SOFT COMPUTING

ITE1015

Prediction of Heart Disease Occurrence

By

Atchi Sumanth Raju (19BIT0033)

Chaitanya Singh (19BIT0044)

Under the guidance of

Prof. Balakrushna Tripathy

School of Information Technology

and

Engineering

Fall Semester 2021-22

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Abstract:

AI, a significant fragment of man-made reasoning, has begun to pervade different businesses, among which medical care is unmistakable. Around here, precise expectations of the presence or nonappearance of coronary illness and substantially more are right now created by calculations. Such data can give significant bits of knowledge to clinicians who tailor their finding and treatment to patient-explicit requirements when shown well ahead of time. The current situation is that the medical services industry is gathering tremendous measures of information. All things considered, tragically, not all information is being gathered to find covered up examples and settle on successful choices. Hence, there are critical deviations from the specific worth in the expectations. This undertaking plans to foster a site where clients can give their information on wellbeing determinants, for example, circulatory strain and drug, smoking or drinking propensities, weight file, pulse, uneasiness, yellow fingers, and different capacities that are significant Assume parts in anticipating cellular breakdown in the lungs and coronary illness. The gathered information is exposed to a forecast of the "10-year risk" for the event of coronary illness.

Keywords – Support Vector Machine, Decision Tree Classifier, Logistic Regression, KNN classifier Naive Bayes, Artificial Neural Networks, K-star algorithm, etc.

Introduction

In the US and other underdeveloped nations, half of the passing are caused because of cardiovascular illnesses. Likewise, in numerous nations driving reason for death is a coronary illness. Among many sorts of coronary illnesses, coronary illness prompted the most elevated number of fatalities as these sicknesses happen unexpectedly, or as a rule, they are being analyzed at the last stages, where the patients and specialists are defenseless to fix the infection.

Thus, we thought of this venture thought of making a site with a great UI and more exact expectations of these illnesses to perceive the condition in the beginning stage itself and take gauges appropriately. Lab aides should utilize innovation for business as well as for the better living individuals. Coronary illness Forecast, Backing vector classifier, Arbitrary Woodland Classifier, Slope sponsor classifier, XG Lift classifier, strategic relapse.

This venture objective is to foster a site in which clients can give their information of wellbeing factors like Circulatory strain levels and Prescription, Smoking/Drinking propensities, Weight List, Pulse, Nervousness, Yellow Fingers, and different elements that play a fundamental part in the expectation of coronary illness and more other normal these days factors for this event. The information gathered will go through the expectation of "10-Year Risk" for the occasion of coronary illness.

Nowadays, cardiovascular disease is maybe the most fundamental issue relating to human prosperity. - the treatment of heart issues has actually been communicated in an examination that has gotten tremendous thought in the clinical structure all throughout the planet. Heart contaminations are perhaps the most boss explanations behind death all throughout the planet. At the center, 17.7 million passing results from heart contamination which implies about 31% all through the world in 2016,

as shown by World Prosperity Affiliation (WHO). In the US and other made countries, about segment, in light of everything, are achieved by coronary sickness; in like manner, 33% surprisingly passing generally speaking are related to coronary disease. Heart ailment impacts individuals' prosperity just as the economies and costs of countries too. - the most ordinary heart issues are those of microvascular starting, on a very basic level cardiovascular issues and stroke.

PROBLEM IDENTIFIED:

Being proactive is better than being reactive. What if there is a system to predict the future outcome? And what if it is predicting something about a disease with the help of top-notch data sets and cutting-edge technologies? Many diseases get severe when it is not properly medicated before it starts to spread/grow. Heart disease can be controlled and sustained more effectively with proper food habits, lifestyle, medicine and exercise. Predicting the like hood for diseases like these, would be very much helpful in taking precautionary steps and also to cure them. The predicted outcome can be used to prevent/control these diseases and prove to be a great system in the field of medical science.

H/W REQUIREMENTS (details about Application Specific Hardware)

- Laptop
- Internet
- i5 Processor Based Computer or higher

S/W REQUIREMENTS (details about Application Specific Software)

- Front-End: HTML, CSS, BOOTSTRAP
- Back-End: MYSQL
- ML model training: PYTHON – Scikit-Learn/Keras(Google Colab)
- ML model Deployment: Flask

LITERATURE REVIEW:

Authors	Methodologies or Techniques	Advantages	Issues	Metrics used
Mythili T, Dev Mukherjee, Nikhitha Padiala, Abhiram Naidu (2013)	Support Vector Machine, decision trees, logistic regression	proposes a rule-based model to compare the accuracies of applying rules to the individual results of applied algorithms.	When an extensive dataset is used for prediction, SVM cannot perform well, and also training time required is more.	accuracy
Sharma Purushottam , Dr. Kanak Saxena, Richa Sharma (2015)	Decision tree classifier	This system helps medical practitioners to make effective decision-making based on certain parameters.	Decision trees are inclined to errors in problems with many classes of the classification Accuracy for testing and training phase	Accuracy for testing and training phase
Boshra Brahmi (2015)	K-nearest neighbor, Sequential Minimal Optimization, Naive Bayes algorithm, J48 (type of decision tree algorithm)	generated various data mining techniques to evaluate the diagnosis and prediction of coronary heart disease	The process is time-consuming. ANN shows results significantly for heart disease prediction.	accuracy
Noura Ajam (2015)	Artificial Neural Networks included with the back-propagation algorithm	innovate its own method of the information from where it receives during training time and can solve complex tasks	Here the neural network is provided only with inputs, but there are no known targets given to this method.	accuracy

Wiharto Wiharto, Mcom, Hari Kusnanto, DrPH, Herianto, DrEng (2016)	K-Star Algorithm	the problem of data imbalance is accounted	most of the data available between the level/type of coronary heart disease are unbalanced and also it is a binary classifier. Low System Performance	F-measure
Mr. P Sai Chandrasekhar Reddy, Mr. Puneet Palagi , Ms. Jaya (2017)	ANN algorithm	ability to implicitly determine complex nonlinear relationships among independent and dependent variables	Doesn't automatically detects important features without any human supervision.	accuracy
SP Rajamhoana, C Kalyan Devi, Uma Maheswari, Kiruba (2018)	Artificial neural network using feature selection	When feature selection is applied, to decrease the search space, greedy-based sequential forward & backward selection is used.	Techniques for predicting and classifying cardiovascular disease are time-consuming.	accuracy
Bo Jin, Chao Che et al (2018)	EHR Sequential Data Modeling	It is difficult to use the existing Electronic Health Record data directly because the data is sparse and non-standardized. Sequentially, this paper generated a robust and effective architecture for the prediction of heart failure	Leverage similar representations of medical concepts through word vectors, but they focused on temporal modelling in the use of LSTM to predict heart failure.	accuracy

Antonio Vilasi, Pasquale De Meo, Giacomo Fiumara, Sabrina Mezzatesta, Claudia Torino(2019)	non-linear SVC with RBF kernel algorithm	more effective in high dimensional spaces and less time consuming	the target variable was available but some 167 patient features were missing.	accuracy
Avinash Golande, Pavan Kumar T (2019)	Decision Tree, KNN, K-mean clustering, and Adaboost	investigation tells us about different technologies that are used in different papers with a different count of attributes with different accuracies turn on the tools that are designed for execution.	Decrease of accuracy rate due to fewer combinations of data mining techniques	accuracy
Khaled Mohamad Amulstafa (2020)	KNN, SVM, Adaboost, SGD, and Decision Table	Using contrasting classification algorithms for the categorization of Hierarchical datasets gives promising results in terms of classification accuracy	set holds 76 attributes which include the class attribute but in this paper, only a subset of 14 attributes are implemented for heart disease prediction	accuracy
Samir Patel, Devansh Shah, Santosh Kumar Bharti (2020)	K-nearest neighbor, Naïve Bayes, random forest algorithm, and decision tree	It is a substitute to the routine prediction modeling approach using a computer to gain an understanding of complex and non-linear interactions among different factors by	consider only 14 essential attributes and there is a need to implement a more complex and combination of models	accuracy

		reducing the errors in predicted and factual outcomes.		
Nilam Harkulkar, Swati Nadkarni , Dr. Bhavesh Patel (2020)	CNN Deep Learning Model	it spontaneously detects the essential features without any human supervision. CNN is also computationally efficient	Lack of ability to be spatially invariant to the input data and the work can be extended to study ensemble models or combining different parameters for hidden layers	accuracy
Sarthak Agrawal, Harshit Jindal, Rachna Jain, Rishabh Khera(2021)	logistic regression, random forest classifier, and KNN	This heart disease prediction system intensifies medical care and brings down the cost by using computer-aided approaches.	With large data, the prediction stage might be slow and also Require high memory to store all of the training data.	accuracy
Xiao-Yan Gao,Abdelmegeid Amin Ali,Hassan Shaban Hassan Eman M. Anwar (2021)	KNN, Support vector machine, Random forest, Decision tree, Naive Bayes	Entity methods which are known as bagging and boosting with feature extraction algorithms i.e. PCA and LDA (Principal Component Analysis and Linear Discriminant Analysis respectively)that are used to build predicting heart disease performance.	Researchers have declared that there are no issues with the system	Accuracy, ROC, F-measure, precision, recall
Jaymin Patel, Samir Patel, TejalUpadhyay	J48 algorithm, Logistic model tree algorithm, Random	J48 algorithm utilizes an acquisitive method to form decision trees for the	prediction system should not presuppose any prior knowledge regarding	accuracy

(2013)	Forest algorithm, WEKA Tool	classification and also uses for decreased-error pruning.	patient records from which it is contrasting.	
R.Chitra and V.Seenivasagam (2013)	"Classification" technique used as data mining technique, "Neural Network" used as an intelligent technique. SLM	The proposed hybrid intelligent algorithm magnifies the accuracy in predicting coronary heart disease. The proposed system will provide an aid for the physicians to diagnose the disease in a more efficient way.	Very high cost and takes Pre-processing of data is no less than a big challenge.	accuracy
Jayshri S. Sonawane , D.Patil (2014)	Learning vector Quantization neural network algorithm	The neural network during this system accepts 13 clinical features as input and predicts that there's a presence or absence of heart condition within the patient, along side different performance measures.	There are many chances of errors, unwanted biases and also it takes longer time in accurate diagnosis of disease. The diagnosis of coronary heart disease is strenuous	accuracy
Sivagowry.S , Dr.Durairaj.M (2014)	Particle Swarm Optimization (PSO) algorithm and ANN	The PSO is much easier than genetic algorithm to lay into operation. In PSO, each particle can adjusts its flying memory and it can also adjust its companion's flying involvement so that to fly within the search space with velocity.	reducing the amount of attributes without affecting the accuracy is that the problem taken to review	ROC Value
Ilayaraja M,	Frequent Item sets	Association rule mining is	Applied only for small	Risk level

Meyyappan T (2015)	Association Rule Mining Medical Data Mining	a most efficient algorithm for extracting frequent item sets from huge data and avoids the generation of unnecessary item sets	datasets (<500)	(using minimum support value)
Prachi Paliwal and Mahesh Malviya (2015)	Introduced a new method based on fitness value. 10 attributes were used which are responsible for heart disease.	In the proposed method they use the concepts of fitness number. In this approach as per the given condition for heart attack the given data can convert into binary format	Taken only ten attributes	accuracy
Ashwini Shetty A1 , Chandra Naik (2016)	Neural Network and Genetic Algorithm	The hybrid system which uses global optimization which is also a advantage of Genetic Algorithm in the initializing the Neural Network weight.	Less attributes are considered	accuracy
Prajakta Ghadge, Vrushali , Kajal, Prajakta Deshmukh (2016)	Big Data solution	big data infrastructure gives for both predictive modeling and information extraction	Refers to a particular size of dataset	accuracy
Azam Dekamin, Ahmad Shaibatalhamdi (2017)	hybrid model with Naive Bayes learners, decision tree, and K-nearest algorithms, RMS	A combination of classification algorithms and ensemble methods were applied to develop a prediction with gives only very few errors.	Takes much time and cost of experiment is high	accuracy
Shashikanth R.	Logistic regression,	Increased accuracy for	Prediction of disorder	accuracy

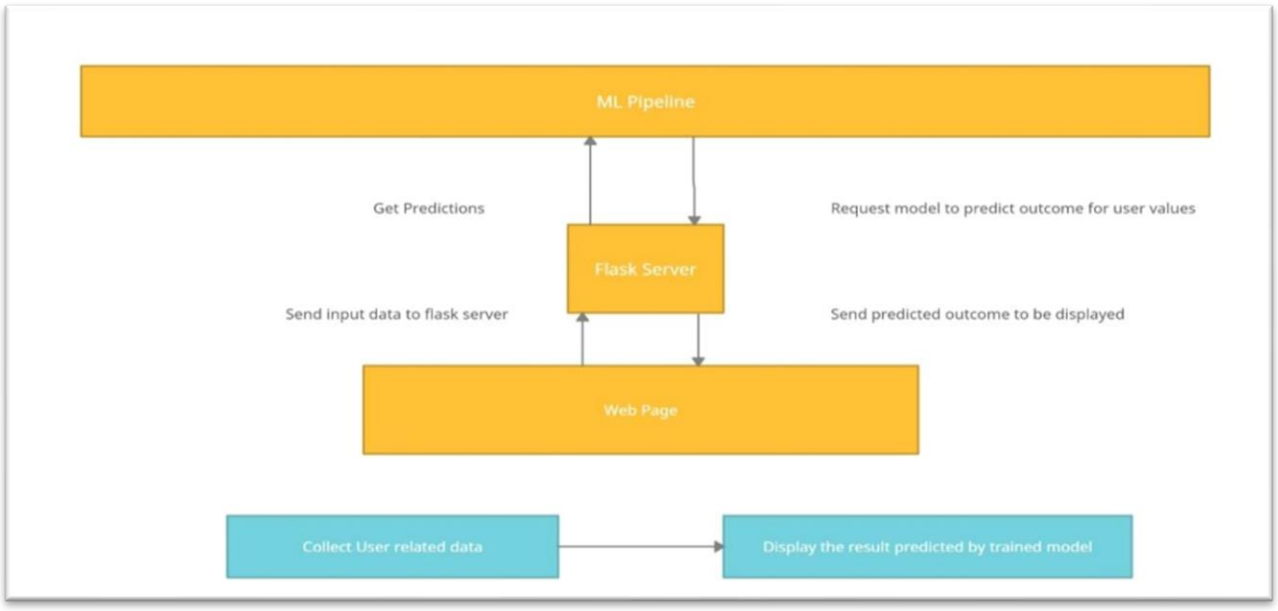
and Chetankumar P. (2019)	Decision tree, Random Forest	effective heart disease diagnosis. Reduce the time complexity of doctors.	results isn't accurate.	
Galla Siva Sai Bindhika, Rajalakshmi, Manchuri Sathvika Reddy, Munaga Meghana (2020)	Random forest Linear model	RFA helps in early prediction of the disease and is used in many ways, where as it is being provided with the input, in order to find the heart rate based on the health condition.	Prediction of disorder results isn't accurate.	F-measure
Apurb Rajdhan, Avi Agarwal, Dundigalla Ravi, Milan Sai. (2020)	Naive Bayes, Decision Tree, Logistic Regression and Random Forest classifier	It is a substitute to routine prediction modeling approach employing a computer to realize an understanding of complex and non-linear interactions among various factors by reducing the errors in predicted and factual outcomes	Did not do web application based on Random Forest Algorithm as well as for large dataset	accuracy, Recall,Precision, Recall and f-measure scores
Anusha MB, Chaitra K , Chandana HM , Kiran G, Swathi (2020)	KNN N Naïve bayes , Support vector machine, logistic regression , weighted KNN	Repeating feature selection and modelling for various combinations of attributes	No combinational and complex models to increase the accuracy of predicting the early onset of heart disease.	Risk level
Li Yang, Haibin Wu, Xiaoqing Jin, Pinpin Zheng,	Random forest Multivariate regression model	Random Forest used here had the benefits that unlike most ML algorithms, it could accept dirty data,	The main limitation of this study is that it was in lacked external validation.	AUC

Shiyun Hu, Jing Yan, Xiaoling Xu,Wei Yu. (2020)		and in contrast to some traditional regression models, it also could model nonlinear relations and accept both regression and classification problems at meanwhile.		
Armin Yazdani, Kasturi Dewi Varathan, Asad Waqar Malik & Wan Azman Wan Ahmad, Yin Kia Chiam (2021)	proposes an algorithm that measures the strength of the significant features that contribute to heart disease prediction using Weighted Associative Rule Mining.		The machine learning techniques utilized in feature selection phase is restricted to the foremost popular techniques utilized in heart condition prediction.	confidence score

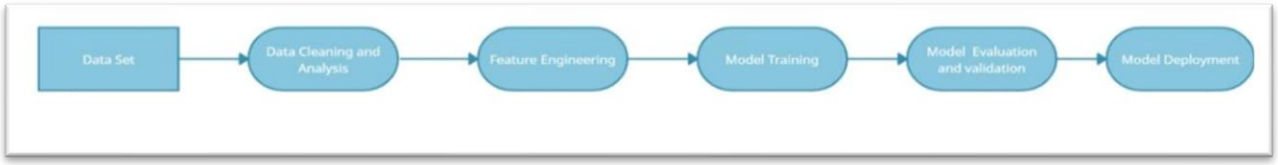
Gaps in Literature:

In the research papers mentioned above, we've found various kinds of drawbacks like lower accuracy was detected due to lack of sufficient number of attributes in the dataset. The models seemed to be more time consuming due to more pre-processing time. Some models require high memory to store all of the training data. High cost is also a major problem identified in the papers. No combinational and complex models were used to increase the accuracy of predicting the early onset of heart disease.

SYSTEM ARCHITECTURE



WORKFLOW



CONTENT MODULES:

Libraries used – seaborn, sklearn, matplotlib, pandas, numpy, pickle

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It builds on top of matplotlib and integrates closely with pandas data structures.

Pickle module implements binary protocols for serializing and de-serializing a Python object structure. *"Pickling"* is the process whereby a Python object hierarchy is converted into a byte stream, and *"unpickling"* is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

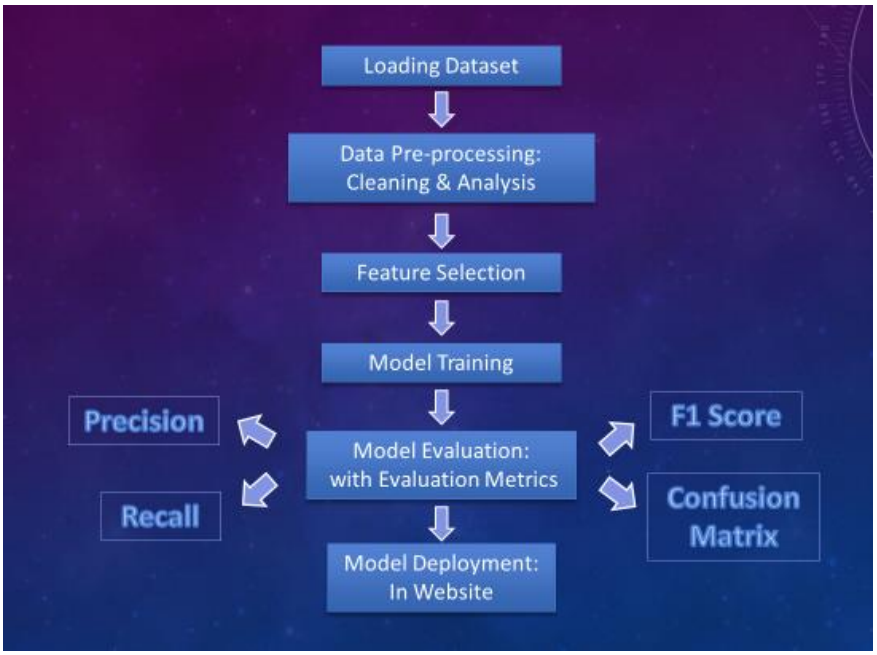
NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones powerful N-dimensional array object, Sophisticated (broadcasting) functions, Tools for integrating C/C++ and Fortran code and Useful linear algebra, Fourier transform, and random number capabilities.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It offers vareity of functionalities like make interactive figures that can zoom, pan, update, customize visual style and layout, export to many file formats, embed in JupyterLab and Graphical User Interfaces and can also use a rich array of third-party packages built on Matplotlib.

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

ALGORITHMIC STEPS:



MODULES AND ITS DESCRIPTION

- **Loading Dataset:** We are using "pandas" a machine learning library in our project for further processing. This loads the dataset.
- **Data pre-processing:** It is an important process as it ensures valid data is given to machine to learn. So, any null values in data are replaced with other values like mean, median or less-dominant value in order to balance the dataset and ensuring result is un-biased. We had to do 'Feature Scaling' using 'Standardization' technique. This technique rescales value such that it has distribution with 0 mean value and variance equal to 1.
- **Feature Engineering:** This is used to select particular features which are more impactful on result and which improves the performance.
- **Training the Model:** We train our model with a dataset of 4238 records and Supervised ML algorithms are used to train the model and their respective F1-Scores are recorded.

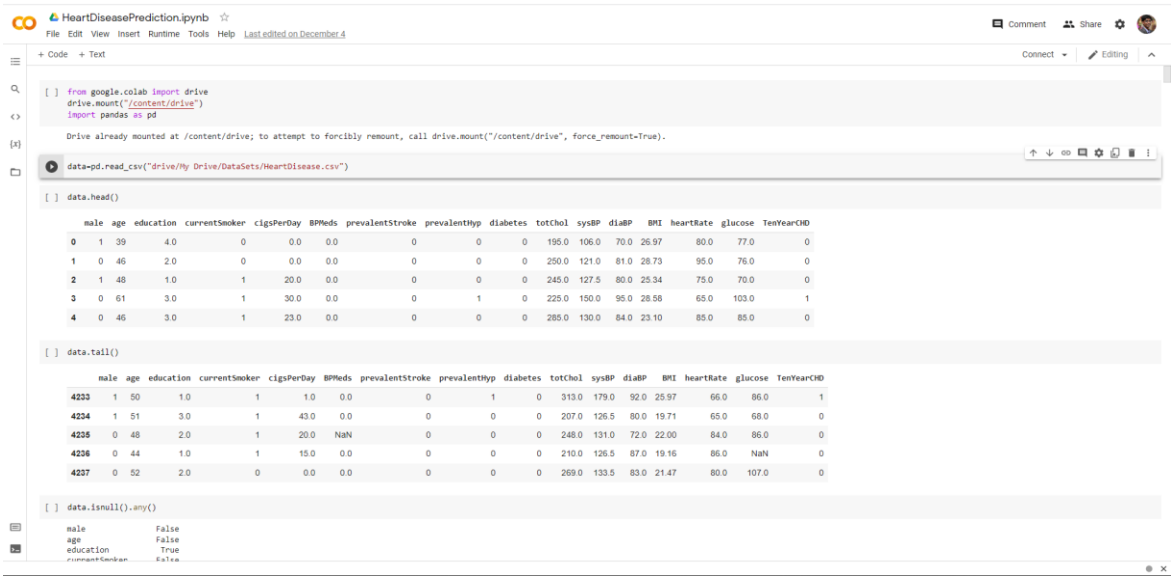
- **Testing the M EVALUATION METRICS:**

1. Precision = True Positive / (True Positive + False Positive)
2. Recall = True Positive / (True Positive + False Negative)
3. F1-score = (2 * (Precision * Recall)) / (Precision + Recall)
4. Confusion Matrix

- **Model Deployment**: Convert .ipynb file to pickle file(.pkl) and load the file in “app.py”. We collect data from user through web-interface, data collected is passed through “Flask” to the trained model as input and the result predicted by the model is displayed to user.

Snapshots:

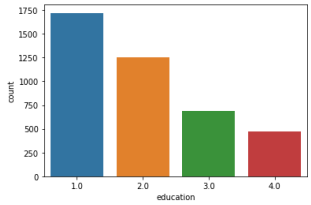
Pre-processing



+ Code + Text

```
[ ] sns.countplot(data['education'])
```

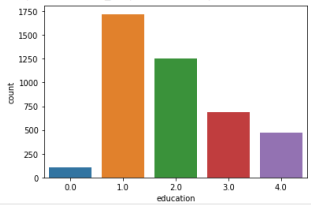
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f169e91d7d0>
```




```
[ ] data['education']=data['education'].fillna(0)
```

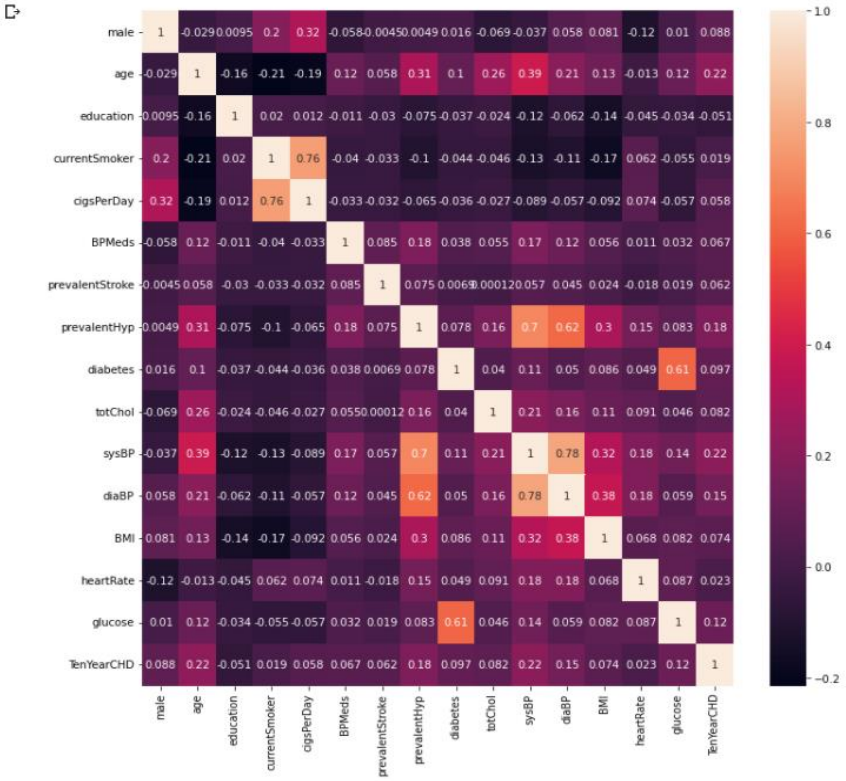
```
[ ] sns.countplot(data['education'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f169efdf290>
```



Correlation Analysis

```
import seaborn as sns
cor=data.corr()
plt.figure(figsize=(12,12))
ax=sns.heatmap(cor,annot=True)
```



Data transformation and splitting

```
[ ] from sklearn.preprocessing import StandardScaler
    scaler=StandardScaler()
    x=scaler.fit_transform(x)

[ ] from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state = 42)

    x_train.shape

    (3389, 15)

[ ] x_test.shape

    (848, 15)

[ ] import sklearn
    sorted(sklearn.metrics.SCORERS.keys())

    ['accuracy',
     'adjusted_mutual_info_score',
     'adjusted_rand_score',
     'average_precision',
     'balanced_accuracy',
     'completeness_score',
     'explained_variance',
     'f1',
     'f1_macro',
     'f1_micro',
     'f1_samples',
     'f1_weighted',
     'fowlkes_mallows_score',
     'homogeneity_score',
     'jaccard',
     'jaccard_macro',
     'jaccard_micro',
     'jaccard_samples',
     'jaccard_weighted',
     'max_error',
     'mutual_info_score',
     'neg_brier_score',
     'neg_log_loss',
     'neg_mean_absolute_error',
     'neg_mean_absolute_percentage_error',
     'neg_mean_gamma_deviance',
     'neg_mean_poisson_deviance',
```

Decision Tree Classifier

```
[ ] from sklearn.tree import DecisionTreeClassifier
    decision_tree=DecisionTreeClassifier()
    scores_decisionTree=cross_val_score(decision_tree,x,y,cv=10,scoring="f1_micro")

[ ] scores_decisionTree

    array([0.74056604, 0.76886792, 0.74528302, 0.75943396, 0.74292453,
           0.74528302, 0.73584906, 0.76595745, 0.73286052, 0.73049645])

[ ] scores_decisionTree.mean()

    0.7467521967973594

[ ] models.append("Decision Tree")
    f1Scores.append(scores_decisionTree.mean())

[ ] decision_tree2=DecisionTreeClassifier(criterion="entropy",min_samples_split=10,splitter="random")
    scores_decisionTree2=cross_val_score(decision_tree2,x,y,cv=10,scoring="f1_micro")

    scores_decisionTree2

    array([0.79245283, 0.81367925, 0.75235849, 0.80896226, 0.78537736,
           0.79716981, 0.81839623, 0.81323877, 0.79905437, 0.80378251])

    scores_decisionTree2.mean()

    0.7984471876533299

[ ] models.append("Decision Tree 2")
    f1Scores.append(scores_decisionTree2.mean())
```

Logistic Regression and SVM Classifier

```
[ ] from sklearn.model_selection import cross_val_score
    from sklearn.linear_model import LogisticRegression
    logreg=LogisticRegression()
    scores=cross_val_score(logreg,x,y,cv=10,scoring="f1_micro")
```

```
[ ] print(scores)
```

```
[0.84433962 0.86320755 0.85377358 0.86084906 0.8490566  0.8490566
 0.86792453 0.85815603 0.85579196 0.84869976]
```

```
[ ] scores.mean()
```

```
0.8550855301306927
```

```
[ ] models.append("Logitic Regression")
    f1Scores.append(scores.mean())
```

```
[ ] models, f1Scores
```

```
(['Logitic Regression'], [0.8550855301306927])
```

```
[ ] lr=LogisticRegression()
    lr.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
[ ] y_predicted=lr.predict(x_test)
```

```
[ ] score_lr = lr.score(x_test, y_test)
    print(score_lr)
```

```
0.8466981132075472
```

```
[ ] from sklearn.metrics import classification_report
```

```
[ ] print(classification_report(y_test,y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.85      0.99      0.92       714
           1       0.67      0.06      0.11       134
```

```
[ ] from sklearn.svm import SVC
    svm1=SVC(kernel="poly",degree=5,gamma="auto")
    scores_svm1=cross_val_score(svm1,x,y,cv=10,scoring="f1_micro")

[ ] scores_svm1

    array([0.83254717, 0.84669811, 0.83726415, 0.86084906, 0.8254717 ,
           0.84198113, 0.86556604, 0.8534279 , 0.8392435 , 0.82742317])

[ ] scores_svm1.mean()

    0.8430471921138321

[ ] models.append("SVM 1")
    f1Scores.append(scores_svm1.mean())

    svm2=SVC(kernel="rbf")
    scores_svm2=cross_val_score(svm2,x,y,cv=10,scoring="f1_micro")
    print(scores_svm2)
    print(scores_svm2.mean())

    [0.8490566  0.84669811 0.84669811 0.85377358 0.84669811 0.84433962
     0.84433962 0.85106383 0.84869976 0.84869976]
    0.84800671305589

[ ] models.append("SVM 2")
    f1Scores.append(scores_svm2.mean())
```

Random Forest Classifier, Gradient Boosting Classifier and Stacking Classifier

```
[ ] from sklearn.ensemble import RandomForestClassifier
    randomforest=RandomForestClassifier(max_depth=10)
    scores_randomforest=cross_val_score(randomforest,x,y,cv=10,scoring="f1_micro")
    print(scores_randomforest)
    print(scores_randomforest.mean())

    [0.84433962 0.84669811 0.85141509 0.85377358 0.84433962 0.84198113
     0.85377358 0.85579196 0.84869976 0.85106383]
    0.8491876310272536

[ ] models.append("Random Forest")
    f1Scores.append(scores_randomforest.mean())

    from sklearn.ensemble import GradientBoostingClassifier
    gradBoost=GradientBoostingClassifier(loss="exponential",learning_rate=0.001, n_estimators=50, max_depth=10)
    scores_gradBoost=cross_val_score(gradBoost,x,y,cv=10,scoring="f1_micro")
    print(scores_gradBoost)
    print(scores_gradBoost.mean())

    [0.8490566  0.8490566  0.8490566  0.8490566  0.84669811 0.84669811
     0.84669811 0.84869976 0.84869976 0.84869976]
    0.8482420045497123

[ ] models.append("Gradient Boosting Classifier")
    f1Scores.append(scores_gradBoost.mean())

[ ] from xgboost import XGBClassifier
    xgBoost= XGBClassifier(learning_rate=0.01, n_estimators=25, max_depth=15,gamma=0.6, subsample=0.52,colsample_bytree=0.6,seed=27,
                           reg_lambda=2, booster='dart', colsample_bylevel=0.6, colsample_bynode=0.5)
    scores_xgBoost=cross_val_score(xgBoost,x,y,cv=10,scoring="f1_micro")
    print(scores_xgBoost)
    print(scores_xgBoost.mean())

    [0.8490566  0.8490566  0.8490566  0.8490566  0.84669811 0.84669811
     0.84669811 0.84869976 0.84869976 0.84869976]
    0.8482420045497123

[ ] models.append("XG Boost Classifier")
    f1Scores.append(scores_xgBoost.mean())

[ ] from sklearn.ensemble import StackingClassifier
    estimators=[('xgBoost',XGBClassifier(learning_rate=0.01, n_estimators=25, max_depth=15,gamma=0.6, subsample=0.52,colsample_bytree=0.6,seed=27,
```

```
[ ] from sklearn.ensemble import StackingClassifier
    estimators=[('xgBoost',XGBClassifier(learning_rate=0.01, n_estimators=25, max_depth=15,gamma=0.6, subsample=0.52,colsample_bytree=0.6,seed=27,
                         reg_lambda=2, booster='dart', colsample_bylevel=0.6, colsample_bynode=0.5)),
                ('randomforest',RandomForestClassifier(max_depth=10)),
                ('svm',SVC(kernel="rbf"))]
    stackClassifier=StackingClassifier(estimators=estimators,cv=10,final_estimator=LogisticRegression())
    stackClassifier.fit(x_train,y_train)

    StackingClassifier(cv=10,
                       estimators=[('xgBoost',
                                    XGBClassifier(booster='dart',
                                                  colsample_bylevel=0.6,
                                                  colsample_bynode=0.5,
                                                  colsample_bytree=0.6, gamma=0.6,
                                                  learning_rate=0.01, max_depth=15,
                                                  n_estimators=25, reg_lambda=2,
                                                  seed=27, subsample=0.52)),
                                   ('randomforest',
                                    RandomForestClassifier(max_depth=10)),
                                   ('svm', SVC())],
                       final_estimator=LogisticRegression())

[ ] stackClassifier.score(x_test,y_test)

    0.8419811320754716

[ ] y_predicted_stackClass=stackClassifier.predict(x_test)

    print("F1-score of Stacking classifier",f1_score(y_test,y_predicted,average="micro"))

    F1-score of Stacking classifier 0.8466981132075472

[ ] models.append("Stacking Classifier")
    f1Scores.append(0.8466981132075472)

[ ] from sklearn.ensemble import GradientBoostingClassifier
    gradBoost=GradientBoostingClassifier(loss="exponential",learning_rate=0.01, n_estimators=50, max_depth=10,random_state=42)
    gradBoost.fit(x_train,y_train)

    GradientBoostingClassifier(learning_rate=0.01, loss='exponential', max_depth=10,
                               n_estimators=50, random_state=42)

[ ] ypredGradBoost=gradBoost.predict(x_test)
```

Grid Search using different parameters (Hyper-tuning)

```
[ ] from sklearn.model_selection import GridSearchCV

[ ] param_grid1={'n_estimators':range(20,81,10)}
    gridSearch1=GridSearchCV(estimator = GradientBoostingClassifier(learning_rate=0.1, min_samples_split=500,min_samples_leaf=50,max_depth=8,max_features='sqrt',subsample=0.8,random_state=42),
    param_grid = param_grid1, scoring='f1_micro', cv=5)
    gridSearch1.fit(x_train,y_train)

    GridSearchCV(cv=5,
                 estimator=GradientBoostingClassifier(max_depth=8,
                                                      max_features='sqrt',
                                                      min_samples_leaf=50,
                                                      min_samples_split=500,
                                                      random_state=42,
                                                      subsample=0.8),
                 param_grid={'n_estimators': range(20, 81, 10)},
                 scoring='f1_micro')

[ ] results=gridSearch1.cv_results_
    print("          Params       MeanTestScore    StdTestScore   Rank")
    for i in range(len(results['params'])):
      print(results['params'][i],'     ',round(results['mean_test_score'][i],4),'    ',round(results['std_test_score'][i],4),'     ',results['rank_test_score'][i])

             Params       MeanTestScore    StdTestScore   Rank
    {'n_estimators': 20}      0.8498      0.0005       7
    {'n_estimators': 30}      0.851       0.0009       5
    {'n_estimators': 40}      0.8516      0.0014       4
    {'n_estimators': 50}      0.8507      0.0025       6
    {'n_estimators': 60}      0.8522      0.0033       1
    {'n_estimators': 70}      0.8516      0.005        3
    {'n_estimators': 80}      0.8516      0.0064       2

[ ] gridSearch1.best_params_, gridSearch1.best_score_

    ({'n_estimators': 60}, 0.8521705598619626)

[ ] param_grid2 = {'max_depth':range(5,16,2), 'min_samples_split':range(20,201,20)}
    gridSearch2=GridSearchCV(estimator = GradientBoostingClassifier(learning_rate=0.1, n_estimators=60, max_features='sqrt', subsample=0.8,random_state=42),
    param_grid = param_grid2, scoring='f1_micro', cv=5)
    gridSearch2.fit(x_train,y_train)

    GridSearchCV(cv=5,
                 estimator=GradientBoostingClassifier(max_features='sqrt',
                                                      n_estimators=60,
                                                      random_state=42,
```

```
[ ] param_grid2 = {'max_depth':range(5,16,2), 'min_samples_split':range(20,201,20)}
    gridSearch2=GridSearchCV(estimator = GradientBoostingClassifier(learning_rate=0.1, n_estimators=60, max_features='sqrt', subsample=0.8,random_state=42),
    param_grid = param_grid2, scoring='f1_micro', cv=5)
    gridSearch2.fit(x_train,y_train)
```

```
GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(max_features='sqrt',
                                                  n_estimators=60,
                                                  random_state=42,
                                                  subsample=0.8),
             param_grid={'max_depth': range(5, 16, 2),
                         'min_samples_split': range(20, 201, 20)},
             scoring='f1_micro')
```

```
results=gridSearch2.cv_results_
print("            Params                      MeanTestScore    StdTestScore   Rank")
for i in range(len(results['params'])):
  print(results['params'][i],'    ',round(results['mean_test_score'][i],4),'    ',round(results['std_test_score'][i],4),'     ',results['rank_test_score'][i])
```

```
             Params                      MeanTestScore    StdTestScore   Rank
{'max_depth': 5, 'min_samples_split': 20}      0.8486      0.0056       6
{'max_depth': 5, 'min_samples_split': 40}      0.8498      0.0072       3
{'max_depth': 5, 'min_samples_split': 60}      0.8498      0.009        2
{'max_depth': 5, 'min_samples_split': 80}      0.8489      0.0051       5
{'max_depth': 5, 'min_samples_split': 100}      0.8463      0.0079      16
{'max_depth': 5, 'min_samples_split': 120}      0.8477      0.007        9
{'max_depth': 5, 'min_samples_split': 140}      0.8457      0.0059      18
{'max_depth': 5, 'min_samples_split': 160}      0.848       0.0058       8
{'max_depth': 5, 'min_samples_split': 180}      0.8475      0.007       10
{'max_depth': 5, 'min_samples_split': 200}      0.8513      0.0081       1
{'max_depth': 7, 'min_samples_split': 20}      0.8421      0.0089       43
{'max_depth': 7, 'min_samples_split': 40}      0.8451      0.0089       22
{'max_depth': 7, 'min_samples_split': 60}      0.8413      0.0049       47
{'max_depth': 7, 'min_samples_split': 80}      0.8451      0.0066       20
{'max_depth': 7, 'min_samples_split': 100}      0.8486      0.0048       7
{'max_depth': 7, 'min_samples_split': 120}      0.8445      0.0077       26
{'max_depth': 7, 'min_samples_split': 140}      0.8457      0.0057       18
{'max_depth': 7, 'min_samples_split': 160}      0.8448      0.0044       24
{'max_depth': 7, 'min_samples_split': 180}      0.8472      0.0063       12
{'max_depth': 7, 'min_samples_split': 200}      0.8472      0.0066       11
{'max_depth': 9, 'min_samples_split': 20}      0.8436      0.0073       32
{'max_depth': 9, 'min_samples_split': 40}      0.8395      0.0092       56
{'max_depth': 9, 'min_samples_split': 60}      0.8416      0.0078       45
{'max_depth': 9, 'min_samples_split': 80}      0.8427      0.0054       40
{'max_depth': 9, 'min_samples_split': 100}      0.8451      0.0052       21
{'max_depth': 9, 'min_samples_split': 120}      0.8445      0.0051       27
{'max_depth': 9, 'min_samples_split': 140}      0.8451      0.0033       23
{'max_depth': 9, 'min_samples_split': 160}      0.8436      0.0076       31
```

```
[ ] gridSearch2.best_params_, gridSearch2.best_score_
```

```
({'max_depth': 5, 'min_samples_split': 200}, 0.8512873470063573)
```

Parameters fixed till now:

n_estimators-50, max_depth-5

```
[ ]
    param_grid3 = {'min_samples_leaf':range(10,81,10),'min_samples_split':range(200,301,20)}
    gridSearch3=GridSearchCV(estimator = GradientBoostingClassifier(learning_rate=0.1, n_estimators=50,max_depth=5,max_features='sqrt', subsample=0.8, random_state=42),
    param_grid = param_grid3, scoring='f1_micro', cv=5)
    gridSearch3.fit(x_train,y_train)
```

```
GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(max_depth=5,
                                                  max_features='sqrt',
                                                  n_estimators=50,
                                                  random_state=42,
                                                  subsample=0.8),
             param_grid={'min_samples_leaf': range(10, 81, 10),
                         'min_samples_split': range(200, 301, 20)},
             scoring='f1_micro')
```

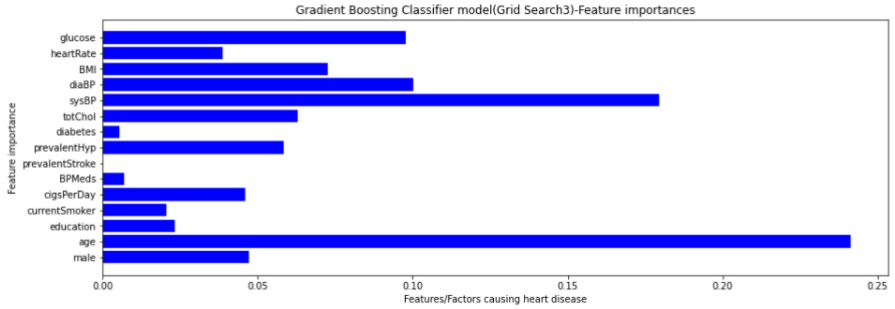
```
[ ] gridSearch3.best_params_, gridSearch3.best_score_
```

```
({'min_samples_leaf': 60, 'min_samples_split': 220}, 0.8542363280654282)
```

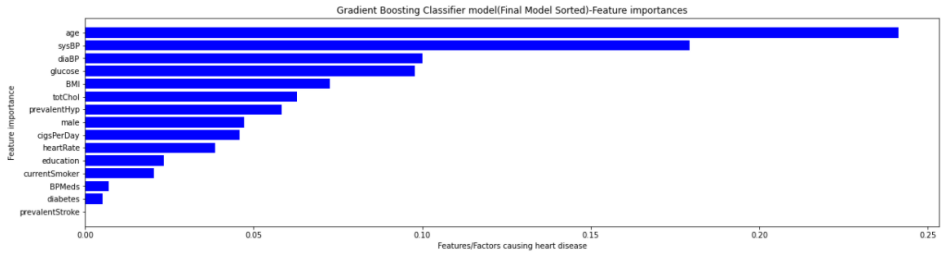
```
[ ] gridSearch3.best_estimator_.fit(x,y)
```

```
GradientBoostingClassifier(max_depth=5, max_features='sqrt',
                           min_samples_leaf=60, min_samples_split=220,
                           n_estimators=50, random_state=42, subsample=0.8)
```

```
[ ] gradBoost_importance=list(gridSearch3.best_estimator_.feature_importances_)
    allfeatures=data.drop('TenYearCHD',axis=1)
    features=list(allfeatures.columns)
    fig = plt.figure(figsize = (15, 5))
    plt.barh(features, gradBoost_importance, color ='blue')
    plt.title('Gradient Boosting Classifier model(Grid Search3)-Feature importances')
    plt.xlabel("Features/Factors causing heart disease")
    plt.ylabel("Feature importance")
    plt.show()
```




```
zipped_lists=zip(gradBoost_importance,features)
zipped_lists=list(sorted(zipped_lists))
importanceSorted,featuresSorted = zip(*zipped_lists)
fig = plt.figure(figsize = (20, 5))
plt.barh(list(featuresSorted), list(importanceSorted), color ='blue')
plt.title('Gradient Boosting Classifier model(Final Model Sorted)-Feature importances')
plt.xlabel("Features/Factors causing heart disease")
plt.ylabel("Feature importance")
plt.show()
```



```
[ ] param_grid4 = {'max_features':range(2,9)}
    gridSearch4=GridSearchCV(estimator = GradientBoostingClassifier(learning_rate=0.1, n_estimators=50, max_depth=5, min_samples_split=220, min_samples_leaf=60, subsample=0.8, random_state=42),
    param_grid = param_grid4, scoring='f1_micro', cv=5)
    gridSearch4.fit(x_train,y_train)
```

```
GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(max_depth=5,
                                                  min_samples_leaf=60,
                                                  min_samples_split=220,
                                                  n_estimators=50,
                                                  random_state=42,
                                                  subsample=0.8),
             param_grid={'max_features': range(2, 9)}, scoring='f1_micro')
```

```
[ ] results=gridSearch4.cv_results_
    print("          Params       MeanTestScore    StdTestScore   Rank")
    for i in range(len(results['params'])):
      print(results['params'][i],'    ',round(results['mean_test_score'][i],4),'    ',round(results['std_test_score'][i],4),'     ',results['rank_test_score'][i])
```

```
          Params       MeanTestScore    StdTestScore   Rank
{'max_features': 2}      0.8513       0.0052       4
{'max_features': 3}      0.8542       0.0039       1
{'max_features': 4}      0.8516       0.0066       3
{'max_features': 5}      0.8489       0.0065       6
{'max_features': 6}      0.8519       0.0038       2
{'max_features': 7}      0.8486       0.0066       7
{'max_features': 8}      0.8507       0.0055       5
```

```
[ ] gridSearch4.best_params_, gridSearch4.best_score_
```

```
({'max_features': 3}, 0.8542363280654282)
```

```
[ ] models.append("Gradient Boosting Classifier - After hyper parameter tuning")
    f1Scores.append(0.8542363280654282)
```

```
[ ] gridSearch4.best_estimator_
```

```
GradientBoostingClassifier(max_depth=5, max_features=3, min_samples_leaf=60,
                           min_samples_split=220, n_estimators=50,
                           random_state=42, subsample=0.8)
```

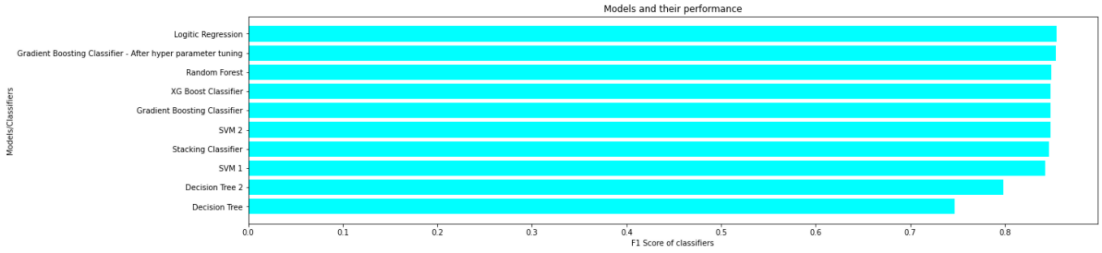
```
[ ] finalModel=GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=5,
                           max_features=3, max_leaf_nodes=None,
```

Comparison of F1 scores of all the models

```
models, f1Scores
```

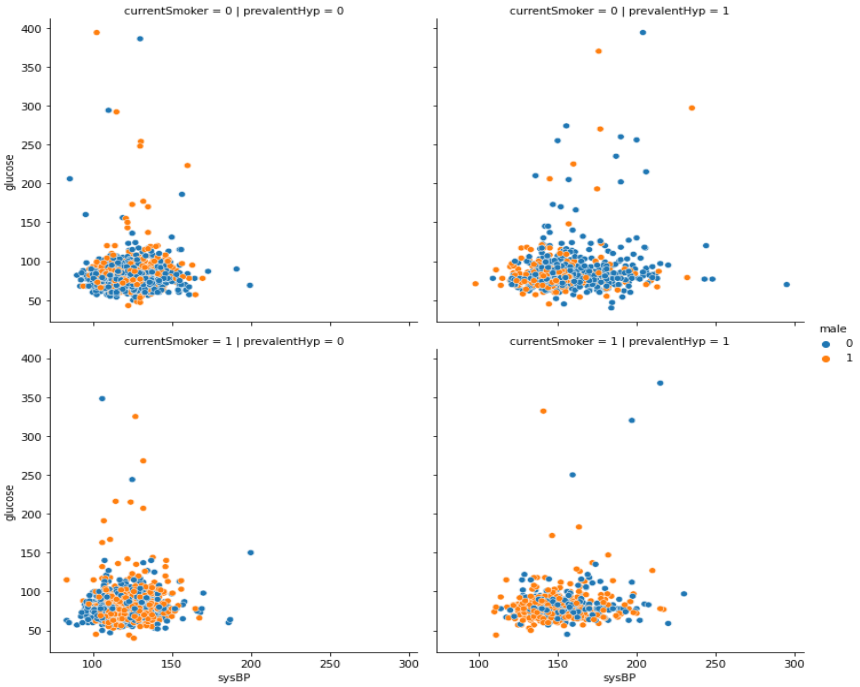
```
(['Logitic Regression',
  'Decision Tree',
  'Decision Tree 2',
  'SVM 1',
  'SVM 2',
  'Random Forest',
  'Gradient Boosting Classifier',
  'XG Boost Classifier',
  'Stacking Classifier',
  'Gradient Boosting Classifier - After hyper parameter tuning'],
 [0.8550855301306927,
  0.7467521967973594,
  0.7984471876533299,
  0.8430471921138321,
  0.84800671305589,
  0.8491876310272536,
  0.8482420045497123,
  0.8482420045497123,
  0.8466981132075472,
  0.8542363280654282])
```

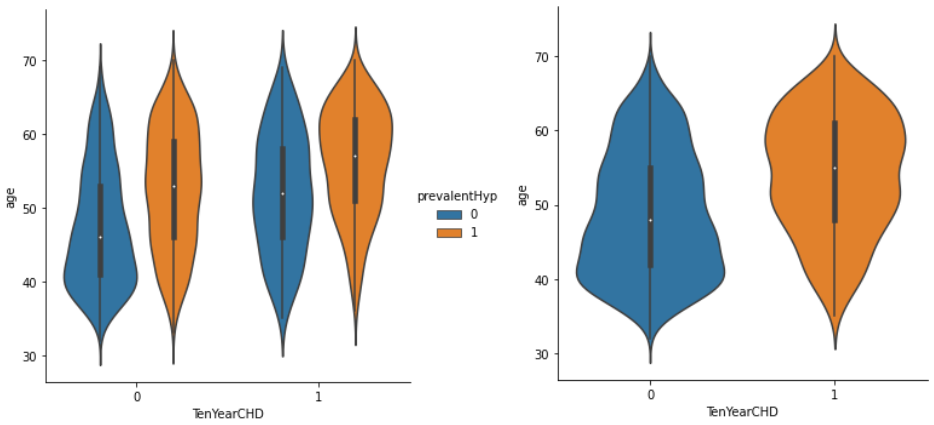
```
[ ] zipped_lists=zip(f1Scores, models)
    zipped_lists=list(sorted(zipped_lists))
    f1ScoresSorted,modelsSorted = zip(*zipped_lists)
    fig = plt.figure(figsize = (20, 5))
    plt.barh(list(modelsSorted), list(f1ScoresSorted), color ='cyan')
    plt.title('Models and their performance')
    plt.xlabel("F1 Score of classifiers")
    plt.ylabel("Models/Classifiers")
    plt.show()
```

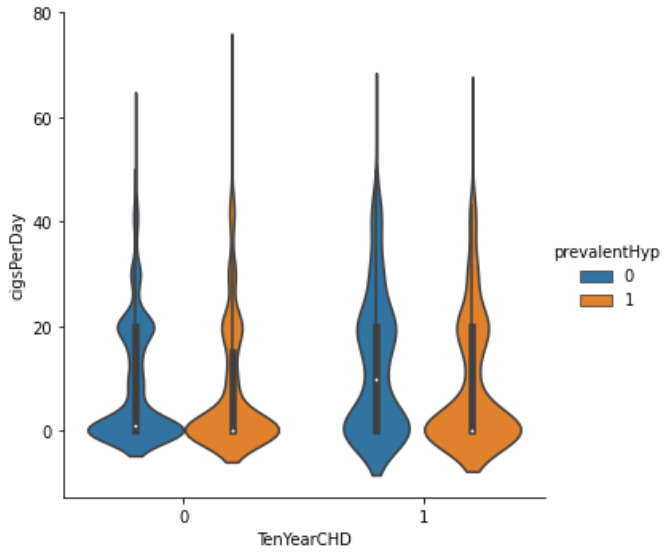


EDA

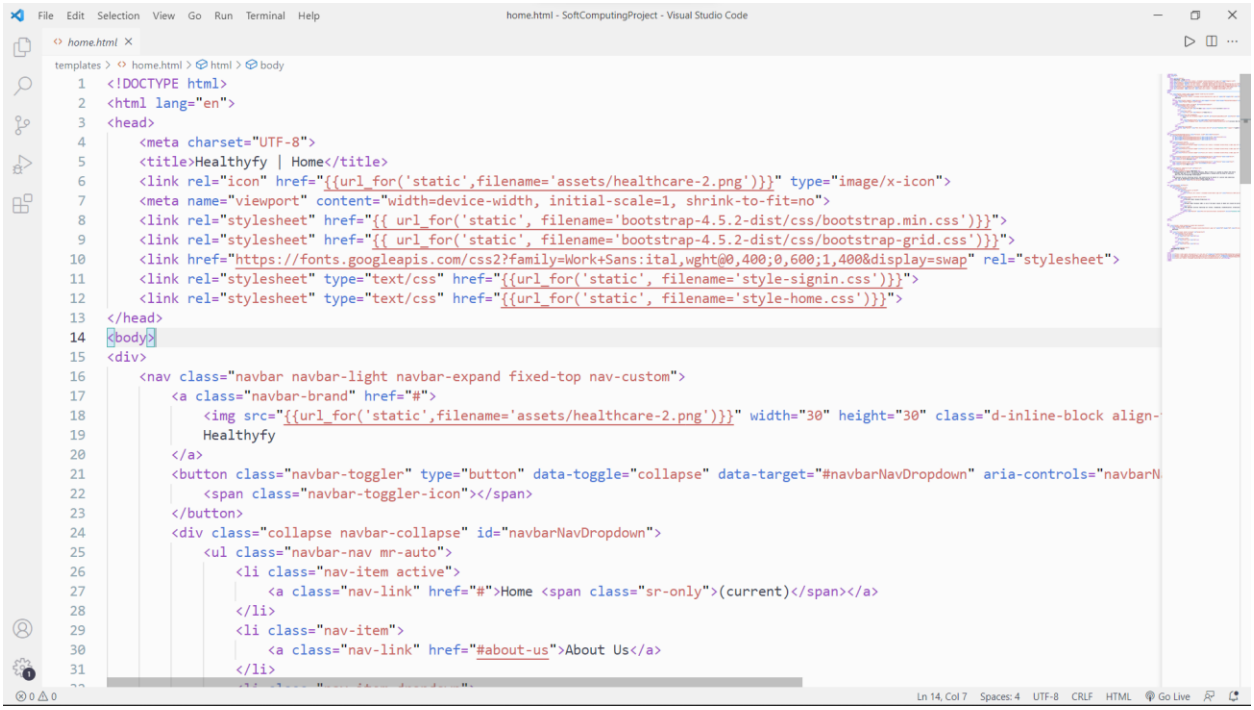
Data Visualization

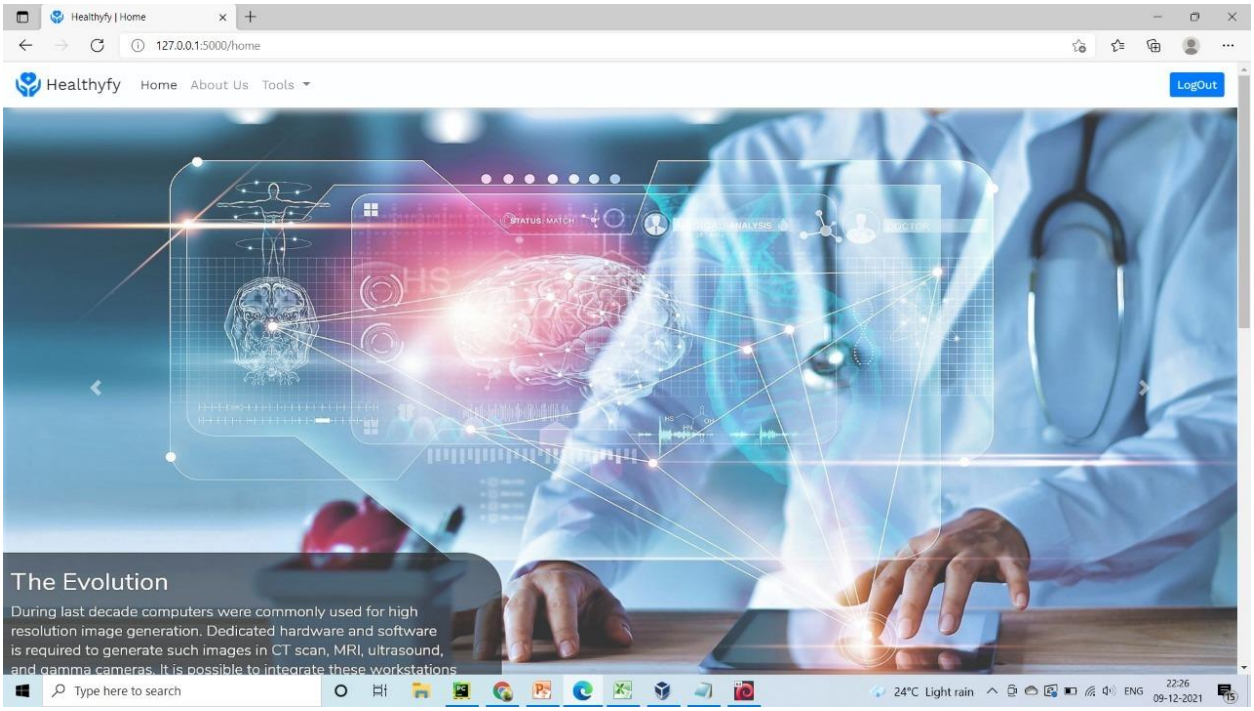

currentSmoker = 0 | prevalentHyp = 0
currentSmoker = 0 | prevalentHyp = 1
currentSmoker = 1 | prevalentHyp = 0
currentSmoker = 1 | prevalentHyp = 1
glucose
sysBP
male
0
1


age
TenYearCHD
prevalentHyp
0
1

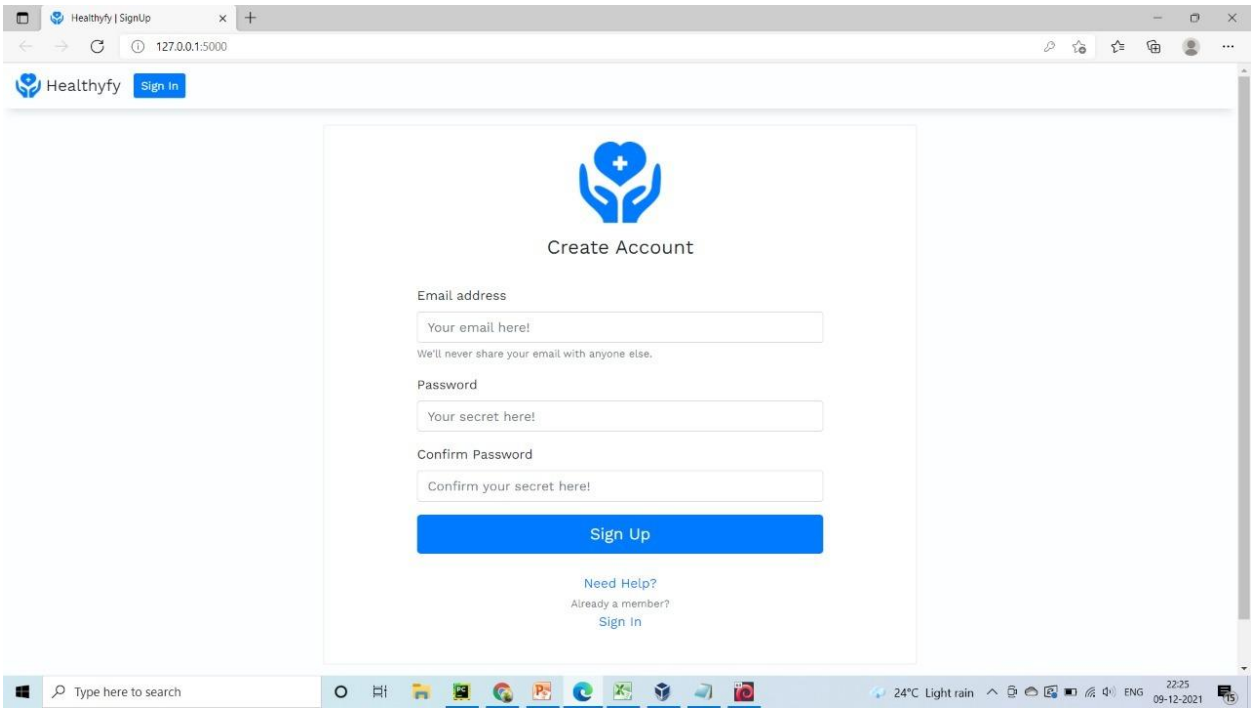

cigsPerDay
TenYearCHD
prevalentHyp
0
1

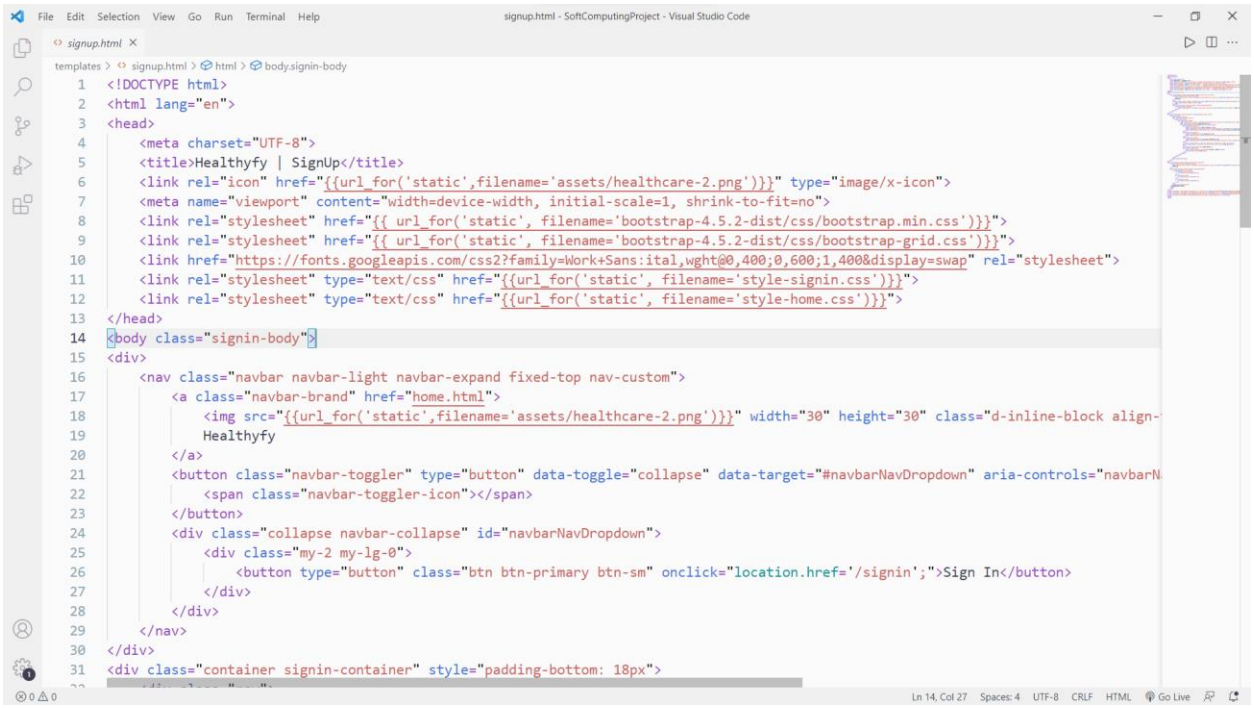
Home Page:


File Edit Selection View Go Run Terminal Help
home.html - SoftComputingProject - Visual Studio Code
home.html
templates > home.html > html > body
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Healthyfy | Home</title>
6 <link rel="icon" href="{{url_for('static',filename='assets/healthcare-2.png')}}" type="image/x-icon">
7 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
8 <link rel="stylesheet" href="{{ url_for('static', filename='bootstrap-4.5.2-dist/css/bootstrap.min.css')}}">
9 <link rel="stylesheet" href="{{ url_for('static', filename='bootstrap-4.5.2-dist/css/bootstrap-grid.css')}}">
10 <link href="https://fonts.googleapis.com/css2?family=Work+Sans:ital,wght@0,400;0,600;1,400&display=swap" rel="stylesheet">
11 <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='style-signin.css')}}">
12 <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='style-home.css')}}">
13 </head>
14 <body>
15
16 <nav class="navbar navbar-light navbar-expand fixed-top nav-custom">
17
18 <img src="{{url_for('static',filename='assets/healthcare-2.png')}}" width="30" height="30" class="d-inline-block align-
19 Healthyfy
20
21 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarN
22
23 </button>
24
25 <ul class="navbar-nav mr-auto">
26 <li class="nav-item active">
27 Home (current)
28
29 <li class="nav-item">
30 About Us
31
Ln 14, Col 7 Spaces: 4 UTF-8 CRLF HTML Go Live


Healthyfy | Home
127.0.0.1:5000/home
Healthyfy
Home
About Us
Tools
LogOut
The Evolution
During last decade computers were commonly used for high resolution image generation. Dedicated hardware and software is required to generate such images in CT scan, MRI, ultrasound, and gamma cameras. It is possible to integrate these workstations
Type here to search
24°C Light rain
ENG
22:26
09-12-2021

Sign Up:


Healthyfy | SignUp
127.0.0.1:5000
Healthyfy
Sign In
Create Account
Email address
Your email here!
We'll never share your email with anyone else.
Password
Your secret here!
Confirm Password
Confirm your secret here!
Sign Up
Need Help?
Already a member?
Sign In
Type here to search
24°C Light rain
ENG
22:25
09-12-2021


File Edit Selection View Go Run Terminal Help
signup.html - SoftComputingProject - Visual Studio Code
signup.html
templates > signup.html > html > body.signin-body
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Healthyfy | SignUp</title>
<link rel="icon" href="{{url_for('static',filename='assets/healthcare-2.png')}}" type="image/x-icon">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<link rel="stylesheet" href="{{ url_for('static', filename='bootstrap-4.5.2-dist/css/bootstrap.min.css')}}">
<link rel="stylesheet" href="{{ url_for('static', filename='bootstrap-4.5.2-dist/css/bootstrap-grid.css')}}">
<link href="https://fonts.googleapis.com/css2?family=Work+Sans:ital,wght@0,400;0,600;1,400&display=swap" rel="stylesheet">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='style-signin.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='style-home.css')}}">
</head>
<body class="signin-body">

<nav class="navbar navbar-light navbar-expand fixed-top nav-custom">

<img src="{{url_for('static',filename='assets/healthcare-2.png')}}" width="30" height="30" class="d-inline-block align-
Healthyfy

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarN

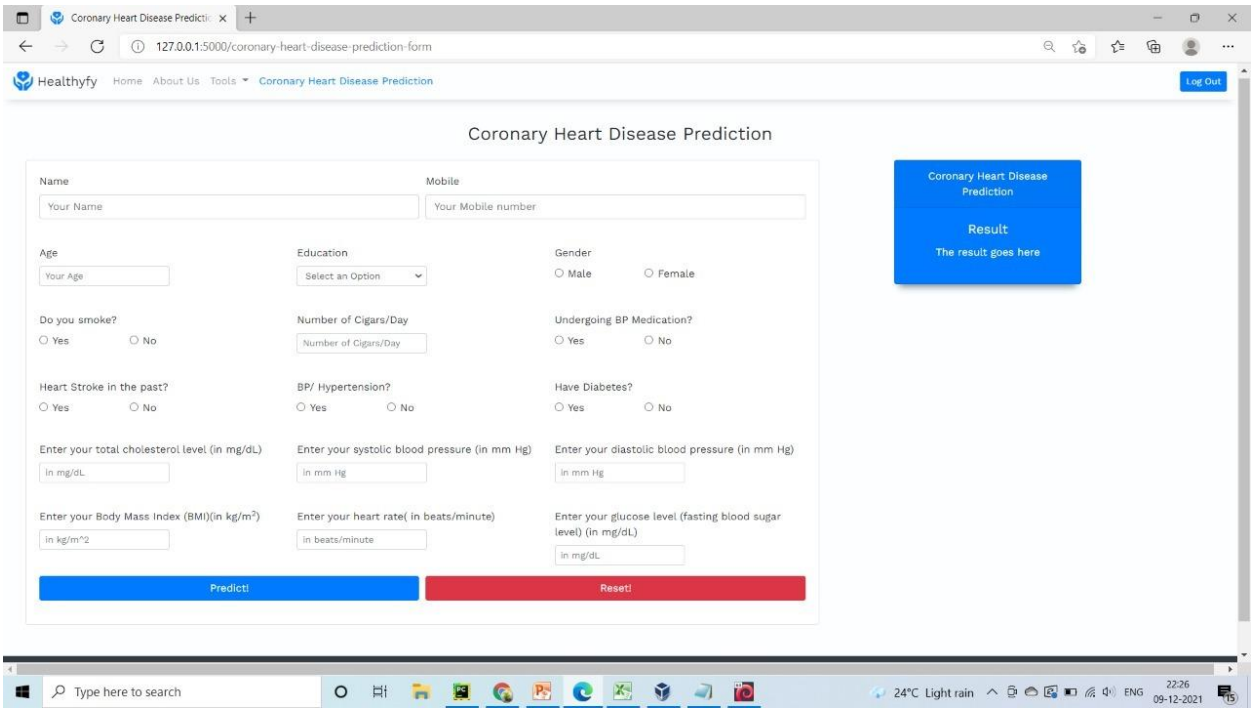
</button>

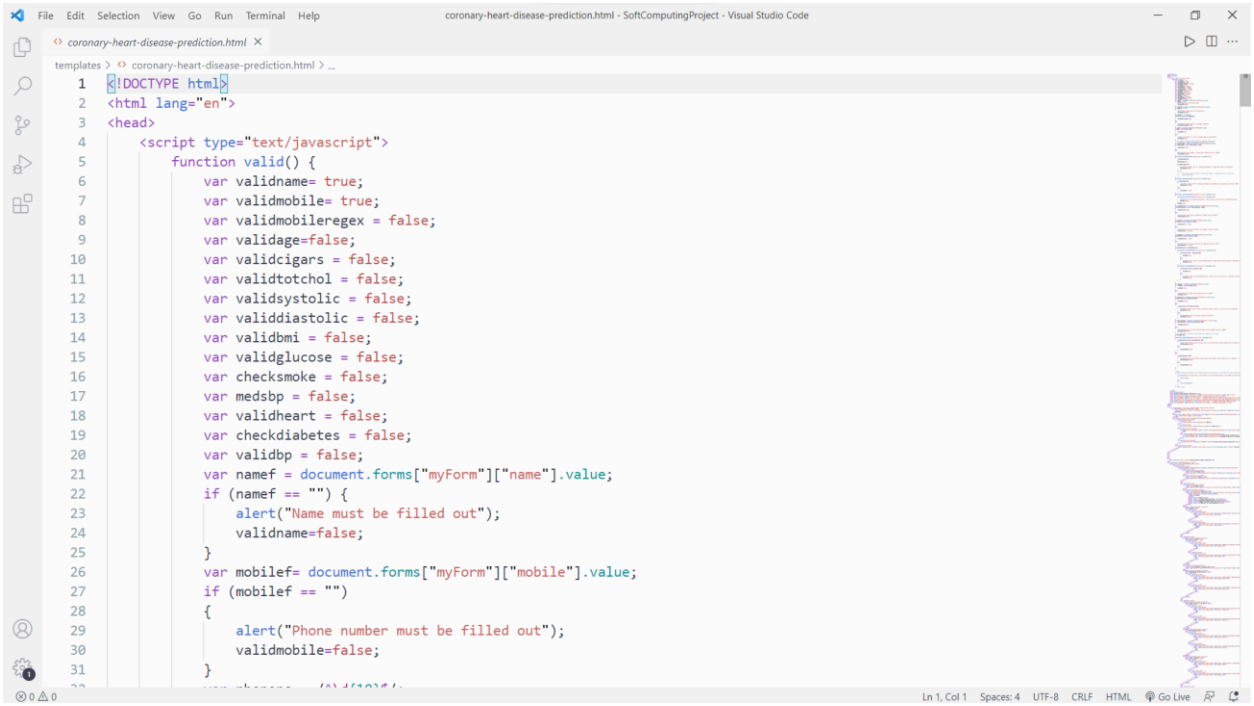
<button type="button" class="btn btn-primary btn-sm" onclick="location.href='/signin';">Sign In</button>

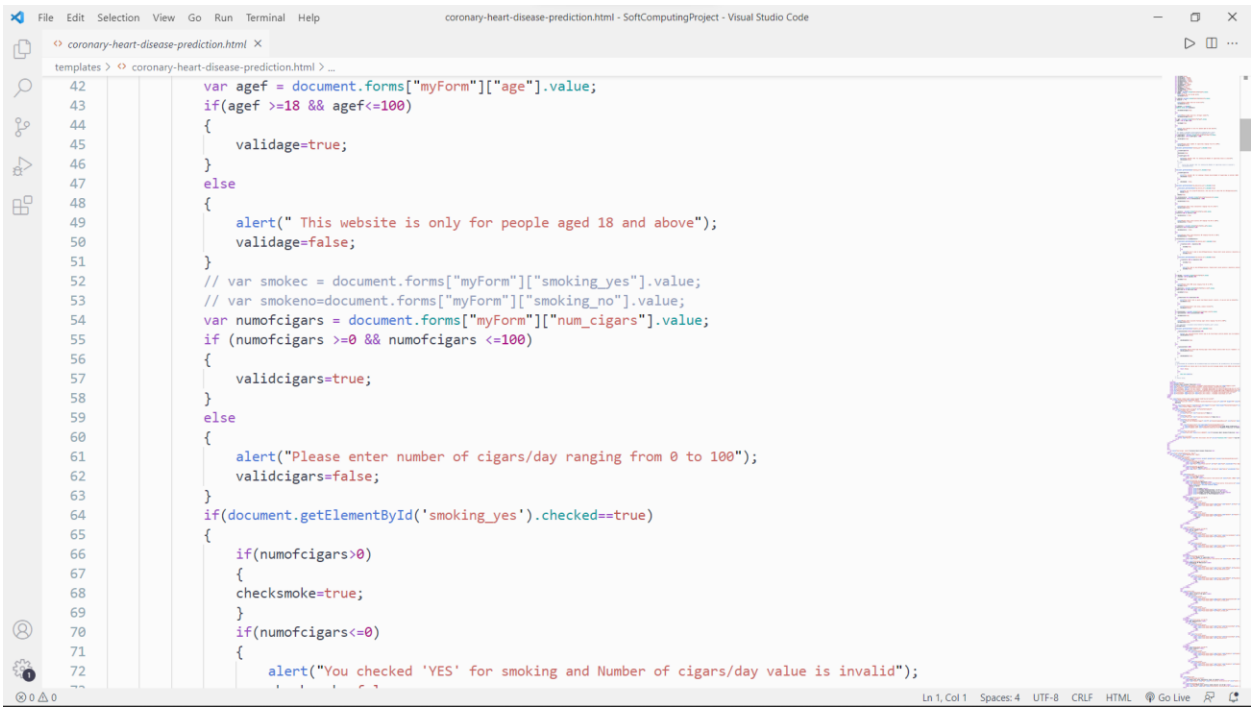
</nav>

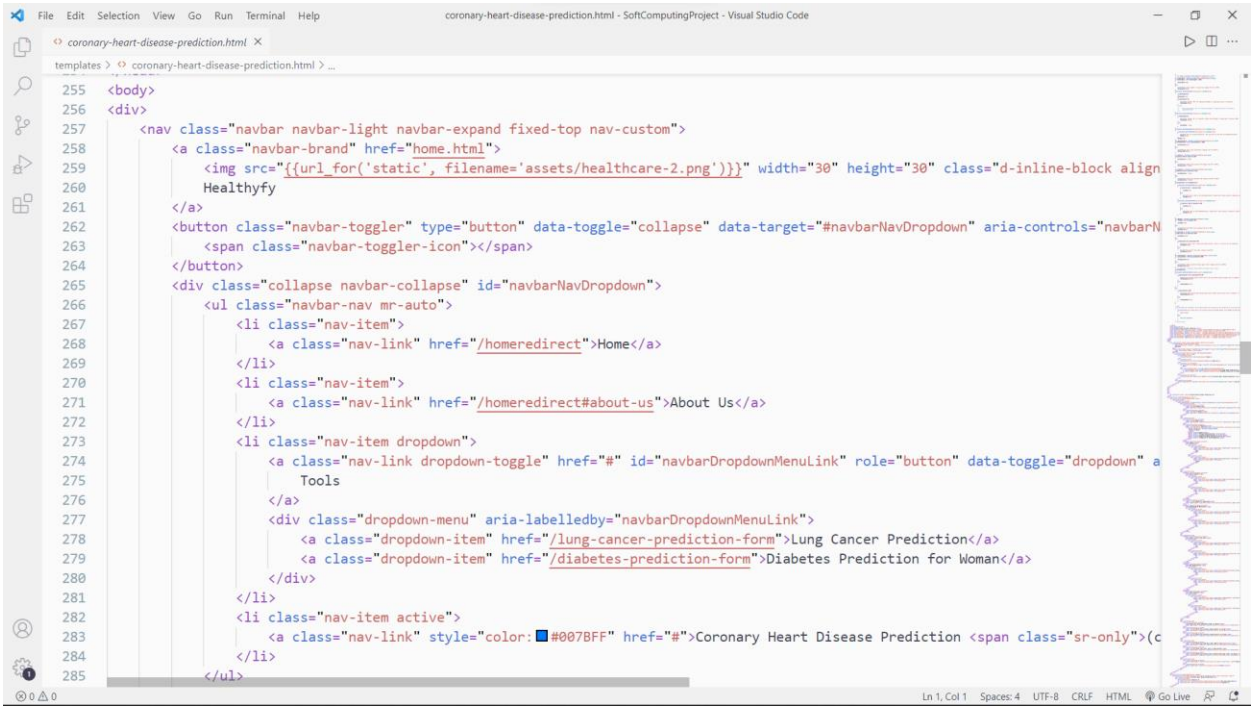
Ln 14, Col 27 Spaces: 4 UTF-8 CRLF HTML Go Live

Prediction Page:






File Edit Selection View Go Run Terminal Help
coronary-heart-disease-prediction.html - SoftComputingProject - Visual Studio Code
coronary-heart-disease-prediction.html
templates > coronary-heart-disease-prediction.html > ...
var agef = document.forms["myForm"]["age"].value;
if(agef >=18 && agef<=100)
{
validage=true;
}
else
{
alert(" This website is only for people aged 18 and above");
validage=false;
}
// var smokec = document.forms["myForm"]["smoking_yes"].value;
// var smokeno=document.forms["myForm"]["smoking_no"].value;
var numofcigars = document.forms["myForm"]["num_cigars"].value;
if (numofcigars >=0 && numofcigars <=100)
{
validcigars=true;
}
else
{
alert("Please enter number of cigars/day ranging from 0 to 100");
validcigars=false;
}
if(document.getElementById('smoking_yes').checked==true)
{
if(numofcigars>0)
{
checksmoke=true;
}
if(numofcigars<=0)
{
alert("You checked 'YES' for smoking and Number of cigars/day value is invalid");
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live


File Edit Selection View Go Run Terminal Help
coronary-heart-disease-prediction.html - SoftComputingProject - Visual Studio Code
coronary-heart-disease-prediction.html
templates > coronary-heart-disease-prediction.html > ...
<body>

<nav class="navbar navbar-light navbar-expand fixed-top nav-custom">

<img src="{{url_for('static', filename='assets/healthcare-2.png')}}" width="30" height="30" class="d-inline-block align
Healthyfy

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarN

</button>

<ul class="navbar-nav mr-auto">
<li class="nav-item">
Home

<li class="nav-item">
About Us

<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-toggle="dropdown" a
Tools

Lung Cancer Prediction
Diabetes Prediction for Woman

<li class="nav-item active">
<a class="nav-link" style="color: #007BFF" href="#">Coronary Heart Disease Prediction (c

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live

Results and Analysis:

People with hyper tension and a record of previous strokes have high risk of getting a heart stroke in the future 10 years. The distribution of people with heart diseases is wider between the ranges of 5 to 10 cigarettes per day. Cigarette smoking is highly dangerous and can cause heart diseases if it is done regularly. It is observed that aged people who are suffering from hypertension are more in numbers who have the risk of getting the disease. Younger people with no hypertension are not prone to get this disease. Therefore, age is positively correlated with coronary heart disease variable. It is also observed that, men are more prone to getting this disease compared to women. People with smoking and drinking habits are mostly men. Alcohol consumption and number of cigarettes per day are also contributing factors for the heart disease. Having high glucose levels is also dangerous as people with diabetes.

Conclusion:

From our experiment, it is clearly visible that Out of all the models used, **Logistic Regression** came out to be the best, followed by Gradient Boosting Classifier, Random Forest, XG Boost Classifier, SVM, Stacking Classifier and Decision Tree. But we implemented **Gradient Boosting Classifier** in our website, because of Logistic Regression was already existing in many projects. And we have successfully created a full stack website with the application of this model. This project, heart disease prediction has a great scope since Logistic Regression can predict if a person will have heart disease or not with an accuracy of 85.5%.

References

1. Mythili T, Dev Mukherjee, Nikitha padiala, Abhiram Naidu. A Heart Disease Prediction Model using SVM-Trees-Logistic Regression. 2013
2. Sharma Purushotham, Dr. Kanak Saxena, Richa Sharma. Efficient Heart Disease Prediction.2016.
3. Boshra Brahmi. Prediction and Diagnosis of Heart Disease by Data Mining Techniques.2015.
4. Noura Ajam. Heart Disease Diagnoses using Artificial Neural Network. 2015
5. Wiharato Wharto, Hari Kusnanto, Herianto. Intelligence System for Diagnosis Level of Coronary Heart Disease with K-Star Algorithm.2016
6. Mr. P Sai Chandrasekhar Reddy, Mr. Puneet Palagi, Ms. Jaya. Heart Disease Prediction Using ANN Algorithm in Data Mining. 2017.
7. SP Rajamohana, C Akalya Devi, Uma Maheswari. Analysis of Neural Networks Based Heart Disease Prediction System.2018.
8. Bo Jin, Chao Che et al. Predicting the Risk of Heart Failure with EHR Sequential Data Modeling.2018.
9. Sabrina Mezzatesta, Claudia Torino, Pasquale De Meo, Giacomo Fiumara, Antonio Vilasi. A Machine Learning-based approach for predicting the outbreak of cardiovascular diseases in patients on dialysis. 2019.
10. Avinash Golande, Pavan Kumar T. Heart Disease Prediction Using Effective Machine Learning Techniques. 2019.
11. Khaled Mohamad Amulstafa . Prediction of heart disease and classifier's sensitivity analysis.2020.
12. Devansh Shah, Samir Patel, Santosh Kumar Bharti. Heart Disease Prediction using Machine Learning Techniques.2020.
13. Nilam Harkulkar, Swati Nadkarni, Bhavesh Patel. Heart Disease Prediction using CNN, Deep Learning Model.2020.
14. Harshit Jindal, Sarthak Agarwal, Rishabh Khera, Rachna Jain, Preeti Nagrath. Heart Disease Prediction using Machine Learning Algorithms. 2021
15. Xiao-Yan Gao, Abdelmegeid Amin, Hassan Shaban Hassan, Eman M.Anwar. Improving the Accuracy for Analyzing Heart Disease Prediction based on Ensemble Method. 2021

16. Jaymin Patel, Samir B Patel, TejalUpadhyay. Heart Disease Prediction using Machine Learing and Datamining Technique. 2016.
17. R Chitra, V Sreenivasagam. Review of heart disease prediction system using data mining and hybrid intelligent techniques. 2013.
18. Jayshri S. Sonawane, D. Patil. Prediction of Heart Disease using Vector Quantization Algorithm. 2014
19. Sivagowry S, Dr. Durairaj.M. An Empirical study on applying data mining techniques for the analysis and prediction of heart disease. 2014.
20. Ilayaraja M, Meyyappan T. Efficient Data Mining Method to Predict the Risk of Heart Diseases Through Frequent itemsets. 2015.
21. Prachi Paliwal, Mahesh Malviya. Prediction for Heart Attack Problem Using Various Classification. 2015.
22. Aswhini Shetty, Chandra Naik. Different Data Mining Approaches for Predicting Heart Disease. 2016.
23. Prajakta Ghadge, Vrushali, Kajal, Prajakta Deshmukh. Intelligent Heart Attack Prediction System Using Big Data. 2016.
24. Azam Dekamin, Ahmad Shaibatalhamdi. A Datamining approach for coronary artery Disease prediction. 2017.
25. Shashikanth R, Chetankumar P. Predictive Model of cardiac arrest in smokers using machine learning based on heart rate variability parameter. 2020
26. Galla Siva Sai Bindhika, Munaga Meghana, Mnachuri Sathvika Reddy, Rajalakshmi. Heart Disease Prediction Using Machine Learning Techniques. 2020
27. Apurb Rajdhan, Milan Sai, Avi Agarwal, Dundigulla Ravi. Heart Disease Prediction Uisng Machine Learning. 2020
28. Anusha MB, Chaitra K, Chandana HM, Kiran G, Swathi D. Heart Disease Prediction Using Machine Learning Techniques. 2020
29. Li Yang, Hiabin Wu, Xiaoqing Jin, Pinpin Zheng, Shiyun Hu, Xiaoling Xu, Wei Yu, Jing Yan. Study of Cardiovascular Disease prediction Model based on random forest. 2020.
30. Amin Yazdani, Kasturi Dewi Varathan, Yin Kia Chiam, Asad Waqar Malik, Wan Azman Wan Ahamd. Identification of significant features and data mining techniques in predicting heart disease. 2018