پارت 1:

ابتدا دو دیتاست train و test را بارگذاری و تعریف کرده و سپس شروع به یک بررسی اولیه می کنیم تا اگر نیازی به پیش پردازش و یا تمیز کردن دیتا بود، عملیات های لازم را اعمال کنیم.

دیتای ما شامل 21 ستون می‌باشد که هیچ مقدارnull ی ندارد.

تمام ستون‌ها از نوع عددی(int) می‌باشند و هیچ مقدار رشته‌ای(string) نداریم، نتیجتا دیتای ما نیاز به آماده سازی خاصی ندارد.

در نگاه اولمتوجه می‌شویم که رم و قدرت باطری بیشترین ارتباط را با رنج قیمت دارند.

خروجی دیتای خام ما برای نمایش اطلاعات فایل train:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	4	19	1	1	0	0
96	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	10	16	1	1	1	2
97	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	1	5	1	1	0	3
98	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	10	19	1	1	1	0
99	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919	19	4	2	1	1	1	3

خروجی دیتای خام ما برای نمایش اطلاعات فایل test:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7	2	0	1	0
	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0	7	1	0	0
	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10	10	0	1	1
	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0	7	1	1	0
	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8	7	1	0	1
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	996	1700	1	1.9	0	0	1	54	0.5	170	...	17	644	913	2121	14	8	15	1	1	0
96	997	609	0	1.8	1	0	0	13	0.9	186	...	2	1152	1632	1933	8	1	19	0	1	1
97	998	1185	0	1.4	0	1	1	8	0.5	80	...	12	477	825	1223	5	0	14	1	0	0
98	999	1533	1	0.5	1	0	0	50	0.4	171	...	12	38	832	2509	15	11	6	0	1	0
99	1000	1270	1	0.5	0	4	1	35	0.1	140	...	19	457	608	2828	9	2	3	1	0	1

:2

اولی رو گذاشتم پرایس رنج به عنوان تارگت و فیچر ram رو انتخاب کردم به دلایل منطقی میتونیم از یه دونه هم بیشتر انتخاب کنیم ولی حدود هشتاد درصد صحت داشت.

:3

تحلیل مولفه اساسی (PCA) یک «روش تبدیل خطی» ساده و در عین حال محبوب و کارآمد محسوب می‌شود.

تحلیل مولفه اساسی (PCA) قصد دارد همبستگی بین متغیرها را شناسایی کند.

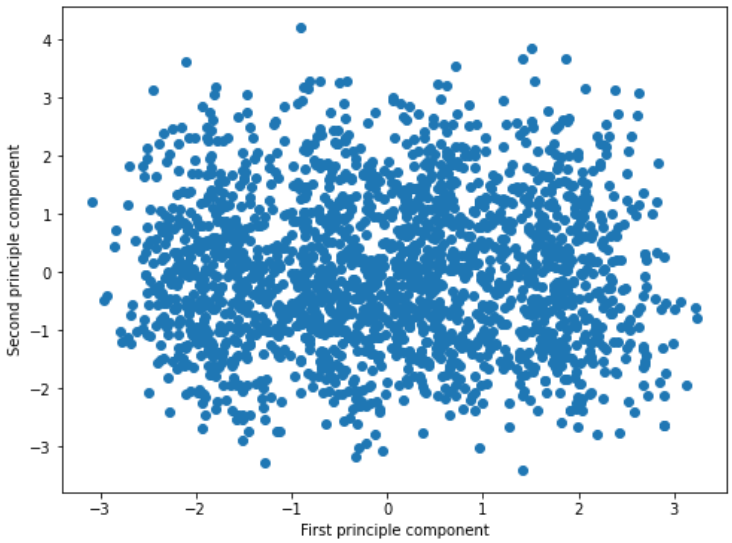
اگر یک همبستگی قوی بین متغیرها وجود داشته باشد، می توانیم اغدام به کاهش ابعاد معنادار کنیم. به طور کلی ، آنچه در PCA به وقوع می‌پیوندد پیدا کردن جهت واریانس بیشینه در داده‌های ابعاد بالا و طرح‌ریزی کردن آن در زیرفضایی با ابعاد کمتر به طوری است که بیشترین اطلاعات حفظ شوند.

شرط استفاده از PCA این است که دیتایمان را اسکیل کرده باشیم.

دیتای اسکیل شده ی ما در تصویر زیر آورده شده است که ستون ها بر اساس مقدار امتیاز هایی مقدار دهی شده اند:

	0	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	
	02597	-0.990050	0.830779	-1.019184	-0.762495	-1.043966	-1.380644	0.340740	1.349249	-1.101971	...	-1.408949	-1.146784	0.391703	-0.784983	0.283103	1.462493	-1.786861	-1.006018	0.986097	-0
	95139	1.010051	-1.253064	0.981177	-0.992890	0.957886	1.155024	0.687548	-0.120059	-0.664768	...	0.585778	1.704465	0.467317	1.114266	-0.635317	-0.734267	0.559641	0.994018	-1.014099	0
	37686	1.010051	-1.253064	0.981177	-0.532099	0.957886	0.493546	1.381165	0.134244	0.209639	...	1.392684	1.074968	0.441498	-0.310171	-0.864922	-0.368140	0.559641	0.994018	-1.014099	0
	19319	1.010051	1.198517	-1.019184	-0.992890	-1.043966	-1.215274	1.034357	-0.261339	0.646842	...	1.286750	1.236971	0.594569	0.876859	0.512708	-0.002014	0.559641	-1.006018	-1.014099	0
	25906	1.010051	-0.395011	-1.019184	2.002254	0.957886	0.658915	0.340740	0.021220	-1.101971	...	1.268718	-0.091452	-0.657666	-1.022389	-0.864922	0.730240	0.559641	0.994018	-1.014099	-0
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19	11860	1.010051	-1.253064	0.981177	-0.992890	0.957886	-1.656260	1.034357	-0.967737	0.646842	...	1.300273	1.477661	-1.342799	0.164641	-0.405712	1.462493	0.559641	0.994018	-1.014099	-1
19	53694	1.010051	1.321096	0.981177	-0.992890	-1.043966	0.383299	-1.046495	1.320993	-0.227564	...	0.608317	1.651235	-0.085031	-0.310171	0.971917	0.913303	0.559641	0.994018	0.986097	0
19	30773	-0.990050	-0.762748	0.981177	-0.762495	0.957886	0.217930	0.687548	-0.911225	1.521249	...	0.502383	0.880565	0.860139	-0.784983	-1.094526	-1.100394	0.559641	0.994018	-1.014099	1
19	22527	-0.990050	-0.762748	-1.019184	-0.071307	0.957886	0.769162	-1.393304	0.134244	0.209639	...	-0.696707	-1.345816	-1.157454	1.351672	0.971917	1.462493	0.559641	0.994018	0.986097	-1
19	58331	1.010051	0.585621	0.981177	0.159088	0.957886	0.714039	1.381165	0.784130	0.646842	...	-0.365380	-1.151413	1.655004	1.589078	-0.405712	-1.649584	0.559641	0.994018	0.986097	1

در انتها خروجی بر روی نمودار به این شکل است:



:4

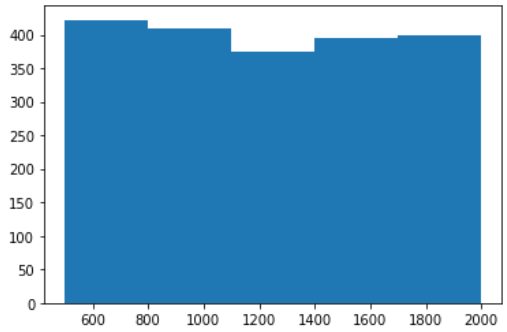
	precision	recall	f1-score	support
0	0.92	0.93	0.92	105
1	0.85	0.82	0.84	91
2	0.87	0.92	0.89	92
3	0.99	0.95	0.97	112
accuracy			0.91	400
macro avg	0.91	0.91	0.91	400
weighted avg	0.91	0.91	0.91	400

:6-1

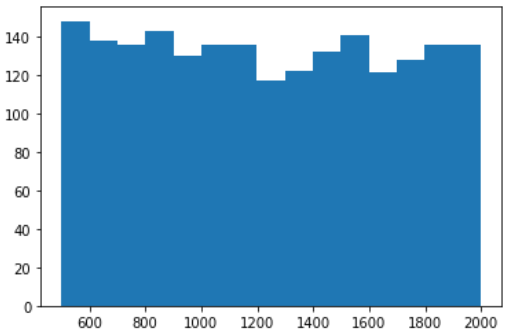
منطق این تکنیک این است که داده ها رو میشه بر اساس همسایه هاش هموار سازی کرد .به عبارت دیگه این تکنیک نگاه میکنه که همسایه های یک داده چطوری هست و سعی میکنه داده رو شبیه همسایه هاش کنه. اگر یک داده با همسایه هاش زیاد فرق داشته باشه نشون دهنده اینکه داده نویزی هستش و باید هموار سازی روش انجام بشه.

نکته مهم در مورد این روش آن است که این روش در مورد داده های عددی کاربرد دارد که خوشبختانه داده های ما هم عددی است.

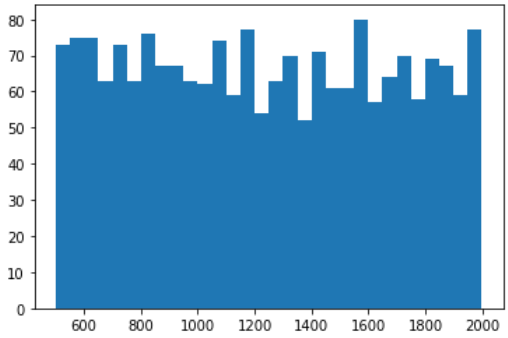
تصویر خروجی مربوط به bins=5:



تصویر خروجی مربوط به bins=15:



تصویر خروجی مربوط به bins=30:



attery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range	bin
842	0	2.2	0	1	0	7	0.6	188	2	...	756	2549	9	7	19	0	0	1	1	
1021	1	0.5	1	0	1	53	0.7	136	3	...	1988	2631	17	3	7	1	1	0	2	
563	1	0.5	1	2	1	41	0.9	145	5	...	1716	2603	11	2	9	1	1	0	2	
615	1	2.5	0	0	0	10	0.8	131	6	...	1786	2769	16	8	11	1	0	0	2	
1821	1	1.2	0	13	1	44	0.6	141	2	...	1212	1411	8	2	15	1	1	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
794	1	0.5	1	0	1	2	0.8	106	6	...	1890	668	13	4	19	1	1	0	0	
1965	1	2.6	1	0	0	39	0.2	187	4	...	1965	2032	11	10	16	1	1	1	2	
1911	0	0.9	1	1	1	36	0.7	108	8	...	1632	3057	9	1	5	1	1	0	3	
1512	0	0.9	0	4	1	46	0.1	145	5	...	670	869	18	10	19	1	1	1	0	
510	1	2.0	1	5	1	45	0.9	168	6	...	754	3919	19	4	2	1	1	1	3	

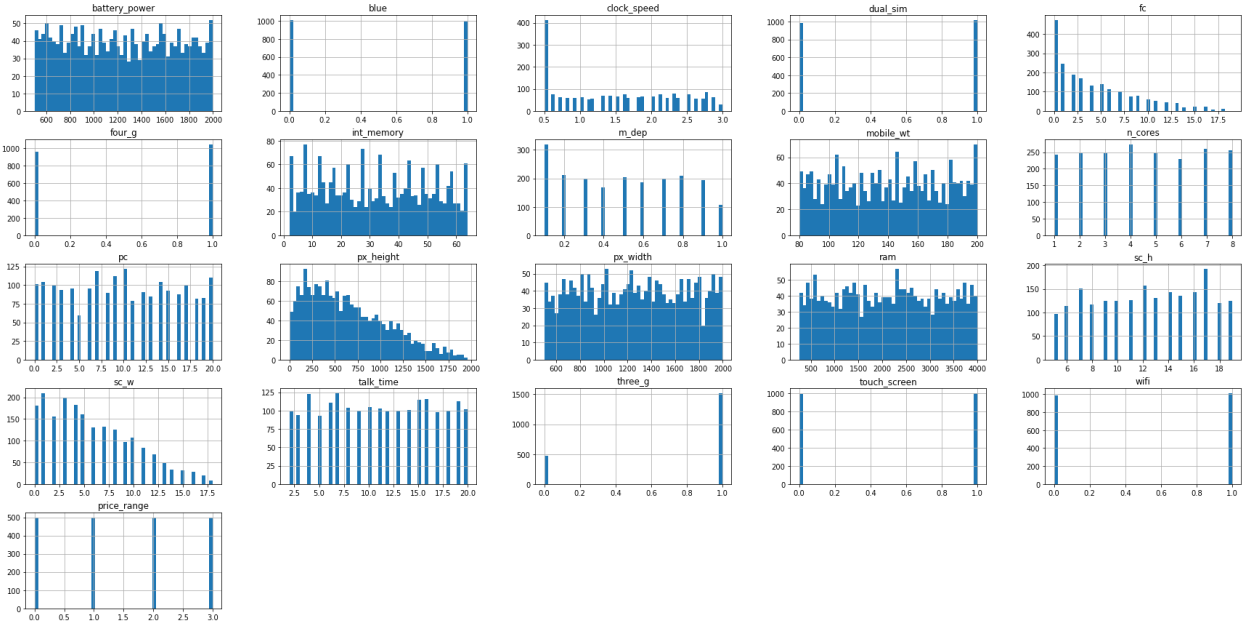
6-2:

کدبندی One-Hot ستون‌های دودویی (binary) جدیدی می‌سازد که هر یک مربوط به یکی از مقادیری هستند که متغیر به خود می‌گیرد. این نوع رمزگذاری یک ویژگی باینری جدید برای هر دسته ممکن ایجاد می کند و مقدار 1 را به ویژگی هر نمونه که با دسته اصلی آن مطابقت دارد اختصاص می دهد.

می‌توان با استفاده از آن به راحتی تغییر مقیاس داد. با استفاده از مقادیر عددی، ما به راحتی احتمالی را برای مقادیر خود تعیین می کنیم.

6-3:

تبدیل log مورد استفاده برای تبدیل داده‌های کج‌شده به انطباق تقریباً با نرمال است. اگر داده های اصلی از یک توزیع log-normal یا تقریباً مشابه پیروی کنند، آنگاه داده های تبدیل شده با log از توزیع نرمال یا نزدیک به نرمال پیروی می کنند.



:6-4

دو ستون جدید برای مساحت و حجم ایجاد می‌کنیم.

	m_dep	px_height	px_width	Area px	volume cm
0	0.6	20	756	15120	6.350794
1	0.7	905	1988	1799140	881.633259
2	0.9	1263	1716	2167308	1365.488697
3	0.8	1216	1786	2171776	1216.269966
4	0.6	1208	1212	1464096	614.958446
...	...	...	...	...	...
1995	0.8	1222	1890	2309580	1293.444990
1996	0.2	915	1965	1797975	251.732107
1997	0.7	868	1632	1416576	694.165277
1998	0.1	336	670	225120	15.759377
1999	0.9	483	754	364182	229.448885

:8

بحث bootstrapping ، کلیت ایدش این است که توی طراحی مدل یا یه آزمایش یه چیزی رو به صورت رندوم تغییر بدید و در نهایت از همه این نتیجه ها استفاده کنید(مثلا میانگین بگیرید) از دید یادگیری ماشین و آمار، این چیزی که می خوایم رندوم باشه می تونه، دیتاست باشه، می تونه خوده

مدل باشه)مثلا توی خوده شبکه عصبی، می تونید هر بار با یه initialization مجزا، یا learning rate های مختلف شبکه رو آموزش بدید یا ساختار شبکه رو عوض کنید و....

توی baggin ما در واقع چندتا دیتاست، از یه دیتاست ایجاد می کنیم و مدلهامون را روش آموزش می دهیم. و بعدش نتایج همه این مدل ها رو توی پیش بینی استفاده می کنیم.

اما توی k-fold Cross Validation ما در واقع یه دیتاست رو به چندین قسمت تقسیم می کنیم و یه قسمت رو می ذاریم کنار، روی یه قسمت آموزش میدیم و بعدش روی قسمتی که گذاشتیم کنار دقت رو اندازه می گیریم. که بیشتر به این خاطره که با دقت بیشتری بفهمیم مدل ما چه طوری عمل می کنه و بحثش برای افزایش عملکرد مدل نیست. و در نهایت ما یک مدل به دست میاریم.

:9

5x2 cross-validation نوع خاصی از cross-validation تودرتو است.

اشاره دارد به ۵ repetition از یک 2folde.

راهی به منظور انجام آزمون‌های آماری و همچنین برآورد خوبی از واریانس آن خطاها است.

:10

در تحلیل خوشه‌ای، روش elbow یک روش اکتشافی است که در تعیین تعداد خوشه‌ها در یک مجموعه داده استفاده می‌شود. این روش شامل ترسیم تغییرات توضیح داده شده به عنوان تابعی از تعداد خوشه ها، و انتخاب eEElbow منحنی به عنوان تعداد خوشه های مورد استفاده است. اما روش elbow فقط یک ویژگی خوشه‌بندی جهانی را اندازه‌گیری می‌کند.

پارت 2:

برای آماده سازی دیتا برای بخش 1 ما نیاز داریم تا متغییر رشته ای heatingType را تبدیل به مقدار عددی کنیم به این شکل که هر عدد نماینده ی یک رشته است.

حال ما برای پاکسازی داده دو راه حل داریم که یکی پاک کردن آن ردیف ها است و دیگری بدست آوردن آن متغییر های پوچ با استفاده از میانه و میانگین و واریانس.

بهتر است که از جایگذاری آنها با میانگین و میانه و دیگر مقادیر خودداری کنیم. چرا که اولا در ستون دوم 37947 و در ستون سوم 26449 مقدار از دست رفته داریم و همچنین ستون دوم از نوع شئ می‌باشد و احتمالا جایگذاری مقادیر از دست رفته با میانه و میانگین کار ما را سخت تر میکند. در نتیجه ما ردیف هایی که مقادیر از دست رفته دارند را حذف می‌کنیم.

:1

برای به دست آوردن مقدار MSE باید هرمقدار پیشبینی شده را از مقدار مشاهده شده کم کرده و به توان دو برسانیم و مجموع این تفاضلات را جمع کرده و تقسیم بر تعداد کنیم که فرمول روبه رو را بدست میاوریم:

MSE formula = (1/n) * Σ(actual – forecast)^2

: ridge-3

رگرسیون ریج یک نسخه معمولی از رگرسیون خطی است. رگرسیون ریج الگوریتم های یادگیری ماشین را قادر می سازد تا نه تنها با داده ها مطابقت داشته باشند ، بلکه وزن مدل را تا حد ممکن کوچک نگه دارند.

مقیاس داده ها با استفاده از استاندارد مقیاس قبل از استفاده از رگرسیون ریج ضروری است ، زیرا به مقیاس ویژگی های ورودی حساس است. اکنون بیایید از طریق الگوریتم رگرسیون ریج به درک نحوه منظم سازی یک مدل Liner با استفاده از الگوریتم Ridge بپردازیم.

: lasso-3

این الگوریتم یک نوع دیگه از منظم سازی Linear Regression هست. مثل Ridge یک مقدار Regularizationبه تابع هزینه اضافه میکنه. یک ویژگی مهم این الگوریتم این هست که وزن های فیچر های کم اهمیت رو حذف میکنه (برابر صفر قرار میده) به عبارت دیگه Lasso Regression به طور خودکار Feature Selection رو انجام میده.