به نام خدا

گزارش مینی پروژه ۱

مبانی سیستم های هوشمند

دکترعلیاری

پاییز ۱۴۰۳

فرزاد مقدم

۴۰۰۰۹۴۵۳

لینک مخزن گیت هاب: https://github.com/Farzadmoghaddam/Intelligent-Systems

لینک گوگل کولب سوال اول:

https://colab.research.google.com/drive/1ZkWXj6dXPdtaprSp4J1AZDPj5g_LWynB?usp=sharing

لینک گوگل کولب سوال دوم:

https://colab.research.google.com/drive/1dEDjr5_Rp1JaiOcw9Jsfyo9YoBrJ52BT?usp=sharing

کد های اولیه:

- نصب کتابخانه شامل تابع gdown و فراخوانی آن
- آدرس دهی فایل دیتاست از طریق آی دی فایل در گوگل درایو و دسترسی از طریق تابع gdown
- ایمپورت کتابخانه های ضروری numpy، pandas، matplotlib، seaborn و ...
- خواندن فایل دیتاست csv و ریختن آن در df
- نمایش ۵ سطر اول دیتاست برای مشاهده کلی دیتا

توجه:

همانطور که در قسمت اطلاعات مربوط به دیتاست در سایت ککگل نیز اشاره شده است برخی از ستون های داده که ساختگی بوده و باعث پیش بینی با دقت ۱۰۰ درصدی میشوند را در همان ابتدا قبل از انجام هرکاری به صورت دستی حذفشان کرده و سپس فایل در گوگل درایو آپلود شد.

در رسم نمودار همبستگی ویژگی ها با یکدیگر نیز این ارتباط نزدیک و ضریب هبستگی ۱ بین ستون های اشاره شده و تارگت مشاهده میشد.

در صورت عدم حذف آن ستون ها پس از آموزش مدل مدنظر، خروجی با دقت ۱۰۰ درصدی پیش بینی میشد.

پرسش اول

۱.۱

خلاصه ای از داده:

این مجموعه داده شامل اطلاعاتی درباره ۱۰,۱۲۷ مشتری از یک بانک است که شامل ۲۰ ویژگی مختلف به همراه ۱ ستون تارگت است. مدیریت بانک قصد دارد که با استفاده از این مجموعه داده با ویژگی هایی از قبیل سن، حقوق، وضعیت تاهل، سقف اعتبار کارت، نوع حساب و ... بتوان پیش بینی کرد که کدام یک از مشتریان متمایل به ترک حساب اعتباری و عدم استفاده از حساب خود هستند تا در صورت شناسایی این دسته افراد، نسبت به بهتر کردن خدمات بانکی در برابر این افراد اقدام کنند تا از ترک حساب خود منصرف شوند.

نرخ ترک مشتریان طبق گفته وب سایت کگل در این بانک حدود ۱۶.۰۷٪ است، که کار را برای مدل‌های پیش‌بینی کمی دشوارتر می‌کند.

ویژگی ها:

- **CLIENTNUM(شماره مشتری):شماره منحصربفرد برای هر فردی که در بانک حساب اعتباری دارد.**
- **Attrition_Flag(وضعیت باز یا بسته بودن حساب)(تارگت)**
- **Customer_Age(سن)**
- **Gender(جنسیت)**
- **Dependent_count(تعداد وابستگی ها)**
- **Education_Level(میزان تحصیلات)**
- **Marital_Status(وضعیت تاهل)**
- **Income_Category(درآمد سالانه)**
- **Card_Category(نوع حساب و کارت اعتباری)**
- **Months_on_book(مدت زمان همکاری با بانک)**
- **Total_Relationship_Count**
- **Months_Inactive_12_mon**
- **Contacts_Count_12_mon**
- **Credit_Limit**
- **Total_Revolving_Bal**
- **Avg_Open_To_Buy**
- **Total_Amt_Chng_Q4_Q1**
- **Total_Trans_Amt**

- Total_Trans_Ct
- Total_Ct_Chng_Q4_Q1
- Avg_Utilization_Ratio

تعداد ۱۰۱۲۷ نمونه در این مجموعه داده موجود است.

```
df.shape
(10127, 21)
```

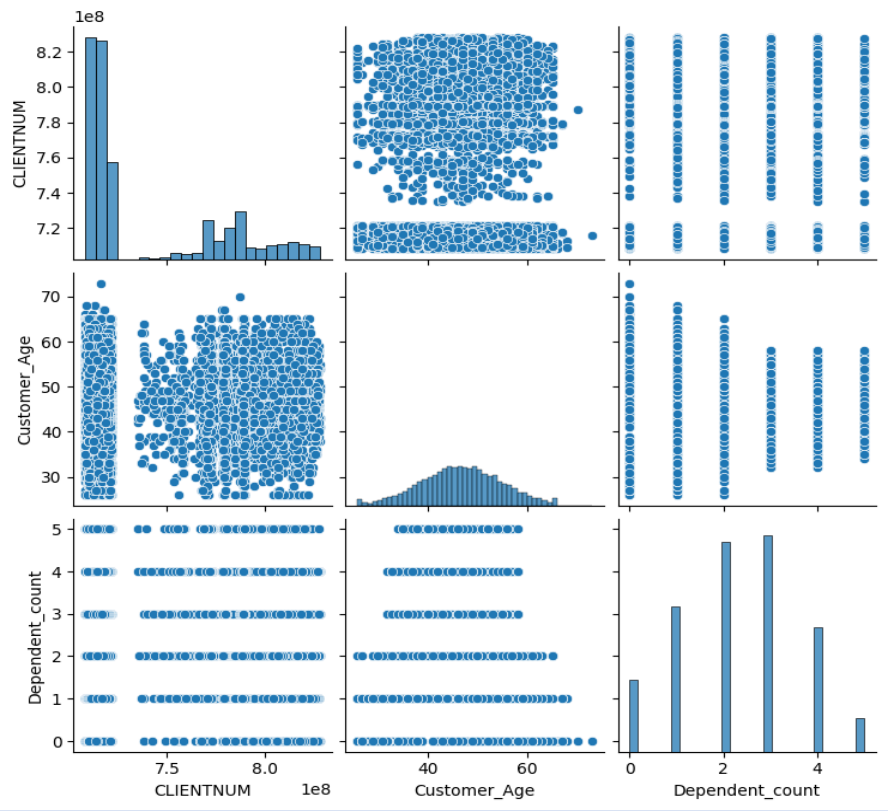
۲.۱

نمایش پخش داده

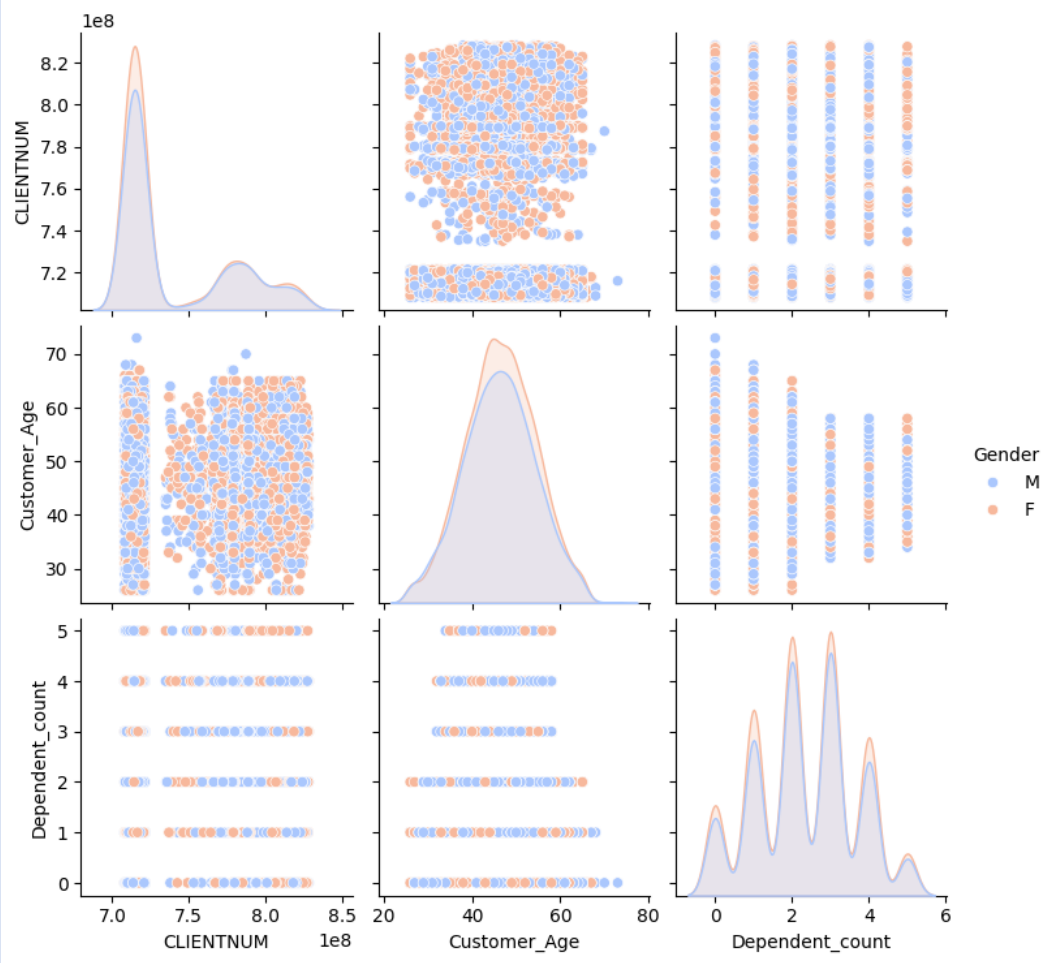
ویژگی های انتخابی:

“CLIENTNUM” “Attrition_Flag” “Customer_Age” “Gender” “Dependent_coun”

نمایش داده ها به صورت نمودار پخش داده برای نگاه کلی و سریع به نحوه پخش داده ها بسیار کاربردی است.
Sns.pairplot در حالت دیفالت به نمایش پخش داده ستون هایی که مقادیر عددی دارند میپردازد و به همین خاطر از ۵ ستون داده انتخابی از دیتاست خروجی نمودار pairplot متشکل از ۹ نمودار است و در واقع ۲ ستون از ویژگی هایمان(“Attrition_Flag“و”Gender“)مقادیر عددی ندارند و categoricalهستند در نمودار اولیه نشان داده نمیشوند.
برای نمایش پخش داده این نوع از داده ها(categorical) میتوان به مقادیر عددی تبدیلشان کرد(Encoding) و در دیتاست جاگذاری کرد و یا اینکه به صورت رنگی در نمایش و با استفاده ازآپشن (hue) پخش داده به نمایش آنها پرداخت.



نمایش پخش داده ۳ ستون عددی از ۵ ستون انتخابی



نمایش پخش داده با اعمال نمایش رنگی ستون جنسیت

همانطور که مشاهده میشود **pairplot** هر ستون نسبت به خودش به صورت هیستوگرام رسم شده است.

هر نمودار نسبت تغییرات یک ویژگی نسبت به سایر ویژگی ها را رسم میکند و رنگ ها نشان دهنده جنسیت هستند.

۳.۱

همبستگی

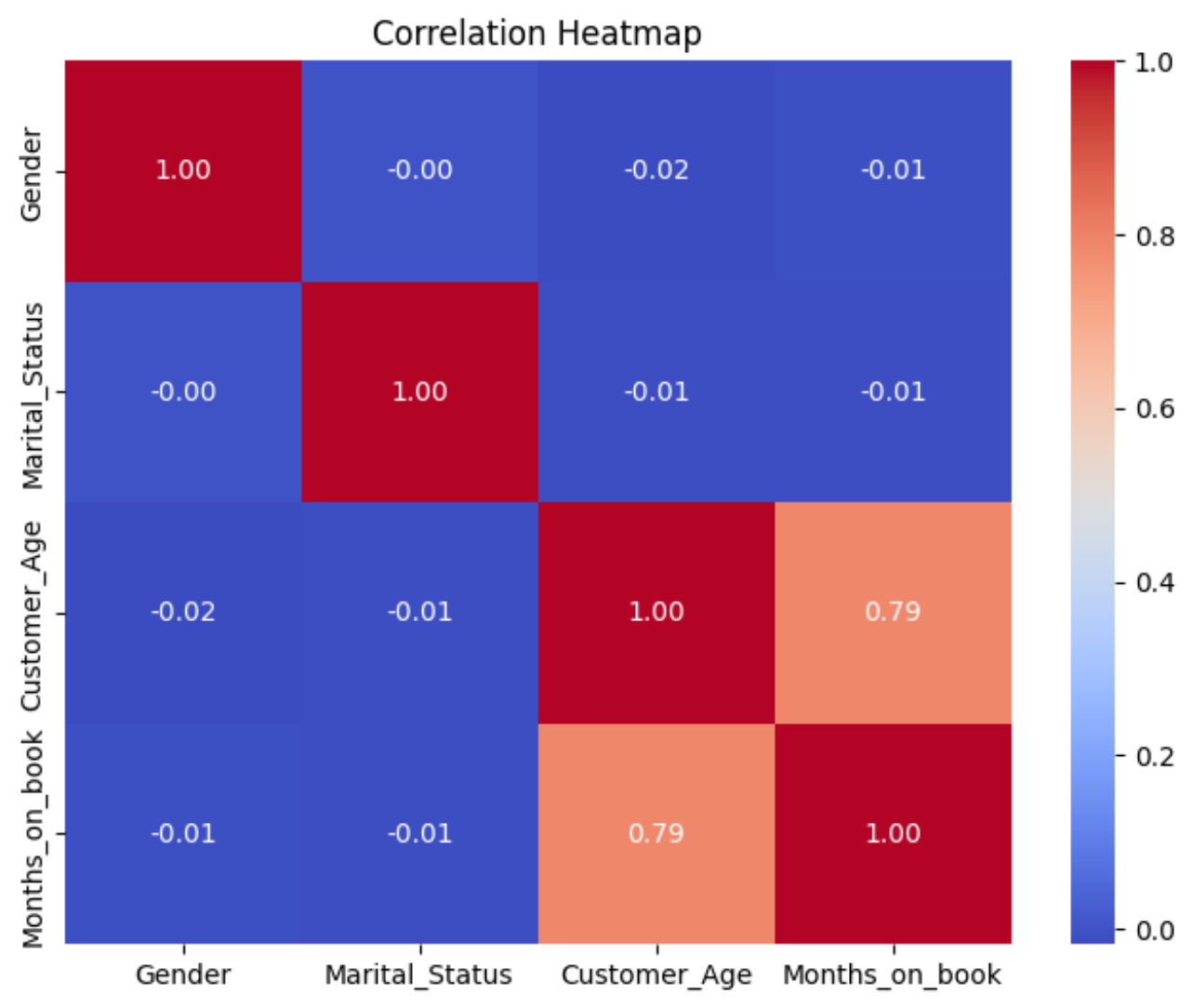
ویژگی های طبقه بندی شده انتخابی: Gender و Marital_Status

ویژگی های پیوسته انتخابی: Customer_Age و Months_on_book

برای این کار در ابتدا نیازمند تبدیل مقادیر طبقه بندی شده ستون های کلاس بندی شده به عدد هستیم پس مقادیر این ستون را به اعداد گسسته تبدیل میکنیم .

برای این کار ابتدا تعداد کلاس های موجود در هر ستون را با استفاده از تابع **value_counts** میشماریم و سپس با استفاده از ابزار **LabelEncoder** واقع در کتابخانه **sklearn.preprocessing** مقادیر متنی را تبدیل به عدد میکنیم و در داخل دیتاست به جای ستون های قبلی جاگذاری میکنیم.

حال با استفاده از **corr.** به محاسبه همبستگی بین ۴ ستون انتخابی میپردازیم و سپس نمودار حرارتی را نمایش میدهیم.



بدیهی است همبستگی هر ستون با خود مقدار ۱ را داشته باشد.

همبستگی قابل توجهی (۰.۷۹) میان سن و زمان همکاری فرد با بانک دیده میشود.

سایر نتایج نمایش داده شده از ستون های اتخابی نزدیک به ۰ بوده و مقادیر قابل توجهی نیستند.

۴.۱

با یک نگاه کلی به دیتاست مشاهده میشود که در برخی از نمونه ها به جای ویژگی مورد نظر از واژه **Unknown** استفاده شده است که آنها را به عنوان نمونه های ناکامل در نظر میگیریم و سطر های شامل این نوع ازداده ها را از دیتاست کلی حذف میکنیم.

بعد از یافتن نمونه های مورد نظر و حذف آنها، دیتاست نهایی به این تعداد از سطر ها کاهش میابد.

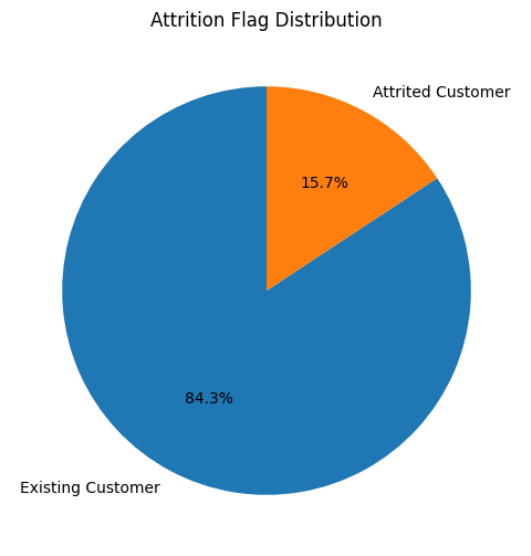
```
df.shape
(7081, 21)
```

۵.۱

ستون **Attrition_Flag** دارای ۲ کلاس با نام های **Existing Customer** و **Attrited Customer** است.

Attrition_Flag	
Existing Customer	5968
Attrited Customer	1113

رسم نمودار دایره ای پخش داده در این ستون



همانطور که نشان داده شد نزدیک به ۱۵.۷ درصد از مشتریان این بانک، همکاریشان با بانک را قطع کرده اند.

با توجه به عدم تعادل بین داده های دو کلاس از این ستون(۵۹۶۸داده در مقابل ۱۱۱۳!!!) ، این ویژگی دارای عدم تعادلی(**unbalancing**) است.

وجود عدم تعادل در عملکرد نهایی مدل تاثیر دارد زیرا که در این صورت مدلمان در حین یادگیری، کلاسی را که فراوانی زیادی دارد را بیشتر یاد میگیرد و در مقابل کلاس دارای فراوانی کم را کمتر یاد خواهد گرفت و در نتیجه دقت پیش بینی کلاس با فراوانی کمتر در حین ارزیابی مدل کمتر خواهد بود و ممکن است داده ای که متعلق به کلاس با فراوانی کمتر است را به کلاس با فراوانی بیشتر نسبت دهد.

همچنین در این حالت ممکن است در نهایت مدلمان دارای دقت بالایی باشد منتها داشتن دقت بالا در صورت وجود ناتعادلی در بین کلاس ها بیانگر مدل با عملکرد بهتر نیست. چه بسا مدل تمامی داده های تست موجود در کلاس با فراوانی کمتر را اشتباه پیش بینی کرده باشد ولی به دلیل عدم تعادل بین تعداد داده های کلاس ها این عملکرد اشتباه در حین محاسبه دقت مدل به چشم نیاید که برای ارزیابی همچین حالتی باید به سراغ تحلیل ماتریس درهم ریختگی و recall و F1-Score برویم.

راهکار های اصلاح عدم تعادلی

- **Undersampling: کم کردن تعداد داده های کلاس با تعداد فراوانی بیشتر مثل حذف داده های تکراری و یا پرت**
- **Oversampling: تولید داده های تصادفی و به اصطلاح تولید نویز حول داده های کلاسی که فراوانی کمتری دارد.**
- **Combination: اعمال هر دو روش به صورت ترکیبی**
- **Algorithmic Approaches: استفاده از الگوریتم هایی که به خوبی با داده های نامتعادل کار میکنند مثل الگوریتم های Random Forest ، XGBoost ، LightGBM که با وزن دهی به کلاس ها با تعداد فراوانی کمتر ، توزیع داده را به حالت تعادلی می رسانند.**
- **Cost-Sensitive Learning: اعمال هزینه بیشتر برای اشتباهات در پیش بینی کلاس با فراوانی کمتر به این ترتیب مدل تشویق میشود که دقت بیشتری در پیش بینی این نوع کلاس ها داشته باشد.**
- **استفاده از F1-Score به عنوان معیار ارزیابی برای داده های نامتوازن مناسب تر است.**

اگر متعادلسازی قبل از اسپلیت انجام شود، داده‌های validation و test توزیع مشابهی با داده‌های Train خواهند داشت. این امر باعث می‌شود ارزیابی عملکرد مدل دقیق‌تر باشد.

به عنوان مثال در اعمال روش Undersampling چون با کاهش تعداد نمونه ها روبه رو هستیم پس بهتر است این عمل ابتدا بر روی کل دیتاست اعمال شود و سپس به تقسیم دیتاست به ۳ بخش اقدام کنیم.

۶.۱

حالت ۱: بدون متعادل سازی داده ها

در ابتدا تمامی کلاس های categorical را به صورت عددی تبدیل میکنیم.

دیمانسیون و هد دیتا ست رو چک میکنیم و مشاهده میکنیم که نامتعادل اند.

سپس ستون فیچر ها (X) و تارگت(Y) را مشخص میکنیم.

دیمانسیون x و y رو چک میکنیم.

داده های x را نرمالایز میکنیم.

x و y را با train_test_split تقسیم بندی میکنیم با randon_state=53 و نسبت ۲۰ درصد به main و test

بار دیگر برای به دست آوردن داده ارزیابی اینبار داده های بخش main را به دو بخش train و vali تقسیم میکنیم.

در ادامه مدل طبقه بندی xgboost و همچنین accuracy_score, classification_report, confusion_matrix را فراخوانی میکنیم.

سپس مدل را با داده آموزش، فیت میکنیم.

همانطور که مشاهده میشود مدل طبقه بندی xgboost تمامی داده های آموزش را به خوبی یاد گرفته است و دقت ۱۰۰ درصدی دارد.الارغم اینکه در بکارگیری مدل هایی طبقه بندی logesticregression و svm ، دقت پیش بینی برای داده های آموزش نزدیک ۸۰درصد بود.

ماتریس درهم ریختگی بیانگر آن است که تمامی مقادیر در دو کلاس به درستی پیش بینی شده اند.

ارزیابی نتایج داده آموزش حالت ۱:

```
train acuuracy is 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       815
           1       1.00      1.00      1.00      4300

    accuracy                           1.00      5115
   macro avg       1.00      1.00      1.00      5115
weighted avg       1.00      1.00      1.00      5115

[[ 815    0]
 [   0 4300]]
```

ارزیابی نتایج داده تست حالت ۱ :

```
test acuuracy is 0.9717779868297272
              precision    recall  f1-score   support

           0       0.93      0.88      0.90       158
           1       0.98      0.99      0.98       905

    accuracy                           0.97      1063
   macro avg       0.95      0.93      0.94      1063
weighted avg       0.97      0.97      0.97      1063

[[139  19]
 [ 11 894]]
```

ارزیابی نتایج داده اعتبارسنجی حالت ۱ :

```
validation acuuracy is 0.8983050847457628
              precision    recall  f1-score   support

           0       0.77      0.52      0.62       227
           1       0.91      0.97      0.94      1189

    accuracy                           0.90      1416
   macro avg       0.84      0.74      0.78      1416
weighted avg       0.89      0.90      0.89      1416

[[ 117  110]
 [  34 1155]]
```

حالت ۲: با متعادل سازی داده ها

- **Undersampling: کم کردن تعداد داده های کلاس با تعداد فراوانی بیشتر مثل حذف داده های تکراری و یا پرت**

Attrition_Flag	count
1	5968
0	1113

با توجه به فراوانی کلاس های تارگت و مقایسه آنها متوجه میشویم که نا متعادل است.

ابتدا کلاس هارا از هم جداسازی میکنیم.

به دو گروه y_Existing و y_Attrited جداسازی میکنیم.

تعداد اعضای کلاس با تعداد فراوانی بیشتر را به اندازه تعداد فراوانی کلاس دیگر کاهش میدهیم و y_Existing_New را میسازیم.

برای ساختن ستون تارگت جدید کلاس های جدید را به هم میچسبانیم و y_new را میسازیم.

عملیات مشابه قسمت قبل را برای دیتا جدید انجام میدهیم و با همان مدل طبقه بندی داده ها را فیت میکنیم.

ارزیابی نتایج داده آموزش حالت ۲:

```
new train acuuracy is 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       714
           1       1.00      1.00      1.00       710

    accuracy                           1.00      1424
   macro avg       1.00      1.00      1.00      1424
weighted avg       1.00      1.00      1.00      1424

[[714   0]
 [  0 710]]
```

ارزیابی نتایج داده تست حالت ۲:

```
new test acuuracy is 0.968609865470852
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       224
           1       0.96      0.98      0.97       222

    accuracy                           0.97       446
   macro avg       0.97      0.97      0.97       446
weighted avg       0.97      0.97      0.97       446

[[215   9]
 [  5 217]]
```

ارزیابی نتایج داده اعتبارسنجی حالت ۲:

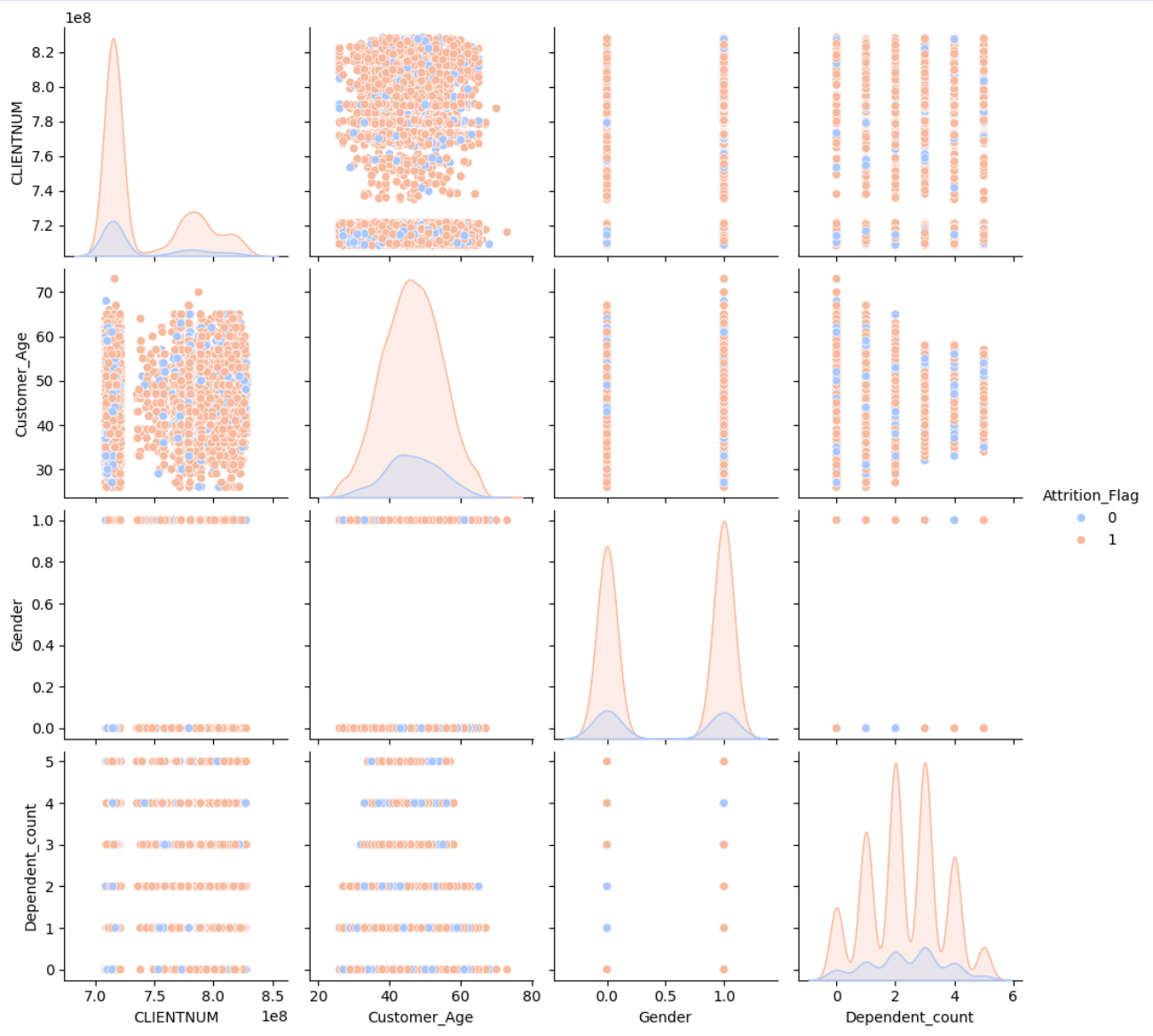
```
new val acuuracy is 0.952247191011236
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       175
           1       0.96      0.94      0.95       181

    accuracy                           0.95       356
   macro avg       0.95      0.95      0.95       356
weighted avg       0.95      0.95      0.95       356

[[168   7]
 [ 10 171]]
```

نتیجه گیری:

با انجام عمل متعادل سازی دقت مدل در تفکیک TP و TN ها بهتر شد و دقت مدل به مراتب بهبود یافت.

به قیمت کاهش تعداد نمونه ها در کلاس با فراوانی بیشتر

امتیازی



نمایش پخش داده روی ستون های ابتدایی بر اساس کلاس های مختلف ویژگی Attrition_Flag

پرسش دوم

۱.۲

در ابتدا دیتافریم پاندا متشکل از دو ستون **Index** و **Target** را تشکیل میدهیم.

	Index	Target
0	0	40.251965
1	1	39.530101
2	2	37.799217
3	3	37.328371
4	4	28.653943
...	...	...

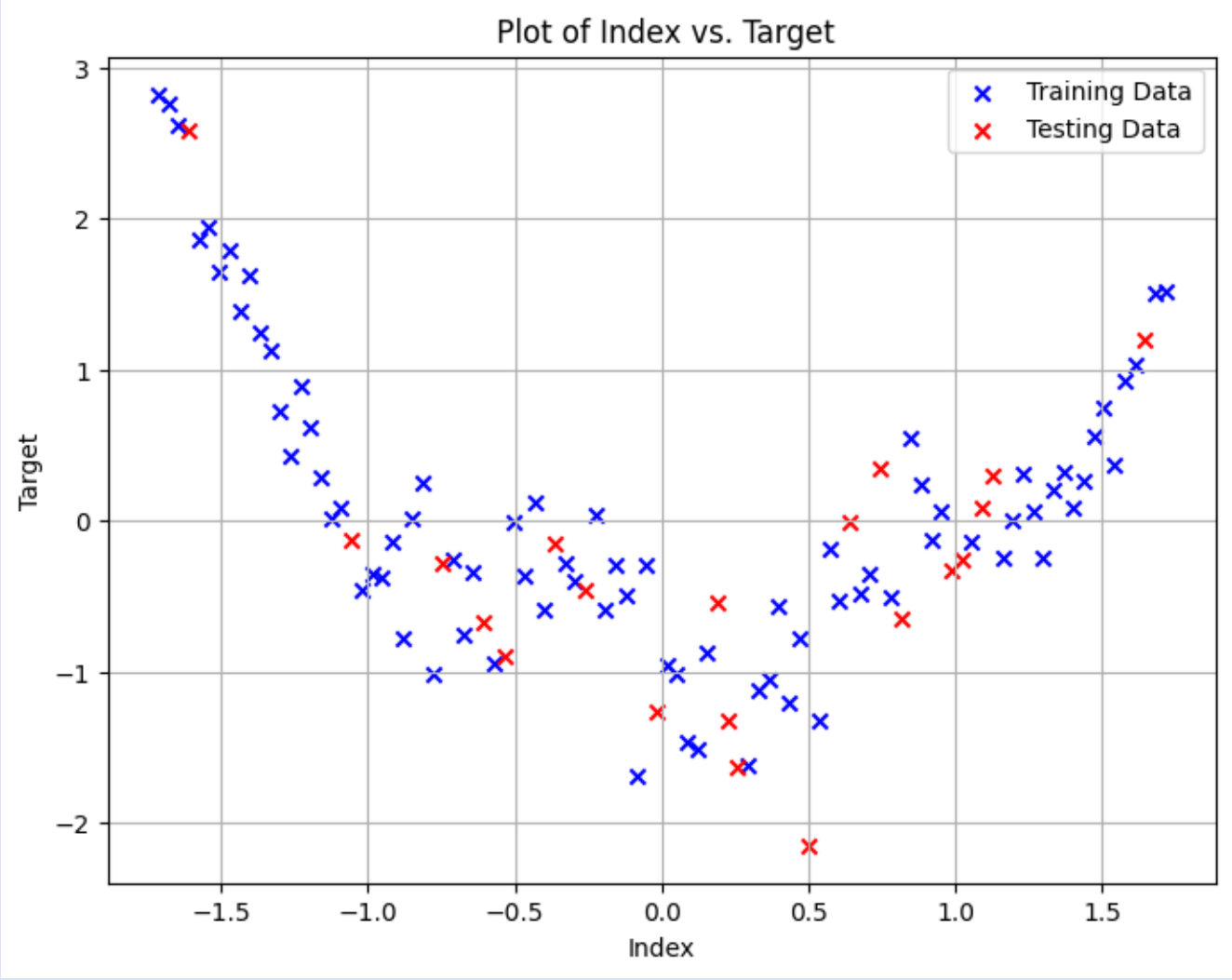
```
[88] df.shape

     (100, 2)
```

سپس ستون های **x** و **y** را تشکیل میدهیم.

داده های **x** و **y** را به دو بخش آموزش و تست تقسیم میکنیم.

```
x_train shape (80, 1)
x_test shape (20, 1)
y_train shape (80, 1)
y_test shape (20, 1)
```

پخش داده های آموزش و تست:



۲.۲

معیار های سنجش عملکرد مدل های رگرسیون:

۱. **میانگین قدر مطلق خطا(MAE): میانگین قدر مطلق تفاوت بین مقادیر پیش‌بینی شده و مقادیر واقعی. این معیار به مقیاس داده‌ها وابسته نیست.**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

۲. **میانگین مربعات خطا(MSE): . این معیار به خطاهای بزرگتر وزن بیشتری می‌دهد. این معیار به خطاهای بزرگتر وزن بیشتری می‌دهد.**

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

۳. جذر میانگین مربعات خطا(**RMSE**)

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

۴. **R-squared**: نشان می‌دهد که چه مقدار از واریانس متغیر وابسته توسط مدل توضیح داده شده است.

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

۵. خطای درصد میانگین مطلق(**MAPE**): میانگین قدر مطلق خطاهای درصد

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

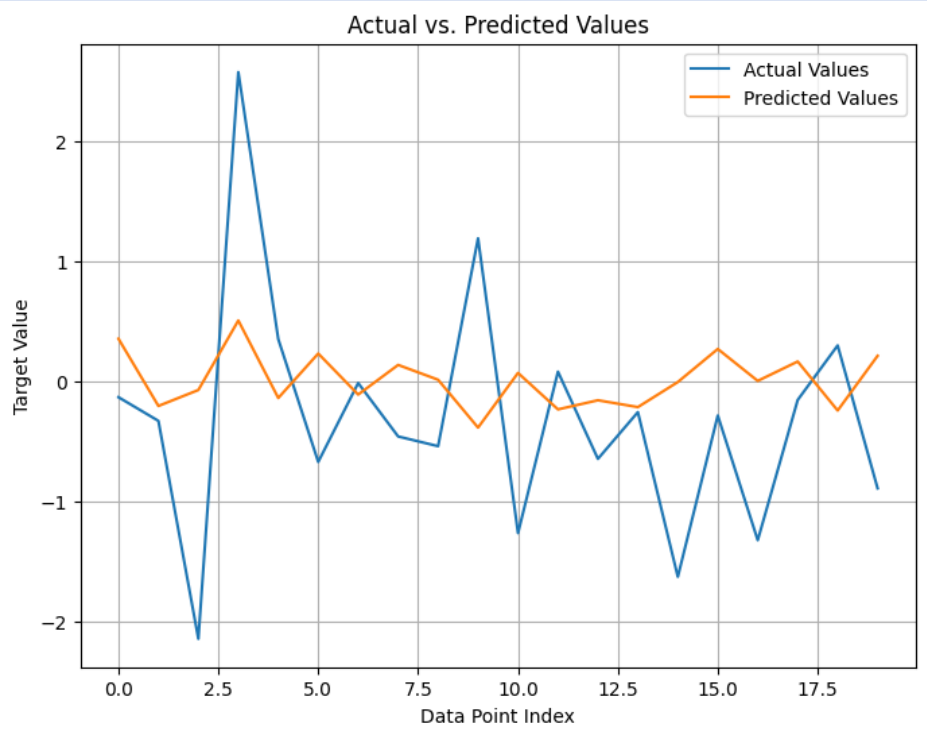
۶. خطای ریشه میانگین مربعات لگاریتمی(**RMSLE**): برای لگاریتم مقادیر پیش‌بینی شده و واقعی. این معیار برای زمانی که مقادیر واقعی بسیار متفاوت هستند، مناسب است.

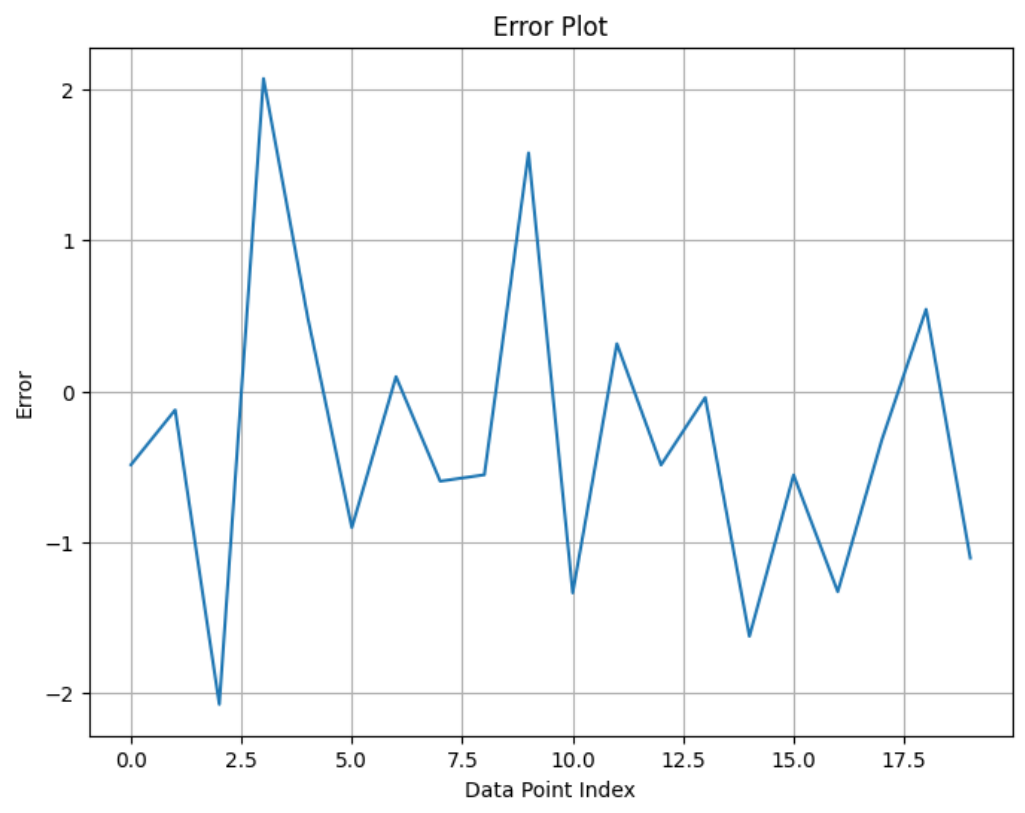
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(x_i+1) - \log(y_i+1))^2}$$

```python
class Linear_Regression():

  def __init__(self, learning_rate, no_of_iterations):
    self.learning_rate = learning_rate
    self.no_of_iterations = no_of_iterations


  def fit(self, x, y):
    self.m, self.n = x.shape
    self.x = x
    self.y = y
    self.w = np.zeros(self.n)
    self.b = 0

    for i in range(self.no_of_iterations):
      self.update_weights()


  def update_weights(self,):
    y_pred = self.predict(self.x)

    dw = - (2 * (self.x.T).dot(self.y - y_pred)) / self.m
    db = - 2 * np.sum(self.y - y_pred) / self.m

    self.w = self.w - self.learning_rate * dw
    self.b = self.b - self.learning_rate * db


  def predict(self,x):
     return (x.dot(self.w) + self.b).reshape(x.shape[0], 1)
```

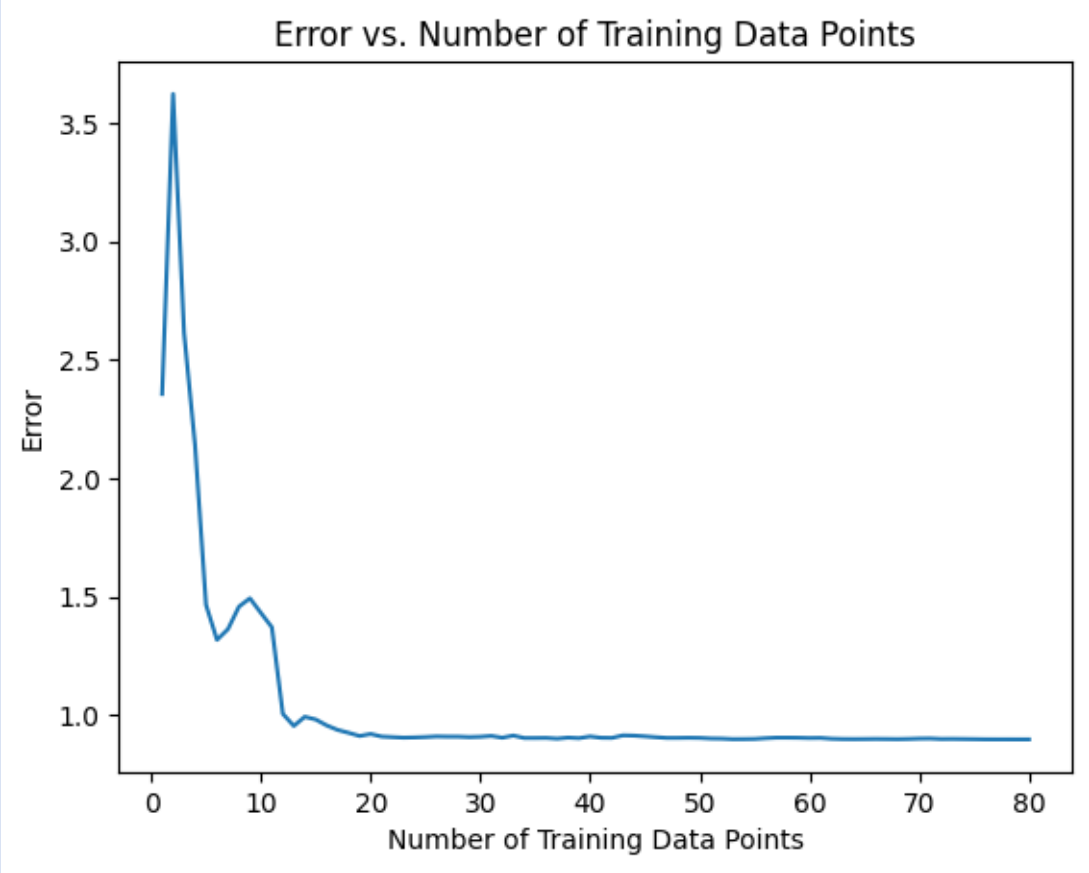
با توجه به نمودار پخش داده میتوان مشاهده کرد که پخش داده به صورت خطی نیست و مدلی با تابع از درجات بالاتر قابلیت بهتری در تخمین داده ای ما خواهد داشت به شرطی که مدل **overfit** نشود.

```
R-squared: -0.10772618866526651
Mean Squared Error: 1.0757031793617677
Mean Absolute Error: 0.8315543992561143
```


Actual vs. Predicted Values
Actual Values
Predicted Values
Target Value
Data Point Index
2
1
0
−1
−2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Error Plot
Error
Data Point Index
2
1
0
−1
−2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

۴.۲



با افزایش تعداد داده های آموزش در ابتدا خطا رفته رفته کاهش می یابد تا جایی که دیگر به نزدیکی های مینیمم لوکال رسیده و مقدار خطا ثابت میماندو دیگر کمتر نمیشود. اما نزدیک صفر است.

۵.۲

در حالت کلی، افزایش داده آموزشی می تواند به بهبود عملکرد مدل و کاهش خطای آن کمک کند. اما اینکه آیا می توانیم خطای مدل را به اندازه خطای انسان کاهش دهیم، به عوامل مختلفی بستگی دارد:

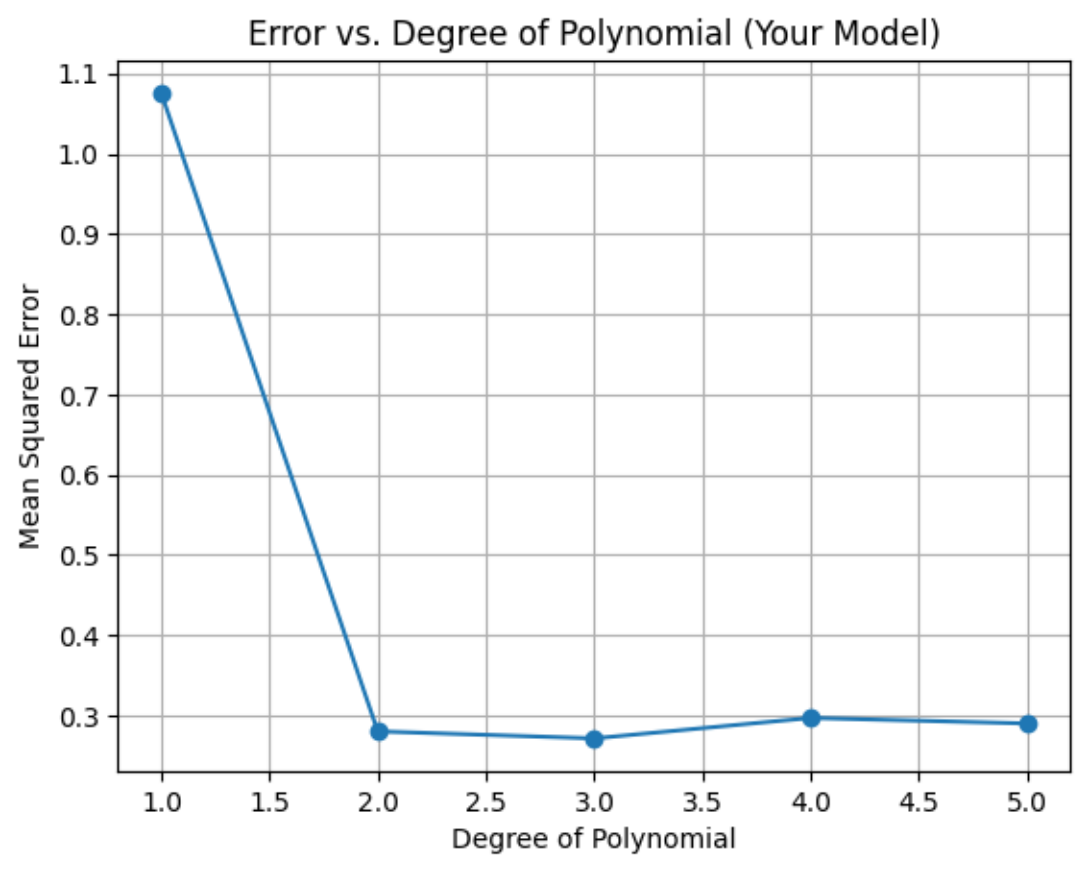
پیچیدگی فعالیت: اگر فعالیت بسیار پیچیده باشد و انسان به طور ذاتی در آن برتری قابل توجهی داشته باشد، ممکن است رسیدن به خطای ۱ برای مدل دشوار باشد، حتی با داده های بسیار زیاد.

کیفیت داده: داده آموزشی باید نماینده خوبی از توزیع واقعی داده ها باشد. اگر داده های آموزشی نویز زیادی داشته باشند یا دارای توزیع متفاوتی نسبت به داده های واقعی باشند، افزایش داده ممکن است کمکی به کاهش خطا نکند.

ظرفیت مدل: مدل یادگیری ماشین باید ظرفیت کافی برای یادگیری پیچیدگی های فعالیت را داشته باشد. اگر مدل خیلی ساده باشد، ممکن است نتواند به خطای ۱ برسد، حتی با داده های فراوان.

الگوریتم یادگیری: الگوریتم یادگیری مورد استفاده نیز در عملکرد مدل تاثیر دارد. برخی از الگوریتم ها برای فعالیت های خاص مناسب تر هستند.

۶.۲



نمودار خطا بر حسب تعداد جملات چندجمله ای

نه، لزوما با افزایش تعداد جمله های مدل، خطای آزمون همواره کاهش نمی یابد. در ابتدا، با افزایش تعداد جمله های مدل (مانند افزایش درجه چندجمله ای)، خطای آزمون ممکن است کاهش یابد. این به دلیل این است که مدل پیچیده تر شده و قادر است اطلاعات بیشتری از داده ها را یاد بگیرد و الگوهای پیچیده تری را در آنها تشخیص دهد. در نتیجه، مدل به طور دقیق تری داده های آزمون را پیش بینی می کند و خطای آزمون کاهش می یابد. اما با ادامه افزایش تعداد جمله های مدل، خطای

آزمون در نهایت شروع به افزایش می کند و(Overfitting) رخ میدهد. در این حالت، مدل آنقدر پیچیده می شود که به جای یادگیری الگوهای اصلی داده ها، شروع به یادگیری نویز و ویژگی های تصادفی موجود در داده های آموزشی می کند. در نتیجه، مدل به خوبی نمی تواند داده های جدید و دیده نشده (مانند داده های آزمون) را پیش بینی کند و خطای آزمون افزایش می یابد.

۷.۲

Ridge:

Ridge Regression یک نوع رگرسیون خطی است که از regularization برای جلوگیری از overfitting استفاده می‌کند. در واقع، Ridge یک نسخه بهبود یافته از رگرسیون خطی معمولی است که در آن یک عبارت پنالتی به تابع هزینه اضافه می‌شود.

Lasso

Lasso (Least Absolute Shrinkage and Selection Operator) یک نوع رگرسیون خطی است که از regularization برای جلوگیری از overfitting و همچنین انتخاب ویژگی استفاده می‌کند. در واقع، Lasso مشابه Ridge Regression است، اما نوع regularization مورد استفاده در آن متفاوت است.

ElasticNet

ElasticNet یک نوع رگرسیون خطی است که از ترکیبی از L1 و L2 regularization برای جلوگیری از overfitting و انتخاب ویژگی استفاده می‌کند. در واقع، ElasticNet با ترکیب مزایای Ridge Regression و Lasso Regression، یک مدل قدرتمند و انعطاف‌پذیر ایجاد می‌کند.

```
from sklearn.linear_model import LinearRegression
y_pred1 = LinearRegression().fit(x_train, y_train).predict(x_test)
r2 = r2_score(y_test, y_pred1)
mse = mean_squared_error(y_test, y_pred1)
mae = mean_absolute_error(y_test, y_pred1)
print(f"R-squared: {r2}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```

```
R-squared: -0.10772618857510308
Mean Squared Error: 1.0757031792742109
Mean Absolute Error: 0.8315543992204759
```

```python
from sklearn.linear_model import Ridge
y_pred2 = Ridge(alpha=1.0).fit(x_train, y_train).predict(x_test)
r2 = r2_score(y_test, y_pred2)
mse = mean_squared_error(y_test, y_pred2)
mae = mean_absolute_error(y_test, y_pred2)
print(f"R-squared: {r2}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```

```
R-squared: -0.10765288816448892
Mean Squared Error: 1.075631997888817
Mean Absolute Error: 0.8311075723294457
```

```python
from sklearn.linear_model import Lasso
y_pred3 = Lasso(alpha=1.0).fit(x_train, y_train).predict(x_test)
r2 = r2_score(y_test, y_pred3)
mse = mean_squared_error(y_test, y_pred3)
mae = mean_absolute_error(y_test, y_pred3)
print(f"R-squared: {r2}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```

```
R-squared: -0.15808351193192705
Mean Squared Error: 1.1246047340025997
Mean Absolute Error: 0.8022805840449205
```

```python
from sklearn.linear_model import ElasticNet
y_pred4 = ElasticNet(alpha=1.0, l1_ratio=0.5).fit(x_train, y_train).predict(x_test)
r2 = r2_score(y_test, y_pred4)
mse = mean_squared_error(y_test, y_pred4)
mae = mean_absolute_error(y_test, y_pred4)
print(f"R-squared: {r2}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```

```
R-squared: -0.15808351193192705
Mean Squared Error: 1.1246047340025997
Mean Absolute Error: 0.8022805840449205
```

در lasso و Elastic مقدار MAE ۰.۸۰ بودخ و کمتر از مقدار در Ridge و LR است.در عوض MSE در LR و Ridge نسبت به lasso و Elastic کمتر است.

۷.۲

درباره reguralization (منظم سازی)

به منظور جلوگیری از overfitting در آموزش داده های آموزش و فیت شدن مدل از رویکرد منظم سازی(regularization) برای تنظیم مدل و در نتیجه برازش بر داده‌های جدید استفاده می‌شود.

بهره‌گیری از تکنیک‌های منظم‌سازی برای به حداقل رساندن احتمال overfitting و مطمئن شدن از عملکرد بهینه مدل ضرورت دارد.همانطور که میدانیم مدل overfit شده دقت پایینی در پیش بینی داده های تست خواهد داشت. در حقیقت منظم‌سازی یا regularization نوعی رگرسیون است که در آن ضرایب مدل به صفر میل می‌کنند. به بیان دیگر این تکنیک برای جلوگیری از ریسک overfitting، مدل را از یادگیری الگوهای پیچیده و دشوار منع می‌کند.

در واقع در فرایند منظم‌سازی، مقداری به عنوان جریمه به تابع زیانی که مدل قصد کمینه کردن آن را دارد اضافه می‌شود. این جریمه، مدل را مجبور به پایین نگه داشتن مقدار پارامترها (مانند مقادیر وزنی در شبکه‌های عصبی یا ضرایب در رگرسیون) می‌کند و به این شکل از وقوع overfitting جلوگیری می‌شود.

نقش regularization در یادگیری ماشین چیست؟

جلوگیری از بیش برازش: مهم‌ترین نقش regularization در یادگیری ماشین، جلوگیری از بیش‌برازش مدل است. مشکل رایجی که بر اساس آن، مدل بیش از حد نسبت به مجموعه آموزشی حساس می‌شود. در نتیجه دقت مدل برای مجموعه آموزشی بسیار بالا و در مواجه با داده‌های جدید بسیار پایین است. منظم‌سازی یا regularization با جریمه مقادیر وزنی بزرگ، احتمال بیش‌برازش را کاهش می‌دهد.

بهبود قابلیت تعمیم‌پذیری مدل: منظم‌سازی با ساده‌سازی مدل، باعث ارتقاء سطح عملکرد آن نسبت به داده‌های آموزشی و جدید می‌شود. چنین مدلی به‌جای تمرکز بر جزییات مجموعه آموزشی، الگوهای پایه دیتاست را شناسایی و استخراج می‌کند.

مدیریت هم‌خطی چندگانه: عمده کاربرد regularization زمانی است که همبستگی بالایی میان ویژگی‌ها وجود داشته باشد. به عنوان مثال، منظم‌سازی L2 یا ستیغی مقدار بالا واریانس ضرایب را کاهش می‌دهد. به این صورت دقت پیش‌بینی‌های مدل افزایش می‌یابد.

انتخاب ویژگی: تکنیک منظم‌سازی L1 یا لاسو نقش مهمی در جریمه ضرایب مدل دارد. تا جایی که مقدار برخی از ویژگی‌ها با صفر برابر شده و زیرمجموعه کوچکتری از ویژگی‌ها باقی می‌ماند. اهمیت این کاربرد زمانی مشخص می‌شود که انتخاب ویژگی امری لازم و ضروری برای ساده‌سازی و افزایش بهره‌وری مدل باشد.

پیشگیری از نویز: منظم‌سازی، حساسیت مدل را نسبت به ویژگی‌های خاص مجموعه آموزشی مانند نویز و مقادیر پَرت کاهش می‌دهد و در عوض مدل بر ویژگی‌های کاربردی و موثر در پیش‌بینی متمرکز می‌شود.

تغییر واریانس به سوگیری: همزمان با کاهش احتمال بیش‌برازش، تکنیک regularization سوگیری یا «بایاس (Bias) «مدل را افزایش می‌دهد. موازنه‌ای که در صورت پیچیدگی بالا مدل می‌تواند مفید باشد.

بهره‌گیری از مدل‌های پیچیده: با پیاده‌سازی regularization امکان استفاده از مدل‌های پیچیده‌تر مهیا می‌شود. برای مثال در شبکه‌های عصبی عمیق از تکنیک «حذف تصادفی (Dropout) «برای جلوگیری از بیش برازش کمک می‌گیرند.

تسهیل همگرایی: منظم‌سازی از جمله رویکردهای مفید برای همگرایی سریع و راحت‌تر مدل‌هایی است که از تکنیک گرادیان کاهش استفاده می‌کنند.

انواع regularization در یادگیری ماشین

منظم‌سازی L1 (لاسو): با کاهش پارامترها و ضرایب مدل، نقش مهمی در فرایند انتخاب ویژگی دارد.

منظم‌سازی L2 (ستیغی): در این تکنیک به‌طور مساوی از ضرایب مدل کاسته می‌شود و در جلوگیری از هم‌خطی چندگانه و حفظ پایداری مدل موثر است.

شبکه اِلاستیک: زمانی از شبکه اِلاستیک استفاده می‌شود که همبستگی میان ویژگی‌ها زیاد باشد یا بخواهیم از طریق کاهش پارامترها، انتخاب ویژگی متوازنی انجام دهیم.

حذف تصادفی: انتخاب تصادفی زیرمجموعه‌ای از ویژگی‌های دیتاست، به شکل‌گیری شبکه‌ای مقاوم در برابر بیش‌برازش منجر می‌شود.

توقف زودهنگام :(Early Stopping) از طولانی شدن فرایند آموزش و در نتیجه بیش‌برازش جلوگیری می‌کند. روشی ساده و اغلب کارآمد برای منظم‌سازی.

نرمال‌سازی دسته‌ای :(Batch Normalization) با نرمال‌سازی نمونه‌های داده، دیگر نیازی به پیاده‌سازی انواع دیگر regularizationو حذف تصادفی نیست.

محدودیت وزنی: این‌گونه مطمئن می‌شویم که مقادیر وزنی از یک حد مشخص فراتر نرفته و همراه با بهبود تعمیم‌پذیری مدل، احتمال بیش‌برازش نیز به حداقل برسد.

«داده افزایی» (Data Augmentation): شاید از نظر ریاضیاتی چندان شباهتی با انواع دیگر منظم‌سازی نداشته باشد اما با افزایش مصنوعی اندازه دیتاست، باعث تعمیم‌پذیری بهتر مدل می‌شود.

https://blog.faradars.org/regularization