BIRMINGHAM CITY University

CMP6202
AI and Machine Learning Project
2023–2024

Individual Report

STROKE PREDICTION

Forename	Surname	Student ID	Instructor Name	ML Model(s) developed
DHANANJAY	TEWARI	21150306	Dr Nouh Sabri Elmitwally	STROKE PREDICTION

Table of Contents

1 Report Introduction

Stroke, identified as the second leading cause of death worldwide by the World Health Organization, accounts for approximately 11% of total global mortality. Recognizing the critical impact of strokes on public health, this report focuses on leveraging a dataset to predict the likelihood of a patient experiencing a stroke. The dataset incorporates diverse patient information, allowing for the development of an AI model trained to estimate the probability of a heart stroke occurrence.

1.1 Dataset identification.

The Dataset that has been investigated in this report is the "Stroke prediction dataset" collected form Kaggle by fedesoriano. This dataset has 11 attributes which contributes in the cause of the stroke.

These attributes are as follows:

ID: *67 to 72940 (Numerical)*
Defines a user represented by a numerical value as the unique identifier (ID).

GENDER: *Male, Female or Other (Categorical)*
Defines if the user in question is a Male, Female or they identify as another gender.

AGE: *0 to 100 (Numerical)*
The age of the user in question.

HYPERTENSION: *0 or 1 (Numerical)*
Defines if the user in question has hypertension or not (High Blood Pressure).

HEART DISEASE: *0 or 1 (Numerical)*
Defines if the user in question has heart disease or not.

WORK TYPE: *Private, Self-employed, Govt_job, Never_worked and children (Categorical)*
Defines the user in question's work type, if they're private, self-employed, government job or have never worked; if the user is a child, it will be defined as 'children'.

EVER MARRIED: *yes or no (Categorical)*
Defines if the user in question has ever been married or not.

RESIDENCE: *Urban or rural (Categorical)*
Defines if the user in question lives in an urban or a rural area.

AVG GLUCOSE LEVEL: *55 to 272 (Numerical)*
Defines the average glucose level of the user in question.

BODY MASS INDEX: *10 to 98 (Numerical)*
Defined the body mass index of the user in question.

SMOKING: *formally smoked, never smoked, smokes* and Unknown (Categorical)

This defines the status of the user in question's smoking activity, if they're currently smoking, formally, never snoke.

STROKE: *0 or 1 (Numerical)*
This is the target value; it defines if the user in question has had a stroke or not.

.

1.2 Supervised learning task identification

This model focuses on predicting the occurrence of strokes in individuals. It tackles a classification problem where the target variable indicates whether a stroke will occur or not. Out of the available variables, 10 are directly associated with the prediction of strokes. The model utilizes the data from these variables to make predictions regarding the likelihood of a stroke occurring.

2 Exploratory Data Analysis

2.1 Question(s) identification

- **Size of the Dataset:**

The dataset contains 5110 entries, indexed from 0 to 5109.

- **Types of Variables:**

It's crucial to identify the types of variables (e.g., numerical, categorical) as they significantly impact the predictive modelling process. The variable types influence the choice of preprocessing techniques and model selection.

- **Missing Values:**

Identifying and handling missing values is important to ensure accurate model training and predictions. Missing values can lead to unexpected errors or incorrect predictions.

- **Distribution of Variables:**

Understanding the distribution of data, especially in the target variables, is critical. It helps determine if the dataset is balanced or imbalanced. Imbalanced data might require techniques like

resampling (oversampling/under sampling) to address class imbalance for better model performance.

- **Handling Missing Values:**

Determining the best strategy for handling missing values is essential. Options include imputation (replacing missing values with a statistic like mean or median), deletion of rows/columns with missing values, or using algorithms that handle missing data effectively.

This structured approach ensures a comprehensive understanding of the dataset's characteristics, enabling informed decisions on preprocessing, handling missing values, and selecting appropriate modelling techniques for optimal predictive performance.

2.2 Splitting the dataset

The dataset exhibits a significant imbalance toward the '0' class of the classifier. This imbalance poses a challenge when performing a simple train-test split, potentially leading to a scarcity of minority class instances in the testing data. To address this issue, a Stratified Train-Test Splitting approach is employed, ensuring a balanced representation of classes in both the training and testing sets.

Stratified Train-Test Splitting involves partitioning the dataset into training and testing subsets while maintaining the proportional representation of classes in both sets. This technique enhances the reliability of model evaluation by preserving the original class distribution, crucial in imbalanced datasets. The splitting ratio chosen is 75-25, enabling a substantial portion of the data to train the model while ensuring a representative testing subset. Alternatively, Stratified KFold splitting, dividing the data into five stratified folds, achieves a similar objective across multiple iterations, ensuring robustness in model evaluation.

2.3 Exploratory Data Analysis process and results

The dataset utilized for training and evaluating the model has been sourced from Kaggle. This dataset serves as the raw data that requires cleaning and preprocessing before initiating the machine learning training process. To facilitate this, a thorough data analysis is imperative. This analysis involves gaining a comprehensive understanding of the data, exploring the relationships between variables, and assessing how these factors may influence the model's performance.

2.3.1 Size, shape and features of the dataset:

Dataset is in the .csv format which is analysed in python using Pandas, Matplotlib, seaborn etc libraries.

The Dataset is fetched from the file and converted into pandas Dataframe. Using:

```
Stroke=pd.read_csv("/content/drive/MyDrive/healthcare-dataset-stroke-data.csv")
```

To check the size, shape and mean , median and mode of the features of the data set the following code is used:

```
Stroke.info()
Stroke.shape
Stroke.describe()
```

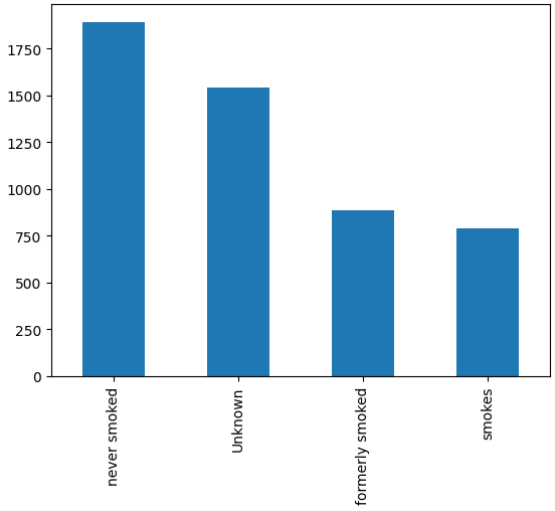
According the output the initial entries in the dataset was 5110.

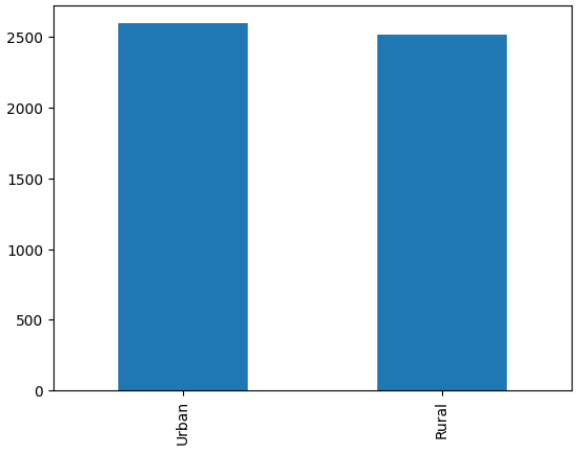
2.3.2 features analysis(individually):

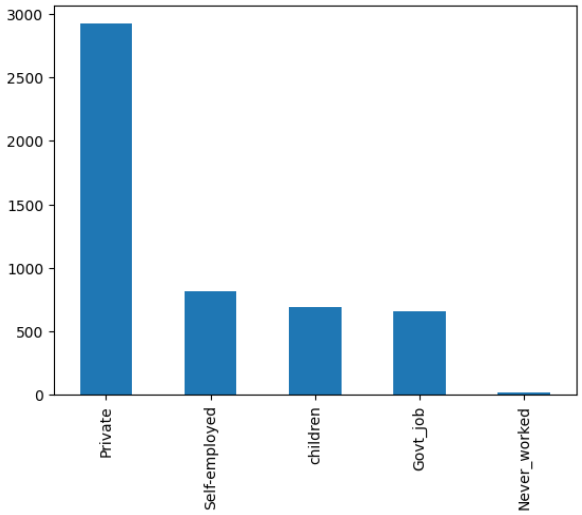
After getting the features names and dtype from the Dataset info. Features are analysed individually.

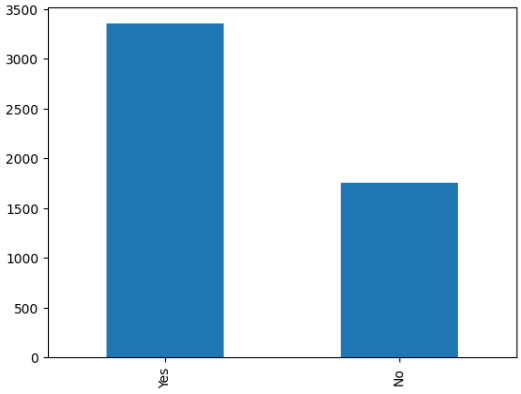
Some of the features which has string data type and has categorical values like smoking status, Residence type , worktype, ever married, gender are analysed with their count for each of their catagories and no. of catagories they have. This is done using bar plot.

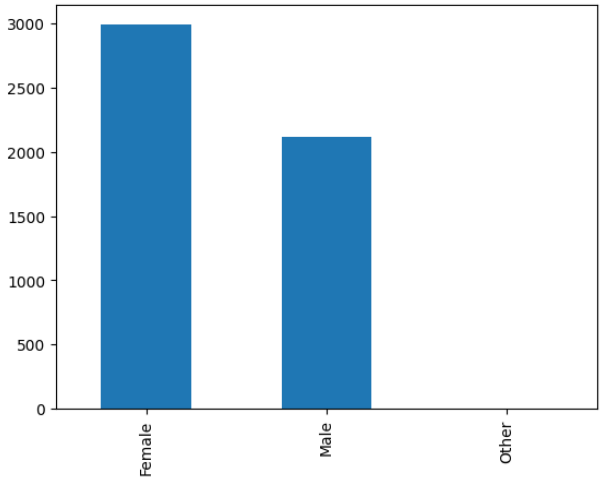
```
Stroke['smoking_status'].value_counts().plot.bar()
Stroke['Residence_type'].value_counts().plot.bar()
Stroke['work_type'].value_counts().plot.bar()
Stroke['ever_married'].value_counts().plot.bar()
Stroke['gender'].value_counts().plot.bar()
```







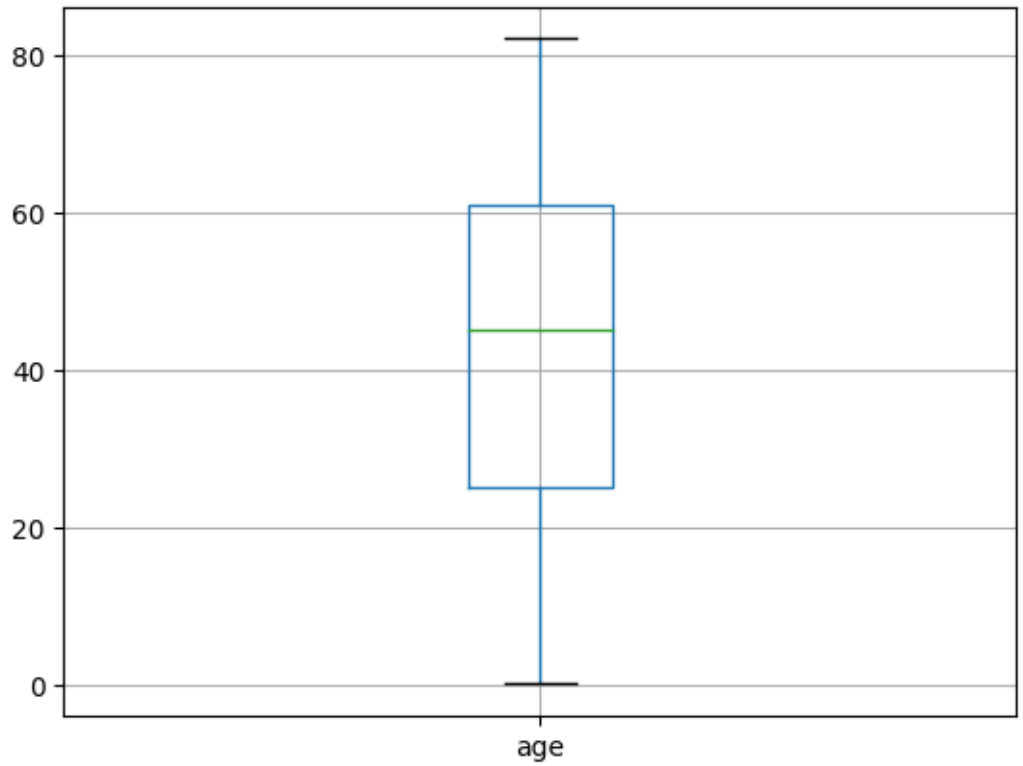


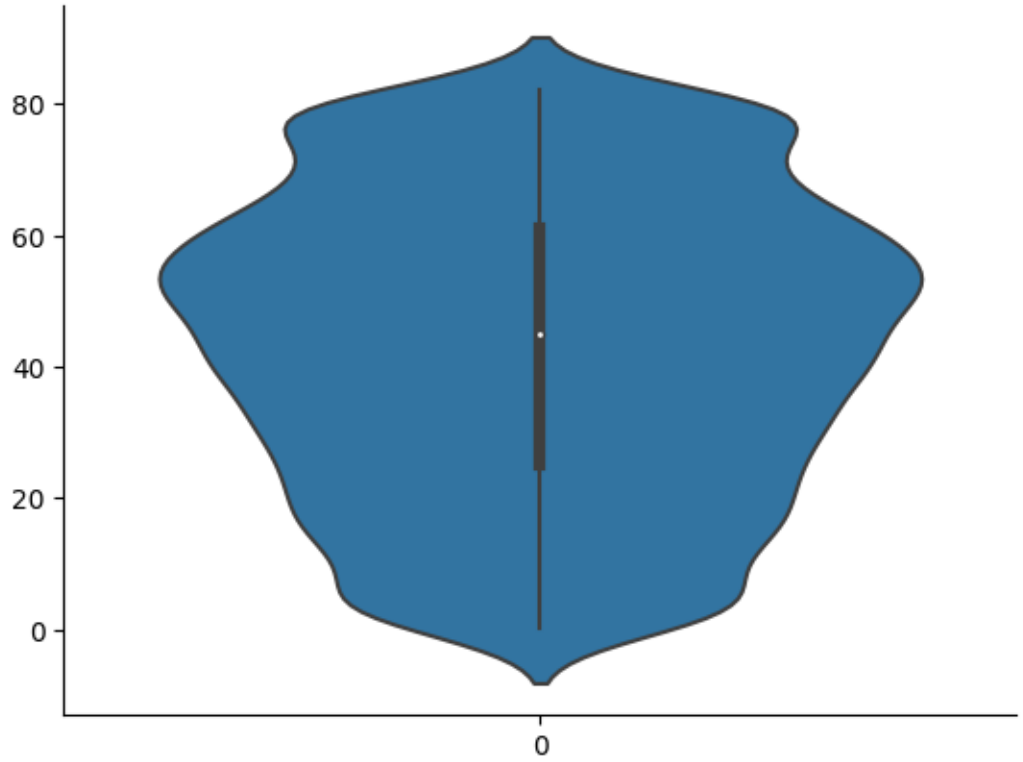


The feature which are numerical data is their analysis is done using the box plot and violinplot to get the outliers, distribution and density of the following.

```
plt.figure()
Stroke.boxplot(['age'])
sns.violinplot(Stroke['age'])
sns.despine()
```

This analysis gives us the info that the info of the range of the following features.

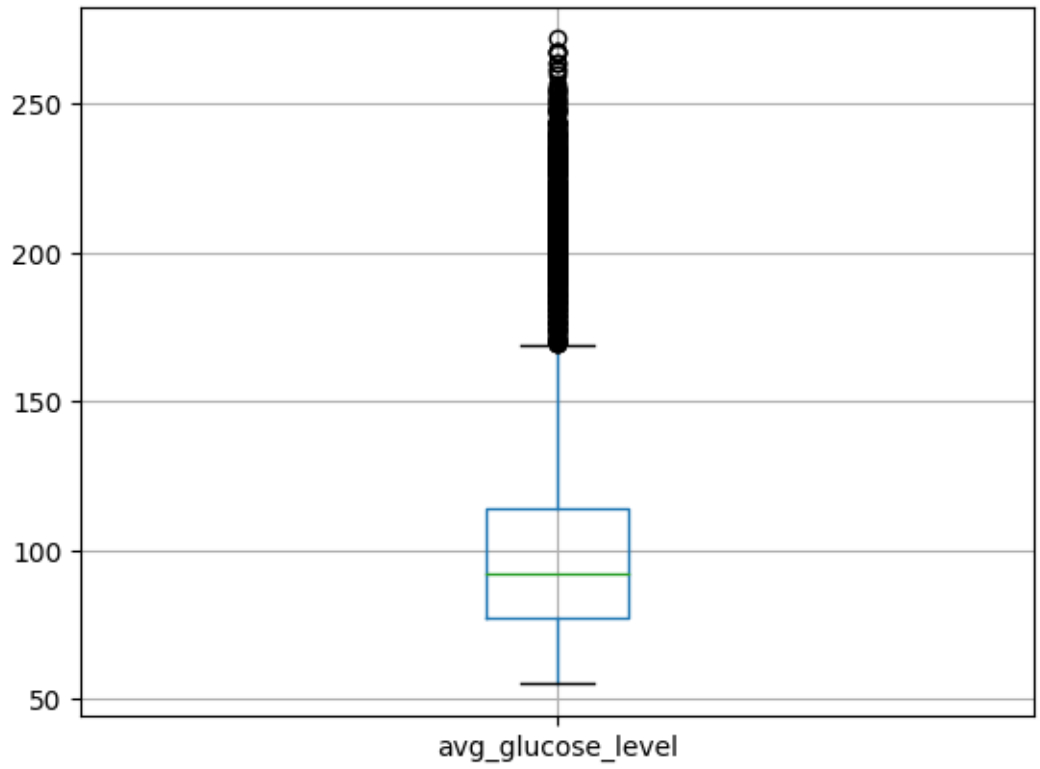


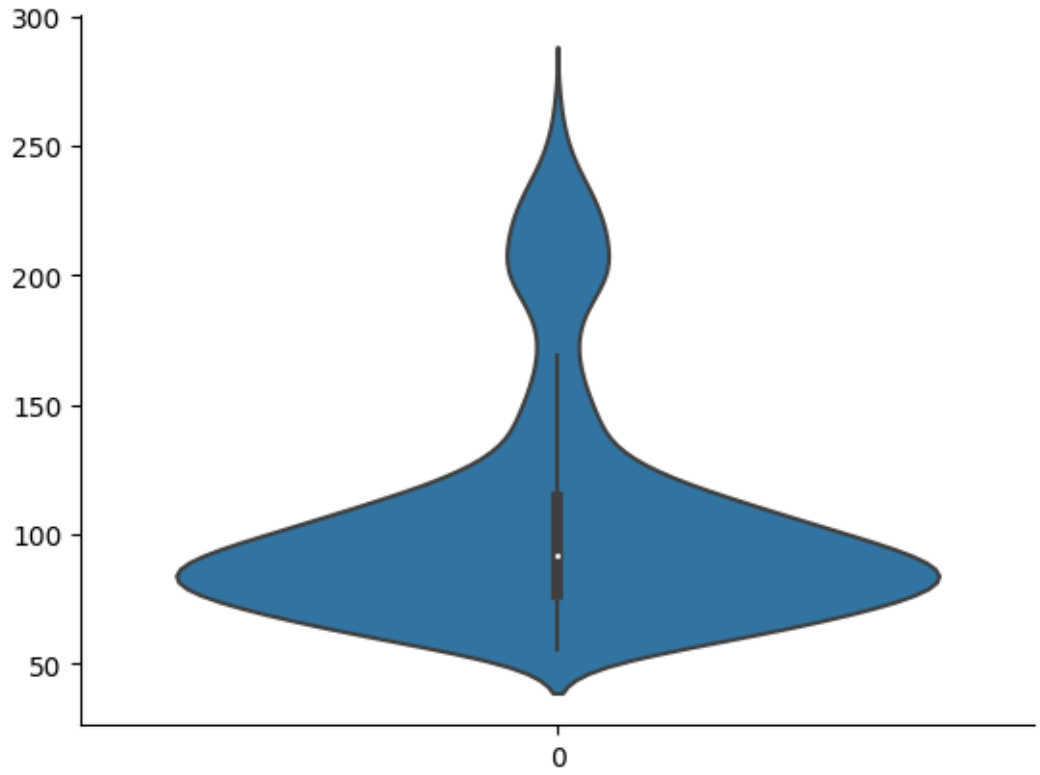


It is found that the age feature has no outliers and is having the range of 0 to 85(approx.). the maximum density around the group 20 to 60

```
plt.figure()
```

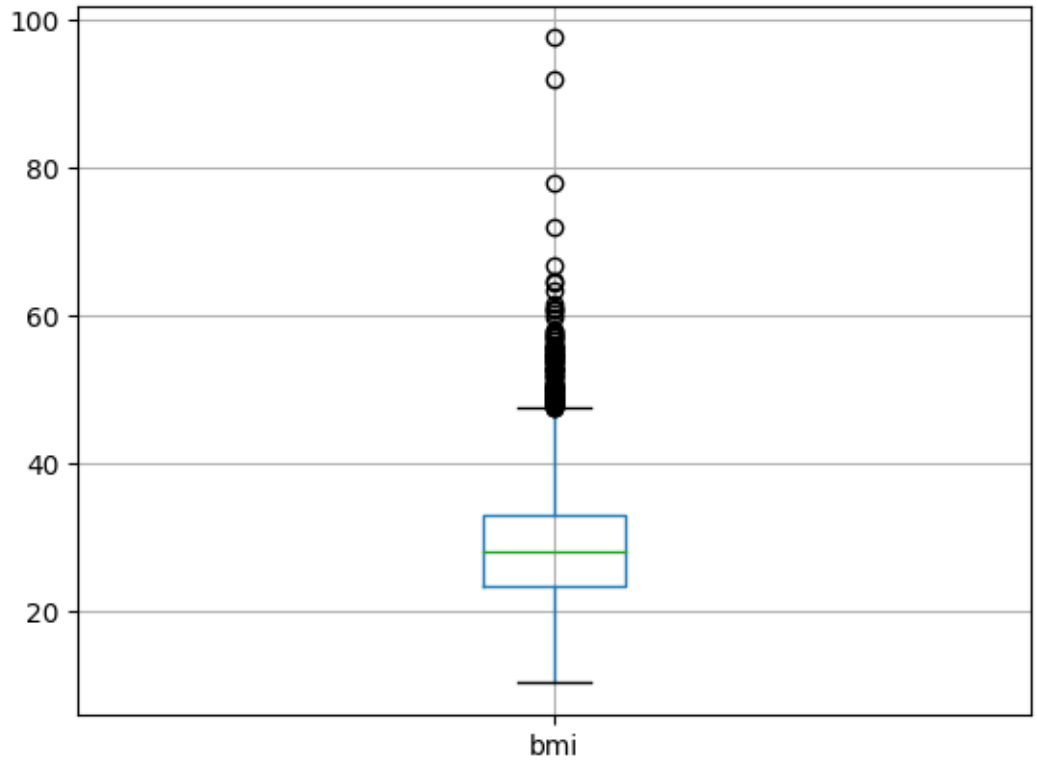
```
Stroke.boxplot(['avg_glucose_level'])
sns.violinplot(Stroke['avg_glucose_level'])
sns.despine()
```

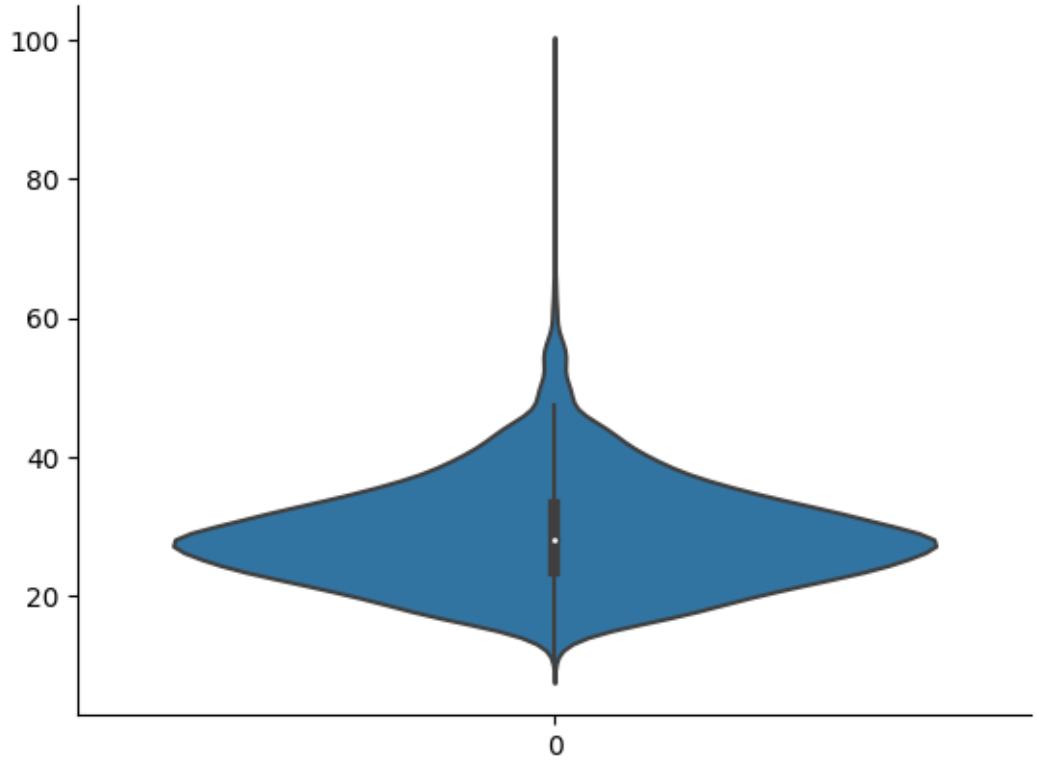




The Avg. glucose feature has ouliers in the upper bound of the range. The range of the data is around 50 to 170(approx.). the density of the data is around 60 to 120.

```
plt.figure()
Stroke.boxplot(['bmi'])
sns.violinplot(Stroke['bmi'])
sns.despine()
```





The BMI feature has the outliers in the upper bound of the range of 10 to 50(approx.). with density of around 20 to 40.

```
Stroke.isna().sum()
```

The na sum was found to get the null values of the dataset which has been found in BMI features with the count around 201.there was ‘Unknown’ values found in the feature smoking status and ‘Other’ values found in gender.

The above all three values are considered to be null as they will have no effect on the dataset they have been droped as well as the outliers as they will alter the prediction accuracy:

```
Stroke=Stroke.dropna()

Stroke=Stroke[Stroke['smoking_status'] !="Unknown"]
Stroke=Stroke[Stroke.gender !="Other"]
```

```
Q1=Stroke['avg_glucose_level'].quantile(0.25)
Q3=Stroke['avg_glucose_level'].quantile(0.75)
IQR=Q3 - Q1

lower_bound=Q1 -1.5 * IQR
upper_bound=Q3 +1.5 * IQR

outliers=Stroke.index[(Stroke['avg_glucose_level']<lower_bound)|(Stroke
['avg_glucose_level']>upper_bound)]
Stroke=Stroke.drop(outliers)

Q1=Stroke['bmi'].quantile(0.25)
Q3=Stroke['bmi'].quantile(0.75)
IQR=Q3 - Q1

lower_bound=Q1 -1.5 * IQR
upper_bound=Q3 +1.5 * IQR

outliers=Stroke.index[(Stroke['bmi']<lower_bound)|(Stroke['bmi']>upper_
bound)]
Stroke=Stroke.drop(outliers)
```

After dropping the null values the overview of the data set taken again using:

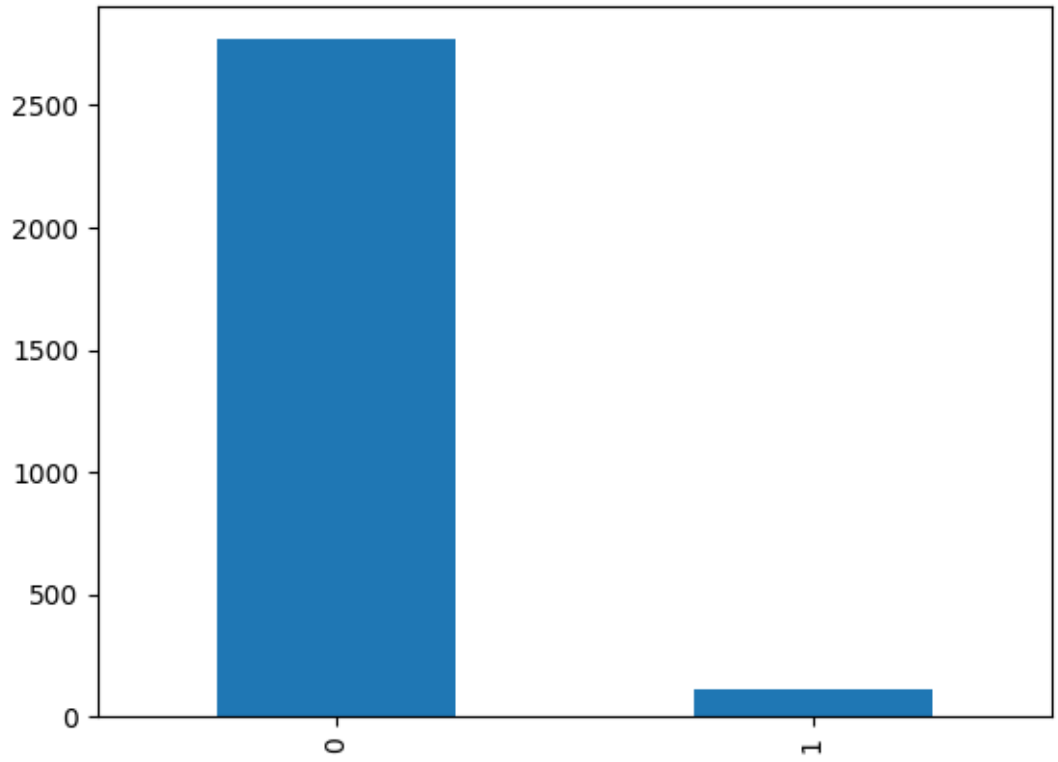
```
Stroke.isna().sum()
```

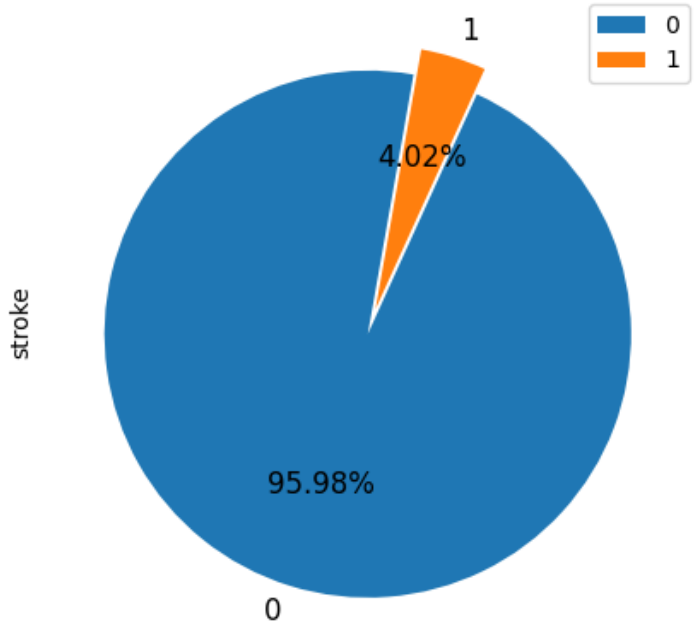
```
Stroke.info()
Stroke.shape
Stroke.describe()
```

The number of data left was 3425.with no null values found this time so the dropping was the success.

The Target variable is checked for the distribution of the classes using the bar plot and pie plot.

```
Stroke['stroke'].value_counts().plot.bar()
Stroke.stroke.value_counts().plot(kind='pie', subplots=True,
autopct='%1.2f%%'
,explode = (0.05, 0.05), startangle=80, legend=True,fontsize=12,
figsize=(8,5), textprops={'color':'black'})
```





It was found that the data was highly imbalanced with majority class 0 and minority class 1. With 95.98% and 4.02% respectively.

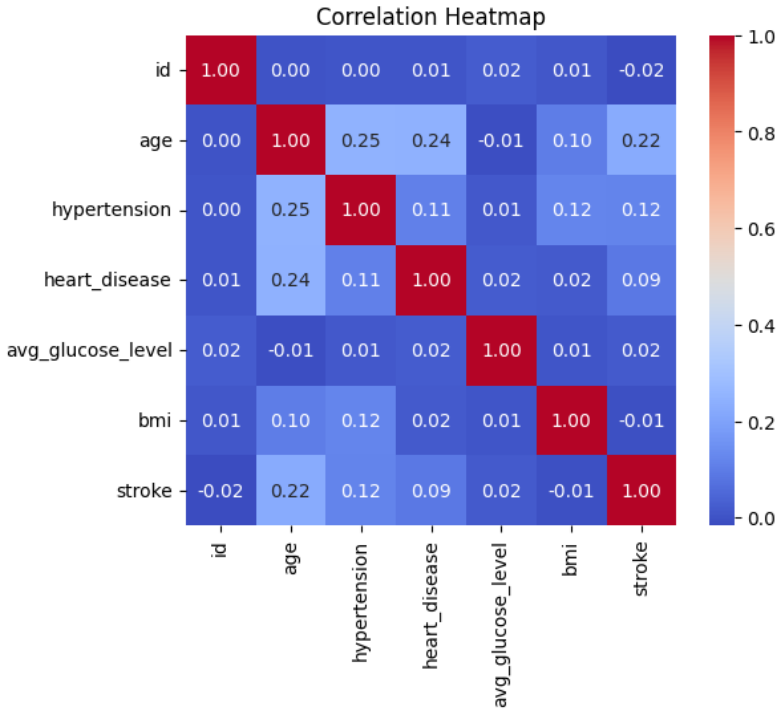
2.3.3 Feature Relationship(Correlation)

The relationship between the features can be found using the correlation heatmap.

According to the correlation:

Relationship with Target vs Features before scaling and label encoding:

```
crr=Stroke.corr()
sns.heatmap(crr,annot=True,cmap="coolwarm",fmt='.2f',square=True)
plt.title("Correlation Heatmap")
plt.show()
```



ID: least related to the target variable by correlation of (-0.02) contribute to occurance in the opposite result to target variable.

Age: most related to the target variable by correlation of (0.22)

Hypertension: also contributes in the cause of target variable. Correlation(0.12)

Heart disease: contributes in the occurance of target variable with (0.09)

Avg. glucose Level: contributes in the occurance with correlation (0.02)

Bmi : opposing the occurance of target variable (-0.01)

Relationship b/w target and Feature after scaling and label encoding:

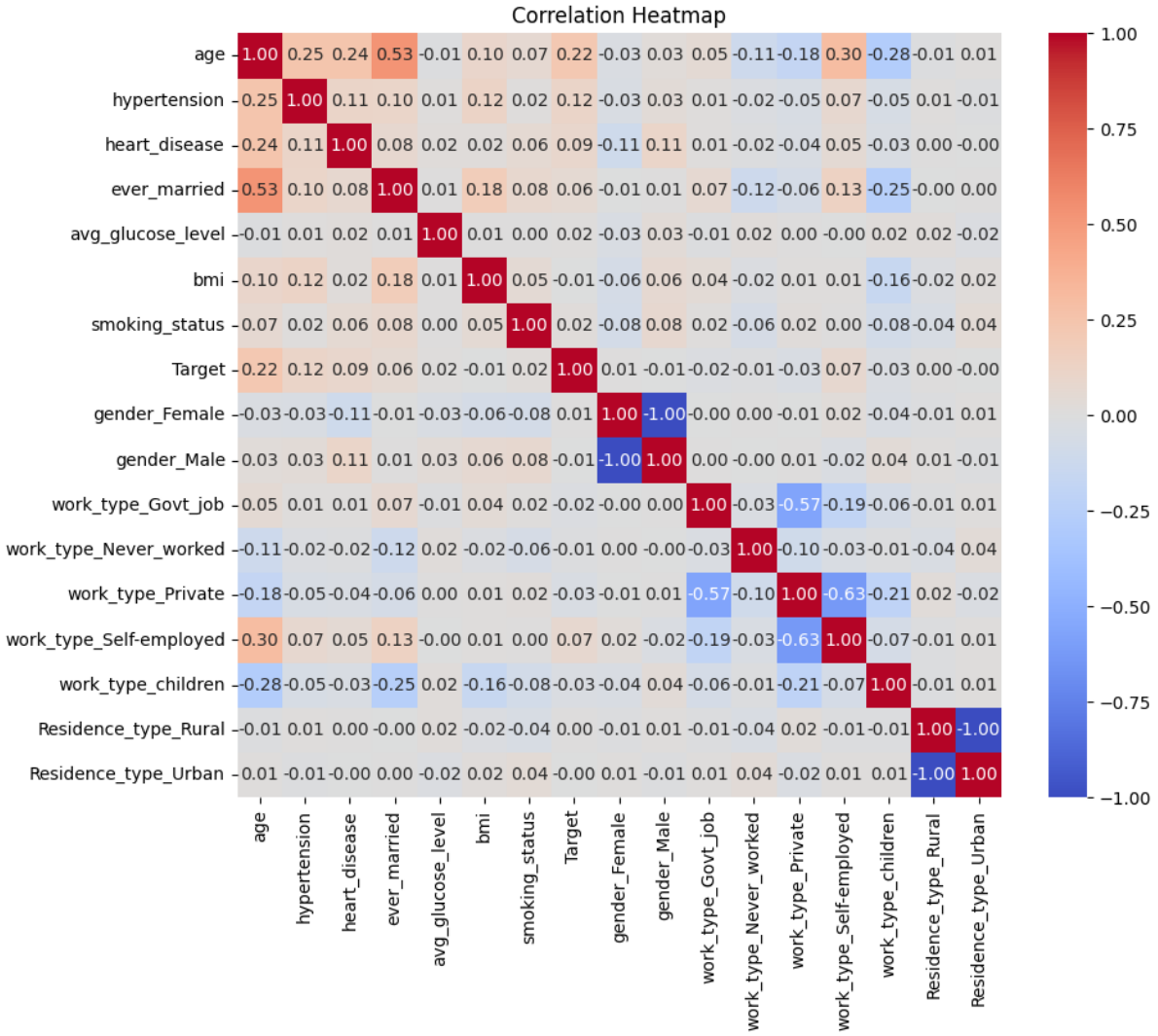
```
Stroke=Stroke.drop(columns=['id'])
Stroke.rename(columns={'stroke':'Target'}, inplace=True)


Stroke=pd.get_dummies(Stroke,columns=['gender','work_type','Residence_t
ype'])
Stroke['smoking_status']=Stroke['smoking_status'].replace({'never
smoked':0,'formerly smoked':1,'smokes':2})
Stroke['ever_married']=Stroke['ever_married'].replace({'Yes':1,'No':0})
scaler=MinMaxScaler()

scaler.fit(Stroke)

Stroke_scaled=pd.DataFrame(scaler.transform(Stroke),columns=Stroke.colu
mns)
```

```
crr=Stroke_scaled.corr()
sns.heatmap(crr,annot=True,cmap="coolwarm",fmt='.1f',square=True)
plt.title("Correlation Heatmap")
plt.show()
```



According to the heatmap there is a relationship between target variables found either positive or negative but residence type has no general relationship with the Target.

2.4 EDA conclusion

The exploratory data analysis revealed crucial insights into the dataset, allowing for an understanding of its structure, the distribution of features, handling missing values, and outliers. It highlighted the imbalance in the target variable and explored correlations between features and the target. Preprocessing steps, such as handling missing data and outliers, were applied to ensure the data's quality for model training. The analysis provides a foundation for further model development, emphasizing the need for techniques to address class imbalance and feature engineering to enhance predictive performance.

3 Experimental Design

3.1 Identification of your chosen supervised learning algorithm(s)

The Random Forest algorithm leverages an ensemble approach by building numerous decision trees during training. For regression tasks, it outputs the mean prediction, while for classification, it relies on the mode of the classes. This method often yields more balanced predictions, particularly when employing parameters like class weight and utilizing oversampling techniques to address imbalanced datasets.

Among the available options, Random Forest was selected over alternatives like K-Nearest Neighbors (KNN) and Support Vector Machines (SVM). KNN, while effective in certain scenarios, may encounter challenges with imbalanced datasets, especially when the minority class is notably smaller. Its reliance on local neighbourhood information might lead to overlooking the minority class, impacting its predictive performance.

On the other hand, SVM, another considered choice, showed inferior performance compared to Random Forest in the comparative analysis. This outcome supported the preference for Random Forest due to its ability to handle imbalanced data better and deliver improved results in the specific context of the analysis..

3.2 Identification of appropriate evaluation techniques

Classification metrics like Accuracy, Precision, Recall, F1-score, Specificity, and the Confusion Matrix are vital in assessing the performance of classification models:

Accuracy: It measures the percentage of correctly classified instances out of the total instances in the dataset. It's calculated as (TP + TN) / (TP + TN + FP + FN).

Precision: It signifies the proportion of correctly predicted positive instances among all instances predicted as positive. It's calculated as TP / (TP + FP).

Recall (Sensitivity): It measures the proportion of correctly predicted positive instances among all actual positive instances. It's calculated as TP / (TP + FN).

F1-score: It represents the harmonic mean of precision and recall, providing a single metric that balances both. It's useful when dealing with imbalanced classes as it considers both false positives and false negatives. It's calculated as 2 * (Precision * Recall) / (Precision + Recall).

Specificity: It measures the proportion of correctly predicted negative instances among all actual negative instances. It's calculated as TN / (TN + FP).

Confusion Matrix: It's a table that presents a summary of the model's performance. It consists of four metrics: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). It helps visualize the model's performance in making correct and incorrect predictions across different classes.

These metrics collectively provide insights into how well a classification model performs in correctly identifying instances of different classes and are essential in evaluating the model's effectiveness for the specific task at hand.

3.3 Data cleaning and Pre-processing transformations

3.3.1 Dropping Na Values and Irrelevant columns:

After conducting an Exploratory Data Analysis (EDA), it was observed that the 'bmi' column contains NaN (missing) values, and the 'smoking_status' column includes 'Unknown' entries. Given the healthcare nature of the data, filling these missing values could introduce synthetic information, potentially diverging from real-world scenarios. Thus, the decision was made to drop the NaN values.

Additionally, through analysis, it was determined that the 'Residence_type' feature doesn't significantly contribute to the analysis. Consequently, the 'Residence_type' column was removed from further consideration.

3.3.2 Label Encoding and one hot Encoding for string datatypes

Certain features in the dataset, like smoking status, work type, gender, and ever_married, are currently represented as string data types. There's a need to transform these features into a format suitable for machine learning models.

For features with ordinal categorical data (where the categories have a specific order or inheritance):

- Smoking status: Manual label encoding has been applied. The categories "formerly smoked," "never smoked," and "smokes" are encoded with integers. This encoding follows an order where "never smoked" is assigned 0, "formerly smoked" is represented as 1, and "smokes" is denoted as 2.
- Ever married: Similar manual encoding has been performed, assigning 0 for "no" and 1 for "yes."

Regarding nominal categorical data (where categories have no inherent order):

- Work type, gender, and ever married: One-hot encoding is utilized. Each category within these features is transformed into its binary representation, generating new columns for each category. This method enables the model to interpret these categorical variables without assuming any ordinal relationship between the categories..

3.3.3 Scaling/Normalizing:

After encoding, the features have undergone scaling to ensure uniformity in their scales. This step is crucial because features like BMI and average glucose level exhibit significantly higher magnitudes, which can potentially influence the learning process. The normalization technique has been employed for scaling, ensuring that the values of all features fall within the range of 0 to 1. This normalization process aids in preventing the undue influence of features with large magnitudes, promoting more effective learning techniques.

3.3.4 Splitting the Dataset:

After feature scaling, the dataset is divided into x (features) and y (target variables). Subsequently, the x and y data are split into training and testing datasets. To maintain proportional representation of both target variables in both training and testing sets, the dataset undergoes a stratified KFold split. This approach ensures that the highly imbalanced data, where the minority class has nearly 0 proportion, is mitigated. The KFold technique repeatedly trains and tests the model a specified number of times, in this case, 5 times. This process enhances the robustness of the model evaluation and helps in addressing imbalanced data concerns.

3.3.5 Sampling(Adding or removing synthetic data to dataset to make it balanced):

addressing the imbalance in the dataset, particularly the scarcity of data for the minority class, is crucial to prevent bias in the trained model. To mitigate this issue, oversampling is employed, involving the addition of synthetic data to the minority class to balance its representation with the majority class. It's noteworthy that oversampling is specifically applied to the training dataset, not the testing dataset. The testing dataset is reserved for evaluating the model's performance on real-world data to ensure its accuracy. Meanwhile, the training data is pivotal for instructing the model, and a shortage of data for the minority class can lead to a biased model.

In the context of this model, the RandomOverSampler technique is utilized. This technique duplicates existing instances from the minority class, effectively augmenting the training dataset. In this instance, the count of training data is increased from 2161 to 4148, contributing to a more balanced and unbiased training process.

3.4 Limitations and Options

The primary constraint in the dataset stems from its imbalanced distribution of classes within the target variables. Given its nature as healthcare data, the significant class imbalance may compromise the model's precision, particularly for the minority classes.

To enhance predictive accuracy, a more effective approach might involve prioritizing data gathering efforts rather than relying solely on oversampling techniques. While oversampling introduces synthetic data, it may not accurately represent certain scenarios, making precise predictions challenging in specific cases.

4 Predictive Modelling / Model Development

4.1 The predictive modelling process

The resampled training data is then employed to train a Random Forest classifier. Several parameters are incorporated into the Random Forest classifier to optimize its performance:

- Class Weight: Given the imbalanced nature of the dataset, achieving a balanced result is essential. To address this, the minority class is trained with greater weight than the majority class. The compute_class_weight library is utilized to calculate the appropriate class weights for achieving a balanced outcome.

- Random State: The random_state parameter is employed to ensure reproducibility by obtaining consistent results in each run.

- N Estimators: This parameter specifies the number of trees to be created in the Random Forest.

Following the model training, predictions are made on both seen (training data) and unseen data (testing data). The accuracy, classification report (including precision, recall, F1-score, support, accuracy, macro average, and weighted average), and confusion matrices are then utilized to evaluate these predictions.

```
x=Stroke.drop('Target',axis=1)
y=Stroke['Target']


class_weights=compute_class_weight("balanced",classes=[0,1],y=yTrain)
class_weight_dict={i:class_weights[i] for i in
range(len(class_weights))}
rf=RandomForestClassifier(n_estimators=100,random_state=42,class_weight
=class_weight_dict)

k_folds=5
kf=StratifiedKFold(n_splits=k_folds,shuffle=True,random_state=42)
for train_index,test_index in kf.split(x,y):
  X_train,X_test=x.iloc[train_index],x.iloc[test_index]
  Y_train,Y_test=y.iloc[train_index],y.iloc[test_index]
```

```
  smote=RandomOverSampler(random_state=42)

X_train_resampled,Y_train_resampled=smote.fit_resample(X_train,Y_train)

  rf.fit(X_train_resampled,Y_train_resampled)

  y_pred=rf.predict(X_test)
  accuracy=accuracy_score(Y_test,y_pred)

  yPred=rf.predict(X_train_resampled)
  accuracy1=accuracy_score(Y_train_resampled,yPred)
  print("----------------------------------------------------------------
------")
  print("--------Shape of the Traing Data {}-------------------------
------". format(X_train_resampled.shape))

  print("Accuracy: {}\n".format(accuracy))
  print("Classification
Report:\n{}\n".format(classification_report(Y_test,y_pred)))
  print("Confusion Matrices:
\n{}".format(confusion_matrix(Y_test,y_pred)))


  print("Accuracy: {}\n".format(accuracy1))
  print("Classification
Report:\n{}\n".format(classification_report(Y_train_resampled,yPred)))
  print("Confusion Matrices:\n
{}".format(confusion_matrix(Y_train_resampled,yPred)))
  print("----------------------------------------------------------------
------")
```

4.2 Evaluation results on “seen” and “Unseen” data

The result for the seen data as follow:

```
-------Seen Data:--------------
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2213
           1       1.00      1.00      1.00      2213

    accuracy                           1.00      4426
   macro avg       1.00      1.00      1.00      4426
```

```
weighted avg       1.00      1.00      1.00      4426

Confusion Matrices:
 [[2213    0]
 [   0 2213]]
```

Whereas for the unseen data the model has given different results everytime. The result are as follow:

```
--------Shape of the Traing Data (4426, 14)----------------------------
-----
-------Unseen Data:-----------
Accuracy: 0.9480069324090121

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       553
           1       0.00      0.00      0.00        24

    accuracy                           0.95       577
   macro avg       0.48      0.49      0.49       577
weighted avg       0.92      0.95      0.93       577

------------------------------------------------------------------
--------Shape of the Traing Data (4424, 14)----------------------------
-----
-------Unseen Data:-----------
Accuracy: 0.9532062391681109

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       554
           1       0.25      0.09      0.13        23

    accuracy                           0.95       577
   macro avg       0.61      0.54      0.55       577
weighted avg       0.93      0.95      0.94       577


Confusion Matrices:
[[548   6]
 [ 21   2]]


--------Shape of the Traing Data (4426, 14)----------------------------
-----
-------Unseen Data:-----------
Accuracy: 0.9548611111111112

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       553
           1       0.20      0.04      0.07        23

    accuracy                           0.95       576
```

```
   macro avg       0.58      0.52      0.52       576
weighted avg       0.93      0.95      0.94       576


Confusion Matrices:
[[549   4]
 [ 22   1]]


--------Shape of the Traing Data (4426, 14)-----------------------------
-----
-------Unseen Data:-----------
Accuracy: 0.953125

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       553
           1       0.17      0.04      0.07        23

    accuracy                           0.95       576
   macro avg       0.56      0.52      0.52       576
weighted avg       0.93      0.95      0.94       576


Confusion Matrices:
[[548   5]
 [ 22   1]]


------------------------------------------------------------------------
--------Shape of the Traing Data (4426, 14)-----------------------------
-----
-------Unseen Data:-----------
Accuracy: 0.9444444444444444

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       553
           1       0.00      0.00      0.00        23

    accuracy                           0.94       576
   macro avg       0.48      0.49      0.49       576
weighted avg       0.92      0.94      0.93       576


Confusion Matrices:
[[544   9]
 [ 23   0]]
```

The model displays signs of overfitting as it excels on the familiar training data but struggles with the unseen test data, especially regarding the minority class (class 1). To counter overfitting, potential remedies include leveraging regularization techniques such as adjusting parameters like min_samples_split, min_samples_leaf, max_depth, and max_features. Additionally, augmenting the

dataset with more samples from the minority class or exploring alternative algorithms tailored for handling imbalanced data could be beneficial.

5 Evaluation and further modelling improvements

Step 1: Model Selection

The Dataset which is explored here predicting stroke dataset. With target variable Stroke with values “0” and “1”. So as the target result is classified into two classes it is a classification model with binary classification.During the it was found that the dataset is imbalanced with the majority class 0 and minority class 1.

Step 2: Data Preparation

NA values: The NA values are found in the bmi column also some unknown values in smoking column.

Encoding: The dataset columns which contains string values are converted into numerical values for nominal values (columns=work type, gender): one hot encoding, ordinal values: Label encoding.

Scaling: The dataset is normalised using minmaxscaler to get all the feature values between min (0) and Max(1).

Step 3: Model Development

- Chosen model: Random Forest Classifier
- The Splitting of the data set is done by k fold which also do the cross validation of the model.
- The model was trained with oversampled training dataset while containing the parameters like class weight for balanced training, random_state for reproducibility, n_estimator to define the no. of tree it’s going to create.

Step 4: Model Evaluation

- Performance of the Model is Evaluated using the Accuracy, Classification report. From where it was found that the Model is overfitted and biased towards the majority class.
- Using the Confusion matrices gives the overview that it has more true negatives and less false negatives whereas it has more false positives and less true positives. Which is considered to be a biased result favouring the majority class.
- Interpretation: The model

Step 5: Model Improvement

- Hyperparameter Tuning: The Hyperparameter tunning is required for the model improvement as the model is overfitted and giving biased result. So for feature engineering getting the best value for the parameter’s tunning would be very helpful.
- Using Grid Search for the hypertunning. The code below gives the best parameters and also check the best model accuracy.

```
param_grid= {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
    'max_samples': [None, 0.5, 0.7, 0.9],
    'warm_start':[True,False]
   }

search=GridSearchCV(estimator=rf,param_grid=param_grid, cv=5,
scoring='accuracy')
search.fit(xTrainResampled,yTrainResampled)

bestParams=search.best_params_
print("Best Hyperparameters: ", bestParams)

bestModel =search.best_estimator_
test_accuracy=bestModel.score(xTest,yTest)
print("test Accuracy",test_accuracy)
```

- Feature engineering techniques for random forest modelling like max depth, min sample leaf, max features, min samples split etc can be used to improve the model and to get their best values hyperparameter tunning is done. The best parameter for the random forest found by grid search are as follows.

```
Best Hyperparameters:  {'max_depth': None, 'max_features': 'sqrt',
'max_samples': 0.9, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

- test Accuracy 0.9486823855755895

- After hyperparameter tuning the following is the final model for prediction:

```
class_weights=compute_class_weight("balanced",classes=[0,1],y=yTrain)
class_weight_dict={i:class_weights[i] for i in
range(len(class_weights))}
rf=RandomForestClassifier(n_estimators=100,random_state=42,class_weight
=class_weight_dict,max_depth= None, max_features= 'sqrt', max_samples=
0.9, min_samples_leaf= 1, min_samples_split= 2, ccp_alpha= 0.0,
warm_start= True,criterion='entropy')
```

- **n_estimators=100:**The number of trees to be created. In this case 100.
- **random_state=42:** ensures the reproducibility of the results.
- **class_weight=class_weight_dict:** assigns weight to the classes. Here indicating custom weights are provided to the classes(more weight to minority class)

- **max_depth= None:** The maximum depth of the trees. None means nodes are expanded until all leaves are pure or until all leaves contain less than **min_samples_split samples.**
- **max_features= 'sqrt':** no. of features consider to split at each node here it is square root of no. of features.
- **max_samples= 0.9:**maximum sample to draw from dataset to train each tree. Here its 90%
- **min_samples_leaf= 1:**min number of leaf samples required to be at leaf node here atleast 1 required.
- **min_samples_split= 2 :** min number of sample required to split an internal node.
- **ccp_alpha= 0.0 :** pruning cost complexity. 0 means no pruning
- **warm_start= True :** incremental training model uses previous call to fit and adds more estimators to the ensemble.
- **criterion='entropy':**measure the quality of split.The setting of 'entropy' indicates that the model uses the information gain criterion to decide how to split nodes in the trees.
- With which I got the following results in StratifiedKfold:

```
---------------------------------------------------------------------
--------Shape of the Traing Data (4426, 14)---------------------------------
-------Unseen Data:-----------
Accuracy: 0.9480069324090121

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       553
           1       0.00      0.00      0.00        24

    accuracy                           0.95       577
   macro avg       0.48      0.49      0.49       577
weighted avg       0.92      0.95      0.93       577


Confusion Matrices:
[[547   6]
 [ 24   0]]


-------Seen Data:--------------
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2213
           1       1.00      1.00      1.00      2213

    accuracy                           1.00      4426
   macro avg       1.00      1.00      1.00      4426
weighted avg       1.00      1.00      1.00      4426


Confusion Matrices:
```

```
[[2213    0]
 [   0 2213]]
_____Feature Importance_____
Feature ranking::
1. Feature 0:0.4055956448836676
2. Feature 4:0.23170000175091968
3. Feature 5:0.19998671285423975
4. Feature 6:0.052211522150903036
5. Feature 1:0.020067501553893124
6. Feature 3:0.01650100941966415
7. Feature 12:0.013486136974095533
8. Feature 2:0.012708837745779777
9. Feature 8:0.011840214781006024
10. Feature 11:0.01071417138293838
11. Feature 7:0.010606927752333115
12. Feature 9:0.009809281740318617
13. Feature 13:0.004250109535089701
14. Feature 10:0.0005219274751514324
----------------------------------------------------------------------



----------------------------------------------------------------
--------Shape of the Traing Data (4424, 14)---------------------------------
-------Unseen Data:-----------
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       554
           1       1.00      1.00      1.00        23

    accuracy                           1.00       577
   macro avg       1.00      1.00      1.00       577
weighted avg       1.00      1.00      1.00       577


Confusion Matrices:
[[554   0]
 [  0  23]]


-------Seen Data:--------------
Accuracy: 0.8711573236889693

Classification Report:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89      2212
           1       1.00      0.75      0.85      2212

    accuracy                           0.87      4424
   macro avg       0.90      0.87      0.87      4424
weighted avg       0.90      0.87      0.87      4424

```

```
Confusion Matrices:
 [[2206    6]
 [ 564 1648]]
___________Feature Importance_____________
Feature ranking::
1. Feature 0:0.4055956448836676
2. Feature 4:0.23170000175091968
3. Feature 5:0.19998671285423975
4. Feature 6:0.052211522150903036
5. Feature 1:0.020067501553893124
6. Feature 3:0.01650100941966415
7. Feature 12:0.013486136974095533
8. Feature 2:0.012708837745779777
9. Feature 8:0.011840214781006024
10. Feature 11:0.01071417138293838
11. Feature 7:0.010606927752333115
12. Feature 9:0.009809281740318617
13. Feature 13:0.004250109535089701
14. Feature 10:0.0005219274751514324
-----------------------------------------------------------------------



/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:452: UserWarning: Warm-start fitting without increasing n_estimators does not fit new trees.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:452: UserWarning: Warm-start fitting without increasing n_estimators does not fit new trees.
  warn(
-----------------------------------------------------------------------
--------Shape of the Traing Data (4426, 14)---------------------------------
-------Unseen Data:-----------
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       553
           1       1.00      1.00      1.00        23

    accuracy                           1.00       576
   macro avg       1.00      1.00      1.00       576
weighted avg       1.00      1.00      1.00       576


Confusion Matrices:
[[553   0]
 [  0  23]]


-------Seen Data:--------------
Accuracy: 0.8755083596927248

Classification Report:
              precision    recall  f1-score   support
```

```
•
•           0      0.80     1.00     0.89     2213
•           1      1.00     0.75     0.86     2213
•
•      accuracy                         0.88     4426
•     macro avg      0.90     0.88     0.87     4426
•  weighted avg      0.90     0.88     0.87     4426
•
•
•  Confusion Matrices:
•   [[2207    6]
•   [ 545 1668]]
•  _____Feature Importance_____
•  Feature ranking::
•  1. Feature 0:0.4055956448836676
•  2. Feature 4:0.23170000175091968
•  3. Feature 5:0.19998671285423975
•  4. Feature 6:0.052211522150903036
•  5. Feature 1:0.020067501553893124
•  6. Feature 3:0.01650100941966415
•  7. Feature 12:0.013486136974095533
•  8. Feature 2:0.012708837745779777
•  9. Feature 8:0.011840214781006024
•  10. Feature 11:0.01071417138293838
•  11. Feature 7:0.010606927752333115
•  12. Feature 9:0.009809281740318617
•  13. Feature 13:0.004250109535089701
•  14. Feature 10:0.0005219274751514324
•  ----------------------------------------------------------------
•
•
•
•  ----------------------------------------------------------------
•  --------Shape of the Traing Data (4426, 14)---------------------------------
•  -------Unseen Data:-----------
•  Accuracy: 1.0
•
•  Classification Report:
•              precision    recall  f1-score   support
•
•           0      1.00     1.00     1.00      553
•           1      1.00     1.00     1.00       23
•
•      accuracy                         1.00      576
•     macro avg      1.00     1.00     1.00      576
•  weighted avg      1.00     1.00     1.00      576
•
•
•  Confusion Matrices:
•  [[553   0]
•   [  0  23]]
•
•
•  -------Seen Data:--------------
•  Accuracy: 0.8752824220515137
•
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89      2213
           1       1.00      0.75      0.86      2213

    accuracy                           0.88      4426
   macro avg       0.90      0.88      0.87      4426
weighted avg       0.90      0.88      0.87      4426


Confusion Matrices:
 [[2207    6]
 [ 546 1667]]
_____Feature Importance_____
Feature ranking::
1. Feature 0:0.4055956448836676
2. Feature 4:0.23170000175091968
3. Feature 5:0.19998671285423975
4. Feature 6:0.052211522150903036
5. Feature 1:0.020067501553893124
6. Feature 3:0.01650100941966415
7. Feature 12:0.013486136974095533
8. Feature 2:0.012708837745779777
9. Feature 8:0.011840214781006024
10. Feature 11:0.01071417138293838
11. Feature 7:0.010606927752333115
12. Feature 9:0.009809281740318617
13. Feature 13:0.004250109535089701
14. Feature 10:0.0005219274751514324
----------------------------------------------------------------------


/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:452: UserWarning: Warm-start fitting without increasing n_estimators does not fit new trees.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:452: UserWarning: Warm-start fitting without increasing n_estimators does not fit new trees.
  warn(
----------------------------------------------------------------
--------Shape of the Traing Data (4426, 14)---------------------------------
-------Unseen Data:-----------
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       553
           1       1.00      1.00      1.00        23

    accuracy                           1.00       576
   macro avg       1.00      1.00      1.00       576
weighted avg       1.00      1.00      1.00       576


```

```
Confusion Matrices:
[[553   0]
 [  0  23]]


-------Seen Data:----------
Accuracy: 0.8707636692272933

Classification Report:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89      2213
           1       1.00      0.74      0.85      2213

    accuracy                           0.87      4426
   macro avg       0.90      0.87      0.87      4426
weighted avg       0.90      0.87      0.87      4426


Confusion Matrices:
 [[2207    6]
 [ 566 1647]]
__________Feature Importance___________
Feature ranking::
1. Feature 0:0.4055956448836676
2. Feature 4:0.23170000175091968
3. Feature 5:0.19998671285423975
4. Feature 6:0.052211522150903036
5. Feature 1:0.020067501553893124
6. Feature 3:0.01650100941966415
7. Feature 12:0.013486136974095533
8. Feature 2:0.012708837745779777
9. Feature 8:0.011840214781006024
10. Feature 11:0.01071417138293838
11. Feature 7:0.010606927752333115
12. Feature 9:0.009809281740318617
13. Feature 13:0.004250109535089701
14. Feature 10:0.0005219274751514324
----------------------------------------------------------------------
```

Step 6: Validation

- The K Fold Validation given me the following accuracy and average accuracy:

```
k=5
kf=KFold(n_splits=k,shuffle=True,random_state=42)
result=cross_val_score(rf,xTrainResampled,yTrainResampled,cv=kf,scoring='accuracy')
print("KFold results:: {}".format(result))
print("kFold Average:: {}".format(result.mean()))
```

```
KFold results:: [0.99036145 0.9939759  0.98554217 0.99034982
0.99155609]
kFold Average:: 0.9903570857616231
```

6 Conclusion

6.1 Summary of results

The model's evaluation results highlight its outstanding performance on unseen data, achieving a flawless accuracy of 100%. The classification report underscores its precision, recall, and F1-score excellence for both classes (0 and 1), and the confusion matrix reflects impeccable predictions across the board.

However, when applied to seen data, the model demonstrates a lower accuracy of approximately 87.1%. The classification report reveals varied precision, recall, and F1-scores for classes 0 and 1, with a notable dip in recall (74%) for class 1, indicating some misclassification. The confusion matrix further illustrates instances of misclassifications, particularly for class 1.

Examining feature importance, the model identifies features 0, 4, and 5 as the most influential in predicting the target variable, with other features following in descending order of importance.

While the model excels on unseen data, addressing the observed misclassifications in the seen data, especially for class 1, may require further exploration and fine-tuning. Adjustments or enhancements could enhance the model's overall performance, providing more reliable predictions in real-world scenarios.

```
class_weights=compute_class_weight("balanced",classes=[0,1],y=yTrain)
class_weight_dict={i:class_weights[i] for i in
range(len(class_weights))}
rf=RandomForestClassifier(n_estimators=100,random_state=42,class_weight
=class_weight_dict,max_depth= None, max_features= 'sqrt', max_samples=
0.9, min_samples_leaf= 1, min_samples_split= 2, ccp_alpha= 0.0,
warm_start= True,criterion='entropy')
```

6.2 Reflection on Individual Learning

During the process of training a model, from Selecting to final model There was lot learn.

The process of EDA helped me to learn the different visualisation techniques and understanding different plots and graph for the features and their meanings. How does encoding work and why is the one hot encoding and label encoding is done according to the categorical values (Ordinal or nominal).The process of scaling and normalization was little bluured at the start but when reached at the stage of training model it became more clearer that scaling is used so all features have same ranges so one feature donot overpower other during training stage.

The process of oversampling and undersampling became more familiar and how they work on they work on the imbalanced data. The main question during the this model which cam into consideration was Why Oversampling is not done before splitting the data into training and testing. But as the process modelling goes on it became clearer that synthetic can be used to train the model but for the testing only real world data is used to authenticate the accuracy of the dataset like whether it is predicting real world data accurately or not).

Training and evaluating the model help me learned the accuracy of different models with my dataset and also gives the knowledge of which type of feature engineering can be done with different type of model. Evaluation of the dataset help to learn how to check the functionality of the model and which different types of evaluation techniques can be used evaluate the model.

This Report helped to get the basic inderstanding of some of the Machine learning principles.In future , I would like to improve my knowledge of more model, preprocessing techniques.

7 References

- Introduction to AI and ML: BCU Moodle. (2023-24). "Artificial Intelligence and Machine Learning" [Course: CMP6202 Artificial Intelligence and Machine Learning A S1 2023/4 (bcu.ac.uk)]. Birmingham City University Moodle platform. Available at: [Course: CMP6202 Artificial Intelligence and Machine Learning A S1 2023/4 (bcu.ac.uk)] (Accessed: 2023-24).
- Scaling, Normalisation and feature Engineering: Feature Engineering: Scaling, Normalization, and Standardization - GeeksforGeeks [Online]. Available at: [Feature Engineering: Scaling, Normalization, and Standardization - GeeksforGeeks] (Accessed: 2023-24).
- Why do scale data?: Why Do We Scale Data In Machine Learning | Robots.net [Online]. Available at: [Why Do We Scale Data In Machine Learning | Robots.net] (Accessed: 2023-24).
- Why do we scale data? Why do we scale our data in Machine Learning? (deepchecks.com) [Online]. Available at: [Why do we scale our data in Machine Learning? (deepchecks.com)] (Accessed: 2023-24).
- Smote: Smote Definition & Meaning - Merriam-Webster [Online]. Available at: [: Smote Definition & Meaning - Merriam-Webster] (Accessed: 2023-24).
- Some Machine learning terms: Machine Learning Glossary | Google for Developers [Online]. Available at: [Machine Learning Glossary | Google for Developers] (Accessed: 2023-24).
- Compute Class weight: sklearn.utils.class_weight.compute_class_weight — scikit-learn 1.3.2 documentation [Online]. Available at: [sklearn.utils.class_weight.compute_class_weight — scikit-learn 1.3.2 documentation] (Accessed: 2023-24).
- Difference between predicting seen and unseen data: classification - difference between predicting seen and unseen data - Data Science Stack Exchange. Available at: [classification - difference betwen predicting seen and unseen data - Data Science Stack Exchange] (Accessed: 2023-24).

- Overfitting in ML: Overfitting in Machine Learning: What It Is and How to Prevent It (elitedatascience.com) [Online]. Available at: [Overfitting in Machine Learning: What It Is and How to Prevent It (elitedatascience.com)] (Accessed: 2023-24).
- Sampling in ML: What is Sampling in Machine Learning? - reason.town [Online]. Available at: [What is Sampling in Machine Learning? - reason.town] (Accessed: 2023-24).
- Sampling: Sampling—Statistical approach in Machine learning | by Suresha HP | Analytics Vidhya | Jan, 2021 | Medium | Analytics Vidhya [Online]. Available at: [Sampling — Statistical approach in Machine learning | by Suresha HP | Analytics Vidhya | Jan, 2021 | Medium | Analytics Vidhya] (Accessed: 2023-24).
- Splitting data machine learning: Splitting Data for Machine Learning Models - GeeksforGeeks [Online]. Available at: [Splitting Data for Machine Learning Models - GeeksforGeeks] (Accessed: 2023-24).
- Random Forest: What Is Random Forest? A Complete Guide | Built In [Online]. Available at: [What Is Random Forest? A Complete Guide | Built In] (Accessed: 2023-24).
- KNN, Random Forest, and SVM: What Is Random Forest? A Complete Guide | Built In [Online]. Available at: [What Is Random Forest? A Complete Guide | Built In] (Accessed: 2023-24).