دارای مجموعه داده‌ای هستیم که بصورت متعادل نمونه‌ها در آن توزیع نشده اند. به همین جهت با استفاده از رویکرد خاصی باید میان داده‌ها تعادل ایجاد کنیم. بدین جهت سه سیاست نمونه‌افزایی(over sampling)، نمونه‌کاهی(under sampling) و weighted lose را بر روی مجموعه داده اعمال می‌کنیم.

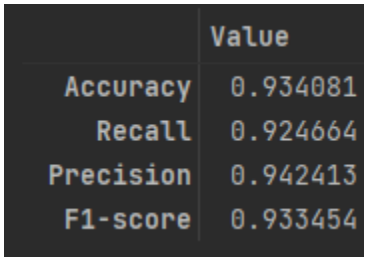
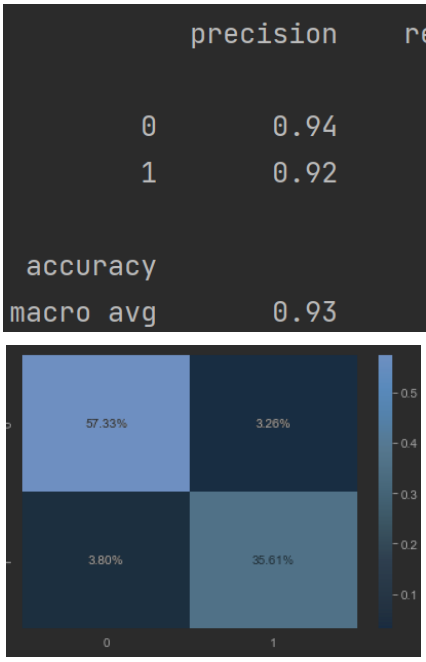
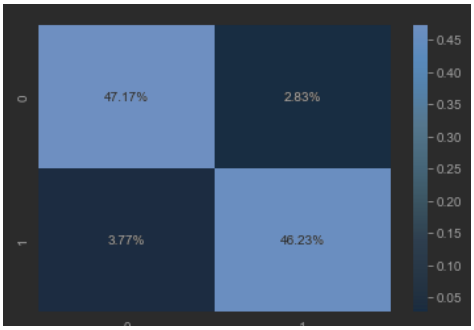
- نمونه افزایی

```
In 53 1 #Oversampling
       2 from imblearn.over_sampling import RandomOverSampler
       3
       4 ros = RandomOverSampler(random_state=42, sampling_strategy='minority')
       5 x_resample_over, y_resample_over = ros.fit_resample(x_train, y_train)
       6 x_resample_over
```

با استفاده از کتابخانه imblearn و با استفاده از کلاس RandomeOverSampling این سیاست را اجرا می‌کنیم. در این سیاست کلاسی که در اقلیت قرار دارد، باید در مجموعه داده زیادتر شود. برای اینکار از نمونه هایی که به دسته اقلیت تعلق دارند، بطور تصادفی نمونه‌هایی انتخاب میشوند و مجددا در مجموعه داده کپی میشوند. بدین شکل نمونه افزایی انجام می‌شود. ایراد این روش آن است که در حین نمونه‌افزایی ممکن است نمونه‌ای را کپی کند که outlier است و موجب بایاس شود.

با انجام نمونه افزایی، مجموعه داده ما بزرگتر خواهد شد. اندازه مجموعه داده از ۳۶۸۰ نمونه به ۴۴۶۰ نمونه افزایش پیدا کرده است.

نتایج ارزیابی با استفاده از Logistic Regression و نمونه‌افزایی برای مجموعه train و test به شرح زیر است:

train	test
	
	

	Value
Accuracy	0.934081
Recall	0.924664
Precision	0.942413
F1-score	0.933454

	precision	recall	f1-score	support
0	0.94	0.95	0.94	558
1	0.92	0.90	0.91	363
accuracy			0.93	921
macro avg	0.93	0.92	0.93	921

- نمونه‌کاهی

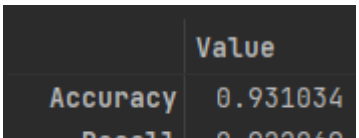
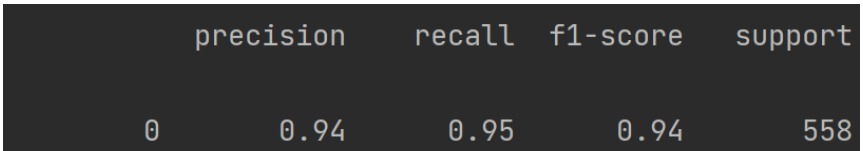
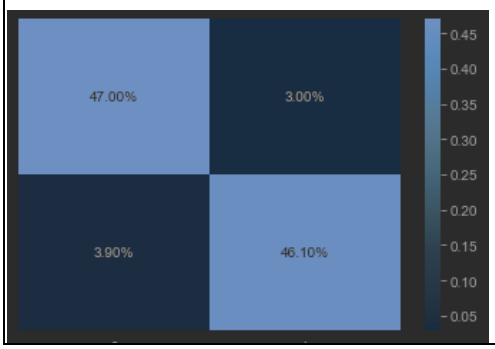
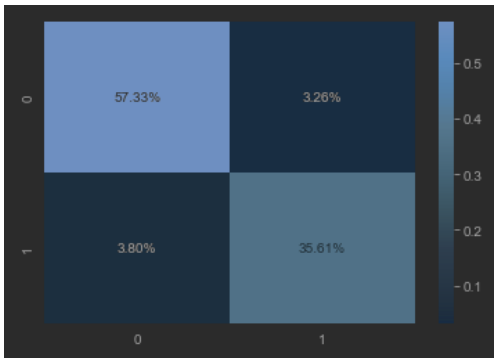
```python
#Undersampling
from imblearn.under_sampling import RandomUnderSampler

ros = RandomUnderSampler(random_state=42, sampling_strategy='majority')
x_resample_under, y_resample_under = ros.fit_resample(x_train, y_train)
x_resample_under
```

در این تکنیک با استفاده از کتابخانه imblearn و کلاس RandomeUnderSampler نمونه‌کاهی را انجام می‌دهیم. در این روش، از کلاسی که دارای نمونه‌هایی است که در اکثریت در اختیار آن است، باید نمونه هایی را بطور تصادفی پاک کنیم. این پاک کردن تا زمانی ادامه پیدا می‌کند که تعداد نمونه‌های کلاس اکثریت به تعداد نمونه‌های کلاس اقلیت برسد.
در این روش نیز اندازه مجموعه داده ما کوچک‌تر خواهد شد. از ۳۶۸۰ نمونه به ۲۹۰۰ نمونه، مجموعه داده ما کاهش نمونه داشته است.

ایراد این روش در آنجاست که نمونه‌ها را بصورت تصادفی پاک می‌کند و ممکن است نمونه‌های تاثیرگذاری را حذف کند.

نتایج ارزیابی Logistic Regression به همراه Under sampling به شرح زیر است:

Train	test
	
	

	Value
Accuracy	0.931034
Recall	0.922069
Precision	0.938904
F1-score	0.930411

	precision	recall	f1-score	support
0	0.94	0.95	0.94	558
1	0.92	0.90	0.91	363
accuracy			0.93	921
macro avg	0.93	0.92	0.93	921

- Weighted Loss

```python
# Logistic Regression with weighted Loss
from sklearn.linear_model import LogisticRegression
lr_reg_wh = LogisticRegression(solver='newton-cg', class_weight='balanced')
lr_reg_wh.fit(x_train, y_train)
```

```
LogisticRegression(class_weight='balanced', solver='newton-cg')
```

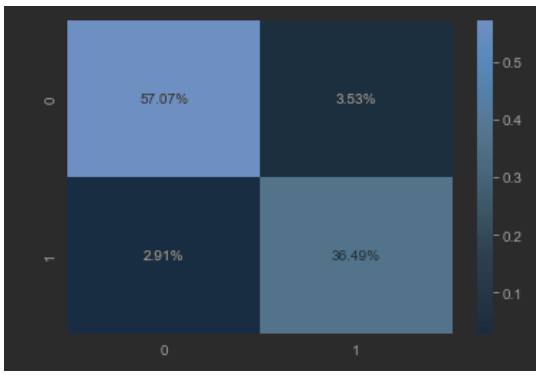
در این رویکرد، مدل Logistic Regression به وزن بیشتری را به خطاهای ایجاد شده در کلاس اقلیت می‌دهد. در نتیجه این امر، مدل توجه بیشتری را در یادگیری از روی داده‌های اقلیت اختصاص می‌دهد. در نتیجه این امر باعث می‌شود که تمایز بهتری بین کلاس‌های اقلیت و اکثریت قائل شود و مرز تصمیم آنها را بدین وسیله دقیق‌تر شناسایی کند.

در این روش برخلاف دو روش قبلی نیاز به افزودن یا کاستن از نمونه های مجموعه داده نیست.

نتایج ارزیابی Weighted Loss به شرح زیر است:

train

	Value
Accuracy	0.935598
Recall	0.926207
Precision	0.911745
F1-score	0.918919



test

	precision	recall	f1-score	support
0	0.94	0.94	0.94	558
1	0.91	0.90	0.91	363
accuracy			0.93	921
macro avg	0.92	0.92	0.92	921