● Preprocessing

○ Year

- Dropped any rows with empty year value or out of range (year < 1921 or year > 2020).
- Replaced the **year** column with **yearsSinceCreation** column and replaced each value with (2020 - value).
- The following code represents the previously explained operations:

```python
# drop rows with invalid values or out of range (797 rows)
def cleanYear(songs):
    songs = songs.dropna(subset=['year'])
    songs.loc[songs.year > 1921, 'year'] = 2020 - songs.year
    songs = songs.rename(columns={'year': 'yearsSinceCreation'})
    return songs
```

○ Removing rows with empty cells

- For each row with at least one empty cell we drop that row.
- It doesn’t affect the data that much because the number of these rows is relatively small.
- The following code represents the previously explained operations:

```python
def removeEmpty(songs):
    for col in songs.columns:
        songs = songs.dropna(subset=[col])
    return songs
```

○ Dropping columns

- Dropping the “id” column because it shouldn’t have any effect on the data.
- Dropping the “name” column because no strong effect exists.

- Dropping the “release_date” column because it has a lot of garbage data and it can be replaced with “yearsSinceCreation” column anyway.
- The following code represents the previously explained operations:

```python
def dropCols(songs):
    songs = songs.drop(columns=['id', 'release_date'])
    songs = songs.drop(columns=['name', 'artists'])
    return songs
```

○ Removing song’s duplicates

- If two or more songs have the same name and artists we consider them as one song and merge the other features.
- For each feature we take the average of their values.
- For binary features like “explicit” we round after taking the average.

- The following code represents the previously explained operations:

```python
# merging groups of songs with the same name and artists
# taking mean values for the other columns
def mergeDuplicates(songs):
    songs = songs.groupby(['artists', 'name'],
    as_index=False).agg({
        'valence':np.average,
        'yearsSinceCreation':np.average,
        'acousticness':np.average,
        'danceability':np.average,
        'duration_ms':np.average,
        'energy':np.average,
        'instrumentalness':np.average,
        'liveness':np.average,
        'loudness':np.average,
        'tempo':np.average,
        'speechiness':np.average,
        'explicit':np.average,
        'mode':np.average,
        'popularity':np.average
    })
    songs.loc[:, 'explicit'] = round(songs.explicit)

    return songs
```

○ Reformatting the artists feature

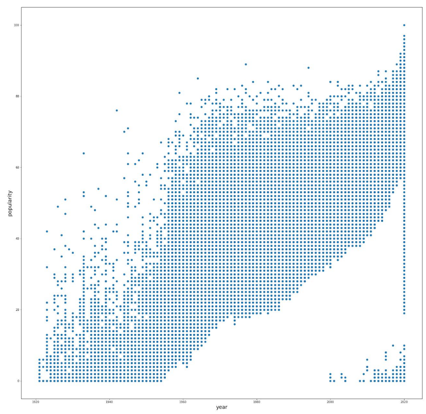
- For each stringified artists list we unstringify it first.
- Using hashing we group each set of artists to one of 5 groups randomly with equal probability.
- We use one-hot-encoding on this categorical feature.
- We join the new 5 columns with the old data frame and remove the artists column.

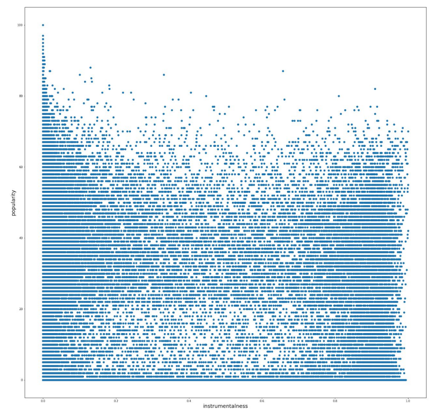
- The following code represents the previously explained operations:

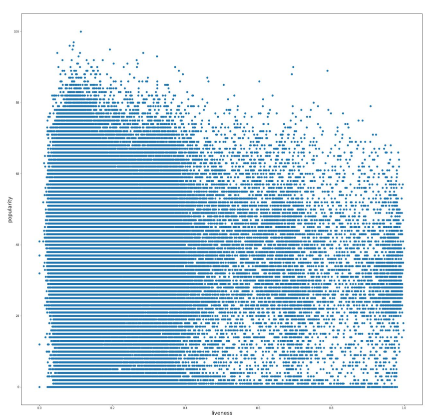
```python
# drop rows with invalid values and destringify
# the list of artists
def cleanArtists(songs):
    print("clean artists")
    songs['artists'] =
    songs['artists'].apply(lambda x: x[1:-1].split(', ')
    if(type(x) == str and len(x)) else [])
    songs['artists'] =
    songs['artists'].apply(lambda x:
    list(map(lambda y: y[1:-1], x)) )
    encoder = ce.HashingEncoder(cols=['artists'],
     n_components=5, return_df=True, drop_invariant=True)
    df = encoder.fit_transform(songs['artists'],
     songs['popularity'])
    songs = df.join(songs)
    return songs
```

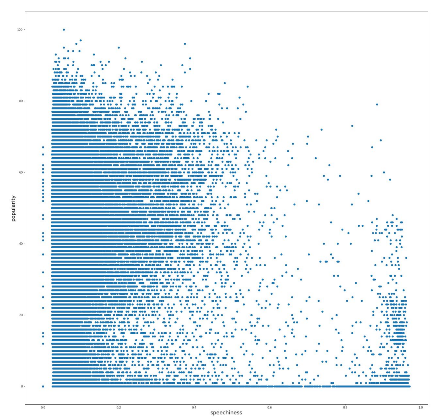
Features analysis

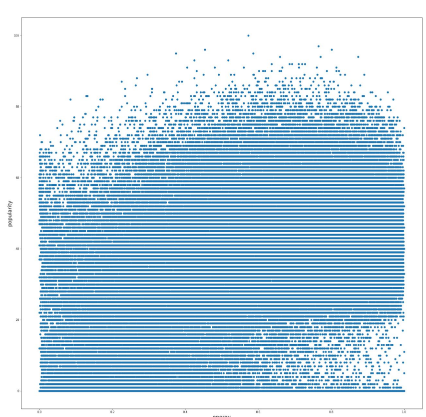
Correlation between the features and the output


popularity
year

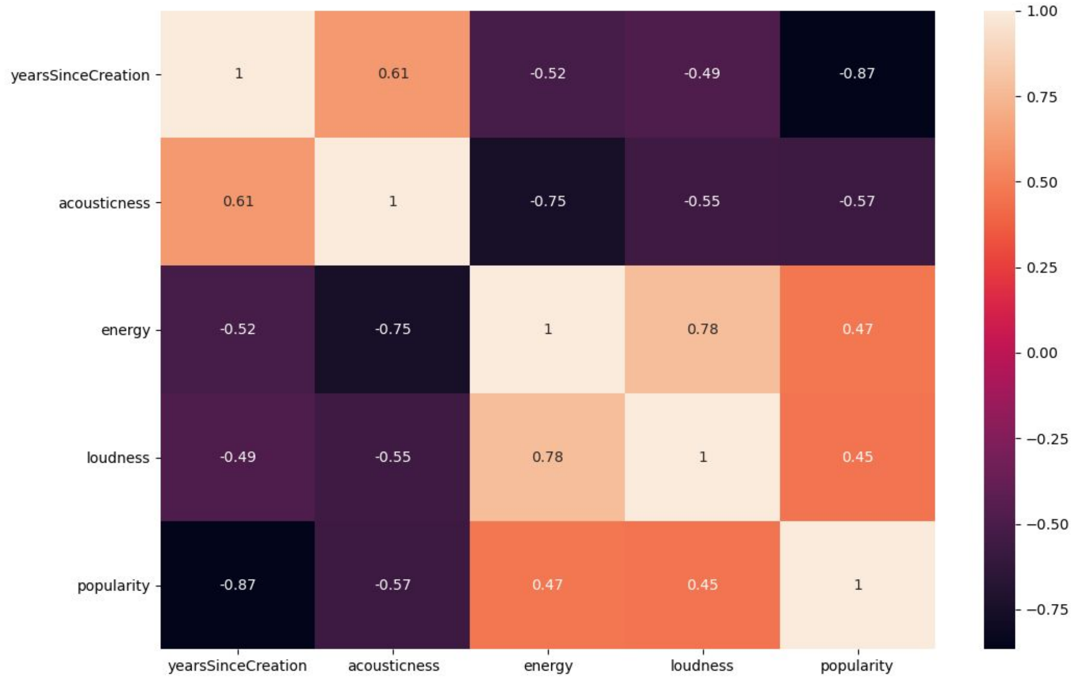

popularity
instrumentalness


popularity
liveness


popularity
speechiness


popularity
energy

○ Correlation between top features



- **regression techniques**

We used all features except id and name (and artists in some cases)

○ Regression with ridge

We examined it with multiple alphas [.01, .1 , 1 , 10] and polynomial degrees [1,2,3,4] , With the normalize flag to perform l2-norm

Results

```
merge duplicates songs (might take a while)
======== all selected features with degree 3 and alpha 0.01 ========
Co-efficients len :  1540
Co-efficients max : 36.517066899829864
Co-efficients min : -43.14532062823003
Intercept :57.177
MSE :99.645
MAE :7.383
r2 :0.784
execution time  9.45405125617981
```

```
======== all selected features except artists with degree 4 and anlpha 0.01 (current best fit) ========
Co-efficients len :  2380
Co-efficients max : 78.72859431061356
Co-efficients min : -334.60969731934983
Intercept :53.567
MSE :97.802
MAE :7.246
r2 :0.788
execution time  15.770998477935791
```

- ## Regression with the top correlated features

First, we tried to get the top correlated features with the wanted output column and then we trained a linear regression model with them changing the alpha and the degree of the polynomial features

Results

```
======== top corr features with degree 3 and anlpha 0.01 ========
Co-efficients len :  35
Co-efficients max : 7.123153438405451
Co-efficients min : -8.101617640741996
Intercept :62.977
MSE :108.913
MAE :7.696
r2 :0.763
execution time  0.4050014019012451
```

```
======== top corr features with degree 4 and anlpha 0.01 ========
Co-efficients len :  70
Co-efficients max : 5.917493573561504
Co-efficients min : -9.632931132117818
Intercept :61.740
MSE :107.141
MAE :7.567
r2 :0.767
execution time  0.43399930000305176
```

- ### Regression with the cross-validation

Training a model with all selected features with cross-validation with different Ks values and test with negative mean squared error and r2

Results

```
======== regression with cv k=10 ========
cv neg MSE : -110.4290330804414
cv r2 : 0.7605163362444929
Co-efficients len :  2380
Co-efficients max : 78.72859431061356
Co-efficients min : -334.60969731934983
Intercept :53.567
MSE :97.802
MAE :7.246
r2 :0.788
execution time  23.467999696731567
```

Training, validation, and testing

- Training: 70% of the dataset.
- Validation: k-fold validation where k=10.
- Testing: 30% of the dataset.

Conclusion

- ### The first intuition:

        - We were assuming that the artists feature is the biggest factor by far.
        - The release year is a respectful factor.
        - Energy has more effect than accousticness.
    - ## Conclusion After analyzing the data:
        - Although artists are a big factor in songs popularity in real life, it doesn't help much to use them in this model, because there are too many artists and it's a categorical feature, so we have to group them into a small number of groups to use (one-hot-encoding) which doesn't produce the best results compared to higher degree regression with no artists feature.
        - It turned out that the release year is the biggest factor we have, with absolute correlation of 87%.
        - Accousticness comes in the second place with more effect than the energy and loudness.
        - Any categorical feature with more than 2 values must be dealt with using methods like (one-hot-encoding).