Configuration Manual

MSc Research Project
MSc Data Analytics

Lilian Ifeoma Enwereobi
Student ID: x20255322

School of Computing
National College of Ireland

Supervisor: Qurrat UI Ain

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Lilian Ifeoma Enwereobi		
Student ID:	X20255322		
Programme:	MSc Data Analytics	**Year:**	2022/2023
Module:	Research Project		
Lecturer:	Qurrat UI Ain		
Submission Due Date:	14-08-2022		
Project Title:	Stoke prediction Using Model Comparison and Feature Selections		
Word Count:	1031 **Page Count: 10**		

Signature: Lilian Ifeoma Enwereobi

Date: 14-08-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	Lilian Ifeoma Enwereobi
Date:	14-08-2023
Penalty Applied (if applicable):	

Configuration Manual

Lilian Ifeoma Enwereobi
Student ID:

1 Introduction

Python was implemented to create algorithms based on machine learning and Deep Learning techniques for Stroke Prediction Research. In this manual, the system's specs and usage are described in depth.

2 Technical Specifications

2.1 Hardware Required

System Software: Windows 10, 64 bits.

RAM: 12 GB

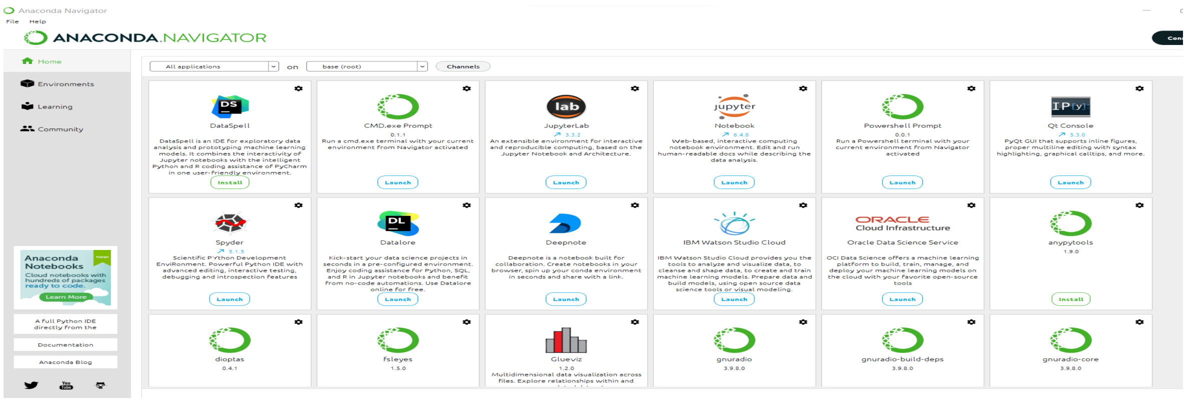
These are the only hardware-related criteria that have been met.

2.2 Software Required

You'll require to have Anaconda Jupyter Notebook installed to run the Python code. At the command prompt, type cd /some folder name to access the start-up folder. Enter jupyter notebook in the keyword search box to launch the Jupyter Notebook app. A fresh page or window in your browser will open with the user interface for the laptop.

Computation Syntax: Python

The most recent version of Anaconda, 5.0.0, as well as the Jupyter Notebook modifications it offers, are necessary since they let users define environment-specific kernels from the Jupyter Notebook interface. Designed to simplify package management and deployment, the anaconda is a package management system of the Python and R computational science programming languages.



DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.
Install

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch

JupyterLab
3.3.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
6.4.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch

Qt Console
5.3.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

Spyder
5.1.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.
Launch

Deepnote
Deepnote is a notebook built for collaboration. Create notebooks in your browser, spin up your conda environment in seconds and share with a link.
Launch

IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.
Launch

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools
Launch

anypytools
1.9.0
Install

dioptas
0.4.1

fsleyes
1.5.0

Glueviz
1.2.0
Multidimensional data visualization across files. Explore relationships within and

gnuradio
3.9.8.0

gnuradio-build-deps
3.9.8.0

gnuradio-core
3.9.8.0

Anaconda Notebooks
Cloud notebooks with hundreds of packages ready to code.
Learn More

A full Python IDE directly from the

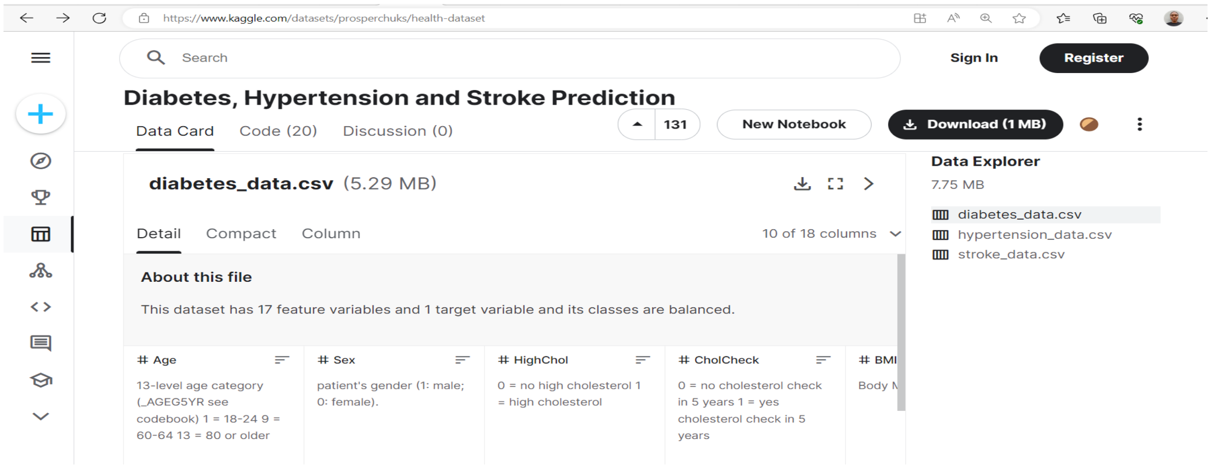
Documentation

Anaconda Blog

Fig 1: Anaconda Navigator Window

3. Collecting Dataset

The dataset can be obtained at https://www.kaggle.com/datasets/prosperchuks/health-dataset. There will be a file called Diabetes, Hypertension, and Stroke Prediction Dataset there, and downloading it only requires a single mouse click.



Diabetes, Hypertension and Stroke Prediction

Data Card Code (20) Discussion (0) 131 New Notebook Download (1 MB)

diabetes_data.csv (5.29 MB)

Detail Compact Column 10 of 18 columns

About this file

This dataset has 17 feature variables and 1 target variable and its classes are balanced.

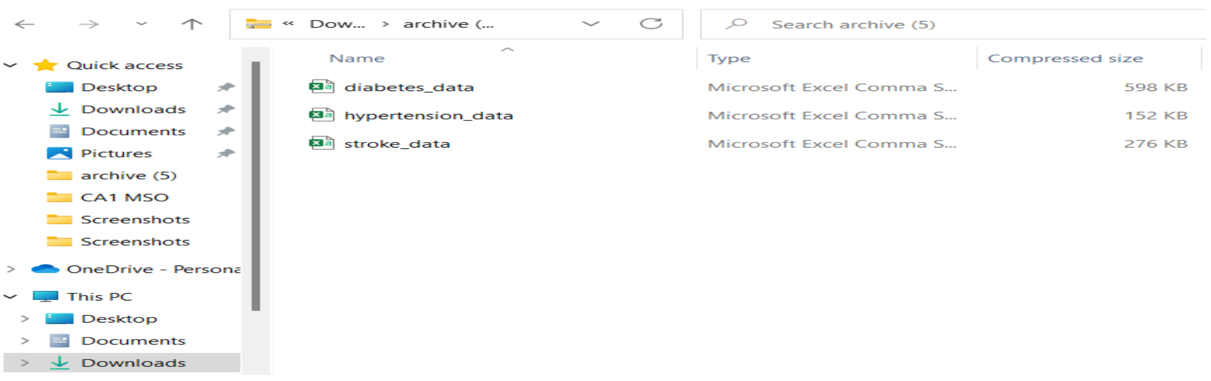
# Age	# Sex	# HighChol	# CholCheck	# BMI
13-level age category (_AGEG5YR see codebook) 1 = 18-24 9 = 60-64 13 = 80 or older	patient's gender (1: male; 0: female).	0 = no high cholesterol 1 = high cholesterol	0 = no cholesterol check in 5 years 1 = yes cholesterol check in 5 years	Body N

Data Explorer
7.75 MB
diabetes_data.csv
hypertension_data.csv
stroke_data.csv

Fig 2: Kaggle Dataset Repository

4. Downloading data for Jupyter Notebook:

The dataset must first be saved on any system-available local drive. It was saved as an archive because it has 3 datasets, so I separated it to take the dataset I needed.



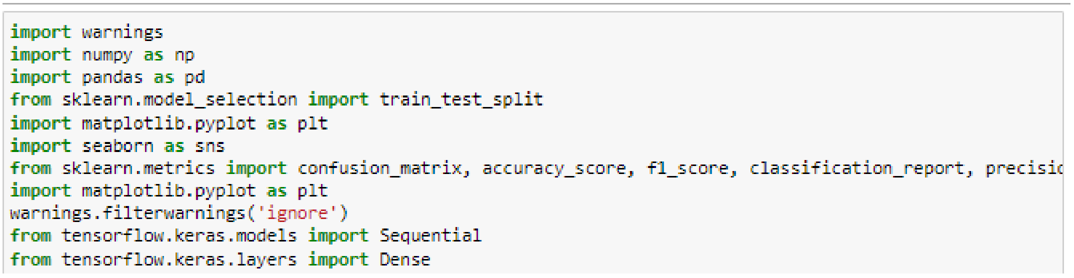
Name	Type	Compressed size
diabetes_data	Microsoft Excel Comma S...	598 KB
hypertension_data	Microsoft Excel Comma S...	152 KB
stroke_data	Microsoft Excel Comma S...	276 KB

Fig 2: My Download Drive held the dataset.

You can use the ANACONDA navigator to open the Jupyter Notebook version. The software used for the development of the notebook version makes use of the Python programming language.

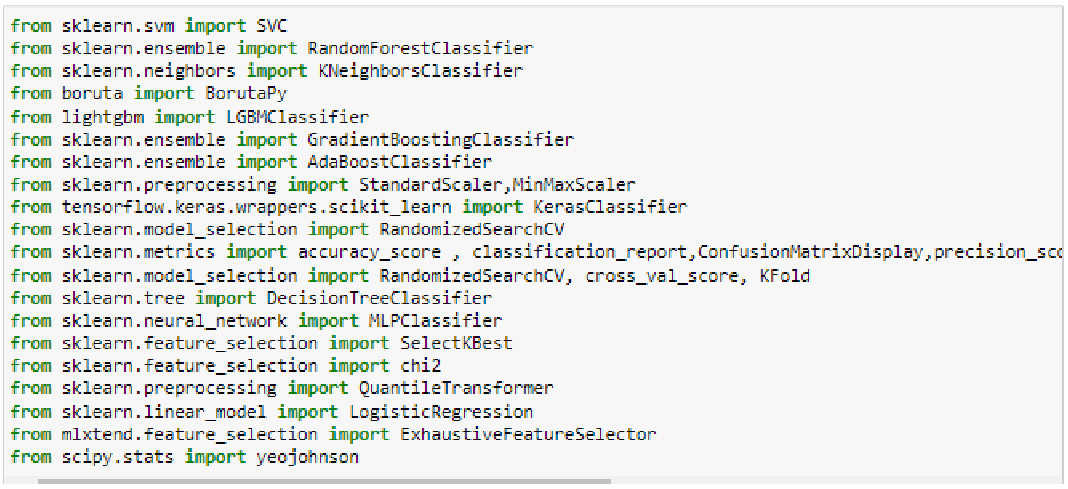
5. Package Installations and Library Importing

The following libraries were utilized in this study for data pre-processing, model construction, and model evaluation:



```
import warnings
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, classification_report, precisic
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Fig 3: Imported Libraries and Packages

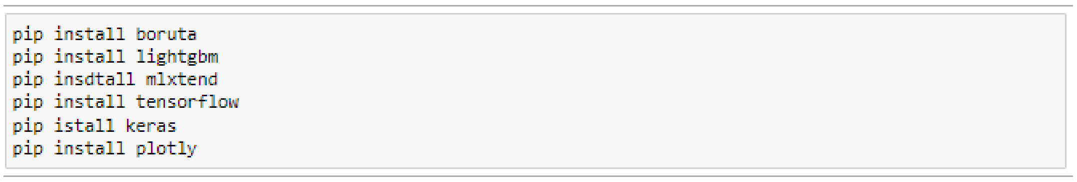


```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from boruta import BorutaPy
from lightgbm import LGBMClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score , classification_report,ConfusionMatrixDisplay,precision_scc
from sklearn.model_selection import RandomizedSearchCV, cross_val_score, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import QuantileTransformer
from sklearn.linear_model import LogisticRegression
from mlxtend.feature_selection import ExhaustiveFeatureSelector
from scipy.stats import yeojohnson
```

Fig 4: Imported libraries and Packages

To create frameworks for Stroke Prediction Using Model Comparison and Feature Selection, these libraries were imported into the Jupyter Notebook.

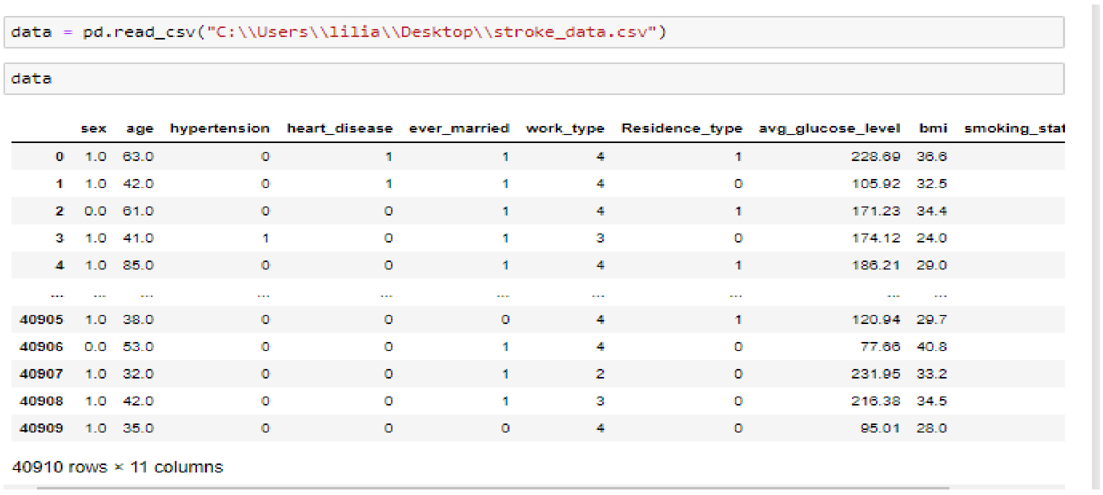
Figure 5 illustrates the process of installing the Boruta, TensorFlow, Keras, lightgbm, mlxtend, and Plotly packages. Installing this package is required to predict stroke.



```
pip install boruta
pip install lightgbm
pip insdtall mlxtend
pip install tensorflow
pip istall keras
pip install plotly
```

Fig 5: Package Installation

As seen in Figure 6, the data is initially inserted through the Data frame and confirmed.



```
data = pd.read_csv("C:\\Users\\lilia\\Desktop\\stroke_data.csv")
```

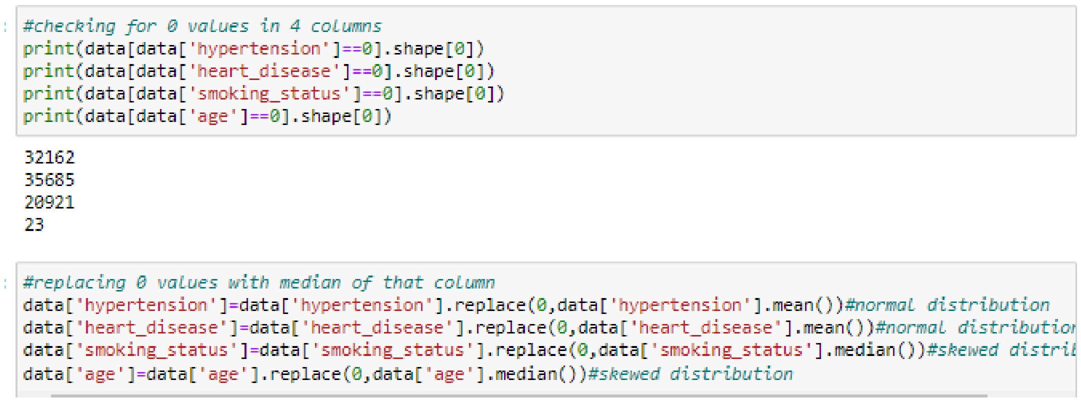
```
data
```

	sex	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_stat
0	1.0	63.0	0	1	1	4	1	228.69	36.6	
1	1.0	42.0	0	1	1	4	0	105.92	32.5	
2	0.0	61.0	0	0	1	4	1	171.23	34.4	
3	1.0	41.0	1	0	1	3	0	174.12	24.0	
4	1.0	85.0	0	0	1	4	1	186.21	29.0	
...	...	...	...	...	...	...	...	...	...	
40905	1.0	38.0	0	0	0	4	1	120.94	29.7	
40906	0.0	53.0	0	0	1	4	0	77.66	40.8	
40907	1.0	32.0	0	0	1	2	0	231.95	33.2	
40908	1.0	42.0	0	0	1	3	0	216.38	34.5	
40909	1.0	35.0	0	0	0	4	0	85.01	28.0	

40910 rows × 11 columns

Fig 6: Caption Data File

6. Checking For Zero Values in some Features and Replacing them with Median



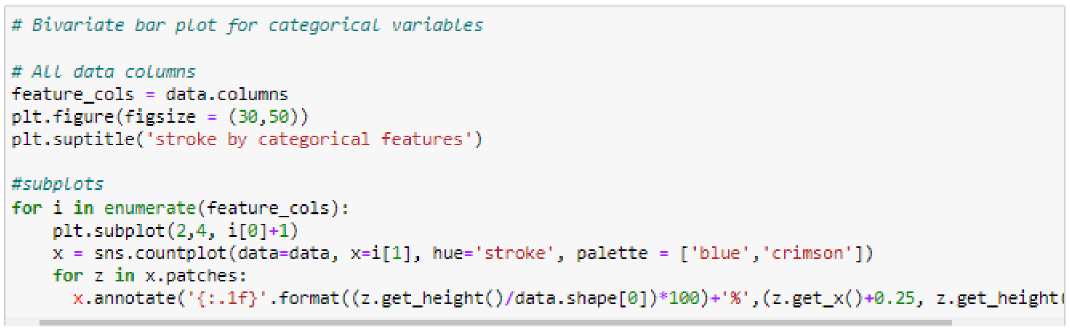
```
#checking for 0 values in 4 columns
print(data[data['hypertension']==0].shape[0])
print(data[data['heart_disease']==0].shape[0])
print(data[data['smoking_status']==0].shape[0])
print(data[data['age']==0].shape[0])
```

```
32162
35685
20921
23
```

```
#replacing 0 values with median of that column
data['hypertension']=data['hypertension'].replace(0,data['hypertension'].mean())#normal distribution
data['heart_disease']=data['heart_disease'].replace(0,data['heart_disease'].mean())#normal distributio
data['smoking_status']=data['smoking_status'].replace(0,data['smoking_status'].median())#skewed distri
data['age']=data['age'].replace(0,data['age'].median())#skewed distribution
```

Fig 7: Checking And Replacing the Zero Values

7. Exploratory Data Analysis

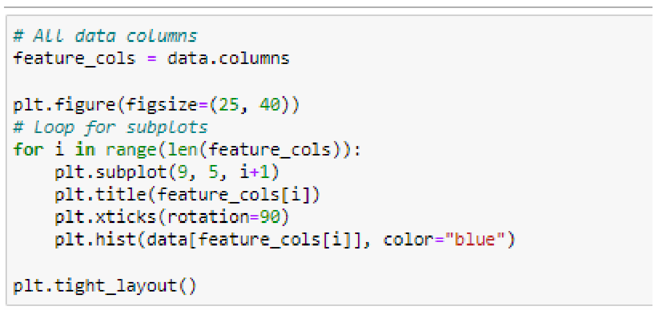


```
# Bivariate bar plot for categorical variables

# All data columns
feature_cols = data.columns
plt.figure(figsize = (30,50))
plt.suptitle('stroke by categorical features')

#subplots
for i in enumerate(feature_cols):
    plt.subplot(2,4, i[0]+1)
    x = sns.countplot(data=data, x=i[1], hue='stroke', palette = ['blue','crimson'])
    for z in x.patches:
      x.annotate('{:.1f}'.format((z.get_height()/data.shape[0])*100)+'%',(z.get_x()+0.25, z.get_height
```

Fig 8: Code for Stroke Proportion Bar Plot

Figure 8 shows the percentage of people who have experienced a stroke compared to those who have not.

7.1. Analysis of All the Features



```
# All data columns
feature_cols = data.columns

plt.figure(figsize=(25, 40))
# Loop for subplots
for i in range(len(feature_cols)):
    plt.subplot(9, 5, i+1)
    plt.title(feature_cols[i])
    plt.xticks(rotation=90)
    plt.hist(data[feature_cols[i]], color="blue")

plt.tight_layout()
```

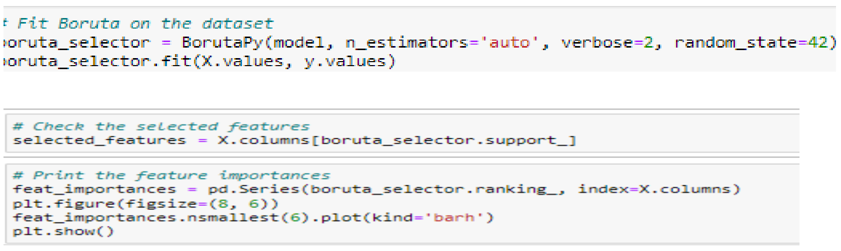
Fig 9: Code for Histograms of all the Features

8. Implementation and Evaluation

8.1. Feature Selection

Three feature selection was used to select important features by eliminating the features that are relevant to stroke predictions.

Boruta Feature selection model



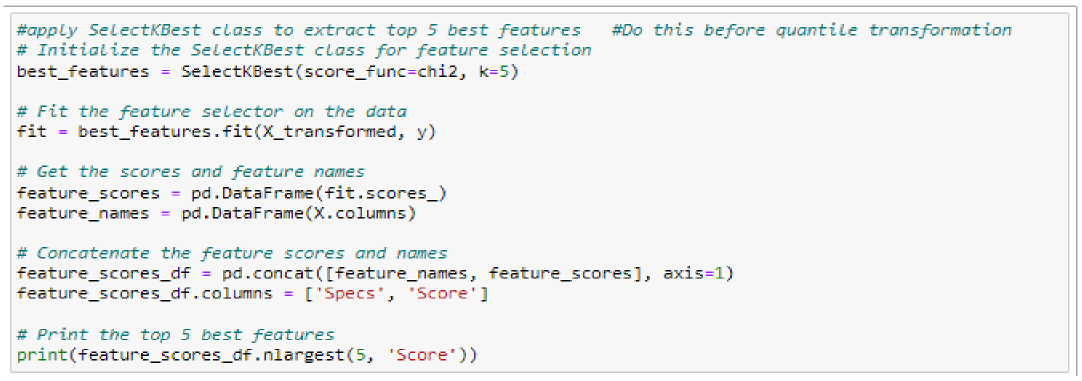
```
t Fit Boruta on the dataset
oruta_selector = BorutaPy(model, n_estimators='auto', verbose=2, random_state=42)
oruta_selector.fit(X.values, y.values)
```

```
# Check the selected features
selected_features = X.columns[boruta_selector.support_]
```

```
# Print the feature importances
feat_importances = pd.Series(boruta_selector.ranking_, index=X.columns)
plt.figure(figsize=(8, 6))
feat_importances.nsmallest(6).plot(kind='barh')
plt.show()
```

Fig 10: Code for implementing Boruta.

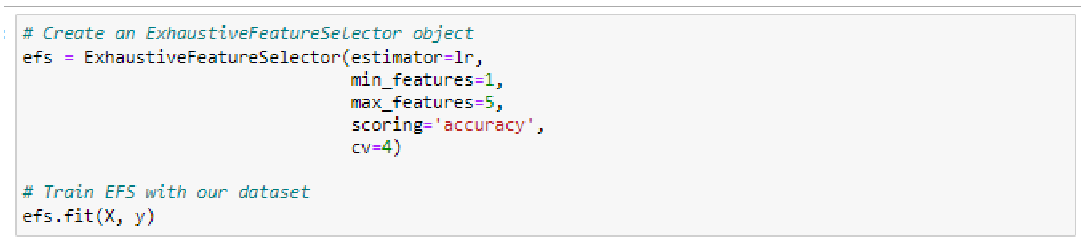
SelectKBest Feature Selection Model



```
#apply SelectKBest class to extract top 5 best features    #Do this before quantile transformation
# Initialize the SelectKBest class for feature selection
best_features = SelectKBest(score_func=chi2, k=5)

# Fit the feature selector on the data
fit = best_features.fit(X_transformed, y)

# Get the scores and feature names
feature_scores = pd.DataFrame(fit.scores_)
feature_names = pd.DataFrame(X.columns)

# Concatenate the feature scores and names
feature_scores_df = pd.concat([feature_names, feature_scores], axis=1)
feature_scores_df.columns = ['Specs', 'Score']

# Print the top 5 best features
print(feature_scores_df.nlargest(5, 'Score'))
```

Fig 11: Code for Implementing SelectKBest.

Exhaustive Feature Selection Model

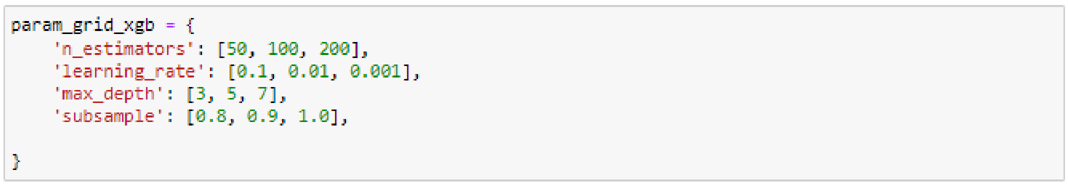


```
# Create an ExhaustiveFeatureSelector object
efs = ExhaustiveFeatureSelector(estimator=lr,
                                min_features=1,
                                max_features=5,
                                scoring='accuracy',
                                cv=4)

# Train EFS with our dataset
efs.fit(X, y)
```

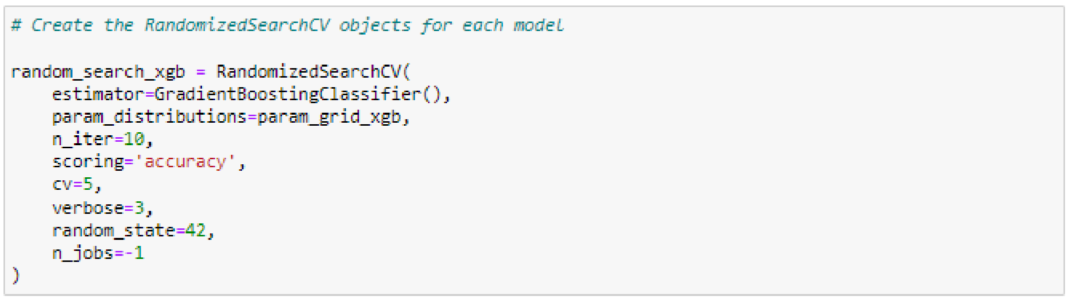
Fig 12: Code for Implementing Exhaustive Model

All the models were hyper-tuned with RandomSearchCV and these are the codes we used on each model and their estimators

Gradient Boosting Classifier (XGB)



```
param_grid_xgb = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.1, 0.01, 0.001],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 0.9, 1.0],

}
```

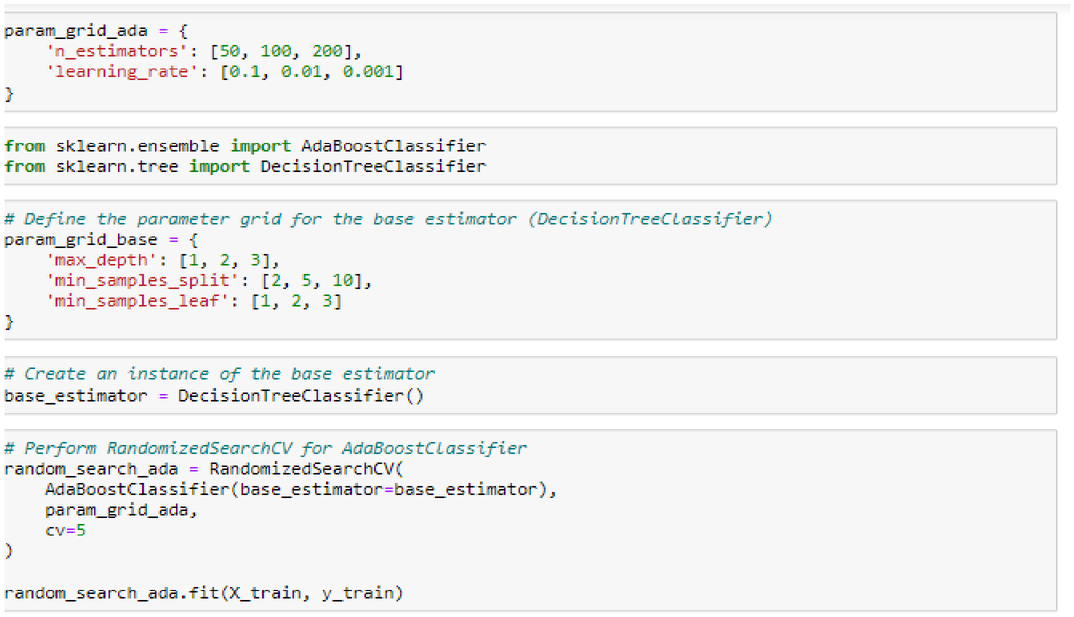


```
# Create the RandomizedSearchCV objects for each model

random_search_xgb = RandomizedSearchCV(
    estimator=GradientBoostingClassifier(),
    param_distributions=param_grid_xgb,
    n_iter=10,
    scoring='accuracy',
    cv=5,
    verbose=3,
    random_state=42,
    n_jobs=-1
)
```

Fig 13: XGB Hyperparameter Tuning.

Figure 13 shows the code for utilizing RandomSearchCV and 10-fold cross-validation to discover the optimal parameters for XGB. The n_estimator for XGB is 100.

AdaBoosting Classifier (Adaboost)



```
param_grid_ada = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.1, 0.01, 0.001]
}
```

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
# Define the parameter grid for the base estimator (DecisionTreeClassifier)
param_grid_base = {
    'max_depth': [1, 2, 3],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 3]
}
```

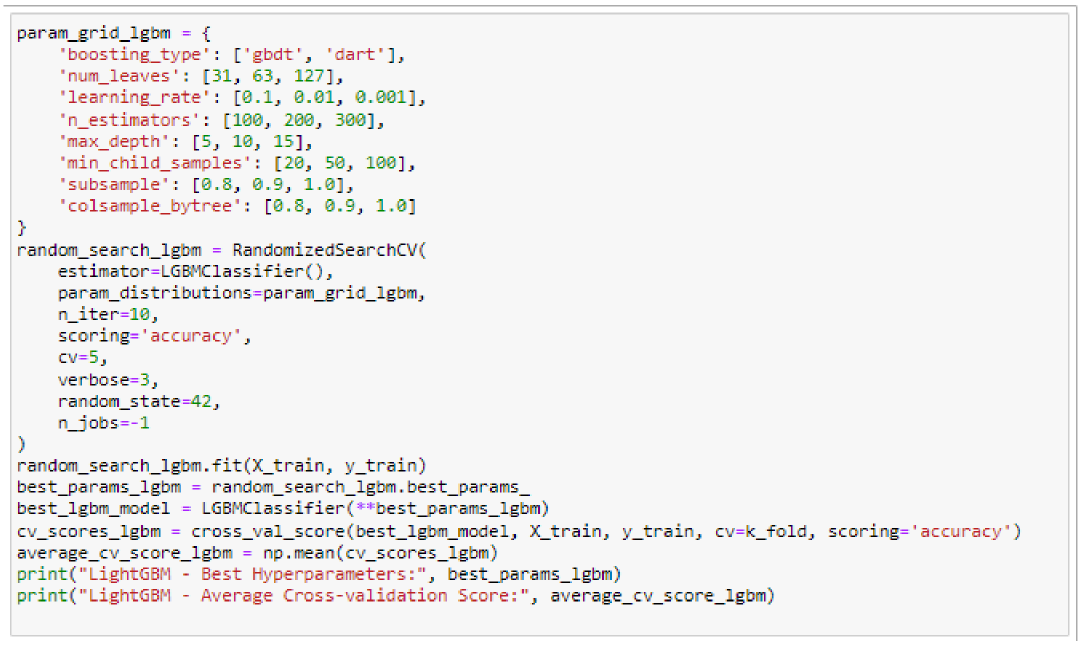
```
# Create an instance of the base estimator
base_estimator = DecisionTreeClassifier()
```

```
# Perform RandomizedSearchCV for AdaBoostClassifier
random_search_ada = RandomizedSearchCV(
    AdaBoostClassifier(base_estimator=base_estimator),
    param_grid_ada,
    cv=5
)

random_search_ada.fit(X_train, y_train)
```

Fig 13: Adaboost Hyperparameter Tuning.

Figure 13 shows the code for utilizing RandomSearchCV and 10-fold cross-validation to discover the optimal parameters for the Adaboost classifier. The n_estimators for the Adaboost classifier is 50.

Light Gradient Boosting Classifier (LightGBM)

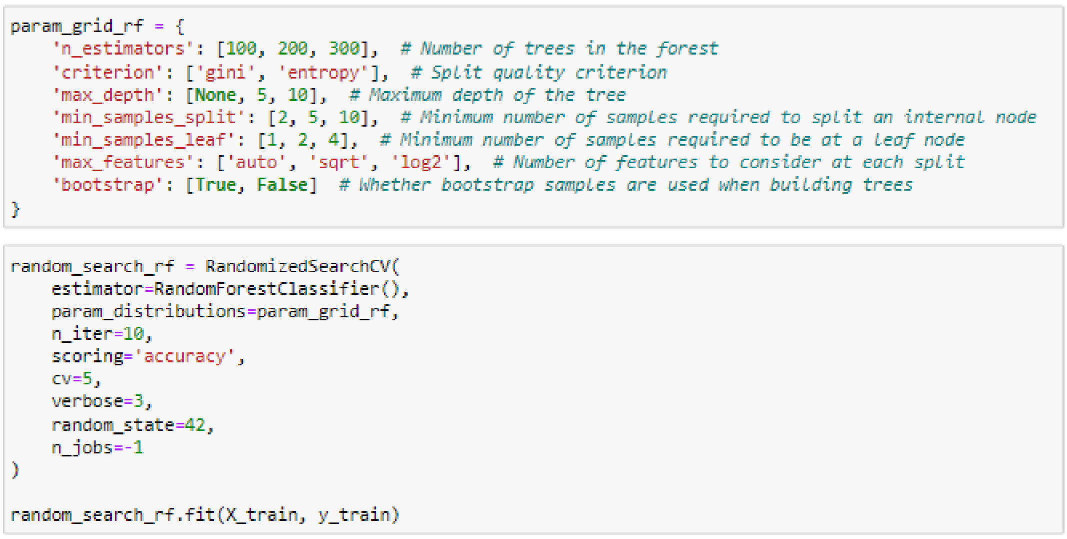


```
param_grid_lgbm = {
    'boosting_type': ['gbdt', 'dart'],
    'num_leaves': [31, 63, 127],
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 15],
    'min_child_samples': [20, 50, 100],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}
random_search_lgbm = RandomizedSearchCV(
    estimator=LGBMClassifier(),
    param_distributions=param_grid_lgbm,
    n_iter=10,
    scoring='accuracy',
    cv=5,
    verbose=3,
    random_state=42,
    n_jobs=-1
)
random_search_lgbm.fit(X_train, y_train)
best_params_lgbm = random_search_lgbm.best_params_
best_lgbm_model = LGBMClassifier(**best_params_lgbm)
cv_scores_lgbm = cross_val_score(best_lgbm_model, X_train, y_train, cv=k_fold, scoring='accuracy')
average_cv_score_lgbm = np.mean(cv_scores_lgbm)
print("LightGBM - Best Hyperparameters:", best_params_lgbm)
print("LightGBM - Average Cross-validation Score:", average_cv_score_lgbm)
```

Fig 14: LightGBM Hyperparameter tuning.

Figure 14 shows the code for utilizing RandomSearchCV and 10-fold cross-validation to discover the optimal parameters for the LightGBM classifier. The n_estimators for the LightGBM classifier is 300.

Random Forest (RF)



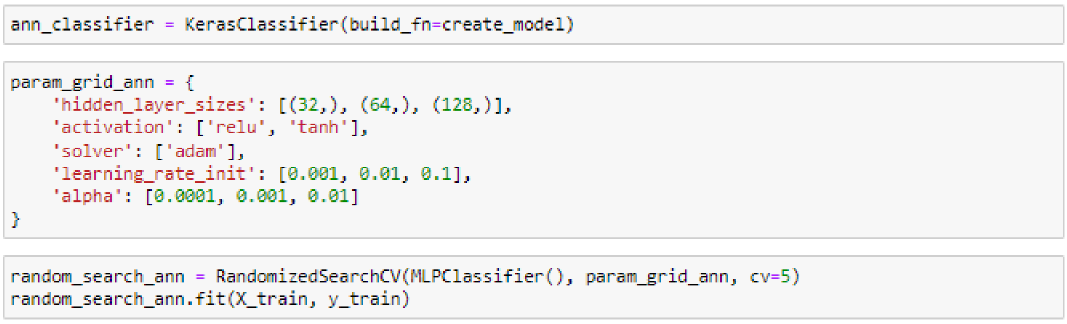
```
param_grid_rf = {
    'n_estimators': [100, 200, 300],  # Number of trees in the forest
    'criterion': ['gini', 'entropy'],  # Split quality criterion
    'max_depth': [None, 5, 10],  # Maximum depth of the tree
    'min_samples_split': [2, 5, 10],  # Minimum number of samples required to split an internal node
    'min_samples_leaf': [1, 2, 4],  # Minimum number of samples required to be at a leaf node
    'max_features': ['auto', 'sqrt', 'log2'],  # Number of features to consider at each split
    'bootstrap': [True, False]  # Whether bootstrap samples are used when building trees
}
```

```
random_search_rf = RandomizedSearchCV(
    estimator=RandomForestClassifier(),
    param_distributions=param_grid_rf,
    n_iter=10,
    scoring='accuracy',
    cv=5,
    verbose=3,
    random_state=42,
    n_jobs=-1
)

random_search_rf.fit(X_train, y_train)
```

Fig 15: Random Forest Hyperparameter Tuning

Figure 15 shows the code for utilizing RandomSearchCV and 10-fold cross-validation to discover the optimal parameters for the Random Forest classifier. The n_estimators for the Random Forest classifier is 200.

Artificial Neural Networks (ANN)



```
ann_classifier = KerasClassifier(build_fn=create_model)
```

```
param_grid_ann = {
    'hidden_layer_sizes': [(32,), (64,), (128,)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam'],
    'learning_rate_init': [0.001, 0.01, 0.1],
    'alpha': [0.0001, 0.001, 0.01]
}
```

```
random_search_ann = RandomizedSearchCV(MLPClassifier(), param_grid_ann, cv=5)
random_search_ann.fit(X_train, y_train)
```

Fig 16: Artificial Neural Network Hyperparameter Tuning

Figure 15 shows the code for utilizing RandomSearchCV and 10-fold cross-validation to discover the optimal parameters for the Artificial Neural Networks.

9. Conclusion

You can adhere to the following steps to carry out the complete piece of code in Jupyter successfully. The system's 12 gigabytes of random-access memory guarantee that the code will run more swiftly and without hiccups.

10. References

Dritsas, E. and Trigka, M., 2022. Stroke risk prediction with machine learning techniques. Sensors, 22(13), p.4670. Sailasya, G. and Kumari, G.L.A., 2021. Analyzing the performance of stroke prediction using ML classification algorithms. International Journal of Advanced Computer Science and Applications, 12(6)