«*In The Name Of GOD*»

[HW-02-Report]

[MACHINE LEARNING]

Hasan Masroor | [403131030] | November 11, 2024

"فهرست مطالب تمرين 02"

Problem 1: Predicting House Price Using Stacked Regression

.1

```
Train = pd.read_csv('train.csv')
Test = pd.read_csv('test.csv')

Train.head()
```

برای این قسمت با استفاده از کد بالا 5 نمونه اول را در زیر نمایش دادیم (یا با استفاده از tail(). نیز 5 نمونه آخر را می‌توانیم نمایش دهیم):

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley
0	1	60	RL	65.0	8450	Pave	NaN
1	2	20	RL	80.0	9600	Pave	NaN
2	3	60	RL	68.0	11250	Pave	NaN
3	4	70	RL	60.0	9550	Pave	NaN
4	5	60	RL	84.0	14260	Pave	NaN

5 rows × 81 columns

LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature
Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
IR1	Lvl	AllPub	...	0	NaN	NaN	NaN

MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	2	2008	WD	Normal	208500
0	5	2007	WD	Normal	181500
0	9	2008	WD	Normal	223500
0	2	2006	WD	Abnorml	140000
0	12	2008	WD	Normal	250000

.2

برای این بخش، بهتر است ابتدا مجموعه داده‌های تست و آموزش را با هم ترکیب کنیم تا از یکنواختی مراحل پردازش داده‌ها برای هر دو مجموعه اطمینان حاصل شود. همچنین، مجموعه داده دارای مقادیر تهی زیادی است، بنابراین ستون‌هایی که تعداد زیادی مقدار تهی دارند را حذف می‌کنیم.

قبل از حذف:

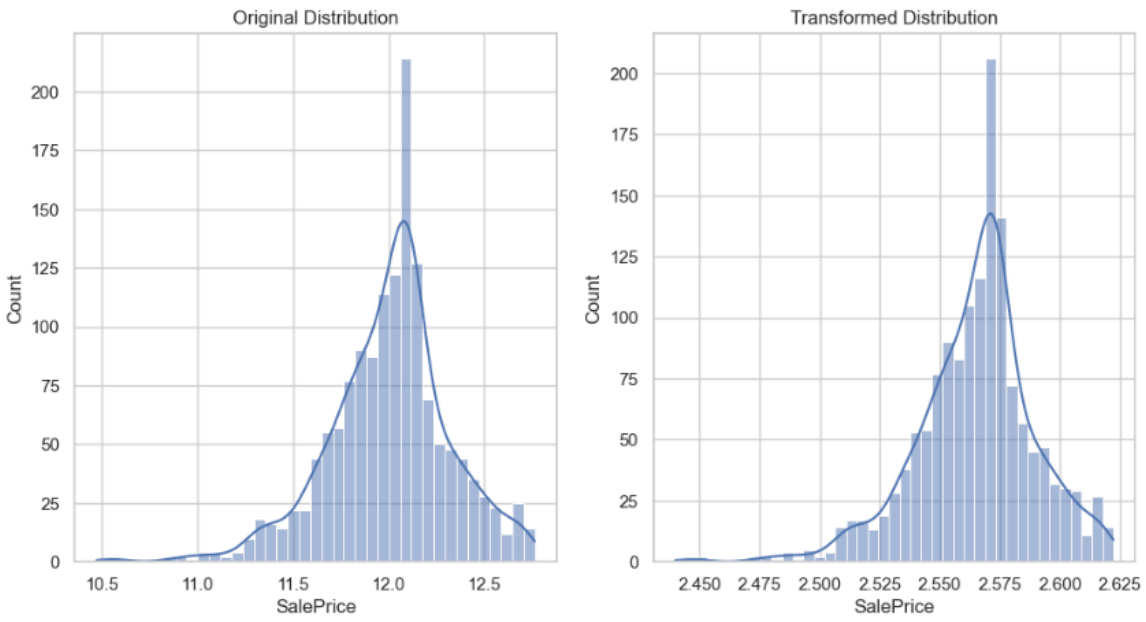
```
(2919, 81)
```

بعد از حذف:

```
(2919, 47)
```

.3

این داده‌ها در ابتدا به منظور پیروی از توزیع نرمال طراحی شده بودند، اما برای بهبود و نرمال‌سازی بیشتر داده‌ها، یک تبدیل لگاریتمی روی آن‌ها اعمال کردیم.



.4

ما قبلاً داده‌های از دست‌رفته را مدیریت کرده، داده‌های پرت را حذف کرده و داده‌ها را نیز تقسیم کرده‌ایم.
همچنین، متغیرهای دسته‌بندی‌شده را کدگذاری کرده و مقیاس‌دهی ویژگی‌ها را اعمال کرده‌ایم.

.5

استفاده از نرخ یادگیری کوچک (مثلاً ۰,۰۱) باعث می‌شود که فرآیند آموزش بهتر عمل کند و منجر به تعمیم بهتر مدل شود. با این حال، این کار نیازمند تعداد بیشتری از دورهای بوستینگ (درخت‌ها) است تا به همان سطح دقت در آموزش برسیم.
نرخ یادگیری بزرگ‌تر (مثلاً ۰,۲) باعث می‌شود که هر درخت تأثیر بیشتری بر مدل داشته باشد، اما اگر تعداد دورهای بوستینگ متناسب تنظیم نشود، ممکن است به بیش برازش (overfitting) منجر شود. نرخ‌های یادگیری بزرگ ممکن است باعث شوند مدل از وزن‌های بهینه برای هر درخت عبور کند و در نتیجه پیش‌بینی‌های کمتری دقیق ارائه دهد.

انتخاب نرخ یادگیری مناسب مستلزم توازن میان زمان آموزش، دقت مدل و خطر بیش‌برازش است. اعتبارسنجی متقاطع و تنظیم پارامترها برای یافتن نرخ یادگیری بهینه برای یک مجموعه داده و مسئله خاص ضروری هستند.

بهترین عملکرد برای Lasso با این معیارها به دست آمد: {'alpha': 1}

.6

نتیجه‌ی پیاده‌سازی در کد در تصویر زیر نمایش داده شده است:

```
lasso_grid Cross-Validation RMSE Scores:  [ 6634.38701797   336.46536341   149.55741912 10793.79428715
   410.0367632 ]
lasso_grid Mean RMSE:  3664.848170171168
elasticnet_grid Cross-Validation RMSE Scores:  [21434.42264315 24081.53547292 20390.10749371 21955.93903086
 23552.06317577]
elasticnet_grid Mean RMSE:  22282.813563283515
kernel_ridge_grid Cross-Validation RMSE Scores:  [62466.51322631 43511.5783172  34761.23047377 43133.78773823
 50546.26909601]
kernel_ridge_grid Mean RMSE:  46883.87577030273
gradient_boosting_grid Cross-Validation RMSE Scores:  [33557.32857383 33307.5367292  27938.10088636 34626.30283382
 38784.79243522]
gradient_boosting_grid Mean RMSE:  33642.81229168429
```

.7

مدل پشته‌ای یک تکنیک ensemble learning است که پیش‌بینی‌های چندین مدل پایه (یادگیرنده) را با استفاده از یک مدل متا ترکیب می‌کند که معمولاً به آن blender یا یادگیرنده متا گفته می‌شود.

رگرسیون پشته‌ای یک کاربرد خاص از استکینگ مدل است که در آن از مدل‌های رگرسیون به عنوان مدل‌های پایه استفاده می‌شود و یک مدل رگرسیونی به عنوان مدل متا به کار می‌رود. رگرسیون پشته‌ای از قدرت پیش‌بینی جمعی چندین الگوریتم رگرسیون بهره می‌برد و پیش‌بینی دقیق‌تر و مقاوم‌تری برای مسائل رگرسیونی فراهم می‌کند.

.8

این قسمت را به طور کامل در کد توضیح دادیم و به پیاده سازی آن پرداختیم.

.9

میانگین مربعات خطا (MSE) و ضریب تعیین (R^2) بهبود نیافتند.
مقایسه با بهترین نتیجه بخش 6 در زیر نشان داده شده است:

```
stacked_mse
```

```
np.float64(312766028378.0012)
```

```
print(np.mean(elasticnet_grid_mse_scores))
```

```
-498374849.3892803
```

```
kernel_ridge_grid_mse_scores
print(np.mean(kernel_ridge_grid_mse_scores))
```

```
-2283822966.1058846
```

```
lasso_grid_mse_scores
print(np.mean(lasso_grid_mse_scores))
```

```
-32164958.545431864
```

Problem 2: Predicting Song Sales Using Machine Learning

.1

بعد از لود مجموعه داده، ستون song_name را به دلیل منحصربه‌فرد بودن نام هر آهنگ و نداشتن ارزش آموزشی حذف می‌کنیم .سپس به ازای هر ستون تعداد مقادیر متفاوت را محاسبه می‌کنیم و اگر تعداد آنها کم‌تر از ۵۰ عدد بود آن ستون دسته‌ای (categorical) و در غیر اینصورت عددی (numerical) است.

ستون‌های دسته‌ای برابر است با:

['key', 'audio_mode', 'time_signature']

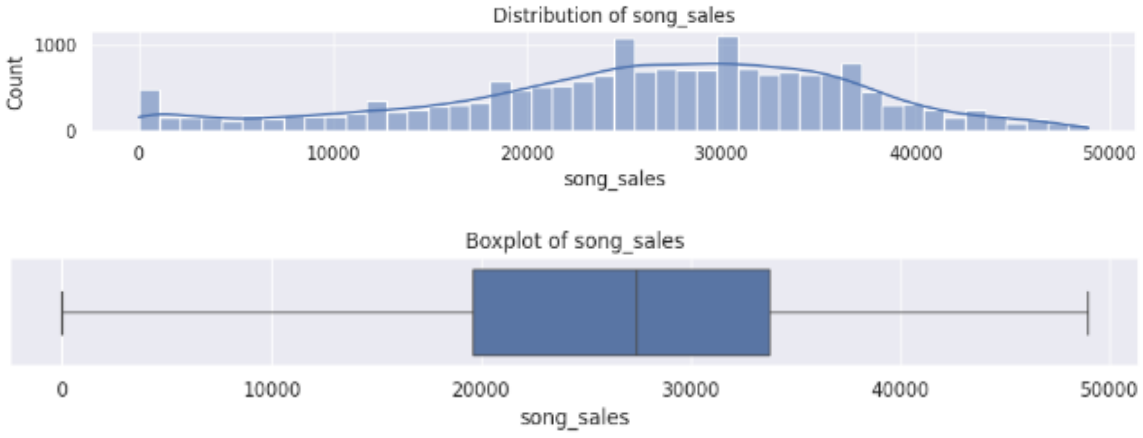
ستون‌های عددی برابر است با:

['song_duration_ms', 'acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'audio_valence', 'song_sales']

.2

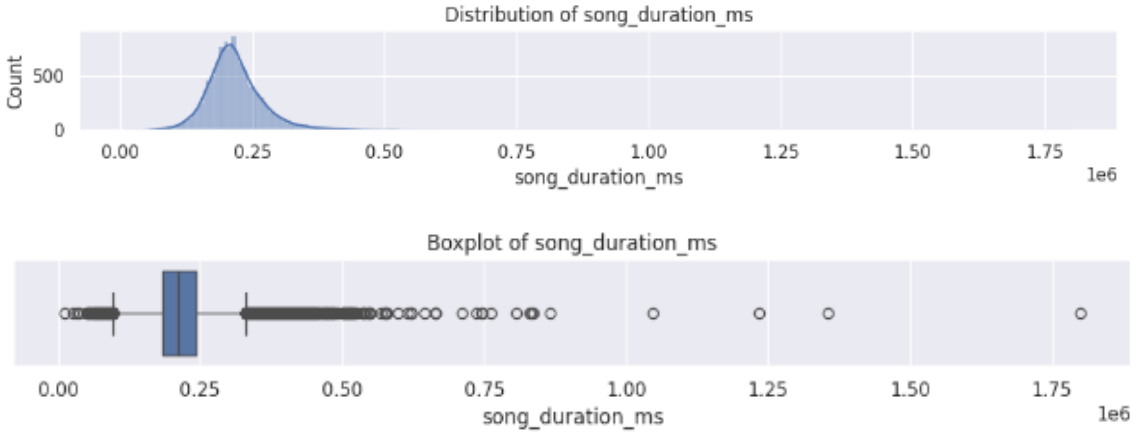
متغیر هدف ستون 'song_sales' و متغیرهای عددی همانطور که در بخش قبلی مشخص شدند برابر اند با :
['song_duration_ms', 'acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'audio_valence']

توزیع هر ستون را با نمودار میله‌ای و جعبه‌ای رسم می‌کنیم:



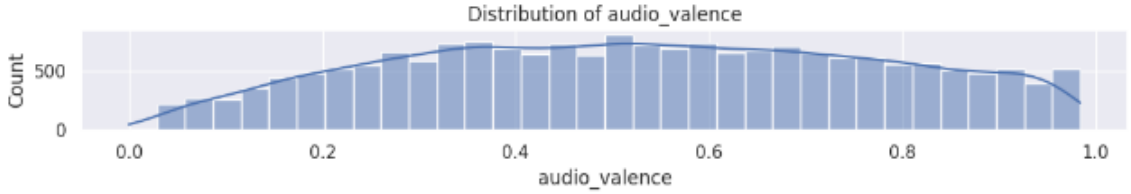
نمودار اول نشان می‌دهد که توزیع به سمت راست متمایل است (right-skewed)، این نشان‌دهنده این است که تعداد کمی از آهنگ‌ها فروش بسیار زیادی داشته‌اند در حالی که بیشتر آهنگ‌ها فروش کمی داشته‌اند .تقریباً بیشتر آهنگ‌ها فروش کمی دارند (بیشترین تعداد داده‌ها در سمت چپ نمودار است که به معنی تعداد زیاد آهنگ‌هایی است که فروش کمی داشته‌اند). در انتهای سمت چپ و راست، تعداد آهنگ‌هایی که فروش کم یا زیاد دارند، به تدریج کاهش می‌یابد. این تایید می‌کند که اکثریت آهنگ‌ها فروش کم و تعداد کمی فروش بالا داشته‌اند.

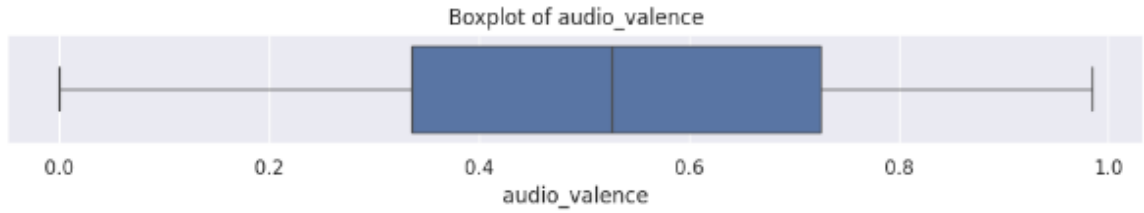
نمودار دوم نشان می‌دهد که میانگین فروش آهنگ حدودا 27000 نسخه است. این بدان معناست که نیمی از آهنگ‌های مجموعه داده‌ها 27000 نسخه یا بیشتر و نیمی از آهنگ‌ها 27000 نسخه یا کم‌تر فروخته شدند. توزیع فروش آهنگ به سمت راست منحرف شده است و این بدان معناست که تعداد آهنگ‌هایی که کم‌تر از 20000 نسخه فروخته‌اند، بیشتر از آهنگ‌هایی است که بیش از 20000 نسخه فروخته‌اند؛ همچنین در این نمودار مقادیر پرت را مشاهده نمی‌کنیم که این نشان‌دهنده این است که بیشتر داده‌ها در محدوده نرمال توزیع قرار دارند.



نمودار اول نشان می‌دهد که توزیع مدت‌زمان آهنگ‌ها به سمت راست متمایل است (right-skewed)، به این معنا که تعداد کمی از آهنگ‌ها مدت‌زمان بسیار زیادی دارند، در حالی که بیشتر آهنگ‌ها در محدوده زمانی کوتاه‌تر قرار می‌گیرند. بیشترین تراکم آهنگ‌ها در بازه 2000 تا 3000 میلی‌ثانیه مشاهده می‌شود. در انتهای راست نمودار، تعداد آهنگ‌هایی که مدت‌زمان بیشتری دارند، به تدریج کاهش می‌یابد. این نشان می‌دهد که آهنگ‌های طولانی‌تر کمتر رایج هستند و اکثر آهنگ‌ها مدت‌زمان معمولی دارند.

نمودار دوم نشان می‌دهد که مقدار میانه مدت‌زمان آهنگ‌ها کمتر از مقدار میانگین قرار دارد که این موضوع تأیید می‌کند توزیع داده‌ها به سمت راست متمایل است. همچنین تعداد زیادی مقدار پرت در سمت راست جعبه مشاهده می‌شود، که نشان‌دهنده آهنگ‌هایی با مدت‌زمان بسیار طولانی‌تر از حد معمول است. این مقدارهای پرت احتمالاً شامل موسیقی‌های بی‌کلام یا آهنگ‌های دیگر با مدت‌زمان طولانی هستند. بیشترین آهنگ‌ها در محدوده‌ای قرار دارند که بین چارک اول (Q1) و چارک سوم (Q3) مشخص شده است، در حالی که آهنگ‌های بسیار طولانی‌تر به عنوان مقادیر پرت خارج از این محدوده قرار گرفته‌اند.

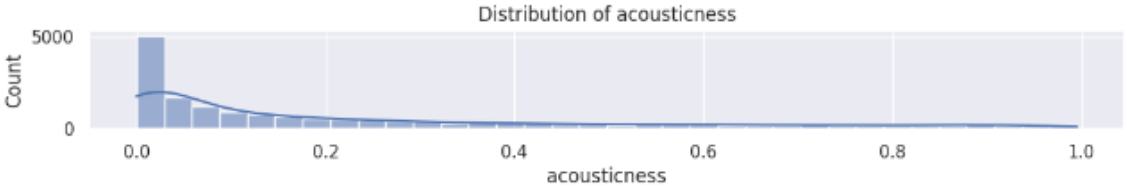


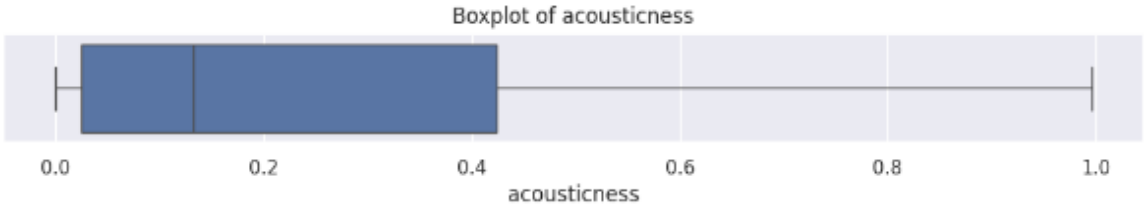


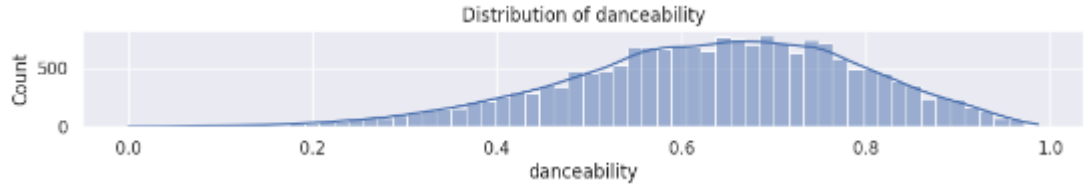
نمودار اول نشان می‌دهد که توزیع audio_valence تقریباً یکنواخت و متقارن است، به این معنا که مقادیر در کل بازه 0 تا 1 پراکنده شده‌اند. با این حال، تراکم داده‌ها در مقادیر پایین‌تر از 0.5 کمتر است و سپس در حوالی 0.5 تا 0.75 به بیشترین مقدار خود می‌رسد. این نشان می‌دهد که اکثر آهنگ‌ها دارای سطحی متوسط از valence هستند، که نشان‌دهنده احساسی متعادل بین غمگین و شاد است.

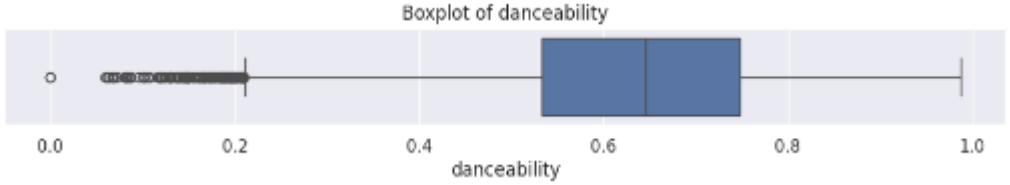
نمودار دوم نشان می‌دهد که داده‌های valence به‌طور کلی در بازه 0 تا 1 قرار دارند و هیچ مقدار پرت (outlier) قابل مشاهده نیست؛ این نشان‌دهنده این است که میزان valence در تمام آهنگ‌ها به‌طور یکنواخت توزیع شده و مقدارهای پرت وجود ندارند. میانه در حدود 0.5 قرار دارد که به این معناست که نیمی از آهنگ‌ها میزان valence بالاتر از 0.5 و نیمی دیگر کمتر از آن دارند. بازه بین چارک اول (Q1) و چارک سوم (Q3)نسبتاً وسیع است، که نشان می‌دهد تنوع زیادی در میزان valence آهنگ‌ها وجود دارد.

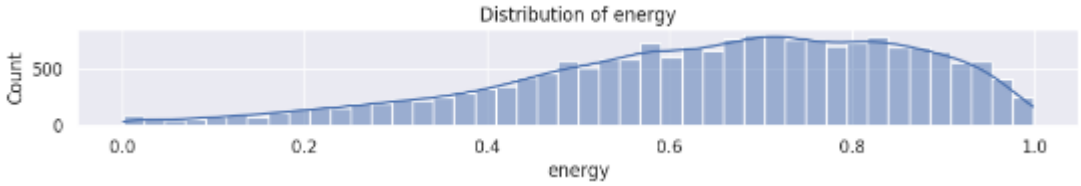
مابقی نمودارها نیز به همین صورت قابل تفسیر است:

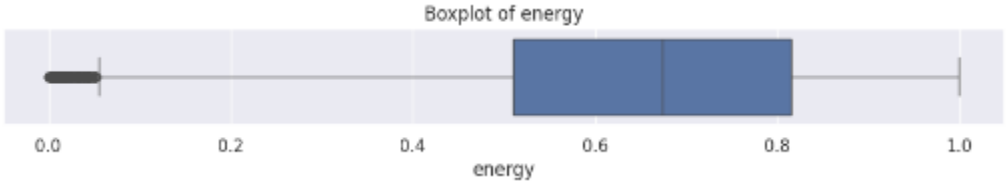


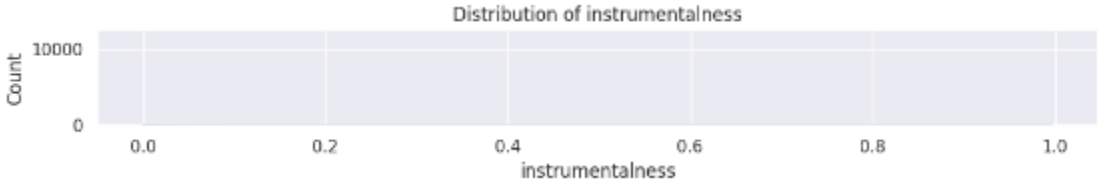


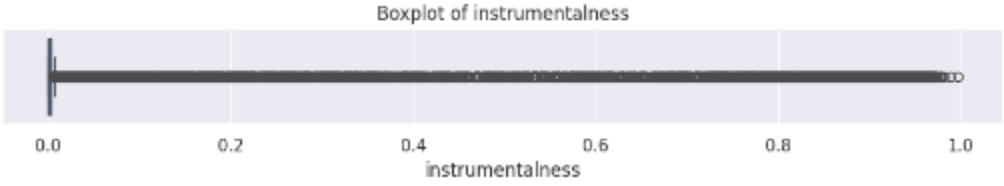

Distribution of danceability
Count
500
0
0.0
0.2
0.4
0.6
0.8
1.0
danceability

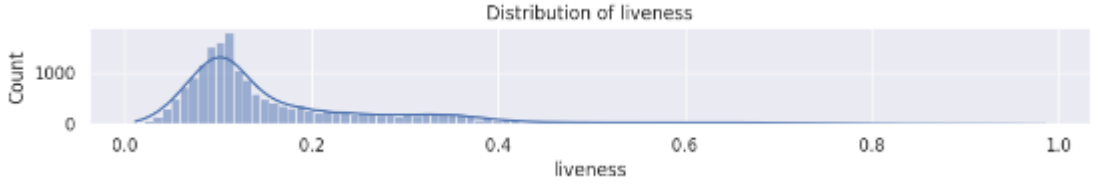

Boxplot of danceability
0.0
0.2
0.4
0.6
0.8
1.0
danceability

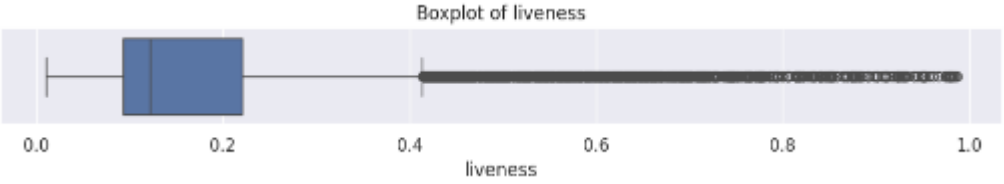

Distribution of energy
Count
500
0
0.0
0.2
0.4
0.6
0.8
1.0
energy

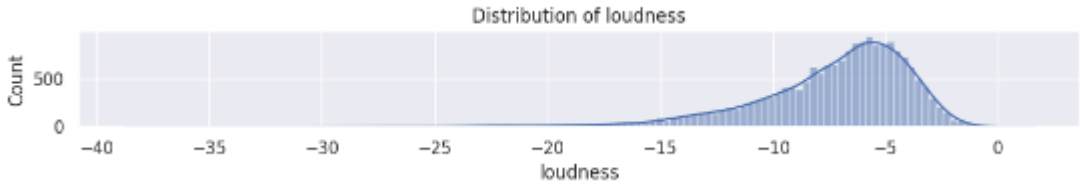

Boxplot of energy
0.0
0.2
0.4
0.6
0.8
1.0
energy


Distribution of instrumentalness
Count
10000
0
0.0
0.2
0.4
0.6
0.8
1.0
instrumentalness

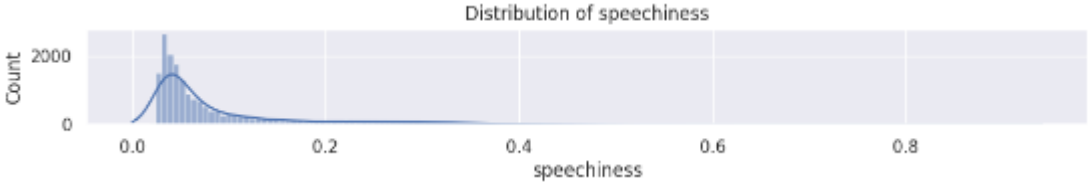

Boxplot of instrumentalness
0.0
0.2
0.4
0.6
0.8
1.0
instrumentalness

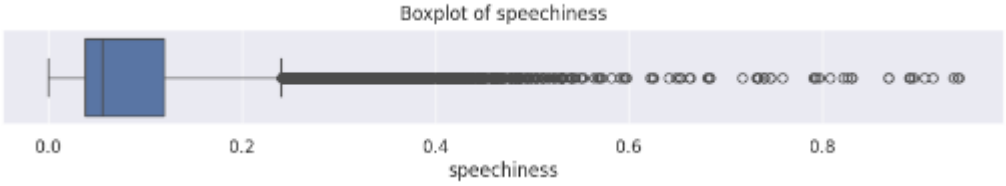

Distribution of liveness
Count
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
liveness

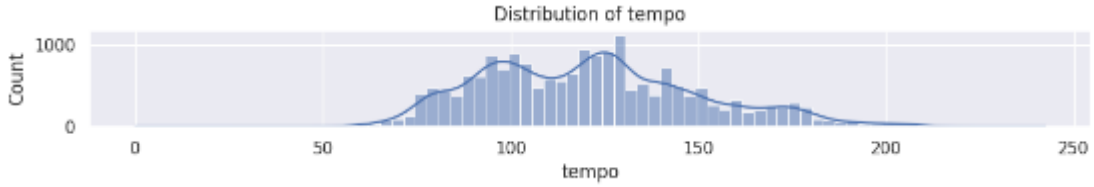

Boxplot of liveness
0.0
0.2
0.4
0.6
0.8
1.0
liveness

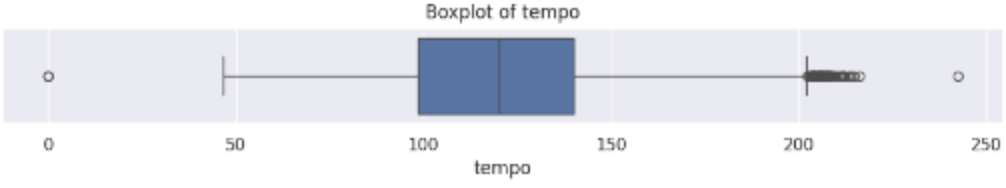

Distribution of loudness
Count
500
0
−40
−35
−30
−25
−20
−15
−10
−5
0
loudness


Boxplot of loudness
−40
−35
−30
−25
−20
−15
−10
−5
0
loudness


Distribution of speechiness
Count
2000
0
0.0
0.2
0.4
0.6
0.8
speechiness


Boxplot of speechiness
0.0
0.2
0.4
0.6
0.8
speechiness


Distribution of tempo
Count
1000
0
0
50
100
150
200
250
tempo


Boxplot of tempo
0
50
100
150
200
250
tempo

3.

- تشخیص descriptive داده‌های پرت

هدف: روش‌های توصیفی با هدف شناسایی نقاط پرت براساس ویژگی‌های آماری داده‌ها می‌باشد. این روش‌ها اطلاعاتی در مورد وجود و ویژگی‌های پرت ارائه می دهند اما لزوما راه حلی برای رسیدگی به آنها ارائه نمی‌دهند .

روش‌ها:

-Z_score : نقاط پرت را براساس تعداد انحراف معیارها که یک نقطه داده از میانگین فاصله دارد، شناسایی می‌کند.

-IQR (محدوده بین مربعی): نقاط پرت را براساس محدوده بین صدک ۲۵ و ۷۵ شناسایی می‌کند.

مزایا: پیاده‌سازی سریع و آسان، بینشی در مورد توزیع داده‌ها ارائه می‌دهد.

معایب: تصمیم روشنی در مورد نحوه رسیدگی به موارد پرت ارائه نمی‌کند.

- تشخیص prescriptive داده‌های پرت

هدف: روش‌های prescriptive با ارائه اقدامات خاصی برای انجام در هنگام شناسایی موارد پرت یک گام فراتر می‌روند. آنها راه‌هایی را برای مدیریت یا کاهش تأثیر داده‌های پرت بر تحلیل یا مدل پیشنهاد می‌کنند.

روش‌ها:

-Winsorization: مقادیر شدید را با مقادیر کم‌تر جایگزین می‌کند.

-Trimming: درصد معینی از داده‌ها را از دو سر توزیع حذف می‌کند.

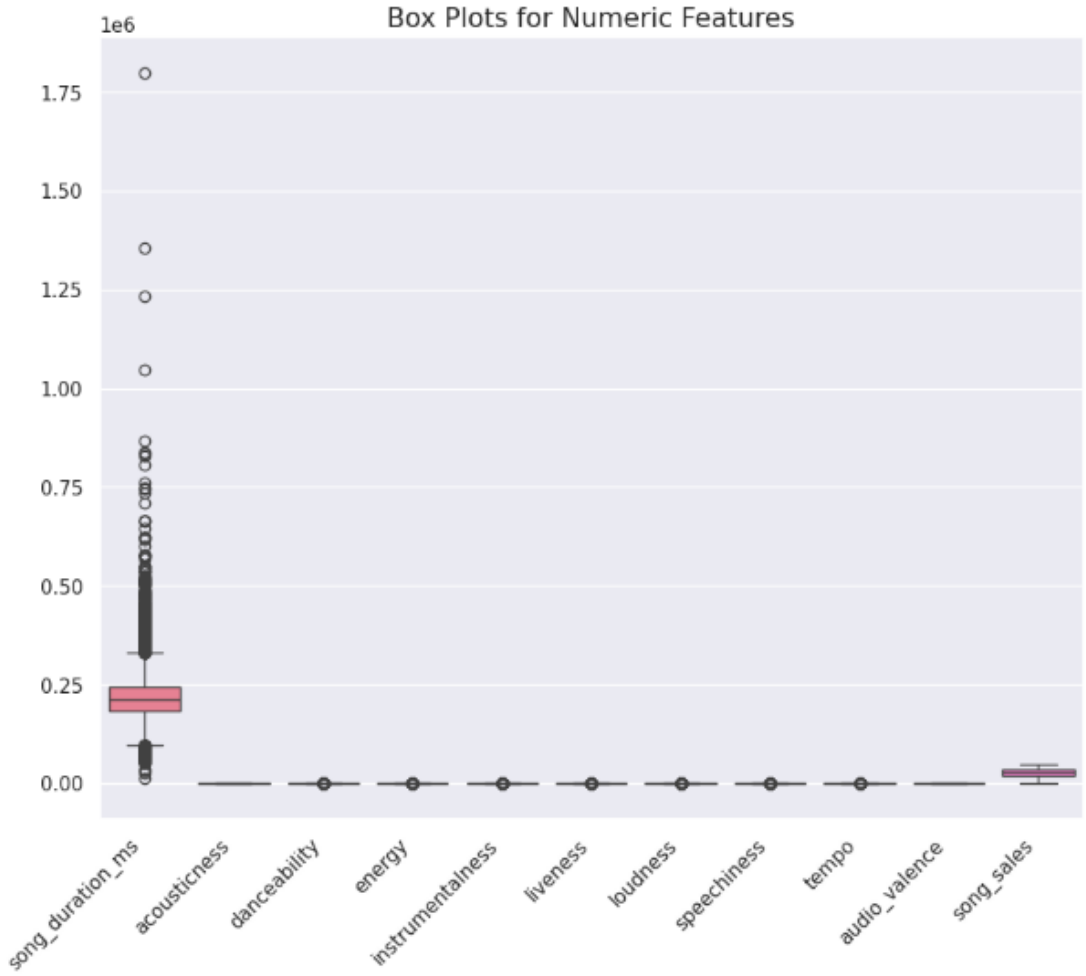
مزایا: گام‌های مشخصی را برای مقابله با موارد پرت ارائه می‌دهد، می‌تواند استحکام تحلیل‌ها یا مدل‌های آماری را بهبود بخشد .

معایب: اگر با دقت اعمال نشود، ممکن است تعصب ایجاد کند و همیشه برای همه انواع داده‌ها مناسب نباشد.

برای مساله پیش‌بینی فروش آهنگ بر اساس مجموعه داده‌های ارائه‌شده، مهم است که در ابتدا برای درک توزیع داده‌ها و شناسایی نقاط پرت بالقوه، روش تشخیص descriptive انجام شود. این به ما بینشی در مورد ویژگی‌هایی که ممکن است دارای مقادیر شدید باشند می‌دهد. روش‌های prescriptive مانند winsorization یا trimming ، شامل تغییر داده‌ها هستند. در این زمینه، تغییر داده‌های فروش ممکن

است توصیه نشود، زیرا می‌تواند دقت پیش‌بینی‌ها را دستکاری کند. بنابراین رویکرد descriptive برای تحلیل اولیه مناسب‌تر است.

نمودار جعبه‌ای ستون‌های عددی به صورت زیر است:

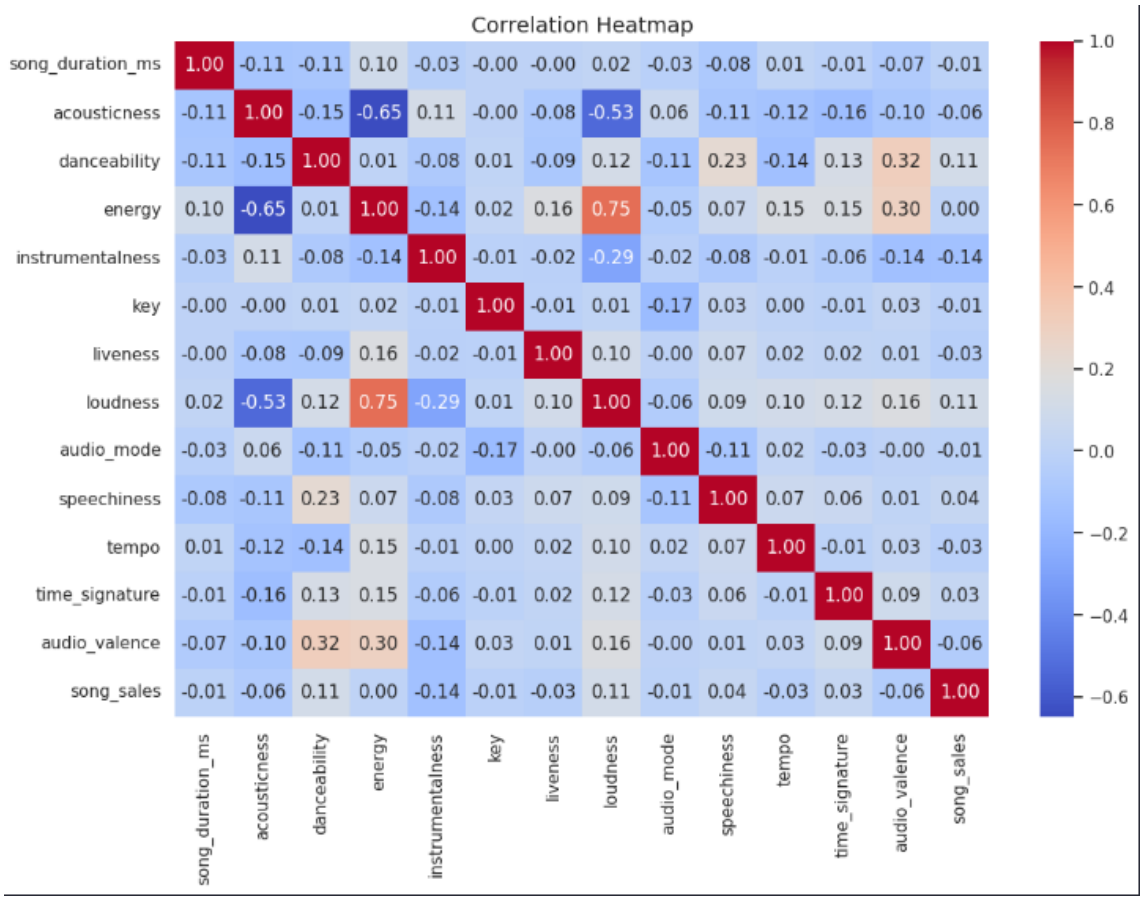


همانطور که از نمودار بالا نیز مشخص است، همه مقادیر ستون‌ها در بازه میانگین قرار دارند به جز ستون 'song _duration_ ms' که دارای داده‌های پرت بسیاری است. برای حذف مقادیر پرت، می‌توان از روش‌های descriptive مانند Z-Score یا IQR استفاده کرد که در اینجا از روش Z-Score با حد آستانه ٤ استفاده شده است. تعداد داده‌های پرت شناسایی شده برابر با ٥١٦ است.

روش Z-Score به این صورت عمل می‌کند که به ازای هر داده اختلاف آن را با میانگین آن ستون به دست آورده و تقسیم بر انحراف معیار آن ستون می‌کند. اگر عدد به دست آمده از حدآستانه بیشتر باشد، آن داده پرت است.

.4

برای ترسیم روابط بین ستون‌های مجموعه داده از روش‌های مختلفی می‌توان استفاده کرد که ابتدا به Heatmap می‌پردازیم:



در این ماتریس، رنگ‌ها قدرت و جهت همبستگی‌ها را نشان می‌دهند. همبستگی‌های مثبت در رنگ های گرم‌تر و همبستگی‌های منفی در رنگ‌های سردتر است. ستون acousticness با ستون‌های energy و loudness همبستگی منفی دارد. ستون energy و loudness همبستگی مثبت دارند. برای بررسی همبستگی بین ستون‌های دسته‌ای (categorical) می‌توانیم از آزمون square-chi استفاده کنیم. این روش به این صورت است که به صورت دو به دو ستون‌ها را دسته‌بندی کرده و مقدار p-value آنها را محاسبه می‌کنیم؛ اگر مقدار p-value کم‌تر یا مساوی پنج صدم باشد، این دو ستون با یکدیگر همبستگی دارند.

var2	audio_mode	key	time_signature
var1			
audio_mode	NaN	0.000000	0.011298
key	0.000000	NaN	0.000321
time_signature	0.011298	0.000321	NaN

همه ستون‌های categorical این مجموعه داده با همدیگر همبستگی دارند. برای ستون‌های عددی نیز می‌توانیم از ماتریس واریانس/کواریانس استفاده کنیم که در ادامه این ماتریس قرار دارد. برای مقدار کواریانس مقادیر نزدیک به ۱ یا ۱- نشان دهنده یک رابطه خطی قوی‌تر بین ویژگی‌ها است.

	Variance	song_duration_ms	acousticness	danceability	\
song_duration_ms	2.694973e+09	1.000000	-0.112737	-0.106396	
acousticness	7.999329e-02	-0.112737	1.000000	-0.152501	
danceability	2.365809e-02	-0.106396	-0.152501	1.000000	
energy	4.321300e-02	0.097142	-0.648300	0.007050	
instrumentalness	4.300940e-02	-0.034578	0.111968	-0.079780	
liveness	1.630138e-02	-0.000997	-0.083690	-0.088405	
loudness	1.147700e+01	0.019898	-0.534330	0.117892	
speechiness	9.455632e-03	-0.081989	-0.106569	0.225218	
tempo	8.120911e+02	0.012408	-0.120953	-0.140134	
audio_valence	5.916980e-02	-0.068412	-0.097743	0.316302	
song_sales	1.153917e+08	-0.009040	-0.064408	0.107525	

	energy	instrumentalness	liveness	loudness	speechiness	\
song_duration_ms	0.097142	-0.034578	-0.000997	0.019898	-0.081989	
acousticness	-0.648300	0.111968	-0.083690	-0.534330	-0.106569	
danceability	0.007050	-0.079780	-0.088405	0.117892	0.225218	
energy	1.000000	-0.138843	0.157461	0.748284	0.068513	
instrumentalness	-0.138843	1.000000	-0.020651	-0.289716	-0.077314	
liveness	0.157461	-0.020651	1.000000	0.100515	0.073502	
loudness	0.748284	-0.289716	0.100515	1.000000	0.091658	
speechiness	0.068513	-0.077314	0.073502	0.091658	1.000000	
tempo	0.145982	-0.012477	0.023657	0.102251	0.068189	
audio_valence	0.298163	-0.144493	0.013052	0.159699	0.011517	
song_sales	0.002348	-0.138718	-0.031205	0.110691	0.037284	
...						
speechiness	0.068189	0.011517	0.037284			
tempo	1.000000	0.025537	-0.025640			
audio_valence	0.025537	1.000000	-0.056669			
song_sales	-0.025640	-0.056669	1.000000			

در ردیف ستون acousticness و ستون ms_duration_song مقدار تقریبا ۰,۱۱۲۷- است. این نشان دهنده رابطه منفی بین میزان آکوستیک و مدت زمان آهنگ است. با افزایش مدت زمان آهنگ، آکوستیک کاهش می‌یابد .ستون speechiness و ستون danceability تقریبا ۰,۲۲۵۲ است. این نشان دهنده رابطه مثبت نسبتا قوی بین speechiness و danceability است. آهنگ‌هایی با قابلیت رقص‌پذیری بالاتر تمایل به میزان گفتار

بیشتری دارند. مقدار ستون acousticness و ستون energy تقریبا ۰,٦٤٨٣- است. این حاکی از یک رابطه منفی قوی بین آکوستیک و انرژی است. با افزایش انرژی یک آهنگ، آکوستیک کاهش می یابد و بالعکس. رابطه بین ویژگی‌های دیگر را هم به همین منوال می‌توانیم بررسی کنیم.

5.

چندخطی(multicollinearity) به وضعیتی اشاره دارد که در آن دو یا چند متغیر مستقل در یک مدل رگرسیون همبستگی بالایی دارند. این همبستگی می‌تواند منجر به مشکلاتی در هنگام تفسیر مدل شود، زیرا تشخیص اثرات فردی هر متغیر پیش‌بینی‌کننده بر متغیر وابسته می‌تواند چالش برانگیز باشد. در ادامه چند نکته کلیدی در رابطه با مسائل ناشی از چندخطی بودن وجود دارد :

- **خطایInflated Standard :**چند خطی بودن می‌تواند منجر به خطای Inflated Standard ضرایب رگرسیون شود .این امر تشخیص اینکه کدام پیش‌بینی‌کننده‌ها واقعا به طور قابل‌توجهی به مدل کمک می‌کنند، دشوارتر می‌کند.
- **ضرایب غیرقابل اعتماد:** ضرایب متغیرهای همبسته می‌توانند ناپایدار شوند و علائم آنها حتی ممکن است در پاسخ به تغییرات کوچک در داده‌ها تغییر جهت دهند.
- **دشواری در تفسیر:** تفسیر سهم هر یک از متغیرهای پیش‌بینی‌کننده به متغیر وابسته دشوار می‌شود.

تشخیص چندخطی بودن:

یکی از روش‌های رایج برای تشخیص چند خطی بودن، محاسبه ضریب تورم واریانس یا Variance Inflation Factor (VIF)برای هر متغیر پیش‌بینی‌کننده است. VIF اندازه‌گیری می‌کند که چقدر واریانس یک ضریب رگرسیون تخمین زده شده به دلیل همخطی افزایش یافته است. یک VIF بالا (معمولا بیشتر از ۱۰) نشانه این است که چند خطی ممکن است یک مشکل باشد.

مدیریت چند خطی:

- حذف یکی از متغیرهای همبسته: اگر دو یا چند متغیر با هم همبستگی بالایی دارند، ممکن است مناسب باشد که فقط یکی از آنها را در مدل نگه داریم.
- ترکیب متغیرهای همبسته: اگر در زمینه داده‌ها منطقی باشد، می‌توان متغیرهای ترکیبی ایجاد کرد که ترکیبی از متغیرهای همبسته را نشان می‌دهد .
- انتخاب ویژگی یا کاهش ابعاد: تکنیک هایی مانند PCA را می‌توان برای تبدیل متغیرهای همبسته به مجموعه‌ای از متغیرهای غیر همبسته استفاده کرد.
- Regularization: تکنیک‌هایی مانند Lasso یا رگرسیون Ridge می‌تواند با افزودن یک عبارت جریمه به معادله رگرسیون به کاهش چندخطی کمک کند.

- جمع آوری داده‌های بیشتر: گاهی اوقات، جمع آوری داده‌های بیشتر می‌تواند به کاهش چندخطی کمک کند. اگر هر یک از مقادیر VIF به طور قابل توجهی بالاتر از ۱۰ باشد، وجود چند خطی را نشان می‌دهد.

مراحل پیش پردازش:

اگر چندخطی تشخیص داده شد، یک یا چند مرحله از مراحل زیر را در نظر می‌گیریم:

حذف یکی از متغیرهای همبسته، استفاده از تکنیک‌های انتخاب ویژگی، روش‌های کاهش ابعاد مانند PCA، استفاده از تکنیک‌های regulrization برای تعیین اینکه آیا بین ستون‌های مجموعه داده فعلی رابطه چندخطی وجود دارد، می‌توانیم مقدار VIF را دو به دو برای ستون‌ها محاسبه کنیم. اگر هر یک از مقادیر VIF به طور قابل توجهی بالاتر از ۱۰ باشد، نشان دهنده چندخطی بودن است.

	song_duration_ms	acousticness	danceability	energy	\
song_duration_ms	14.265089	3.177762	21.62545	30.732653	
acousticness	3.177762	3.177762	21.62545	30.732653	
danceability	21.62545	21.62545	21.62545	30.732653	
energy	30.732653	30.732653	30.732653	30.732653	
instrumentalness	1.400073	1.400073	1.400073	1.400073	
key	3.208983	3.208983	3.208983	3.208983	
liveness	2.681362	2.681362	2.681362	2.681362	
loudness	11.940959	11.940959	11.940959	11.940959	
audio_mode	2.80587	2.80587	2.80587	2.80587	
speechiness	2.154095	2.154095	2.154095	2.154095	
tempo	18.285176	18.285176	18.285176	18.285176	
time_signature	88.901818	88.901818	88.901818	88.901818	
audio_valence	7.716693	7.716693	7.716693	7.716693	

	instrumentalness	key	liveness	loudness	audio_mode	\
song_duration_ms	1.400073	3.208983	2.681362	11.940959	2.80587	
acousticness	1.400073	3.208983	2.681362	11.940959	2.80587	
danceability	1.400073	3.208983	2.681362	11.940959	2.80587	
energy	1.400073	3.208983	2.681362	11.940959	2.80587	
instrumentalness	1.400073	3.208983	2.681362	11.940959	2.80587	
key	3.208983	3.208983	2.681362	11.940959	2.80587	
liveness	2.681362	2.681362	2.681362	11.940959	2.80587	
loudness	11.940959	11.940959	11.940959	11.940959	2.80587	
audio_mode	2.80587	2.80587	2.80587	2.80587	2.80587	
...						
speechiness	2.154095	18.285176	88.901818	7.716693		
tempo	18.285176	18.285176	88.901818	7.716693		
time_signature	88.901818	88.901818	88.901818	7.716693		
audio_valence	7.716693	7.716693	7.716693	7.716693		

مقادیر قطری در جدول معمولا زمانی بالاتر هستند که خود متغیر با سایر متغیرها همبستگی بالایی داشته باشد. به طور مثال ستون‌های tempo، loudness، energy، danceability،song_duration_ms و time_signature دارای مقدار VIF بیشتر از 10 هستند که نشان می‌دهد ممکن است مشکلات چندخطی بودن برای این ستون وجود داشته باشد. به نظر می رسد که چندین متغیر دارای مقادیر VI F بالایی هستند که نشان دهنده مسائل بالقوه چند خطی بودن است. قابل ذکر است ،song_duration_ms، رقص‌پذیری و انرژی دارای VI F نسبتاً بالایی هستند.

VIF برای متغیرهای تکی:

14.27 = VIF :song_duration_ms

این نشان می‌دهد که song_duration_ms با سایر متغیرهای مجموعه داده همبستگی زیادی دارد.

3.18 = VIF: acousticness

VIF برای acousticness در مقایسه با song_duration_ms نسبتا پایین است. در حالی که هنوز مقداری همبستگی با سایر متغیرها نشان می‌دهد، به اندازه song_duration_ms شدید نیست.

21.63 = VIF :danceanility

danceanility دارای VIF بسیار بالایی است که نشان‌دهنده مشکلات چندخطی قوی است.

30.73 = VIF :energy

مانند danceability،energy دارای VIF بسیار بالایی است که نشان‌دهنده چندخطی بودن شدید است. این متغیر با سایر پیش‌بینی کننده‌ها همبستگی بالایی دارد.

Instrumentalness: VIF = 40.1 VIF برای instrumentalness نسبتا پایین است، که نشان می‌دهد همبستگی پایینی با سایر متغیرهای مجموعه داده دارد.

3.21 :VIF :Key

متغیر key مقداری همبستگی را با سایر متغیرها نشان می‌دهد اما به شدت danceability یا energy نیست.

2.6 = VIF :Liveness

متغیر liveness مقداری همبستگی با سایر متغیرها نشان می‌دهد اما خیلی مشکل‌ساز نیست.

11.94 = VIF :Loudness

loudness دارای VIF نسبتا بالایی است که نشان می‌دهد با سایر پیش‌بینی‌ها همبستگی دارد. با این حال، به شدت

energy یا danceability نیست.

2.81 = VIF :audio_mode

متغیر loudness مقداری همبستگی با متغیرهای دیگر نشان می‌دهد اما خیلی مشکل‌ساز نیست.

Speechiness 15.2 = VIF :Speechiness با سایر متغیرها همبستگی نشان می‌دهد اما خیلی مشکل‌ساز نیست.

18.29 = VIF :Tempo

tempo دارای VIF بسیار بالایی است که نشان دهنده مشکلات شدید چندخطی است. این متغیر با سایر پیش‌بینی کننده‌ها همبستگی بالایی دارد.

88.90 = VIF :time_signature

time_signature دارای VIF بسیار بالایی است که نشان می‌دهد به شدت با متغیرهای دیگر همبستگی دارد.

7.72 = VIF :audio_valence

audio_valence مقداری همبستگی با سایر متغیرها نشان می‌دهد، اما خیلی مشکل ساز نیست.

متغیرهایی مانند tempo، energy،danceability و time_signature دارای مقادیر بسیار بالای VIF هستند که نشان‌دهنده مشکلات چندخطی شدید است. این می تواند برآورد اثرات فردی آنها را چالش برانگیز کند.

loudness، song_duration_ms و audio_valence نیز مقادیر VIF نسبتا بالایی دارند که نشاندهنده سطحی از چندخطی بودن است.

.6

در ابتدا ستون هدف را از مجموعه داده جدا می‌کنیم و سپس ستون‌های عددی را با StandardScaler نرمال‌سازی می‌کنیم. بعد از آن مجموعه داده را به نسبت 80 به 20 به train و test تقسیم میکنیم .

یک بار داده‌های بخش آموزش را به الگوریتم رگرسیون خطی که نوشتیم می‌دهیم و خروجی آن که وزن‌های متغیرهای مجموعه داده است را می‌گیریم. ورودی این الگوریتم داده‌های بخش آموزش، ستون هدف بخش آموزش، گام یادگیری که برابر یک صدم قرار داده شده و تعداد دفعات تکرار که 100 هزار قرار دارد، است.

این الگوریتم به این صورت عمل می‌کند که در ابتدا وزن‌های متغیرها را مقداردهی اولیه می‌کند و یک اصطلاح بایاس به داده‌های ورودی اضافه می‌کند و تعداد مشخصی بار تکرار می شود (100 هزاربار) .پیش‌بینی‌ها را براساس وزن‌های فعلی محاسبه می‌کند و خطا (تفاوت بین پیش‌بینی‌ها و مقادیر واقعی) را محاسبه می‌کند. برای محاسبه گرادیان، شیب را محاسبه می‌کند که جهت شیب‌دارترین صعود تابع هزینه را نشان می‌دهد. سپس وزن‌ها را در

جهتی به‌روزرسانی می‌کند که هزینه را به حداقل می‌رساند و با نرخ یادگیری مقیاس‌بندی می‌شود. وزن‌های آموخته شده در انتها بازگردانده می‌شود.

این وزن‌های آموخته شده را به تابع predict می‌دهیم تا با داده‌های بخش test خروجی موردنظر را تولید کند.

این بار همین داده‌های بخش آموزش را با استفاده از کتابخانه sklearn و ماژول linear_model می‌دهیم تا وزن‌ها را یاد بگیرد و با تابع predict مخصوص خود کتابخانه، خروجی موردنظر را از داده‌های test تولید کند.

حال خروجی‌های تولید شده توسط این کتابخانه و تابعی که توسط خودمان نوشته شده را با mean_squared_error و r2_score و ستون هدف بخش test مقایسه می‌کنیم. همچنین زمان اجرای تابع رگرسیون خطی خودمان و کتابخانه را نیز باهم مقایسه می‌کنیم. نتایج به دست آمده در ادامه قرار دارد:

```
Mean Squared Error (Custom): 108014624.59630416
Mean Squared Error (Scikit-learn): 108014625.38866526

R2 Score (Custom): 0.04714830278283311
R2 Score (Scikit-learn): 0.047148295793014605

Prediction Time (Custom): 0.008478403091430664 seconds
Prediction Time (Scikit-learn): 0.003144979476928711 seconds

Training Time (Custom): 57.13363814353943 seconds
Training Time (Scikit-learn): 0.08011031150817871 seconds
```

7.

رگرسیون Lasso و رگرسیون Ridge دو تکنیک محبوبی هستند که برای منظم کردن مدل‌های رگرسیون خطی استفاده می‌شوند. این‌ها برای جلوگیری از برازش بیش از حد و بهبود تعمیم مدل ،یک عبارت جریمه به تابع هزینه اضافه می‌کنند.

- رگرسیون Lasso

هدف رگرسیون Lasso که مخفف "Last Absolute Shrinkage and Selection Operator" است مقدار مطلق ضرایب (جریمه L1) را به تابع هزینه اضافه می‌کند.

اثر: Lasso تمایل دارد برخی از ضرایب را تا انتها به صفر برساند و به طور موثر انتخاب ویژگی را با حذف متغیرهای دارای اهمیت کم‌تر انجام می‌دهد. موارد استفاده از آن برای زمانی است که شک داریم که فقط تعداد کمی از ویژگی‌ها واقعا مرتبط هستند.

مزایا: می‌تواند برای انتخاب ویژگی استفاده شود، به انتخاب مدل پراکنده کمک می‌کند.

معایب: تمایل به انتخاب تنها یک متغیر از بین متغیرهایی که همبستگی بالایی دارند.

- رگرسیون Ridge

هدف: رگرسیون Ridge مقدار مجذور ضرایب (جریمه L2) را به تابع هزینه اضافه می‌کند.

اثر: رگرسیون Ridge ضرایب را به صفر نمی‌رساند اما آنها را به سمت یکدیگر کوچک می‌کند. این بدان معنی است که مدل را از اهمیت دادن بیش از حد به هر یکی از ویژگی‌ها منصرف می‌کند. موارد استفاده از آن هنگامی که تعداد زیادی ویژگی داریم و می‌خواهیم از بیش برازش جلوگیری کنیم مناسب است.

مزایا: می‌تواند چندخطی را به خوبی مدیریت کند، با جلوگیری از مقادیر زیاد ضرایب، مدل را تثبیت می‌کند.

معایب: انتخاب ویژگی را انجام نمی‌دهد.

در بخش پیاده‌سازی این دو رگرسیون مانند تابعی که برای بخش رگرسیون خطی نوشته بودیم، استفاده می‌کنیم با این تفاوت که عبارت تابع هزینه کمی تغییر یافته است که توضیح این دو تابع هزینه در ادامه قرار دارد.

تابع هزینه رگرسیون Lasso:

$$\frac{1}{2m}\sum(y-y`)^2+\lambda\sum|w_i|$$

تابع هزینه رگرسیون Ridge:

$$\frac{1}{2m}\sum(y-y`)^2+\lambda\sum{w_i}^2$$

نتایج به دست آمده از پیاده‌سازی به شرح زیر است:

```
Mean Squared Error (Ridge): 108091956.86037835
Mean Squared Error (Lasso): 108005648.02424583

R2 Score (Ridge): 0.04714830278283311
R2 Score (Lasso): 0.047148295793014605

Training Time (Ridge): 50.228381395339966 seconds
Training Time (Lasso): 43.357879638671875 seconds
```

نتایج به دست آمده از بخش قبلی که مربوط به رگرسیون خطی است به شرح زیر است:

```
Mean Squared Error (Custom): 108014624.59630416
Mean Squared Error (Scikit-learn): 108014625.38866526

R2 Score (Custom): 0.04714830278283311
R2 Score (Scikit-learn): 0.047148295793014605

Prediction Time (Custom): 0.008478403091430664 seconds
Prediction Time (Scikit-learn): 0.003144979476928711 seconds

Training Time (Custom): 57.13363814353943 seconds
Training Time (Scikit-learn): 0.08011031150817871 seconds
```

همانطور که قابل مشاهده است ،با وجود این که در نتایج به دست آمده ممکن است خطایی به وجود آمده باشد اما همه مدل‌ها حتی مدلی که توسط کتابخانه آموزش داده شده بود، نتایج نزدیک به هم است. علت آن ممکن است در کیفیت داده‌های آموزشی باشد.

8.

برای بررسی امکان بیش برازش در مجموعه داده تست یا اعتبارسنجی، باید عملکرد مدل را در این مجموعه داده‌ها تجزیه و تحلیل کرد.

- **معیارهای عملکرد:** با نگاه کردن به معیارهای عملکرد مانند Mean Absolute Error، Mean Squared Error، Root Mean Squared Error و R2-Score در هر دو مجموعه داده تست و اعتبارسنجی می‌توان شروع کرد. اگر مدل به طور قابل توجهی در مجموعه اعتبارسنجی در مقایسه با مجموعه تست بهتر عمل کند، می‌تواند نشانه‌ای از بیش برازش باشد.
- **منحنیهای یادگیری(learning curves):** منحنی‌های یادگیری را ترسیم می‌کنیم که عملکرد مدل را در مجموعه های آموزشی و اعتبارسنجی در طول دوره ها نشان می‌دهد. اگر منحنی آموزش در حال بهبود باشد در حالی که منحنی اعتبارسنجی راکد یا رو به زوال است، این نشانه بیش برازش است.
- **تکنیک‌های تنظیم‌سازی (regularization):** تکنیک‌هایی مانند dropout، تنظیم‌سازی L1 یاL2 یا تنظیم‌سازی دسته‌ای را در معماری مدل پیاده‌سازی می‌کنیم. این روش‌ها می‌توانند با اضافه کردن جریمه‌ها به وزن‌ها در حین تمرین، به کاهش بیش برازش کمک کنند.
- **توقف زودهنگام:** توقف زودهنگام شامل نظارت بر عملکرد مدل در یک مجموعه اعتبارسنجی جداگانه در طول آموزش است. مجموعه اعتبارسنجی بخشی از داده‌هایی است که مدل قبلا هرگز ندیده است. پس از هر دوره، عملکرد مدل را در این مجموعه اعتبارسنجی ارزیابی می‌کنیم. اگر در هر نقطه‌ای، عملکرد مجموعه اعتبارسنجی از بهتر شدن متوقف شود یا شروع به بدتر شدن کند، توقف

زودهنگام شروع می‌شود. این بدین معناست که احتمالا مدل شروع به بیش برازش با داده‌های آموزشی می‌کند.

- **افزایش داده‌ها:** در صورت امکان از تکنیک‌های تقویت داده‌ها برای افزایش تنوع داده‌های آموزشی استفاده کنیم. این به مدل کمک می‌کند تا نمونه‌های دیده نشده را بهتر تعمیم دهد.
- **Cross-validation**: اگر مجموعه داده امکان استفاده از تکنیک‌هایی مانند k-fold را می‌دهد از آنها استفاده کنیم. این شامل تقسیم داده‌ها به زیر مجموعه‌های متعدد و آموزش/اعتبارسنجی مدل بر روی ترکیب‌های مختلف است و تخمین بهتری از عملکرد مدل ارائه می‌دهد.
- **تنظیم هایپرپارامتر:** هایپرپارامترها را با دقت تنظیم کنیم به خصوص آنهایی که مربوط به تنظیم‌سازی (regularization) هستند. با استفاده از تکنیک‌هایی مانند جستجوی شبکه‌ای (Grid search) یا جستجوی تصادفی یک جستجوی سیستماتیک انجام دهیم.
- **از یک مجموعه داده Holdout استفاده کنیم:** در صورت امکان ،یک مجموعه داده جداگانه نگه داریم که هرگز در طول آموزش یا اعتبارسنجی استفاده نشده است. این مجموعه فقط باید برای ارزیابی نهایی پس از تنظیم مدل استفاده شود.

با به کارگیری این تکنیک ها، می‌توان بیش برازش را در پروژه شناسایی و کاهش داد که در نهایت منجر به یک مدل بهتر و قابل تعمیم می‌شود.

9.

تقسیم متغیر هدف به فواصل بازه‌ای و در نظر گرفتن آن به عنوان یک مساله دسته‌بندی، دیدگاه متفاوتی را نشان می‌دهد. در این مورد خاص، با پنج دسته مختلف (بدترین_فروشنده تا بهترین_فروشنده)، شکلی از دسته‌بندی ordinal را نشان می‌دهد.

- دسته‌بندی در مقابل رگرسیون

در رگرسیون، هدف پیش‌بینی یک مقدار عددی پیوسته است که می‌تواند هر عدد واقعی در یک محدوده مشخص باشد. رگرسیون خطی برای این کار مناسب است زیرا رابطه بین متغیرهای مستقل و هدف عددی را مدل می‌کند.

در دسته‌بندی، هدف پیش‌بینی این است که یک داده متعلق به کدام دسته یا کلاس است. هر دسته مجزا و متقابلا منحصر به فرد است.

با تقسیم فروش به فواصل بازه‌ای، اساسا یک مساله دسته‌بندی ایجاد می‌کنیم که در آن سعی می‌شود پیش‌بینی شود که فروش یک آهنگ در کدام بازه قرار می‌گیرد.

- چالش‌ها:

-از دست دادن اطلاعات: هنگامی که داده‌های پیوسته را در فواصل بازه‌ای ذخیره می‌کنیم، ذاتا برخی از اطلاعات را از دست می‌دهیم. به عنوان مثال دو آهنگ با فروش 9999 و 10001 در دسته‌های مختلف قرار می گیرند حتی اگر از نظر ارزش بسیار نزدیک باشند.

-**انتخاب فواصل بازه‌ای:** انتخاب فواصل بسیار مهم است. اگر فواصل زمانی بسیار گسترده باشد، ممکن است تمایزات مهم را از دست بدهیم. اگر آنها خیلی کوچک باشند، ممکن است در برخی دسته‌ها تعداد داده بسیار کمی داشته باشیم.

-**عدم تعادل در کلاس‌ها:** بسته به نحوه انتخاب فواصل، ممکن است کلاس‌های نامتعادل داشته باشیم. برای مثال اگر اکثر آهنگ‌ها در دسته (بدترین_فروشنده)قرار بگیرند، ممکن است مدل نسبت به پیش‌بینی این دسته تعصب داشته باشد.

-**قابلیت تفسیر:** تفسیر نتایج پیچیده‌تر می شود. به جای یک پیش‌بینی عددی ساده، اکنون احتمال قرار گرفتن یک آهنگ در یک دسته فروش خاص را تفسیر می‌کنیم.

-**انتخاب مدل:** الگوریتم‌های دسته‌بندی مختلف (به عنوان مثال رگرسیون لجستیک، درخت‌های تصمیم و ...) نقاط قوت متفاوتی دارند و ممکن است عملکرد متفاوتی در این مساله داشته باشند.

با توجه به این رویکرد، ممکن است بخواهیم از الگوریتم‌های دسته‌بندی استفاده کنیم. تکنیک‌هایی مانند رگرسیون لجستیک، درخت‌های تصمیم، جنگل‌های تصادفی یا ماشین‌های بردار پشتیبان می‌توانند مناسب باشند.

-**معیارهای ارزیابی:** صحت، دقت ،recall،F1-measure و ROC-AUC معیارهای مناسبی برای ارزیابی عملکرد یک مدل دسته‌بندی هستند.

به طور خلاصه، استفاده از فواصل بازه‌ای برای متغیر هدف، مسئله را به یک کار دسته‌بندی تبدیل می‌کند. این می‌تواند یک رویکرد درست باشد اما ملاحظاتی را در مورد از دست دادن اطلاعات، انتخاب فواصل و عدم تعادل بازه‌ای بالقوه معرفی میکند. ارزیابی دقیق و انتخاب مدل‌های مناسب برای این نوع مساله مهم است.

10.

خروجی نتایج مدل به صورت زیر است:

```
Mean Squared Error: 108014624.59630416
R2 Score: 0.04714830278283311
Mean Absolute Error: 8221.2120048122
Root Mean Squared Error: 10393.008447812605
```

از آنجایی که بهترین نتایج خروجی برای MSE،MAE و RMSE صفر و برای R2-Score برابر 1 است؛ در نتیجه خروجی مدل آن‌چنان خوب نیست. یک دلیل آن می‌تواند پیش پردازش غیردقیق و حذف نکردن ستون‌های با همبستگی بالا باشد. هر چقدر که پیش پردازش مجموعه داده بهتر باشد، یادگیری مدل دقیق‌تر و در نتیجه دقت مدل بهتر می‌شود.

.11

از روش‌هایی که می‌توانیم مدل را بهبود دهیم می‌توانیم به پیش پردازش بهتر و حذف متغیرهای با همبستگی بالا اشاره کنیم .همچنین می‌توانیم با data augmentation سایز مجموعه داده را افزایش دهیم و یا از مدل‌های پیچیده‌تری مثل polynomial regression به جای linear regression استفاده کنیم. علاوه بر آن می‌توانیم متغیرهایی با توان بالا اضافه کنیم و یا متغیرها را به فضای دیگری منتقل کنیم که راحت‌تر مساله خروجی‌ها را تولید کند. یکی دیگر از روش‌های افزایش اندازه مجموعه داده نیز می‌تواند اضافه کردن ضرب متغیرها به مجموعه داده باشد.

Problem 3: Feature Selection and Classification Using Weighted K-Nearest Neighbors (KNN)

.1

در ابتدا دیتاست مربوطه را لود می‌کنیم. به عنوان پیش پردازش برای مدل KNN لازم است تا Standardization مناسبی صورت گیرد از این رو از minmax استفاده می کنیم تا الگوریتم KNN برای محاسبه فواصل تحت تاثیر بازه مقادیر ویژگی‌ها قرار نگیرد.

```
features = df.iloc[:,:-1]
target_value = df.iloc[:,-1]

def minmax_norm(X:pd.Series):
    return (X - np.min(X)) / (np.max(X) - np.min(X))

for col in features.columns:
    df[col] = minmax_norm(df[col])
```

با این کار بازه مقادیر تمامی ویژگی‌ها بین صفر و یک قرار می‌گیرد. سپس برای بخش بندی کردن داده‌ها به دو مجموعه آموزش و تست، از تابع زیر استفاده می‌کنیم. در این تابع ابتدا مقادیر مختلف target_value شناخته می‌شوند و برای هر یک از مقادیر ممکن از target_value تعدادی sample به صورت رندوم انتخاب می‌گردد. این روش مبتنی بر stratified sampling می باشد.

```
def stratified_train_test_split(df, target_column, test_size=0.3):
    train_set = pd.DataFrame()
    test_set = pd.DataFrame()

    classes = df[target_column].unique()

    for class_ in classes:
        class_subset = df[df[target_column] == class_]

        test_subset_size = int(len(class_subset) * test_size)

        test_subset = class_subset.sample(test_subset_size)
        train_subset = class_subset.drop(test_subset.index)

        train_set = pd.concat([train_set, train_subset])
        test_set = pd.concat([test_set, test_subset])

    return train_set, test_set
```

```
train_df , test_df = stratified_train_test_split(df,'label',0.1)
```

در ادامه اندازه دو مجموعه آموزش و تست حاصل را به کمک shape بیان می‌کنیم.

برای مجموعه آموزش (270, 17) و برای مجموعه تست (30, 17) می‌باشد .

.2

در این بخش ابتدا تابعی می‌نویسیم که با دریافت مجموعه آموزش، تست و وزن‌ها weighted distance را حساب کرده و بین حاصل این فواصل، Majority vote بگیرد. بدین شکل که به ازای هر یک از اعضای مجموعه تست، ابتدا فاصله اش را تا هر یک از اعضای مجموعه آموزش برای هر یک از ویژگی‌ها حساب می‌کنیم و سپس در وزن ویژگی مربوطه ضرب می‌کنیم .فواصل را مرتب کرده و k همسایه نزدیک را دریافت می‌کنیم و سپس بین آنان رای اکثریت را اعمال می‌نماییم.

```
def weighted_knn(weights, train_points:pd.DataFrame, new_points:pd.DataFrame ,k = 5 ,target_value_name = 'label'):
    predictions = []
    for _, point in new_points.iterrows():
        distances = []
        for _, p in train_points.iterrows():
            counter = 0
            distance = 0
            for col in train_points.columns:
                if col != 'label' and col != 'Unnamed: 0':
                    distance += (weights[counter] * ((point[col] - p[col])**2))
                    counter += 1
            distance = np.sqrt(distance)
            distance = round(distance,ndigits=2)
            distances.append((distance,p[target_value_name]))
        # print(distances)
        distances = sorted(distances)[:k]

        counter_class_0 = 0
        counter_class_1 = 0

        for d in distances:
            if d[1] == 0:
                counter_class_0 += 1
            else:
                counter_class_1 += 1

        if counter_class_0 > counter_class_1:
            predictions.append(0)
        else:
            predictions.append(1)

    return predictions
```

با استفاده از تابع بالا ، پیش‌بینی زیر را انجام داده‌ایم . (وزن پیش فرض 1 گرفته شده که همان KNN عادی محسوب می‌شود).

[0,

0,

0,

0,

0,

0,

0,

1,

0,

0,

برای معیار های بررسی نتایج، از confusion matrix شروع کرده وtp,tn,fp,fn را حساب کرده‌ایم؛ سپس براساس این مقادیر precision ,recall و در نهایت f1_score را حساب کرده‌ایم.

```
def confusion_matrix(Y_true,Y_pred):
    TP = 0
    TN = 0
    FP = 0
    FN = 0
    for i in range(len(Y_pred)):
        if Y_pred[i] == Y_true[i]:
            if Y_true[i] == 0:
                TN += 1
            elif Y_true[i] == 1:
                TP += 1
        if Y_pred[i] != Y_true[i]:
            if Y_true[i] == 0:
                FP += 1
            elif Y_true[i] == 1:
                FN += 1
    return TP,TN,FP,FN

def precision(Y_true,Y_pred):
    TP,TN,FP,FN = confusion_matrix(Y_true,Y_pred)
    if TP+FP == 0:
        return np.inf
    return (TP)/(TP+FP)

def recall(Y_true,Y_pred):
    TP,TN,FP,FN = confusion_matrix(Y_true,Y_pred)
    return (TP) / (TP + FN)

def f1_score(Y_true,Y_pred):
    TP,TN,FP,FN = confusion_matrix(Y_true,Y_pred)
    precision_ = precision(Y_true,Y_pred)
    recall_ = recall(Y_true,Y_pred)
    return (2*precision_*recall_)/(precision_ + recall_)
```

نتیجه f1 score:

```
f1 score for weighted knn: 0.896551724137931
```

همچنین برای معیار ارور loss log از رابطه زیر استفاده می‌کنیم:

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1-y_i) \cdot log(1-p(y_i))$$

طبق فرمول بالا تابع برای محاسبه این معیار را می‌نویسیم:

```
def log_loss(y_true, y_pred):
    epsilon = 1e-15
    y_pred = np.clip(y_pred, epsilon, 1 - epsilon)
    return -np.mean(y_true * np.log(y_pred) + (1 - y_true) * np.log(1 - y_pred))
```

```
print(f'log loss for weighted knn: {log_loss(np.array(test_df.iloc[:,-1]),np.array(pred))}')
```

با توجه به اینکه مدل ما 0 و 1 پیش‌بینی می‌کند و نه احتمال، بنابراین پیش‌بینی می‌شود این معیار در ابتدا مقدار بالایی داشته باشد چرا که کاملا مطمئن خروجی می دهد که این موجب بالا رفتن خطا در این معیار می‌شود.

.3

در این بخش با استفاده از تابع پیاده سازی شده در قسمت قبلی وتابع minimize ، سعی می‌کنیم کم‌ترین مقدار خروجی تابع ارور را به ازای مقادیر مختلف وزن‌ها بدست آوریم.

```
initial_weights = np.zeros(train_df.shape[1] - 2)
result = minimize(log_loss_for_weighted_knn, initial_weights, args=(train_df, test_df, 5, 'label'),
                  method='L-BFGS-B', bounds=[(0, 2)],
                  options={'disp': True, 'eps': 1e-2})
```

قسمتی از نتایج این بخش که در حال بررسی بهترین وزن‌ها برای کم‌ترین میزان خطا می‌باشد به صورت زیر است:

```
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 6.907755278982137
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.26941485070301
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 17.269388197455335
log loss for weighted knn: 16.118095650958313
```

.4

پیش تر الگوریتم Weighted KNN به صورت کامل پیادهسازی شد (در بخش 2). در این بخش با استفاده از پیادهسازی صورت گرفته و تابع نوشته شده برای بدست آوردن error برای وزنهای مختلف، به مقایسه نتایج KNN با وزنهای 1 و KNN با وزنهای پیدا شده در قسمت پیشین میپردازیم.

```
normal_knn_error = log_loss_for_weighted_knn([1 for i in range(15)],train_df,test_df)
print(f'normal knn result error : {normal_knn_error}')

weighted_knn_error = log_loss_for_weighted_knn(optimized_weights,train_df,test_df)
print(f'weighted knn result error : {weighted_knn_error}')
```

```
log loss for weighted knn: 5.756489385732788
normal knn result error : 5.756489385732788
log loss for weighted knn: 5.756489385732788
weighted knn result error : 5.756489385732788
```

نتایج به خوبی نشان می دهند که در حالت دوم که وزنهای مناسب را پیدا کردهایم و مقدار Error کمتر است.

.5

برای این بخش ابتدا تابعی مینویسیم که به صورت رندوم 5 تا از ویژگیهای این مجموعه داده را بگیرد و لیست اسامی و اندیس های آنان را در خروجی نمایش دهد.

```python
import random
import traceback

def select_5_subset_features():
    feature_subset = []
    feature_indices = []
    for i in range(5):
        random_ = random.randint(1,15)
        feature_subset.append(f'f{random_}')
        feature_indices.append(random_)
    return feature_subset,feature_indices
```

در ادامه این تابع را 8 بار صدا می‌زنیم و در هر یک از این 8 بار صدا زدن تابع مجموعه آموزش و تست را دوباره توسط آن 5 ویژگی انتخابی رندوم می‌سازیم و آن را به ورودی می‌دهیم و میزان خطا را برای آن ویژگی‌ها با وزن‌های یکسان 1 بدست می‌آوریم و بهترین این 8 تا را انتخاب می‌کنیم.

```python
while i < 8:
    try:
        feature_s ,feature_i = select_5_subset_features()
        print(feature_s)
        feature_s.append('label')
        tr_df = train_df[feature_s]
        te_df = test_df[feature_s]
        #  weights =  optimized_weights[feature_i]
        weights = [1,1,1,1,1]
        error = log_loss_for_weighted_knn(weights,tr_df,te_df)
        if error <  best_error:
            best_error = error
            best_feature_subsets = feature_s
            best_feature_indices = feature_i
        i += 1
```

برای مثال بخشی از خروجی این کد به شکل زیر است:

```
['f8', 'f13', 'f15', 'f5', 'f13']
Reselecting
['f6', 'f4', 'f3', 'f13', 'f12']
log loss for weighted knn: 12.66432462445794
['f3', 'f10', 'f6', 'f9', 'f6']
Reselecting
['f10', 'f8', 'f9', 'f11', 'f7']
log loss for weighted knn: 13.815643824202636
['f3', 'f3', 'f2', 'f11', 'f1']
Reselecting
['f5', 'f11', 'f8', 'f15', 'f12']
log loss for weighted knn: 10.361739531463895
['f9', 'f12', 'f1', 'f10', 'f8']
log loss for weighted knn: 20.723532369423136
```

که همان‌طور که قابل مشاهده می‌باشد، دسته f2,f13,f14,f1,f3 بهترین دسته می‌باشند و مقدار خطای آنان برابر 3.4 است.

.6

در این بخش ابتدا اندیس 5 ویژگی با بالاترین ضرایب وزنی را از متغیر optimized_weights که پیش تر حساب کردیم در می‌آوریم و با استفاده از این ضرایب ویژگی‌های مهم‌تر را انتخاب می‌کنیم. از مجموعه آموزش و تست، این ویژگی‌ها را مد نظرقرار داده و مجددا خطای این روش را نیز بدست می‌آوریم. کد زیر این مراحل را نشان می‌دهد.

```python
selected_features = []
for i  in range(len(indices_of_top_5)):
    index = indices_of_top_5[i] + 1
    selected_features.append(f'f{index}')
print(selected_features)
columns = selected_features
columns.append('label')
tr_df = train_df[columns]
te_df = test_df[columns]
error = log_loss_for_weighted_knn(optimized_weights[indices_of_top_5],tr_df,te_df)
print(f'error of the selected 5 features : {error}')
```

ویژگی‌های انتخاب شده توسط این روش و میزان خطای بدست آمده از طریق این روش به صورت زیر می‌باشد:

```
['f14', 'f15', 'f7', 'f2', 'f8']
log loss for weighted knn: 9.210393678471528
error of the selected 5 features : 9.210393678471528
```

7.

در نهایت برای سه روش زیر خطای پیش‌بینی را حساب و در کنار هم گزارش و مقایسه می‌کنیم:

- **روش اول:** با داشتن ضرایب 1 برای همه ویژگی‌ها همان روش معمول KNN است.
- **روش دوم:** به صورت رندوم 5 تا از ویژگی‌ها را انتخاب کرده و با استفاده از آنها پیش‌بینی انجام داده و خطایش اندازه گیری شده است.
- **روش سوم:** با استفاده از Weighted KNN آن 5 تا ویژگی‌هایی که تشخیص داده شده باید ضرایب بزرگتری داشته باشند و مهم تر هستند را انتخاب کرده و با استفاده از آنان پیش‌بینی کرده وخطا را محاسبه کرده‌ایم.

همان طور که از نتایج مشخص است، بهترین روش با کم‌ترین میزان خطا استفاده از Weighted KNN برای انتخاب ویژگی‌های مهم بوده است و پس از آن نیز random feature selection بهتر از استفاده از تمام ویژگی هاست.

«. . . آبان‌ماه ۱۴۰۳ . . .»