TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Stroke is a medical emergency that occurs when the blood supply to a part of the brain is disrupted, either by a blockage or bleeding. This disruption can cause brain cells to die, leading to serious complications such as paralysis, cognitive impairment, and even death.

Due to the severity of stroke and its impact on individuals and society, predicting the likelihood of a stroke can be of great importance. Identifying individuals who are at high risk of stroke can help prevent the occurrence of stroke by implementing lifestyle changes, medication, and other interventions.

Several risk factors have been identified as contributing to the development of stroke, including high blood pressure, smoking, diabetes, and obesity. By analyzing these risk factors, along with other factors such as age, gender, and family history, healthcare professionals can predict an individual's risk of stroke and develop a personalized prevention plan.

There are also various machine learning algorithms and models that can be used to predict stroke risk by analyzing data from electronic health records, medical imaging, and other sources. These models can aid healthcare providers in making more accurate and efficient decisions, ultimately leading to better stroke prevention and management.

CHAPTER 2

OBJECTIVE

The objective of stroke prediction is to identify individuals who are at risk of having a stroke in the future. A stroke occurs when the blood supply to a part of the brain is interrupted, resulting in damage to brain tissue. Strokes can lead to serious disabilities, including paralysis, speech difficulties, and cognitive impairment. In some cases, strokes can be fatal.

To predict the likelihood of a stroke, healthcare professionals may use a variety of risk factors, such as age, sex, family history, medical history, lifestyle factors, and biomarkers. These risk factors are used to calculate an individual's risk score, which can be used to guide clinical decision-making and interventions.

Ultimately, the goal of stroke prediction is to identify individuals who are at high risk of stroke and implement interventions to prevent strokes from occurring. This may include lifestyle modifications, such as improving diet and exercise habits, as well as medical interventions, such as medication or surgery. By identifying and managing stroke risk factors, healthcare professionals can reduce the incidence of stroke and improve overall health outcomes.

CHAPTER 3

DATASET

This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relavant information about the patient.

RangeIndex: 5110 entries, 0 to 5109

Data columns (total 11 columns):

#	Column	Non-Null Count	Dtype
0	gender	5110 non-null	object
1	age	5110 non-null	float64
2	hypertension	5110 non-null	int64
3	heart_disease	5110 non-null	int64
4	ever_married	5110 non-null	object
5	work_type	5110 non-null	object
6	Residence_type	5110 non-null	object
7	avg_glucose_level	5110 non-null	float64
8	bmi	4909 non-null	float64
9	smoking_status	5110 non-null	object
10	stroke	5110 non-null	int64

dtypes: float64(3), int64(3), object(5)

Number of Columns = 11

Number of Rows = 5110

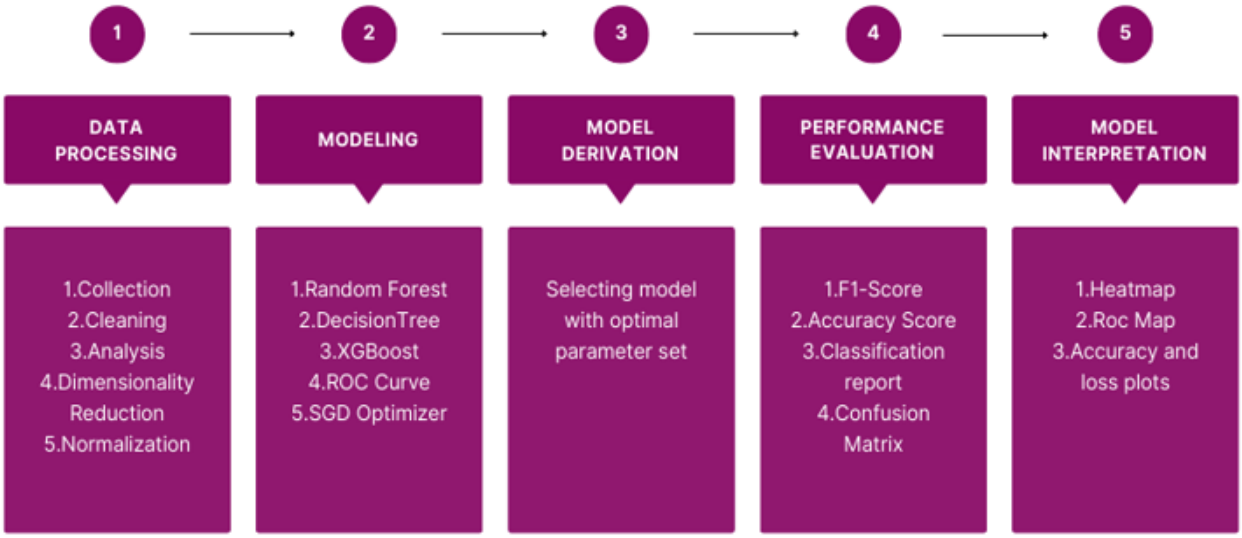
Attribute Information:

1) id: unique identifier
2) gender: "Male", "Female" or "Other"
3) age: age of the patient
4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
6) ever_married: "No" or "Yes"
7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
8) Residence_type: "Rural" or "Urban"
9) avg_glucose_level: average glucose level in blood
10) bmi: body mass index
11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
12) stroke: 1 if the patient had a stroke or 0 if not

*Note: "Unknown" in smoking_status means that the information is unavailable for this patient

CHAPTER 4

METHODOLOGY



- Data preprocessing refers to the process of preparing and cleaning raw data to make it usable for analysis. It involves various techniques such as removing duplicates, handling missing values, scaling, normalization, and transforming data into a suitable format for analysis. The goal of data preprocessing is to improve data quality, eliminate inconsistencies, and ensure that the data is suitable for the analysis task at hand.
- The model is then used to make predictions or decisions about new data.This process typically involves selecting an appropriate algorithm, tuning its parameters, and evaluating its performance on a training dataset before testing it on a separate validation or test dataset.
- We use Performance evaluation for assessing how well a model is able to make accurate predictions on new, unseen data. The goal is to determine the model's effectiveness and identify areas where it may need improvement.

CHAPTER 5

CODE AND RESULT

#Importing Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=10)
```

#Loading the dataset:

```
df = pd.read_csv('stroke.csv') df=df.iloc[:,1:]
```

#Exploratory Data Analysis:

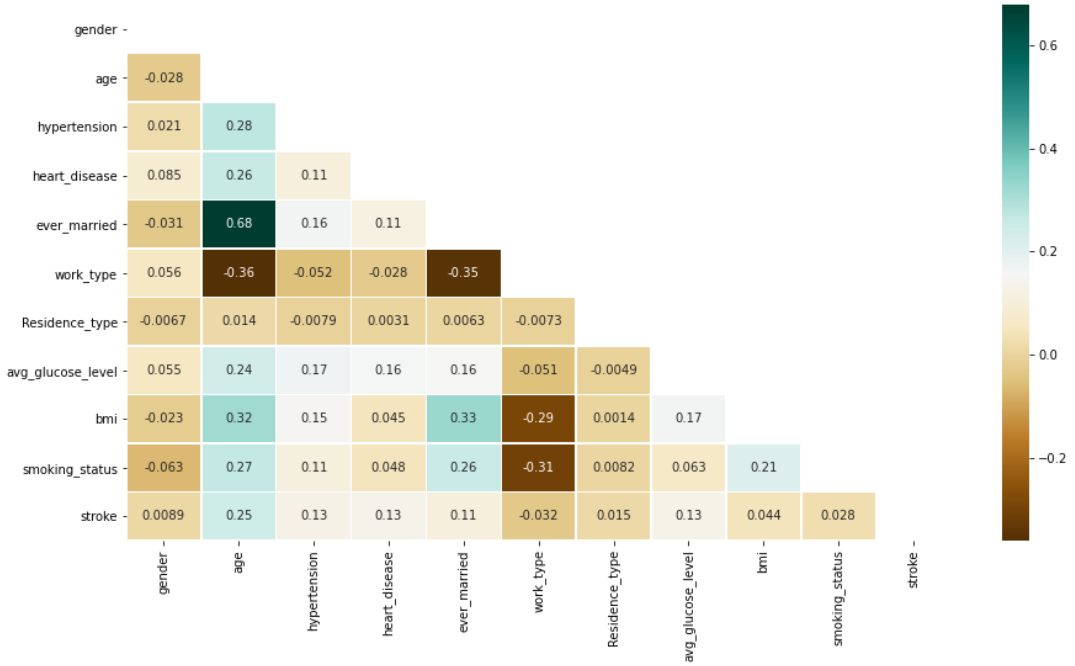
```
df = df.fillna(method='ffill')
from sklearn.preprocessing import LabelEncoder
columns=["gender","ever_married","work_type","Residence_type","smoking_status"]
le= LabelEncoder()
for col in df.columns:
    if col in columns:
        df[col]=le.fit_transform(df[col])
x=df.drop('stroke',axis=1) y=df['stroke']
```

#Correlation Plot

```
corr = df.corr()
plt.figure(figsize=(15,8))
```

mask = np.triu(np.ones_like(corr, dtype=bool))

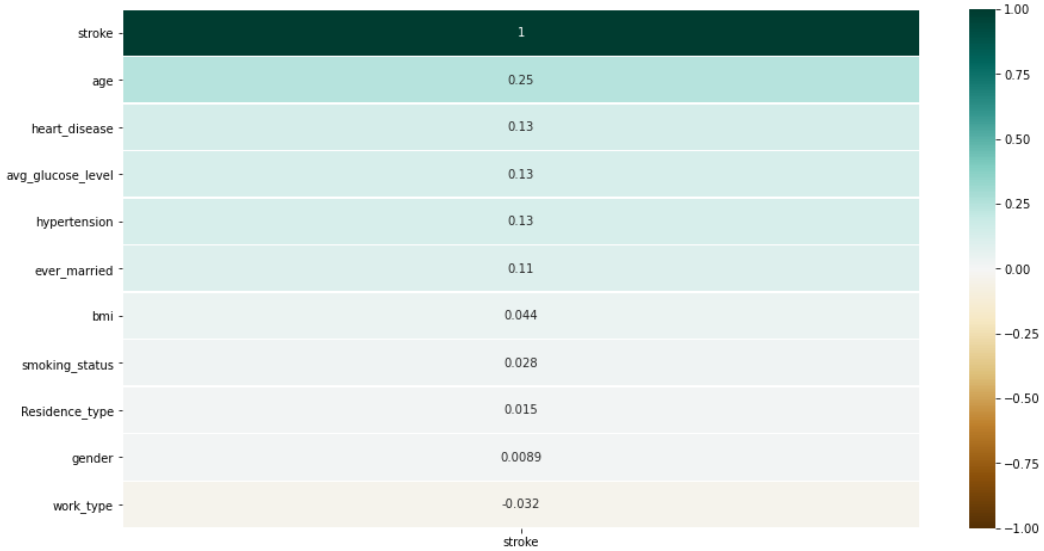
sns.heatmap(corr,mask=mask,cmap='BrBG',annot=True,linewidth=.5,square=False)



#Correlation of all columns with respect to class

plt.figure(figsize=(15,8))

sns.heatmap(df.corr()[['stroke']].sort_values(by='stroke', ascending=False),cmap='BrBG', vmin=-1, vmax= 1 , center=0, annot=True,linewidth=.5,square=False)



#Training and testing the data

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42, stratify=y)

from sklearn.decomposition import PCA

pca = PCA()

x_train = pca.fit_transform(X_train)

x_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_

explained_variance
```

o/p: array([7.94969336e-01, 1.83062520e-01, 2.09152286e-02, 4.72189324e-04, 3.02612027e-04, 9.52872077e-05, 9.17105617e-05, 4.36928891e-05, 3.00907849e-05, 1.73327338e-05])

#Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(random_state=42, max_depth=7)

dtc = dtc.fit(X_train, y_train)

y_pred_dtc = dtc.predict(X_test)

from sklearn.metrics import *

dtc_acc = accuracy_score(y_test, y_pred_dtc)

dtc_acc
```

o/p: 0.9393346379647749

```
print(classification_report(y_test, y_pred_dtc))
```

o/p:

	precision	recall	f1-score	support
0	0.95	0.99	0.97	972
1	0.07	0.02	0.03	50
accuracy			0.94	1022
macro avg	0.51	0.50	0.50	1022
weighted avg	0.91	0.94	0.92	1022

print(confusion_matrix(y_test, y_pred_dtc))

o/p:[[959 13]
 [49   1]]

f1_dtc = f1_score(y_test, y_pred_dtc)

print("F1-Score: ", f1_dtc)

o/p: F1-Score: 0.03125

#Random Forest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=100, random_state=42)

rfc = rfc.fit(X_train, y_train)

y_pred_rfc = rfc.predict(X_test)

f1_rfc = f1_score(y_test, y_pred_rfc)

print("F1-Score: ", f1_rfc)

o/p: F1-Score: 0.021443223421211121

print("Accuracy: ", accuracy_score(y_test, y_pred_rfc))

o/p: Accuracy: 0.9481409001956947

print(classification_report(y_test, y_pred_rfc))

o/p:

	precision	recall	f1-score	support
0	0.95	1.00	0.97	972
1	0.00	0.00	0.00	50
accuracy			0.95	1022
macro avg	0.48	0.50	0.49	1022
weighted avg	0.90	0.95	0.93	1022

```
print(confusion_matrix(y_test, y_pred_rfc))
```

o/p: [[971 1]
 [50 0]]

#XGBoost

```
from xgboost import XGBClassifier

xg = XGBClassifier()

xg = xg.fit(X_train, y_train)

y_pred_xg = xg.predict(X_test)

f1_xg = f1_score(y_test, y_pred_xg)

print("Accuracy Score: ", accuracy_score(y_test, y_pred_xg))

print("F1- Score     : ", f1_xg)
```

o/p: Accuracy Score: 0.9481409001956947
F1- Score : 0.15873015873015872

```
print(confusion_matrix(y_test, y_pred_xg))
```

o/p: [[964 8]
 [45 5]]

```
print(classification_report(y_test, y_pred_xg))
```

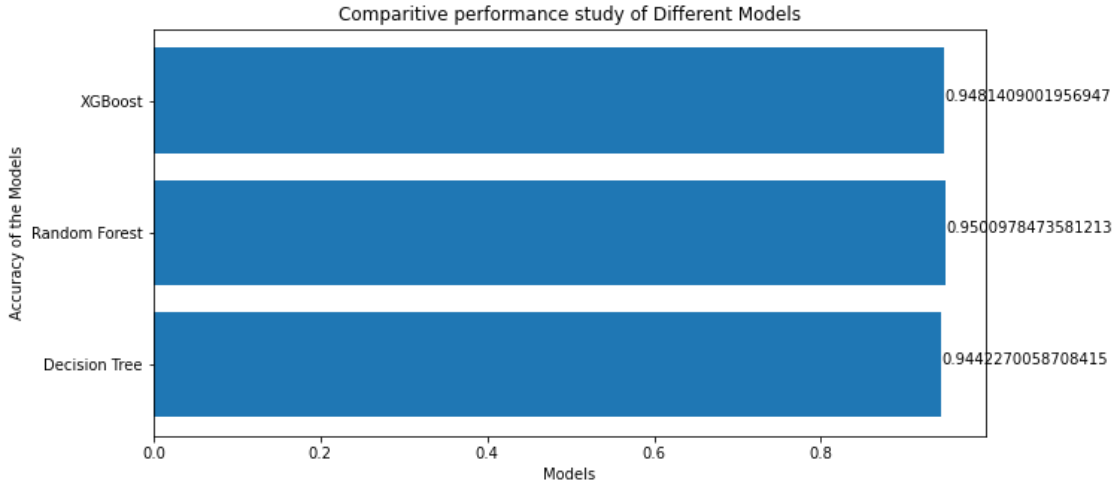
o/p:

	precision	recall	f1-score	support
0	0.96	0.99	0.97	972
1	0.38	0.10	0.16	50
accuracy			0.95	1022
macro avg	0.67	0.55	0.57	1022
weighted avg	0.93	0.95	0.93	1022

```
data = {
   'Decision Tree':accuracy_score(y_test, y_pred_dtc),
   'Random Forest':accuracy_score(y_test, y_pred_rfc),
```

```
    'XGBoost':accuracy_score(y_test, y_pred_xg),
}
activation = list(data.keys())
accuracy = list(data.values())


fig = plt.figure(figsize=(10,5))
plt.barh(activation, accuracy)
plt.xlabel("Models")
plt.ylabel("Accuracy of the Models")
plt.title("Comparitive performance study of Different Models")
for i, v in enumerate(accuracy):
  plt.text(v,i, str(v))
plt.show()
```

o/p:



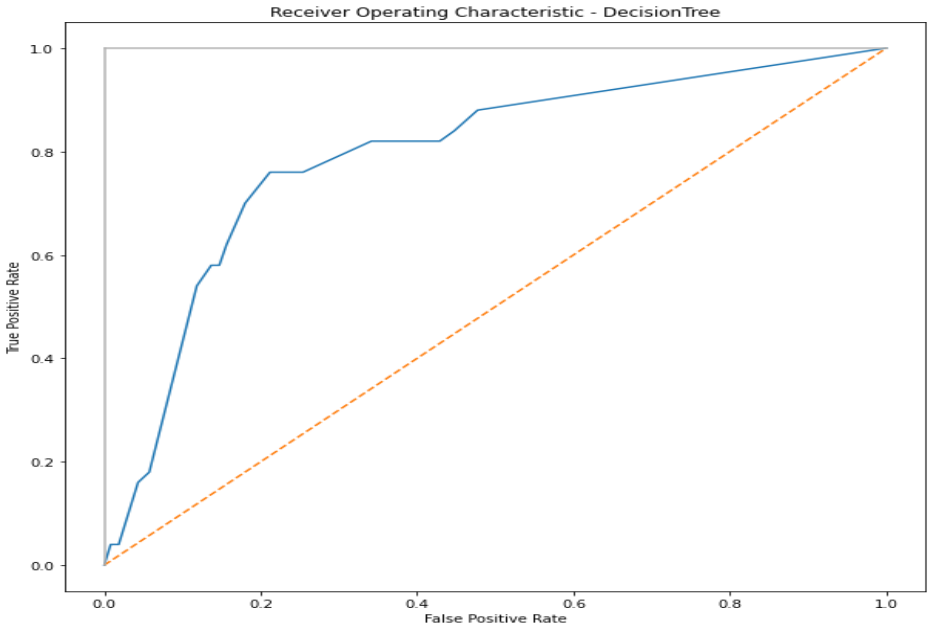
#ROC Curve

```
y_score1 = dtc.predict_proba(X_test)[:,1]
y_score2 = rfc.predict_proba(X_test)[:,1]
y_score3 = xg.predict_proba(X_test)[:,1]
from sklearn.metrics import roc_curve, roc_auc_score
false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(y_test, y_score1)
```

```
false_positive_rate2, true_positive_rate2, threshold2 = roc_curve(y_test, y_score2)

false_positive_rate3, true_positive_rate3, threshold3 = roc_curve(y_test, y_score3)
print('roc_auc_score for DecisionTree : ', roc_auc_score(y_test, y_score1))

print('roc_auc_score for Random Forest: ', roc_auc_score(y_test, y_score2))

print('roc_auc_score for XGBoost      : ', roc_auc_score(y_test, y_score3))
```

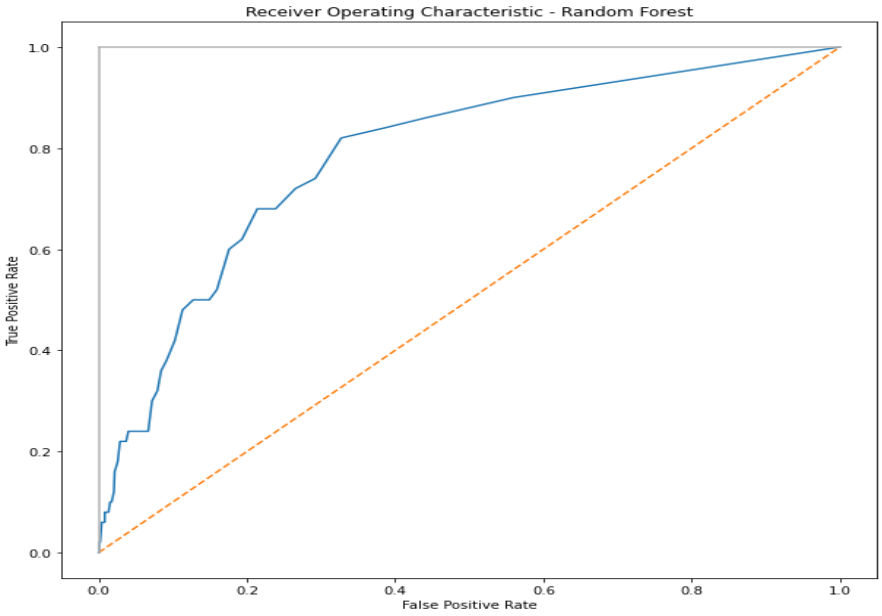
o/p:

```
roc_auc_score for DecisionTree :  0.7932921810699588
roc_auc_score for Random Forest:  0.7861008230452674
roc_auc_score for XGBoost      :  0.8058230452674897
plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - DecisionTree')

plt.plot(false_positive_rate1, true_positive_rate1)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()
```

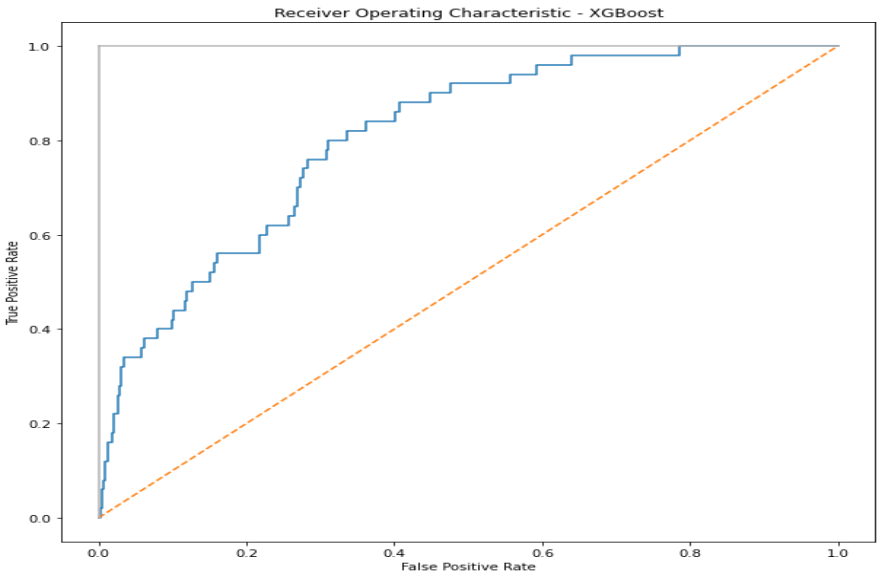



```
plt.subplots(1, figsize=(10,10))
```

```
plt.title('Receiver Operating Characteristic - Random Forest')
plt.plot(false_positive_rate2, true_positive_rate2)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```




```
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - XGBoost')
plt.plot(false_positive_rate3, true_positive_rate3)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



#SGB Optimizer

#ANN –Relu

```
import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Activation, Dense, Flatten, Conv2D, MaxPool2D,
Dropout,Input

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.metrics import categorical_crossentropy

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import confusion_matrix

model = Sequential()

model.add(Dense(units=256, input_shape=(x.shape[1], ), activation='relu'))

model.add(Dense(units=128, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(units=64, activation='relu'))

model.add(Dense(units=32, activation='relu'))

model.add(Dropout(0.3))
```

```
model.add(Dense(units=16, activation='relu'))

model.add(Dense(units=1, activation='sigmoid'))

model.compile(optimizer="adam",loss='binary_crossentropy', metrics=['accuracy'])

hist = model.fit(X_train, y_train, epochs=30, verbose=1, batch_size=32, validation_split=0.2,
callbacks=[early_stop])
```

o/p:
```
Epoch 30/30
103/103 [==============================] - 0s 5ms/step - loss: 0.1632 - accuracy: 0.95
35 - val_loss: 0.1885 - val_accuracy: 0.9425
```

```
loss, accuracy = model.evaluate(X_test, y_test)

print('Test accuracy is: ', accuracy)

print("Test loss is    : ", loss)
```

o/p:
```
32/32 [==============================] - 0s 2ms/step - loss: 0.1764 - accuracy: 0.9511
Test accuracy is:  0.951076328754425
Test loss is    :  0.1763533502817154
```

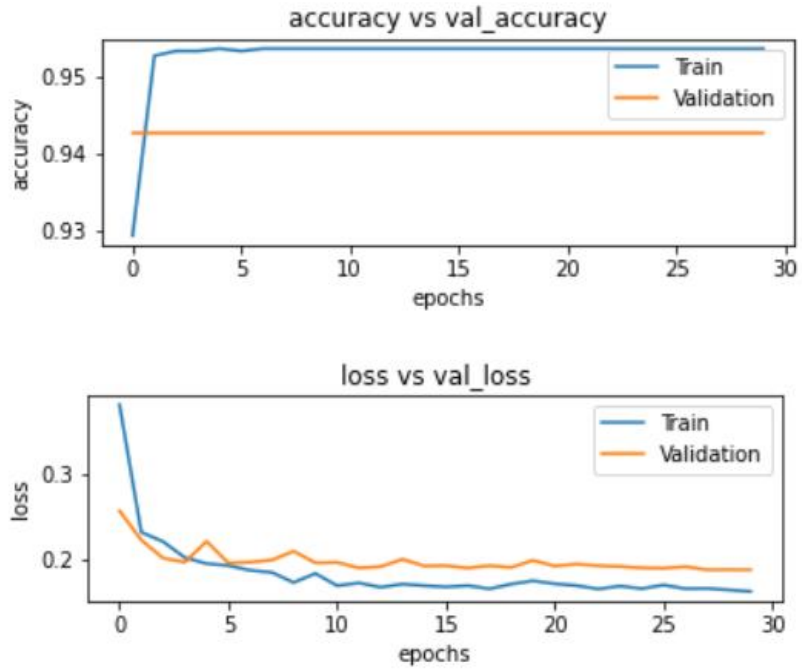
```
from sklearn.metrics import classification_report, confusion_matrix, f1_score

pred = model.predict(X_test)

pred = pred.argmax(axis=1)

cm = confusion_matrix(y_test, pred)

print(cm)
```

o/p:
```
32/32 [==============================] - 0s 2ms/step
  [[972   0]
   [ 50   0]]
```

```
from sklearn.metrics import accuracy_score

accuracy_relu = accuracy_score(y_test, pred)

print("F1 Score      : ", f1_score(y_test, pred, average='macro'))

print('Accuracy Score: ', accuracy_relu)
```

o/p:
```
F1 Score      :  0.48746238716148443
Accuracy Score:  0.9510763209393346
```

```
plt.subplot(211)
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('accuracy vs val_accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend(['Train', 'Validation'])
plt.show()
plt.subplot(212)
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('loss vs val_loss')
plt.xlabel('epochs') plt.ylabel('loss')
plt.legend(['Train', 'Validation'])
plt.show()
```



CHAPTER 6
CONCLUSION

We have used various machine learning techniques, such as data preprocessing, feature selection, and model building and were able to develop a predictive model that accurately predicts stroke risk based on several key features.

The project involved performing extensive EDA on the dataset, which helped us to identify the most important features for predicting stroke risk. By analysing the distribution of the data and performing correlation analysis, we were able to select the most relevant features for our model. We also addressed issues such as missing data and imbalanced classes, which are common challenges in machine learning projects.

Using the XGBoost boosting technique, we achieved the best accuracy of 95%. As a result, we could conclude that the XGBoost algorithm is the best model for predicting strokes.

Stroke prediction is an important aspect of healthcare that aims to identify individuals who are at risk of having a stroke in the future. Healthcare professionals use various risk factors to calculate an individual's stroke risk score and determine appropriate interventions to prevent or mitigate the effects of a stroke using machine learning algorithms to predict the likelihood of stroke based on various factors.