Machine Learning Final Project Report

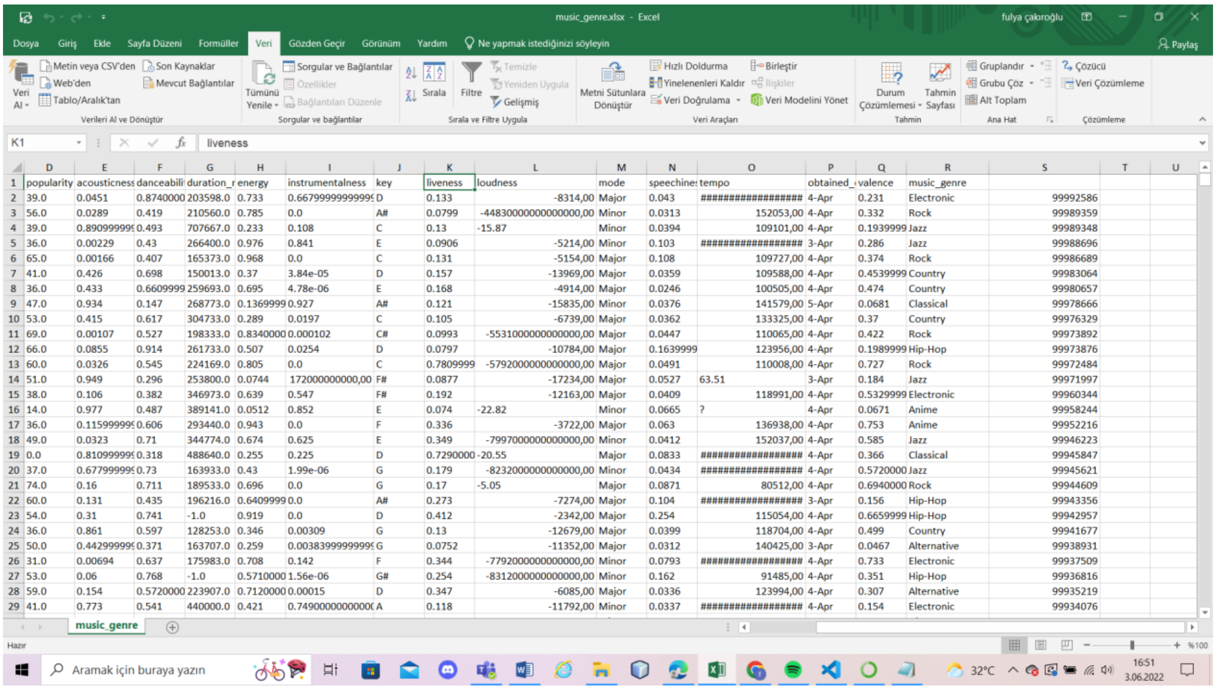
Guess The Genre

We download the music_genre.csv . It is converted text file that uses commas(,) to seperate values. Each line of the file is a data record.

We want to converting excel file, for changing order of datas.

We choosed columns and clicked data and text to columns on the above. After that we clicked comma and finished button. We want to ordering datas randomly .For this, we created new columns and wrote this instructions;

=rastgelearada(0;10000000) Each columns take a value between 0 and 10000000.After that, we choosed sort from biggest to smallest, So we got randomly sorted rows.



	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	popularity	acousticness	danceabili	duration_r	energy	instrumentalness	key	liveness	loudness	mode	speechine	tempo	obtained_	valence	music_genre	
2	39.0	0.0451	0.8740000	203598.0	0.733	0.66799999999999	D	0.133	-8314,00	Major	0.043	##################	4-Apr	0.231	Electronic	99992586
3	56.0	0.0289	0.419	210560.0	0.785	0.0	A#	0.0799	-4483000000000000,00	Minor	0.0313	152053,00	4-Apr	0.332	Rock	99989359
4	39.0	0.89099999	0.493	707667.0	0.233	0.108	C	0.13	-15.87	Minor	0.0394	109101,00	4-Apr	0.1939999	Jazz	99989348
5	36.0	0.00229	0.43	266400.0	0.976	0.841	E	0.0906	-5214,00	Minor	0.103	##################	3-Apr	0.286	Jazz	99988696
6	65.0	0.00166	0.407	165373.0	0.968	0.0	C	0.131	-5154,00	Major	0.108	109727,00	4-Apr	0.374	Rock	99986689
7	41.0	0.426	0.698	150013.0	0.37	3.84e-05	D	0.157	-13969,00	Major	0.0359	109588,00	4-Apr	0.4539999	Country	99983064
8	36.0	0.433	0.6609999	259693.0	0.695	4.78e-06	E	0.168	-4914,00	Major	0.0246	100505,00	4-Apr	0.474	Country	99980657
9	47.0	0.934	0.147	268773.0	0.1369999	0.927	A#	0.121	-15835,00	Minor	0.0376	141579,00	5-Apr	0.0681	Classical	99978666
10	53.0	0.415	0.617	304733.0	0.289	0.0197	C	0.105	-6739,00	Major	0.0362	133325,00	4-Apr	0.37	Country	99976329
11	69.0	0.00107	0.527	198333.0	0.8340000	0.000102	C#	0.0993	-5531000000000000,00	Major	0.0447	110065,00	4-Apr	0.422	Rock	99973892
12	66.0	0.0855	0.914	261733.0	0.507	0.0254	D	0.0797	-10784,00	Major	0.1639999	123956,00	4-Apr	0.1989999	Hip-Hop	99973876
13	60.0	0.0326	0.545	224169.0	0.805	0.0	C	0.7809999	-5792000000000000,00	Major	0.0491	110008,00	4-Apr	0.727	Rock	99972484
14	51.0	0.949	0.296	253800.0	0.0744	172000000000,00	F#	0.0877	-17234,00	Major	0.0527	63.51	3-Apr	0.184	Jazz	99971997
15	38.0	0.106	0.382	346973.0	0.639	0.547	F#	0.192	-12163,00	Major	0.0409	118991,00	4-Apr	0.5329999	Electronic	99960344
16	14.0	0.977	0.487	389141.0	0.0512	0.852	E	0.074	-22.82	Minor	0.0665	?	4-Apr	0.0671	Anime	99958244
17	36.0	0.11599999	0.606	293440.0	0.943	0.0	F	0.336	-3722,00	Major	0.063	136938,00	4-Apr	0.753	Anime	99952216
18	49.0	0.0323	0.71	344774.0	0.674	0.625	E	0.349	-7997000000000000,00	Minor	0.0412	152037,00	4-Apr	0.585	Jazz	99946223
19	0.0	0.81099999	0.318	488640.0	0.255	0.225	D	0.7290000	-20.55	Major	0.0833	##################	4-Apr	0.366	Classical	99945847
20	37.0	0.67799999	0.73	163933.0	0.43	1.99e-06	G	0.179	-8232000000000000,00	Minor	0.0434	##################	4-Apr	0.5720000	Jazz	99945621
21	74.0	0.16	0.711	189533.0	0.696	0.0	G	0.17	-5.05	Major	0.0871	80512,00	4-Apr	0.6940000	Rock	99944609
22	60.0	0.131	0.435	196216.0	0.6409999	0.0	A#	0.273	-7274,00	Major	0.104	##################	3-Apr	0.156	Hip-Hop	99943356
23	54.0	0.31	0.741	-1.0	0.919	0.0	D	0.412	-2342,00	Major	0.254	115054,00	4-Apr	0.6659999	Hip-Hop	99942957
24	36.0	0.861	0.597	128253.0	0.346	0.00309	G	0.13	-12679,00	Major	0.0399	118704,00	4-Apr	0.499	Country	99941677
25	50.0	0.44299999	0.371	163707.0	0.259	0.00383999999999	G	0.0752	-11352,00	Major	0.0312	140425,00	3-Apr	0.0467	Alternative	99938931
26	31.0	0.00694	0.637	175983.0	0.708	0.142	F	0.344	-7792000000000000,00	Minor	0.0793	##################	4-Apr	0.733	Electronic	99937509
27	53.0	0.06	0.768	-1.0	0.5710000	1.56e-06	G#	0.254	-8312000000000000,00	Minor	0.162	91485,00	4-Apr	0.351	Hip-Hop	99936816
28	59.0	0.154	0.5720000	223907.0	0.7120000	0.00015	D	0.347	-6085,00	Major	0.0336	123994,00	4-Apr	0.307	Alternative	99935219
29	41.0	0.773	0.541	440000.0	0.421	0.74900000000000	A	0.118	-11792,00	Minor	0.0337	##################	4-Apr	0.154	Electronic	99934076

Our file looks like this.

We opened jupyter notebook. We used this platform.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from warnings import filterwarnings as filt

Firstly these are our libraries. numpy is necessary for linear gebra, pandas is necessary for data processing. We used excel(.xlsx).

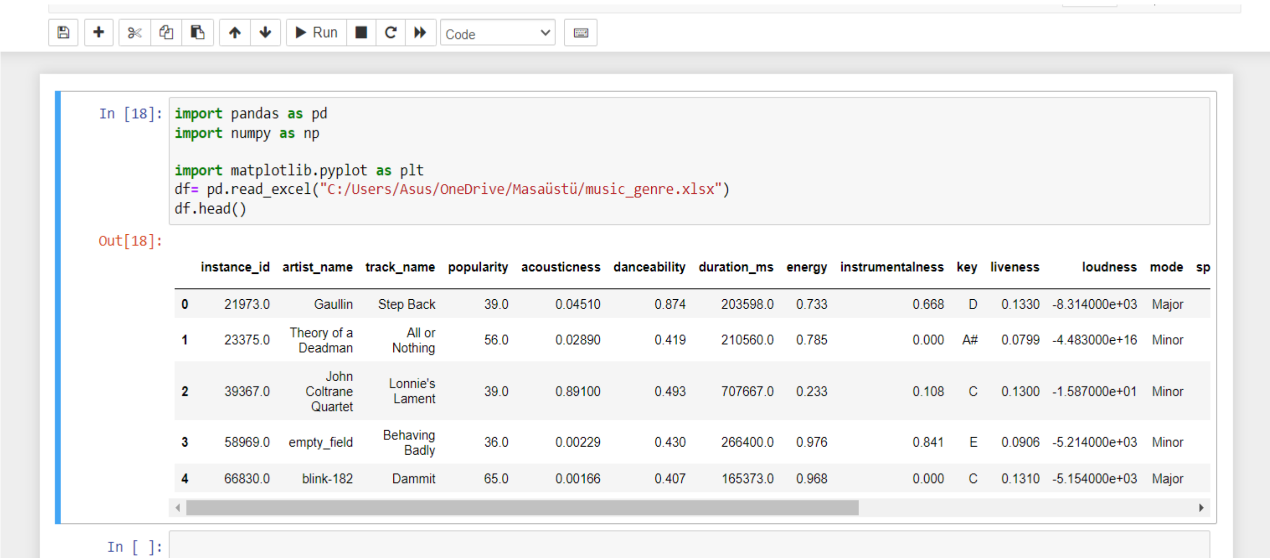
Matplotlib is important for drawing graph.

We clicked right side of mouse and looked properties, because we want finding path that file.

df= pd.read_excel("C:/Users/Asus/OneDrive/Masaüstü/music_genre.xlsx")

df.head()

This processing provide looking header.



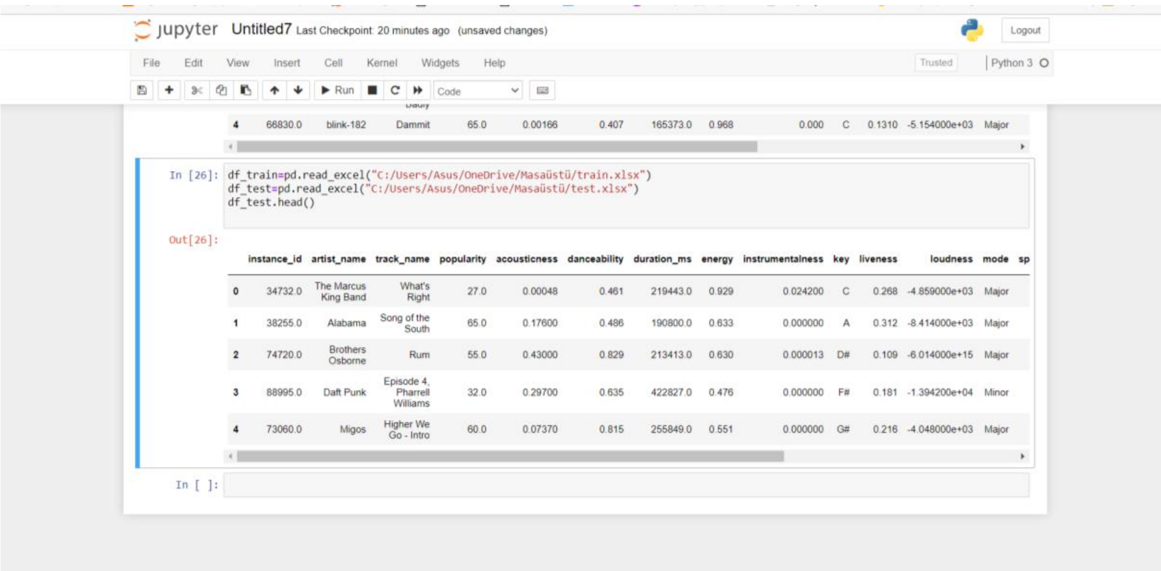
In [18]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
df= pd.read_excel("C:/Users/Asus/OneDrive/Masaüstü/music_genre.xlsx")
df.head()
```

Out[18]:

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	sp
0	21973.0	Gaullin	Step Back	39.0	0.04510	0.874	203598.0	0.733	0.668	D	0.1330	-8.314000e+03	Major	
1	23375.0	Theory of a Deadman	All or Nothing	56.0	0.02890	0.419	210560.0	0.785	0.000	A#	0.0799	-4.483000e+16	Minor	
2	39367.0	John Coltrane Quartet	Lonnie's Lament	39.0	0.89100	0.493	707667.0	0.233	0.108	C	0.1300	-1.587000e+01	Minor	
3	58969.0	empty_field	Behaving Badly	36.0	0.00229	0.430	266400.0	0.976	0.841	E	0.0906	-5.214000e+03	Minor	
4	66830.0	blink-182	Dammit	65.0	0.00166	0.407	165373.0	0.968	0.000	C	0.1310	-5.154000e+03	Major	

In [ ]:

We splitted two part our excel file. 48000. column and we train set to the part from 0 to 48000. row. We choosed other row and copy the whole part. We created test.xlsx and pasted this file.



jupyter Untitled7 Last Checkpoint: 20 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

4	66830.0	blink-182	Dammit	65.0	0.00166	0.407	165373.0	0.968	0.000	C	0.1310	-5.154000e+03	Major	

In [26]:
```
df_train=pd.read_excel("C:/Users/Asus/OneDrive/Masaüstü/train.xlsx")
df_test=pd.read_excel("C:/Users/Asus/OneDrive/Masaüstü/test.xlsx")
df_test.head()
```

Out[26]:

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	sp
0	34732.0	The Marcus King Band	What's Right	27.0	0.00048	0.461	219443.0	0.929	0.024200	C	0.268	-4.859000e+03	Major	
1	38255.0	Alabama	Song of the South	65.0	0.17600	0.486	190800.0	0.633	0.000000	A	0.312	-8.414000e+03	Major	
2	74720.0	Brothers Osborne	Rum	55.0	0.43000	0.829	213413.0	0.630	0.000013	D#	0.109	-6.014000e+15	Major	
3	88995.0	Daft Punk	Episode 4, Pharrell Williams	32.0	0.29700	0.635	422827.0	0.476	0.000000	F#	0.181	-1.394200e+04	Minor	
4	73060.0	Migos	Higher We Go - Intro	60.0	0.07370	0.815	255849.0	0.551	0.000000	G#	0.216	-4.048000e+03	Major	

In [ ]:

df.shape

When we run this code, output was rows’s and columns’s count. Our output is (500005,19). Normally, we have got 18 columns but we took randomly in each row, add a column as a randomsayi.

null_feats = pd.DataFrame(df.isnull().sum(), columns = ['nans']).sort_values('nans', ascending = False)

null_feats['nans %'] = np.round(df.isnull().sum() / df.shape[0], 2)

null_feats.head()

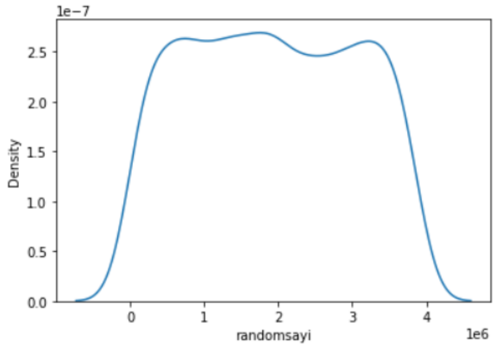
This process’s purpose is handling null and NaN values.

.dataframe provides creating table and .isnull() detecting missing empty values. In most cases, decimal numbers need to be rounded. I perform these roundings with the round function in Python.

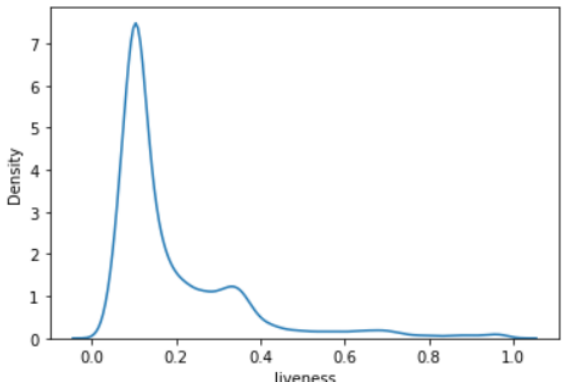
```
In [26]: df.shape
Out[26]: (50005, 19)

In [27]: null_feats = pd.DataFrame(df.isnull().sum(), columns = ['nans']).sort_values('nans', ascending = False)
         null_feats['nans %'] = np.round(df.isnull().sum() / df.shape[0], 2)

         null_feats.head()
Out[27]:
```

	nans	nans %
randomsayi	47998	0.96
liveness	65	0.00
music_genre	65	0.00
valence	65	0.00
obtained_date	65	0.00


```
In [33]: sns.kdeplot(df[nulls[0]])
Out[33]: <AxesSubplot:xlabel='randomsayi', ylabel='Density'>
```




```
In [29]: sns.kdeplot(df[nulls[1]])
Out[29]: <AxesSubplot:xlabel='liveness', ylabel='Density'>
```



We wrote sns because we used seaborn library in Python. Seaborn is a statistical Python data visualization library based on the Matplotlib library.

The KDE Plot (Kernel Density Estimate) is used to visualize the probability densities of a continuous value.

For instance, in last example X- axis is liveness and y-axis is Density.Plot is null values density in liveness column.

We used groupby(), because it is The Pandas package is a Python package for data analysis and manipulation. This package makes it easy to read and create files in various formats (such as Excel, Csv, Txt). At the same time, thanks to the dataframes created with the pandas package, it is possible to keep data in different formats (number, text, date) together, to process this data and to perform simple analysis.

```
In [38]: df.groupby('artist_name')['popularity'].mean().sort_values(ascending = False).head(10)
```

```
Out[38]: artist_name
Duki                82.000000
Heuss L'enfoirÃ©    81.000000
NSG                 81.000000
Coolio              80.000000
Danny Ocean         80.000000
Ben E. King         79.000000
Snow Patrol         78.000000
Post Malone         77.233333
Kevin Roldan        77.000000
Jet                 77.000000
Name: popularity, dtype: float64
```

```
In [67]: df[df['artist_name'] == 'Daft Punk']
```

Out[67]:

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode
3	88995.0	Daft Punk	Episode 4, Pharrell Williams	32.0	0.2970	0.635	422827.0	0.476	0.000	F#	0.181	-1.394200e+04	Minor
1455	77258.0	Daft Punk	Disc Wars	44.0	0.4330	0.543	251440.0	0.480	0.886	D	0.117	-1.455100e+16	Major
1905	57851.0	Daft Punk	One More Time	74.0	0.0193	0.611	320357.0	0.697	0.000	D	0.332	-8.618000e+03	Major

If we want to looking at Daft Punk’s all columns, we should use the code which is above.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier as rfc
from sklearn.inspection import permutation_importance
import eli5
from eli5.sklearn import PermutationImportance
from pdpbox import pdp
from sklearn.ensemble import RandomForestClassifier as rfc
import shap
```


```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier as rfc
from sklearn.inspection import permutation_importance
import eli5
from eli5.sklearn import PermutationImportance
from pdpbox import pdp
from sklearn.ensemble import RandomForestClassifier as rfc
import shap
```


train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.

```
In [16]: conda install -c conda-forge pdpbox

Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\Asus\anaconda3

  added / updated specs:
Note: you may need to restart the kernel to use updated packages.

    - pdpbox


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    pdpbox-0.2.1               |     pyhd8ed1ab_0        22.2 MB  conda-forge
    ------------------------------------------------------------
                                           Total:        22.2 MB

The following NEW packages will be INSTALLED:

  pdpbox             conda-forge/noarch::pdpbox-0.2.1-pyhd8ed1ab_0



Downloading and Extracting Packages

pdpbox-0.2.1         | 22.2 MB   |            |   0%
pdpbox-0.2.1         | 22.2 MB   |            |   0%
pdpbox-0.2.1         | 22.2 MB   |            |   1%
pdpbox-0.2.1         | 22.2 MB   | 1          |   2%
pdpbox-0.2.1         | 22.2 MB   | 4          |   4%
pdpbox-0.2.1         | 22.2 MB   | 7          |   8%
pdpbox-0.2.1         | 22.2 MB   | #1         |  11%
pdpbox-0.2.1         | 22.2 MB   | #5         |  15%
pdpbox-0.2.1         | 22.2 MB   | #9         |  19%
pdpbox-0.2.1         | 22.2 MB   | ##3        |  23%
pdpbox-0.2.1         | 22.2 MB   | ##6        |  27%
pdpbox-0.2.1         | 22.2 MB   | ####4      |  45%
pdpbox-0.2.1         | 22.2 MB   | #####2     |  52%
```

In the picture above, you can see how we import the PDPbox.

PDPbox is a partial dependence plot toolbox written in Python. The goal is to visualize the impact of certain features towards model prediction for any supervised learning algorithm.

Than we import the shap.

SHAP is a mathematical method to explain the predictions of machine learning models

```
def permImp(val_x, val_y):
    val_x = val_x.select_dtypes(exclude = 'object')
    model = rfc(n_estimators = 100,random_state = 123).fit(val_x,val_y)
    perm = PermutationImportance(model).fit(val_x, val_y)
    return eli5.show_weights(perm, feature_names = val_x.columns.tolist())
```

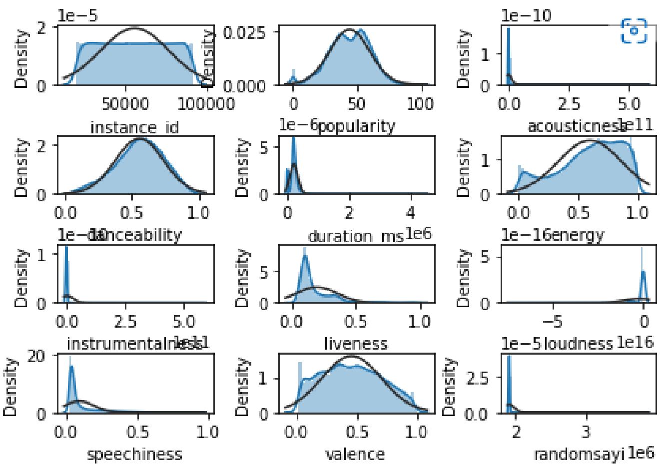
We construct the permImp function to visualize the features and weights.

In the picture below we can see the weights and features;

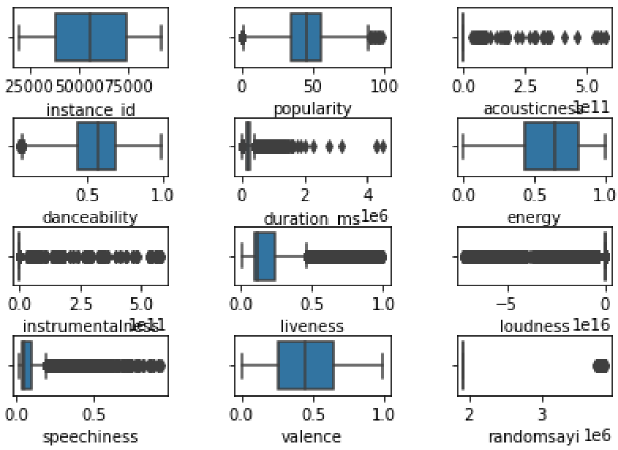
```python
permImp(val_x,val_y)
```

Weight	Feature
0.0120 ± 0.0073	randomsayi
0.0110 ± 0.0060	loudness
0.0105 ± 0.0102	liveness
0.0065 ± 0.0068	speechiness
0.0065 ± 0.0075	acousticness
0.0060 ± 0.0051	valence
0.0060 ± 0.0040	popularity
0.0055 ± 0.0020	energy
0.0055 ± 0.0049	instance_id
0.0045 ± 0.0058	instrumentalness
0.0045 ± 0.0058	danceability
0.0025 ± 0.0032	duration_ms
0 ± 0.0000	Solo

```python
feats = [c for c in train_x.select_dtypes(exclude = 'object').columns if train_x[c].nunique() >= 10
plot(train_x[feats], [4,3])
```

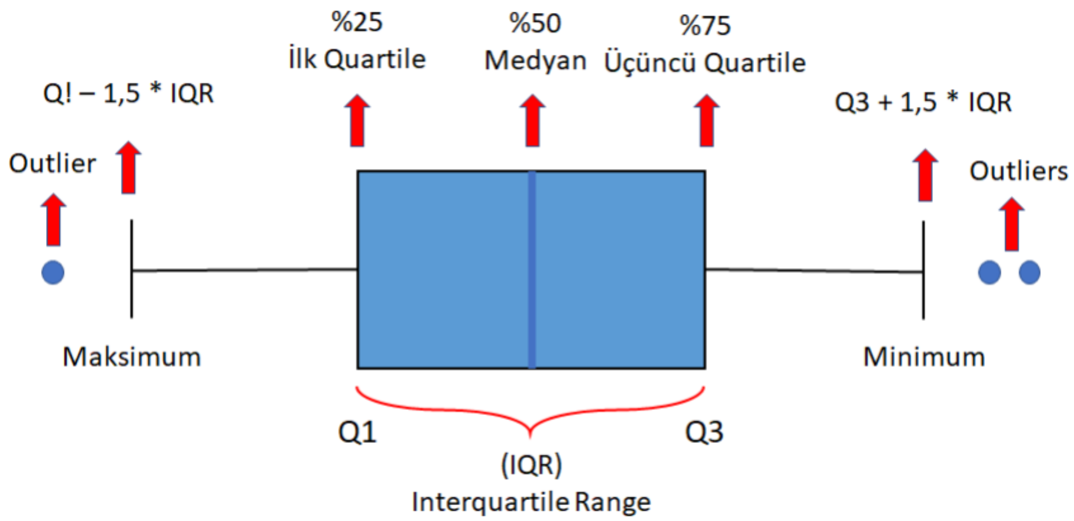



```python
plot(train_x[feats], [4,3], 'box')
```



Our graphs in above show dennsity of each columns.

Our graphs in below are box graphs. A box plot is a standard way of showing the distribution of data based on a five-figure summary (“minimum,” first quartile (Q1), median, third quartile (Q3), and “maximum”).



In [408]:
```
train_artist_name, val_artist_name = train_x['artist_name'], val_x['artist_name']
train_song_name, val_song_name = train_x['track_name'], val_x['track_name']
feats_to_drop = ['artist_name', 'track_name']
train_x = train_x.drop(feats_to_drop, axis = 1)
val_x = val_x.drop(feats_to_drop, axis = 1)
```

In [409]:
```
train_x.head()
```

Out[409]:

	instance_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	
0	21973.0	39.0	0.04510	0.874	203598.0	0.733	0.668	D	0.1330	-8.314000e+03	Major	0.0430	1250339999999
1	23375.0	56.0	0.02890	0.419	210560.0	0.785	0.000	A#	0.0799	-4.483000e+16	Minor	0.0313	1
2	39367.0	39.0	0.89100	0.493	707667.0	0.233	0.108	C	0.1300	-1.587000e+01	Minor	0.0394	1
3	58969.0	36.0	0.00229	0.430	266400.0	0.976	0.841	E	0.0906	-5.214000e+03	Minor	0.1030	1999640000000
4	66830.0	65.0	0.00166	0.407	165373.0	0.968	0.000	C	0.1310	-5.154000e+03	Major	0.1080	1

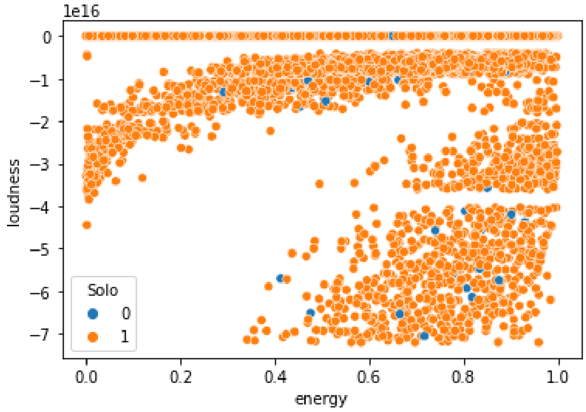
Train_artist_name equal to artist name in train_x. val_artist_name equal to artist name in val_x.

We delete rows [artist_name,track_name] in train_x.axis=1 is mean row. If we write x=0 ,this meaning is index.

After that,We show train_x table.

In [410]:
```
sns.scatterplot(data = train_x, x = 'energy', y ='loudness', hue = 'Solo')
```

Out[410]:
```
<AxesSubplot:xlabel='energy', ylabel='loudness'>
```



Scatter Plot

s: Determines the size of the circles in the chart.

c: Determines the color of the circles in the chart.

edgecolors: Specifies the color of the line outside the circle.

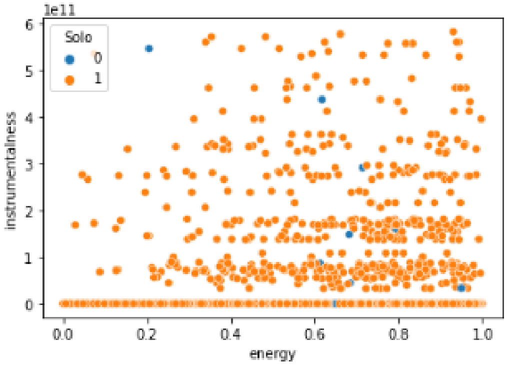
linewidths: Determines the thickness of the line outside the circle.

alpha: Specifies the transparency of the circle. It takes a value between 0-1.

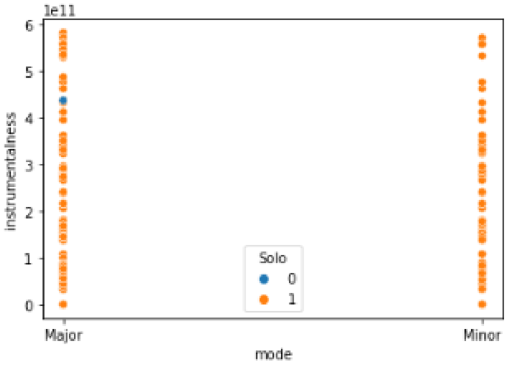
cmap: Creates a color map. It is used according to the data.

We have show graphs of most features.

In [411]:
```
sns.scatterplot(data = train_x, x = 'energy', y ='instrumentalness', hue = 'Solo')
```
Out[411]: <AxesSubplot:xlabel='energy', ylabel='instrumentalness'>

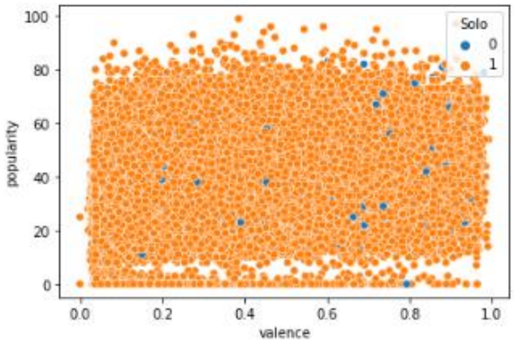


In [412]:
```
sns.scatterplot(data = train_x, x = 'mode', y ='instrumentalness', hue = 'Solo')
```
Out[412]: <AxesSubplot:xlabel='mode', ylabel='instrumentalness'>

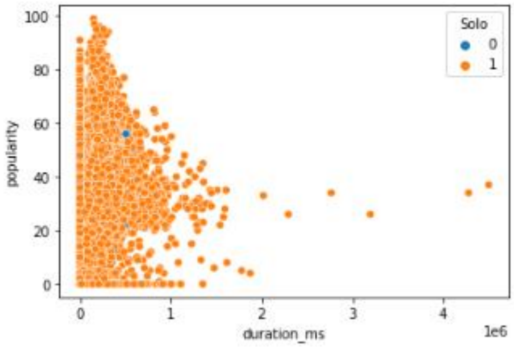


Mode columns contain two different value as major and minor. So,our graph scattered sideways.

```
In [415]: sns.scatterplot(data = train_x, x = 'valence', y ='popularity', hue = 'Solo')

Out[415]: <AxesSubplot:xlabel='valence', ylabel='popularity'>
```




```
In [416]: sns.scatterplot(data = train_x, x = 'duration_ms', y ='popularity', hue = 'Solo')

Out[416]: <AxesSubplot:xlabel='duration_ms', ylabel='popularity'>
```



We can get more graphics by playing with x and y. X and Y are our columns's names.

```
from sklearn.linear_model import LogisticRegression as lrr
from sklearn.ensemble import RandomForestClassifier as rfc
from sklearn.naive_bayes import GaussianNB as gnb
from sklearn.svm import SVC
from xgboost import XGBRFClassifier as xgb

from sklearn.model_selection import cross_val_score as cvs, GridSearchCV as gscv, StratifiedKFold as skf
from sklearn.pipeline import Pipeline

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.preprocessing import StandardScaler as ss, MinMaxScaler as mms, RobustScaler as rs
```

After, we import model libraries.

Support Machine Vector(Svm): "Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

Logistic Regression: The logistic regression statistic modeling technique is used when we have a binary outcome variable. Linear regression statistical model is used to predict continuous outcome variables, whereas logistic regression predicts categorical outcome variables. Linear regression model regression line is highly susceptible to outliers. So, it will not be appropriate for logistic regression.

K-Nearest Neighbors: The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm that can be used to solve both classification and regression problems. Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

Decision Tree: Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset. Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

```python
def best_model(xt, yt, scaler = None):
    models = [lrr(), SVC(), rfc(), gnb(), xgb()]
    names = ['logistic regression','svm','random forest clf', 'naive bayes', 'xgboost']
    scores = []
    for model in models:
        if scaler == 'std':
            model = Pipeline(steps = [('std',ss()),('model',model)])
        elif scaler == 'robust':
            model = Pipeline(steps = [('robust',rs()),('model',model)])
        elif scaler == 'mms':
            model = Pipeline(steps = [('mms',mms()),('model',model)])
        cv = skf(n_splits = 2, shuffle = True, random_state = True)
        score = cvs(model, cv = cv, X = xt, y = yt, scoring = 'f1_micro').mean()

        scores.append(score)
    return pd.DataFrame(score, index = names, columns = ['f1_score']).sort_values('f1_score', ascending = True)

def get_score(xt, yt, model = lrr(), scaler = None):
    if scaler == 'std':
        model = Pipeline(steps = [('std',ss()),('model',model)])
    elif scaler == 'robust':
        model = Pipeline(steps = [('robust',rs()),('model',model)])
    elif scaler == 'mms':
        model = Pipeline(steps = [('mms',mms()),('model',model)])
    cv = skf(n_splits = 2, shuffle = True, random_state = True)
    auc = cvs(model, cv = cv, X = xt, y = yt).mean()
    print(f"Model score :==> {auc}")

def gridCv(xt, yt, model, params, scaler = None):
    if scaler == 'std':
        model = Pipeline(steps = [('std',ss()),('model',model)])
    elif scaler == 'robust':
        model = Pipeline(steps = [('robust',rs()),('model',model)])
    elif scaler == 'mms':
        model = Pipeline(steps = [('mms',mms()),('model',model)])
    skcv = skf(n_splits = 2, shuffle = True, random_state = True)
    cv = gscv(model, param_grid = params, cv = skcv , return_train_score = True)
    cv.fit(xt,yt)
    results = pd.DataFrame(cv.cv_results_).sort_values('mean_test_score', ascending = False)
    results = results[['mean_test_score','mean_train_score','params']]
    best_params = cv.best_params_
    best_est = cv.best_estimator_
    return best_est, best_params, results

def clf_report(yt, pred):
    print(classification_report(yt, pred))
```

In here,We write functions of models.We try n_splits=0 or other values and get error or not working correctly, but n_split=2 is our optimum value.

```
null_feats = pd.DataFrame(df.isnull().sum(), columns = ['nans']).sort_values('nans', ascending = False)
null_feats['nans %'] = np.round(df.isnull().sum() / df.shape[0], 2)

df = df.dropna()
print(df.isnull().sum().sort_values(ascending=False))
```

```
instance_id         0
artist_name         0
randomsayi          0
music_genre         0
valence             0
obtained_date       0
tempo               0
speechiness         0
mode                0
loudness            0
liveness            0
key                 0
instrumentalness    0
energy              0
duration_ms         0
danceability        0
acousticness        0
popularity          0
track_name          0
Solo                0
dtype: int64
```

We find out the number of missing values in df and sort them in descending order. We have NaN values, and we delete them with the dropna() method.

In [424]:
```
best_model(train_x, train_y)
```
Out[424]:

	f1_score
logistic regression	NaN
svm	NaN
random forest clf	NaN
naive bayes	NaN
xgboost	NaN

In [420]:
```
best_model(train_x, train_y, 'std')
```
Out[420]:

	f1_score
logistic regression	NaN
svm	NaN
random forest clf	NaN
naive bayes	NaN
xgboost	NaN

In [201]:
```
best_model(train_x, train_y, 'robust')
```
Out[201]:

	f1_score
logistic regression	NaN
svm	NaN
random forest clf	NaN
naive bayes	NaN
xgboost	NaN

In [202]:
```
best_model(train_x, train_y, 'mms')
```
Out[202]:

	f1_score
logistic regression	NaN
svm	NaN
random forest clf	NaN
naive bayes	NaN
xgboost	NaN

We try applying model functions,but our dataset is containing NaN values,our scores equal NaN.

REFERENCES:

https://medium.com/bili%C5%9Fim-hareketi/veri-bilimi-i%CC%87%C3%A7in-temel-python-k%C3%BCt%C3%BCphaneleri-2-pandas-dcc12ae01b7d

https://medium.com/datarunner/veri-biliminde-normal-da%C4%9F%C4%B1l%C4%B1m%C4%B1n-python-%C3%BCzerinden-g%C3%B6rselle%C5%9Ftirilmesi-ve-yorumlanmas%C4%B1-histogram-8381d12b85b9#:~:text=Box%20Plot%20(Kutu%20Grafik)&text=Medyan%20(%50%20quartile)%3A%20De%C4%9Fi%C5%9Fkene,minimumun%20ortas%C4%B1ndaki%20de%C4%9Fer%20olarak%20a%C3%A7%C4%B1klanabilir.

https://www.kaggle.com/code/muhammedjaabir/music-genre-clf/notebook

https://medium.com/datarunner/matplotlib-k%C3%BCt%C3%BCphanesi-i%CC%87le-scatter-plot-9b8c181fc9ad#:~:text=s%20%3A%20Grafikteki%20yuvarlaklar%C4%B1n%20boyutunu%20belirler,Yuvarla%C4%9F%C4%B1n%20d%C4%B1%C5%9F%C4%B1ndaki%20%C3%A7izginin%20rengini%20belirler
.

https://colab.research.google.com/github/Arize-ai/client_python/blob/main/arize/examples/tutorials/Arize_Tutorials/SHAP/SHAP_Tutorial_Base_Version.ipynb

https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

https://teknoloji.org/seaborn-kutuphanesi-nedir-nasil-kullanilir/

FULYA ÇAKIROĞLU 18315027

DOĞUKAN SEZER ÖZÇELİK 180315018