In [162…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Analysis of Spotify Data

In this notebook I'll analyse a spotify dataset with the following properties:

- 114000 datapoints
- 21 columns

The dataset is available on Kaggle: Spotify Tracks Dataset | MaharshiPandya

For the target variable I will use the `popularity` variable. Below you can see an overview of the structure of the data.

Spotify defines Audio Features for each track, these are numbers that specify characteristics about the track. For eaxmple, the `Instrumentalness` metric is a predictor of how likely the track is to contain vocal content.

Data Exploration Plan

1. Inspect the data size and data types
2. Inspect basic data quality
    - Missing values
    - Value ranges
    - Mapping categorical or numerical values
    - Visualisation
    - Encoding categorical values
3. Hypothesis formulation
4. Hypothesis Testing
5. Conclusion

1. Inspect data size and data types

In [163…
```
data = pd.read_csv("dataset.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        114000 non-null  int64  
 1   track_id          114000 non-null  object 
 2   artists           113999 non-null  object 
 3   album_name        113999 non-null  object 
 4   track_name        113999 non-null  object 
 5   popularity        114000 non-null  int64  
 6   duration_ms       114000 non-null  int64  
 7   explicit          114000 non-null  bool   
 8   danceability      114000 non-null  float64
 9   energy            114000 non-null  float64
 10  key               114000 non-null  int64  
 11  loudness          114000 non-null  float64
 12  mode              114000 non-null  int64  
 13  speechiness       114000 non-null  float64
 14  acousticness      114000 non-null  float64
 15  instrumentalness  114000 non-null  float64
 16  liveness          114000 non-null  float64
 17  valence           114000 non-null  float64
 18  tempo             114000 non-null  float64
 19  time_signature    114000 non-null  int64  
 20  track_genre       114000 non-null  object 
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 17.5+ MB
```

2. Inspect basic data quality

First we'll have to check if there are any inconsistencies in the data and if columns need to be transformed. Let's first check the `object` type columns for number of unique values

In [164…
```
data.select_dtypes(include='object').nunique()
```

Out[164…
```
track_id       89741
artists        31437
album_name     46589
track_name     73608
track_genre      114
dtype: int64
```

Since there are so many values it will not be useful to on-hot encode these columns, apart from the genre column.

Next, let's check for null-values

In [165…
```
data.isna().sum()
```

Out[165…
```
Unnamed: 0          0
track_id            0
artists             1
album_name          1
track_name          1
popularity          0
duration_ms         0
explicit            0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
track_genre         0
dtype: int64
```

In [166…
```
np.where(data.isna())
data.loc[65900]
```

Out[166…
```
Unnamed: 0                      65900
track_id            1kR4gIb7nGxHPI3D2ifs59
artists                            NaN
album_name                         NaN
track_name                         NaN
popularity                           0
duration_ms                          0
explicit                         False
danceability                     0.501
energy                           0.583
key                                  7
loudness                         -9.46
mode                                 0
speechiness                     0.0605
acousticness                      0.69
instrumentalness                0.00396
liveness                         0.0747
valence                           0.734
tempo                           138.391
time_signature                        4
track_genre                       k-pop
Name: 65900, dtype: object
```

There is only one row where null values occur, and it is a single k-pop track. It is likely that something has gone wrong in the parsing of the name of the artist, track and album. Since it is only a single entry, we can safely remove it. However, it may be interesting to look at other k-pop tracks to see if they got parsed correctly.

In [167…
```
data[data['track_genre']=='k-pop'].sample(5)
```

Out[167…

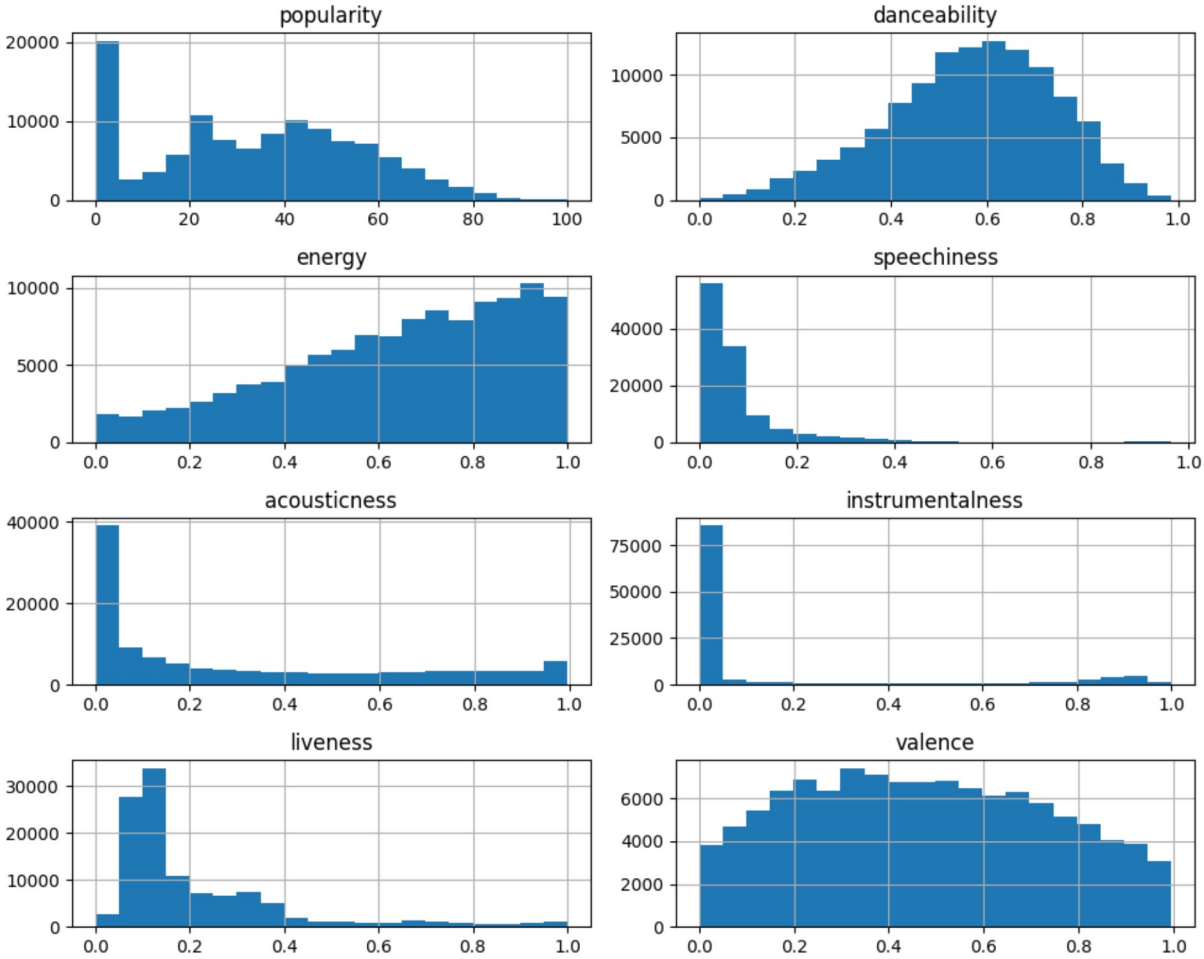
	Unnamed: 0	track_id	artists	album_name	tracl
65594	65594	7mQjVfocpzkKd3PN1TtwKR	Vijay Antony;Jayadev;Rajalakshmi	Dishyum (Original Background Score)	Nenja
65653	65653	3thIARBf9vzQXQYuaOf8Wd	Jubin Nautiyal;Rocky - Shiv	Chitthi	
65734	65734	2Ygw4CPjg1lg4zxTITYY2V	BTS	Proof	
65739	65739	6EfGUQtLk3xVc64tk8ihTL	Ilaiyaraaja	Muthal Mariyathai (Original Motion Picture Sou...	Kili
65502	65502	2e3cJdJ8xWwydl8JIYlCqB	WayV	Love Talk (English Version)	Lov

5 rows × 21 columns

Rerunning the cell above gives a sample of 5 k-pop tracks. It looks like most of them have been translated, with some korean script still available. For now it seems like there are no more issues with this data.

A lot of the variables are numeric, so we can check their basic distribution to start seeing some patterns. Especially the `Audio Features` are interesting. So let's take a look at the following columns: ["popularity", "danceability", "energy", "speechiness", "acousticness", "instrumentalness", "liveness", "valence"].

In [168…
```
audio_feature_cols = ["popularity", "danceability", "energy", "speechiness", "acous
data[audio_feature_cols].hist(figsize=(10, 8), bins=20, layout=(4, 2))
plt.tight_layout()
```

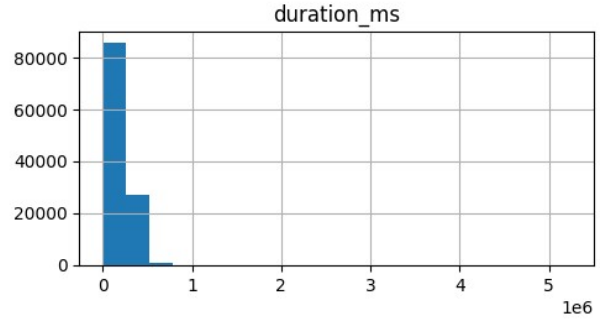


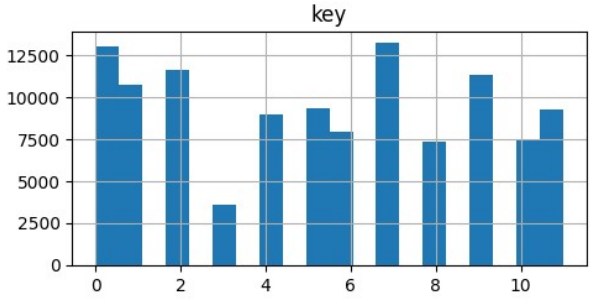
This already gives a lot of insight into how the tracks are distributed. For example, most tracks will have low speechiness, acousticness, instrumentalness and liveness. Which means that most tracks will be studio produced tracks that include sung vocals.

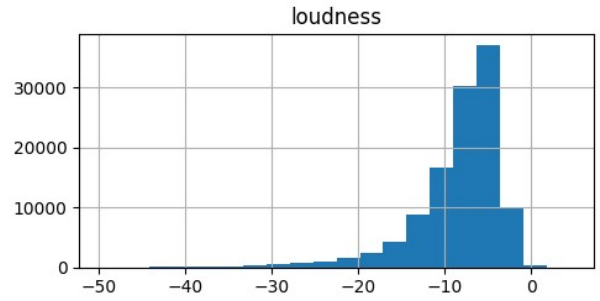
We can also explore the more objective measures of the data in the same way. These are: ["duration_ms", "explicit", "key", "loudness", "tempo", "time_signature", "track_genre"]

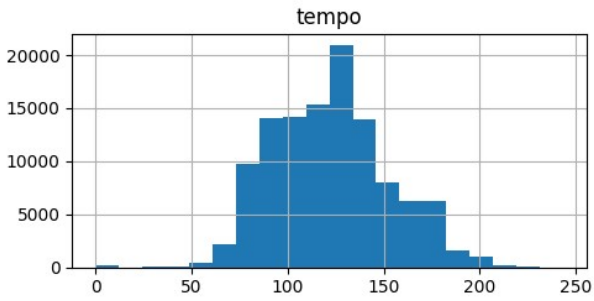
In [169…

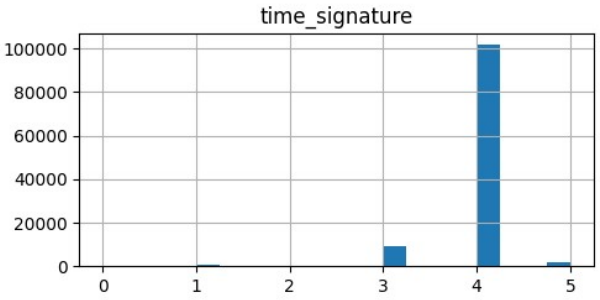
```
audio_property_cols = ["duration_ms", "explicit", "key", "loudness", "tempo", "time
data[audio_property_cols].hist(figsize=(10, 8), bins=20, layout=(3, 2))
plt.tight_layout()
```


duration_ms
1e6

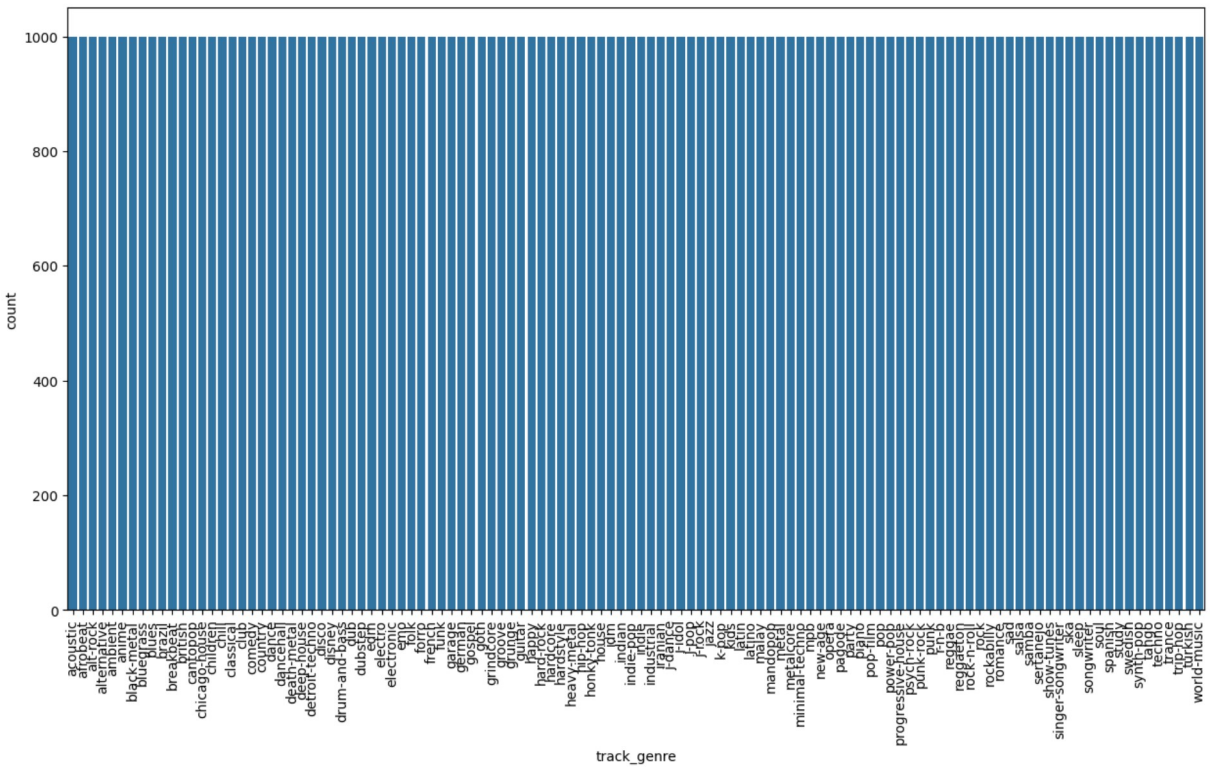

key


loudness


tempo


time_signature

In [170…
```
plt.figure(figsize=(15, 8))
sns.countplot(data=data, x="track_genre")
plt.xticks(rotation=90);
```



From this we can see that the genre is distributed evenly across all categories. So we can safely compare metrics across different genres since we won't have to account for underrepresented data when comparing between genres.

So let's also use one-hot encoding to represent the genres:

In [171…
```
cols_before = len(data.columns)
num_genres = data['track_genre'].nunique()
dummies = pd.get_dummies(data=data['track_genre'], prefix='track_genre')
data = pd.concat([data, dummies], axis=1)
cols_after = len(data.columns)
f"Before: {cols_before}, After: {cols_before}+{num_genres}={cols_after}"
```

Out[171…
```
'Before: 21, After: 21+114=135'
```

So now we have gained some preliminary insight into the data, removed null values and encoded the track_genre. Let's now formulate some hypotheses.

3. Hypothesis formulation

I am interested in finding if any of the Audio Features impact the popularity of a track. So for my first hypothesis I want to check:

Do Audio Features Impact Track Popularity

For each of the audio features look at:

H_0 => The Audio Feature does not impact track popularity.
H_A => The Audio Feature *does* impact track popularity.

For this I will look at the correlation between each audio feature and popularity. First I will look at direct linear correlation, then use a Linear Regression model to capture more complex relationships.

The second hypothesis I want to check is whether genre has impact on popularity:

Does Genre Impact Track Popularity

H_0 => Genre does not impact track popularity.
H_A => Genre *does* impact track popularity.

...

Finally, I want to know if any of the objective measures (like key, tempo and length) have an effect on popularity.

Do Music Properties Like Key and Tempo Impact Track Popularity

H_0 => Music Properties do not impact track popularity.
H_A => Music Properties *do* impact track popularity.

...

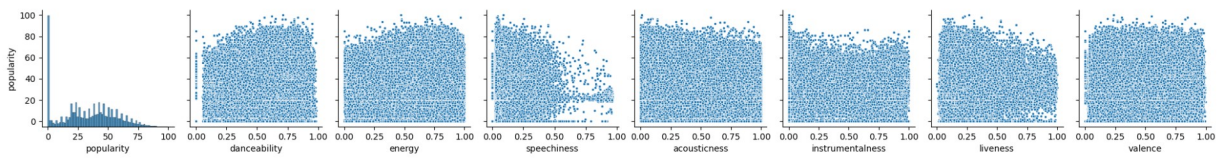
4. Hypothesis Testing

Do Audio Feature Impact Track Popularity

First, let's see if any simple linear correlations are present.

In [172...
```
sns.pairplot(data=data, x_vars=audio_feature_cols, y_vars=['popularity'], markers='
```

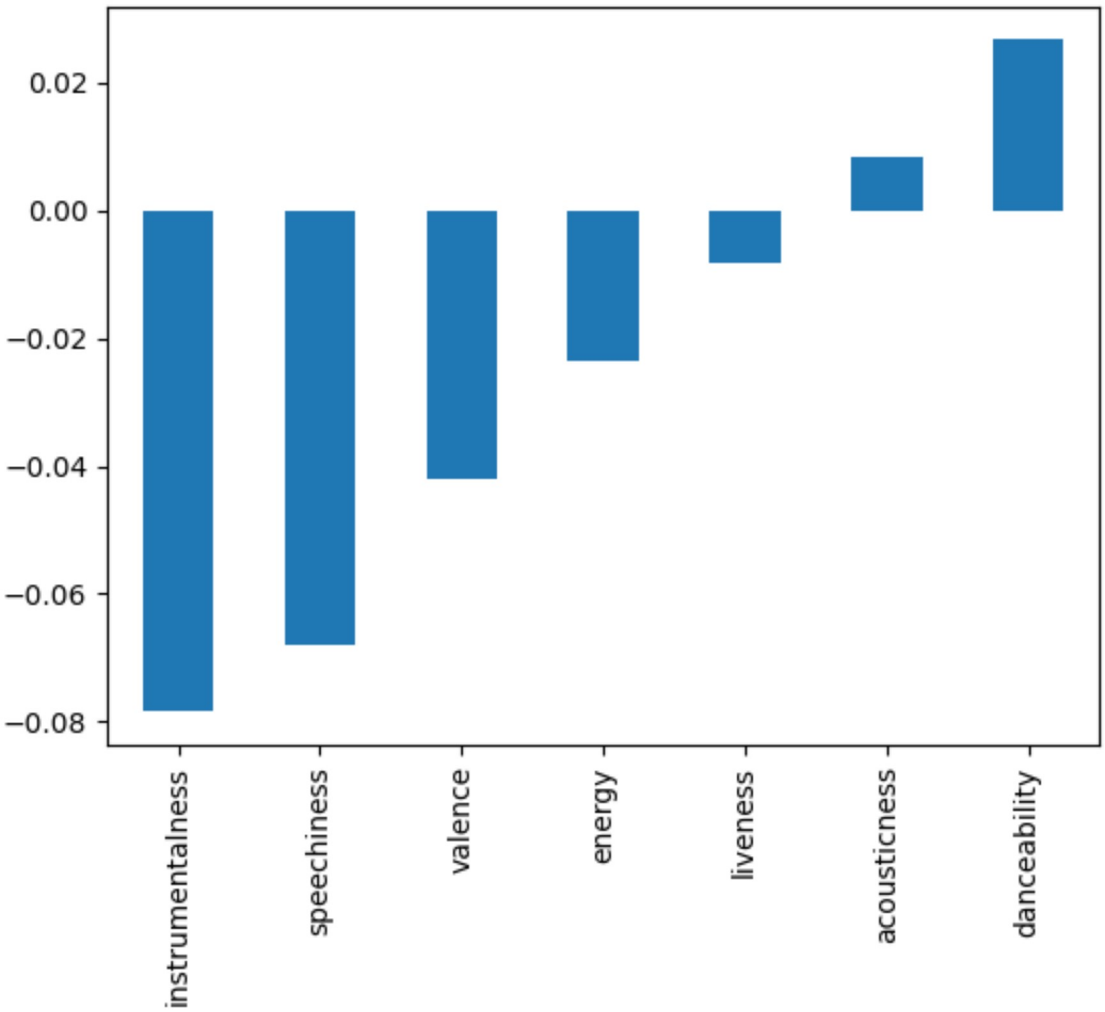
Out[172...
```
<seaborn.axisgrid.PairGrid at 0x166dd1dd1f0>
```



Hmm, quite messy and not very insightful. Let's take a measurement of the correlation values between each of the Audio Features and popularity.

In [173…
```
import statsmodels.api as sm
corr = data[audio_feature_cols].corr(method='spearman')["popularity"].drop('popular
corr.sort_values().plot.bar()
```

Out[173… <Axes: >



Still no strong correlations. Let's take a more informed approach by doing some actual hypothesis testing. For this we use `statsmodels.OLS()` to

In [174…
```
X = data[audio_feature_cols].drop('popularity', axis=1)
y = data['popularity']

X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
model.summary()
```

Out[174...

OLS Regression Results

Dep. Variable:	popularity	**R-squared:**	0.022
Model:	OLS	**Adj. R-squared:**	0.022
Method:	Least Squares	**F-statistic:**	364.7
Date:	Tue, 25 Nov 2025	**Prob (F-statistic):**	0.00
Time:	17:41:31	**Log-Likelihood:**	-5.1445e+05
No. Observations:	114000	**AIC:**	1.029e+06
Df Residuals:	113992	**BIC:**	1.029e+06
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	35.6648	0.418	85.400	0.000	34.846	36.483
danceability	9.2878	0.447	20.791	0.000	8.412	10.163
energy	-0.7862	0.420	-1.874	0.061	-1.608	0.036
speechiness	-12.7612	0.650	-19.644	0.000	-14.034	-11.488
acousticness	-1.2614	0.304	-4.150	0.000	-1.857	-0.666
instrumentalness	-8.8238	0.225	-39.152	0.000	-9.266	-8.382
liveness	1.1891	0.365	3.257	0.001	0.474	1.905
valence	-9.6735	0.311	-31.122	0.000	-10.283	-9.064

Omnibus:	12758.730	**Durbin-Watson:**	0.568
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3496.773
Skew:	-0.015	**Prob(JB):**	0.00
Kurtosis:	2.143	**Cond. No.**	16.0

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

From this we can already see that R^2 is only 0.022, so the model can only explain 2.2% of the effect using these features. The large sample size does make that the model is significant with almost all p-values < 0.05. Only energy has no significant effect on popularity. From this analysis it is hard to generalize to a conclusion, however, we can take a look at the coefficients of the model to see how strongly the Audio Features impact the popularity.

For example, with speechiness, the coefficient being -12.76, when speechiness increases, the popularity will go down propertionally. When speechiness increases by 0.1, popularity will (on average) go down by ~1.2 points. So however significant the effect may be, it is definitely not a strong effect.

- The strongest effects seem to be danceability (positive) and speechiness (negative), instrumentalness (negative) and valence (negative).
- Liveness, acousticness have much smaller effects on popularity.
- Energy has no significant effect on popularity.

These features are not capable of determining the variability of popularity on their own. But possibly interactions between them could do this. Another option is that none of these features properly capture the popularity of a track. These might just be the wrong features to use, and other non-music related features might be more important.

A next step might also be to check for effects corrected by the genre category. It is conceivable that some Audio Features have more impact in specific genres.

Does Genre Impact Track Popularity

Another simple test we can do is see if genre has any impact on popularity. Let's first check if there is any effect overall by using a one-way ANOVA.

In [175…
```
import scipy.stats as stats

# First, group by genre, then unpack into (genre_name, genre_data)
# For each grouped genre dataframe we just grab the popularity column
# Then take the values to get it into the correct format for the ANOVA
genre_groups = [
    genre['popularity'].values for _, genre in data.groupby('track_genre')
]

f, p = stats.f_oneway(*genre_groups)
print(f"F = {f:.3f}, p = {p:.3e}")
```

```
F = 343.462, p = 0.000e+00
```

From this we can see that, with a high f-value and a p < 0.05, the effect of genre overall on popularity is significant. Let's dive in a bit deeper to see where this significance lies.

In [176…
```
import statsmodels.formula.api as smf

model = smf.ols('popularity ~ C(track_genre)', data=data).fit()
```

In [177…
```
model.summary().tables[0]
```

Out[177...

OLS Regression Results

Dep. Variable:	popularity	**R-squared:**	0.254
Model:	OLS	**Adj. R-squared:**	0.253
Method:	Least Squares	**F-statistic:**	343.5
Date:	Tue, 25 Nov 2025	**Prob (F-statistic):**	0.00
Time:	17:41:33	**Log-Likelihood:**	-4.9899e+05
No. Observations:	114000	**AIC:**	9.982e+05
Df Residuals:	113886	**BIC:**	9.993e+05
Df Model:	113		
Covariance Type:	nonrobust		

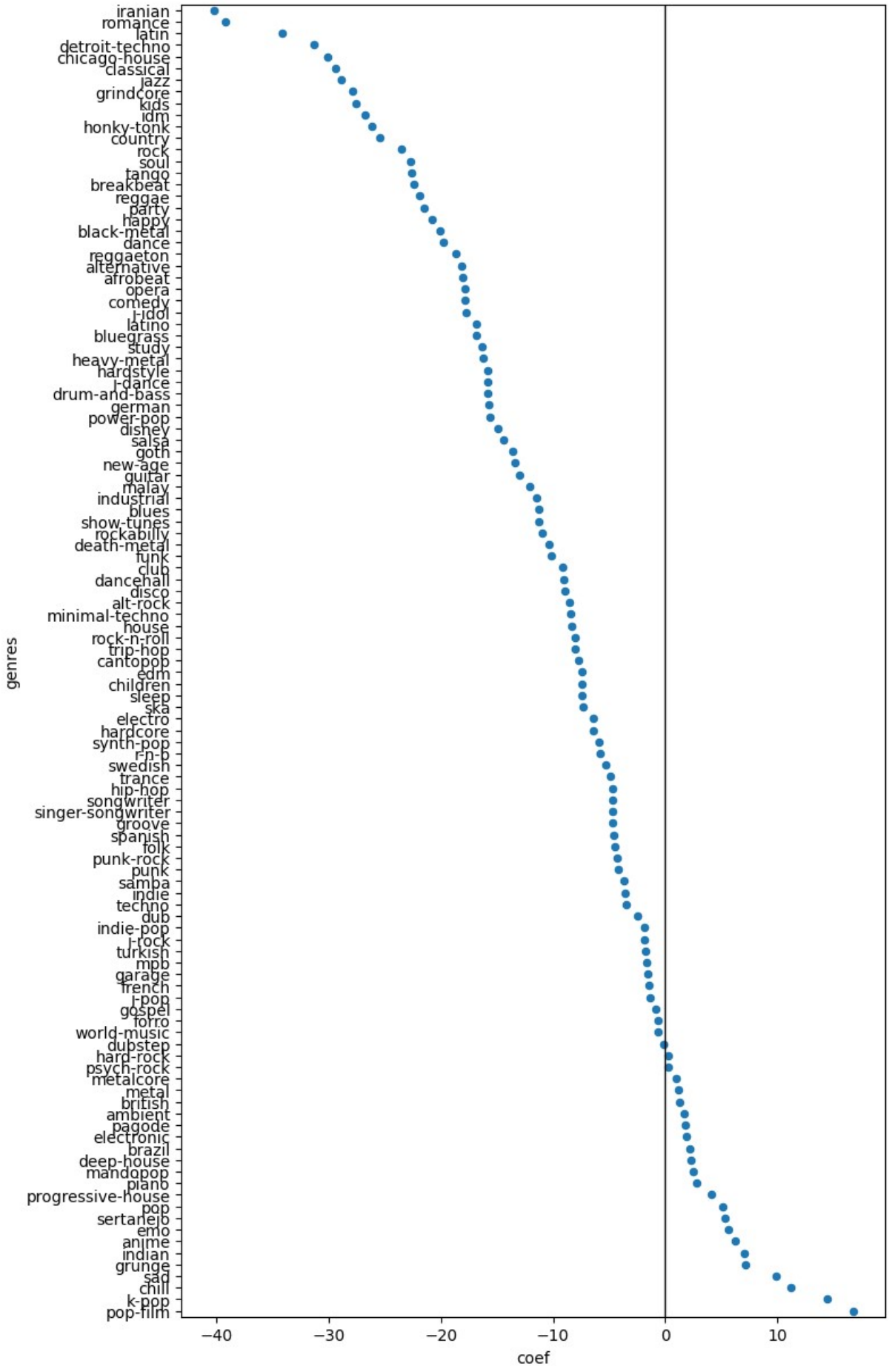
That is already a much stronger effect than the Audio Features from before. Now, with an R^2 of 0.254 (or 25%) there is more explainabilty in this model. We also retrieve a similar F-stat and probability from the simple ANOVA earlier. So our conclusions hold; There is an effect of genre on popularity. Let's visualise the results to see where this effect lies.

In [178...

```
coefs = model.params

plot_data = coefs.to_frame('coef').sort_values('coef')
plot_data.index = plot_data.index.str.extract(r'\[T\.(.*)\]')[0].to_list()
plot_data = plot_data.reset_index().rename(columns={'index': 'genres'})

plt.figure(figsize=(8, 15))
sns.pointplot(data=plot_data, x='coef', y='genres', linestyle='none', markers='.')
plt.axvline(0, color="black", lw=1);
```


iranian
romance
latin
detroit-techno
chicago-house
classical
jazz
grindcore
kids
idm
honky-tonk
country
rock
soul
tango
breakbeat
reggae
party
happy
black-metal
dance
reggaeton
alternative
afrobeat
opera
comedy
j-idol
latino
bluegrass
study
heavy-metal
hardstyle
j-dance
drum-and-bass
german
power-pop
disney
salsa
goth
new-age
guitar
malay
industrial
blues
show-tunes
rockabilly
death-metal
funk
club
dancehall
disco
alt-rock
minimal-techno
house
rock-n-roll
trip-hop
cantopop
edm
children
sleep
ska
electro
hardcore
synth-pop
r-n-b
swedish
trance
hip-hop
songwriter
singer-songwriter
groove
spanish
folk
punk-rock
punk
samba
indie
techno
dub
indie-pop
j-rock
turkish
mpb
garage
french
j-pop
gospel
forro
world-music
dubstep
hard-rock
psych-rock
metalcore
metal
british
ambient
pagode
electronic
brazil
deep-house
mandopop
piano
progressive-house
pop
sertanejo
emo
anime
indian
grunge
sad
chill
k-pop
pop-film
genres
−40
−30
−20
−10
0
10
coef

The top 5 strongest effects are:

Positive effect on popularity:

- pop-film
- k-pop
- chill
- sad
- grunge

Negative effect on popularity:

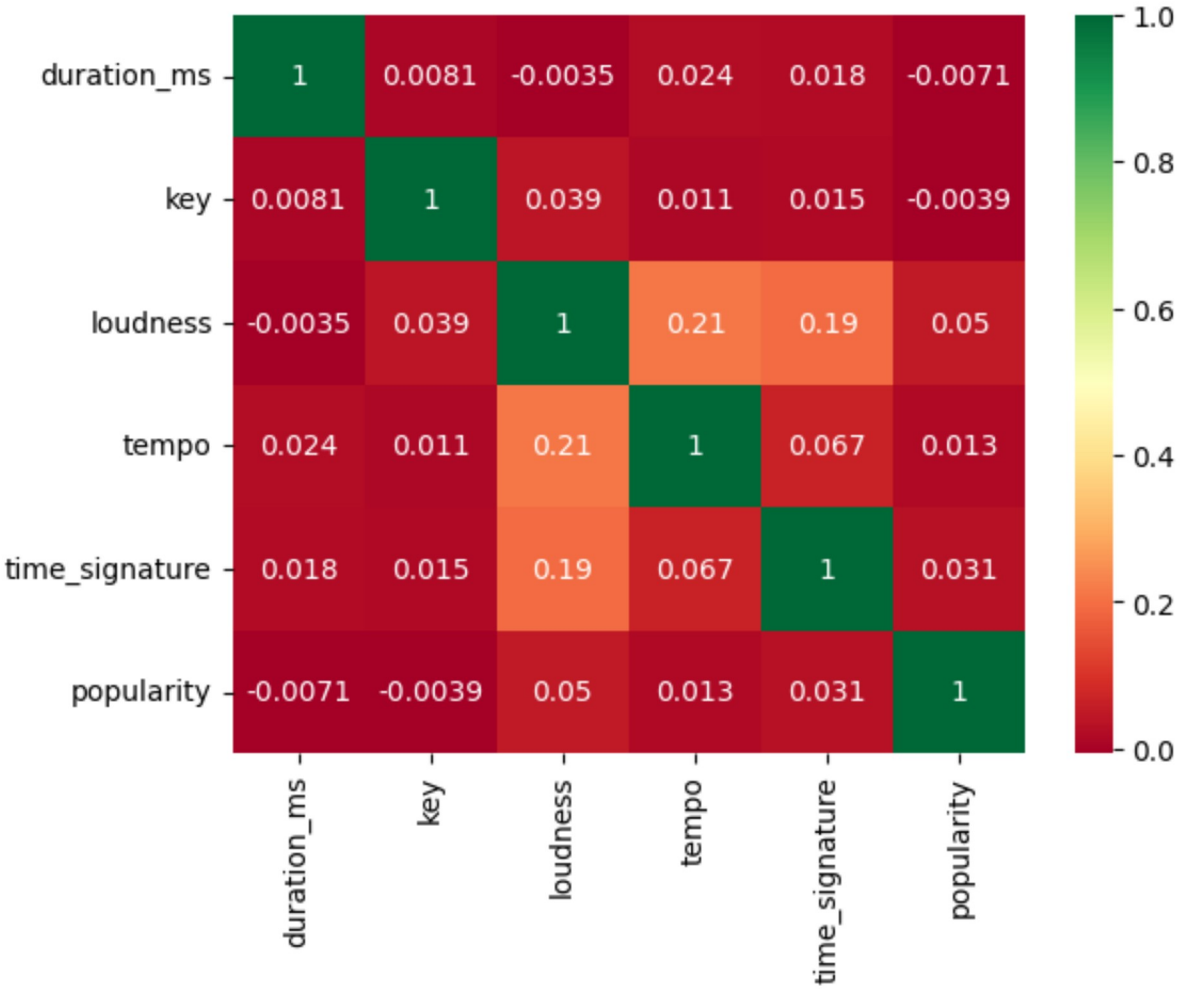
- iranian
- romance
- latin
- detroit-techno
- chicago-house

Do Music Properties Like Key and Tempo Impact Track Popularity

The musical properties under consideration are: ["duration_ms", "key", "loudness", "tempo", "time_signature"]. Again, we first look visually if there are any obvious correlations.

In [179…
```
feature_cols = ["duration_ms", "key", "loudness", "tempo", "time_signature"]
corr = data[feature_cols + ['popularity']].corr()
sns.heatmap(corr, annot=True, cmap="RdYlGn")
```

Out[179… <Axes: >



Hmmm, no correlations present. But what about using some feature engineering to see if there are higher order effects or maybe combined effects. Let's check for the following:

- Polynomial features
- Multiplicative features (interaction features)

Polynomial features and interaction features

For this we can simply use `PolynomialFeatures` from scikit-learn.

In [180…

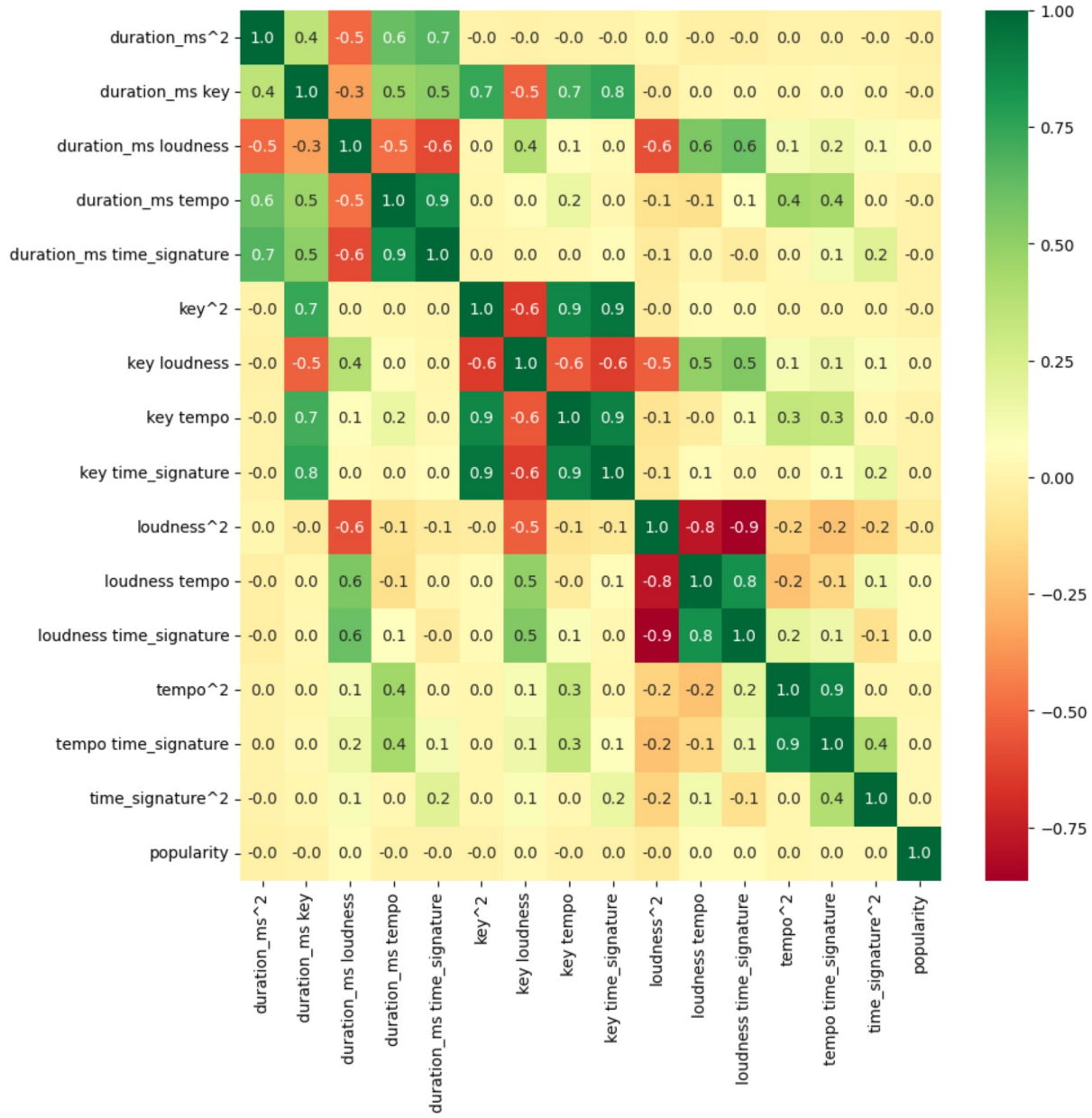
```
from sklearn.preprocessing import PolynomialFeatures

pf = PolynomialFeatures(degree=2)
pf.fit(data[feature_cols])
feat_cols = pf.get_feature_names_out(input_features=feature_cols)
feat_array = pf.transform(data[feature_cols])
feat_poly = pd.DataFrame(feat_array, columns=feat_cols).drop(columns=feature_cols+[

feat_poly['popularity'] = data['popularity']

plt.figure(figsize=(10, 10))
sns.heatmap(feat_poly.corr(), annot=True, fmt='.1f', cmap="RdYlGn")
```

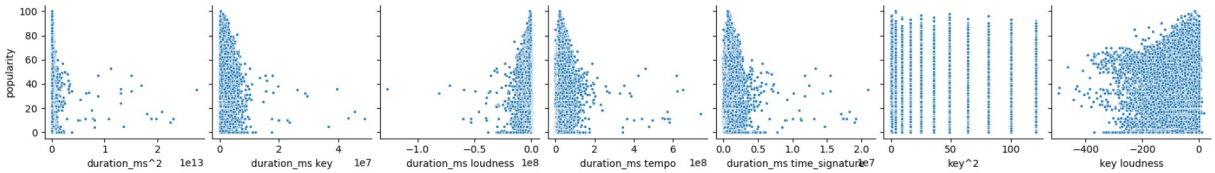
Out[180… <Axes: >

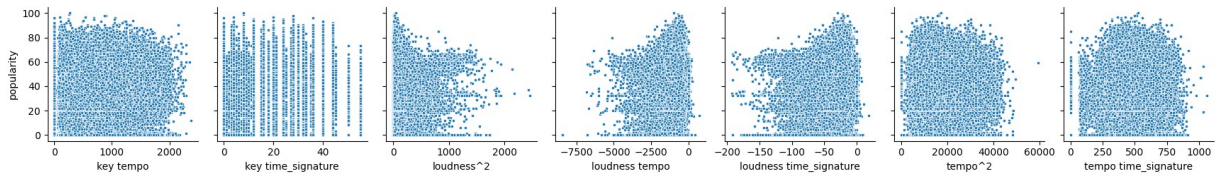


There are actually some strong correlations between some of the combined features. But nothing for the popularity, to illustrate explicitly:

In [181…
```
sns.pairplot(data=feat_poly, x_vars=feat_poly.columns[0:7], y_vars=['popularity'],
sns.pairplot(data=feat_poly, x_vars=feat_poly.columns[7:-2], y_vars=['popularity'],
```

Out[181… <seaborn.axisgrid.PairGrid at 0x166bfa2fc80>





So for this hypothesis we can accept our null hypothesis that there is no significant effect of musical properties on popularity. Even combined features show no effect on this pair of features/target.

5. Conclusions

When considering this dataset with evenly distributed tracks across genres, we observe that genre has a strong effect on popularity. More esotheric measures such as the Audio Features seem to have only a minor effect (though an additional analysis using interaction features might be interesting). Finally, there is no effect on popularity from technical musical properties such as key, tempo and length.

This is to be expected from a platform like spotify, where people are generally not concerned with the technical details of the music they listen to, but more in the general feel of the music (Audio Features) or only in the type of genre.