گزارش پروژه‌ی ۲

زهرا محمدبیگی

۹۷۲۲۲۰۷۹

بخش اول:

قسمت ۱:

حالت(۱):

```
4 kFold_CV2(5, X1, y1)
```

```
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
Mean MSEs on the test sets: 0.0014310303461699393
```

```
1 kFold_CV2(10, X1, y1)
```

```
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
MSE on the test set: 0.0014310303461699393
Mean MSEs on the test sets: 0.0014310303461699395
```

حالت(۲):

```
1 kFold_CV(5, X1, y1)
```

```
MSE on the test set: 0.0014013380212492305
MSE on the test set: 0.0014063516080364974
MSE on the test set: 0.0014926947364295937
MSE on the test set: 0.0014739624957834584
MSE on the test set: 0.001381065420135311
Mean MSEs on the test sets: 0.001431082456326818
```

```
1 kFold_CV(10, X1, y1)
```

```
MSE on the test set: 0.0015179439221928972
MSE on the test set: 0.0012848612653289973
MSE on the test set: 0.0014475766031993487
MSE on the test set: 0.001365071599097838
MSE on the test set: 0.0014582953459220708
MSE on the test set: 0.0015271852790737206
MSE on the test set: 0.0015462075585231205
MSE on the test set: 0.0014016009090330776
MSE on the test set: 0.001457868870535493
MSE on the test set: 0.0013042193044963406
Mean MSEs on the test sets: 0.0014310830657402905
```

حالت(۳):

```
3 kFold_CV(5, X3, y3)
```

```
MSE on the test set: 0.0013378831605116075
MSE on the test set: 0.0013478809356637558
MSE on the test set: 0.001419128130595827
MSE on the test set: 0.0013926241179610087
MSE on the test set: 0.001318716914099375
Mean MSEs on the test sets: 0.0013632466517663148
```

```
1 kFold_CV(10, X3, y3)
```

```
MSE on the test set: 0.0014467565334352265
MSE on the test set: 0.0012292261105447168
MSE on the test set: 0.001398181507420893
MSE on the test set: 0.001297507475358853
MSE on the test set: 0.0013873723748710695
MSE on the test set: 0.0014509809067086376
MSE on the test set: 0.0014514844205513748
MSE on the test set: 0.0013335024612554658
MSE on the test set: 0.0014009704982155943
MSE on the test set: 0.0012363933503696778
Mean MSEs on the test sets: 0.0013632375638731509
```

حالت(۴):

```
3 kFold_CV(5, X4, y4)
```

```
MSE on the test set: 0.0011228050427636 31
MSE on the test set: 0.0010289701468404 62
MSE on the test set: 0.0010612095287429527
MSE on the test set: 0.0010354805111958943
MSE on the test set: 0.0010376068099220142
Mean MSEs on the test sets: 0.0010572144078929908
```

```
1 kFold_CV(10, X4, y4)
```

```
MSE on the test set: 0.00117018495771325
MSE on the test set: 0.0010180738822800157
MSE on the test set: 0.001070483078698131
MSE on the test set: 0.0009864633819172037
MSE on the test set: 0.0010278348098713749
MSE on the test set: 0.0010906622708951978
MSE on the test set: 0.0010865076192318448
MSE on the test set: 0.000979193627694081
MSE on the test set: 0.0011300079429694269
MSE on the test set: 0.0009438397288684028
Mean MSEs on the test sets: 0.0010503251300138928
```

حالت(۵):

```
10 Fold:Mean MSE: 0.001 (0.000)
5 Fold:Mean MSE: 0.001 (0.000)
```

```
Repeated 10 fold:Mean MSE: 0.001 (0.000)
```

```
All errors are MSE
------------------------------------------------------------------------------
alpha:   0.001 | train error: 0.001 | test error: 0.001
alpha:    0.01 | train error: 0.001 | test error: 0.001
alpha:     0.1 | train error: 0.001 | test error: 0.001
alpha:       1 | train error: 0.001 | test error: 0.001
alpha:      10 | train error: 0.001 | test error: 0.001
```

حالت(۶):

```
10 Fold:Mean MSE: 0.015 (0.000)
5 Fold:Mean MSE: 0.015 (0.000)
```

```
1 print(val_errors)
```

```
[0.1056353727523924, 0.1142056205630738, 0.12310714949502266, 0.341178286575496, 0.3876513014543515, 0.3876513014543515]
```

+ Code + Text

```
1 print('best alpha: {}'.format(alphas[np.argmin(val_errors)]))
```

```
best alpha: 0.0001
```

قسمت ۲:

بله در رگرسیون Ridge خطای تست کمتری وجود دارد.احتمالا اکثر متغیرها به خروجی وابسته بوده‌اند و چون Lassoمتغیرها را حذف می کند عملکرد ضعیف تری داشته است.

بخش سوم:

۱.

	precision	recall	f1-score	support
class 0	0.98	0.99	0.99	500
class 1	0.98	0.97	0.97	500
class 2	0.97	0.96	0.97	500
class 3	0.98	0.99	0.99	500
accuracy			0.98	2000
macro avg	0.98	0.98	0.98	2000
weighted avg	0.98	0.98	0.98	2000

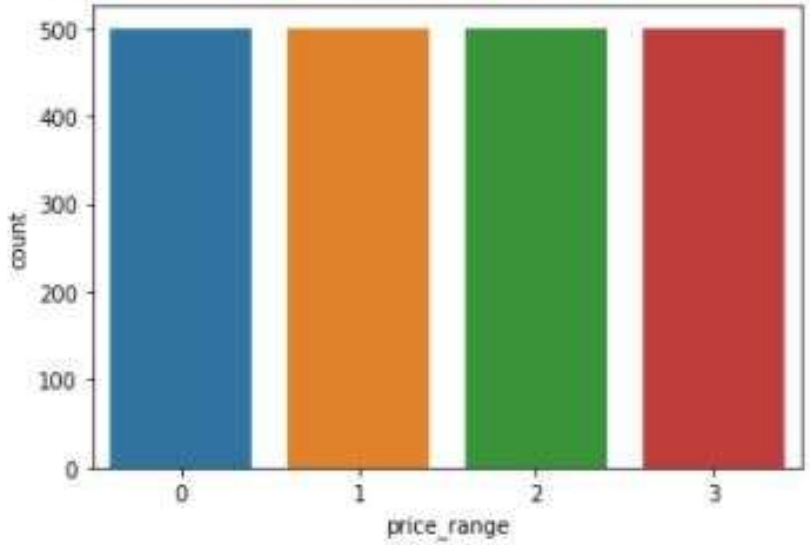
۲.

تعداد نمونه‌های هر کلاس با هم برابر است.

```
3    0.25
2    0.25
1    0.25
0    0.25
Name: price_range, dtype: float64
```

```
1 sns.countplot(x='price_range', data=train_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb2a492f790>
```

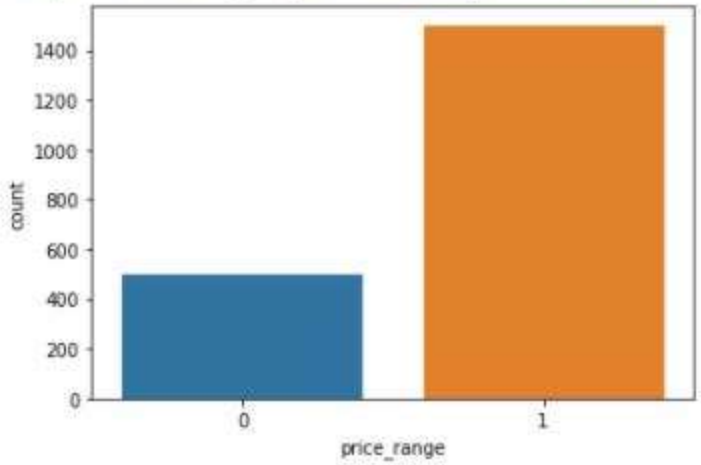


۴. داده ها نامتوازن:

```
1    0.75
0    0.25
Name: price_range, dtype: float64
```

```
1 sns.countplot(x='price_range', data=train_data_new)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb2a3cdd250>
```



	precision	recall	f1-score	support
class 0	0.98	1.00	0.99	500
class 1	1.00	0.99	1.00	1500
accuracy			0.99	2000
macro avg	0.99	0.99	0.99	2000
weighted avg	0.99	0.99	0.99	2000

۵.

کلاس‌بندی بر روی داده‌های نامتوازن باعث عملکرد ضعیف مدل می شود و خطای را در رابطه با داده های اکثریت بخوبی نشان نمی دهد.

۱.روش undersampling

به طور رندوم داده‌های متعلق به کلایس اکثریت را حذف می‌کند تا تعداد داده‌های دو کلاس برابر شود.

۲.روش oversampling

داده‌های متعلق به کلاس اقلیت را افزایش می دهد(با تکرار داده های موجود)تا تعداد داده های هر دو کلاس برابر شود.

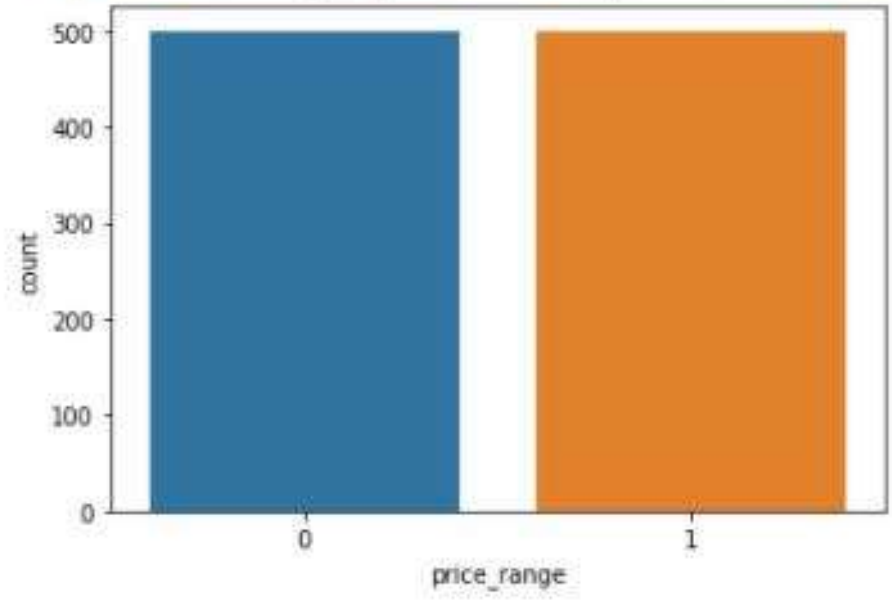
در این قسمت از روش undersampling استفاده شده است.

After under sampling:

```
1    0.5
0    0.5
Name: price_range, dtype: float64
```

```
1 sns.countplot(x='price_range', data=train_data_new)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb2a3c63110>
```



	precision	recall	f1-score	support
class 0	0.97	1.00	0.98	500
class 1	1.00	0.97	0.98	500
accuracy			0.98	1000
macro avg	0.98	0.98	0.98	1000
weighted avg	0.98	0.98	0.98	1000

.۶

در این قسمت و هم چنین قسمت (۱۰) کلاس ها به دو کلاس کاهش پیدا کرده است.کلاس های با برچسب ۰و۱، برچسب ۰ و کلاس های با برچسب ۲و۳، برچسب۱ گرفته اند.

```
1 forwardSelection(X, y)
```

	feature to add	ROC AUC
0	ram	0.975568
1	battery_power	0.987835
2	px_height	0.996785
3	px_width	0.999430
4	mobile_wt	0.999650
5	wifi	0.999720
6	int_memory	0.999725
7	blue	0.999735
8	four_g	0.999740
9	sc_h	0.999720
10	pc	0.999715
11	n_cores	0.999745
12	dual_sim	0.999750
13	fc	0.999735
14	talk_time	0.999725
15	clock_speed	0.999710
16	sc_w	0.999665
17	three_g	0.999635
18	touch_screen	0.999545

.۷

We only use the first 5 features:

```
[ ]  1 X_train4 = train_data3[['ram', 'battery_power', 'px_height','px_width', 'mobile_wt']]
     2 y_train4 = train_data3['price_range']
     3 classification_reports(X_train4, y_train4, model_binary_classes, target_names_2classes)
```

```
                precision    recall  f1-score   support

       class 0       0.99      0.99      0.99      1000
       class 1       0.99      0.99      0.99      1000

      accuracy                           0.99      2000
     macro avg       0.99      0.99      0.99      2000
  weighted avg       0.99      0.99      0.99      2000
```

.۸

```
print(pca.explained_variance_ratio_)
```

```
[0.30219317 0.20079918 0.19987953 0.19948384 0.09764428]
```

```
print(pca.singular_values_)
```

```
[54.95835453 44.79941715 44.69671006 44.65244666 31.24027221]
```

.٩

	precision	recall	f1-score	support
class 0	0.99	0.99	0.99	1000
class 1	0.99	0.99	0.99	1000
accuracy			0.99	2000
macro avg	0.99	0.99	0.99	2000
weighted avg	0.99	0.99	0.99	2000

.١٠

	feature removed	ROC AUC
0	ram	0.000000
1	battery_power	0.975568
2	px_height	0.987835
3	px_width	0.996785
4	mobile_wt	0.999430
5	wifi	0.999650
6	pc	0.999720
7	int_memory	0.999725
8	n_cores	0.999725
9	dual_sim	0.999745
10	four_g	0.999765
11	fc	0.999770
12	talk_time	0.999755
13	sc_h	0.999750
14	blue	0.999735
15	clock_speed	0.999725
16	sc_w	0.999710
17	three_g	0.999665
18	touch_screen	0.999635
19	m_dep	0.999545

we remove all features except "ram" and "battery_power"

```
X_train5 = train_data3[['ram', 'battery_power']]
y_train5 = train_data3['price_range']
classification_reports(X_train5, y_train5, model_binary_classes, target_names_2classes)
```

```
              precision    recall  f1-score   support

     class 0       0.93      0.93      0.93      1000
     class 1       0.93      0.94      0.93      1000

    accuracy                           0.93      2000
   macro avg       0.93      0.93      0.93      2000
weighted avg       0.93      0.93      0.93      2000
```

```
[ ]  1 cv = KFold(n_splits=5, random_state=1, shuffle=True)
     2 scores = cross_val_score(model, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1)
     3 print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy: 0.961 (0.006)
```

```
[ ]  1 cv = KFold(n_splits=10, random_state=1, shuffle=True)
     2 scores = cross_val_score(model, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1)
     3 print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy: 0.963 (0.011)
```