Machine Learning Application
Stroke Prediction

Davide Germinario, Gaelle Debree, Riccardo Scuriatti

INDEX

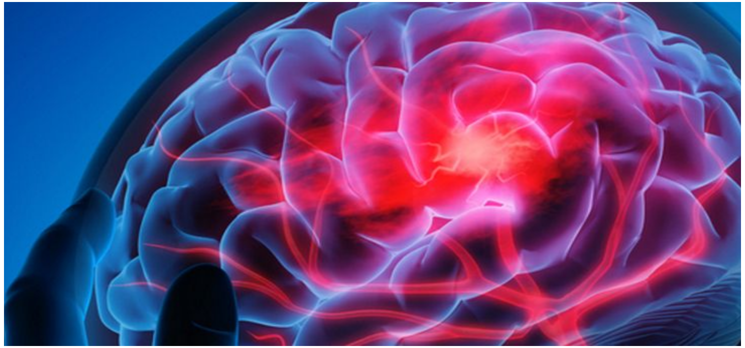
1. Importing the dataset
2. Cleaning the data
3. Descriptive analysis
4. Performance of predictive methods
5. K-means clustering

1. IMPORTING DATASET

Stroke Prediction

The following dataset present 11 clinical features for predicting stroke events.

Source: Kaggle



```
[ ]  # Importing the dataset
     import pandas as pd
     df = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

2. CLEANING THE DATASET

VARIABLES DESCRIPTION

```
[ ]  df.head()
```

	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

- **id**: unique identifier
- **gender**: "Male", "Female" or "Other"
- **age**: age of the patient
- **hypertension**: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- **heart_disease**: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- **ever_married**: "No" or "Yes"
- **work_type:** "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
- **Residence_type**: "Rural" or "Urban"
- **avg_glucose_level**: average glucose level in blood
- **bmi**: body mass index
- **smoking_status**: "formerly smoked", "never smoked", "smokes" or "Unknown" (if the smoking status is unavailable for this patient)
- **stroke**: 1 if the patient had a stroke or 0 if not

STEPS

- Remove ‘ID’ and ‘Work type’
- transform ‘Smoking status’
- Dummy for ‘Gender’, ‘Married’ and ‘Residence’
- Deal with NAs
- Deal with Outliers

REMOVE ‘ID’ AND ‘Work Type’

```
# Drop the 'id' and 'work_type' columns
columns_to_drop = ['id', 'work_type']
df = df.drop(columns=columns_to_drop, axis=1)
```

DEALING WITH ‘Smoking_Status’

```
[ ]  df['smoking_status'].value_counts()
```

```
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: smoking_status, dtype: int64
```

```
[ ]  value_to_remove = 'Unknown'
     df['smoking_status'] = df['smoking_status'].replace(value_to_remove, None)
     df = pd.get_dummies(df, columns=['smoking_status'])
```

DUMMY FOR ‘Gender’, ‘Married’ AND ‘Residence’

```
df['gender'] = (df['gender'] == 'Female').astype(int)
df['ever_married'] = (df['ever_married'] == 'Yes').astype(int)
df['Residence_type'] = (df['Residence_type'] == 'Urban').astype(int)
```

DEALING WITH NAs

The only column which is showing NA values is BMI which shows 201 NAs, we decide to fill the missing data with the mean of the column.

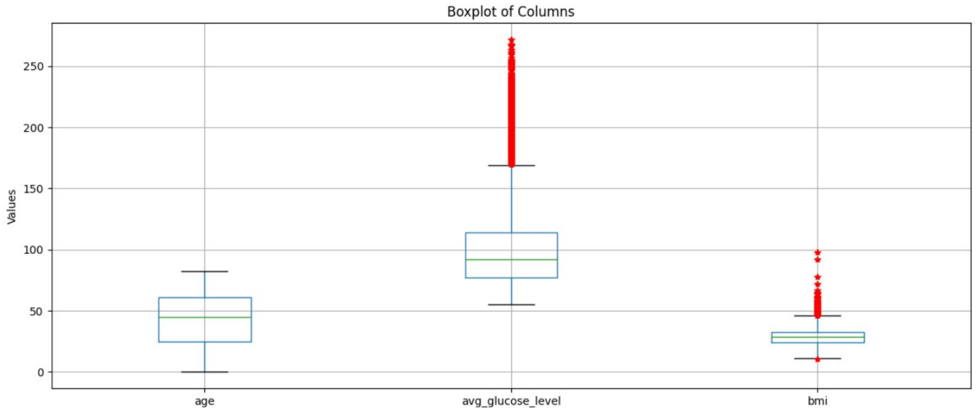
```
# See if my data contains NaNs
nan_count = df.isna().sum()
print(nan_count)
```

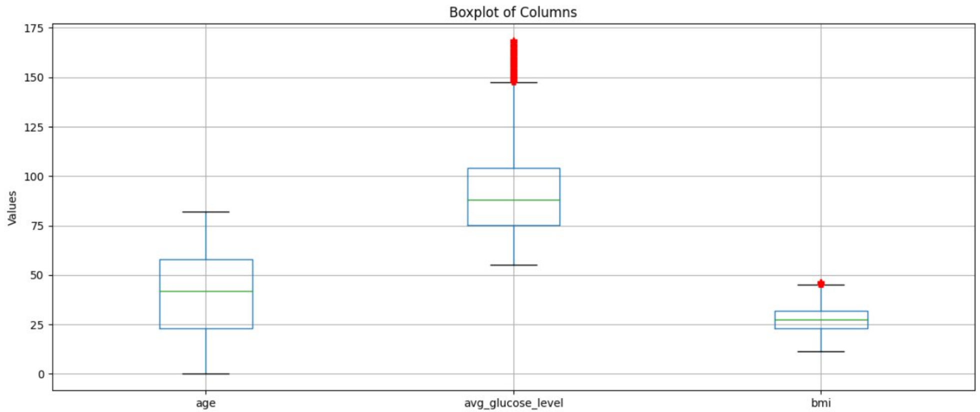
```
Residence_type         0
avg_glucose_level      0
bmi                  201
stroke                 0
```

```
[ ]  #Handling missing values by filling with the mean of the column
     df['bmi'].fillna(df['bmi'].mean(), inplace=True)
```

DEALING WITH OUTLIERS

- We calculate the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the current column.
- Calculates lower and upper bounds to identify outliers .
- Filters the df_filtered DataFrame to include only the rows where the current column values fall within the calculated bounds.



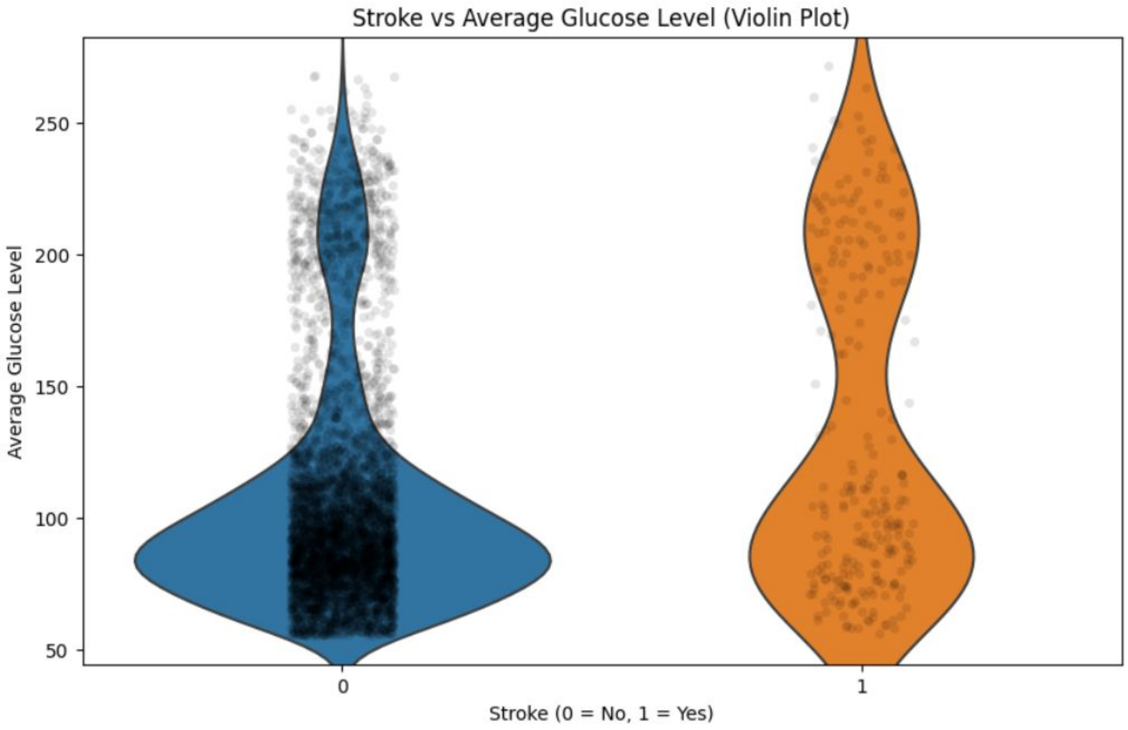


3. DESCRIPTIVE ANALYSIS

DESCRIPTIVE STATISTICS

```
Mean of each column:
 gender                             0.589843
age                                40.896406
hypertension                        0.074243
heart_disease                       0.039171
ever_married                        0.622865
Residence_type                      0.507857
avg_glucose_level                  91.477067
bmi                                27.811399
stroke                              0.037577
smoking_status_formerly smoked      0.161239
smoking_status_never smoked         0.363243
smoking_status_smokes               0.152585
```

DESCRIPTIVE STATISTICS



4. PERFORMANCE OF PREDICTING METHODS

DEFINE THE TARGET AND FEATURES

Defining the target variable of our analysis:

- Stroke: this is the variable that we want to predict

Define the features of our analysis:

- All other variables: these are all the variables that we will use to predict the stroke one

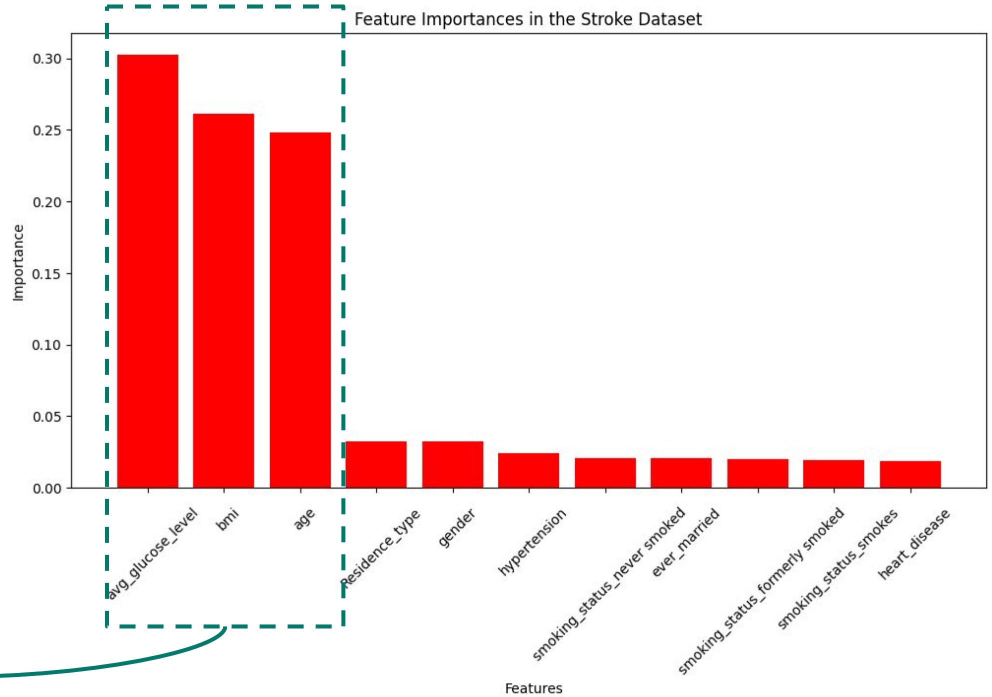
```
# Splitting the data into features (X) and target label (y)
X = df_filtered.drop(['stroke'], axis=1)

# Including only 'stroke' in y
y = df_filtered['stroke']
```

SELECTING RELEVANT FEATURES

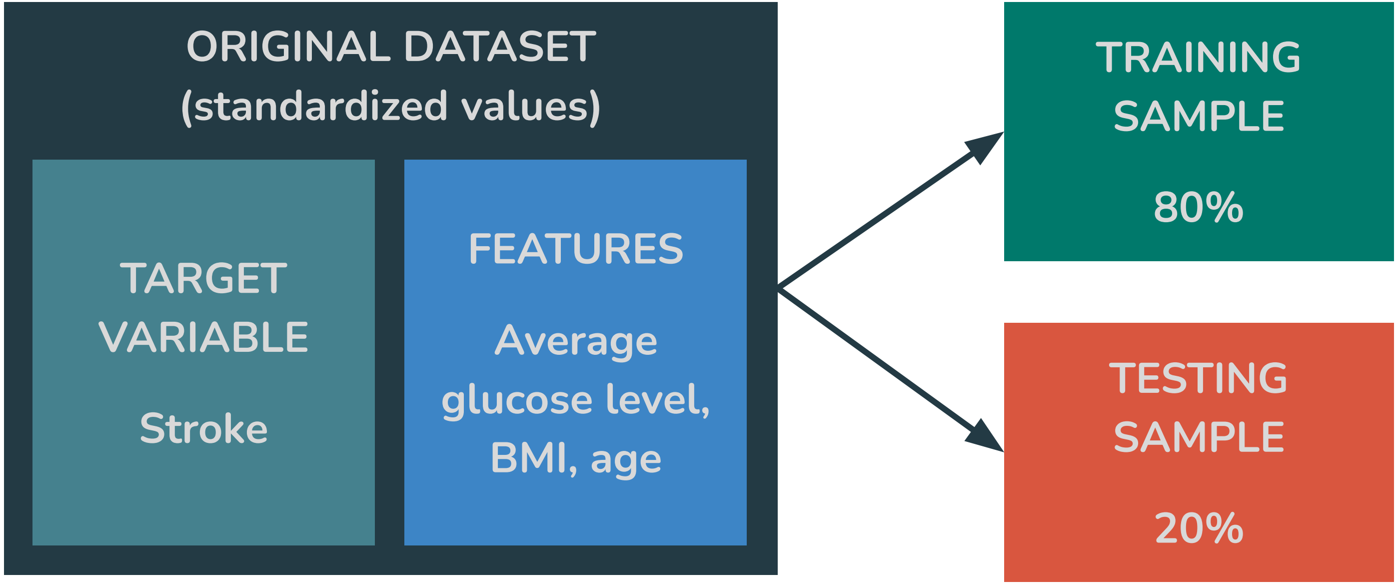
We use the **Random Forest classifier** to plot the features' importance. We select only the **three most important features** since all the others have a very low importance.

1. Average glucose level
2. Body Mass Index (BMI)
3. Age

These features will be used to perform the further predictions.

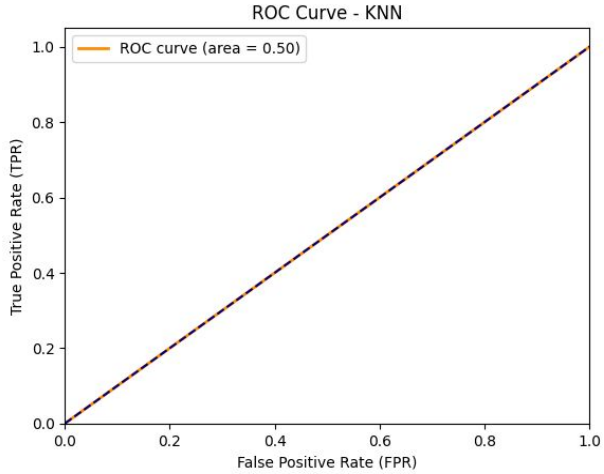


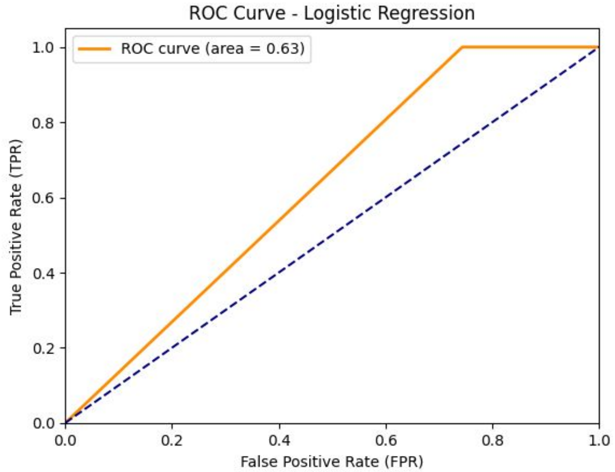
DEFINING TRAINING AND TESTING SAMPLE

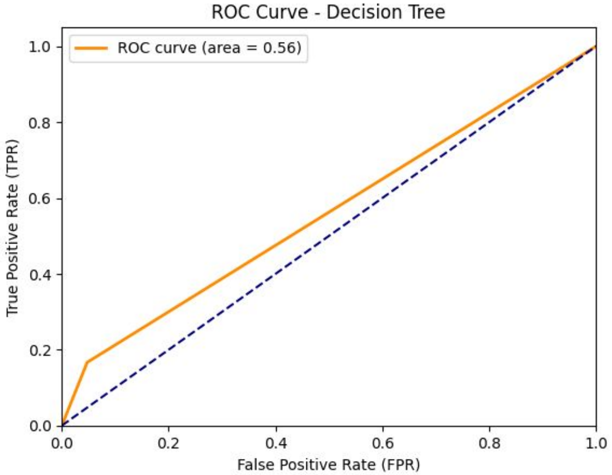


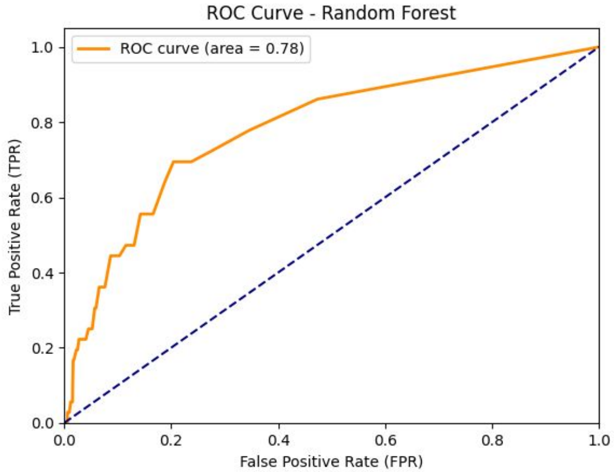
PERFORMANCE OF THE MODELS

	Accuracy	Precision	Recall	F1-Score	AUC-ROC
KNN	0.956	0.929	0.956	0.939	0.5
Logistic Regression	0.959	0.92	0.959	0.939	0.628
Decision Tree	0.92	0.93	0.92	0.925	0.56
Random Forest	0.953	0.92	0.953	0.936	0.781





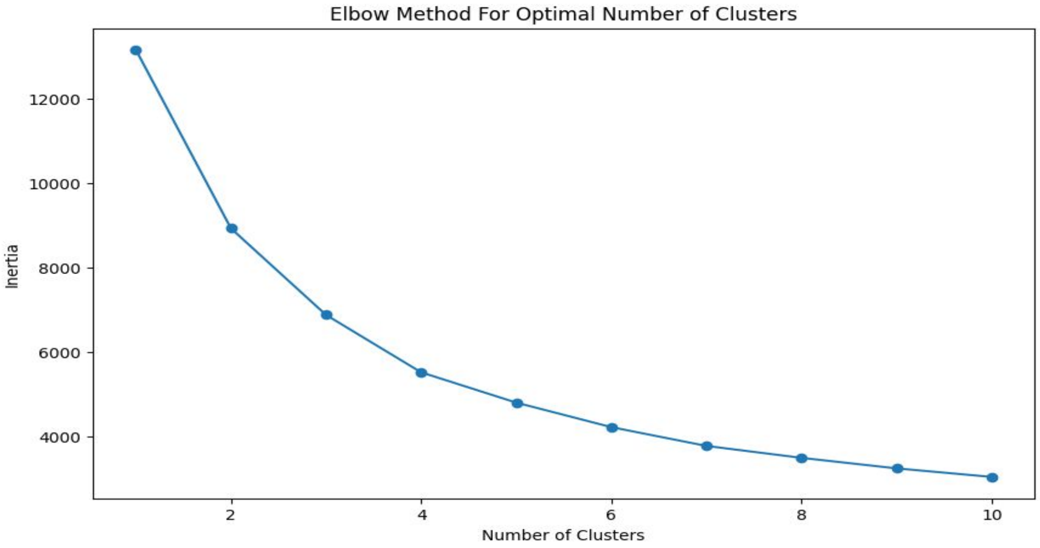




The **Random Forest** is the model for which the area below the ROC curve is the highest, it seems to be the **model predicting the best the stroke risk** whereas the KNN model presents the worse performance.

5. K-MEANS CLUSTERING

ELBOW METHOD




```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import numpy as np

# Selecting features for clustering
features_for_clustering = df_filtered[['age', 'avg_glucose_level', 'bmi']]

# Standardizing the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features_for_clustering)

# Finding the optimal number of clusters using the elbow method
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

# Plotting the elbow graph
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method For Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

3 CLUSTERS

cluster	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	stroke	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes
0	0.589901	54.260500	0.115621	0.064181	0.849929	0.510618	80.890788	31.095230	0.056630	0.220859	0.399245	0.178858
1	0.592347	17.547196	0.002208	0.001472	0.199411	0.509198	86.259779	21.896947	0.002943	0.065489	0.280353	0.090508
2	0.585980	44.634699	0.085433	0.037240	0.726177	0.499452	123.812903	28.993549	0.044907	0.165389	0.403067	0.184009

Cluster 0	Cluster 1	Cluster 2
Older	Younger	Middle-age
High health risks	Low health risks	Future health complications

CONCLUSION

- **Strong correlation** between high level of glucose, high BMI, higher age and risk of stroke.

- **Logisitic Regression Model** seems to be outstanding all other models to predict the stroke, even if the AUC below the **Random Forest** ROC curve is the highest one.

- The use of **3 clusters** would be optimal in this analysis.

REPRODUCIBILITY

Colab:
https://colab.research.google.com/drive/1bmWEGOlSx9s3sNmf1KjWf6LgcQWL_CId?usp=sharing

Stroke predictions dataset:
https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset