Statistical methods for machine learning

Nicolas Audoux

September 2023

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.

Contents

1 Introduction

The goal of this project is to implement the ridge regression algorithm from scratch. To do so, We will use python and the library Numpy. We will them test the algorithm on the Spotify Tracks dataset available on Kaggle.

2 Dataset

The Spotify Tracks dataset is a dataset describing over 100 thousands tracks. Each track is described by 20 parameters. Let's split them between categorical features and numerical one.

- Numerical features
  - Popularity: How popular is a song (100 being the most popular)
  - track_id: The Spotify ID of the track
  - duration_ms: The duration of the track in milliseconds
  - danceability: A score to describe if we can easily dance on this song
  - energy: Measure the intensity and activity of the song
  - loudness: The loudness of the track in dB
  - speechiness: Proportion of spoken words in the track
  - acousticness: probability for the track to be acoustic
  - instrumentalness: Predict if a track contains vocal or not (1 meaning there are no vocal content)
  - liveness: Detect the presence of an audience in the recording
  - valence: Describe whether a track is positive or negative
  - tempo: The estimated tempo of the track
  - time_signature: The estimated time signature of the track

- Categorical features
  - artists: The artist who performed the track
  - album_name: The album name in which the track appears
  - track_name: The name of the track
  - explicit: Whether or nor a track contains explicit lyrics
  - key: The key of the track
  - mode: the modality of the track (major/minor)
  - track_genre: The genre of the track

First we will use only numerical features to predict the popularity of the song and then we will also use categorical features.

3 Ridge Regression

The ridge regression is an algorithm which output a linear predictor. A linear predictor for $X = \mathbb{R}^d$ is defined as followed $h : \mathbb{R}^d \to \mathbb{R}$ such as $h(x) = w^T x$. In a classic linear regression, the predictor is the Empirical risk minimization (ERM) with respect to the square loss. Let's call w_s the coefficient of such predictor for a training set $(x_1, y_1), ..., (x_m, y_m) \in \mathbb{R}^d \times \mathbb{R}$.

$$w_s = \underset{w \in \mathbb{R}^d}{\arg\min} \sum_{t=1}^{m} (w^T x_t - y_t)^2$$

In a ridge regression, we add a regularizer in the ERM function to reduce the variance.

$$w_S = \underset{w \in \mathbb{R}^d}{\arg\min} \sum_{t=1}^{m} (w^T x_t - y_t)^2 + \alpha ||w||^2$$

Where $\alpha > 0$ is the regularization parameter. View all vectors as column vector. Let $y = (y_1, y_2, ..., y_m)$ and S, the $m \times d$ design matrix such that $S^T = [x_1, x_2, ..., x_m]$. Now we have that

$$w_s = \underset{w \in \mathbb{R}^d}{\arg\min} ||Sw - y||^2 + \alpha ||w||^2$$

Since $F(w) = ||Sw - y||^2 + \alpha||w||^2$ is a convex function we just have to satisfies $\nabla F(w) = 0$ to find the best coefficient possible.

$$\nabla||Sw - y||^2 + \alpha||w||^2 = 2S^T(Sw - y) + 2\alpha w 2S^T(Sw - y) + 2\alpha w = 0$$

So

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

It is important to note that the solution doesn't necessarily pass through the origin. In this case we have an equation of the form: $h(x) = w^T x + k$ With $k \in \mathbb{R}$. In order to compute k like a coefficient, we add an extra feature equal to 1 such as $x = (1, f_1, f_2, f_3, ..., f_d)$. Doing so the first coefficient we be equal to k . This value mustn't be penalize like the other coefficients since it doesn't depend on a feature. So, in the penalize term which is αI, we need to put a zero instead of the first one in the identity matrix.

To evaluate each predictor we will use some loss function. For each loss function we will have y the true labels,$\hat{y}$ the predicted labels and $n = ||y|| = ||\hat{y}||$. We will use two loss function: the mean squared error (MSE) and the explain variance (EV) respectively $l_{MSE}(y, \hat{y})$ and $l_{EV}(y, \hat{y})$

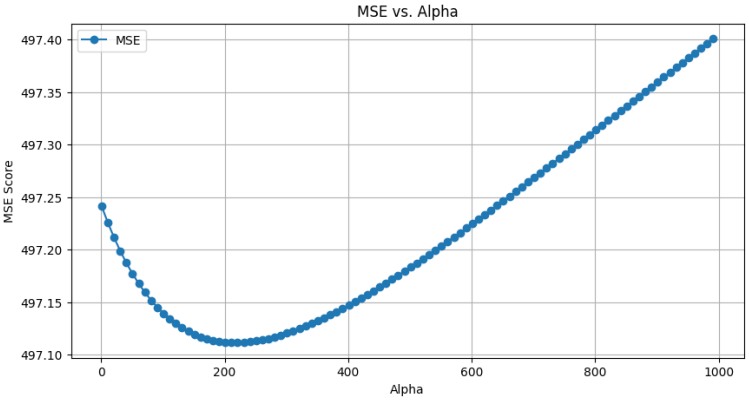
$$l_{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

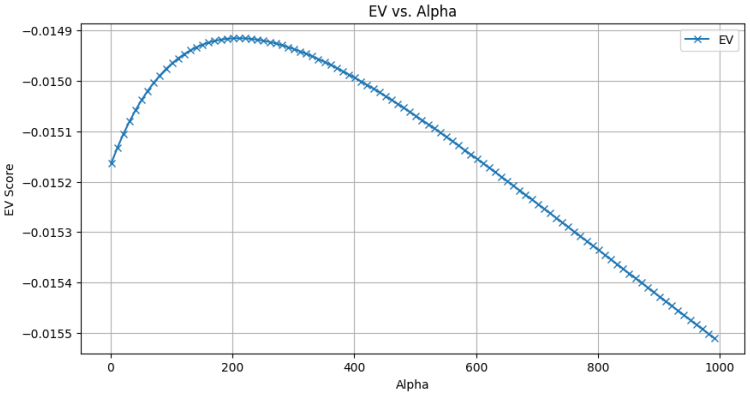
$$l_{EV}(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2}$$

With $\overline{y}$ being the mean of y.

4 Numerical features

In this section we will try to predict the popularity of a song based on 11 numerical features over its 12. Indeed, the id of the track is completely useless to us since this is just a unique random number used to represent the song. It can't bring us information about its popularity. To test our algorithm we will perform hyperparameter tuning for α on a 5-fold cross validation. To find the best value for α we will first try values between 1 and 1000 with a step of 10. We found that the best α is 211 for both MSE and EV.





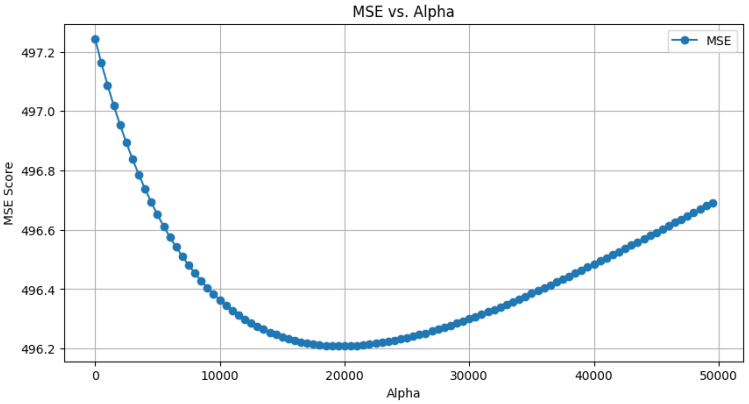
To be more precise we rerun the cross validation with α going from 200 to 220. Doing so, We find that the best value for α is 217 with $l_{MSE}(y, \overline{y}) = 497$ and $l_{EV} \approx 0$.

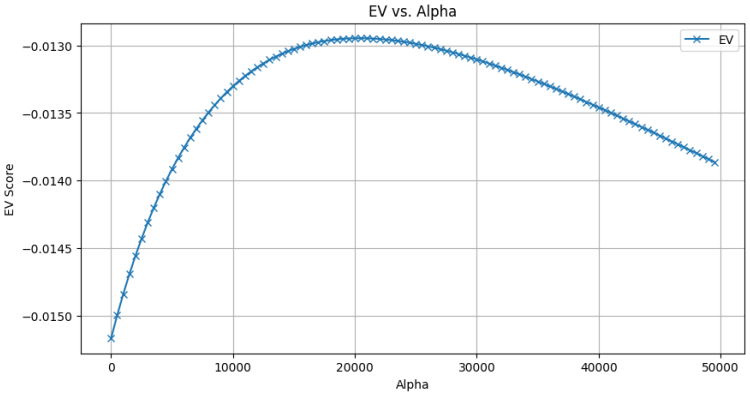
This show us that our model is not efficient at all since it is not able to explain the variance of the data. The fact that we didn't normalize our data can explain this. Indeed, having features with completely different scales can be source of error since features with bigger values will have more weight. To avoid this issue, we will apply the z-score normalization on our data:

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

with x being the value, μ the average and σ the standard deviation.

When running the cross-validation again, we obtained that α should be around 19500 to minimize the MSE an around 20500 to maximize the EV.





Having α much bigger means that we penalize even more the weight of each features. In other words, our predictor tends to predict a values which is close to the mean value.

5 Numerical and categorical features

We will now add categorical features to our numerical features to predict the popularity of a song.

5.1 Converting categorical features

First we can simply add *key* and *mode* to our features. There are already numerical features but they are used to represent category. The mode can be either minor or major and the key is one of the 12 existing keys. That is why we consider them as categorical features and not numerical ones. We can easily add the *explicit* feature by replacing False by 0 and True by 1 which is it quite natural and efficient for our case. To add the *genre* feature, we have to use one hot encoding. In other word each different genre becomes a new feature which have a 1 for value if it is the genre of the track else it has a 0. Doing so, we add 114 new features.

The features remaining (*artists, album_name* and *track_name*) need more discussion. First, let's focus on the artists. This feature can be very useful because is quite natural to think that a famous singer has more chance to create a popular song than a unknown singer. Theoretically a one hot encoder would be the best solution but there are more than 30 000 different singer in the dataset which would result in 30 000 new features. With this many features, it would take way too much time to train the model. Another problem is that even if we had enough time to train the model we wouldn't have enough data. In general an artist will appears 3 or 4 times in the dataset. It is very unlikely to have the same distribution in the training set and the test set which would result in a overfitting model (A predictor very good with seen data and very bad with unseen ones).

A solution to replace the one hot encoder is to replace each artist by a number. It could solve the problems but it creates a new one. Doing so would implies that we create a notion of similarity between artist. For example, if we have Mozart maps to 1 and Bob Marley maps to 2. It would mean, for the model, that Mozart is similar to Bob Marley which is quite

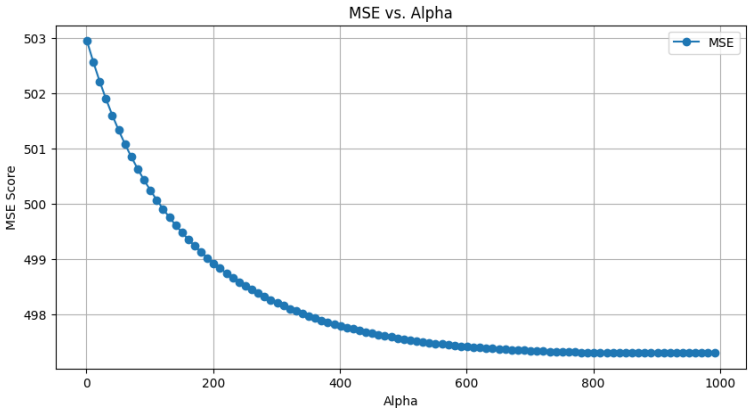
questionable. That makes this solution not suitable for us because we don't have an objective way to sort the artists.

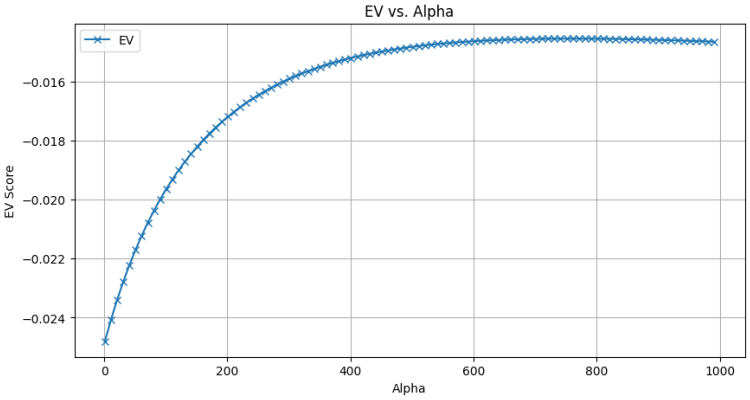
For the album's name and the track's name, we encounter a similar issue. In both case one hot encoding is impossible for the same reason such as replacing each value by a number. However, we could use similarity between names (of a track or an album) quite easily. Indeed, if we use Levenshtein distance, we can say whether two names are close or not. This solution is a way to find an objective way to map our values but it doesn't mean that it's the best one. We could for instance replace each name by a topic is related to (love, space ,travel...).

Finally, even if we could add the song's name and the album's name as feature we won't do it because of the computational time.

5.2 Training the model

Once again, we will use hyperparameter tuning to find the best value for α Do so we will start by doing 5-fold cross validation with α going from 1 to 1000 with a step of 10 Doing so, we find that the best value for α to minimize the MSE is 881 with $l_{MSE}(y, \hat{y}) = 497.3$ and the best value for α to maximize the EV is 761 with $l_{EV}(y, \hat{y}) \approx 0$





Once again the result are not great. We can't find a clear relation between those features and the popularity of a song. Maybe the features we didn't use (like the artist) could help improving the model but even so its very unlikely our model is going to be way better.

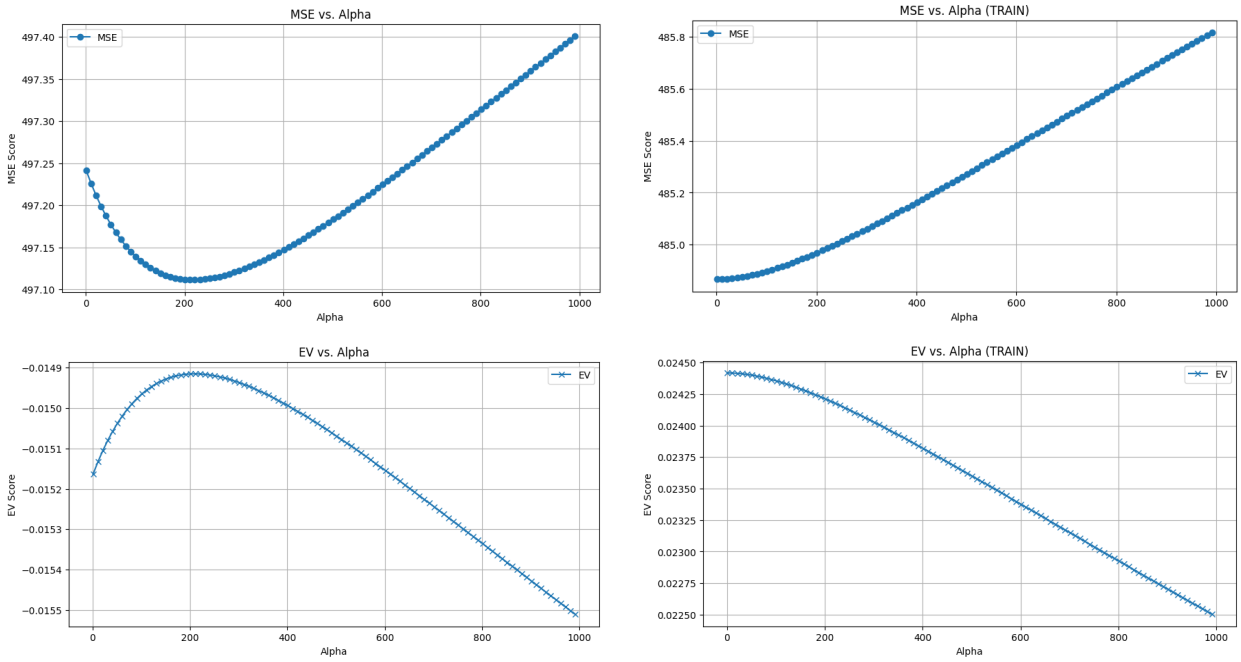
6 Risk estimation

We saw that with or without the categorical features, our model have very poor performances. The question is: why ? The performances of our models can be low for three main reasons:

- Overfitting
- Underfitting
- Bayes

Overfitting

Overfitting occurs when the model is very efficient with already seen data and very inefficient with unseen data. Using the ridge regression reduce the risk of overfitting since we penalize the coefficient of the features which make the train error bigger. However, if the data distribution is not similar between the train set and the set we could still have some overffiting. One way to detect overfitting is to compare test error and train error.



As expected, we can see that no matter what loss function we use to compute the error, the training error is growing with α. We also see that the difference between training error and test error is not very big. With Mean Squared error we find an error of 485 for the training part (with only numerical features) whereas it's 497 for the test part. Since our test error and train error are very close, it's very unlikely that our model is overfitting.

Underfitting

A model is underfitting when it's to simple to capture the relation between the different features. As a result we have poor result for both train and test set. Ideally, we would use the Chernoff-Hoeffding bound to estimate how far from the best predictor we are which is similar to find if our model is underfitting or not. Unfortunately, this estimation required that we have a finite set of predictor generated by our algorithm which is not the case with ridge regression.

It's probable that our model is to simple. Indeed, we are only using linear predictors whereas it's not certain that there is a linear relation between those different features. Moreover, with the penalization of the coefficient, our model is even more simple since it tends to predict the mean value. So, it's probable that our model is underfitting because of the choice of the algorithm we used and how we converted the categorical feaatures.

Bayes error

The Bayes error is the error we can't avoid because of the data distribution. It's the error of the best theoretical predictor possible. Once again we can hardly estimate it since we don't know the data distribution. However, the relation between our features and the popularity of a song is questionable. Indeed, if there are not related, even the best predictor can't have good results.

7 Conclusion

To conclude, estimating the popularity of a song based on its numerical and categorical features using ridge regression seems complicated. First, convert every categorical features into numerical one can be quite challenging if we don't want to add bias into the data. Moreover, even with the best method to convert those features, it's quite likely that the model will underfit since we are only using linear predictors to find the popularity. Lastly, we can't be sure that our features are related to the popularity.

A way to avoid some problems we encountered could be to use different algorithm to create predictors such like neural network, decision tree, etc.