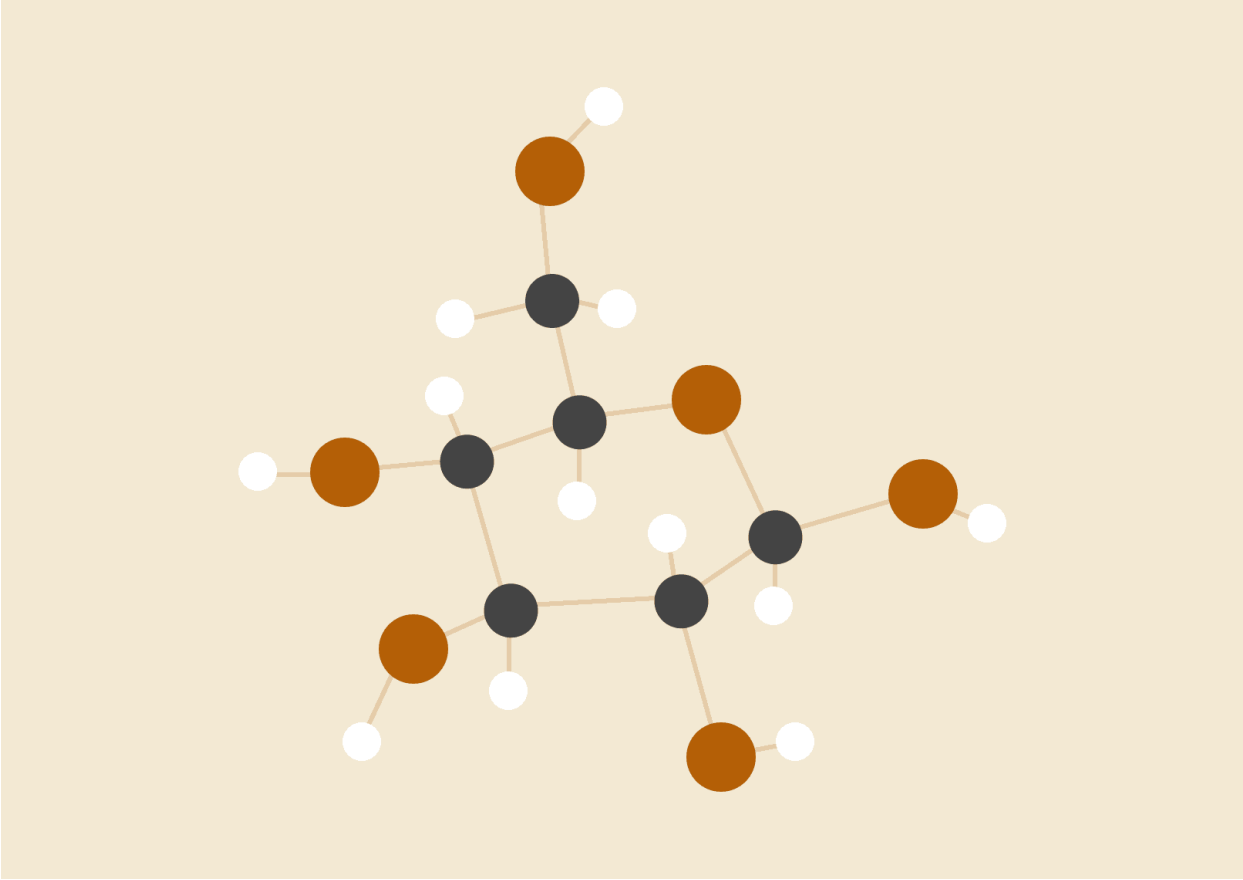
homework-3

Data mining



Alireza Javaheri

99422008

قسمت اول

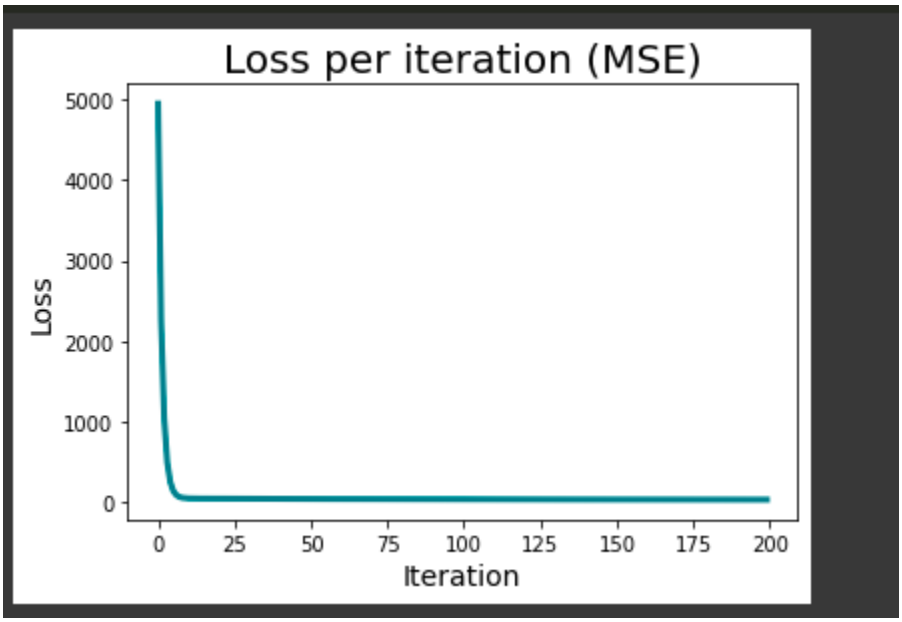
در این قسمت یه کلاس به نام MyLinearRegression رو نوشتیم که الگوریتم linear regression رو اجرا میکند که بدین شکل است.

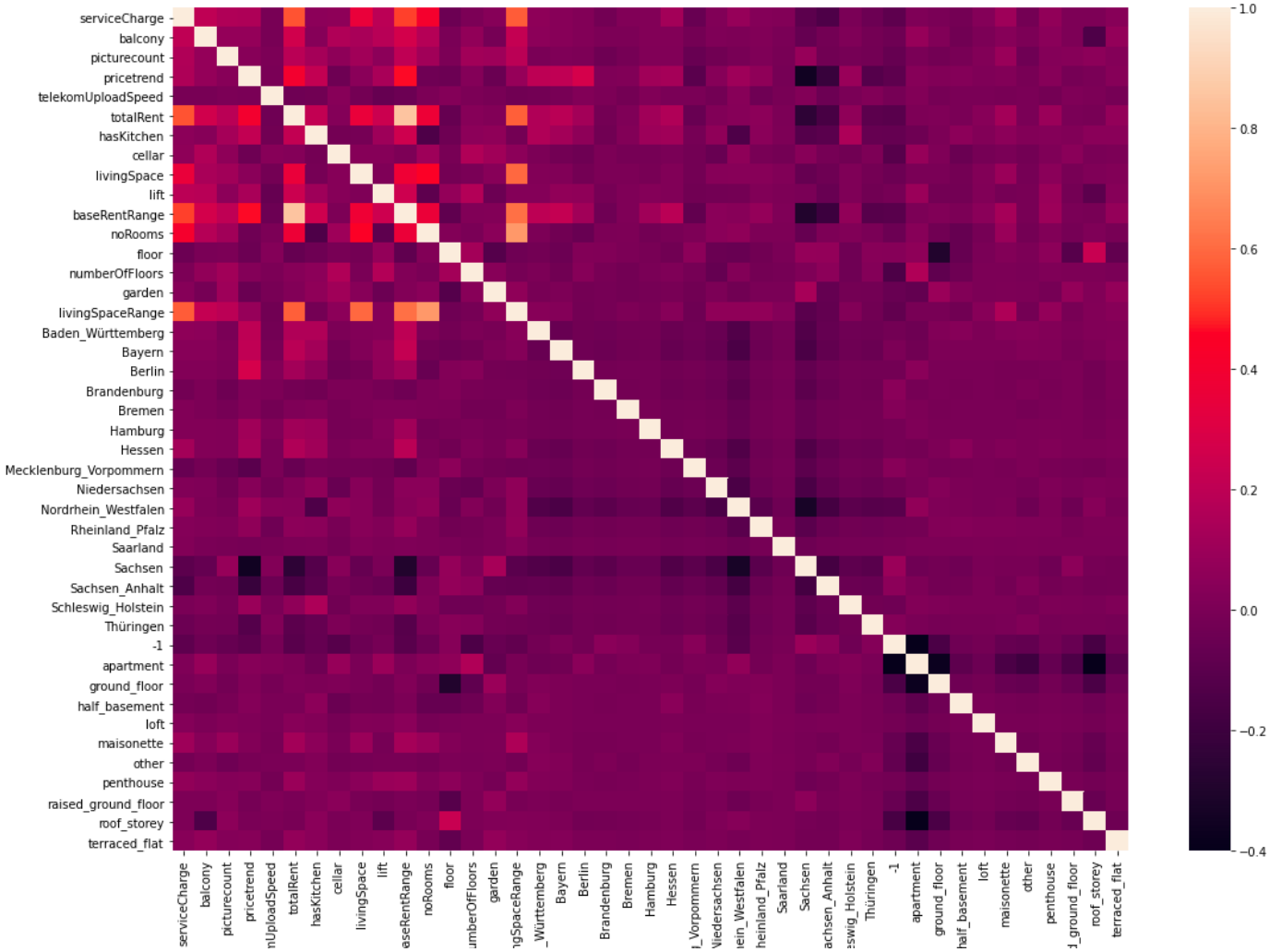
```python
class MyLinearRegression:
    '''
    A class which implements linear regression model with gradient descent.
    '''
    def __init__(self, learning_rate=0.01, n_iterations=1000):
        self.learning_rate = learning_rate
        self.n_iterations = n_iterations
        self.weights, self.bias = None, None
        self.loss = []

    @staticmethod
    def _mean_squared_error(y, y_hat):
        '''
        Private method, used to evaluate loss at each iteration.

        :param: y - array, true values
        :param: y_hat - array, predicted values
        :return: float
        '''
        error = 0
        for i in range(len(y)):
            error += (y[i] - y_hat[i]) ** 2
        return error / len(y)
```

```
    def fit(self, X, y):
        '''
        Used to calculate the coefficient of the linear regression model.

        :param X: array, features
        :param y: array, true values
        :return: None
        '''
        # 1. Initialize weights and bias to zeros
        self.weights = np.zeros((X.shape[1], 1))
        self.bias = 0

        # 2. Perform gradient descent
        for i in range(self.n_iterations):
            # Line equation
            y_hat = np.dot(X, self.weights) + self.bias
            loss = self._mean_squared_error(y, y_hat)
            self.loss.append(loss)

            # Calculate derivatives
            partial_w = (1 / X.shape[0]) * (2 * np.dot(X.T, (y_hat - y)))
            partial_d = (1 / X.shape[0]) * (2 * np.sum(y_hat - y))

            # Update the coefficients
            self.weights -= self.learning_rate * partial_w
            self.bias -= self.learning_rate * partial_d
```

```
    def predict(self, X):
        '''
        Makes predictions using the line equation.

        :param X: array, features
        :return: array, predictions
        '''
        return np.dot(X, self.weights) + self.bias

    def score():
      return r2_score(self.predict(x), y)
```

در حالت اول ورودی دو تا فیچر با بیشترین کورولیشن است (noRooms, livingSpaceRange) که نتایج آن به صورت زیر است.


Loss per iteration (MSE)
Loss
5000
4000
3000
2000
1000
0
0
25
50
75
100
125
150
175
200
Iteration



حالت دوم

همین الگوریتم رو با پکیج sklearn انجام میدیم . با همین دو تا فیچر که با بیشترین کورولیشن با livingSpace رو انجام دادیم با fold-5 و fold-10 زدیم که cross_val_score آن رو به صورت زیر است.

5-fold:

```
array([0.92911069, 0.9424413 , 0.93473943, 0.92943177, 0.92063566])
```

10-fold:

```
array([0.9268104 , 0.93174509, 0.941755  , 0.94136537, 0.95062757,
       0.90637917, 0.93569861, 0.91988941, 0.92677602, 0.91494598])
```

حالت سوم

مدل با استفاده از sklearn با ورودی های ۲ فیچر با بیشترین مقدار و ۲ فیچر با کمترین مقدار کورولیشن (۴ تا فیچر) و تارگت متراژ خونه

فیچر noRooms و livingSpaceRange که بیشترین کورولیشن و فیچر های Sachsen و hasKitchen

5-fold:

```
array([0.93174371, 0.93240715, 0.9238055 , 0.92382117, 0.73906063])
```

10-fold:

```
array([0.93249262, 0.93097445, 0.93028463, 0.93423912, 0.92567815,
       0.92217726, 0.92802837, 0.92001469, 0.92863616, 0.62553785])
```

حالت چهارم

به عنوان فیچر دلخواه کل فیچر ها رو به مدل دادیم

5-fold:

```
array([0.93414199, 0.93730446, 0.93016454, 0.92464568, 0.74040144])
```

10-fold:

```
array([0.93326217, 0.93501918, 0.93539142, 0.93916946, 0.93191536,
       0.92876727, 0.93107338, 0.91932932, 0.93148848, 0.6240881 ])
```

حالت پنجم

با استفاده از Ridge روی فیچر های دلخواه

5-fold:

```
array([0.90725953, 0.93277911, 0.11835627, 0.93343634, 0.93443261])
```

10-fold:

```
array([0.88503201, 0.93225167, 0.93310587, 0.93343942, 0.07083868,
       0.93083049, 0.93372592, 0.93405594, 0.93516067, 0.93476537])
```

حالت ششم

با استفاده از lasso روی فیچر های دلخواه

5-fold:

```
array([0.90626611, 0.93178448, 0.11806386, 0.93201498, 0.93343159])
```

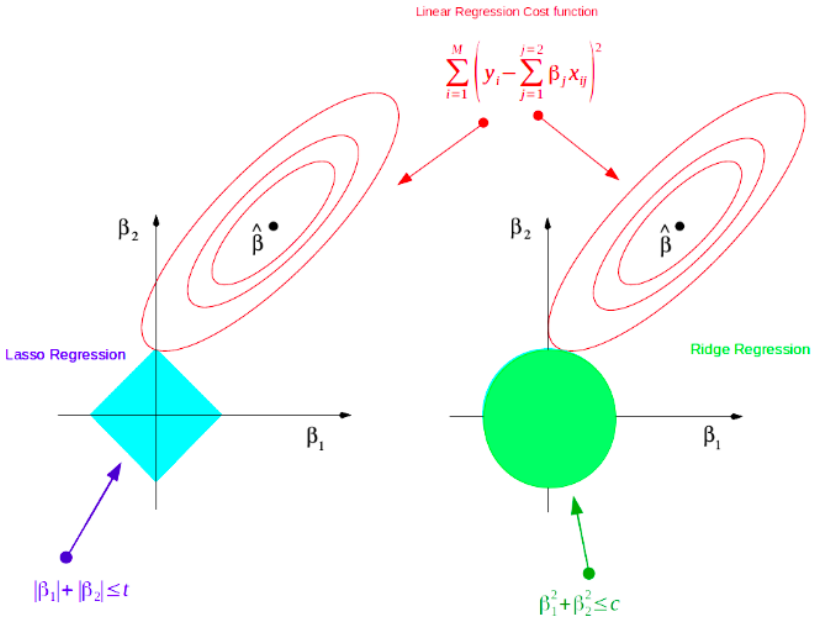
10-fold:

```
array([0.88358494, 0.93105005, 0.93143626, 0.93248182, 0.07067982,
       0.93013867, 0.93225244, 0.93208472, 0.93367574, 0.9334326 ])
```

قسمت دوم

۱ - lasso و ridge در واقع regularization هستند که برای جلوگیری از overfit شدن شبکه مورد استفاده قرار میگرد. که نرم اول یا دوم وزن ها رو به عبارت loss function اضافه میکنند و وقتی ترم مثبتی به loss function اضافه شود الگوریتم برای کم کردن اثر این مقدار مثبت مقدار وزن ها را کاهش میدهد که این باعث میشود که بعضی از وزن ها به سمت صفر میل کند و از complexity مدل میکاهد که باعث جلوگیری از overfiting میشود.

Ridge نرم دوم وزن ها رو اضافه مکیند در صورتی که lasso نرم اول رو و lasso نسبت به ridge تعداد وزن هایی رو به صفر میکند و باعث ایجاد sparcity میشود در صورتی که در ridge باعث ایجاد اسپارسیتی نمیشود



2 - برای انتخاب پارامتر اثر regularization میتوان با کمک cross validation تمامی مقادیر برای این پارامتر را تست کرد.

3- هیچ فرمول خاصی برای انتخاب تعداد فولد ها وجود ندارد اما به صورت کلی یک trade off بین واریانس و بایاس در انتخاب فولد ها وجود دارد به صوری که هر جه تعداد فولدها افزایش پیدا کنه واریانس زیاد و بایاس کم میشود و برعکس تعداد فولد ها کم شود واریانس کم و بایاس زیاد میشود.

4 - Leav one out نیز یک روش cross validation است با این تفاوت که در این روش هر داده یک فولد است پس در هر سری فقط یک داده به عنوان تست در نظر گرفته میشو (به خاصر همین اسمش leave one out) است. همانظور که در بالا گفتیم باعث میشه واریانس زیاد بشه و بایاس خیلی کم بشه و برای دیتاست ها کوچک مناسب است.

5 - این یک روش نمونه گیری است که در آن یک داده ممکن است چندبار انتخاب شود و فرق آن با کراس ولیدیشن این است که در کراس ولیدیشن از داده های تکراری نمونه گیری نمیشود

در machine learning از این روش بیشتر در ensemble learning ها مثل random forest بیشرت

استفاده میشود اما علاوه بر این میتوان همانند cross validation برای محاسبه خطای مدل از آن استفاده کرد بدین صورت که داده هایی که در bootstrap انتخاب نشدند رو به عنوان داده های تست در نظر میگریم.

6- یعنی 5 بار از 2 fold cross validation استفاده کنیم

در واقع در این روش همزمان فرآیند انتخاب مدل و تیون کردن پارامترها انجام میش و برای مدل هایی که میخوایهم پارامتر های رو تیون کنیم بهتر است از این روش استفاده کنیم.

7- از این روش در روش early stopping استفاده کرده ایم که در آن وقتی از یک جایی به بعد نمودار لاس یا دقت آموزش و تست واگرا شوند در آنجا متوقف میشویم چون داره overfit میشود که نشان دهنده همین elbow است.

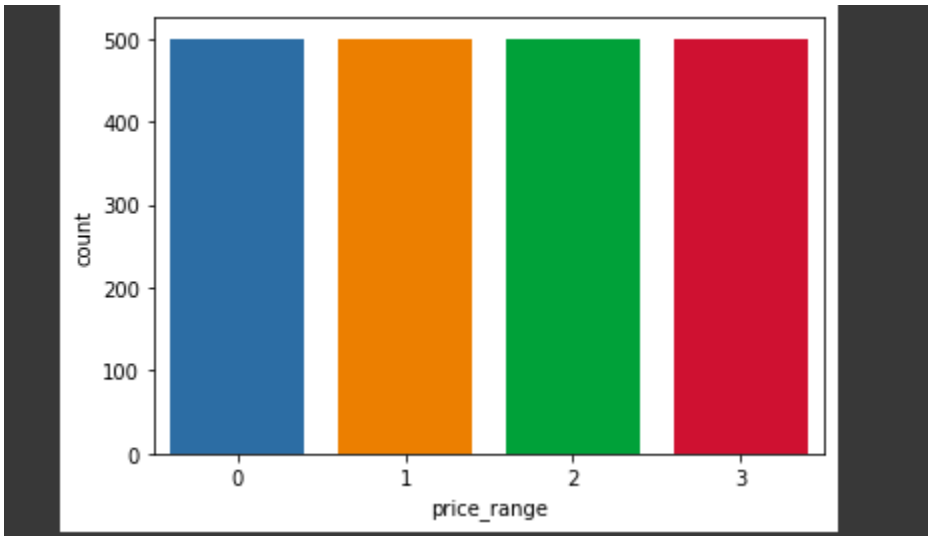
از انجایی هر تابع خطا حاصل جمع خطای bias و variance هست همواره میتوان از این روش استفاده کرد.

قسمت سوم

با استفاده از پکیج sklearn با استفاده از تمامی فیچر ها عمل classification رو انجام دادیم که نتیجه آن به صورت زیر است

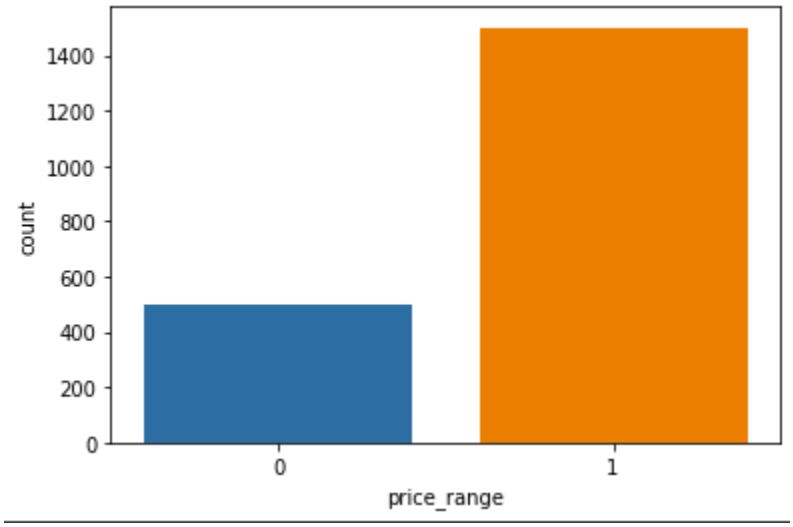
	precision	recall	f1-score	support
0	0.82	0.91	0.86	500
1	0.66	0.52	0.58	500
2	0.56	0.44	0.49	500
3	0.68	0.90	0.77	500
accuracy			0.69	2000
macro avg	0.68	0.69	0.68	2000
weighted avg	0.68	0.69	0.68	2000

در این دیتاست ستون price-range متوازن است که نمودار آن به صورت زیر است



تمامی نمونه های که دارای کلاس 1, 2 ,3 هستند رو 1 و نمونه با لیبل 0 رو دست نزدیم .بعد از این کار دو تا لبل داریم و سپس عمل classification رو انجام دادیم که نتیجه آن به صورت زیر است.

	precision	recall	f1-score	support
0	0.88	0.84	0.86	500
1	0.95	0.96	0.95	1500
accuracy			0.93	2000
macro avg	0.91	0.90	0.91	2000
weighted avg	0.93	0.93	0.93	2000

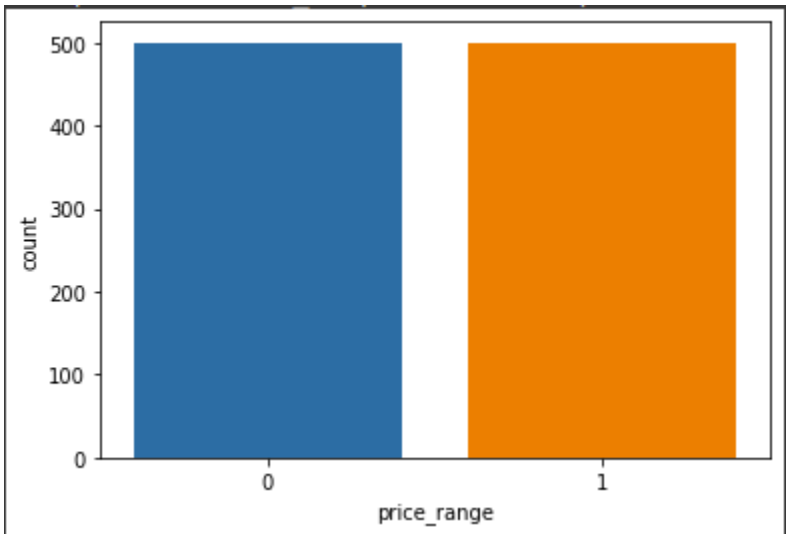


همان طور ه در نمودار بالا میبینم داده ها با تغیرات گفته شده imbalance شده اند. در داده های imbalance چون توزیع یک کلاس خیلی کمتر از کلاس دیگری است ممکن است مدل است الگوری برای پیدا کردن کلاس با توضیع کم را یادنگیرد . که برای این کار از روش های مختلفی استفاده میشود که یکی از آن ها استفاده کرد ما در اینجا از روش هاس sampling استفاده کردیم

روش under sampling:

در این روش تعداد داده های با کلاس زیاد رو کم میکنیم . از بدی های این روش از دست رفت داده ها میباشد.

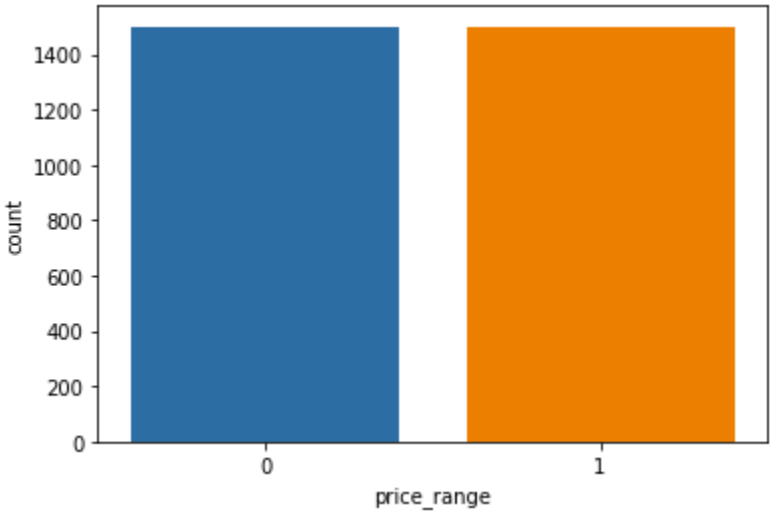
نتایج آن به صورت زیر است.



	precision	recall	f1-score	support
0	0.92	0.92	0.92	500
1	0.92	0.92	0.92	500
accuracy			0.92	1000
macro avg	0.92	0.92	0.92	1000
weighted avg	0.92	0.92	0.92	1000

روش over-sampling:

در روش برخلاف روش قبلی داده های کلاسی که کمتر است رو با روش های مختلف زیاد مکنیم. نتایج آن به صورت زیر است.



	precision	recall	f1-score	support
0	0.92	0.93	0.93	1500
1	0.93	0.92	0.93	1500
accuracy			0.93	3000
macro avg	0.93	0.93	0.93	3000
weighted avg	0.93	0.93	0.93	3000

از روش forward selection برای انتخاب feature ها استفاده کردیم و سپس داده ها با فیچر های انتخاب شده رو به مدل دادیم که نتایج آن به صورت زیر است.

	precision	recall	f1-score	support
0	0.95	0.94	0.94	500
1	0.98	0.98	0.98	1500
accuracy			0.97	2000
macro avg	0.96	0.96	0.96	2000
weighted avg	0.97	0.97	0.97	2000

در این قسمت pca رو روی داده ها اعمال کردیم که نایج آن به صورت زیر است.

	precision	recall	f1-score	support
0	0.87	0.94	0.90	500
1	0.98	0.95	0.97	1500
accuracy			0.95	2000
macro avg	0.92	0.95	0.93	2000
weighted avg	0.95	0.95	0.95	2000

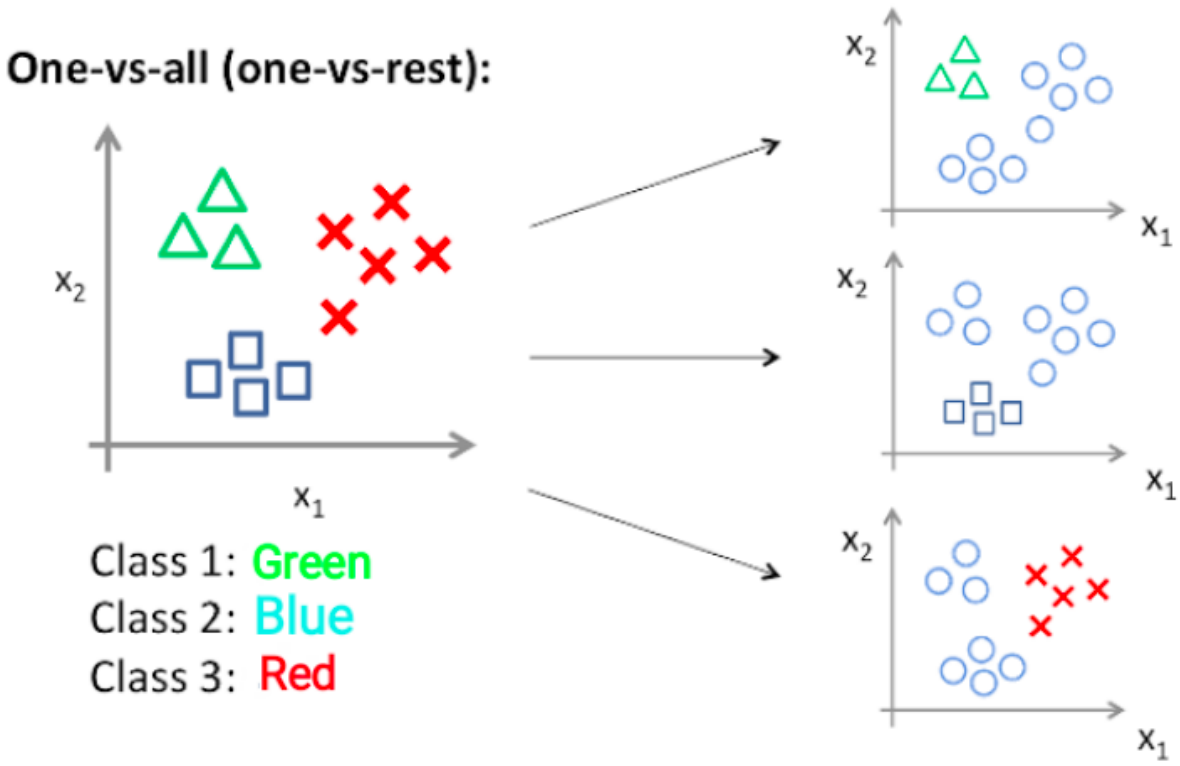
با استفاده از روش backward فیچرها رو انتخاب کردیم که نتایج آن به صورت زیر است.

	precision	recall	f1-score	support
0	0.89	0.85	0.87	500
1	0.95	0.97	0.96	1500
accuracy			0.94	2000
macro avg	0.92	0.91	0.91	2000
weighted avg	0.94	0.94	0.94	2000

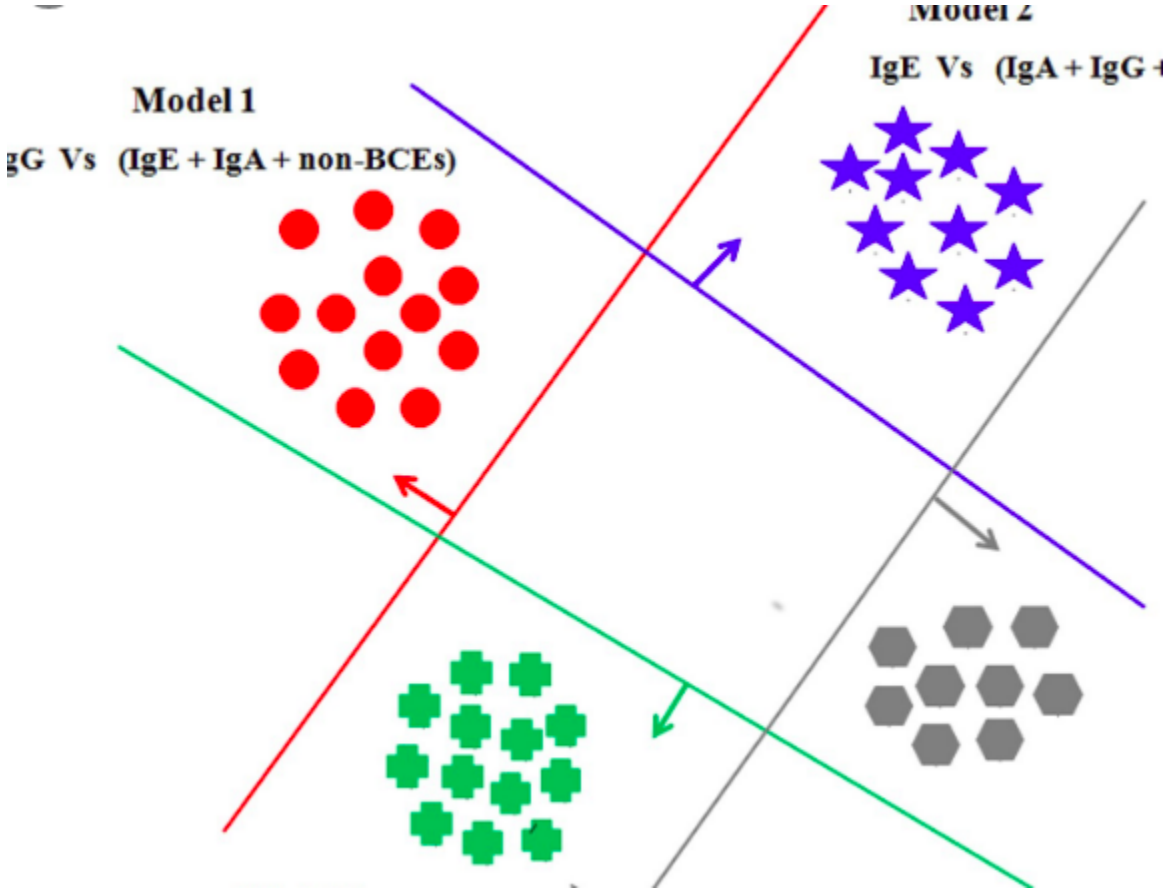
بخش چهارم

1 - از دو روش میتنویم استفاده کنیم

One-vs-rest : بقیه کلاس ها رو یکی در نظر میگریم و کلاس مورد نظر رو با بقیه دسته بندی مکنیم . یعنی به تعداد دسته ها باید کلاس بندی کنیم و دسته بندی که بیشترین دقت رو داشت کلاس مورد نظر است.



One-vs-one: همانند روش قبلی multi classification رو به binary classification تبدیل مکیند. اما در اینجا یکی از کلاس ها در مقابل بقیه نمیگذارد بلکه دو به دو عمل دسته بندی رو انجام میدهد.



۲ - تفاوت محسوس دیده نمیشود.

۳- ما به دنبال این هستیم برای کاهش بار محاسباتی تا حد امکان فیچر های کمتری که البته به دقت مدل هم صدمه نزنن انتخاب کنیم. با استفاده از روش های feature selection میتوانیم این هدف رو برطرف کنیم.

۴ - میتوان از روش های best subset selection و automatic recommendation method استفاده کرد.

۵-هر دو تضمین نمیکنند که بهتری subset feature رو بدن . اگر تعداد داده ها ازفیچر ها بیشتر بود روش backward selection و اگه برعکس بود روش forward بهتره.

برای بهبود میتوان ترکیبی از دو تا رو استفاده کرد .

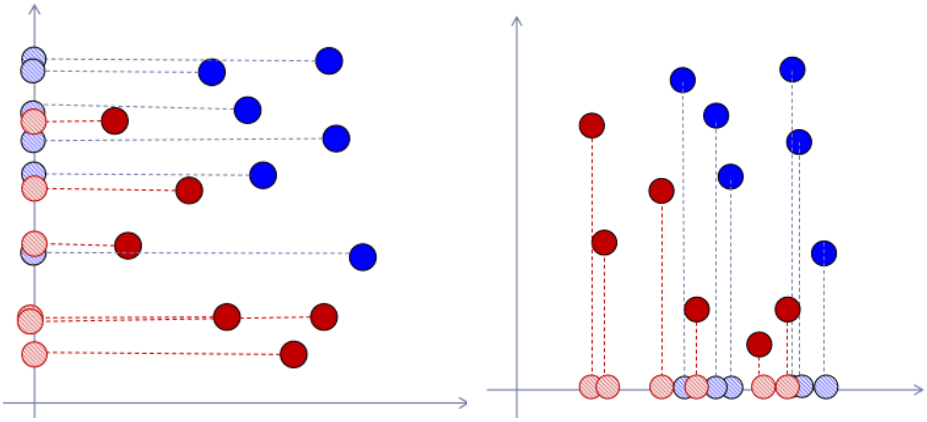
:LDA - 6

Linear discriminant analysis یک روش کاهش بعد است که از آن برای مسائل دسته بندی نیز میتوان استفاده کرد. در اینجا به توضیح مدل خطی آن می پردازیم.

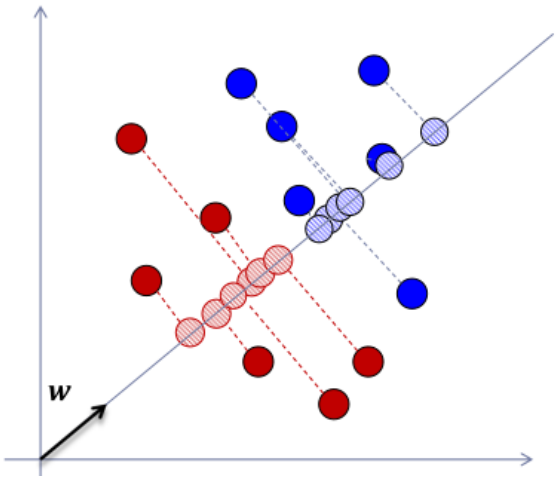
برای مسئله کاهش بعد هدف پیدا کردن ترکیبی خطی از ویژگی هاست به طوری که پراکندگی بین کلاس ها از پراکندگی بین کلاس ها بیشتر باشد.

در مسئله دسته بندی همانطور که گفته شد هدف پیش بینی کردن کلاس داده x است بدین صورت که ابتدا داده ها رو به یک فضایی با ابعاد کوچکتر پروجکت (تصویر) میکنیم سپس داده x در این فضای جدید رو طبقه بندی میکنیم.

در LDA هدف پیدا کردن راستای مناسب برای پروجکت کردن داده بر آن است . فرض کنیم یک داده دو بعدی داریم مانند شکل زیر اگر داده ها رو هر یک از محورها تصویر کنیم به وضوح کمکی برا جداسازی داده های دو کلاس به ما نمیکند.



اما اگر داده ها را در راستای w در شکل زیر تصویر کنیم داده ها ی دو کلاس به صورت مناسب از هم تفکیک میشوند.



پس هدف در مسئله lda پیدا کردن بهترین راستا است که داده ها را در آن پروجکت کنیم.

روش اول این است که جوری داده های کلاس رو به فضای جدید تصویر کنیم که فاصله میانگین داده های در فضای جدید ازهم بیشینه شود یعنی

$$\max_{\boldsymbol{w}} J(\boldsymbol{w}) = (\mu_1' - \mu_2')^2$$
$$\text{s.t.} \|\boldsymbol{w}\| = 1$$

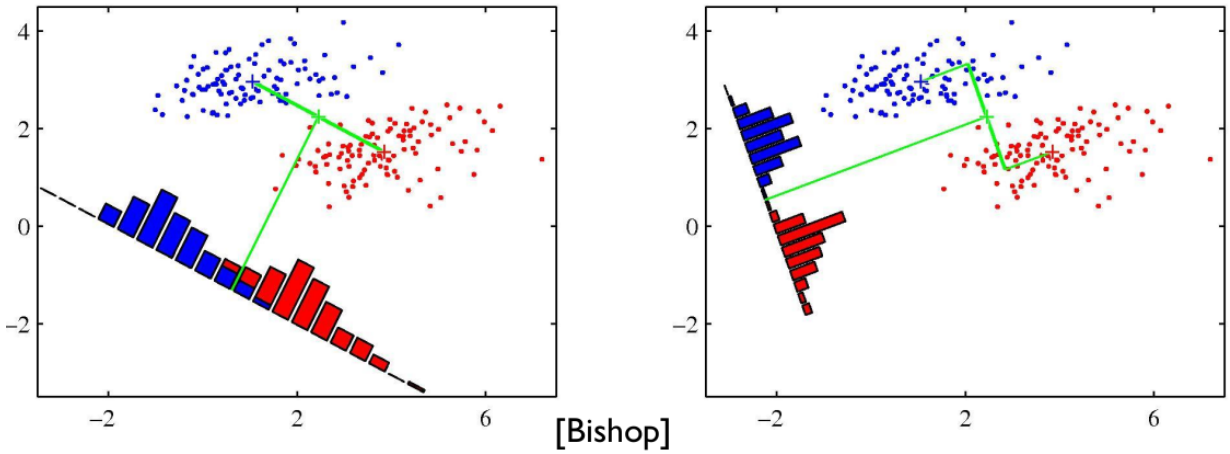
که در فرمول بالا u'1 و u'2 میانگین کلاس داده ها در فضای پروجکت شده هستنند . همان طور که از جبر خطی میدانیم رابطه بین میانگین کلاس داده در فضای اولیه و فضای پروجکت شده به صورت زیر است.

$$\mu_1' = \boldsymbol{w}^T \boldsymbol{\mu}_1 \qquad \boldsymbol{\mu}_1 = \frac{\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \boldsymbol{x}^{(i)}}{N_1}$$

$$\mu_2' = \boldsymbol{w}^T \boldsymbol{\mu}_2 \qquad \boldsymbol{\mu}_2 = \frac{\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_2} \boldsymbol{x}^{(i)}}{N_2}$$

Fisher idea

بیان میکنند علاوه بر حالت قبل که فاصله میانگین کلاس های داده های تصویر شده از هم ماکسیم باشد کاری کنیم که واریانس داده های هر کلاس نیز کم باشد که با این کار همپوشانی داده های کلاس ها نیز مینیم می شود



[Bishop]

فرمول آن بدین صورت است:

$$J(\boldsymbol{w}) = \frac{|\mu_1' - \mu_2'|^2}{s_1'^2 + s_2'^2}$$

که با ساده سازی های جبری به فرمول زیر میرسیم.

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}$$

که در آن

Between-class scatter matrix ⟵ $\boldsymbol{S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}$

Within-class scatter matrix ⟵ $\boldsymbol{S_W = S_1 + S_2}$

$$\boldsymbol{S_1 = \sum_{x^{(i)} \in C_1} \left(x^{(i)} - \mu_1\right)\left(x^{(i)} - \mu_1\right)^T}$$

$$\boldsymbol{S_2 = \sum_{x^{(i)} \in C_2} \left(x^{(i)} - \mu_2\right)\left(x^{(i)} - \mu_2\right)^T}$$

اگر از رابطه بالا مشتق بگیریم و برابر صفر قرار دهیم داریم

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}$$

هدف پیدا کردن راستا w است با حل معادله بالا داریم.

$$\boldsymbol{w \propto S_W^{-1}(\mu_1 - \mu_2)}$$

8 - معیاری است مانند f1 برای دقت مدل در classification . این معیار وقتی به تفسیر معیاری نیاز نداریم و همچنین داده بالانس باشن .. معیار خوبی است.