پیش پردازش:

در ابتدا فایل دیتاست در گوگل درایو آپلود می‌شود تا از هر بار اپلود آن جلوگیری شود سپس با استفاده از کد زیر فایل به صورت دیتافریم پانداس خوانده می‌شود:

```
from google.colab import drive
drive.mount('/content/drive')

import zipfile
import pandas as pd

# Specify the path to the zip file on Google Drive
zip_file_path = '/content/drive/MyDrive/archive.zip'

# Specify the destination folder to extract the contents
extracted_folder_path = '/content/'

# Extract the contents of the zip file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_folder_path)

# Read the dataset.csv file using pandas
csv_file_path = extracted_folder_path + 'dataset.csv'
df = pd.read_csv(csv_file_path)

# Display the first few rows of the DataFrame
df.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	30669	Male	3.0	0	0	No	children	Rural	95.12	18.0	NaN	0
1	30468	Male	58.0	1	0	Yes	Private	Urban	87.96	39.2	never smoked	0
2	16523	Female	8.0	0	0	No	Private	Urban	110.89	17.6	NaN	0
3	56543	Female	70.0	0	0	Yes	Private	Rural	69.04	35.9	formerly smoked	0
4	46136	Male	14.0	0	0	No	Never_worked	Rural	161.28	19.1	NaN	0

بررسی مقادیر خالی:

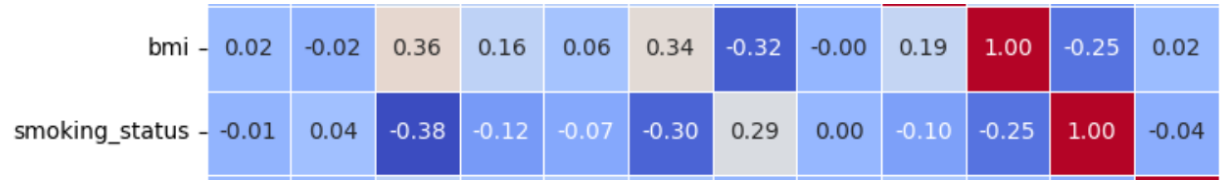
```
# Count NaN values in each column
nan_count_per_column = df.isna().sum()

# Display the result
print("NaN count per column:")
print(nan_count_per_column)
```

```
NaN count per column:
id                     0
gender                 0
```

```
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                 1462
smoking_status     13292
stroke                 0
```

همانطور که دیده می‌شود دو ستون شاخص وزنی و وضعیت سیگار دارای مقادیر خالی هستند. دو راه کار برای مقابله با مقادیر خالی در نظر گرفته می‌شود. راهکار نخست این بود که مقادیر خالی حذف شوند ولی اینجا تعداد داده‌های برچسب نادر را حدود ۵۰ درصد کاهش میداد (از حدود ۷۰۰ به ۴۰۰) بنابراین تصمیم گرفته شد توسط روش KNN Imputer مقادیر خالی توسط ۵ نزدیک ترین داده به آن‌ها پر شود. دلیل این تصمیم گیری این بود که

از ماتریس همبستگی (مطابق شکل زیر) متوجه شدیم این دو ستون همبستگی خوبی با سایر ویژگی‌ها دارند.



یک گام دیگر انجام شده در مرحله پیش پردازش تبدیل داده‌های غیر عددی به عددی است که توسط کتابخانه‌ی label encoder در سایکیت لرن انجام شد. همانطور که در شکل زیر مشخص است، مقادیر موجود در ستون‌های غیرعددی مقدار پرت ندارد و قابل کدگزاری است.

```
Distinct values for column 'gender':
['Male' 'Female' 'Other']

Distinct values for column 'ever_married':
['Yes' 'No']

Distinct values for column 'work_type':
['Private' 'Self-employed' 'Govt_job' 'children' 'Never_worked']

Distinct values for column 'Residence_type':
['Urban' 'Rural']

Distinct values for column 'smoking_status':
['never smoked' 'formerly smoked' 'smokes']
```

در نهایت تعداد داده‌ها به صورت زیر می‌باشد:

```
Stroke column value counts:
0.0    42617
1.0      783
Name: stroke, dtype: int64
```

برای متوازن کردن دیتاست از روش SMOTE[1] استفاده شده است. روش‌های مختلفی نظیر undersampling، oversampling، resampling و... وجود دارند.

SMOTE یک روش محبوب برای تعادل داده‌های نامتوازن است، به ویژه در مسائل طبقه‌بندی که تعداد نمونه‌های هر دسته به طور متفاوت است. دلایلی که SMOTE برای متعادل کردن داده‌ها مناسب است به شرح زیر است:

- در مجموعه‌داده‌های نامتوازن، کلاس اقلیت ممکن است نماینده کافی نداشته باشد که باعث ایجاد مدل‌های ضعیف با دقت پایین در پیش‌بینی کلاس اقلیت می‌شود.
- SMOTE با تولید نمونه‌های مصنوعی برای کلاس اقلیت از طریق انتقال هم‌فضای ویژگی بین نمونه‌های موجود، به تعادل رساندن توزیع کلاس‌ها کمک می‌کند.
- تولید داده مصنوعی باعث تنوع بخشیدن به مجموعه‌داده و ارائه اطلاعات بیشتر به طبقه‌بند می‌شود که ممکن است به بهبود توانایی تعمیم مدل بر روی داده‌های نامرئی کمک کند.
- SMOTE با ایجاد یک مجموعه آموزش متعادل‌تر، ممکن است به بهبود عملکرد تعمیمی مدل بر روی داده‌های ناشناخته منجر شود.

هرچند که SMOTE می‌تواند مفید باشد، ممکن است همیشه بهترین گزینه نباشد و موفقیت آن به ویژگی‌های خاص مجموعه‌داده و ماهیت مسئله بستگی داشته باشد. همیشه بهتر است با تکنیک‌های مختلف آزمایش کرده و تأثیر آنها را بررسی کرد. بنابراین ما اثر undersmapling یعنی کاهش نمونه ها را نیز بررسی میکنیم.

بنابراین نخست دیتاست را به دو دسته آموزش و تست تقسیم بندی می‌کنیم. نسبت تقسیم ۰.۲ است به این معنی که ۸۰ درصد دیتاست اصلی به عنوان آموزشی و ۲۰ درصد به عنوان تست تنظیم می‌شود. در آماده سازی دیتاست به روش SMOTE ۸۰ درصد داده‌ی تست را افزایش داده تا تعداد داده‌های کلاس ۱ از مقدار ۶۲۲ به ۳۴۰۹۸ برسد. اما در روش کاهش نمونه تعداد نمونه‌های برچسب ۰ را از ۳۴۰۹۸ به ۶۲۲ (به صورت تصادفی) کاهش می‌دهیم.

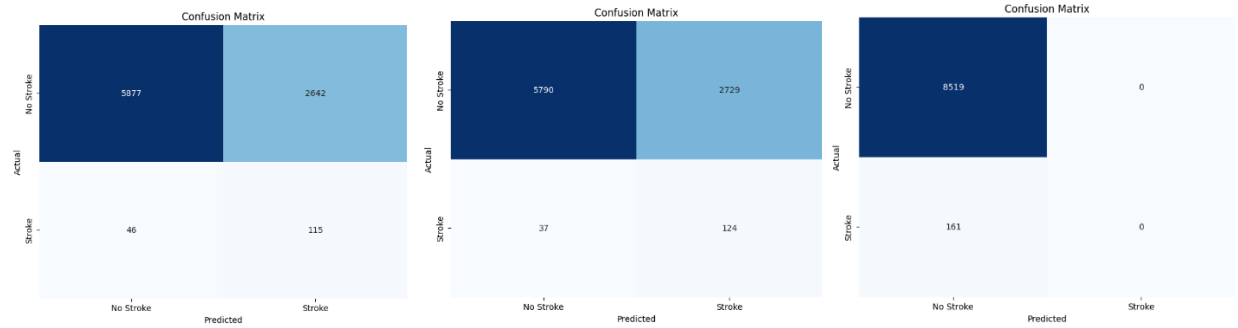
آموزش مدل

برای آموزش مدل، نخست مدل لاجستیک رگرشن انتخاب می‌شود. این مدل یک مدل ساده و رایج با عملکرد خوب در مسائل کلاس بندی است به همین دلیل به عنوان بیس لاین انتخاب شد. علاوه بر این دو مدل دیگر یعنی مدل svm که یک مدل قدرتمند شناخته شده در کلاس بندی است و مدل جنگل تصادفی که یک مدل قدرتمند دیگر بخصوص در ارتباط با داده‌ی نامتوازن است استفاده می‌کنیم.

نتیجه پیاده سازی مدل‌های کلاس بندی به صورت زیر می‌باشد:

[1] Synthetic Minority Over-sampling Technique

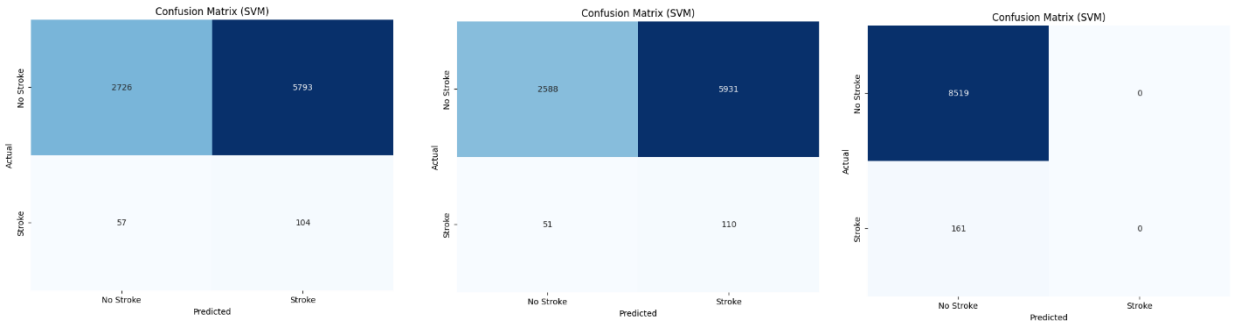
	بازیابی شناسایی سکته (برچسب ۱)		
	بدون متوازن سازی	متوازن سازی با SMOTE	متوازن سازی با کاهش نمونه
لاجستیک رگرشن	۰	۰.۷۱	۰.۷۷
ماشین بردار پشتیبان	۰	۰.۶۸	۰.۶۵
جنگل تصادفی	۰	۰.۰۱	۰.۷۸



شکل ۱: نتیجه لاجستیک رگرشن از راست به چپ: نامتوازن، SMOTE، کاهش نمونه

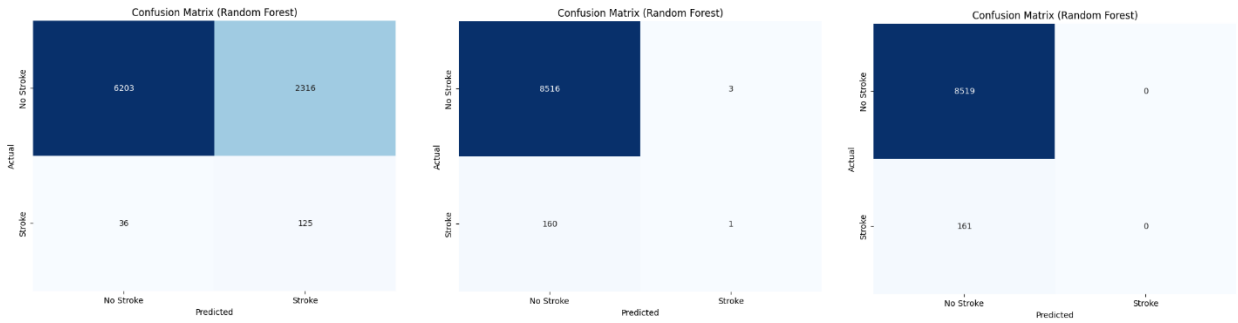
با توجه به حیاتی بودن پیش بینی سکته، شناسایی سکته بسیار مهم تر از شناسایی عدم سکته است بنابراین مهمترین معیار ارزیابی میزان بازیابی نمونه ۱ می‌باشد. همانطور که دیده می‌شود در مدل لاجستیک رگرشن نامتوازن بدترین عملکرد اتفاق افتاده و مدل به نفع نمونه فراوان تصمیم گیری کرده است. در دو حالت دیگر نتایج تا حدود مشابه هستند و با حدود ۷۰ درصد صحت مدل موفق به بازیابی حدود ۷۰ درصد در ارتباط با داده‌ها شده است.

در مدل ماشین بردار پشتیبان، عملکرد به شدت افت کرده است. علیرغم بازیابی نسبتا مناسب، صحت به حدود ۳۰ درصد تقلیل یافته است و مدل به نفع نمونه‌های ۱ به اشتباه اغلب کیس‌ها را با این برچسب نمونه گزاری کرده است که قابل قبول نیست.



شکل ۲: نتیجه مدل ماشین بردار پشتیبان از راست به چپ: نامتوازن، SMOTE، کاهش نمونه

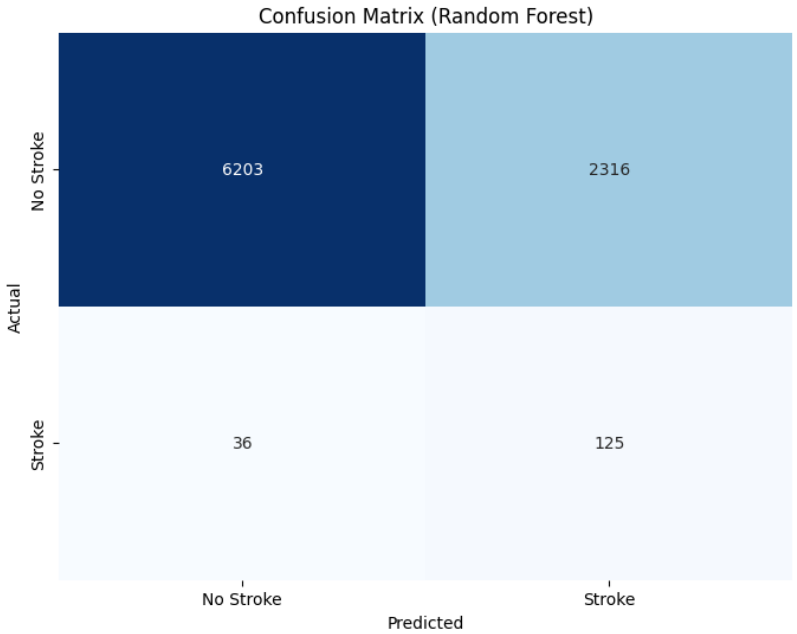
جنگل تصادفی نیز در برخورد با دیتاست‌های SMOTE و نامتوازن به نفع برچسب فراون عمل کرده و تمامی برچسب‌ها را ۰ پیش‌بینی می‌کند و تنها در برخورد با دیتاست کاهش یافته خوب عمل کرده که بهترین عملکرد تا به اینجا می‌باشد.



شکل ۳: نتیجه مدل ماشین بردار پشتیبان از راست به چپ: نامتوازن، SMOTE کاهش نمونه

بهترین نتایج به دست آمده از دیتاست کاهش نمونه یافته با مدل جنگل تصادفی به دست آمده است که نتیجه آن به صورت زیر است:

```
Random Forest Model:
Accuracy: 0.7290

Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      0.73      0.84      8519
         1.0       0.05      0.78      0.10       161

    accuracy                           0.73      8680
   macro avg       0.52      0.75      0.47      8680
weighted avg       0.98      0.73      0.83      8680
```



شبکه عصبی:

در آموزش شبکه عصبی برای مدیریت عدم توازن در دیتاست تنها از دیتاست اصلی نامتوازن استفاده می‌شود اما از وزن گذاری برای کلاس‌های مختلف بر اساس فراوانی نمونه‌هایشان استفاده می‌شود. فرمول مورد استفاده به صورت زیر است:

```
# Calculate class weights
total_samples = len(y_train)
weight_for_class_0 = total_samples / (2 * (total_samples - sum(y_train)))
weight_for_class_1 = total_samples / (2 * sum(y_train))
class_weights_imbalanced = {0: weight_for_class_0, 1: weight_for_class_1}
```

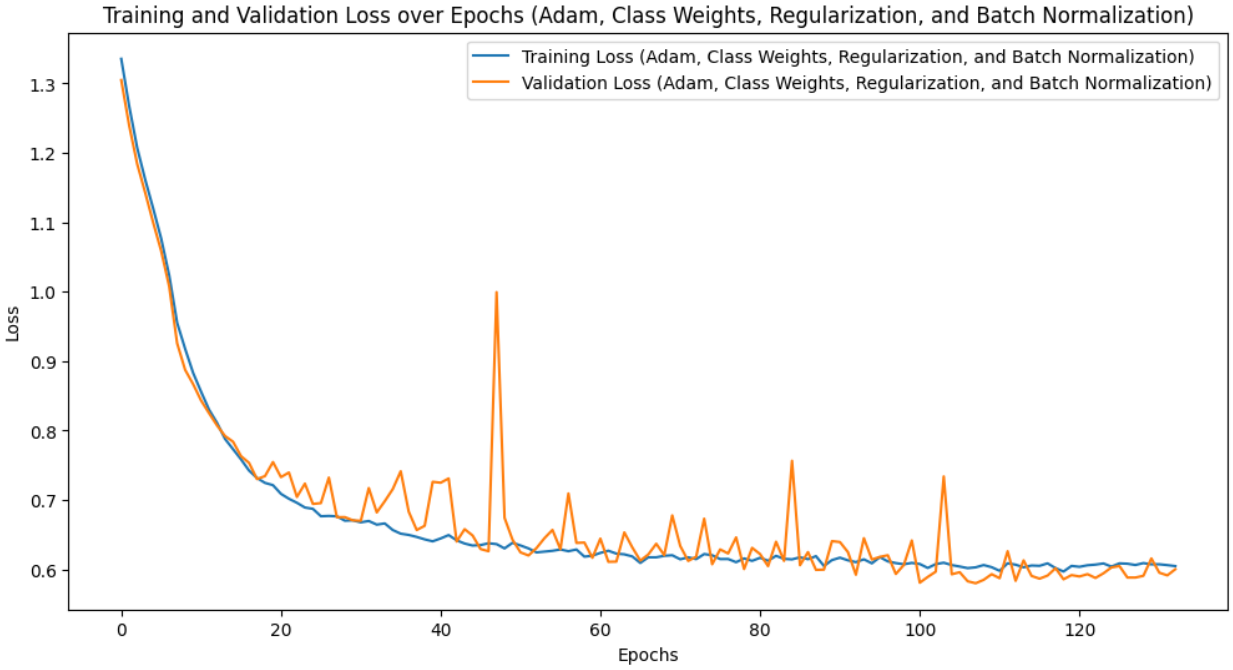
همانطور که دیده می‌شود هر کلاس به تناسب فراوانی اش وزن گرفته است. نتیجه فرمول بالا به این صورت است که در آموزش نمونه‌های کلاس ۰ وزن ۰.۵۰۹ و نمونه‌های کلاس ۱ وزن ۲۷.۹ خواهند داشت. معماری مدل طراحی شده به صورت زیر است.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (None, 64)                768

 batch_normalization_6 (Bat  (None, 64)                256
 chNormalization)

 dense_10 (Dense)            (None, 32)                2080

 batch_normalization_7 (Bat  (None, 32)                128
 chNormalization)

 dense_11 (Dense)            (None, 1)                 33

=================================================================
Total params: 3265 (12.75 KB)
Trainable params: 3073 (12.00 KB)
Non-trainable params: 192 (768.00 Byte)
_________________________________________________________________
```

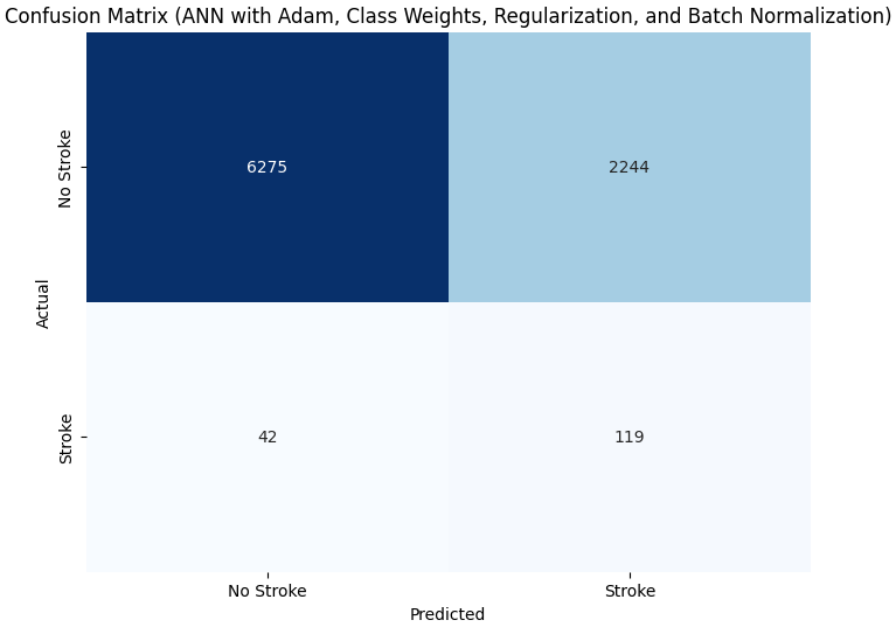
انتخاب این معماری به این دلیل بود که برای وظیفه کلاس بندی به یک شبکه متراکم ساده نیاز است که از ترکیب چند لایه تشکیل شده باشد. این شبکه دارای ۳ لایه اصلی است. در لایه اول ۶۴ نورون داده ورودی را دریافت می‌کنند، دلیل انتخاب این عدد این است که تعداد ویژگی ورودی به قدری نیست که عدد بزرگتری برای لایه ورودی لازم باشد و به نظر می‌رسد شبکه بزرگتری منجر به اورفیت شدن داده می‌شد. در لایه بعد ابعاد کاهش یافته و ۳۲ نورون خروجی لایه قبل را دریافت می‌کنند. این لایه واسط میانی برای کلاسیفیکیشن است. خروجی این لایه به لایه نهایی رفته که یک نورون با فعال ساز سیگموید می‌باشد. مقادیر خروجی بالای ۰.۵ برابر با کلاس ۱ در نظر گرفته می‌شوند. در معماری مدل به طور تجری بچ نرمالیزیشن بین لایه‌ها و رگولاریشن هر لایه منجر به بهبود مدل شد. همچنین در طراحی تابع هزینه همانطور که گفته شد از وزن برای هر کلاس استفاده شده. در نهایت این مدل با ۳۲۶۵ پارامتر که ۳۰۷۳ مورد آن قابل آموزش است، خروجی زیر را در

دو حالت بهینه ساز adam و RMSprop تولید کرد. در ضمن برای آموزش مدل از یک برنامه برای کاهش نرخ آموزش از ۰.۰۰۱ استفاده شده است به طوری که هر ۵۰ ایپاک نرخ آموزش نصف خواهد شد. این کار به همگرایی سریعتر مدل منجر شده است.

استفاده از بهینه ساز RMSprop:

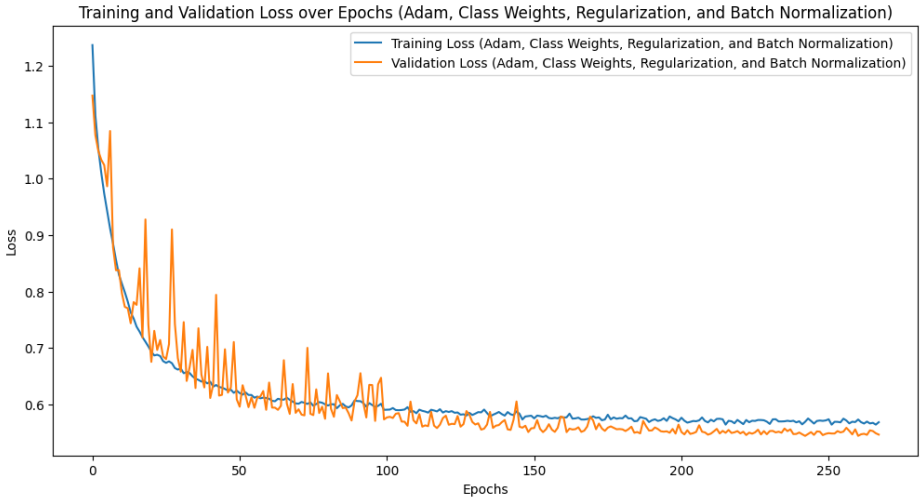


همانطور که دیده می‌شود این مدل به خوبی در نهایت همگرا شده است. مقدار خطای آموزش و خطای اعتبار سنجی هر دو کاهش یافته اند و در مقدار تقریبا ثابتی همگرا شده اند. این آموزش ۱۳۳ ایپاک طول کشیده است و مطابق با تصویر زیر ۱۱۹ مورد از موارد سکته به درستی تشخیص داده شده اند.



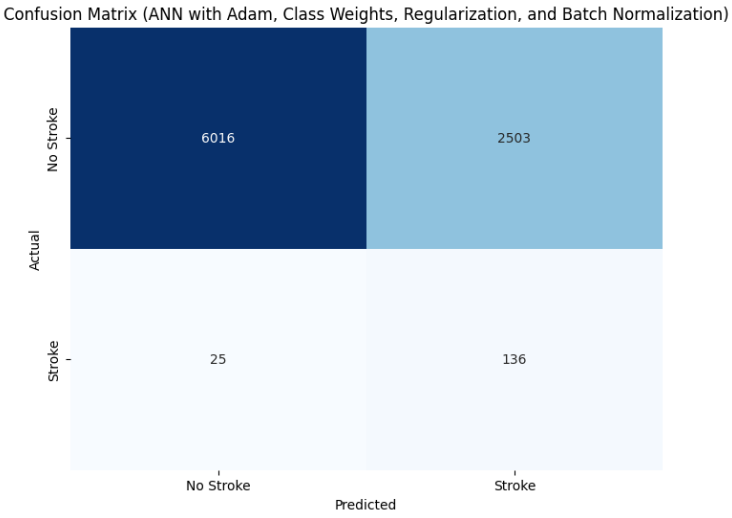
```
Artificial Neural Network (ANN) Model with RMSprop, Class Weights,
Regularization, and Batch Normalization:
Accuracy: 0.7366

Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      0.74      0.85      8519
         1.0       0.05      0.74      0.09       161

    accuracy                           0.74      8680
   macro avg       0.52      0.74      0.47      8680
weighted avg       0.98      0.74      0.83      8680
```

عملکرد این مدل به خوبی مدل جنگل تصادفی نیست و به طور جزئی ضعیف تر از آن عمل کرده است. همانطور که از ماتریس پیچیدگی مشخص است، تعداد نمونه‌های به درستی تشخیص داده شده ۶ مورد کمتر است. در ادامه مدل دیگری که بر اساس بهینه ساز adam کار می‌کند آموزش داده شد که نتایج آن به صورت زیر است:



همانطور که دیده می‌شود هر دو نمودار خطای آموزش و خطای ارزیابی در نهایت به خوبی همگرا شده اند. آموزش این مدل ۲۵۰ ایپاک طول کشیده است. در نهایت نتایج آن به صورت زیر است:

```
Artificial Neural Network (ANN) Model with Adam, Class Weights,
Regularization, and Batch Normalization:
Accuracy: 0.7088

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      0.71      0.83      8519
         1.0       0.05      0.84      0.10       161

    accuracy                           0.71      8680
   macro avg       0.52      0.78      0.46      8680
weighted avg       0.98      0.71      0.81      8680
```



همانطور که دیده می‌شود این مدل بهبود قابل توجی در تشخیص سکته داشته است. تعداد افزایش کیس‌های درست تشخیص داده شده نسبت به بهترین مدل قبلی حدود ۹ درصد یعنی ۱۱ مورد بیشتر بوده است و تنها ۲۵ مورد به اشتباه تشخیص داده نشده اند که کمترین میزان به دست آمده تا کنون است.