۱.۱-

- این دیتاست شامل اطلاعات مشتریانیست که خدمات کارت بانکی بهره می برند یا بهره برده اند. طبق نوشته داخل سایت Kaggle این داده ها توسط رئیس بانک قرار داده شده و همچنین قصد دارند که به کمک این داده ها خدماتشان را طوری در خلاف جهت این آمار تغییر دهند تا مشتریان بیش تر از اینکه خدمات بانک را حذف کنند به آن جذب شوند.
- این دیتاست بدون لحاظ کردن شماره هر مشتری و دو ویژگی آخر(طبق گفته سایت این ویژگی ها بی کاربردند) ۲۱ ویژگی را داراست که این ویژگی ها به شرح زیر می باشند:

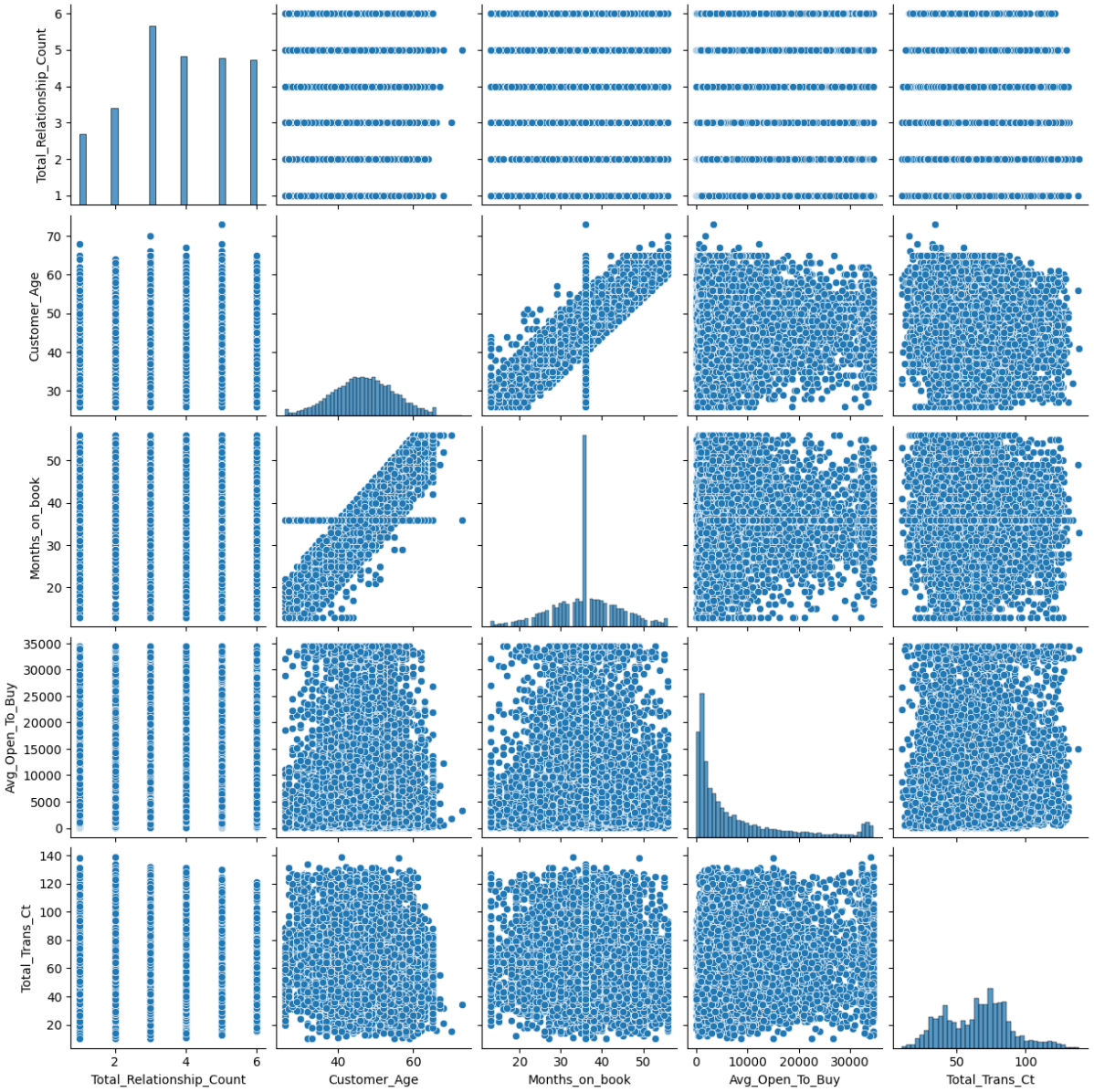
1	Attrition_Flag	۱- وضعیت خروج از خدمات
2	Customer_Age	۲- سن مشتری
3	Gender	۳- جنسیت
4	Dependent_count	۴- تعداد افراد تحت تکفل
5	Education_Level	۵- سطح تحصیلات
6	Marital_Status	۶- وضعیت تأهل
7	Income_Category	۷- دسته‌بندی درآمد
8	Card_Category	۸- دسته‌بندی کارت
9	Months_on_book	۹- تعداد ماه‌ها از عضویت
10	Total_Relationship_Count	۱۰- تعداد کل تعاملات
11	Months_Inactive_12_mon	۱۱- تعداد ماه‌های غیرفعال در ۱۲ ماه گذشته
12	Contacts_Count_12_mon	۱۲- تعداد تماس‌ها در ۱۲ ماه گذشته
13	Credit_Limit	۱۳- سقف اعتبار
14	Total_Revolving_Bal	۱۴- کل مانده اعتباری چرخشی
15	Avg_Open_To_Buy	۱۵- میانگین اعتبار قابل استفاده
16	Total_Amt_Chng_Q4_Q1	۱۶- تغییر کل مبلغ از فصل چهارم به فصل اول
17	Total_Trans_Amt	۱۷- کل مبلغ تراکنش‌ها
18	Total_Trans_Ct	۱۸- تعداد کل تراکنش‌ها
19	Total_Ct_Chng_Q4_Q1	۱۹- تغییر تعداد کل تراکنش‌ها از فصل چهارم به فصل اول
20	Avg_Utilization_Ratio	۲۰- نسبت میانگین استفاده

- طبق گفته سایت این دیتاست شامل 10,000 مشتری است که عدد دقیق داده در ذیل مشخص شده.

```
10127 entries
```

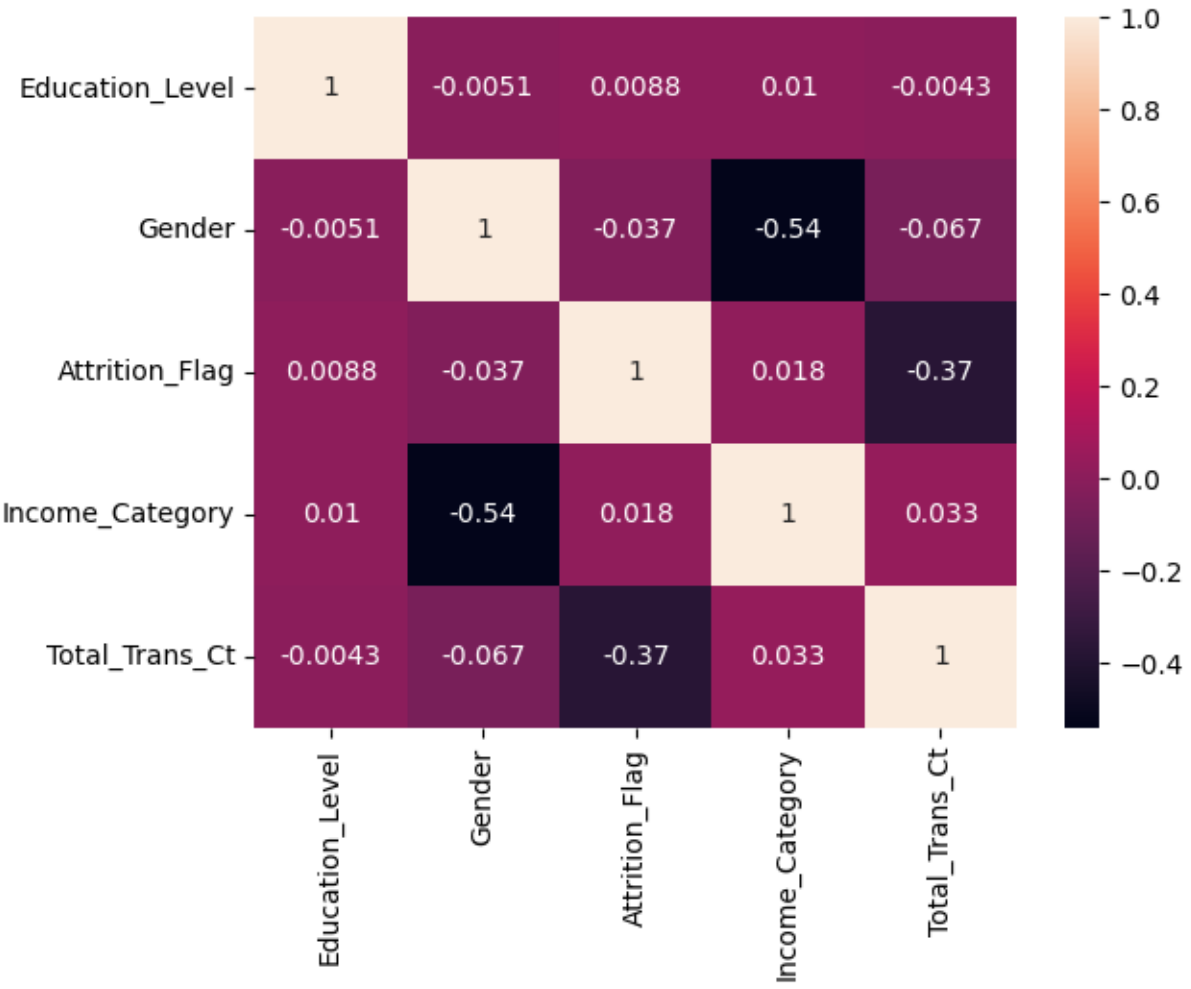
۱.۲-

به کمک کتاب خانه seaborn پخش داده ها را نمایش می دهیم. (ویژگی ها به دلخواه انتخاب شده اند)



۱.۳-

به کمک کتابخانه Matplotlib نمودار حرارتی را با دو ویژگی پیوسته و سه ویژگی گسسته دلخواه که خودمان از قبل آن را عددی کردیم، نمایش می دهیم.



۱.۴-

خیر هیچ داده Nan نمی باشد.

```
[21] data.isna().values.any()
```

```
False
```

۱.۵-

- دارای دو کلاس است :

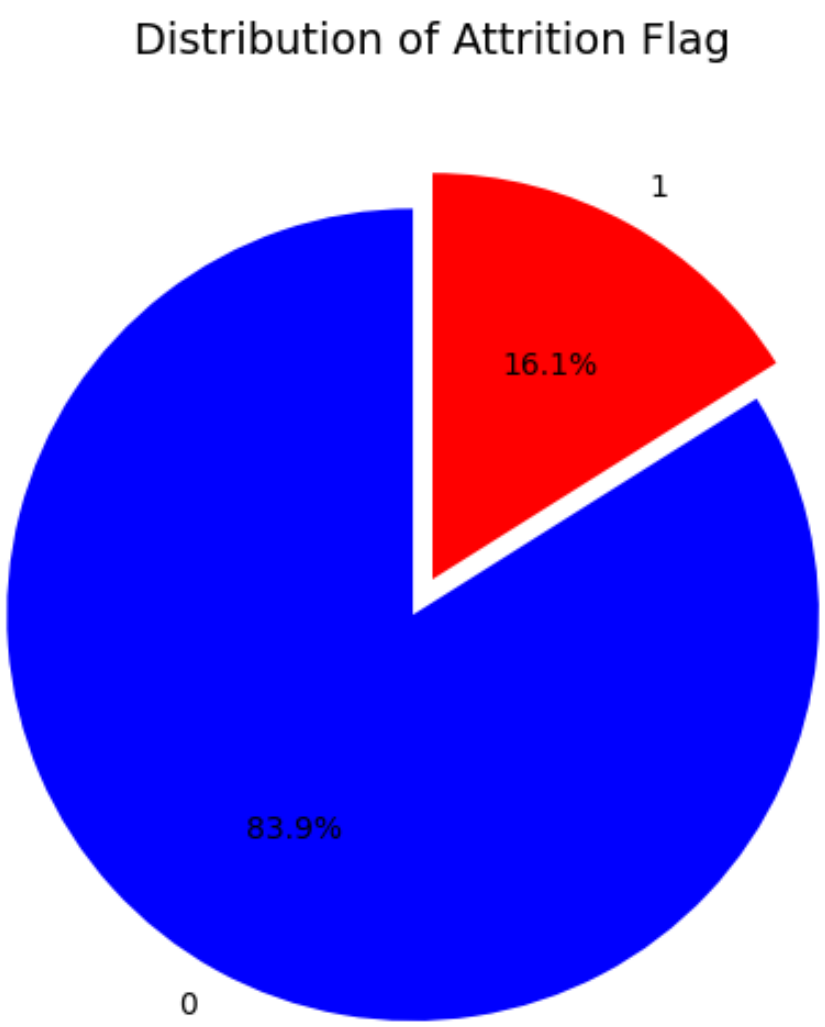
Existed costumer (0):

کاربرانی که همچنان از خدمات کارت بانک اعتباری خود استفاده می کنند

Attrited costumer (1):

کاربرانی که مدت زیادیست که از خدمات کارت بانک اعتباری خود استفاده نمی کنند

Pie Plot ✓



- عدم توازن در مجموعه داده‌ها به وضعیتی اشاره دارد که توزیع داده‌ها بین کلاس‌ها یا دسته‌های مختلف به‌طور قابل‌توجهی نابرابر باشد. به طور مثال در همین ویژگی طیف وسیعی از داده ها جزو مشتریانی هستند که همچنان از خدمات کارت بانکی اعتباری استفاده می کنند و این می تواند منجر به مشکلات زیر در مدل نهایی ما شود:

✓ **سوگیری مدل** :مدل یادگیری ماشین ممکن است به دلیل مشاهده بیشتر نمونه‌های کلاس غالب، به سمت آن گرایش پیدا کند.

✓ **عملکرد ضعیف روی کلاس اقلیت** :مدل ممکن است در پیش‌بینی صحیح کلاس اقلیت دچار مشکل شود و دقت یا یادآوری (Recall) برای این کلاس کاهش یابد.

✓ **معیارهای ارزیابی گمراه‌کننده** :معیارهایی مانند دقت (Accuracy) ممکن است گمراه‌کننده باشند. به‌عنوان مثال، یک مدل که فقط کلاس غالب را پیش‌بینی می‌کند می‌تواند دقت بالایی داشته باشد بدون اینکه مفید باشد .

- برای مقابله با این چالش، می‌توان از روش‌های زیر استفاده کرد:

۱. **بازنمونه‌گیری داده‌ها**

۲. **Oversampling** : این روش شامل تولید داده های مصنوعی می شود. مانند استفاده از تکنیک (SMOTE)

۳. **Undersampling** : در این روش تعداد داده های کلاس غالب را کم میکنیم تا مدل متوازن شود و سوگیری نکند.

۴. **استفاده از توابع زیان وزنی(Weighted Loss Functions)** : این روش شامل وزن دادن به کلاس های اقلیت می شود تا تاثیر آن را در مدل به منظور موازنه داده ها افزایش دهد.

۵. **به‌کارگیری معیارهای ارزیابی مناسب** :به جای تکیه بر دقت(Accuracy) ، معیارهایی همچون Precision ،F1 Score، و Recall کاربرد بیشتری دارند.

همچنین ضروری است داده‌ها به سه بخش تقسیم شوند:

۱. **داده‌های آموزشی :(Training Data)** ۷۵ تا ۸۰ درصد داده ها

۲. **داده‌های آزمون :(Testing Data)** ۱۵تا ۲۰ درصد داده‌ها

۳. **داده‌های اعتبارسنجی :(Validation Data)** ۵ درصد داده‌ها

اگر این تقسیم‌بندی به درستی انجام نشود، ممکن است مدل روی داده‌های دیده‌شده عملکرد بیش‌ازحد خوش‌بینانه‌ای نشان دهد و در عمل، در مواجهه با داده‌های واقعی ضعیف عمل کند. بنابراین، یک تقسیم‌بندی مناسب به ارزیابی مستقل عملکرد مدل روی داده‌های آزمون کمک کرده و توانایی واقعی مدل را نمایان می‌سازد.

- الگوریتم های اشاره شده در قسمت قبلی باید **فقط روی داده‌های آموزشی** انجام شود و نه کل مجموعه داده. این کار باید **پس از تقسیم‌بندی داده‌ها** به بخش‌های آموزشی، آزمون و اعتبارسنجی انجام شود. دلیل این رویکرد به منظور جلوگیری از نشت داده (Data leakage)، حفظ ارزیابی واقعی مدل و در آخر تمرکز بر آموزش مدل می باشد.

۱.۶- ۱)

طبق گفته صورت مسئله ستون اول یعنی Attrited Customers را به عنوان خروجی مدل یعنی Y در نظر می گیریم و بقیه داده ها را به عنوان ورودی یعنی X مدل لحاظ می کنیم.

```
Y = data.iloc[:, 1]
X = data.drop(columns=data.columns[1])
indices = Y.index
```

سپس داده ها را به نسبت دلخواه به سه دسته آموزش، آزمون، اعتبارسنجی تقسیم بندی می کنیم.

در این مدل ۸۵٪ داده ها را تحت آموزش و بقیه را داده های آزمون لحاظ می کنیم. از داده های آزمون نیز ۲۰٪ آن را برای داده های اعتبار سنجی کنار می گذاریم. همچنین عدد Random state طبق توضیح در ابتدای پروژه برابر با دو رقم آخر شماره دانشجویی اینجانب ۷۳ قرار می دهیم.

```
train_indices, test_indices = train_test_split(indices, test_size=0.15,
stratify=Y, random_state=73)
X_train, Y_train = X.loc[train_indices], Y.loc[train_indices]
X_test, Y_test = X.loc[test_indices], Y.loc[test_indices]

test_indices, val_indices = train_test_split(test_indices,
test_size=0.2, random_state=73)
X_test, Y_test = X.loc[test_indices], Y.loc[test_indices]
X_val, Y_val = X.loc[val_indices], Y.loc[val_indices]
```

تقسیم بندی کمی داده ها به شرح زیر شده است:

داده های آموزشی: ۸۶۰۷ عدد

داده های آزمون: ۱۲۱۶ عدد

داده های اعتبارسنجی: ۳۰۴ عدد

در مجموع همان ۱۰،۱۲۷ داده را در اختیار داریم.

سپس پس از استاندارد سازی و اسکیل کردن داده ها به آموزش مدل می پردازیم:

در اینجا از مدل آماده کتابخانه Scikit Learn با نام Random Forest استفاده می کنیم زیرا که این مدل به طور ذاتی نسبت به داده هایی که توازن ندارند حساسیت کمتری نشان می دهد.

```
rf_model = RandomForestClassifier(random_state=73,n_estimators=100)
rf_model.fit(X_train_scaled, Y_train)
```

سپس بعد از آموزش مدل، آنرا به کمک داده های تست آزمون می کنیم و در آخر به اعتبارسنجی مدل می پردازیم:

```
Y_train_pred = rf_model.predict(X_train_scaled)
Y_test_pred = rf_model.predict(X_test_scaled)
Y_val_pred = rf_model.predict(X_val_scaled)
```

سپس گزارش ها و ماتریس سردرگمی (Confusion Matrix) نحوه عملکرد روند مدل و نتایج آزمون و اعتبار سنجی را گزارش می کنیم. (به کمک تابع Report and Plot)

```
def report_and_plot(y_true, y_pred, title):
    print(f"\nClassification Report ({title}):\n")
    print(classification_report(y_true, y_pred))

    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=['Predicted Negative', 'Predicted
Positive'],
                yticklabels=['Actual Negative', 'Actual Positive'])
    plt.title(f'Confusion Matrix Without Balancing ({title})')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()
```

و تابع را در هر سه مرحله صدا میزنیم.

```
report_and_plot(Y_train, Y_train_pred, "Training")
report_and_plot(Y_test, Y_test_pred, "Testing")
report_and_plot(Y_val, Y_val_pred, "Validation")
```

در آخر هم دقت کلی مدل ارائه شده را به شرح زیر گزارش می دهیم:

```
train_acc = accuracy_score(Y_train, Y_train_pred)
test_acc = accuracy_score(Y_test, Y_test_pred)
val_acc = accuracy_score(Y_val, Y_val_pred)

print(f"\nAccuracy Scores:")
```

```
print(f"Training Accuracy: {train_acc:.2f}")
print(f"Testing Accuracy: {test_acc:.2f}")
print(f"Validation Accuracy: {val_acc:.2f}")
```

نتایج به شرح زیر می شود:

گزارش طبقه بندی:

مرحله آموزش:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7224
1	1.00	1.00	1.00	1383
accuracy			1.00	8607
macro avg	1.00	1.00	1.00	8607
weighted avg	1.00	1.00	1.00	8607

مرحله آزمون:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	1030
1	0.93	0.86	0.89	186
accuracy			0.97	1216
macro avg	0.95	0.92	0.94	1216
weighted avg	0.97	0.97	0.97	1216

مرحله اعتبارسنجی:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	246
1	0.92	0.84	0.88	58
accuracy			0.96	304
macro avg	0.94	0.91	0.93	304
weighted avg	0.96	0.96	0.96	304

گزارش عملکردی کلی:

```
Accuracy Scores:
Training Accuracy: 1.00
Testing Accuracy: 0.97
Validation Accuracy: 0.96
```

این گزارش ها به کمک ۳ معیار معروف Precision، Recall و F1 به جهت عملکرد مدل گزارش می شود.

که به روابط آن به شرح زیر است:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

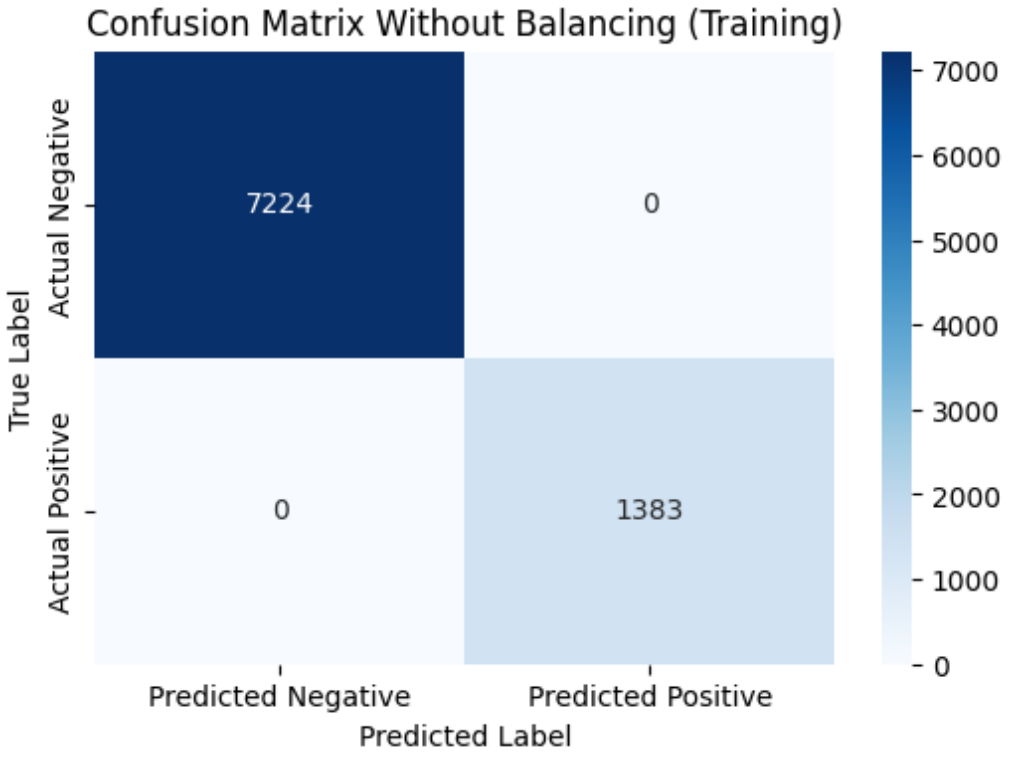
$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

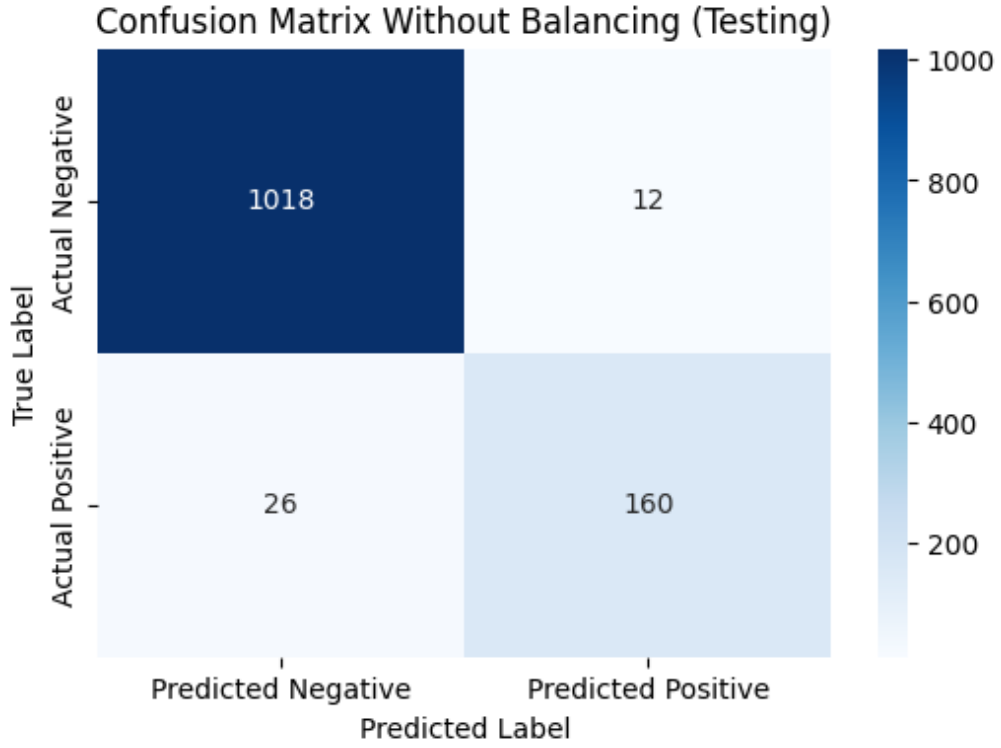
همانطور که مشاهده می شود مدل با دقت نسبتا بالایی هم تعلیم دیده و هم از پس داده های آزمون و اعتبار سنجی برآمده.

حال به تحلیل جدول های سردرگمی می پردازیم:

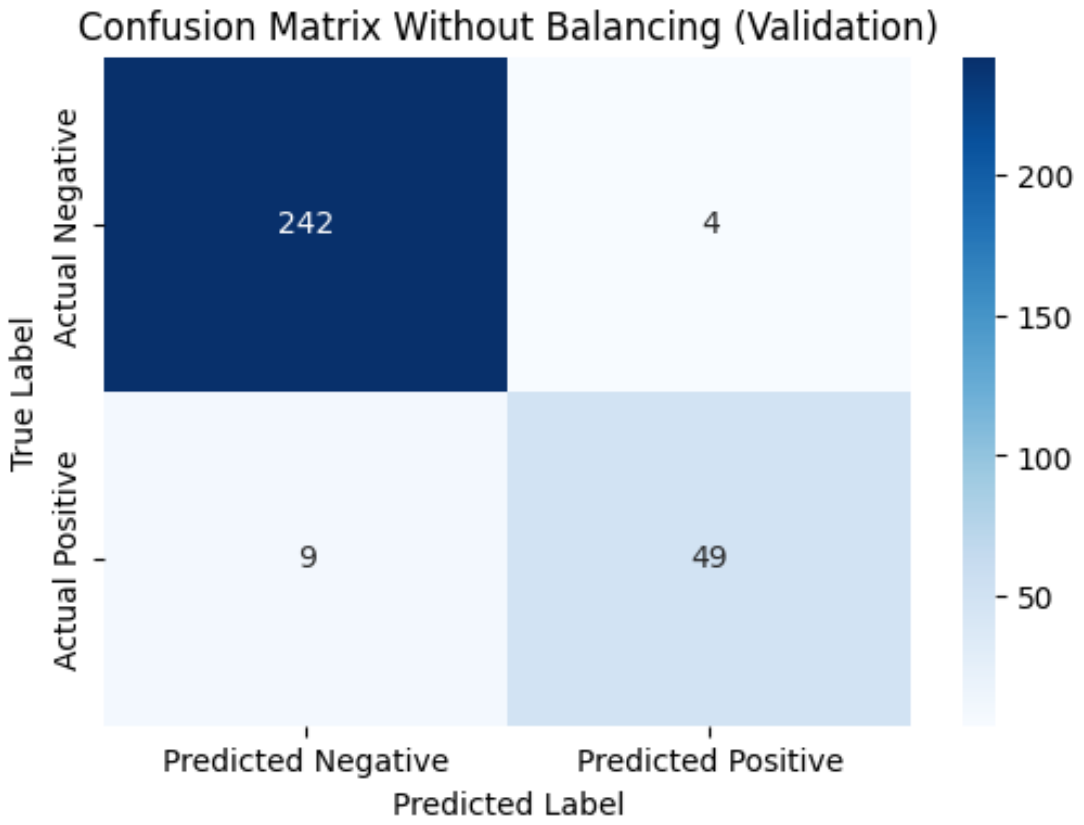
مرحله آموزش:



مرحله آزمون:



مرحله اعتبارسنجی:



همانطور که مشاهده می شود در جدول اول تراکم داده ها در کلاس اول بسیار بیشتر از تراکم داده ها در کلاس دوم است و این منجر به عدم توازن در داده ها می شود.

نتایج داده های تست و اعتبارسنجی منطقی و مناسب به نظر می رسند با توجه به تعداد داده هایی که اشتباه برچسب خوردند (قطر فرعی) نسبت به داده هایی که به طور صحیحی برچسب گذاری شده اند (قطر اصلی) بسیار کمتر است.

۲)

حال در این قسمت سعی می کنیم با تولید داده های مصنوعی از کلاس اقلیت در دیتاست توازن ایجاد کنیم برای اینکار از الگوریتم SMOTE استفاده می کنیم.

کد نوشته شده مشابه کد در بخش قبلیست با این تفاوت که:

داده ها را بر می زنیم به کمک Shuffle:

```
X_train, X_temp, Y_train, Y_temp = train_test_split(X, Y,
test_size=0.15, stratify=Y, shuffle=True, random_state=73)
X_test, X_val, Y_test, Y_val = train_test_split(X_temp, Y_temp,
test_size=0.2, stratify=Y_temp, shuffle=True, random_state=73)
```

برای تطبیق کمی داده ها از SMOTE به صورت زیر برای کلاس اقلیت استفاده می کنیم:

```
smote = SMOTE(random_state=73)
X_train_balanced, Y_train_balanced = smote.fit_resample(X_train,
Y_train)
```

سپس مدل خود را مجددا مشابه بخش قبلی تعریف کرده، آموزش داده، آزمون و در نهایت اعتبارسنجی می کنیم و نتایج را تحلیل می نماییم.

مرحله آموزش:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7224
1	1.00	1.00	1.00	7224
accuracy			1.00	14448
macro avg	1.00	1.00	1.00	14448
weighted avg	1.00	1.00	1.00	14448

مرحله آزمون:

	precision	recall	f1-score	support
0	0.98	0.96	0.97	1021
1	0.83	0.91	0.87	195
accuracy			0.95	1216
macro avg	0.90	0.94	0.92	1216
weighted avg	0.96	0.95	0.96	1216

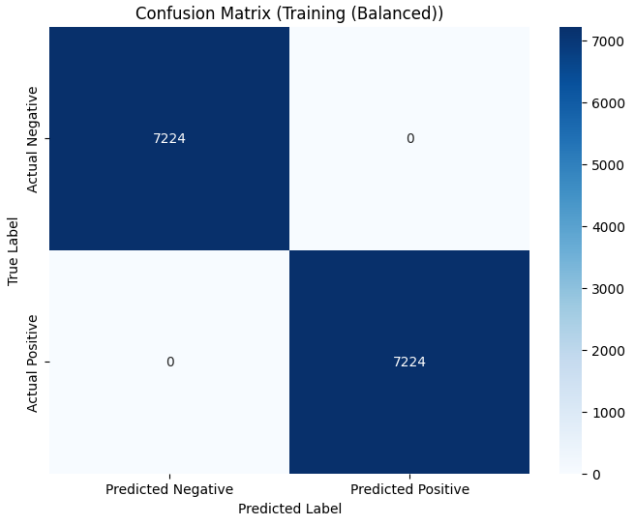
مرحله اعتبارسنجی:

	precision	recall	f1-score	support
0	0.99	0.96	0.97	255
1	0.82	0.94	0.88	49
accuracy			0.96	304
macro avg	0.90	0.95	0.93	304
weighted avg	0.96	0.96	0.96	304

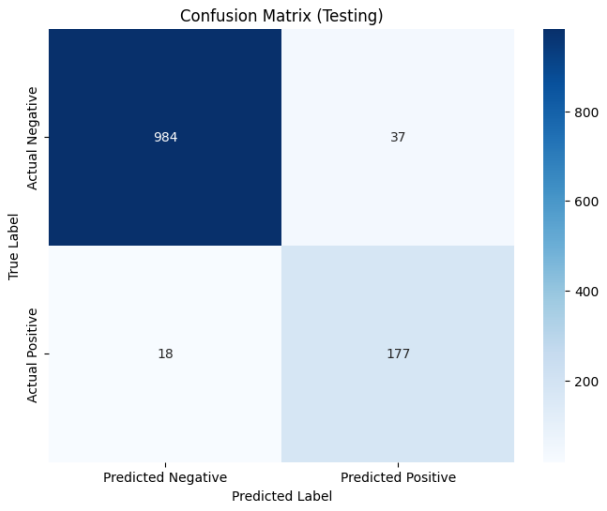
حال نتایج جدول های سردرگمی را گزارش می کنیم:

متعادل شدن داده ها در جدول های زیر کاملا مشهود است.

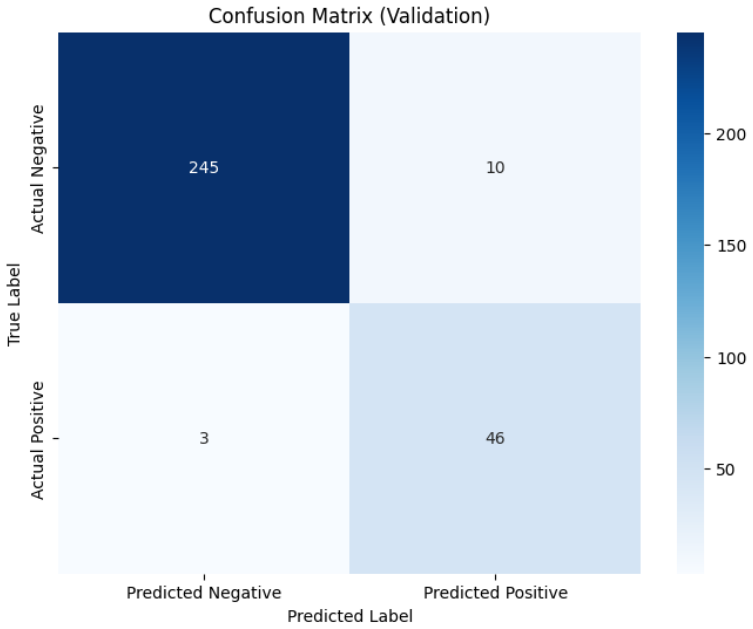
مرحله آموزش:



مرحله آزمون:



مرحله اعتبار سنجی:



دقت نهایی پس از بالانس کردن داده ها:

```
Accuracy Scores:
Training Accuracy (Balanced): 1.00
Testing Accuracy: 0.95
Validation Accuracy: 0.96
```

تحلیل نهایی:

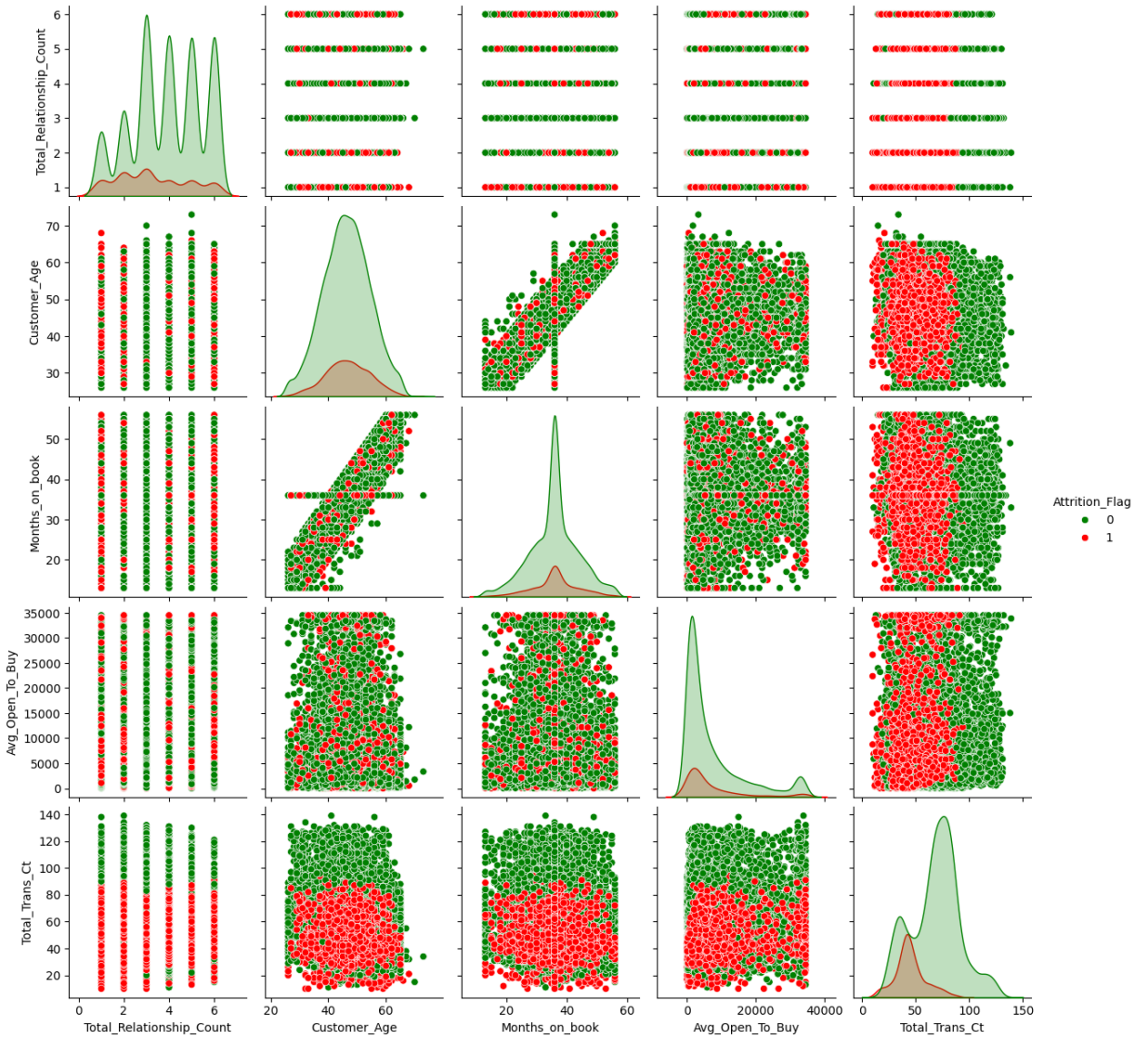
همانطور که مشاهده می شود مجددا با بالانس کردن داده ها نیز مدل خوبی ارائه شده با اندکی کاهش دقت که بسیار قابل چشم پوشی است اما این اتفاق می تواند به دلایل مختلفی صورت بپذیرد:

در مدل بدون تعادل، دقت ممکن است به شدت تحت تأثیر کلاس غالب باشد، زیرا مدل می‌تواند به سادگی اکثریت نمونه‌ها را پیش‌بینی کرده و دقت بالایی کسب کند اما در مدل متوازن، مدل تلاش می‌کند عملکرد خود را برای تمام کلاس‌ها بهبود دهد، حتی اگر این به قیمت کاهش دقت برای کلاس غالب تمام شود.

لازم به ذکر می باشد که متعادل کردن داده ها بر حسب پیچیدگی داده ها لزوما حرکتی سودمند نخواهد بود و ممکن است داده های مصنوعی تولید شده در مدل جدید تاثیر نه چندان مثبتی روی داده ها بگذارند. اما پیشبینی می شود که مدل متعادل نسبت به مدل نا متعادل به مراتب عمومی تر و قابل تکیه تر است.

Abolfazl Valizadeh Lakeh [40010273]
Mini Project 1 Report

امتیازی سوال ۱:



در این نمودار طبق گفته صورت مسئله مقادیر ۰ و ۱ کلاس Attrited customers از هم با دو رنگ تفکیک شده.

۲.۱-

روند افزودن داده به محیط گوگل کولب:

```
!pip install gdown

import numpy as np
file_url =
"https://drive.google.com/uc?id=180FkupJQe0Oiq0A1yJg4m5Ggbfw-G8Sc"
!gdown {file_url} -O data.npy
```

در داده های مذکور ۷۵٪ آنها را به عنوان داده های آموزشی و ۲۵٪ باقی مانده را به عنوان داده های تست در نظر می گیریم و محدوده ۷- تا ۱۳ (محدوده x) آنها را با دو رنگ مختلف سبز و قرمز روی یک نمودار نمایش می دهیم.

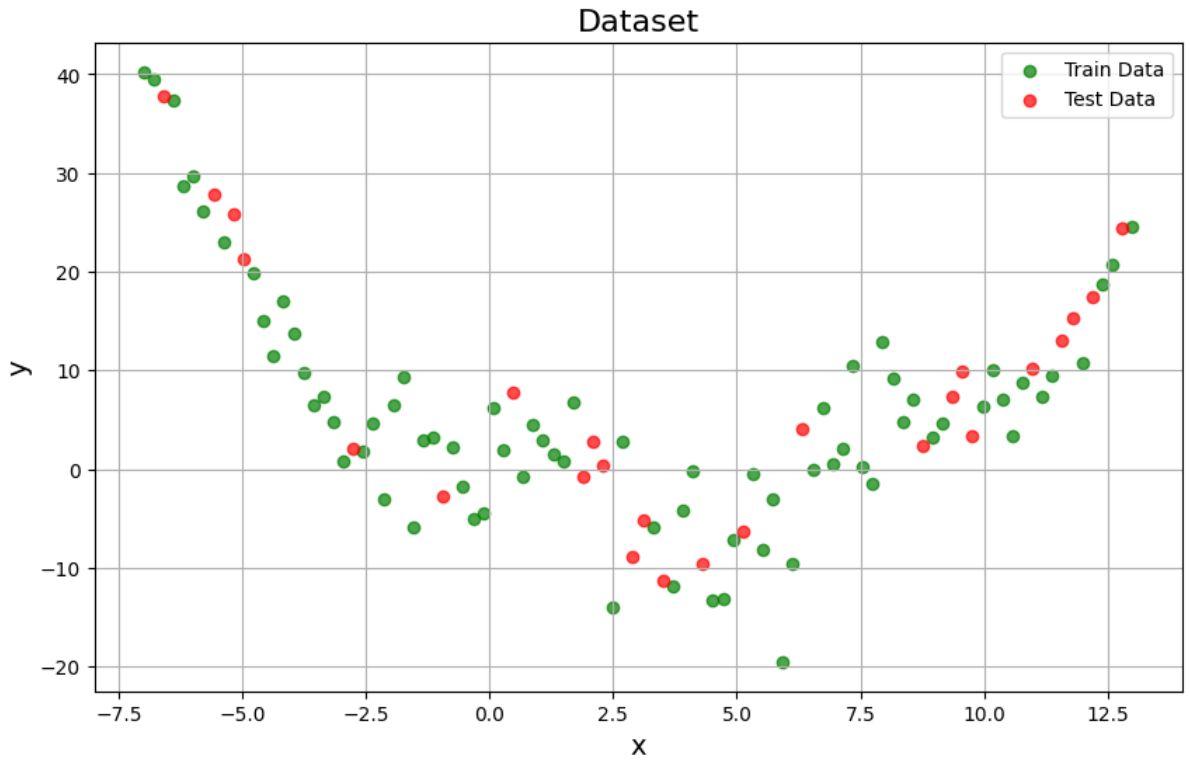
```
y_data = np.load("data.npy")
x_data = np.linspace(-7, 13, len(y_data))
data = pd.DataFrame({'x': x_data, 'y': y_data})

test_size = int(0.25 * len(data))
test_indices = np.random.choice(data.index, size=test_size,
replace=False)
test_data = data.loc[test_indices]
train_data = data.drop(test_indices)

plt.figure(figsize=(10, 6))
plt.scatter(train_data['x'], train_data['y'], color='green',
label='Train Data', alpha=0.7)
plt.scatter(test_data['x'], test_data['y'], color='red', label='Test
Data', alpha=0.7)

plt.title("Dataset", fontsize=16)
plt.xlabel("x", fontsize=14)
plt.ylabel("y", fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```

نمودار به شکل زیر در خواهد آمد:



همانطور که مشاهده می شود از هر محدوده ای داده برای سنجش عملکرد مدل ما وجود دارد.

۲.۲-

خطای میانگین مربعات (MSE)

MSE یک معیار رایج برای ارزیابی دقت مدل‌های رگرسیون است. این معیار میانگین مربع تفاوت مقادیر واقعی و مقادیر پیش‌بینی‌شده را محاسبه می‌کند. این معیار با میانگین گیری روی مربع تفاضل مقدار واقعی متغیر و مقدار پیشبینی شده کار می کند. به دلیل وجود توان ۲، مقادیر خطای بزرگ (پرت) تأثیر بیشتری روی مقدار MSE دارند.

$$\text{MSE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{Mean}} \underbrace{(Y_i - \hat{Y}_i)}_{\text{Error}}^{\underbrace{2}_{\text{Squared}}}$$

خطای نسبی میانگین (MRE):

MRE خطای نسبی را بر اساس مقادیر واقعی محاسبه می‌کند و به صورت درصد بیان می‌شود. این معیار به مقیاس متغیر هدف وابسته نیست. این معیار بی‌واحد است و می‌تواند بین مجموعه داده‌هایی با مقیاس‌های مختلف مقایسه شود. البته برای مقادیر کوچک ممکن است مقدار MRE به شدت بزرگ شود، حتی اگر خطای مطلق کوچک باشد.

$$MeanRelativeError = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|y_{true_i} - y_{pred_i}\right|}{normalizer}$$

ریشه خطای میانگین مربعات(RMSE):

RMSE به نوعی نسخه ریشه‌گیری‌شده MSE است و برای محاسبه میانگین انحراف بین مقادیر واقعی و پیش‌بینی‌شده استفاده می‌شود. RMSE معمولاً به دلیل تفسیر ساده‌تر ترجیح داده می‌شود. مانند MSE، به شدت تحت تأثیر مقادیر پرت است.

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

۲.۳-

در این بخش ما در تلاش هستیم تا مدلی خطی برای رابطه بین x و y ارائه دهیم که در رابطه y = wx + b بگنجد.

هدف ما یافتن w و b هاییست که خطای کل را به حداقل برسانیم.

پس از تقسیم بندی داده ها آن ها را به صورت ستونی در میاوریم:

```
X_train = x_train.reshape(-1, 1)
X_test = x_test.reshape(-1, 1)
Y_train = y_train.reshape(-1, 1)
Y_test = y_test.reshape(-1, 1)
```

میانگین داده ها را محاسبه میکنیم:

```
x_mean_train = np.mean(x_train)
y_mean_train = np.mean(y_train)
```

سپس به محاسبه ضرایب b و w می پردازیم:

```
numerator = np.sum((x_train - x_mean_train) * (y_train - y_mean_train))
denominator = np.sum((x_train - x_mean_train) ** 2)
```

```
w = numerator / denominator
b = y_mean_train - w * x_mean_train
```

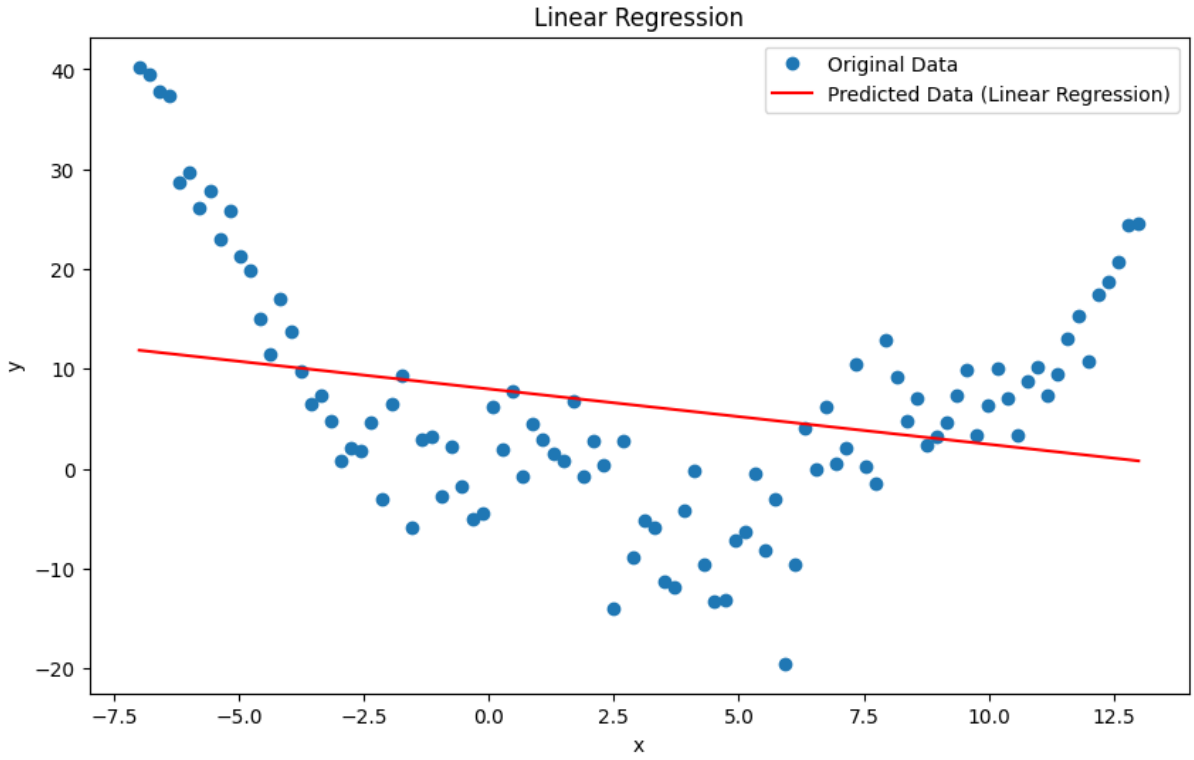
و آن ها را در معادله قرار می دهیم:

```
y_train_pred = w * x_train + b
y_test_pred = w * x_test + b
```

سپس هر سه خطای مذکور را محاسبه می کنیم:

```
mse_train = np.mean((y_train - y_train_pred) ** 2)
mse_test = np.mean((y_test - y_test_pred) ** 2)
mre_train = np.mean(np.abs((y_train - y_train_pred) / y_train))
mre_test = np.mean(np.abs((y_test - y_test_pred) / y_test))
rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)
```

و در باقی کد صرفا نتایج را نمایش می دهیم مدل خطی ما به شکل زیر در خواهد آمد:



Metric	Train	Test
MSE	130.7261901659013	155.97453673343995
MRE	4.509263780758496	1.924903396474474
RMSE	11.43355544727454	12.488976608731395

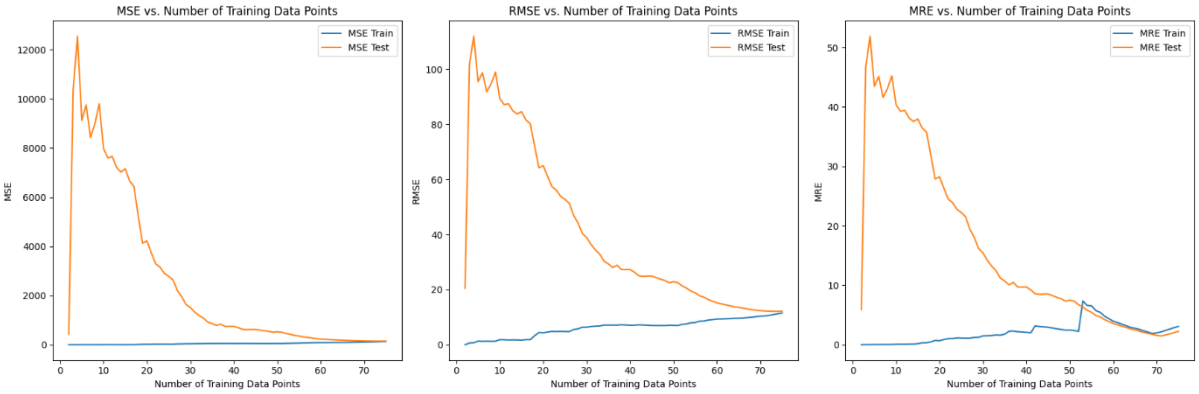
تحلیل: همانطور که مشاهده می شود خطای های مذکور بسیار بالا هستند و این داده ها با مدل بسیار ساده خطی قابل مدل کردن نیستند و باید از مدل های پیچیده تر و با درجات بالا تر استفاده کرد.

۲.۴-

به کمک حلقه زیر هر بار یک داده اضافه میکنیم و مجددا خطا های مذکور را محاسبه می کنیم:

```
for i in range(1, len(X_train) + 1):
    X_train_subset = X_train[:i]
    Y_train_subset = Y_train[:i]
```

مجددا مدل را در هر بار اضافه شدن داده آموزش داده و حاصل خطا ها را در سه نمودار نمایش می دهیم:



همانطور که از نمودار ها پیداست در اول که تعداد داده ها بسیار کم است همه مقدار زیادی دارند و افزایش داده ها کاهش پیدا کرده اند و از یک جایی به بعد تقریبا ثابت شده اند.

۲.۵-

استفاده از داده‌های بیشتر برای آموزش یک مدل یادگیری ماشین می‌تواند به کاهش خطا کمک کند، اما همیشه نمی‌توان تضمین کرد که خطای مدل به اندازه خطای انسان کاهش یابد. اگر خطای مدلی ۱۰ باشد یعنی این مدل underfit است و با افزایش داده ها میتواند عملکرد خیلی بهتری ارائه بدهد . البته باید توجه داشت که زیاد شدن دیتا ممکن است باعث شود مدل overfit شود که در صورت بروز این اتفاق حتی عملکرد بدتر هم میشود. بصورت خلاصه میتوانیم عملکرد مدل را با افزایش دیتاها بهتر کنیم اما قطعا نمیتوانیم از خطای انسان بهتر باشیم چون انسان توانایی درک پیچید گی ها و مسائل از قبل تعریف نشده را دارد. برتری مدل میتواند این باشد که کم هزینه تر از انسان و سریعتر از انسان است و میتواند با یک خطای محدودی نتیجه مطلوب ما را بدهد.

۲.۷-

1. Linear Regression (رگرسیون خطی)

این الگوریتم یکی از ساده‌ترین و پرکاربردترین مدل‌های رگرسیون است. فرض می‌کند که رابطه بین متغیر مستقل و متغیر وابسته خطی است و یک خط بهینه را بر اساس داده‌ها پیدا می‌کند. این خط به گونه‌ای تنظیم می‌شود که مجموع مربعات خطاها (Residual Sum of Squares) حداقل باشد .

$$\epsilon + {}_n\beta_n X + \ldots + {}_2\beta_2 X + {}_1\beta_1 X + {}_0\beta = y$$

مزایا:

- ساده و سریع است.
- برای تفسیر مناسب است و ضرایب به راحتی قابل درک هستند.

معایب:

- برای روابط غیرخطی مناسب نیست.
- حساس به مقادیر پرت و فرض نرمال بودن داده‌ها است.

۲. Ridge Regression (رگرسیون ریج)

رگرسیون ریج نوعی رگرسیون خطی است که برای جلوگیری از بیش‌برازش (Overfitting) از تنظیم (Regularization)استفاده می‌کند. این الگوریتم یک جریمه به مجموع مربعات ضرایب اضافه می‌کند تا پیچیدگی مدل را کنترل کن.

$$\text{Loss Function} = {}^2({}_i\hat{y} - {}_iy)\sum_{i=1}^{n} + {}^2_j\alpha \sum_{j=1}^{p}\beta$$

مزایا:

- کاهش بیش‌برازش در مدل‌هایی با تعداد ویژگی‌های زیاد.
- مناسب برای داده‌های همبسته.

معایب:

- مقادیر α\alphaباید بهینه‌سازی شوند.
- ممکن است برخی ویژگی‌های کم‌اثر را حذف نکند.

۳. Random Forest Regressor (رگرسیون جنگل تصادفی)

این الگوریتم از روش Baggingو درخت‌های تصمیم متعدد استفاده می‌کند. هر درخت بخشی از داده‌ها را یاد می‌گیرد و پیش‌بینی نهایی با میانگین‌گیری پیش‌بینی‌های تمام درخت‌ها به دست می‌آید.

مزایا:

- بسیار انعطاف‌پذیر و قدرتمند برای داده‌های غیرخطی.
- مقاوم در برابر بیش‌برازش.
- به طور طبیعی اهمیت ویژگی‌ها را اندازه‌گیری می‌کند.

معایب:

- محاسباتی پیچیده‌تر و کندتر از مدل‌های خطی.
- ممکن است تفسیر آن دشوار باشد.