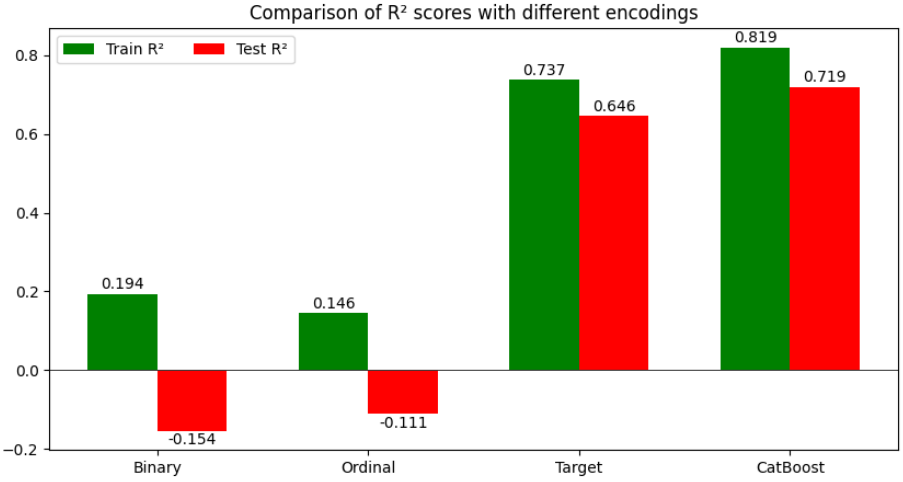
1. Separate dataset in train and test partition according to the 5-Fold CV scheme
2. Fit the encoder using the train data
3. Transform the categorical features of both train and test partitions
4. Standardize all features
5. Train Ridge Regression with $\alpha = 0$ using train set
6. Test model using test set
7. Calculate R^2 and MSE for both training and test sets
8. Go back to 1 until all folds are used
9. Average R^2 and MSE results and compare

All the code to do this procedure is available in the *Encoding.ipynb* file that's attached with the report.

The result of these tests are reported in image 3 and 4.

According to the results, there isn't a clear winner between the standard encoders, since both get negative results in the R^2 test value and high results in the MSE test and train values.

Figure 3: Results of Ridge Regression comparison between encodings (R^2)



Meanwhile, for the supervised encoders, the CatBoost is the better option since it achieves better results both considering R^2 and MSE.

The CatBoost would have been a better option than Target Encoding since the latter needs to be tuned to prevent overfitting caused by data leakage by using a weighting factor (smoothing), especially in the case of classes with unbalanced values while CatBoost should be safer from data leakage since it evades target leakage and prediction shifting thanks to its ordering principle [2].

From these results we can see that Ridge Regression is a way too simple model to properly learn the data of this dataset using the standard encoders, but by employing the help of more advanced encoders we can get acceptable predictions.

4.2.2 Alternative dataset preprocessing - encoding comparison

In this section we repeat the same analysis of the encodings but starting from a dataset that was preprocessed more thoroughly in the ways described in section 2.2.2. It was speculated that this would lead to some information loss and from the experiments, this information loss can be noticed in the *Encoding_alternative.ipynb* and the results reported in table 4.

In this table, we can see that the test results are worse when compared to the results of the previous comparison, meaning that some information was lost by using this alternative processing.

This loss is more noticeable when using the supervised encodings because these techniques work by grouping values for each category and finding the average of the target value but, by increasing the number of unique value for each genre, returns a less reliable estimate for the track_genre encoding.

Figure 4: Results of Ridge Regression comparison between encodings (MSE)

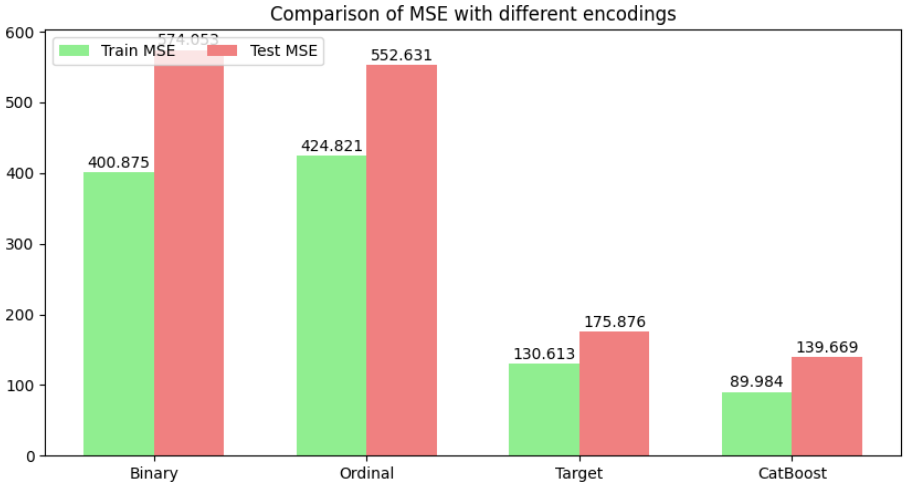


Table 4: Categorical Attributes

Encoding	Train R^2	Test R^2	Train MSE	Test MSE
Binary	0.204	-0.261	336.848	533.606
Ordinal	0.128	-0.224	369.133	518.310
Target	0.781	0.504	92.751	210.037
CatBoost	0.736	0.643	111.969	150.982

4.3 Prediction with all features

This experiment follows the same methodology of the experiment about the search for optimum α using only the numerical part of the dataset, all the related code is available in *ridge_numerical.ipynb*.

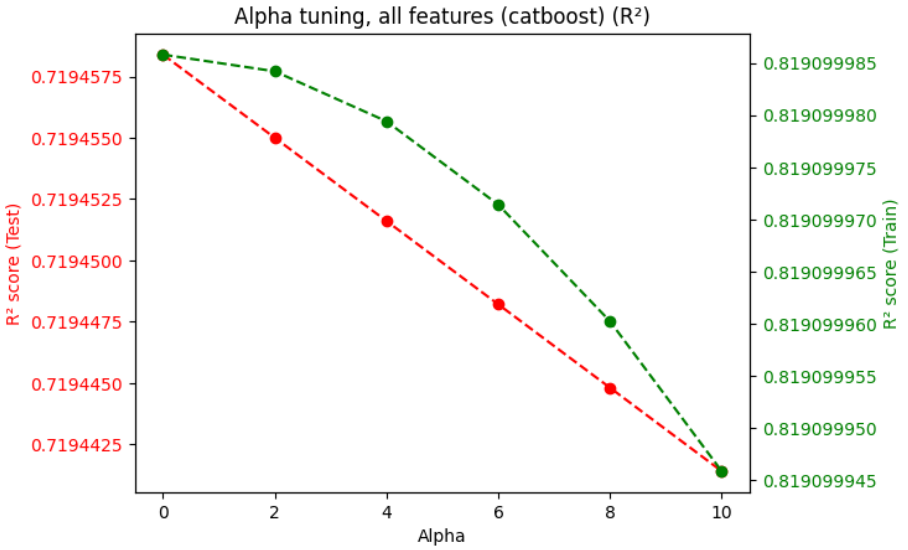
This experiment tried to find the optimal value for the α parameter of ridge regression when encoding with CatBoost.

The best value for α was 0 (or very close to it), starting from $\alpha = 0$, the value for the test R^2 coefficient kept decreasing and both the train and test value kept getting worse without ever improving. This result can be seen in image 5.

The exact same happens with the Target encoding even while reporting worse values than CatBoost encoding, so this seems like a side effect of using supervised encoding techniques.

Trying to do the same analysis but using binary encoding shows that α can have a beneficial effect on the risk of the predictor, as shown in the image 6 where we can see that the best value for α is around 9750, the R^2 coefficient is still negative (-0.057) but it's an improvement over the linear predictor.

Figure 5: Test and train R^2 values for various alpha values using catboost encoding



By analyzing the pure results and comparing again CatBoost and Target, we can see that CatBoost ends up with a coefficient of determination value of 0.719 for the test results (which is equal to the one obtained in the comparison of encodings since the best value for *alpha* is 0) compared to 0.646 for the test results of the Target encoded version.

These results confirm that supervised encoding are far superior to unsupervised encodings to help with the usage of simple models for regression, but they also nullify the benefits of using a Ridge Regression model since their best value for α is zero so when this predictor is equal to the Linear Regression model.

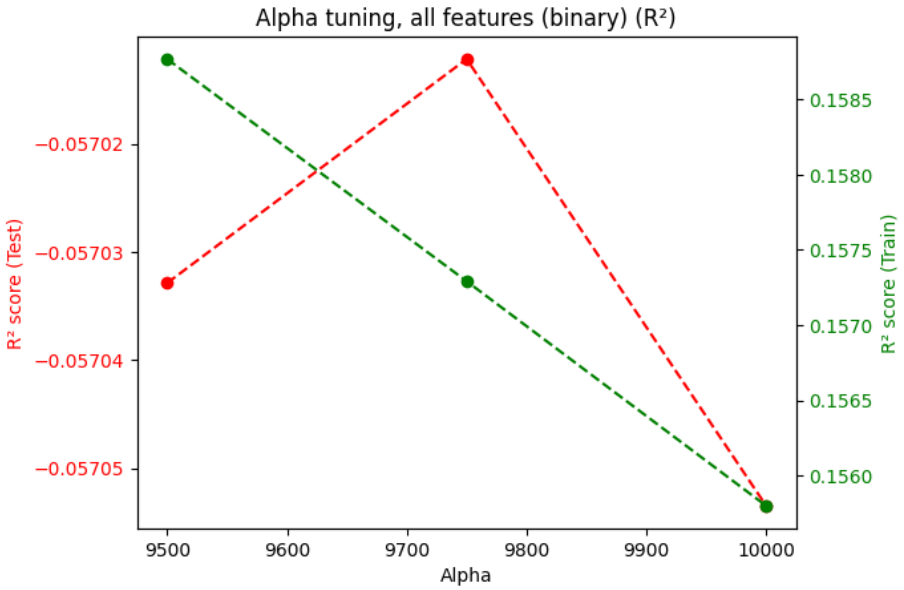
5 Conclusions

In this document, a mix of unsupervised and supervised encoding techniques were analyzed and tested to see which would perform better using a ridge regressor to predict the popularity of the song, given some information about them.

The simple ridge regression model is not complex enough to learn this dataset, using only numerical feature the coefficient of determination value is 0.03, so the information actually learned by the model is not enough to predict the correct value and is a clear case of underfitting since both training and test results are close to 0.

This if further supported by the fact that while comparing the results of the

Figure 6: Test and train R^2 values for various alpha values using binary encoding



numerical only and all features experiments, the performance of the model worsens by adding more information instead of improving when using unsupervised encoding.

Using supervised techniques for encoding helps to get better predictions even with such a simple model, improving performance from a R^2 coefficient very close to 0 to 0.719 but introduces the variable of possible data leakage caused by these techniques.

One side effect of using supervised techniques is that the added regularization parameter loses all its benefits since the best ridge predictor is the one with $\alpha = 0$, negating any of the benefits of using ridge over linear.

5.1 Possible extensions

Some possible extensions of this work could be:

- Look further into the data leakage problem and prediction shift problem (as reported in [2])and check whether it has an effect or not in this case and how much
- Analyze the problem of the ridge regression α parameter sorting no beneficial effect when using the supervised encoders
- Compare the results of this simple model with more complex ones like Kernel Ridge Regression or SVM

References

[1] Daniele Micci-Barreca. 'A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems'. In: *SIGKDD Explor. Newsl.* 3.1 (July 2001), pp. 27–32. ISSN: 1931-0145. DOI: 10.1145/507533.507538. URL: https://doi.org/10.1145/507533.507538.

[2] Liudmila Prokhorenkova et al. *CatBoost: unbiased boosting with categorical features*. 2019. arXiv: 1706.09516 [cs.LG].

A Appendix - Reproducibility

All the data and graphs reported were generated using Python 3.10.12 on Ubuntu 22.04 using WSL2 on Windows 11.

All the python packages used, along with their versions, are available in the file *requirements.txt* available in the repository.

All the code is available in the *repository* and was left as is when the project was handed in.

All the random seeds, where needed, were set to 1 to ensure reproducibility of the results.