Identifying Risk of Stroke Using Predictive Modelling

Presented by: Diane Yeo Su Ting

Student PI No.: H1882431

Supervisor: Dr. Wang Di

Examiner: Mr. Chua Poh Chai

Outline

Topic Introduction

Related Work

CRISP-DM

Initial Model Evaluation

Conclusion

Further Work and Limitation(s)

Introduction

Neurological attack whereby the brain is deprived of blood flow and oxygen to function.

Worrying health issue

Stroke moved up from 3rd to 2nd largest cause of death.

Problem Statement

Reduce misdiagnosis in *Emergency Rooms* (ERs) and *Primary Care Doctors* (PCDs)

Related Work

Age

Older, higher risk of stroke (Khan & Vohra, 2006)

Smoker

1.5-2.9x more at-risk (Khan & Vohra, 2006)

At-risk 10 years earlier (Frank et al., 2021)

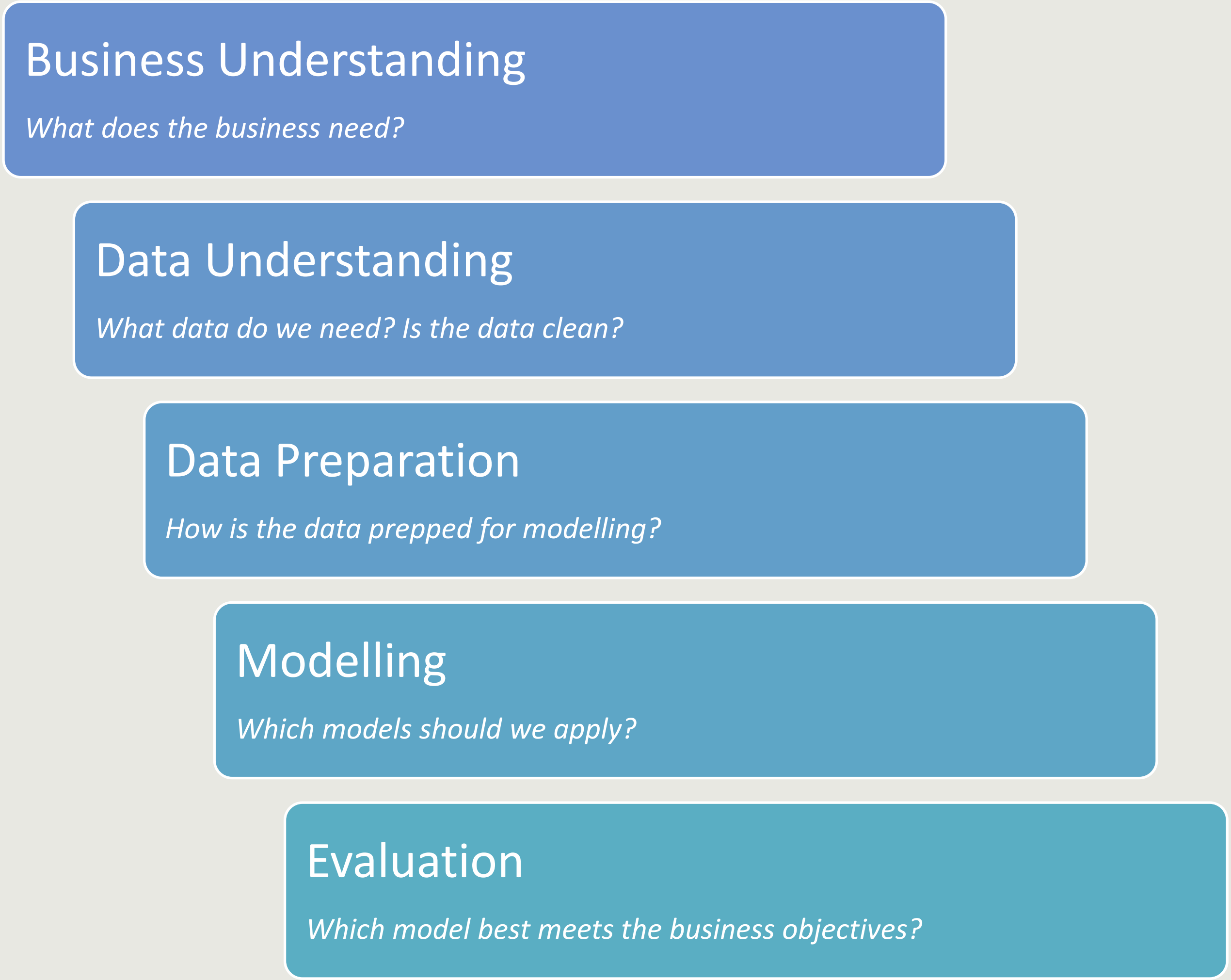
Heart disease & Diabetes

Past ischemic attack history (Khan & Vohra, 2006)

Diabetic individuals, especially females (Khan & Vohra, 2006) (Roy-O Reilly & McCullough, 2018)

CRISP-DM

Cross-Industry Standard Process for Data Mining



Business Understanding

What does the business need?

Data Understanding

What data do we need? Is the data clean?

Data Preparation

How is the data prepped for modelling?

Modelling

Which models should we apply?

Evaluation

Which model best meets the business objectives?

Business Understanding

Business Problem

Identify individuals who are at-risk of incidence of stroke, provide them with the necessary treatments ASAP.

Business Analytics Problem

- Build and identify the best performing predictive model to predict individuals at-risk of incidence of stroke
- Based on:
  1. Demographical profile
  2. Medical History
  3. Conditions

Data Understanding

Features	Description	Data Type	Values
id	Patient record id	Int	9046
gender	Patient gender	Object	Male
age	Patient age	Float	67.0
hypertension	Whether patient has hypertension	Int	0
heart_disease	Whether patient has/had heart disease(s)	Int	1
ever_married	Patient marital status	Object	Yes
work_type	Patient employment type	Object	Private
Residence_type	Patient residential area	Object	Urban
avg_glucose_level	Patient glucose reading	Float	228.69
bmi	Patient BMI reading	Float	36.60000
smoking_status	Patient smoking status	Object	formerly smoked
stroke	Occurrence of stroke	Int	1

5,110 Samples

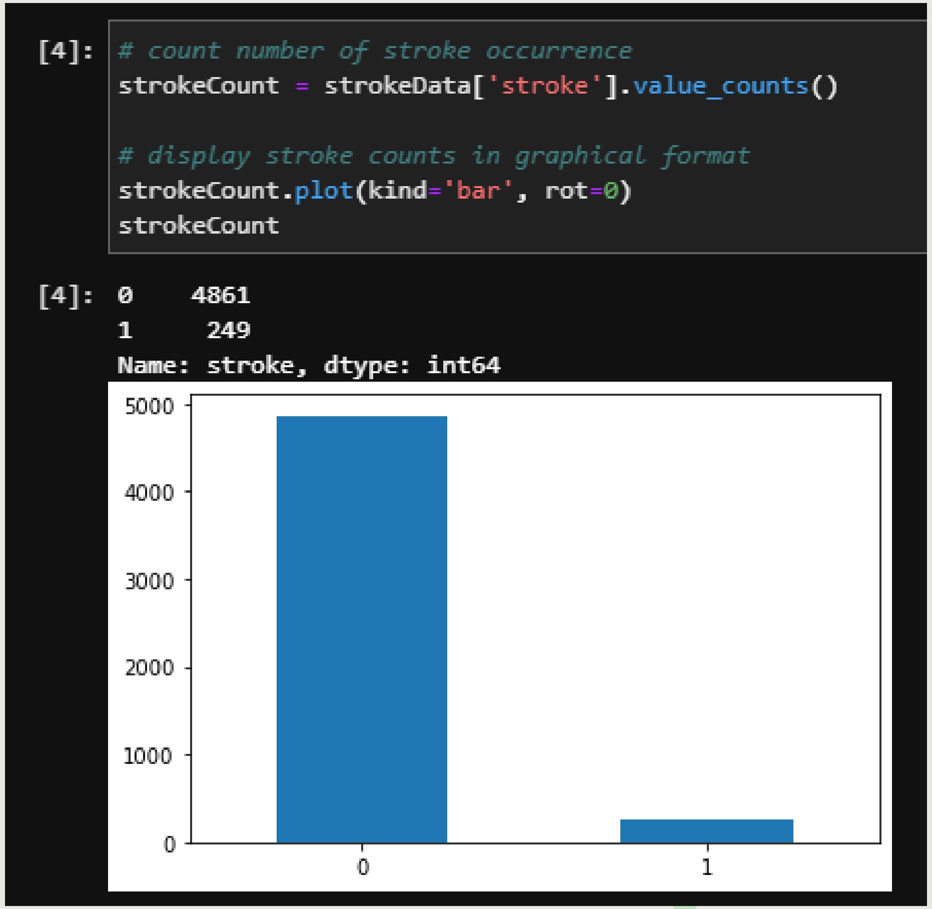
11 Input Features

Target Variable

Data Exploration

Stroke

- Rare event
- Largely imbalanced
- Skewed towards non-incidence of stroke
  - *‘0’ (no stroke)*: 4,861
  - *‘1’ (stroke)*: 249

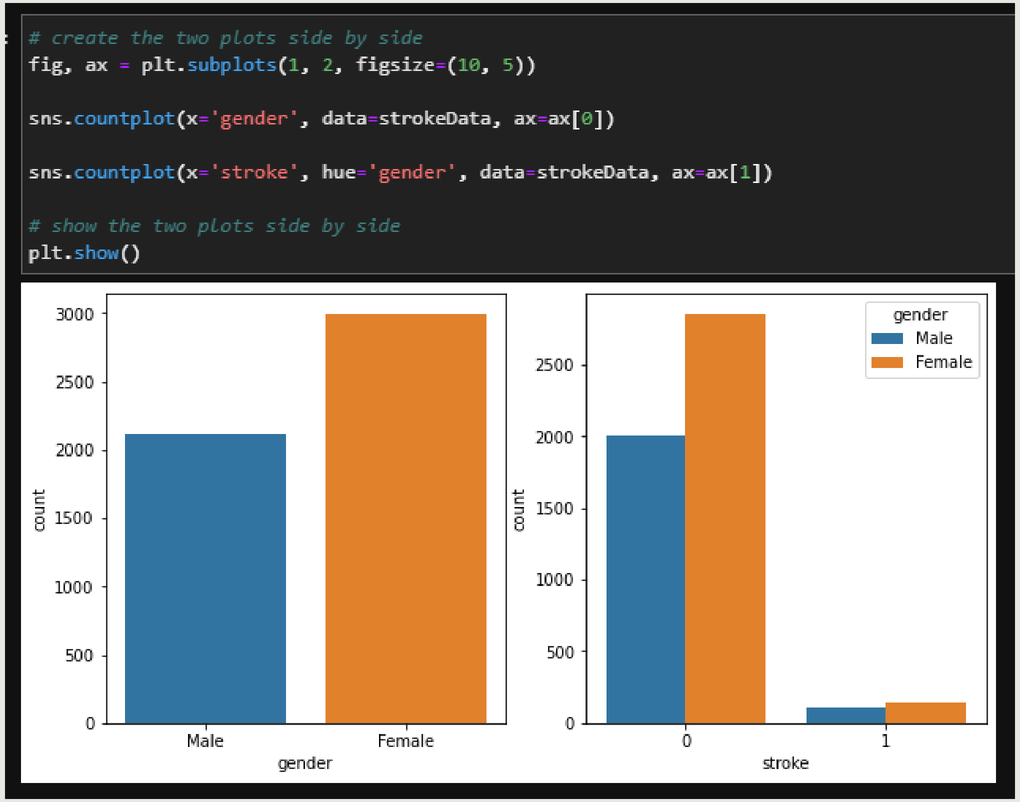


```
[4]: # count number of stroke occurrence
     strokeCount = strokeData['stroke'].value_counts()

     # display stroke counts in graphical format
     strokeCount.plot(kind='bar', rot=0)
     strokeCount
```

```
[4]: 0    4861
     1     249
     Name: stroke, dtype: int64
```

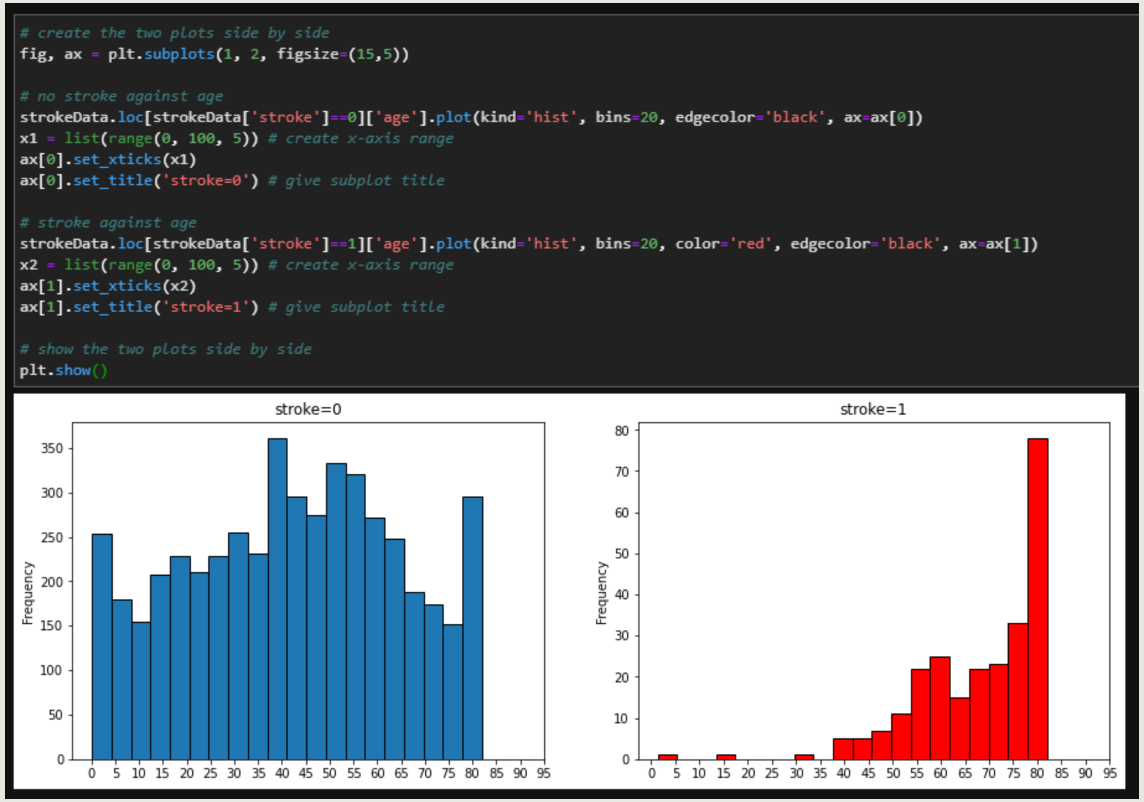
Data Exploration

Gender



```
# create the two plots side by side
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

sns.countplot(x='gender', data=strokeData, ax=ax[0])

sns.countplot(x='stroke', hue='gender', data=strokeData, ax=ax[1])

# show the two plots side by side
plt.show()
```

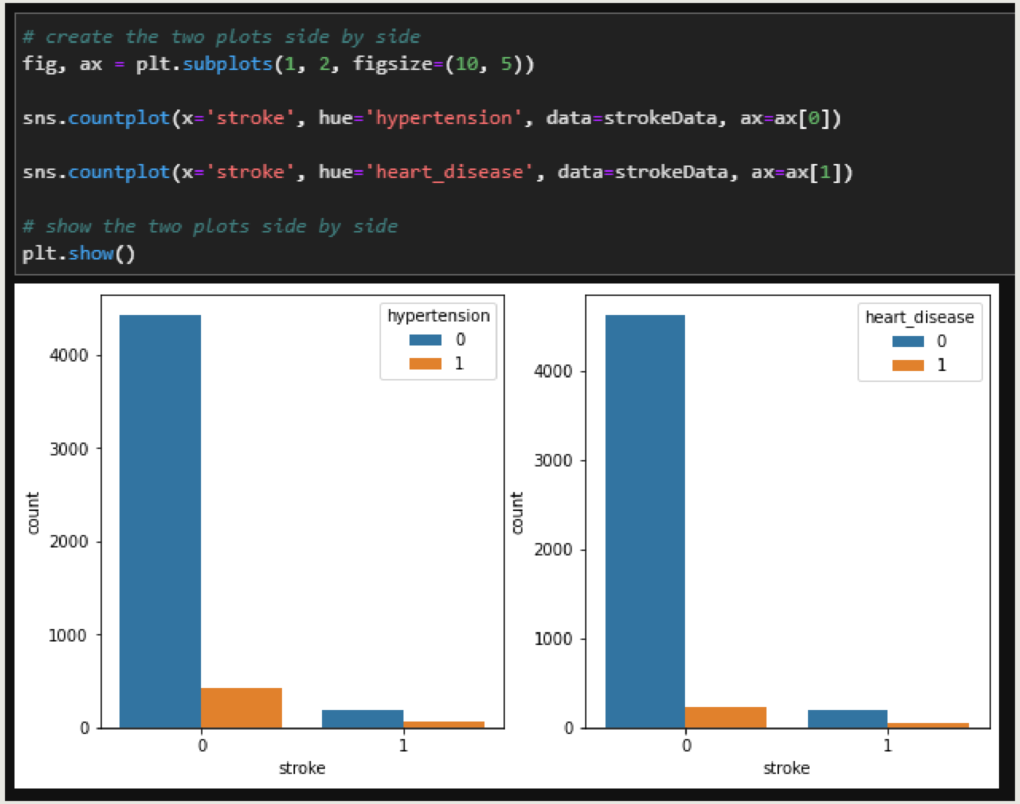
Age



```
# create the two plots side by side
fig, ax = plt.subplots(1, 2, figsize=(15,5))

# no stroke against age
strokeData.loc[strokeData['stroke']==0]['age'].plot(kind='hist', bins=20, edgecolor='black', ax=ax[0])
x1 = list(range(0, 100, 5)) # create x-axis range
ax[0].set_xticks(x1)
ax[0].set_title('stroke=0') # give subplot title

# stroke against age
strokeData.loc[strokeData['stroke']==1]['age'].plot(kind='hist', bins=20, color='red', edgecolor='black', ax=ax[1])
x2 = list(range(0, 100, 5)) # create x-axis range
ax[1].set_xticks(x2)
ax[1].set_title('stroke=1') # give subplot title

# show the two plots side by side
plt.show()
```

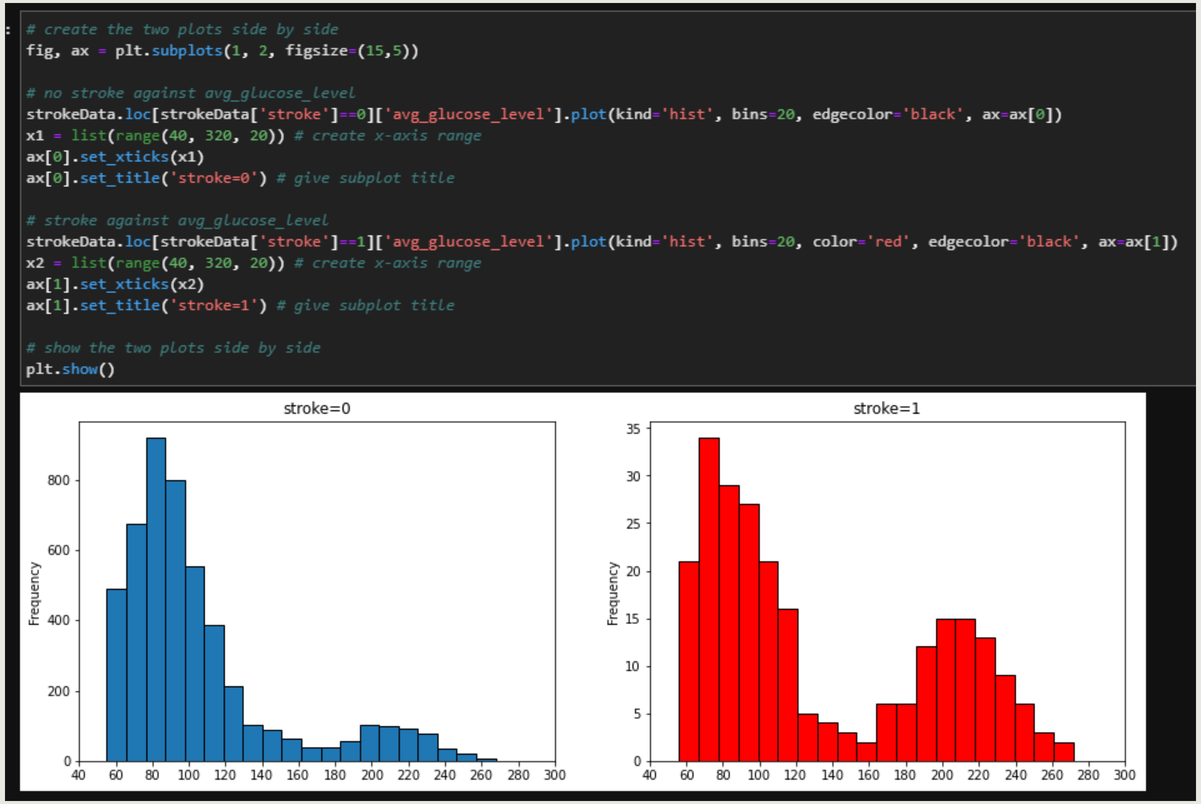
Data Exploration

Hypertension & Heart Disease



```
# create the two plots side by side
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

sns.countplot(x='stroke', hue='hypertension', data=strokeData, ax=ax[0])

sns.countplot(x='stroke', hue='heart_disease', data=strokeData, ax=ax[1])

# show the two plots side by side
plt.show()
```

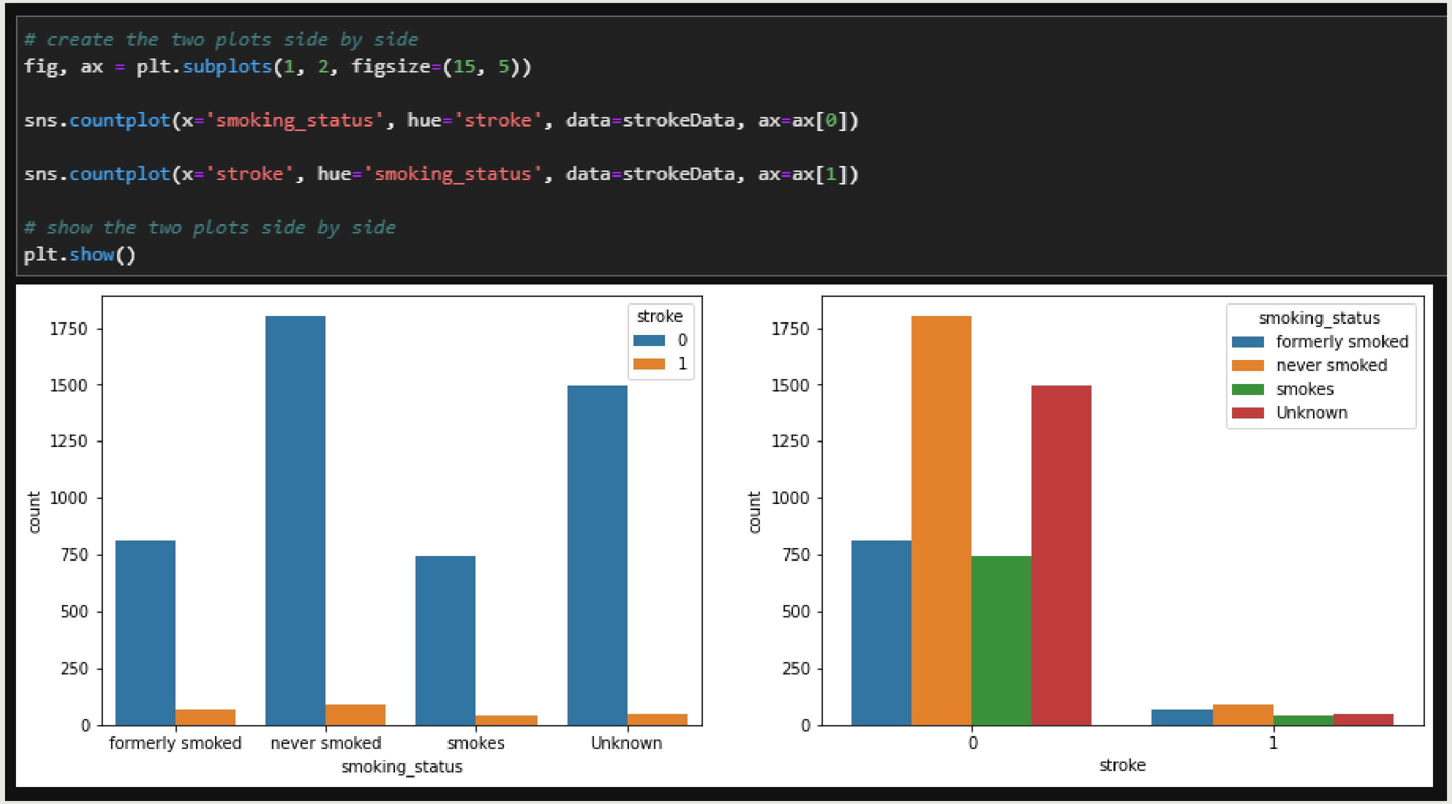
Glucose Levels



```
# create the two plots side by side
fig, ax = plt.subplots(1, 2, figsize=(15,5))

# no stroke against avg_glucose_level
strokeData.loc[strokeData['stroke']==0]['avg_glucose_level'].plot(kind='hist', bins=20, edgecolor='black', ax=ax[0])
x1 = list(range(40, 320, 20)) # create x-axis range
ax[0].set_xticks(x1)
ax[0].set_title('stroke=0') # give subplot title

# stroke against avg_glucose_level
strokeData.loc[strokeData['stroke']==1]['avg_glucose_level'].plot(kind='hist', bins=20, color='red', edgecolor='black', ax=ax[1])
x2 = list(range(40, 320, 20)) # create x-axis range
ax[1].set_xticks(x2)
ax[1].set_title('stroke=1') # give subplot title

# show the two plots side by side
plt.show()
```

Data Exploration

Smoking Status



```
# create the two plots side by side
fig, ax = plt.subplots(1, 2, figsize=(15, 5))

sns.countplot(x='smoking_status', hue='stroke', data=strokeData, ax=ax[0])

sns.countplot(x='stroke', hue='smoking_status', data=strokeData, ax=ax[1])

# show the two plots side by side
plt.show()
```

Tools Used

Python

scikit learn matplotlib seaborn

Why?

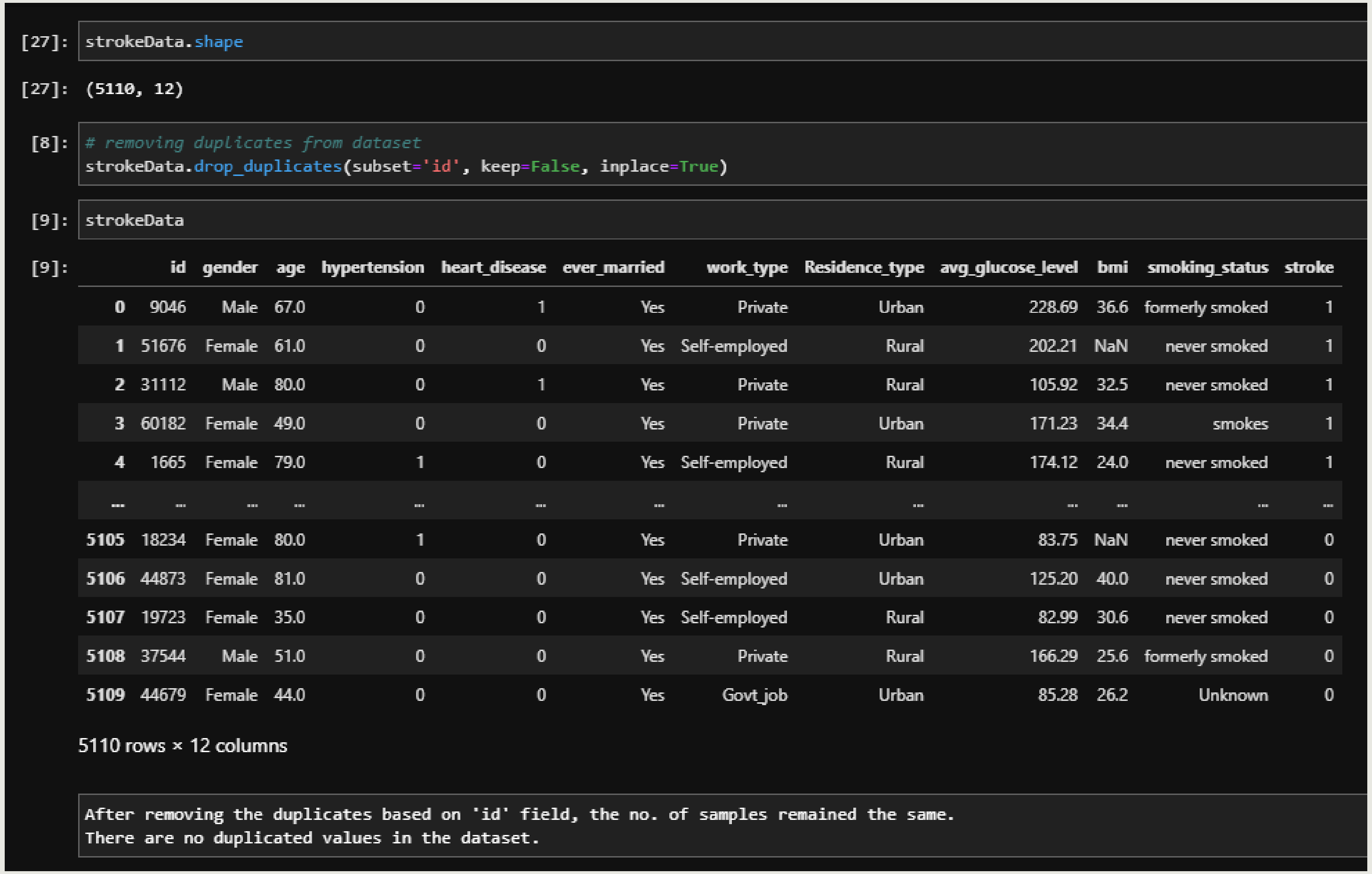
Versatile (many libraries)

Challenge myself

IBM SPSS Modeler not widely used outside

Data Preparation

1. Removing Duplicated Samples
   - All samples are unique
   - 5,110 samples



```
[27]: strokeData.shape

[27]: (5110, 12)

[8]:  # removing duplicates from dataset
      strokeData.drop_duplicates(subset='id', keep=False, inplace=True)

[9]:  strokeData

[9]:
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

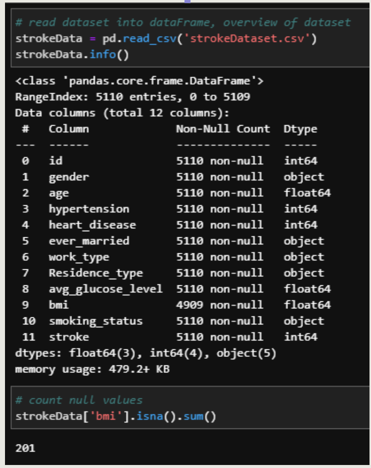
5110 rows × 12 columns

```
After removing the duplicates based on 'id' field, the no. of samples remained the same.
There are no duplicated values in the dataset.
```

Data Preparation

2. Replace Missing Values
   - BMI has 201 missing values
     - 40 had stroke

Features	Count
id	5,110
gender	5,110
age	5,110
hypertension	5,110
heart_disease	5,110
ever_married	5,110
work_type	5,110
Residence_type	5,110
avg_glucose_level	5,110
bmi	4,909
smoking_status	5,110
stroke	5,110

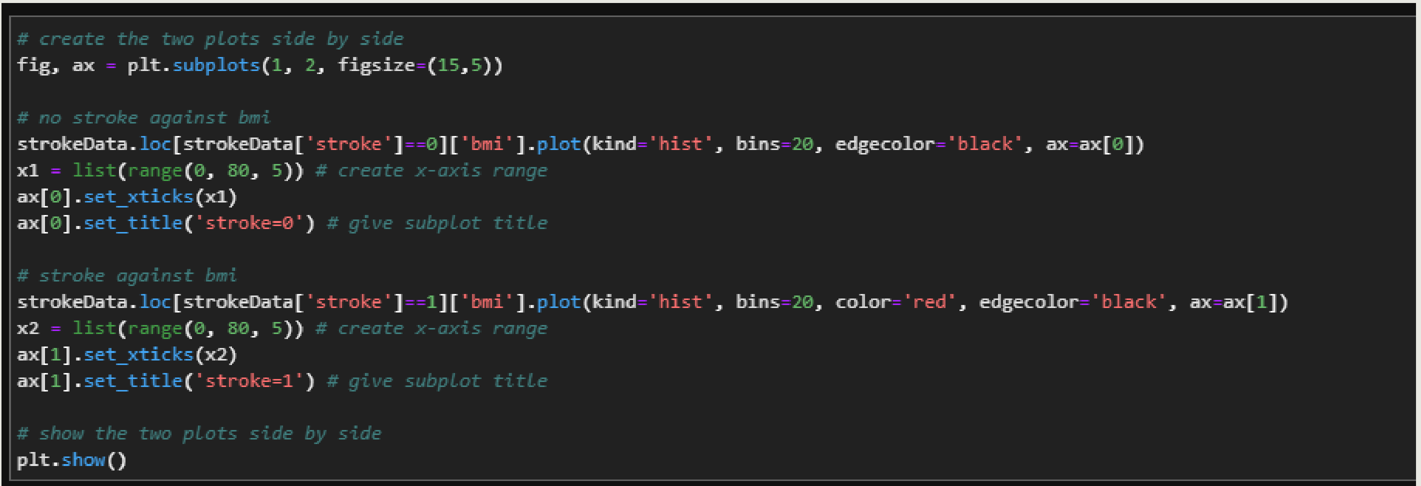


```
# read dataset into dataFrame, overview of dataset
strokeData = pd.read_csv('strokeDataset.csv')
strokeData.info()
```

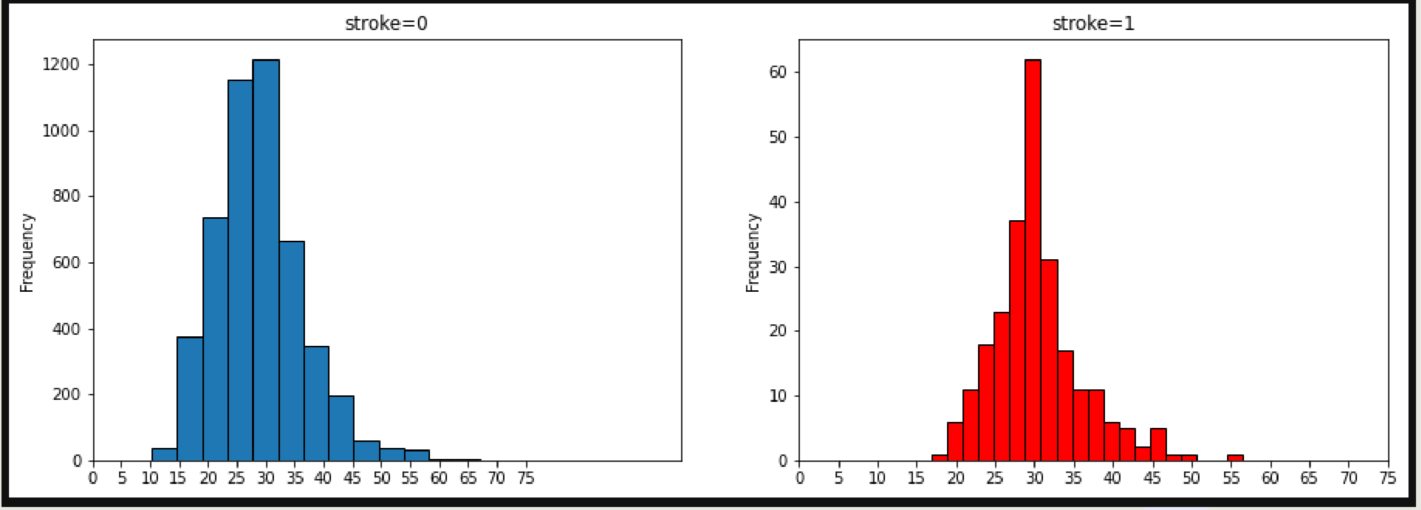
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

```
# count null values
strokeData['bmi'].isna().sum()
```

```
201
```

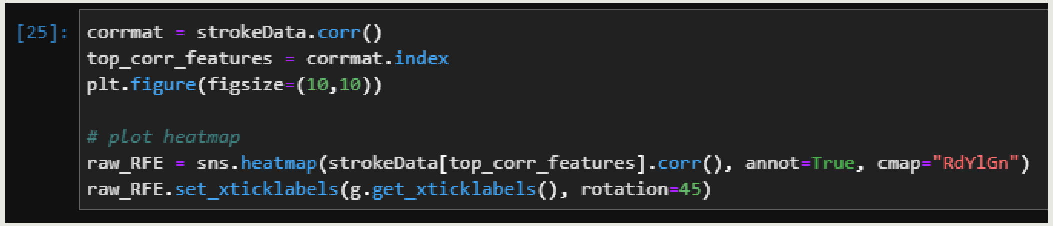


```
# create the two plots side by side
fig, ax = plt.subplots(1, 2, figsize=(15,5))

# no stroke against bmi
strokeData.loc[strokeData['stroke']==0]['bmi'].plot(kind='hist', bins=20, edgecolor='black', ax=ax[0])
x1 = list(range(0, 80, 5)) # create x-axis range
ax[0].set_xticks(x1)
ax[0].set_title('stroke=0') # give subplot title

# stroke against bmi
strokeData.loc[strokeData['stroke']==1]['bmi'].plot(kind='hist', bins=20, color='red', edgecolor='black', ax=ax[1])
x2 = list(range(0, 80, 5)) # create x-axis range
ax[1].set_xticks(x2)
ax[1].set_title('stroke=1') # give subplot title

# show the two plots side by side
plt.show()
```



Data Preparation

2. Replace Missing Values (cont'd)
   - BMI has 201 missing values
     - 40 had stroke



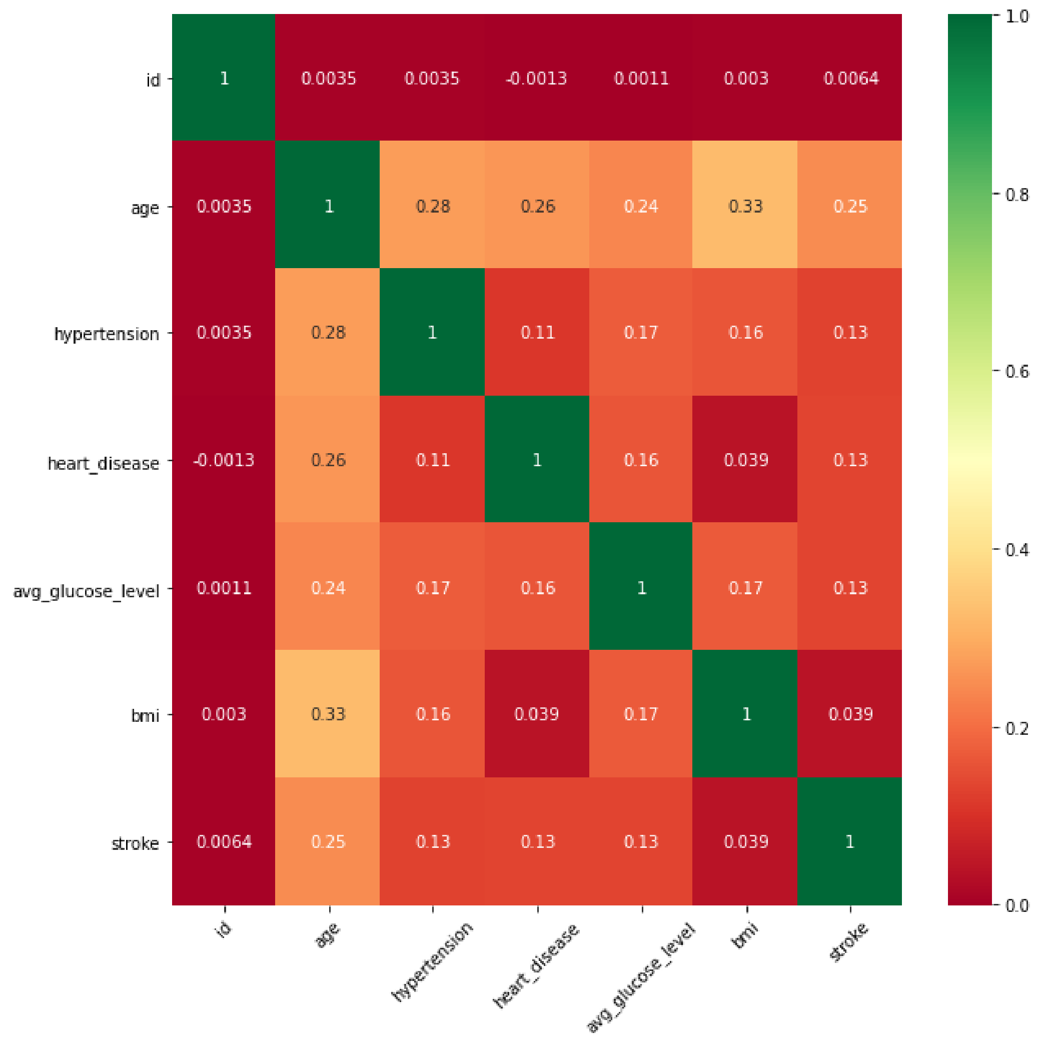
```
[25]: corrmat = strokeData.corr()
      top_corr_features = corrmat.index
      plt.figure(figsize=(10,10))

      # plot heatmap
      raw_RFE = sns.heatmap(strokeData[top_corr_features].corr(), annot=True, cmap="RdYlGn")
      raw_RFE.set_xticklabels(g.get_xticklabels(), rotation=45)
```

	id	bmi
0	9046	36.600000
1	51676	NaN

before

	id	bmi
0	9046	36.600000
1	51676	28.893237

after



Data Preparation

3. Transforming Categorical Features
   - Scikit-learn
     - Cannot take categorical features
   - *OneHotEncoder*
     - Categorical → Numeric
     - Each entry would be a feature on its own

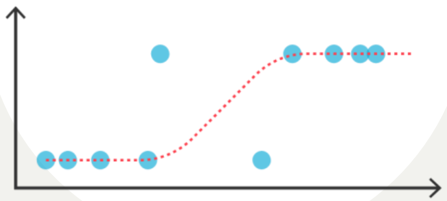
	gender
0	Male
1	Female
2	Male

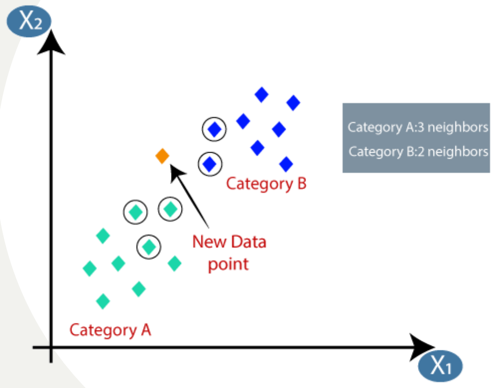
before

	gender_Female	gender_Male
0	0.0	1.0
1	1.0	0.0
2	0.0	1.0

after

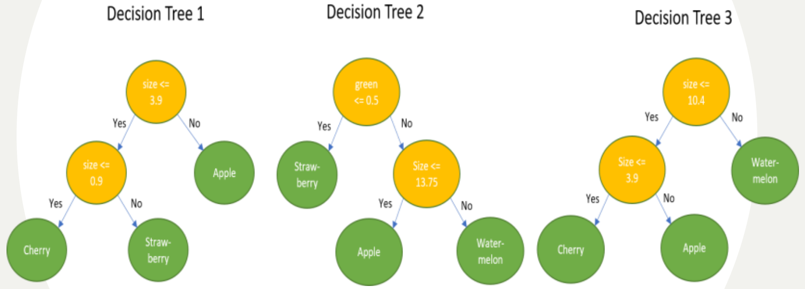
Features	Data Type
age	Float
hypertension	Int
heart_disease	Int
avg_glucose_level	Float
bmi	Float
stroke	Int
gender_Female	Float
gender_Male	Float
married_No	Float
married_Yes	Float
work_Govt_job	Float
work_Never_worked	Float
work_Private	Float
work_Self-employed	Float
work_children	Float
residence_Rural	Float
residence_Urban	Float
smoke_Unknown	Float
smoke_formerly smoked	Float
smoke_never	Float
smoke_smokes	Float

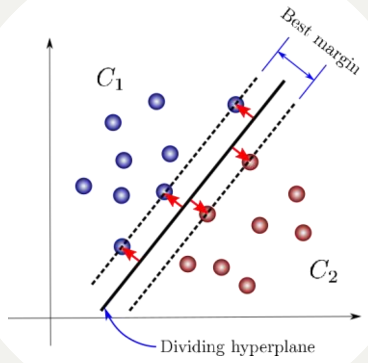




Models

1. Logistic Regression (LR)
2. K-Nearest Neighbours (kNN)
3. Random Forest (RF)
4. Support Vector Machines (SVM)





Train-Test Split

Train – 70% | Test – 30%

Incidence of Stroke	Count
0	4,861
1	249

IMBALANCED

Possible Solutions

1. Oversampling
2. Under-sampling
3. Combination

Incidence of Stroke	Count
0	3,403
1	3,403

Incidence of Stroke	Count
0	160
1	160

?

Evaluation Metrics

1. Accuracy
   - How accurate the model is
2. Precision rate
   - Correctly predicted at-risk vs. not at-risk
3. Recall rate
   - aka Sensitivity rate
   - How accurate the model is able to identify relevant data
   - *True Positives*
4. F1-score
   - Harmonic mean
   - Precision and Recall equally important

Imbalanced Dataset Results

LR	Precision	Recall	F1-score	Support
noStroke	0.95	1.00	0.98	1,458
stroke	1.00	0.01	0.03	75
accuracy			0.95	1,533
macro avg	0.98	0.51	0.50	1,533
weighted avg	0.95	0.95	0.93	1,533

kNN	Precision	Recall	F1-score	Support
noStroke	0.95	1.00	0.97	1,458
stroke	0.00	0.00	0.00	75
accuracy			0.95	1,533
macro avg	0.48	0.50	0.49	1,533
weighted avg	0.90	0.95	0.93	1,533

RF	Precision	Recall	F1-score	Support
noStroke	0.95	1.00	0.97	1,458
stroke	0.00	0.00	0.00	75
accuracy			0.95	1,533
macro avg	0.48	0.50	0.49	1,533
weighted avg	0.90	0.95	0.93	1,533

SVM	Precision	Recall	F1-score	Support
noStroke	0.95	1.00	0.97	1,458
stroke	0.00	0.00	0.00	75
accuracy			0.95	1,533
macro avg	0.48	0.50	0.49	1,533
weighted avg	0.90	0.95	0.93	1,533

Oversampling Dataset Results

LR	Precision	Recall	F1-score	Support
noStroke	0.98	0.75	0.85	1,458
stroke	0.12	0.69	0.21	75
accuracy			0.74	1,533
macro avg	0.55	0.72	0.53	1,533
weighted avg	0.94	0.74	0.82	1,533

kNN	Precision	Recall	F1-score	Support
noStroke	0.98	0.68	0.81	1,458
stroke	0.11	0.79	0.20	75
accuracy			0.69	1,533
macro avg	0.55	0.73	0.50	1,533
weighted avg	0.94	0.69	0.78	1,533

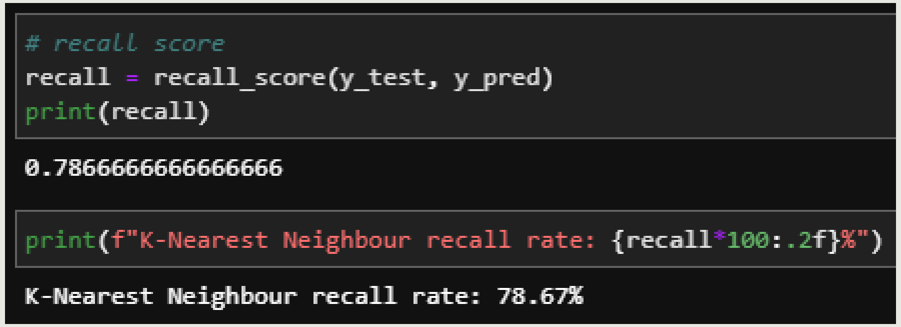
RF	Precision	Recall	F1-score	Support
noStroke	0.96	0.97	0.96	1,458
stroke	0.26	0.23	0.24	75
accuracy			0.93	1,533
macro avg	0.61	0.60	0.60	1,533
weighted avg	0.93	0.93	0.93	1,533

SVM	Precision	Recall	F1-score	Support
noStroke	0.98	0.74	0.84	1,458
stroke	0.12	0.72	0.21	75
accuracy			0.74	1,533
macro avg	0.55	0.73	0.53	1,533
weighted avg	0.94	0.74	0.81	1,533

Conclusion

- Choice of model for oversampling: K-Nearest Neighbours
  - Highest Recall rate: 78.67%

kNN	Precision	Recall	F1-score	Support
noStroke	0.98	0.68	0.81	1,458
stroke	0.11	0.79	0.20	75
accuracy			0.69	1,533
macro avg	0.55	0.73	0.50	1,533
weighted avg	0.94	0.69	0.78	1,533



```
# recall score
recall = recall_score(y_test, y_pred)
print(recall)

0.7866666666666666

print(f"K-Nearest Neighbour recall rate: {recall*100:.2f}%")

K-Nearest Neighbour recall rate: 78.67%
```

Future Work

- Under-sampling
- Combination
- Further fine-tuning of parameters
- More models
  - Decision Trees, Neural Networks
- Compare Results
- Find out which feature/s affects incidence of stroke

Limitation

- Missing BMI cannot simply be replaced by mean or median
- Dealing with an actual human

Thank You!