Statistical Methods for Machine Learning Project

Implementation of a Ridge Regression Model from scratch

Giovanni Arvati

September 2023

Contents

1 Project outline

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:

- first, train the model using only the numerical features
- second, 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,
- in both cases, experiment with different training parameters,
- use 5-fold cross validation to compute your risk estimates,
- 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.

2 Declaration

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 offences 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.

3 Dataset

3.1 Dataset explanation

For the project, the dataset that has been used is a Spotify Tracks Dataset available on the Kaggle website. The dataset contains information of around 114 thousand tracks and it is divided in 21 columns, as below:

- **track_id**: The Spotify ID for the track
- **artists**: The artists' names who performed the track
- **album_name**: The album name in which the track appears
- **track_name**: Name of the track
- **popularity**: The popularity of a track is a value between 0 and 100, with 100 being the most popular.
- **duration_ms**: The track length in milliseconds
- **explicit**: Whether or not the track has explicit lyrics (boolean)
- **danceability**: How suitable a track is for dancing
- **energy**: Is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
- **key**: The key the track is in.
- **loudness**: The overall loudness of a track in decibels (dB)
- **mode**: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0
- **speechiness**: Speechiness detects the presence of spoken words in a track.
- **acousticness**: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
- **instrumentalness**: Predicts whether a track contains no vocals.
- **liveness**: Detects the presence of an audience in the recording.
- **valence**: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
- **tempo**: The overall estimated tempo of a track in beats per minute (BPM).

- **time_signature**: An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of 3/4, to 7/4.
- **track_genre**: The genre in which the track belongs

3.2 Preprocessing

To work with the dataset, it was imported into a python script where it was possible to work on it. The dataset was in the form of a csv file, that was imported and transformed in a Pandas dataframe. The only preprocessing operations done to the dataset were:

1. Removing the first column because it was a copy of the index column of the pandas dataframe
2. Removing all the records with 0 popularity. This because out of 114k total records there were 16k records with a value 0 in popularity. The popularity value is the target of the regression model and so, I thought that having more than the 10% of the records with a value of 0 would have biased too much the model and would have brought to worst results.

4 Ridge Regression Model

To implement the ridge regression from scratch it was necessary to compute the ridge estimator, that can be defined as:

$$\mathbf{B} = (\mathbf{X}^\top \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

where $\mathbf{X}$ is the **design matrix** and it is composed by a first column of all ones and the other columns are the columns in the dataset that are chosen to predict the target feature. α is the regularization factor and has to be greater or equal than 0. $\mathbf{I}$ is the identity matrix and $\mathbf{y}$ is a vector equal to the target feature column of the dataset.

The python implementation of this formula is reported below:

```python
import numpy as np

def ridge_regression_fit(train, target, predictors, alpha):
    X = train[predictors].copy()
    y = train[target].copy()

    mean = X.mean()
    std = X.std()

    X = (X - mean) / std #standardization
    X["intercept"] = 1
    X = X[['intercept'] + predictors]

    penalty = alpha * np.identity(X.shape[1])
    penalty[0][0] = 0 #First element 0 by convention

    B = np.linalg.inv(X.T @ X + penalty) @ X.T @ y
    B.index = ["intercept"] + predictors

    return B, mean, std
```

After having computed the ridge estimator $\mathbf{B}$, to generate the predictions the only thing to do was to calculate the design matrix with the records in the *test set* and then multiply it with the ridge estimator.

The code is implemented as follow:

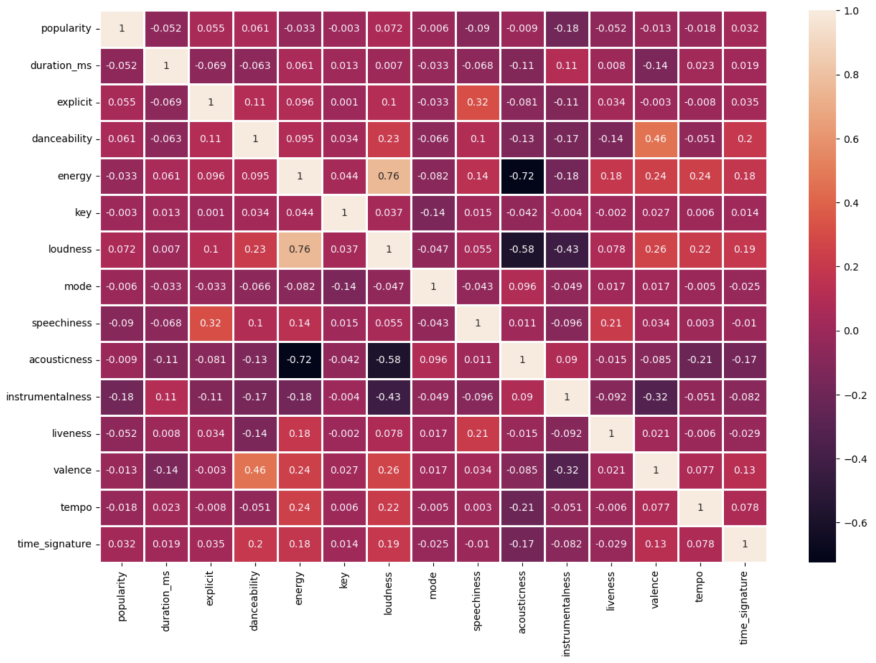
```python
def ridge_regression_predict(test, predictors, mean, std, B):
    test_X = test[predictors]
    test_X = (test_X - mean) / std #standardization
    test_X["intercept"] = 1
    test_X = test_X[['intercept'] + predictors]
    predictions = test_X @ B
    return predictions
```

5 Predictors Choices

Unlike the target feature that is always the **popularity** column of the dataset, in the choices of the predictor features some tests were made with different features. First, were used only numerical features and only and then two categorical features were added, but this will be seen in detail in the following sections.

5.1 Numerical Features

Before decided on what features work I took a look at the correlation matrix of the numerical features of the dataset to see what features have the most direct correlation to the target feature.



5.1.1 Positive Correlation Features

Despite none of these features have a strong correlation with the **popularity** feature, the first try was done by trying to use only the features that have a positive correlation.
The features considered for this case were:

- **explicit**
- **danceability**
- **loudness**
- **time_signature**

5.1.2 Semantic Approach

The second approach was to add other features to those previously chosen. The new ones were chosen using a semantic approach, that is, they were chosen by choosing those that, in subjective judgment, could influence the popularity of a song. The features added are:

- **speechiness**
- **acousticness**
- **instrumentalness**
- **liveness**

5.1.3 All Features

The last approach on numerical features was to train the model with all the numerical features in the dataset.

5.2 Categorical Features

After having worked with only numerical features, it was time to add some categorical features to the model. Among all the non-numerical features, the two chose were:

- **track_genre**
- **artist**

5.2.1 Track Genre

The **track_genre** column was encoded in two different ways to see if the different encoding could have been brought to significant differences. First, it is used a simple One-Hot Encoding. This was possible because there are slightly more than 100 different genres and so, it was possible to apply this type of encoding without increasing the number of columns exponentially. Alternatively, the same column was encoded using the so called target encoding.

5.2.2 Artist

The artist column has been encoded using the target encoding, because differently from the track_genre column, here there were thousands of different artists. So it was not possible to use the One-Hot Encoding without generating a design matrix with a huge number of column that would have brought to really poor performances.

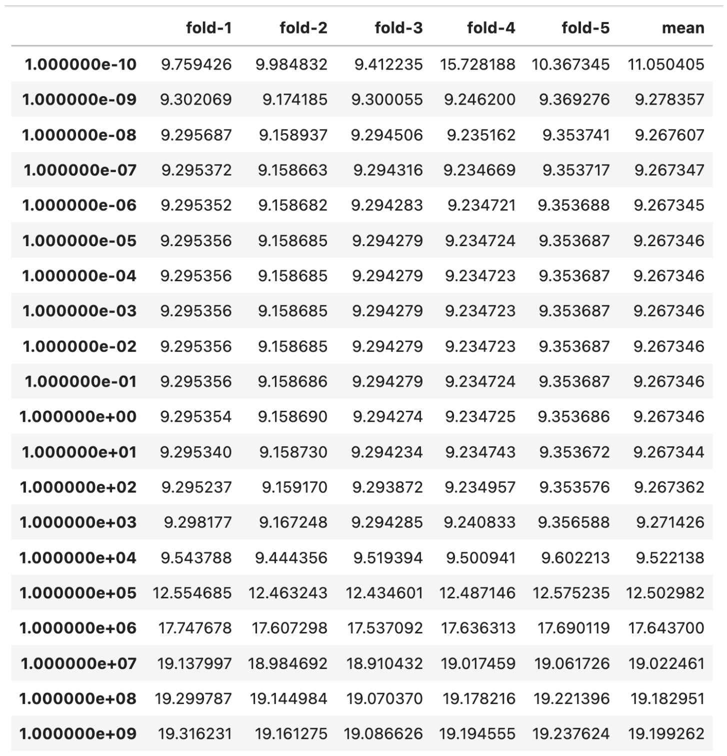
6 5-Fold Cross Validation

The model was trained using a 5-Fold Cross Validation, therefore the dataset was divided into 5 parts and each time the model was trained on 4 of the 5 parts and the remaining one was used to test the model. This is done for all the 5 possible combination of the 5 folds.
Besides this, every time the model was trained, it was trained with different values of the **alpha** parameter. In particular, every training was done trying alpha equal to 10^i where $i \in [-10, 10]$.

7 Results

For each test of the model there were computed some metrics to better understand the quality of the model. The metrics used are: **Root Mean Squared Error** (RMSE), **Mean Absolute Error** (MAE) and the **R-Squared** (R2). These metrics were computed for each values of alpha and for each fold's change. In this way it was possible to calculated which was the best alpha for the model. In particular, the best alpha was the one that minimize the RMSE over all the 5 different folds of the 5-Fold Cross Validation.

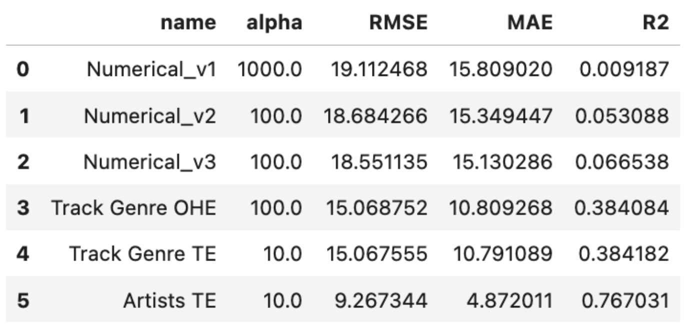
The Root Mean Squared Errors resulting from the testing of the model when it was trained with all the predictors (all numerical plus the two categorical features) can be seen below.



	fold-1	fold-2	fold-3	fold-4	fold-5	mean
1.000000e-10	9.759426	9.984832	9.412235	15.728188	10.367345	11.050405
1.000000e-09	9.302069	9.174185	9.300055	9.246200	9.369276	9.278357
1.000000e-08	9.295687	9.158937	9.294506	9.235162	9.353741	9.267607
1.000000e-07	9.295372	9.158663	9.294316	9.234669	9.353717	9.267347
1.000000e-06	9.295352	9.158682	9.294283	9.234721	9.353688	9.267345
1.000000e-05	9.295356	9.158685	9.294279	9.234724	9.353687	9.267346
1.000000e-04	9.295356	9.158685	9.294279	9.234723	9.353687	9.267346
1.000000e-03	9.295356	9.158685	9.294279	9.234723	9.353687	9.267346
1.000000e-02	9.295356	9.158685	9.294279	9.234723	9.353687	9.267346
1.000000e-01	9.295356	9.158686	9.294279	9.234724	9.353687	9.267346
1.000000e+00	9.295354	9.158690	9.294274	9.234725	9.353686	9.267346
1.000000e+01	9.295340	9.158730	9.294234	9.234743	9.353672	9.267344
1.000000e+02	9.295237	9.159170	9.293872	9.234957	9.353576	9.267362
1.000000e+03	9.298177	9.167248	9.294285	9.240833	9.356588	9.271426
1.000000e+04	9.543788	9.444356	9.519394	9.500941	9.602213	9.522138
1.000000e+05	12.554685	12.463243	12.434601	12.487146	12.575235	12.502982
1.000000e+06	17.747678	17.607298	17.537092	17.636313	17.690119	17.643700
1.000000e+07	19.137997	18.984692	18.910432	19.017459	19.061726	19.022461
1.000000e+08	19.299787	19.144984	19.070370	19.178216	19.221396	19.182951
1.000000e+09	19.316231	19.161275	19.086626	19.194555	19.237624	19.199262

So, the best alpha is the one corresponding to the cell with the lowest value in the *mean* column. In the case above, the best alpha is 10.

The table above was generated for all the different type of training done to the model and for each of them the best alpha with the corresponding RMSE, MAE and R2 are saved. These resulting in the following table:



	name	alpha	RMSE	MAE	R2
0	Numerical_v1	1000.0	19.112468	15.809020	0.009187
1	Numerical_v2	100.0	18.684266	15.349447	0.053088
2	Numerical_v3	100.0	18.551135	15.130286	0.066538
3	Track Genre OHE	100.0	15.068752	10.809268	0.384084
4	Track Genre TE	10.0	15.067555	10.791089	0.384182
5	Artists TE	10.0	9.267344	4.872011	0.767031

With Numerical_v1, Numerical_v3, Numerical_v3 is meant the three different choices of numerical features as respectively explained in the previous section. OHE and TE are acronyms that stand, respectively, for One-Hot Encoding and Target Encoding.

As shown by the table, using only numerical features best results are obtained when all the features are taken into account. Moreover, adding categorical features helps a lot in getting better results. In particular, adding only the track genre helps but there are no significant differences in using One-Hot Encoding to the respect to Target Encoding.
The addition of the artist feature has the greater impact in terms of results and this could be due, in part, to the choice of using the target encoding as encoding method.