Phase 3

"Given the demographics and health-related factors such as age, gender, smoking status, hypertension, and diabetes, can we predict the likelihood of an individual experiencing a stroke in the next five years?"

Subhojeet Roy

UBIT: SUBHOJEE | COURSE: 587 | PERSON NUMBER: 50469240

Instructions on Running the Project. (readme.txt)

The Easy Way:

- Deployed as a Flask APP on GCP App Engine. Hit the URL and wait for a few seconds for the UI to show up: https://brain-stroke-risk-dot-moonlit-autumn-380818.ue.r.appspot.com/

The Home-Grown way:

- From the root of the phase3 directory, i.e., “src/Phase3”:
  - Execute “python3 app.py” on the command line terminal.
  - Ensure that you have python runtime with Flask, pandas, NumPy and scikit-learn libraries installed.
  - If you receive any errors on the command line, please refer to the requirements.txt file to do a “pip install” on the missing library to install it on your python environment.
  - Wait for a few seconds for the Flask Server to start-up.
  - Open your browser and navigate to: http://127.0.0.1:5000, if the Flask runtime uses a different port number(displays on the cmd line terminal), use that url instead. e.g., below:

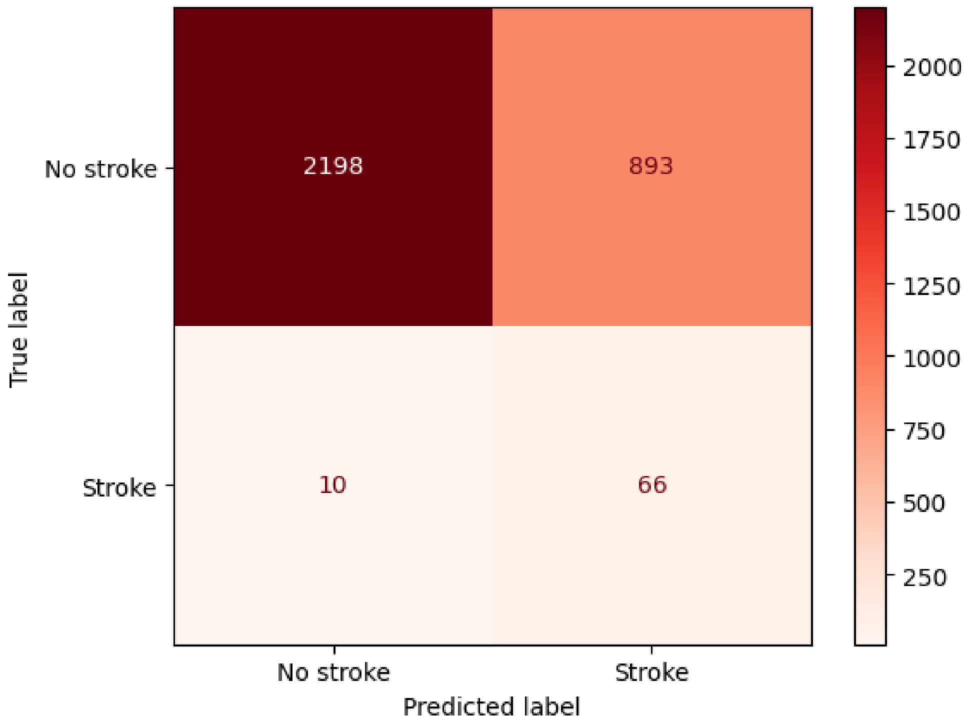
```
C:\Users\roysu\roysub\Spring23\DIC\Phase3Final\Phase_3\src\phase3>python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 131-791-169
```

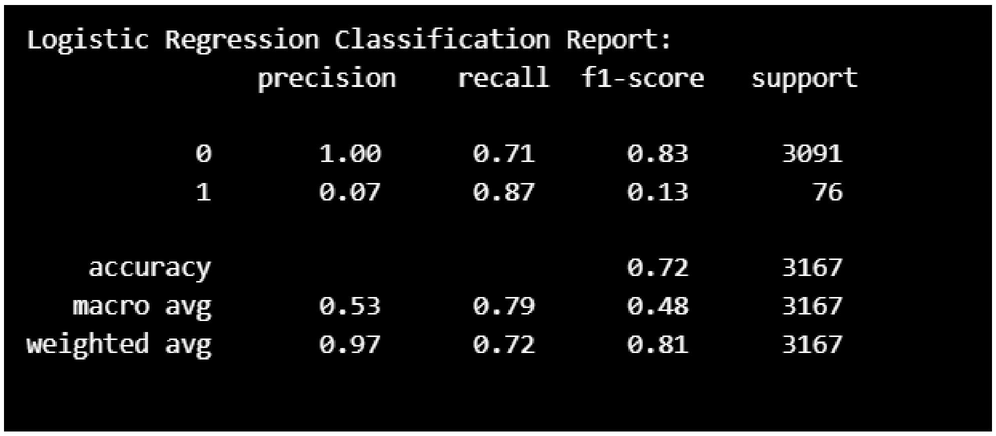
Which ML Algorithm does our Product end up using and why?

Out of all the models I evaluated Logistic Regression seems to perform the best with highest recall and F1 scores, but with low precision. I am focusing more on the F1 Score of the positive class in our use case i.e., “stroke.”

In medical screening tests, a high recall is essential to ensure that all potential patients with a particular condition are identified. A false negative (failing to identify someone with the condition) can have severe consequences, such as delayed treatment or worsening of the condition. In this case, it is often acceptable to have a higher number of false positives (identifying healthy individuals as having the condition), as they can be ruled out with further testing.

As such I ended up choosing the best performing model with the highest recall value for positive class of prediction.

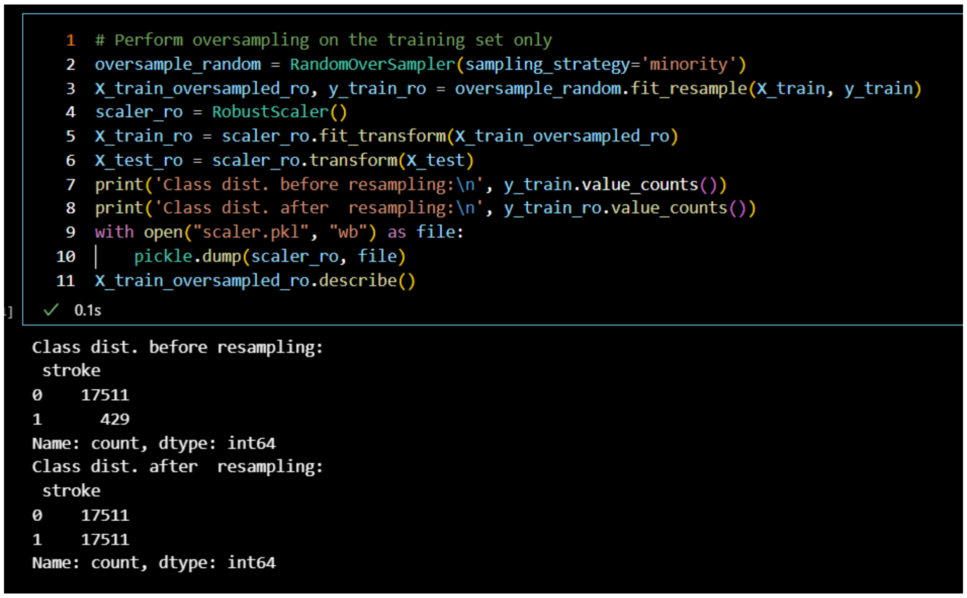




```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.71      0.83      3091
           1       0.07      0.87      0.13        76

    accuracy                           0.72      3167
   macro avg       0.53      0.79      0.48      3167
weighted avg       0.97      0.72      0.81      3167
```

How was our model used? What Parameters were tuned?

- To find good hyper parameters for tuning used GridSearchCV.
- Elastic Net regularization was used since it can help prevent overfitting and improve model generalization by adding a combination of L1 and L2 regularization terms to the loss function.
- 'C': The values used were in the grid range from 0.001 to 10 on a logarithmic scale. This range allowed us to search for the optimal regularization strength that balances between overfitting and underfitting.
- 'solver': The algorithm used for optimization. 'saga' is a suitable solver for logistic regression with Elastic Net penalty because it supports all penalty options (L1, L2, and Elastic Net). Other solvers may not be compatible with the Elastic Net penalty.
- 'max_iter': was set to 10000, helped ensure that the solver has enough iterations to converge.

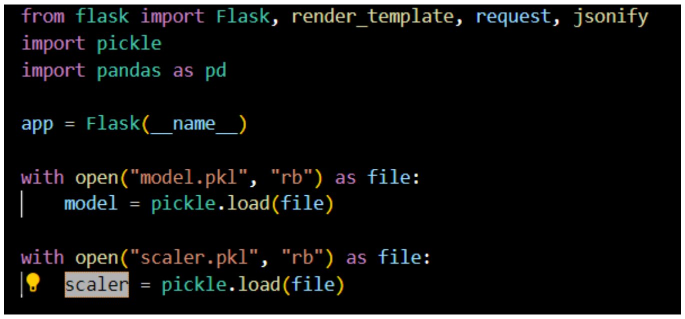


```
# Perform oversampling on the training set only
oversample_random = RandomOverSampler(sampling_strategy='minority')
X_train_oversampled_ro, y_train_ro = oversample_random.fit_resample(X_train, y_train)
scaler_ro = RobustScaler()
X_train_ro = scaler_ro.fit_transform(X_train_oversampled_ro)
X_test_ro = scaler_ro.transform(X_test)
print('Class dist. before resampling:\n', y_train.value_counts())
print('Class dist. after  resampling:\n', y_train_ro.value_counts())
with open("scaler.pkl", "wb") as file:
    pickle.dump(scaler_ro, file)
X_train_oversampled_ro.describe()
```

```
✓ 0.1s
```

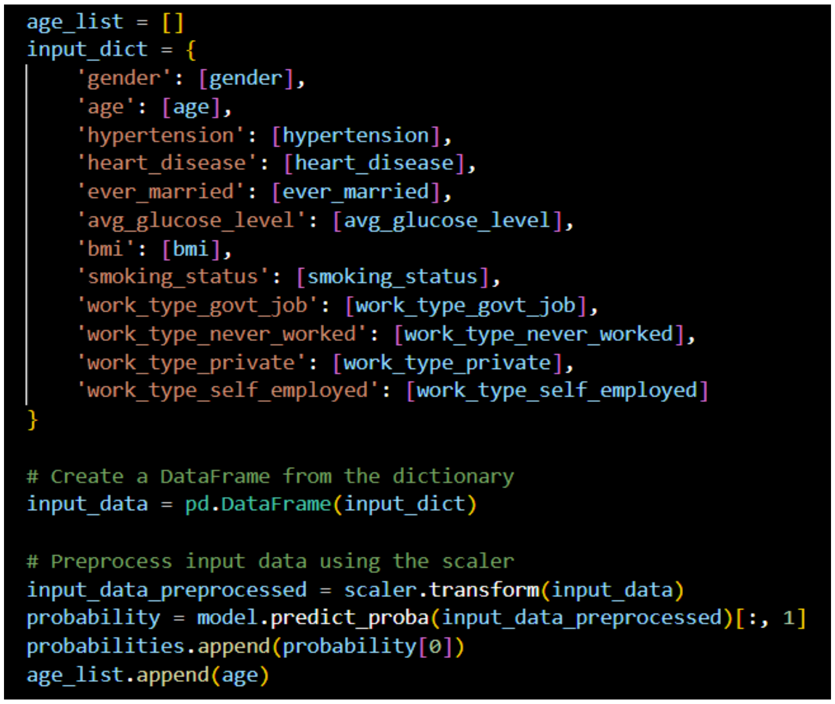
```
Class dist. before resampling:
 stroke
0    17511
1      429
Name: count, dtype: int64
Class dist. after  resampling:
 stroke
0    17511
1    17511
Name: count, dtype: int64
```

Before Training our Model, we Applied RobustScaler algorithm to transform our train and test data. This was done to ensure our model generalizes all the features on the same scale while learning the weights and when predicting on the test set. RobustScaler also helps in reducing the impact of outliers on the scaled data.
The ***Scaler and Model*** were saved into a Pickle File named “scaler.pkl” and “model.pkl”, **our product internally loads the scaler and model**:



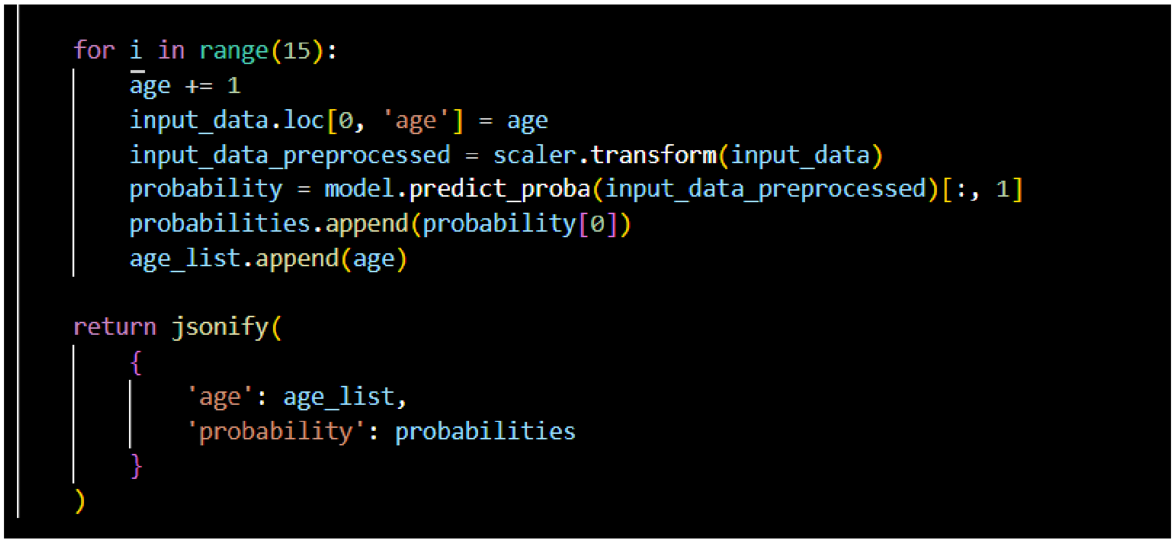
```
from flask import Flask, render_template, request, jsonify
import pickle
import pandas as pd

app = Flask(__name__)

with open("model.pkl", "rb") as file:
    model = pickle.load(file)

with open("scaler.pkl", "rb") as file:
    scaler = pickle.load(file)
```

The transformations gets applied when the end user of our product enters their own data:



```
age_list = []
input_dict = {
    'gender': [gender],
    'age': [age],
    'hypertension': [hypertension],
    'heart_disease': [heart_disease],
    'ever_married': [ever_married],
    'avg_glucose_level': [avg_glucose_level],
    'bmi': [bmi],
    'smoking_status': [smoking_status],
    'work_type_govt_job': [work_type_govt_job],
    'work_type_never_worked': [work_type_never_worked],
    'work_type_private': [work_type_private],
    'work_type_self_employed': [work_type_self_employed]
}

# Create a DataFrame from the dictionary
input_data = pd.DataFrame(input_dict)

# Preprocess input data using the scaler
input_data_preprocessed = scaler.transform(input_data)
probability = model.predict_proba(input_data_preprocessed)[:, 1]
probabilities.append(probability[0])
age_list.append(age)
```

One can notice that the **backend** increments the age and calls. **`model.predict_proba()`** for 15 iterations, this was done to give users of our product to evaluate their risk of a stroke over the next 15 years. The predicted probability values for the positive class of Stroke, is returned as an array json object for the front end to visualize and draw a plot on.

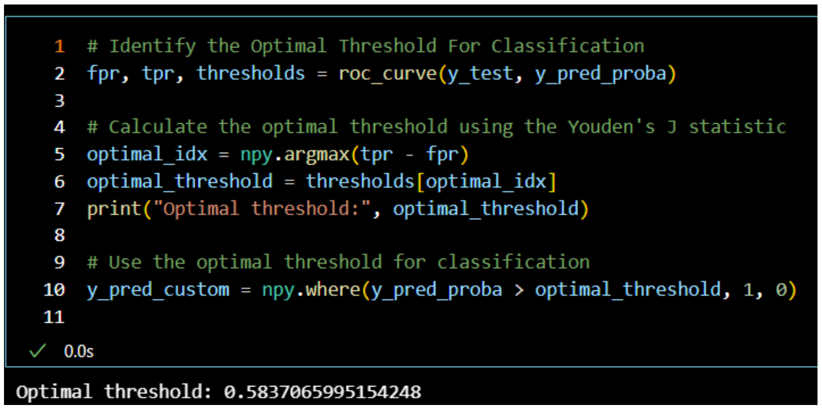


```
for i in range(15):
    age += 1
    input_data.loc[0, 'age'] = age
    input_data_preprocessed = scaler.transform(input_data)
    probability = model.predict_proba(input_data_preprocessed)[:, 1]
    probabilities.append(probability[0])
    age_list.append(age)

return jsonify(
    {
        'age': age_list,
        'probability': probabilities
    }
)
```

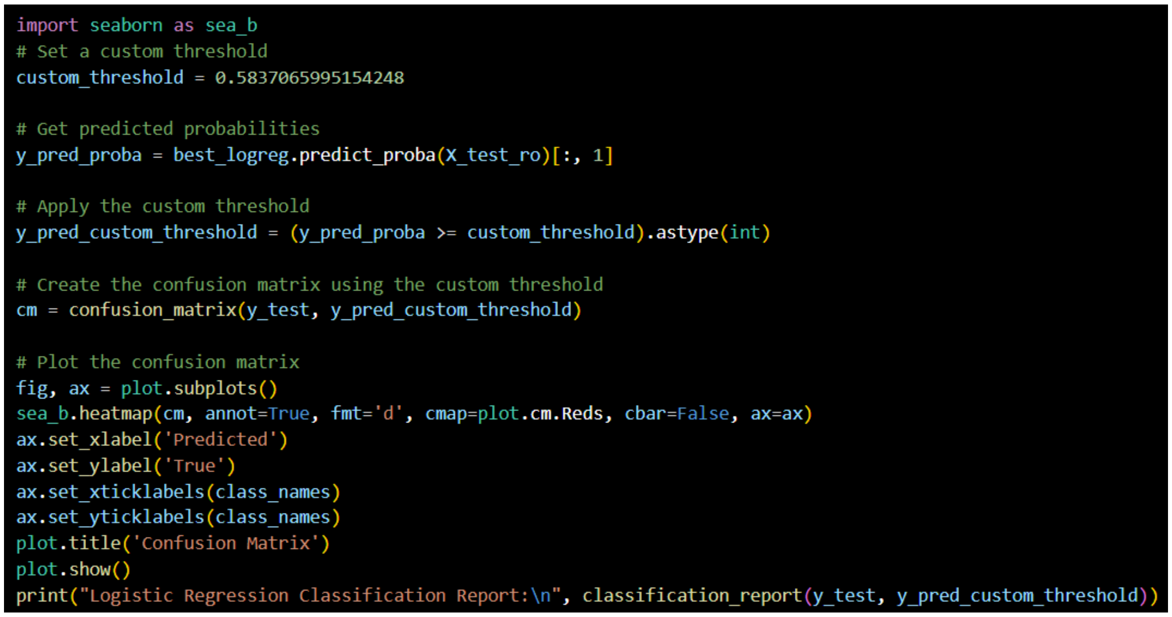
Further Tuning from phase 2:

- .predict_proba() has by default a threshold of > 0.5, for the prediction to be 1. However, the 0.5 threshold might not always be the best choice for every application, especially when the cost of misclassification differs between false negatives and false positives. In medical diagnosis, false negatives (failing to identify a patient who has a stroke) can have more severe consequences than false positives (identifying a patient as having a stroke when they don't).

- The optimal threshold was calculated using Youden's J statistic, which maximizes the difference between the true positive rate and the false positive rate.

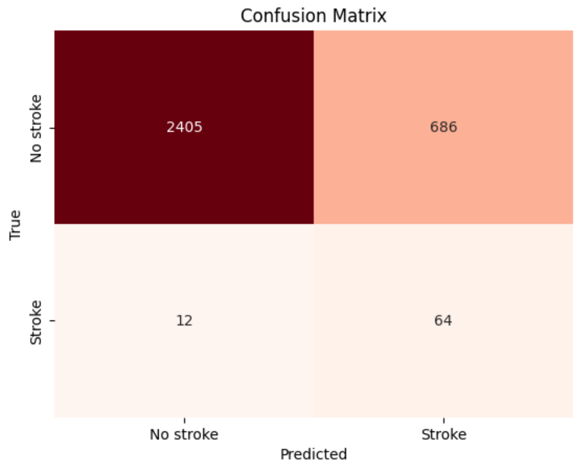


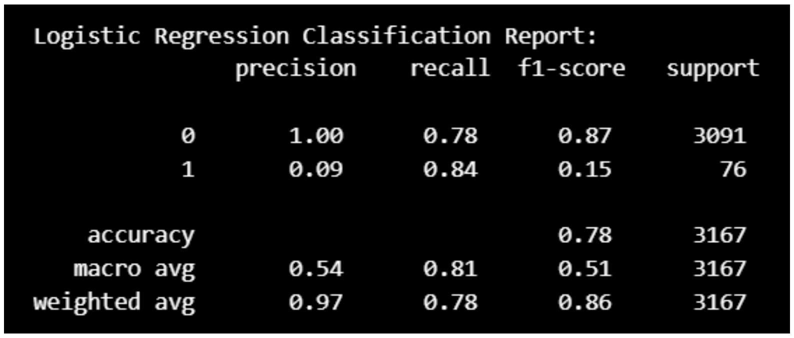
```
# Identify the Optimal Threshold For Classification
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

# Calculate the optimal threshold using the Youden's J statistic
optimal_idx = npy.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Optimal threshold:", optimal_threshold)

# Use the optimal threshold for classification
y_pred_custom = npy.where(y_pred_proba > optimal_threshold, 1, 0)

✓ 0.0s
Optimal threshold: 0.5837065995154248
```



```
import seaborn as sea_b
# Set a custom threshold
custom_threshold = 0.5837065995154248

# Get predicted probabilities
y_pred_proba = best_logreg.predict_proba(X_test_ro)[:, 1]

# Apply the custom threshold
y_pred_custom_threshold = (y_pred_proba >= custom_threshold).astype(int)

# Create the confusion matrix using the custom threshold
cm = confusion_matrix(y_test, y_pred_custom_threshold)

# Plot the confusion matrix
fig, ax = plot.subplots()
sea_b.heatmap(cm, annot=True, fmt='d', cmap=plot.cm.Reds, cbar=False, ax=ax)
ax.set_xlabel('Predicted')
ax.set_ylabel('True')
ax.set_xticklabels(class_names)
ax.set_yticklabels(class_names)
plot.title('Confusion Matrix')
plot.show()
print("Logistic Regression Classification Report:\n", classification_report(y_test, y_pred_custom_threshold))
```

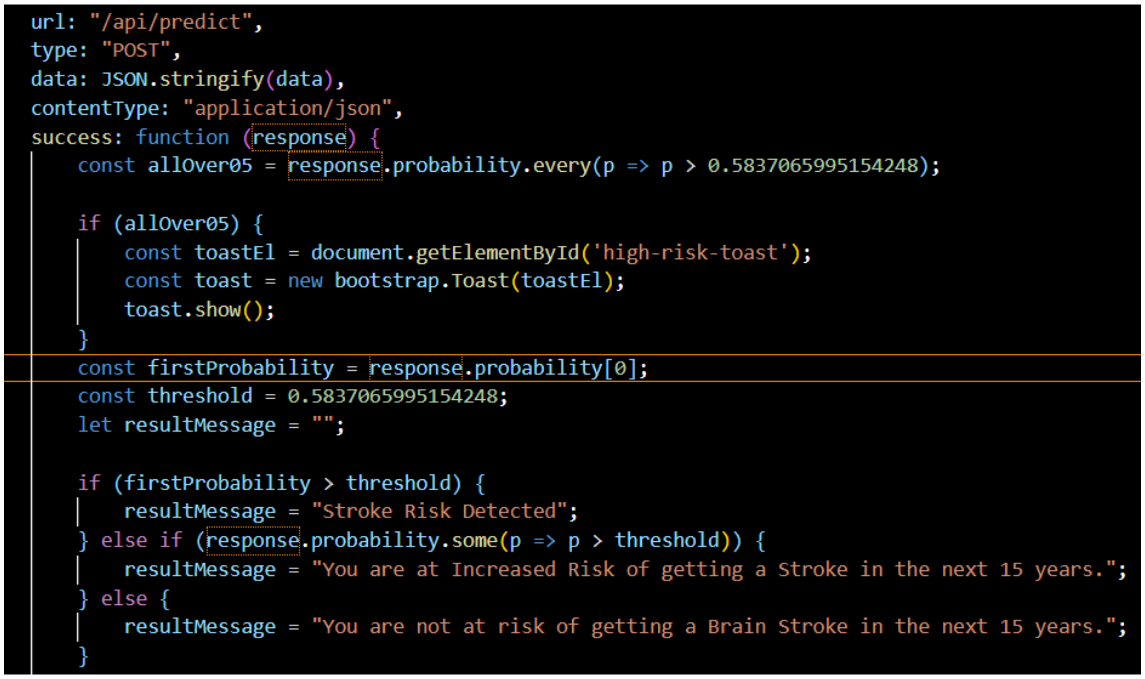
<u>Results after custom probability threshold:</u> **6%** improve in accuracy, **2%** improve in F1 score for positive class.





```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.78      0.87      3091
           1       0.09      0.84      0.15        76

    accuracy                           0.78      3167
   macro avg       0.54      0.81      0.51      3167
weighted avg       0.97      0.78      0.86      3167
```

<u>How was the tuning applied to the final Product?</u>



```
url: "/api/predict",
type: "POST",
data: JSON.stringify(data),
contentType: "application/json",
success: function (response) {
    const allOver05 = response.probability.every(p => p > 0.5837065995154248);

    if (allOver05) {
        const toastEl = document.getElementById('high-risk-toast');
        const toast = new bootstrap.Toast(toastEl);
        toast.show();
    }
    const firstProbability = response.probability[0];
    const threshold = 0.5837065995154248;
    let resultMessage = "";

    if (firstProbability > threshold) {
        resultMessage = "Stroke Risk Detected";
    } else if (response.probability.some(p => p > threshold)) {
        resultMessage = "You are at Increased Risk of getting a Stroke in the next 15 years.";
    } else {
        resultMessage = "You are not at risk of getting a Brain Stroke in the next 15 years.";
    }
```

- If the probability for positive prediction is greater than 0.5837065995154248 for all the age values, we notify the user with a High-Risk Toast, which gives them immediate feedback and resources to act by paying more attention to their health.

- If the probability of stroke increases anytime above the threshold in the next 15 years, we display the user with an appropriate message.

What can users learn from the Product?

- Users can learn from this product by gaining insights into their stroke risk based on various factors such as age, gender, average blood sugar level, history of hypertension, heart disease, work type, and smoking history. This product can help users become more aware of their potential risk for stroke and take preventive measures to reduce that risk. It also helps users understand the importance of maintaining a healthy lifestyle to minimize the chances of stroke.

- They get insights on how the future risk of getting a stroke looks like. Should they incorporate a change into their lifestyle in some ways that help mitigate the risk of getting a stroke.

- Self-Awareness at home: As human beings, we are bound by daily responsibilities that forbid us from taking care of ourselves and ignoring our own health. There's a reason why we say, "Prevention is better than cure" and this project could help identify an individual, given their current information what is the likelihood of them having a stroke.

- It can be further developed into a project where an individual can determine the risk factors that are contributing to their higher risk of having a stroke and give them feedback on the preventative measures/steps that they should take in avoiding it.

- Research: Our model could be used to generate hypotheses for further research on stroke risk factors and indicators i.e., the features of the model e.g., smoking status, blood sugar level, etc., leading to a better understanding of stroke occurrence and prediction.

- Improved patient care: By identifying individuals who are at high risk for stroke, healthcare providers can offer targeted interventions, such as lifestyle modifications, medication management, and other preventive measures, to reduce the risk of stroke occurrence and improve patient outcomes.

Ideas for extending the project or exploring related avenues:

- Personalized recommendations: Offer personalized lifestyle recommendations based on user input to help lower their risk of stroke. This could include exercise routines, dietary changes, or advice on managing stress.

- A Web Platform for any disease: By extending the project by collecting reliable data and researching on the best ML Algorithm to provide relevant results the product can be extended to a wider platform and audience who seek scrutiny for not just their risk of getting a stroke but other preventable diseases.

- Integration with healthcare providers: Collaborate with healthcare providers to enable users to share their risk assessment results, facilitating better communication and personalized care.

- Real-time monitoring: Integrate the app with wearable devices, such as smartwatches or fitness trackers, to track users' vital signs and provide real-time feedback on their health status and adjust stroke risk predictions accordingly.

- Additional risk factors: Expand the model to include more risk factors, such as family history of stroke, obesity, and other relevant medical conditions.

- multi-language support: Add support for multiple languages to make the product more accessible to users worldwide.

- Educational resources: Provide educational resources about stroke prevention and management, including articles, videos, and infographics.

- Community features: Implement a forum or chat platform where users can connect with others who share similar stroke risk profiles, enabling them to share experiences, advice, and support.

- Mobile app: Develop a mobile app version of the product for increased accessibility and convenience.

By extending the project or exploring related avenues, the product can have a more significant impact on users by addressing a wider range of factors related to stroke prevention and management.