1. INTRODUCTION

1.1.Overview

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. In recent times, Heart Stroke prediction is one of the most complicated tasks in the medical field. In the modern era, approximately one person dies per minute due to heart Stroke. Data science plays a crucial role in processing huge amount of data in the field of healthcare. Machine learning has become one of the most demanding field in modern technology. It is a form of artificial intelligence where the model can analyze the data, identify patterns and predict the outcome with minimal human intervention. Heart stroke prediction in adults can be done by using various machine learning algorithms. It has become an intrigued research problem as there are various factors or parameters that can influence the outcome. The factors include work type, gender, residence type, age, average glucose level, body mass index, smoking status of the individual and any previous heart disease.
As heart stroke prediction is a complex task, there is a need to automate the prediction process to avoid risks associated with it and alert the patient well in advance.

1.2.Purpose

The project aims to predict the chances of Heart Stroke and classifies the patient's risk level by implementing different Machine Learning techniques such as KNN, Decision Tree and Random Forest. From these models the Best performing model is selected and saved. Here we will be building a flask application that uses a machine learning model to get the prediction of heart stroke. We will also train our model on IBM Cloud and deployment on IBM Cloud .

2. LITERATURE SURVEY

2.1.Existing problem

In, heart disease prediction is done using NaÃ¯ve Bayes and Genetic algorithms. The model has been trained on a UCI dataset with attributes like gender, age, resting blood pressure, cholesterol, fasting blood sugar, old peak, etc. It is a web-based machine learning application where the user inputs his medical details based on these attributes to predict his heart disease. The algorithm calculates the probability of having a heart disease and the result is displayed on the web page itself.

In various classification algorithms are studied and the most accurate model is obtained for predicting the heart disease in the patient. It was found that Random Forest and XGBoost were the most efficient algorithms while K- Nearest Neighbor was found to be the most ineffective one.In a novel heart attack prediction mechanism is proposed mainly using Decision Tree Classifier algorithm. The model first learns the deep features based

on the attributes provided in the dataset andthen trains on the learned features to obtain the outcome or prediction.

In a survey is proposed on the various machine learning algorithms that could be used for the heart disease prediction. The authors have summarized the various algorithms and then worked towards finding the best algorithm by analysing the various features.In, heart disease prediction has been performed using the four algorithms- Logistic Regression, NaÃ¯ve Bayes, Random Forest and Decision Tree. The objective is to effectively study whether the patient has any heart disease. The health professional enters the input values from the patients health report. The data is then fed into the machine learning model which provides the probability of having the heart disease.

In, heart stroke prediction is analysed using various machine learning algorithms and the Receiver Operating Curve (ROC) is obtained for each algorithm. It has been implemented using Apache Spark and it shows that the Gradient Boosting Algorithm gives the highest ROC score of 0.90. The analysis of the features has been done by using univariate and multivariate plots to obtain the correlation between the several features.

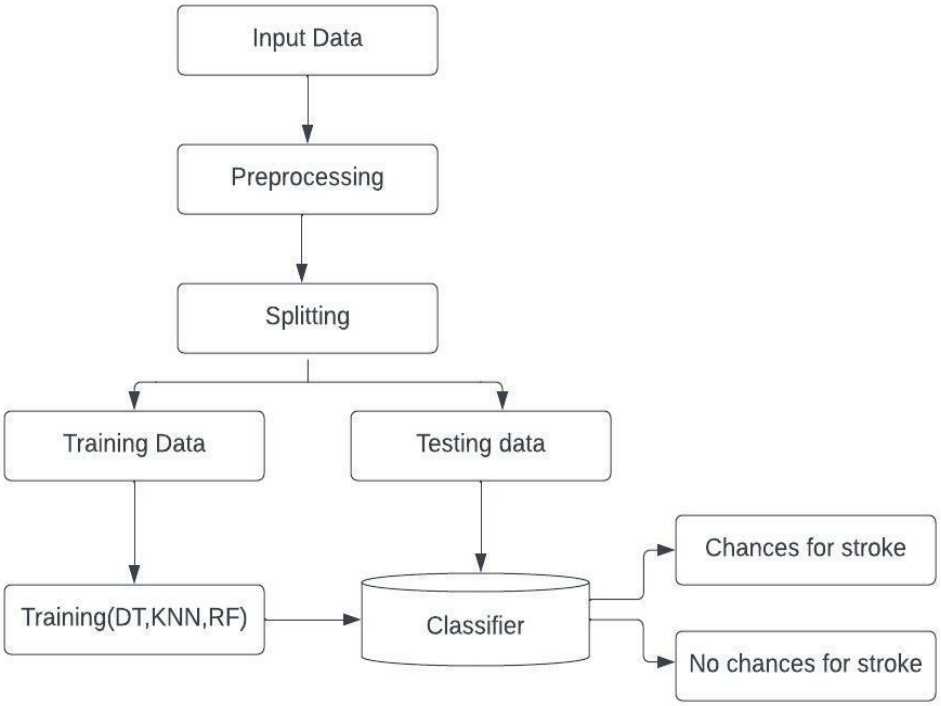
2.2 Proposed solution

Model is proposed to predict whether the individual will have heart stroke or not based on several input parameters like age, gender, smoking status, work type, etc. The dataset is trained on various machine learning algorithms and their performance is analysed to find out which one would be the best to effectively predict heart stroke. The accuracies obtained from each algorithm is plotted to show the comparative analysis of each algorithm.

First data collection is done followed by data pre-processing to obtain a cleaned dataset with no null values or duplicate values for better training and higher accuracy. Then data visualization is performed which gives a clear idea about the dataset through the visualization graphs and makes it easier to identify the patterns, trends and outliers. After this, the dataset is split into training and testing datasets and fed into the various classification models to obtain the prediction. The confusion matrix along with the accuracies of the models are obtained to find out the most effective algorithm that could be used for the prediction. The model has also been trained using a custom input value to check for the accuracy.

3. THEORITICAL ANALYSIS

3.1.Block diagram



3.2 HADWARE / SOFTWAREDESINING

- **Tools**

For application development, the following Software Requirements are:

1. **Operating System:** Windows 7 and above
2. **Language**:Python, Html
3. **Tools**: Microsoft Excel (Optional).
4. **Technologies used**:Flask application

- **Software requirements**

1. **Operating System**:Any OS with clients to access the internet
2. **Network**:Wi-Fi Internet or cellular Network
3. **Visio Studio**:Create and design Data Flow and Block Diagram

4. **GitHub**:Versioning Control
5. **Google Chrome**:Medium to display and run the html code

- **Hardware Requirements**

For application development, the following Software Requirements are:

1. **Processor**: Intel or high.
2. **RAM**: 1024 MB.
3. **Space on disk**: minimum 100mb.
4. **For running the application**: Device: Any device that can access the internet.
5. **Minimum space to execute**: 20 MB

4. EXPERIMENTAL INVESTIGATIONS

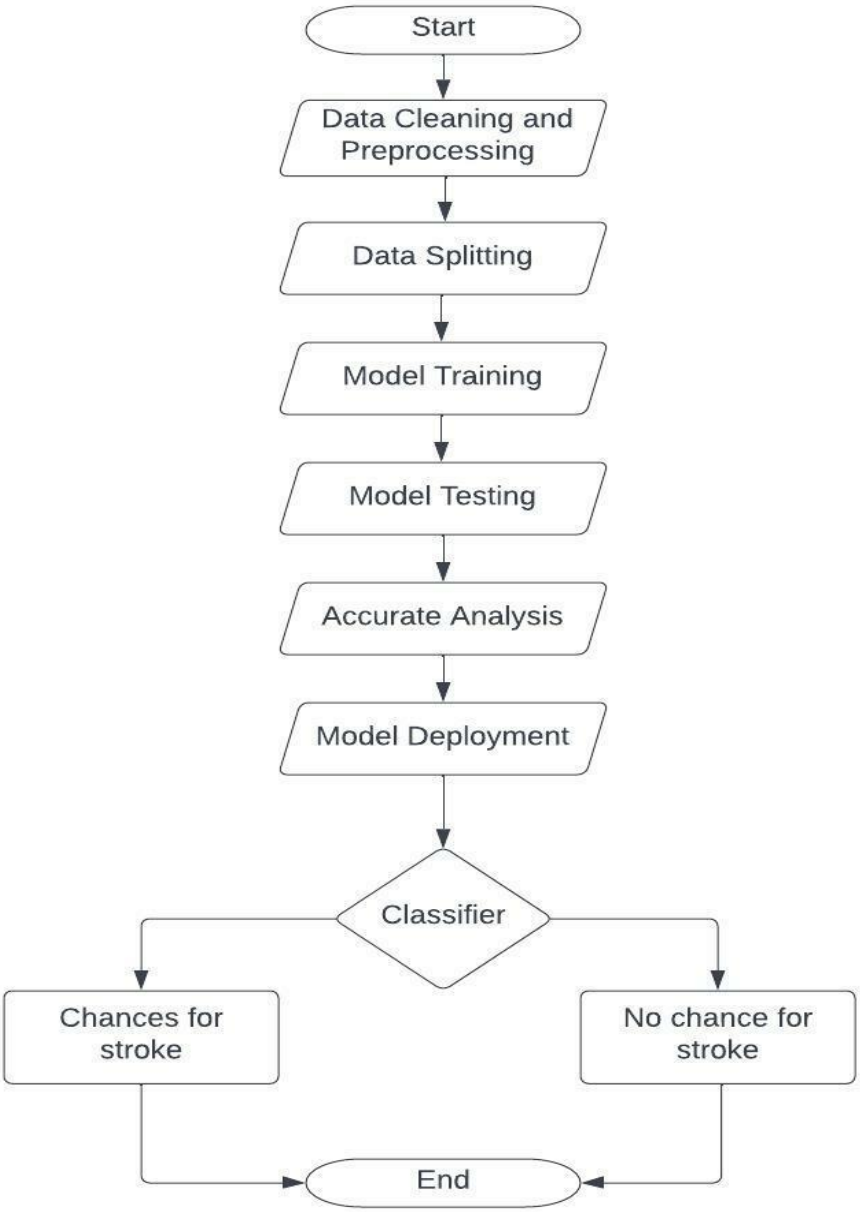
Survey work is done with respect to the project, which includes the burden of stroke, causes of stroke and subtypes, causes of mortality from stroke and risk factors of stroke. Finally, it concludes by discussing various aspects for prevention of stroke. The survey work also finds out some of the major flaws associated with all these related topics, so that a suitable solution can be proposed in order to overcome these drawbacks.

The proposed solution is based on the factors of a person like Gender, Age, Hypertension, Heart disease, ever married, work type, residence, average glucose level, BIM, smoking status, which classifies that particular person into a specific category which may be either one of the following :

1. he/she may have chance of stroke
2. he/she may not have chance of stroke

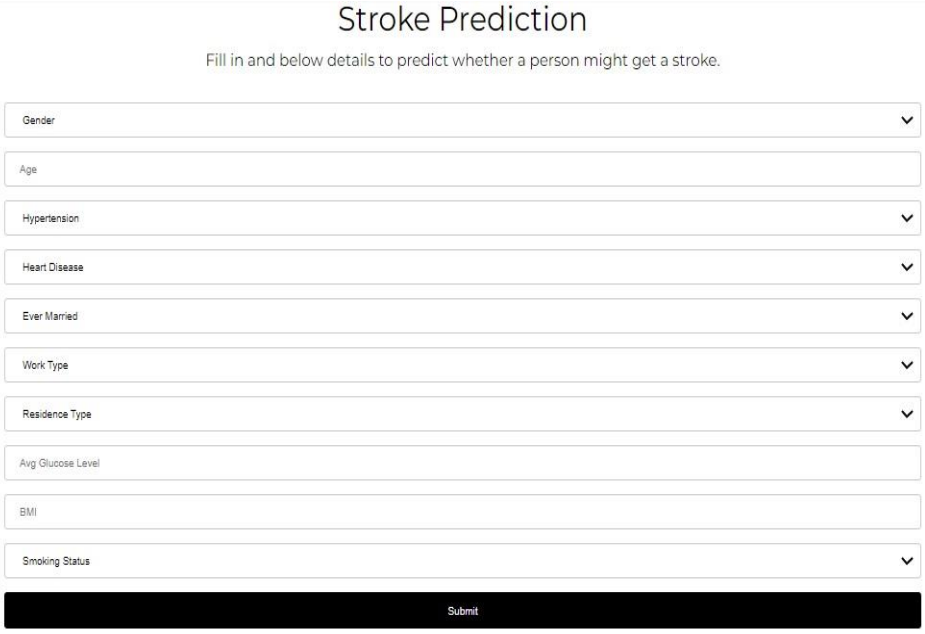
Also if the person falls into the 1st category mentioned above then that person will get alerted as early as possible as that it will help to reduce the mortality rate due to heart stroke .

5. Flow Chart


Start
Data Cleaning and Preprocessing
Data Splitting
Model Training
Model Testing
Accurate Analysis
Model Deployment
Classifier
Chances for stroke
No chance for stroke
End

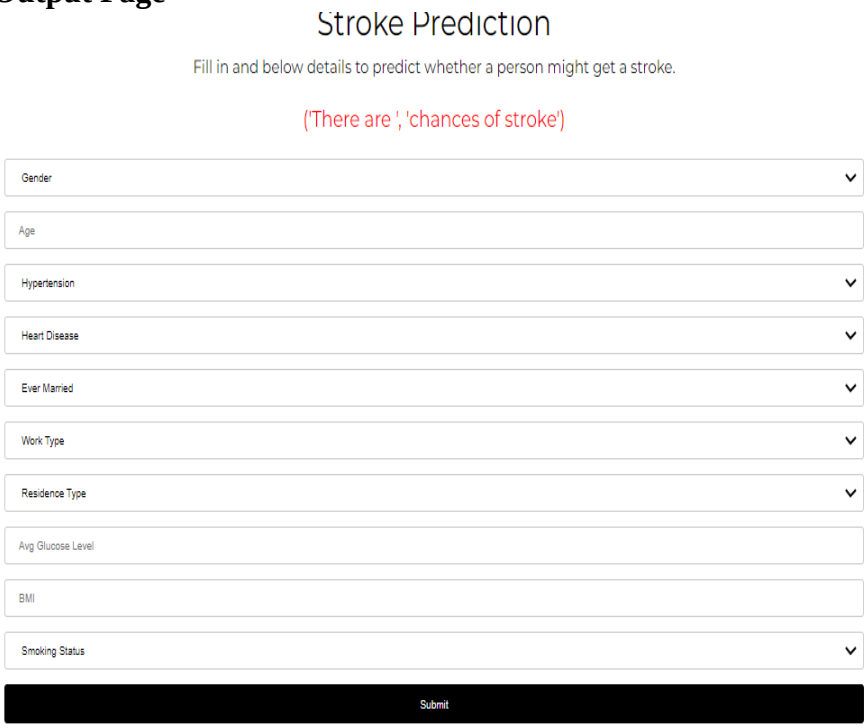
6. RESULT

6.1Input Page



All the 10 fields as shown in above picture are independent variables used to predict the stroke.

6.2 Output Page



After giving all the inputs the predictions is shown in red color in the picture above which says that “There are chances of stroke”.

7. ADVANTAGES & DISADVANTAGES

❖ Advantages

- Increased accuracy for effective heart disease diagnosis.
- Handles roughest(enormous) amount of data using random forest algorithm and feature selection.
- Reduce the time complexity of doctors.
- Cost effective for patients
- The project is that integration of clinical decision support with computer-based patient records could reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcome.
- This suggestion is promising as data modelling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions.

❖ Disadvantages

- Prediction of cardiovascular disease results is not accurate.
- Data mining techniques does not help to provide effective decision making.
- Cannot handle enormous datasets for patient records.

8. APPLICATIONS

In stroke, machine learning techniques are increasingly used in various areas including outcome prediction after endovascular treatment.With consideration of its expected impact on ischemic stroke management, we developed models using machine learning techniques to predict long-term stroke outcomes.
We then compared the predictability to the Acute Stroke Registry and Analysis of Lausanne (ASTRAL) score, which isa well-known prognostic mode.
Machine learning models were trained with all variables as inputs to classify patients likely to have favourable outcomes.
The proposed working model can also help in reducing treatment costs by providing Initial diagnostics in time. The model can also serve the purpose of training tool for medical students and will be a soft diagnostic tool available for physician and cardiologist.

9. CONCLUSION

As heart diseases and strokes are increasing rapidly across the world and causing deaths, it becomes necessary to develop an efficient system that would predict the heart stroke effectively before hand so that immediate medical attention can be given.This project hence helps to predict the stroke risk using prediction model and provide personalized warning and the lifestyle correction message through a web

application. By doing so, it urges medical users to strengthen the motivation of health management and induce changes in their health behaviours. In the proposed system, the most effective algorithm for stroke prediction was obtained after comparative analysis of the accuracy scores of various models like Decision tree ,KNN,Random forest, Logical Regression models, etc. The most effective among these models wasRandom forest with accuracy score of 100%.

10. FUTURE SCOPE

The project can be further enhanced by deploying the machine learning model obtained using a web application and a larger dataset could be used for prediction to give higher accuracy and produce better results.
This project helps to predict the stroke risk using prediction model in older people and for people who are addicted to the risk factors as mentioned in the project. In future, the same project can be extended to give the stroke percentage using the output of current project. This project can also be used to find the stroke probabilities in young people and underage people by collecting respective risk factor information's and doctors consulting.

11. BIBILOGRAPHY

1. PR3223 - Prediction_of_Stroke_Report - SRIKANTH S.pdf
2. https://www.researchgate.net/publication/342437236_Prediction_of_Stroke_Using_Machine_Learning
3. https://www.ijert.org/research/comparative-analysis-and-implementation-of-heart-stroke-prediction-using-various-machine-learning-techniques-IJERTV10IS060406.
4. Medical software user interfaces, stroke MD application design (IEEE)" - Elena Zamsa

12. APPENDIX

❖ **Source code of python**

```
pip install -U scikit-learn==0.23
import sklearn
sklearn.__version__
# Ignore  the warnings
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

# datavisualisation and manipulation
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
#importpandas_profiling as pdp

#configure
# sets matplotlib to inline and displays graphs below the corressponding cell.
%matplotlib inline
style.use('fivethirtyeight')
sns.set(style='whitegrid',color_codes=True)

#import the necessary modelling algos.

#classifiaction.
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC,SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier,AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb
import lightgbm as lgbm
#importcatboost as cb
from sklearn.ensemble import AdaBoostClassifier
#model selection
from sklearn.model_selection import train_test_split,cross_validate
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
#preprocessing
from sklearn.preprocessing import MinMaxScaler,StandardScaler,LabelEncoder
#evaluation metrics
from sklearn.metrics import mean_squared_log_error,mean_squared_error, r2_score,mean_absolute_error # for regression
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score # for classification
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score

import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
```

```
def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_53ebe756cad443f3b91684cd45c8e830 = ibm_boto3.client(service_name='s3',
ibm_api_key_id='wfzSUNnmMx3Y-eve_qc_rAPCQIXhIkpRUM_I75siwPH6',
ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
config=Config(signature_version='oauth'),
endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

body = client_53ebe756cad443f3b91684cd45c8e830.get_object(Bucket='heartstrokeprediction-donotdelete-pr-8le368jxk4qprq',Key='healthcare-dataset-stroke-data.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()

for i in ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']:
print(df[i].unique())

sns.barplot(x = 'stroke', y = 'smoking_status', data = df)

replace_values = {'Unknown': 'never smoked'}
df = df.replace({'smoking_status': replace_values})
df.head()

fig, axes = plt.subplots(nrows = 5, ncols = 2, figsize = (12, 20))
sns.boxplot(x = 'avg_glucose_level', data = df, ax=axes[0][0])
sns.boxplot(x = 'age', data = df, ax=axes[1][0])
sns.boxplot(x = 'bmi', data = df, ax=axes[2][0])
sns.boxplot(x = 'smoking_status', y = 'age', hue = 'stroke', data = df, ax=axes[3][0])
sns.boxplot(x = 'hypertension',y = 'age', hue = 'stroke', data = df, ax=axes[3][1])
sns.boxplot(x = 'heart_disease', y= 'age', hue = 'stroke', data = df, ax=axes[0][1])
sns.boxplot(x = 'ever_married',y = 'age', hue = 'stroke', data = df, ax = axes[1][1])
sns.boxplot(x = 'work_type',y = 'age',hue = 'stroke', data = df, ax = axes[2][1])
sns.boxplot(x = 'Residence_type',y = 'age',hue = 'stroke', data = df, ax = axes[4][0])
sns.boxplot(x = 'smoking_status',y = 'age', hue = 'stroke', data = df, ax = axes[4][1])

#Now, we will show the correlation between the parameters with a heatmap:
plt.figure(figsize = (18, 9))
sns.heatmap(df.corr(), annot = True)
```

```
plt.show()

df.describe()

def remove_outliers(data):
arr=[]
#print(max(list(data)))
q1=np.percentile(data,25)
q3=np.percentile(data,75)
iqr=q3-q1
mi=q1-(1.5*iqr)
ma=q3+(1.5*iqr)
#print(mi,ma)
for i in list(data):
if i<mi:
i=mi
arr.append(i)
elifi>ma:
i=ma
arr.append(i)
else:
arr.append(i)
#print(max(arr))
return arr

df['bmi'] = remove_outliers(df['bmi'])
df['avg_glucose_level'] = remove_outliers(df['avg_glucose_level'])
print('Outliers successfully removed')

fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (10, 5))
sns.boxplot(x = 'bmi', data = df, ax=axes[1][0])
sns.boxplot(x = 'avg_glucose_level', data = df, ax=axes[0][0])

for i in ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']:
print(df[i].unique())
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Residence_type'] = le.fit_transform(df['Residence_type'])
df['ever_married'] = le.fit_transform(df['ever_married'])
df['gender'] = le.fit_transform(df['gender'])
df['work_type'] = le.fit_transform(df['work_type'])
df['smoking_status'] = le.fit_transform(df['smoking_status'])

X = df.iloc[:,0:10].values
y = df.iloc[:,10].values
```

```
from sklearn.preprocessing import OneHotEncoder
one = OneHotEncoder()
a = one.fit_transform(X[:,0:1]).toarray()
b = one.fit_transform(X[:,5:6]).toarray()
c = one.fit_transform(X[:,9:10]).toarray()
X = np.delete(X,[0,5,9],axis = 1)

sns.countplot(y)
#!pip install imblearn
'"from imblearn.over_sampling import SMOTE

sm = SMOTE()
X_res, y_res = sm.fit_resample(X, y)

print("Before OverSampling, counts of label '1': {}".format(sum(y==1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y==0)))

print('After OverSampling, the shape of train_X: {}'.format(X_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_res==1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_res==0)))'"
sns.countplot(y_res)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
42)
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
dtc_pred = dtc.predict(X_test)
print('***Decision Tree Model Results***')
print(confusion_matrix(dtc_pred, y_test))
print(classification_report(dtc_pred, y_test))

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
print('** Random Forest**')
print(confusion_matrix(rf_pred, y_test))
print(classification_report(rf_pred, y_test))

lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
print(confusion_matrix(lr_pred, y_test))
print('** Logistic Regression**')
```

```
print(classification_report(lr_pred, y_test))

svc = SVC()
svc.fit(X_train, y_train)
svc_pred = svc.predict(X_test)
print('**Support vector classifier**')
print(confusion_matrix(svc_pred, y_test))
print(classification_report(svc_pred, y_test))

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
print('**K-Nearest Neighbours**')
print(confusion_matrix(knn_pred, y_test))
print(classification_report(knn_pred, y_test))

%%time
parameters = {
"max_depth": [3, 5, 7, 9, 11, 13],
}
model_dtc = DecisionTreeClassifier(
random_state=42,
class_weight='balanced',
)
model_dtc = GridSearchCV(
model_dtc,
parameters,
cv=5,
)
model_dtc.fit(X_train, y_train)
model_dtc_pred = model_dtc.predict(X_test)
print('** Applying Grid Search to Decision tree**')
print(classification_report(model_dtc_pred, y_test))

print(f'Best parameters {model_dtc.best_params_}')
print(
f'Mean cross-validated accuracy score of the best_estimator: ' + \
f'{model_dtc.best_score_:.3f}'
)
cross_valid_scores['desicion_tree'] = model_dtc.best_score_
print('-----')

%%time
parameters = {
"n_estimators": [5, 10, 15, 20, 25],
"max_depth": [3, 5, 7, 9, 11, 13],
```

```
}
model_rf = RandomForestClassifier(
random_state=42,
class_weight='balanced',
)
model_rf = GridSearchCV(
model_rf,
parameters,
cv=5,
)
model_rf.fit(X_train, y_train)
model_rf_pred = model_rf.predict(X_test)
print('** Applying Grid Search to Random Forest**')
print(classification_report(model_rf_pred, y_test))
print(f'Best parameters {model_rf.best_params_}')
print(
f'Mean cross-validated accuracy score of the best_estimator: '+ \
f'{model_rf.best_score_:.3f}'
)
cross_valid_scores['random_forest'] = model_rf.best_score_

import joblib
joblib.dump(model_rf,"model")
```

- **Flask Application coding**

```
from flask import Flask, request,render_template
import joblib
import numpy as np
import requests


app = Flask(__name__)

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "Z3uWFvZLMkCxrP7McP2otDti2HgOYk-CoiQ1YbhqbIwi"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

@app.route('/')
def predict():
return render_template('Manual_predict.html')
```

```
@app.route('/y_predict',methods=['POST'])
def y_predict():
g= request.form["Gender"]
if (g == 'f'):
g1,g2,g3=1,0,0
if (g == 'm'):
g1,g2,g3=0,1,0
if (g == 'o'):
g1,g2,g3=0,0,1
age= request.form["Age"]
q= request.form["Hypertension"]
if (q == 'n'):
q=0
if (q == 'y'):
q=1
hd= request.form["heart_disease"]
if (hd == 'n'):
hd=0
if (hd == 'y'):
hd=1
em= request.form["ever_married"]
if (em == 'n'):
em=0
if (em == 'y'):
em=1
wt= request.form["work_type"]
if (wt == 'ch'):
wt1,wt2,wt3,wt4,wt5 = 1,0,0,0,0
if (wt == 'gvt'):
wt1,wt2,wt3,wt4,wt5 = 0,1,0,0,0
if (wt == 'unemp'):
wt1,wt2,wt3,wt4,wt5 = 0,0,1,0,0
if (wt == 'pvt'):
wt1,wt2,wt3,wt4,wt5 = 0,0,0,1,0
if (wt == 'self'):
wt1,wt2,wt3,wt4,wt5 = 0,0,0,0,1
rt= request.form["Residence_type"]
if (rt == 'rural'):
rt=0
if (rt == 'urban'):
rt=1
agl= request.form["avg_glucose_level"]
bmi= request.form["BMI"]
sm= request.form["smoking_status"]
if (sm == 'for'):
sm1,sm2,sm3=1,0,0
if (sm == 'ns'):
sm1,sm2,sm3=0,1,0
if (sm == 's'):
```

```
sm1,sm2,sm3=0,0,1

inp= np.array([g1,g2,g3,wt1,wt2,wt3,wt4,wt5,sm1,sm2,sm3,age,q,hd,em,rt,agl,bmi])
print(inp)

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"fields": [["g1","g2","g3","wt1","wt2","wt3","wt4","wt5","sm1","sm2","sm3","age","q","hd","em","rt","agl","bmi"]], "values": [[0,1,0,0,0,1,0,0,1,0,0,67,0,1,1,1,168.32,36.6]]}]}

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/566bf3ca-c249-4e0e-a528-c26461ae2b8a/predictions?version=2022-03-05', json=payload_scoring, headers={'Authorization': 'Bearer ' + mltoken})
#print("Scoring response")
print(response_scoring.json())
op = response_scoring.json()
pred= op["predictions"][0]['values'][0][0]
print(pred)

#pred = model.predict([inp])
#print(pred)

if(pred==0):
result="no chances of stroke"
else:result="chances of stroke"

return render_template('Manual_predict.html', \
prediction_text=('There are \
',result))


if __name__ == '__main__':
app.run(host='0.0.0.0', debug=False)
```