Stroke Risk Factors & Analysis

MIS373 Final Project
Chakrabarti 11:00 - 12:30

Project Group 9

Anchit Nath

Evan Lock

Prerana Amargol

Robert Carstens

Willy Resendiz Torres

Presentation Overview

- Intro to Data Set
- Problem Statement
- Data preparation & EDA
- Regression Task & Normalization
- Odds Interpretation
- Summary Statistics
- Limitations
- Conclusion
- References & Questions

Project Intro & Goals

Dataset Details

- Data set from Kaggle with large sample size and detailed variables
- Variables include age, blood pressure, cholesterol, smoking, education, etc.
- Data sourced from research articles by AHA and CDC, and surveys by NAMCS and NHAMCS (1999-2022)
- Surveys use multi-stage probability design to select representative samples

Problem Statement

- Which features have the strongest impact on whether a patient had a stroke
- Used statistical techniques such as regression analysis and feature selection

Importance of Problem

- Heart disease is a leading cause of death and disability worldwide
- Identifying risk factors can help prevent and treat heart disease
- Predictors can improve prediction models and personalized interventions
- Improved patient outcomes and reduced healthcare costs

Exploratory Analysis

Size of Data Set

- Initially, the dataset had 4238 values and 16 features

```
[ ]  df.shape
```

```
(4238, 16)
```

- After dropping null values, that number dropped to 3656

```
[ ]  # drop null values
     df = df.dropna()
     df.shape
```

```
(3656, 16)
```

Initial Summary + Segmentation

- Initial Summary of features

```
df[['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI','heartRate','glucose']].describe(include=[np.number]).T
```

	count	mean	std	min	25%	50%	75%	max
age	3656.0	49.557440	8.561133	32.00	42.00	49.00	56.00	70.0
cigsPerDay	3656.0	9.022155	11.918869	0.00	0.00	0.00	20.00	70.0
totChol	3656.0	236.873085	44.096223	113.00	206.00	234.00	263.25	600.0
sysBP	3656.0	132.368025	22.092444	83.50	117.00	128.00	144.00	295.0
diaBP	3656.0	82.912062	11.974825	48.00	75.00	82.00	90.00	142.5
BMI	3656.0	25.784185	4.065913	15.54	23.08	25.38	28.04	56.8
heartRate	3656.0	75.730580	11.982952	44.00	68.00	75.00	82.00	143.0
glucose	3656.0	81.856127	23.910128	40.00	71.00	78.00	87.00	394.0

- No Heart Stroke

```
dfNo = df[df['target']==0]
dfNoSummary = dfNo[['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI','heartRate','glucose']].describe(include=[np.number]).T
dfNoSummary
```

	count	mean	std	min	25%	50%	75%	max
age	3099.0	48.708938	8.383279	32.00	42.00	48.00	55.00	70.00
cigsPerDay	3099.0	8.758632	11.715691	0.00	0.00	0.00	20.00	70.00
totChol	3099.0	235.169732	43.078009	113.00	205.00	232.00	261.00	453.00
sysBP	3099.0	130.280736	20.413624	83.50	116.00	127.00	141.00	243.00
diaBP	3099.0	82.148919	11.320205	52.00	74.00	81.00	88.00	142.50
BMI	3099.0	25.642975	3.965283	15.54	23.01	25.23	27.86	51.28
heartRate	3099.0	75.626331	11.953256	44.00	68.00	75.00	82.00	143.00
glucose	3099.0	80.620200	19.128713	40.00	71.00	78.00	86.00	386.00

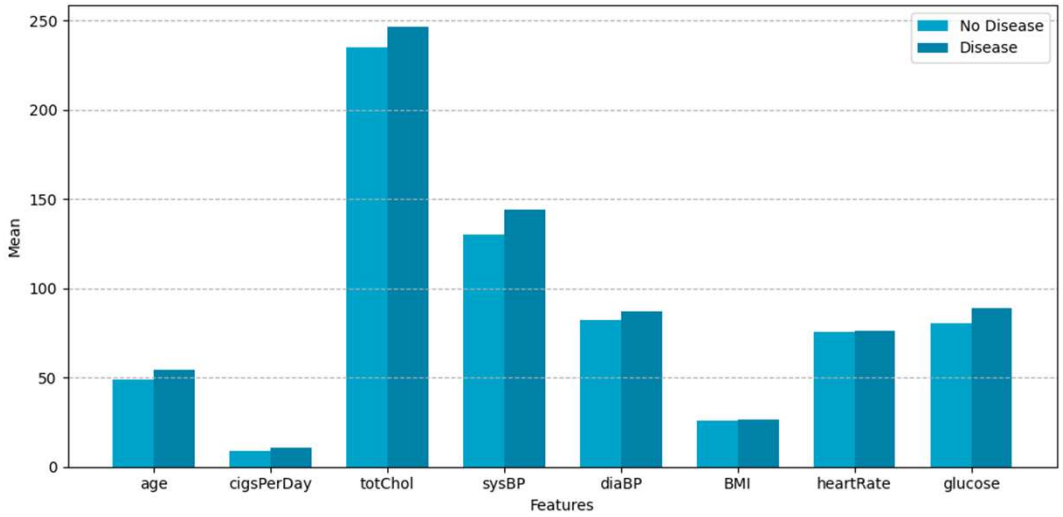
- Heart Stroke

```
dfYes = df[df['target']==1]
dfYesSummary = dfYes[['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI','heartRate','glucose']].describe(include=[np.number]).T
dfYesSummary
```

	count	mean	std	min	25%	50%	75%	max
age	557.0	54.278276	7.992338	35.00	49.00	55.00	61.00	69.0
cigsPerDay	557.0	10.488330	12.904685	0.00	0.00	1.00	20.00	60.0
totChol	557.0	246.350090	48.336365	124.00	214.00	243.00	272.00	600.0
sysBP	557.0	143.981149	26.966224	83.50	125.00	139.00	159.00	295.0
diaBP	557.0	87.157989	14.398497	48.00	78.00	85.00	95.00	140.0
BMI	557.0	26.569838	4.509435	15.96	23.63	26.11	28.94	56.8
heartRate	557.0	76.310592	12.141349	50.00	68.00	75.00	84.00	120.0
glucose	557.0	88.732496	40.785655	40.00	72.00	79.00	90.00	394.0

Visualized

```
[ ]  fig, ax = plt.subplots(figsize=(10, 5))

     bar_width = 0.35

     rects1 = ax.bar(np.arange(len(dfNoSummary)), dfNoSummary['mean'], width=bar_width, color='b', label='No Stroke')
     rects2 = ax.bar(np.arange(len(dfYesSummary)) + bar_width, dfYesSummary['mean'], width=bar_width, color='g', label='Stroke')

     ax.set_xticks(np.arange(len(dfNoSummary)) + bar_width / 2)
     ax.set_xticklabels(dfNoSummary.index, rotation=90)

     ax.set_ylabel('Mean')
     ax.set_xlabel('Features')

     ax.legend()

     plt.tight_layout()
     plt.show()
```



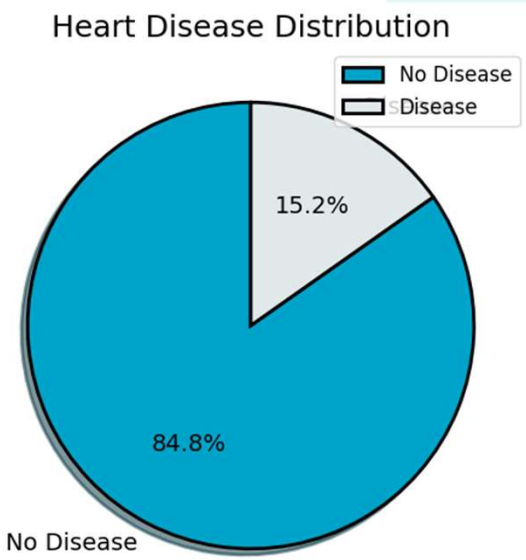
How many have heart disease?

- Total Number

```
[ ]  # number of patients in the data set with and without
     df['target'].value_counts()
```

```
0    3099
1     557
Name: target, dtype: int64
```

- Percentage



Heart Disease by Age

- Average Age

```
[ ]  # average age of people with heart disease in the data set
     avgAgeYes = df[df['target'] == 1]['age'].mean()
     print('The average age of people with heart disease in this data set is ' + str(avgAgeYes))

     # average age of people without heart disease in the data set
     avgAgeNo = df[df['target'] == 0]['age'].mean()
     print('The average age of people without heart disease in this data set is ' + str(avgAgeNo))
```

```
The average age of people with heart disease in this data set is 54.278276481149014
The average age of people without heart disease in this data set is 48.70893836721523
```

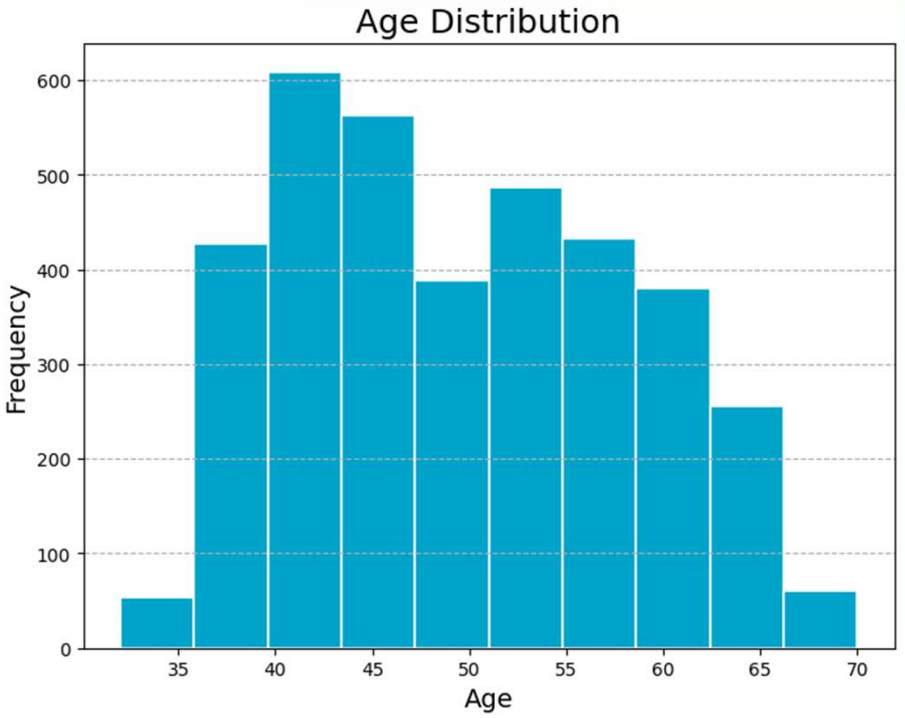
- Proportions by Age Group

What proportion of the age groups have heart disease?

```
[ ]  # Create age groups
     age_bins = [0, 40, 50, 60, 70]
     age_labels = ['<40', '40-50', '50-60', '60+']
     df['age_group'] = pd.cut(df['age'], bins=age_bins, labels=age_labels)

     # Calculate the proportion of people with heart disease for each age group
     age_group_count = df.groupby(['age_group', 'target'])['age_group'].count().unstack()
     age_group_count['Total'] = age_group_count.sum(axis=1)
     age_group_count['Disease_Proportion'] = round(age_group_count[1] / age_group_count['Total'], 2)

     # Display the results
     print(age_group_count[['Disease_Proportion']])
```

```
target     Disease_Proportion
age_group
<40                      0.05
40-50                    0.10
50-60                    0.21
60+                      0.29
```

- Visualized



Regression Analysis

Initial Regression

- Regressed on all variables
- Split data into 70/30 training/testing

Training Accuracy

```
[ ]  prediction_train = model.predict(X_train)
     print(metrics.accuracy_score(y_train, prediction_train))
```

```
0.8464243845252052
```

Testing Accuracy

```
[ ]  prediction = model.predict(X_test)
     print(metrics.accuracy_score(y_test, prediction))
```

```
0.8605287146763901
```

- Found baseline accuracy

```
Number of positive examples = 402
Number of negative examples = 2157
Number of examples where baseline is correct = 942
Baseline accuracy = 0.8587055606198724
```

Initial Regression

- Heavily weighted variables:

prevalentHyp	0.524972
C(Gender)[Male]	0.297302
C(education)[T.uneducated]	0.215255
diabetes	0.113760
BPMeds	0.101265
prevalentStroke	0.021044
sysBP	0.020993
age	0.016356
cigsPerDay	0.010267
glucose	0.005491
totChol	0.000294
heartRate	-0.026753
diaBP	-0.027309
BMI	-0.056060
C(education)[T.postgraduate]	-0.076507
currentSmoker	-0.101235
C(education)[T.primaryschool]	-0.308050
C(Gender)[Female]	-0.532817

- Hypertension
- Gender
- Education

Initial Regression

- Low weighted variables:

prevalentHyp	0.524972
C(Gender)[Male]	0.297302
C(education)[T.uneducated]	0.215255
diabetes	0.113760
BPMeds	0.101265
prevalentStroke	0.021044
sysBP	0.020993
age	0.016356
cigsPerDay	0.010267
glucose	0.005491
totChol	0.000294
heartRate	-0.026753
diaBP	-0.027309
BMI	-0.056060
C(education)[T.postgraduate]	-0.076507
currentSmoker	-0.101235
C(education)[T.primaryschool]	-0.308050
C(Gender)[Female]	-0.532817

- Age
- Cigs
- BP
- Glucose
- Cholesterol

Heavy vs Low Weighted Variables

- Hypertension
- Gender
- Education

→ *Binary*

- Age
- Cigs
- BP
- Glucose
- Cholesterol

→ *Numeric*

Normalized Regression

- Normalized numeric data
  - Scale of 0 - 1
  - Compare to each other and to binary variables

```
def normalize(column):
    return (column - column.min()) / (column.max() - column.min())
```

BMI	heartRate	glucose
26.97	80.0	77.0
28.73	95.0	76.0
25.34	75.0	70.0
28.58	65.0	103.0
23.10	85.0	85.0

BMI	heartRate	glucose
0.277024	0.363636	0.104520
0.319680	0.515152	0.101695
0.237518	0.313131	0.084746
0.316045	0.212121	0.177966
0.183228	0.414141	0.127119

Normalized Regression

- Regressed on all variables
- Same baseline and training accuracy
- *Slightly* lower testing accuracy

```
Training accuracy 0.8495506057053537
Testing accuracy 0.8587055606198724
```

Normalized Regression

- Heavily weighted variables:

age	2.204454
sysBP	2.195798
glucose	1.594276
totChol	1.368694
cigsPerDay	0.695932
diaBP	0.545916
prevalentStroke	0.466151
diabetes	0.434814
BMI	0.421605
C(Gender)[Male]	0.310555
BPMeds	0.267738
prevalentHyp	0.227181
currentSmoker	0.201213
C(education)[T.uneducated]	0.030000
C(education)[T.postgraduate]	-0.061429
C(education)[T.primaryschool]	-0.266625
heartRate	-0.291778
C(Gender)[Female]	-0.308757

- Age
- BP
- Glucose
- Cholesterol
- Cigs

Normalized Regression

- Low weighted variables:

age	2.204454
sysBP	2.195798
glucose	1.594276
totChol	1.368694
cigsPerDay	0.695932
diaBP	0.545916
prevalentStroke	0.466151
diabetes	0.434814
BMI	0.421605
C(Gender)[Male]	0.310555
BPMeds	0.267738
prevalentHyp	0.227181
currentSmoker	0.201213
C(education)[T.uneducated]	0.030000
C(education)[T.postgraduate]	-0.061429
C(education)[T.primaryschool]	-0.266625
heartRate	-0.291778
C(Gender)[Female]	-0.308757

- Education
- Smoker
- Hypertension

Regression Interpretation

Odds Interpretation

- Coefficients converted to odds and probabilities

```
# Weights converted to odds ratio
odds_ratio = np.exp(weights)
odds_ratio.sort_values(ascending=False)
```

```
age                                  9.065298
sysBP                                8.987174
glucose                              4.924764
totChol                              3.930213
cigsPerDay                           2.005577
diaBP                                1.726188
prevalentStroke                      1.593847
diabetes                             1.544676
BMI                                  1.524406
C(Gender)[Male]                      1.364182
BPMeds                               1.307005
prevalentHyp                         1.255057
currentSmoker                        1.222886
C(education)[T.uneducated]           1.030454
C(education)[T.postgraduate]         0.940420
C(education)[T.primaryschool]        0.765960
heartRate                            0.746934
C(Gender)[Female]                    0.734359
```

	count	mean	std	min	25%	50%	75%	max
age	3099.0	48.708938	8.383279	32.00	42.00	48.00	55.00	70.00
cigsPerDay	3099.0	8.758632	11.715691	0.00	0.00	0.00	20.00	70.00
totChol	3099.0	235.169732	43.078009	113.00	205.00	232.00	261.00	453.00
sysBP	3099.0	130.280736	20.413624	83.50	116.00	127.00	141.00	243.00
diaBP	3099.0	82.148919	11.320205	52.00	74.00	81.00	88.00	142.50
BMI	3099.0	25.642975	3.965283	15.54	23.01	25.23	27.86	51.28
heartRate	3099.0	75.626331	11.953256	44.00	68.00	75.00	82.00	143.00
glucose	3099.0	80.620200	19.128713	40.00	71.00	78.00	86.00	386.00

Probabilities Interpretation

- Odds converted to probabilities

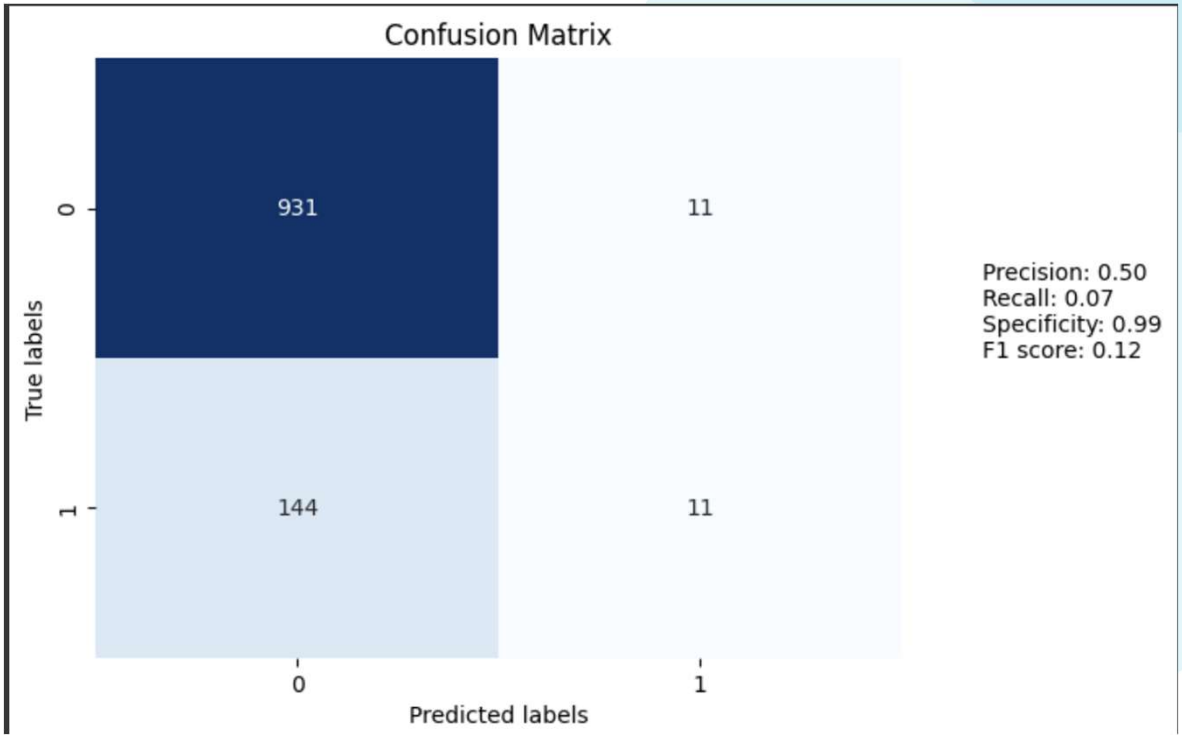
```
probability = np.exp(weights)/(1+np.exp(weights))
probability.sort_values(ascending=False)
```

```
age                                0.900649
sysBP                              0.899872
glucose                            0.831217
totChol                            0.797169
cigsPerDay                         0.667285
diaBP                              0.633187
prevalentStroke                    0.614472
diabetes                           0.607023
BMI                                0.603867
C(Gender)[Male]                    0.577021
BPMeds                             0.566538
prevalentHyp                       0.556552
currentSmoker                      0.550134
C(education)[T.uneducated]         0.507499
C(education)[T.postgraduate]       0.484648
C(education)[T.primaryschool]      0.433736
heartRate                          0.427569
C(Gender)[Female]                  0.423418
```

Summary Statistics

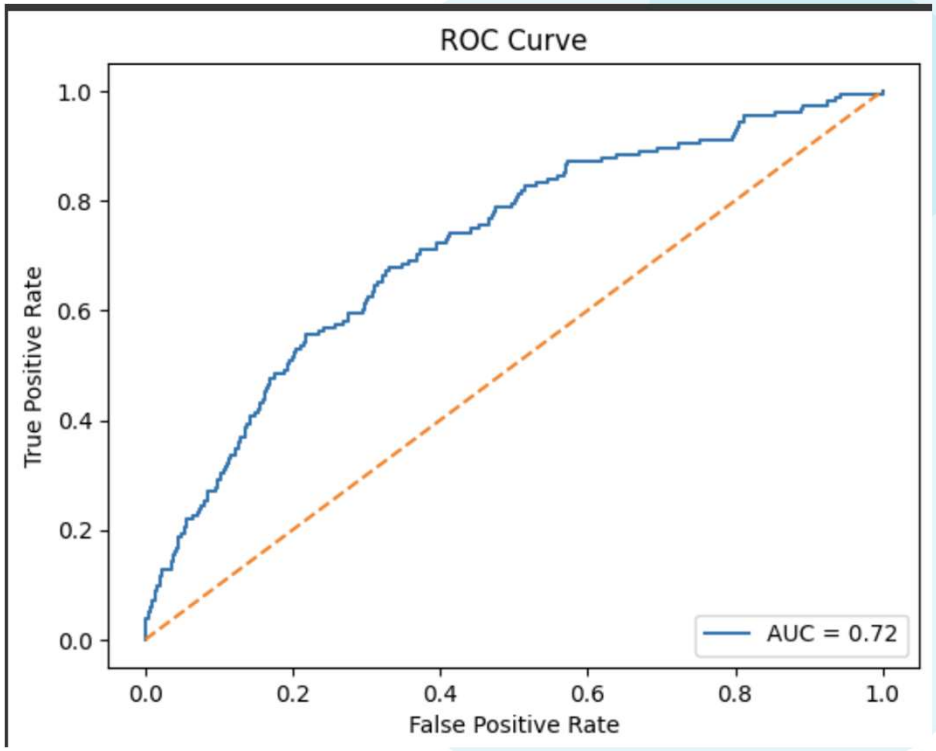
Confusion Matrix

- Shows the number of true positives, true negatives, false positives, and false negatives
- **Recall:** A measure of the proportion of actual positive cases that are correctly identified as positive
- **Precision:** A measure of the proportion of positive predictions that are correct
- **Specificity:** A measure of the proportion of actual negative cases that are correctly identified as negative



ROC Curve and AUC Value

- The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for different threshold values.
- The AUC value is a scalar value that summarizes the overall performance of a binary classifier across all possible threshold values. It ranges from 0 to 1, with a higher value indicating better performance
- The AUC value is 0.72, which indicates that the model is performing better than random guessing.



Insights & Limitations

Insights

- The most impactful variables for heart stroke are age, systolic blood pressure, glucose level, total cholesterol, and cigarettes per day
- Factors that we expected to be more impactful (bmi, heart rate) could have been correlated with other predictors
- Our model had a relatively high number of false positives and a low number of false negatives
- Overall accuracy was around 86% after normalizing the dataset and including only the impactful predictors

Limitations

- Ideally, a larger dataset would be conducive for more rigorous analysis and more generalizable findings
- Additionally, more classification and machine learning models could be utilized on this dataset to determine if our findings are reliable, since logistic regression might not be the best model
- Our analysis focused on a limited set of predictor variables, which may not capture all relevant factors that could affect the outcome variable.

THANKS!

Any questions?