



---

# TASK 2 – CLASSIFICATION AND MODEL IMPROVEMENT

---

PDAN8411



JUNE 7, 2022  
SEAN TEUCHERT  
ST10041890

## Contents

Introduction: .....	2
What is classification? .....	3
Types of Learners in Classification: .....	3
Classification Algorithms:.....	3
Why is the dataset appropriate for classification? .....	4
What is the question that the analysis will answer? .....	4
Type of analysis performed: .....	4
How will the results be verified? .....	4
What features will be extracted? .....	7
Over & Under Fitting:.....	7
Imports Required:.....	7
How will train and test the Naïve Bayes Classifier? .....	8
The accuracy of the predictions: .....	8
How the results will be plotted? .....	9
How changing the training/test splits affects the predictions and accuracy: .....	9
K-NN algorithm vs Naïve Bayes Classifier: .....	9
How was the model evaluated and improved? .....	10
Summary of Code: .....	11
Summary of Results: .....	12
Conclusion: .....	13
References.....	14

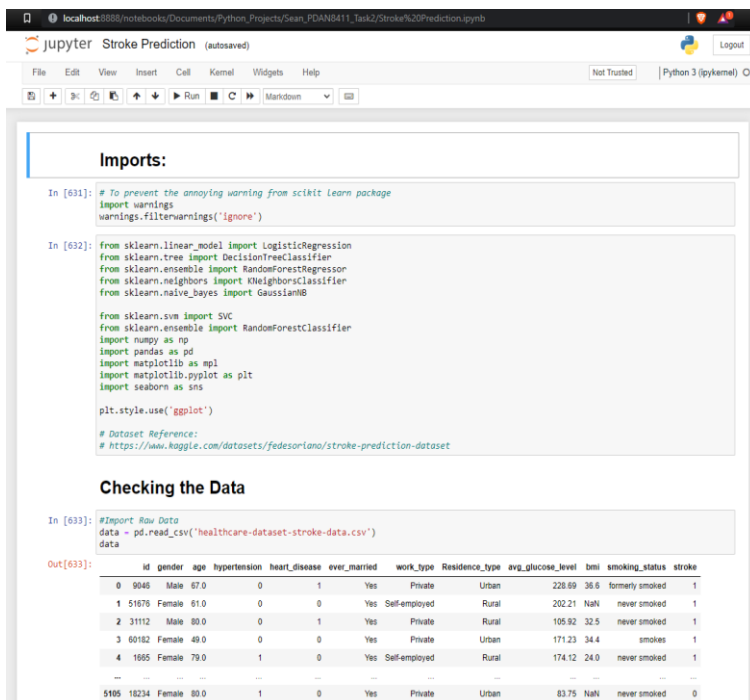
## Introduction:

The dataset used to explain classification and build the model is of 11 clinical features for predicting stroke events. According to the World Health Organization (WHO) stroke is the second leading cause of death globally (11% of total deaths). The dataset is used to predict whether a patient is likely to have a stroke based on the input parameters (Gender, Age, Various diseases, Smoking status). A stroke is a medical emergency and occurs when the blood flow to a part of the brain is interrupted or reduced, preventing brain tissue from getting oxygen and nutrients. Within Minutes brain cells can begin to die, therefore time is crucial. This highlights the importance of acting preventative rather than reactive. A predictive model can assist with diagnosing potential stroke victims.

The author of the dataset is Fedesoriano, and the dataset can be found on Kaggle using the following link:

<https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>

A Jupyter Notebook was used to develop the initial code, but I used Visual Studio Code to write the finalized code. The file types are the same therefore the code can be run in Jupyter Notebook or Visual Studio Code.



The screenshot shows a Jupyter Notebook titled "Stroke Prediction" with a toolbar at the top. The code is as follows:

```
In [631]: # To prevent the annoying warning from scikit learn package
import warnings
warnings.filterwarnings('ignore')

In [632]: from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

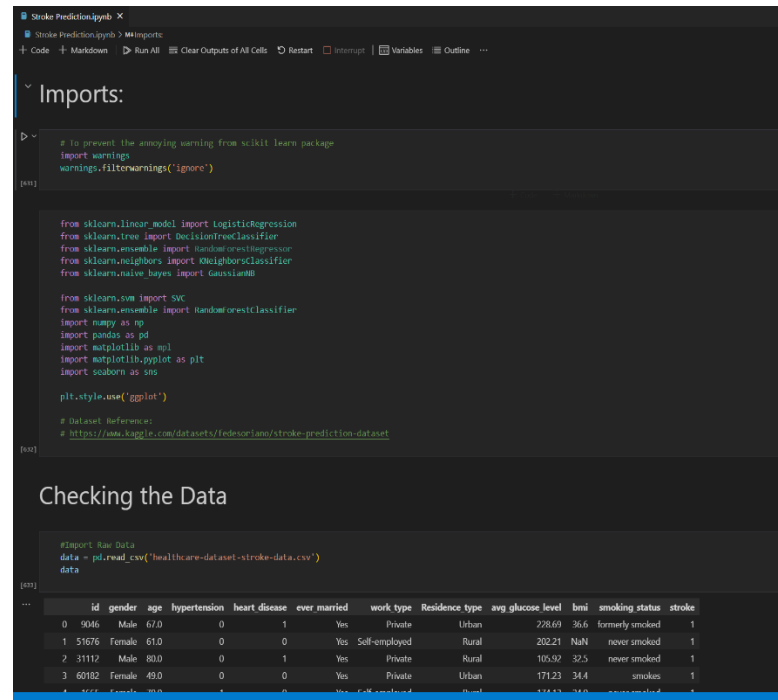
plt.style.use('ggplot')

# Dataset Reference:
# https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
```

Below the code, the output shows the first few rows of the dataset:

	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	105.82	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

Figure 1: Jupyter Notebook



The screenshot shows a Jupyter Notebook titled "Stroke Prediction.ipynb" within the Visual Studio Code interface. The code is as follows:

```
# To prevent the annoying warning from scikit learn package
import warnings
warnings.filterwarnings('ignore')

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')

# Dataset Reference:
# https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
```

Below the code, the output shows the first few rows of the dataset:

	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	105.82	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0

Figure 2: Jupyter Notebook via Visual Studio Code

## What is classification?

Classification is a task that requires the use of machine learning algorithms that learn how to assign a class label to input data (Brownlee, 2020). Classification is a type of supervised learning. An example is classifying three different fruits and assigning a class label (Banna, Orange, Apple) to the corresponding fruit. A dataset with data about different fruit variables (size, color, shape, etc.) is used in a model to predict the class labels.

## Types of Learners in Classification:

There are two types of learners in classification:

- Lazy Learner: stores the training data until test data appears. Classification is performed based on the most related data in the stored training data (Asiri, 2018).
- Eager Learner: constructs a classification model based on the given data before receiving data for classification (Asiri, 2018). It must be able to commit to a single hypothesis that covers the entire instance space.

When compared to Eager Learners, Lazy Learners take less time training but takes more time to predict vice versa.

## Classification Algorithms:

There are six types of classification algorithms:

- Logistic Regression:
- Decision Tree:
- Random Forest:
- Support Vector Classifier
- K-Nearest Neighbor:
- Naïve Bayes:

### Why is the dataset appropriate for classification?

The dataset is appropriate for classification because it contains categorical and numerical data that is labeled. The numerical data can be converted to categorical data by using dummy values. The dataset is also made up of historical data about victims that suffered from a stroke and people that have not suffered from a stroke. This will enable us to create a good baseline for factors that may increase the risk of stroke. The dataset contains 5110 records but only 5% are stroke victims. This can become a problem with having a small target group but using SMOTE can increase the accuracy of the model and solve this problem. To classify a potential stroke victim, it is important include all potential factors in the model as not one single factor can determine a stroke. Classification is the best analysis method for this dataset, there are multiple factors that may contribute to having a stroke and one factor may occur in one case but not the other, therefore it will be inefficient to use multiple regression.

### What is the question that the analysis will answer?

The purpose of the analysis is to answer the following questions:

- Is there a relationship between various healthy and unhealthy habits to heart stroke?
  - Does age have an impact on strokes?
  - Does Body Mass Index and Glucose levels have impact on the risk of stroke?
  - Does Smoking increase the risk of a stroke?
  - Can Heart Disease increase the risk of stroke?
  - Does workload impact the risk of having a stroke?
- Is it possible to predict whether a person is likely to have a stroke when given their medical history?

### Type of analysis performed:

Descriptive analysis has been performed to visualize the dataset variables and how they impact the risk of stroke. Afterwards a predictive model and classification algorithms have been used to build a model to predict if a person is at risk of having a stroke based on their health statistics.

### How will the results be verified?

The results will be verified using sklearn accuracy test, f1\_score, precision score, cross validation and by comparing the predicted values to the actual values.

**\*NB Three algorithms were used and all three were verified using the following methods. The example below is using the K-NN algorithm:**

The results have been verified by using sklearn accuracy test:

```
print(f"Accuracy Score : {round(accuracy_score(y_test, prediction) * 100, 2)}%")
```

Accuracy Score : 96.09%

Figure 3: Accuracy score method

Accuracy score is used to tell us how often we can expect our model will correctly predict an outcome out of the total number of times it made predictions (Kumar, 2022). An accuracy score of 96.09 is good and this means that majoring of the time it will make accurate predictions.

The results have been verified by using sklearn classification report:

```
print(classification_report(y_test, prediction))
```

	precision	recall	f1-score	support
0	0.98	0.94	0.96	979
1	0.94	0.99	0.96	965
accuracy			0.96	1944
macro avg	0.96	0.96	0.96	1944
weighted avg	0.96	0.96	0.96	1944

Figure 4: Classification Report

From sklearn's classification report, we can get the precision, recall and f1\_score scores for the model. The precision score measures the proportion of positively predicted labels that are actually correct (Kumar, 2022). A precision score of 0.98 for predicted non-stroke and 0.94 for predicted stroke is a good score. This means that majority of the positive predicted values are correct.

Recall score represents the model's ability to correctly predict the positives out of actual positives (Kumar, 2022). A recall score of 0.94 for non-stroke and 0.99 for stroke is a good score. This tells us that the model is good at identifying all actual positive out of all positives that exist within the dataset.

F1\_score represents the model score as a function of precision and recall score. F1\_score is a performance metric that gives equal weight to both the Precision and Recall for measuring its performance in terms of accuracy (Kumar, 2022). A f1\_score of

0.96 for non-stroke and stroke is a good score. This tells us that the model is optimized, and it is providing accurate outputs.

The results have been verified by using sklearn cross validation:



Figure 5: Cross Validation Method

From the method used in figure 5, we can identify the cross-validation scores for each algorithm. Using this method, we can identify which classification algorithm is the best for the dataset. It is important to note that these scores are before Hyper Parameter tuning therefore the scores are lower than figure 3 & 4. A cross validation score of 90.8% for K-Nearest Neighbors is a good score. This tells us that the model will accurately predict outcomes if this algorithm is used. The score can be improved by tuning the Hyper Parameters. A low standard deviation will also ensure that the results are normally distributed and there wont be many outliers in the predictions.

## What features will be extracted?

The following features have been extracted from the dataset:

- gender
- age
- hypertension
- heart\_disease
- ever\_married
- work\_type
- Residence\_type
- Avg\_glucose\_level
- bmi

## Over & Under Fitting:

Overfitting refers to a model that models the training data too well or has been over trained with a large amount of data. This can become a problem because the model will not be able to correctly analyze a new dataset. To combat overfitting, I have made use of cross validation and the K-Folds method.

Underfitting refers to a model that cannot capture the underlying trend of the data, this is a result of using the incorrect model to analyze the data or by not providing enough test data. To combat underfitting, I have selected a large dataset and built the model using SMOTE to increase the sample size of the stroke victims.

## Imports Required:

- warnings – used to ignore unnecessary scikit learn warnings
- Pandas – used to clean the data
- Numpy – used for arrays and metrics
- Seaborn – used to visualize the data
- Matplotlib.pyplot – used to visualize the data
- Sklearn.model\_selection import train\_test\_split – used to split the dataset into train and test data
- Sklearn.linear\_model – Logistic Regression Classifier
- Sklearn.tree – Decision Tree Classifier
- Sklearn.neighbors – K-Nearest Neighbors Classifier
- Sklearn.naive\_bayes – Naïve Bayes Classifier
- Sklearn.svm – Support Vector Classifier
- Sklearn.ensemble – Random Forest Classifier
- Sklearn.metrics – imports scoring algorithms
- sklearn.model\_selection – K-Fold method import



## How will train and test the Naïve Bayes Classifier?

The Naïve bayes classifier will be trained and tested using an 80% train and 20% test data split. The dataset contains 5110 observations; therefore, a larger train data split will provide a more accurate results and ensure the model isn't training itself to that specific dataset.

The screenshot below demonstrates the method to split the dataset:

```
> # split the data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train, test, test_size = 0.2, random_state = 42, shuffle = True)

[666]
```

Figure 6: Train\_test\_split method

This method splits the dataset into train and test data. Shuffle has been set to true to ensure that all the data is used, this will increase the accuracy of the model.

The screenshot below demonstrates the method to fit and predict

```
# Naive Bayes

gnb.fit(X_train, y_train)

prediction = gnb.predict(X_test)

print(f"Accuracy Score : {round(accuracy_score(y_test, prediction) * 100, 2)}%")

]

Accuracy Score : 66.62%
```

Figure 7: Fit and Predict method

Using the Naïve Bayes classifier, the model has been fitted and can predict possible outcomes. The print method gets the accuracy score using the sklearn package.

## The accuracy of the predictions:

During analysis, three classification algorithms were used and tested. After Hyper parameter tuning, K-NN algorithm had an accuracy score of 96.1% which outperformed the Naïve bayes algorithm that had an accuracy score of 72.2%. The best overall performing algorithm was the Random Forest Classifier with an accuracy score of 96.5%. K-NN algorithm and Random Forest algorithm both share the same F1\_score therefore either algorithm will work for this model.

### How the results will be plotted?

The data will be visualized using a combination of graphs. The graphs that will be used are the following:

- Pie charts
- Bar graphs
- Line graphs
- Distribution graphs

### How changing the training/test splits affects the predictions and accuracy:

The split required to analyze the dataset depends on the size of the dataset. It is better to have a larger train split (80% train & 20% test) if the dataset contains many records. The dataset used during analysis contains 5110 observations, therefore it is better to use an 80% & 20% split. If the test data size was increased the noise in the model will increase, resulting in the model making inaccurate predictions when given a new dataset. Reducing the test data split will result in inaccurate results due to the lack of data to evaluate the predictions.

### K-NN algorithm vs Naïve Bayes Classifier:

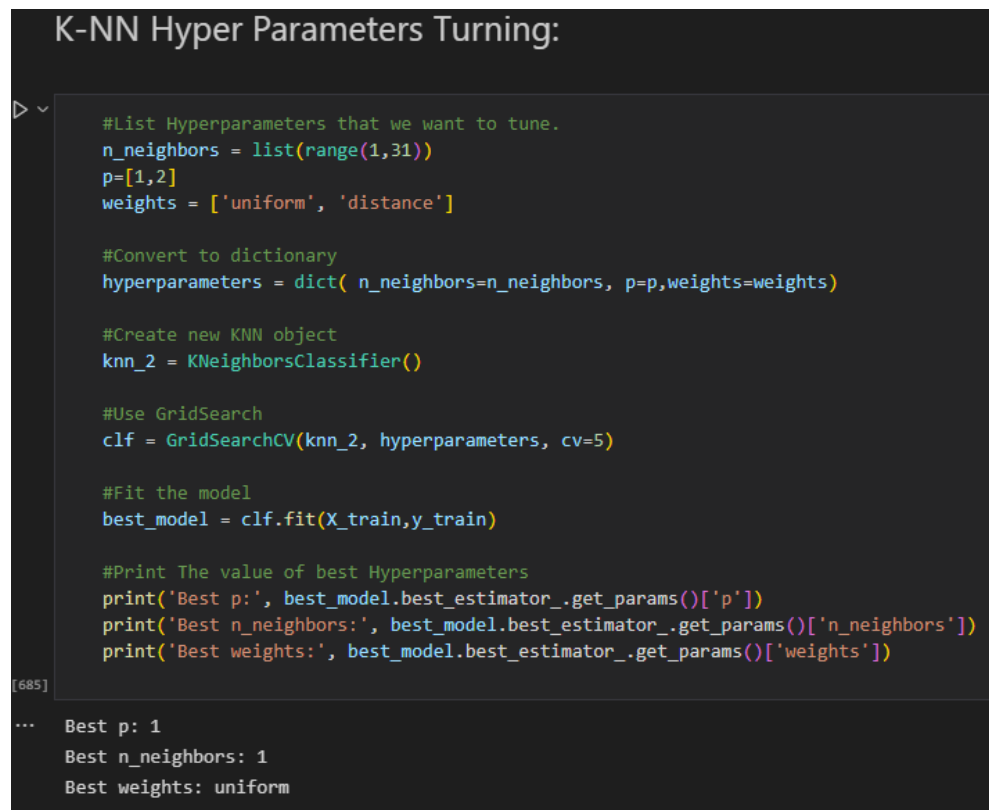
Before Hyper Parameter tuning, K-NN algorithm had an accuracy score of 90.1% which outperformed the Naïve bayes algorithm that had a score of 66.6%. The K-NN algorithm also outperformed Naïve Bayes algorithm in f1\_score. After Hyper Parameter tuning, K-NN algorithm had an accuracy score of 96.1% which outperformed the Naïve bayes algorithm that had an accuracy score of 72.2%. To conclude, the best algorithm to use for this model is the K-Nearest Neighbors algorithm. It has a better accuracy score and f1\_score which means it will produce more accurate outcomes.

How was the model evaluated and improved?

The model was evaluated by predicting outcomes and comparing the different algorithms based on their accuracy scores and cross validation scores. A graph was also used to evaluate the predicted values compared to the actual values.

**\*NB the following steps were applied to all three algorithms**

The screenshot below demonstrates the GridSearchCV method:



```
K-NN Hyper Parameters Turning:

#List Hyperparameters that we want to tune.
n_neighbors = list(range(1,31))
p=[1,2]
weights = ['uniform', 'distance']

#Convert to dictionary
hyperparameters = dict( n_neighbors=n_neighbors, p=p,weights=weights)

#Create new KNN object
knn_2 = KNeighborsClassifier()

#Use GridSearch
clf = GridSearchCV(knn_2, hyperparameters, cv=5)

#Fit the model
best_model = clf.fit(X_train,y_train)

#Print The value of best Hyperparameters
print('Best p:', best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])
print('Best weights:', best_model.best_estimator_.get_params()['weights'])

[685]
... Best p: 1
Best n_neighbors: 1
Best weights: uniform
```

Figure 8: GridSeachCV method

After the first iteration, GridSearchCV was used to find the optimal Hyper Parameters. The method stores the Hyper Parameter variables in a dictionary and uses the GridSearchCV method to iterate through five folds and test for the best Hyper Parameters. After identifying the best Hyper Parameters, a pipeline is used to scale the data and predict

The screenshot below demonstrates the pipeline used to improve the model:

```
# KNN Pipeline

pipeline = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors = 1, p = 1, weights = 'uniform'))

pipeline.fit(X_train, y_train)
prediction = pipeline.predict(X_test)

print(f"Accuracy Score : {round(accuracy_score(y_test, prediction) * 100, 2)}%")
```

Accuracy Score : 96.09%

Figure 9: K-NN Pipeline

A pipeline is used to scale the dataset and select the best Hyper Parameters before fitting the model. StandardScaler is used to resize the distribution of values to ensure the mean of the observed values is 0 and the standard deviation is 1. This will improve the accuracy of the model. As we can see in figure 9, the accuracy score has increased to 96.1 % from 90.1% (According to figure 5 cross validation score). By using the StandardScaler and the best Hyper Parameters, the model's accuracy improved by 6%.

### Summary of Code:

The model created follows the structure of Figure 10:

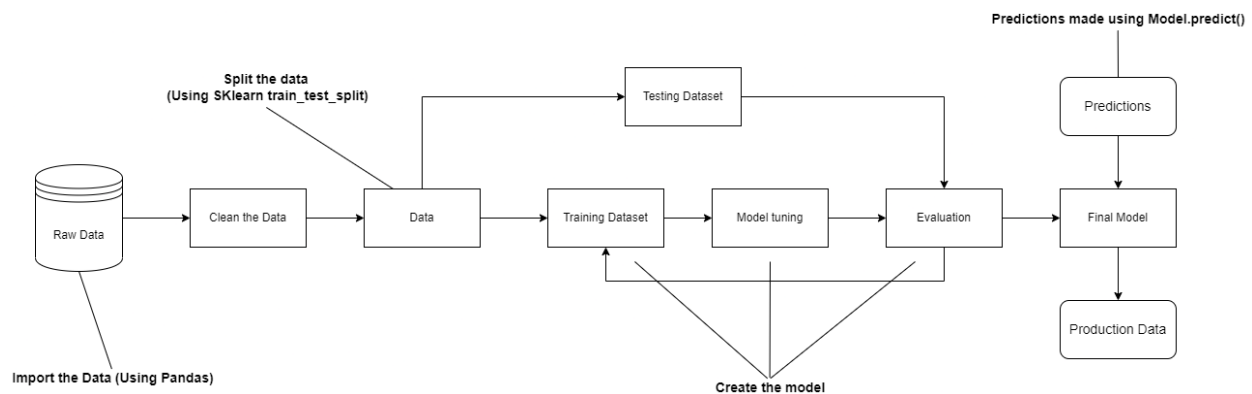


Figure 10: Machine Learning Model Diagram

The raw data is first imported using Pandas. The data is visualized using graphs to gain an understanding of any potential trends. The data is then cleaned using Featured engineering to eliminate any null values and to increase the sample group size. Next step is to split the dataset into training and test data. The dataset is tested using the K-NN , Naïve Bayes and Random Forest algorithm. Once the dataset has been tested, GridSearchCV is used to identify the best Hyper parameters and the model is retrained using the best Hyper Parameters and StandardScaler. After tuning the model, the model is evaluated using the sklearn accuracy score, f1\_score and cross validation.

The last step is to predict outcomes and make use of a graph to compare the predicted values to the actual values.

### Summary of Results:

After building the model and performing analysis the following results were produced:

Looking at the distribution of the variables:

- Good distribution for age
- BMI contained outliers
- Average glucose level is normally distributed

After visualizing the dataset and performing Descriptive analysis, the following results were determined:

- Only 5% of the data is stroke victims
- The dataset contains more females than males
- Proportionally Males are more likely to have a stroke
- Being married increases the risk of stroke
- Living in Urban area increases the risk of stroke
- Working increases the risk of stroke
- Smoking increases the risk of stroke
- Strokes are more common among older people (55 or up)
- Hypertension increases the risk of stroke
- Heart disease increases the risk of stroke
- Average glucose level is 0 – 50
- High average glucose level increases risk of stroke
- Majority of BMIs fall within 20-30
- High BMI does not increase risk of stroke

After building the model without StandardScaler and non-tuned Hyper Parameters, the following results were determined:

- Naïve Bayes accuracy score is 90.1%
- K-NN accuracy score is 66.6 %
- Random Forest accuracy score is 96.2%

After building the model with a pipeline, StandardScaler and tuned Hyper Parameters, the following results were determined:

- Naïve Bayes accuracy score is 77.2%
- K-NN accuracy score is 96.1 %
- Random Forest accuracy score is 96.5%

## Conclusion:

After visualizing the data, it is clear that all the variables are important. Each variable has the potential to increase the risk of stroke, but it is not the direct cause for having a stroke. For Example, a person that smokes has a higher chance of having a stroke but it is not guaranteed that they will have a stroke because they are a smoker. After building the model using the health statistics and habits, the model achieved an accuracy score of 96.1% using the K-NN algorithm. Naïve Bayes algorithm under performed when compared to K-NN and the overall best algorithm by a small margin is the Random Forest Classifier. This model can be used to identify and diagnose potential stroke patients. When dealing with stroke patients, it is important to be efficient and be able to react quickly because brain cells can start to die within minutes after having a stroke. This means it is better to act preventative and identify potential stroke patients to ensure they receive the best care possible and as soon as possible. The outcome achieved from this model is that classification (specifically K-NN or Random Forest classifier) can be used to predict which patients may suffer from a stroke based on their medical history. All the analysis questions are answered by the results produced by the model and the model was able to successfully identify potential stroke patients.

## References

Asiri, S., 2018. *Towards Data Science*. [Online]

Available at: <https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623>

[Accessed 06 06 2022].

Brownlee, J., 2020. *Machine Learning Mastery*. [Online]

Available at: <https://machinelearningmastery.com/types-of-classification-in-machine-learning/>

[Accessed 06 06 2022].

Kumar, A., 2022. *Vital Flux*. [Online]

Available at: <https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/>

[Accessed 06 06 2022].