



Diabetes Prediction Using Machine Learning

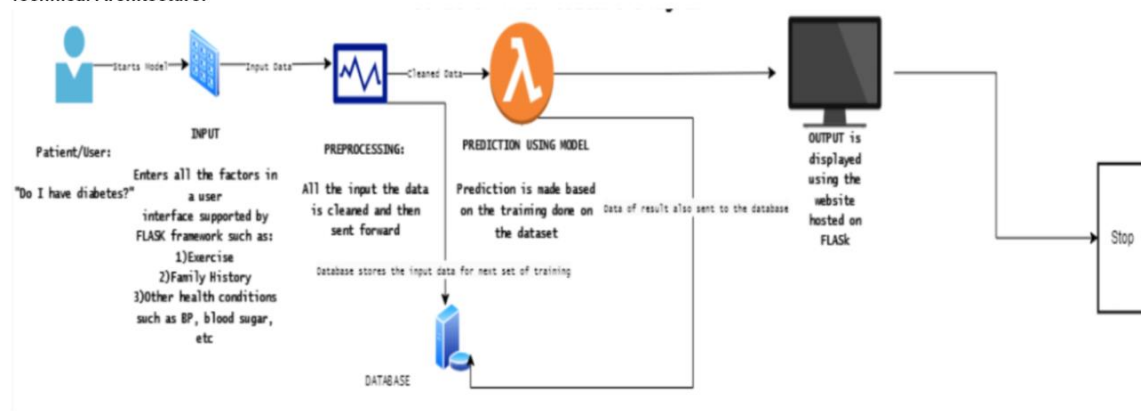
Diabetes Prediction Using Machine Learning

This research initiative focuses on harnessing machine learning algorithms to forecast the likelihood of diabetes onset in individuals based on their medical records and pertinent factors, including age, body mass index (BMI), familial medical history, and lifestyle choices. The dataset under scrutiny encompasses comprehensive clinical parameters, spanning blood pressure, BMI, heart conditions, and cholesterol levels.

The primary aim is to construct a predictive model capable of accurately identifying individuals with a high susceptibility to diabetes, thereby facilitating early intervention and preemptive measures against the disease. Leveraging machine learning methodologies to scrutinize substantial data sets allows for the recognition of discernible patterns and the formulation of precise predictions, potentially translating into life-saving implications.

In essence, this undertaking holds promise for the healthcare domain, given its potential to enhance the early identification and prevention of diabetes, thus fostering improved health outcomes for both individuals and communities.

Technical Architecture:



Project Flow:

- User engagement with the user interface (UI) to input data.
- Integrated model analysis of the input data provided by the user.
- Display of the prediction outcome on the UI following the model analysis.

To achieve these objectives, the following activities need to be executed:

- Problem Definition / Problem Understanding
 - Clarification of the business problem.
 - Specification of the business requirements.
 - Conducting a comprehensive review of the pertinent literature.
 - Examination of the social or business impact of the project.
- Data Collection & Preparation
 - Acquisition of the requisite dataset.
 - Preprocessing and refinement of the collected data.
- Exploratory Data Analysis
 - Utilization of descriptive statistical techniques for data examination.
 - Application of visual analysis methods for data comprehension.
- Model Development
 - Implementation of model training utilizing diverse algorithms.
 - Validation of the model through rigorous testing procedures.
- Performance Evaluation
 - Thorough assessment of the model's performance using multiple evaluation metrics.
- Model Deployment
 - Preservation of the most optimal model.
 - Integration of the model with the designated web framework.

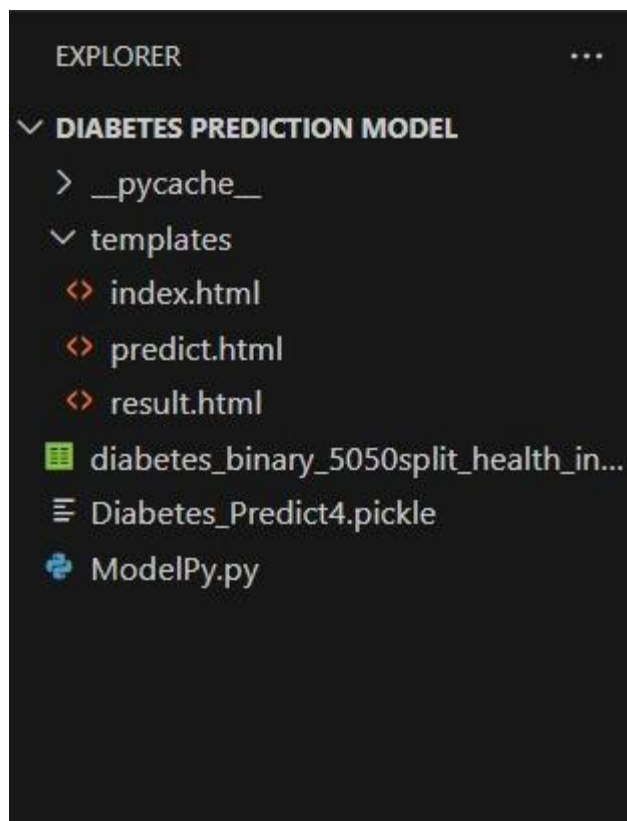
Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- ML Concepts
 - Supervised learning: <https://www.javatpoint.com/supervised-machine-learning>
 - Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning> •
 - Decision tree: <https://www.javatpoint.com/machine-learning-decision-tree-classificationalgorithm>
 - Random forest: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
 - KNN: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning> •
 - Xgboost: <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>
 - Evaluation metrics: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>
 -
- NLP:-https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_python.htm
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Random.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Business Problem Specification

The central business problem tackled in this project involves the early detection and prognosis of diabetes using machine learning algorithms. The primary objective is to engineer a predictive model proficient in accurately identifying individuals with an elevated risk of developing diabetes, leveraging their health records and pertinent factors. The significance of timely diabetes detection lies in the potential enhancement of healthcare outcomes, cost reduction, and consequential benefits for healthcare providers and insurance entities. Hence, the development of a precise and dependable predictive model holds considerable potential for shaping healthcare outcomes and associated costs.

Activity 2: Business Requirements

Business requirements denote the explicit demands and expectations of business stakeholders pertaining to the intended project outcome. In the context of the diabetes prediction initiative, the following represent the key business requisites:

- **Accurate Prediction:** The predictive model must demonstrate accuracy in discerning individuals at a heightened risk of diabetes, based on their health records and other pertinent factors.
- **Efficiency:** The model's efficiency and promptness in processing substantial data volumes to deliver timely predictions are imperative.
- **Scalability:** Ensuring the model's scalability to handle voluminous datasets and accommodate potential expansions in data volume is crucial.
- **Flexibility:** The model's adaptability to accommodate shifts in data sources or input parameters is paramount.
- **User-Friendliness:** The model should be user-friendly and comprehensible for healthcare providers and insurance entities.
- **Integration:** Seamless integration of the model with prevailing healthcare systems and processes is necessary.
- **Security:** Ensuring the model's security and safeguarding of patient data privacy is critical.
- **Compliance:** Adherence to pertinent healthcare regulations and standards is vital.

Activity 3: Literature Review

The literature survey plays a pivotal role in the comprehensive understanding of diabetes prediction employing machine learning techniques:

- "Machine Learning for Diabetes Prediction: A Review" by E. Şahin et al.
- "Predicting Type 2 Diabetes Mellitus Using Machine Learning Techniques" by S. Chakraborty et al.
- "Deep Learning for Diabetes Prediction: A Review" by Y. Zhao et al.
- "Diabetes Prediction Using Machine Learning Techniques: A Comparative Study" by S. B. Gawali and R. K. Kamat.
- "Machine Learning for Early Detection of Diabetic Retinopathy" by A. Gulshan et al.

Activity 4: Social or Business Impact

Accurate diabetes prediction through machine learning holds substantial promise for both social and business spheres. It aids in the identification of individuals prone to diabetes, leading to early intervention and preventive measures. From a business standpoint, precise prediction supports healthcare providers and insurers in more effectively managing healthcare costs and resources. Moreover, it fosters tailored healthcare, improving patient outcomes and adherence to treatment protocols. Ultimately, the accurate prediction of diabetes using machine learning has the potential to enhance patient outcomes, curtail healthcare expenses, and promote personalized healthcare.

Milestone 2: Data Collection & Preparation

Machine learning heavily relies on data, representing the fundamental component that facilitates algorithm training. Thus, this segment enables the acquisition of the requisite dataset.

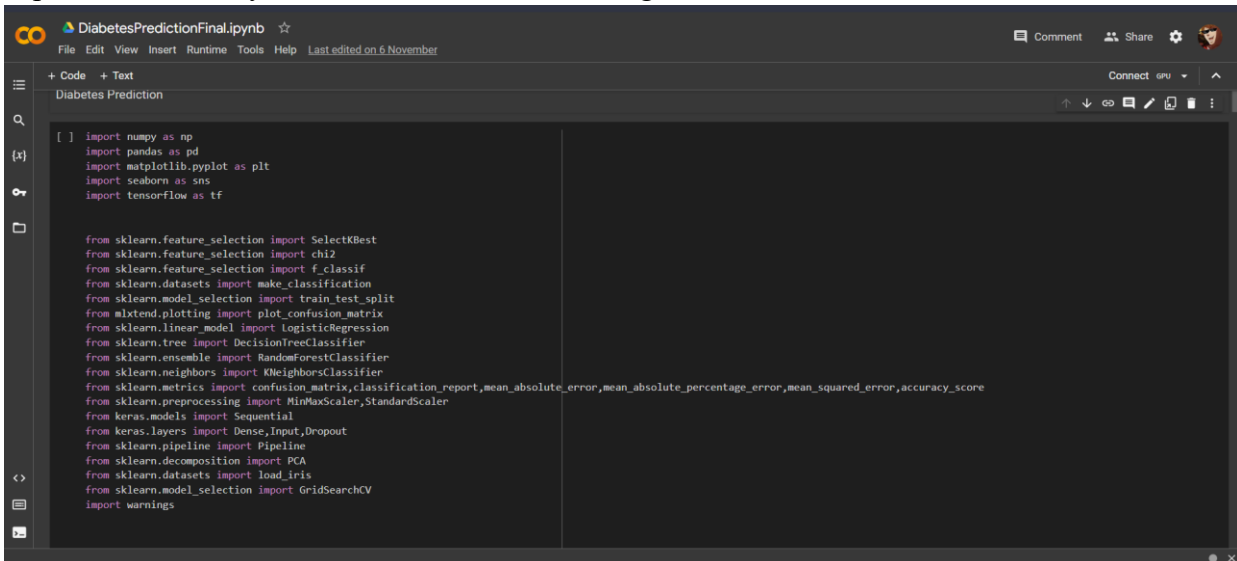
Activity 1: Dataset Acquisition

Several well-known open sources, such as kaggle.com and the UCI repository, serve as viable options for dataset collection. For this project, a .csv dataset was procured from kaggle.com. To access the dataset, please utilize the provided link to the Diabetes Health Indicators Dataset on Kaggle.

Following the dataset procurement, it is essential to conduct thorough data comprehension using visualization and analytical techniques. It is pertinent to note that various techniques can be applied for comprehensive data understanding. However, this project has implemented a selected set of techniques. Additionally, one may consider employing multiple techniques for an enhanced understanding of the data.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

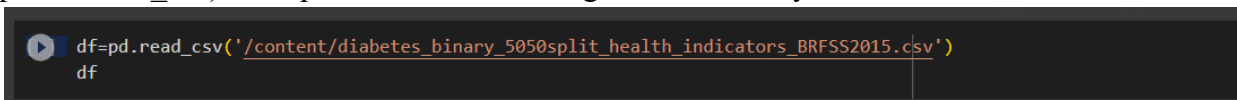


```
[ ] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import f_classif
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report, mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, accuracy_score
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from keras.models import Sequential
from keras.layers import Dense, Input, Dropout
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.model_selection import GridSearchCV
import warnings
```

Activity 1.2: Read the Dataset

Our dataset format is in .csv. We can read the dataset with the help of pandas(using pandas.read_csv). As a parameter we have to give the directory of the csv file.



```
df=pd.read_csv('/content/diabetes_binary_5050split_health_indicators_BRFSS2015.csv')
df
```

Activity 2: Data Preparation

Having comprehended the nature of the data, we now proceed to preprocess the acquired dataset.

The obtained dataset might not be conducive for training the machine learning model due to potential data inconsistencies. Therefore, it is imperative to meticulously cleanse the dataset to ensure favorable outcomes.

This process entails the following steps:

- Addressing missing values
- Managing categorical data

- Dealing with outliers

Activity 2.1: Handling missing values

Let's know the info and describe of our dataset first. To find the shape of our data, the `df.shape` method is used. To find the data type, `df.info()` function is used

```
[ ] df.isnull().any()

Diabetes_binary      False
HighBP               False
HighChol             False
CholCheck            False
BMI                  False
Smoker               False
Stroke               False
HeartDiseaseorAttack False
PhysActivity         False
Fruits               False
Veggies              False
HvyAlcoholConsump    False
AnyHealthcare        False
NoDocbcCost          False
GenHlth              False
MentHlth              False
PhysHlth             False
DiffWalk             False
Sex                  False
Age                  False
Education             False
Income               False
dtype: bool
```

For checking the null values, `df.isnull()` function is used. To sum those null values we use `.sum()` function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
[ ] df.isnull().sum()
```

```
Diabetes_binary      0
HighBP              0
HighChol             0
CholCheck            0
BMI                 0
Smoker              0
Stroke              0
HeartDiseaseorAttack 0
PhysActivity         0
Fruits              0
Veggies             0
HvyAlcoholConsump    0
AnyHealthcare        0
NoDocbcCost          0
GenHlth              0
MentHlth             0
PhysHlth             0
DiffWalk             0
Sex                  0
Age                  0
Education            0
Income               0
dtype: int64
```

Activity 2.2: Handling Categorical Values

Given the presence of categorical data within our dataset, it is imperative to convert these categorical entries into integer encoding or binary encoding.

To achieve this conversion from categorical to numerical features, we employ specific encoding techniques. While multiple methods exist, our project utilizes Label encoding, facilitated by list comprehension.

In our project, Label encoding has been executed across multiple columns housing categorical features.

Activity 2.3: Handling Imbalance Data

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of some columns feature with some mathematical formula

From the below diagram, we could visualize that some of the feature has outliers Boxplot from seaborn library is used here.

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features

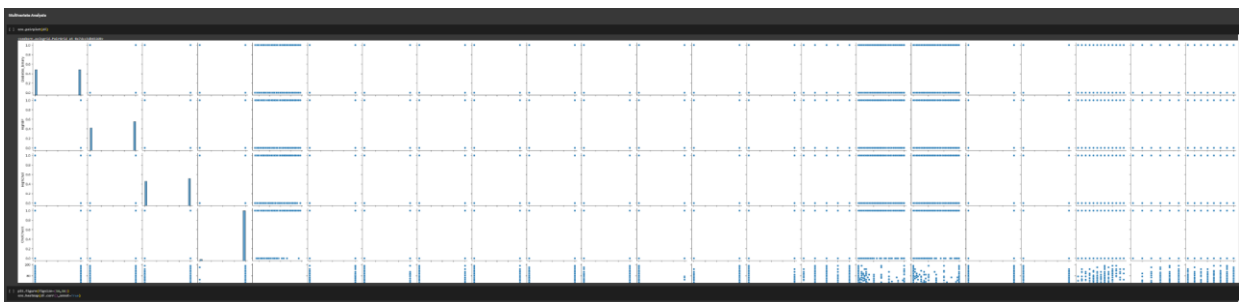
```
df.describe()
```

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost
count	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	70692.000000	...	70692.000000	70692.000000
mean	0.500000	0.563458	0.525703	0.975259	29.856985	0.475273	0.062171	0.147810	0.703036	0.611795	...	0.954960	0.093914
std	0.500004	0.495980	0.499342	0.155336	7.113954	0.499392	0.241468	0.354914	0.456924	0.487345	...	0.207394	0.291712
min	0.000000	0.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000	25.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1.000000	0.000000
50%	0.500000	1.000000	1.000000	1.000000	29.000000	0.000000	0.000000	0.000000	1.000000	1.000000	...	1.000000	0.000000
75%	1.000000	1.000000	1.000000	1.000000	33.000000	1.000000	0.000000	0.000000	1.000000	1.000000	...	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	98.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000

8 rows x 22 columns

Activity 2: Visual analysis

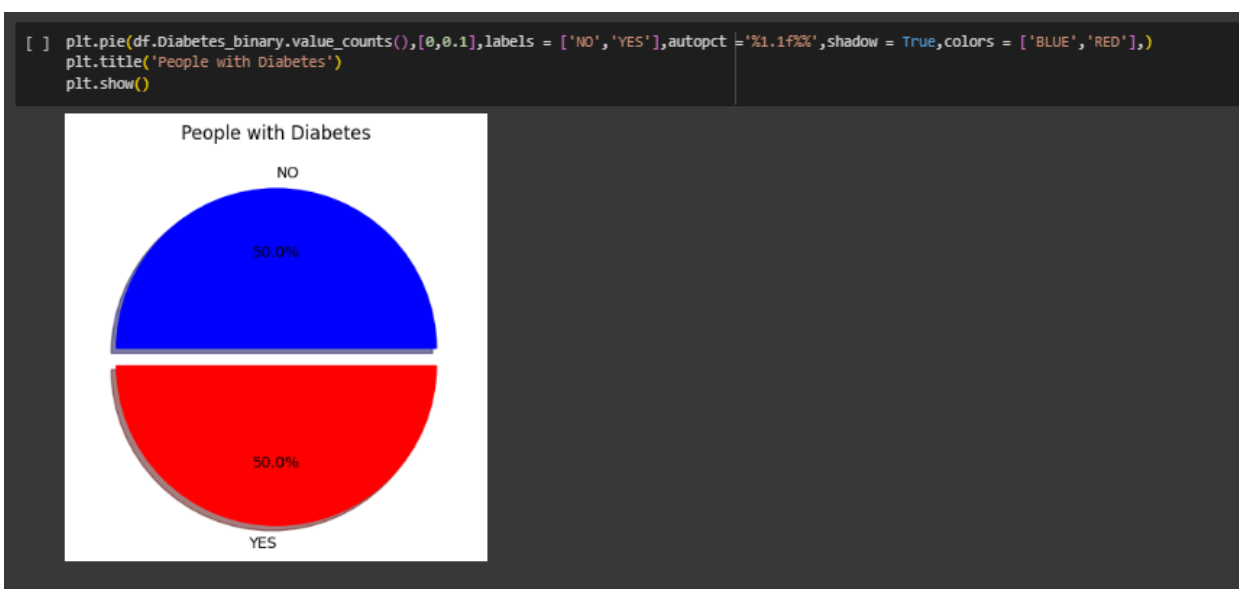
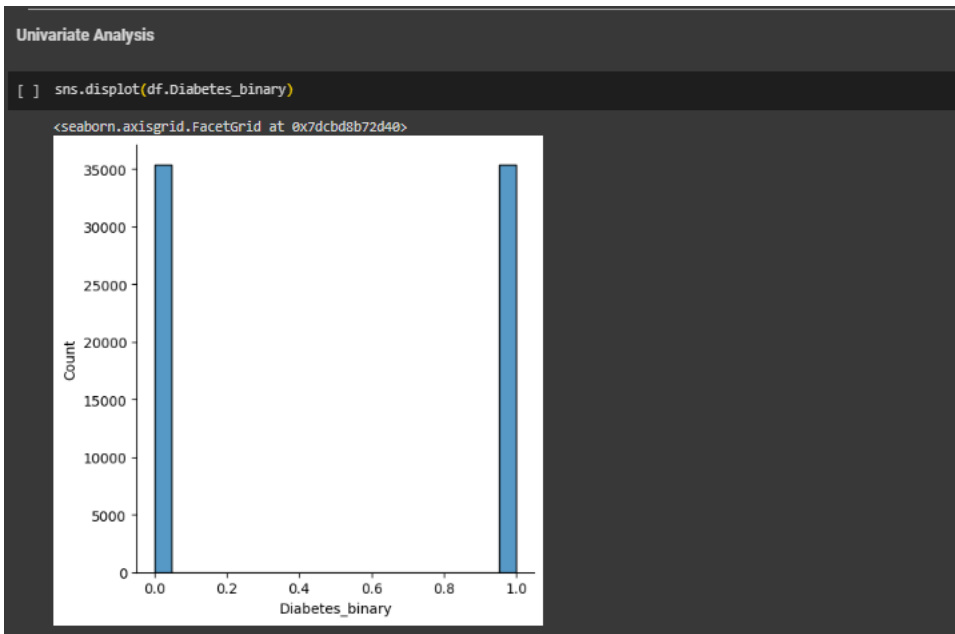
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.



Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and count plot.

In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features.



Activity 2.2: Bivariate analysis

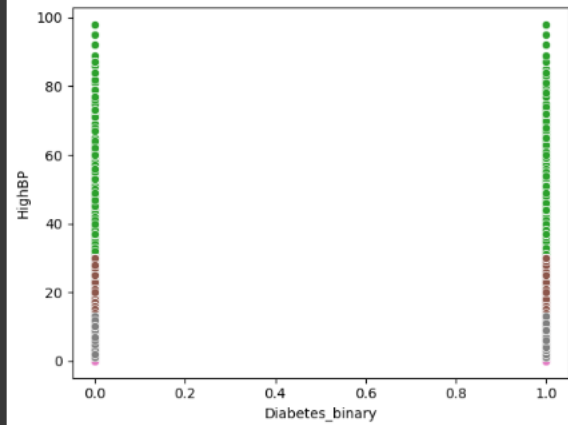
To find the relation between two features we use bivariate analysis. Here we are visualizing. Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

```

sns.scatterplot(x=df.Diabetes_binary,y=df.HighBP)
sns.scatterplot(x=df.Diabetes_binary,y=df.HighChol)
sns.scatterplot(x=df.Diabetes_binary,y=df.BMI)
sns.scatterplot(x=df.Diabetes_binary,y=df.PhysActivity)
sns.scatterplot(x=df.Diabetes_binary,y=df.GenHlth)
sns.scatterplot(x=df.Diabetes_binary,y=df.PhysHlth)
sns.scatterplot(x=df.Diabetes_binary,y=df.DiffWalk)
sns.scatterplot(x=df.Diabetes_binary,y=df.Age)

```

<Axes: xlabel='Diabetes_binary', ylabel='HighBP'>



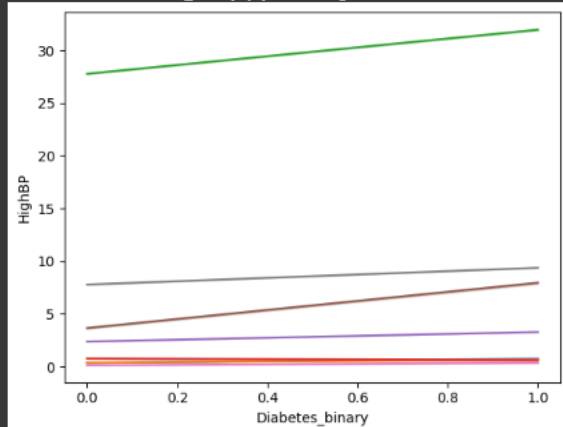
Bivariate Analysis

```

[ ] sns.lineplot(x=df.Diabetes_binary,y=df.HighBP)
sns.lineplot(x=df.Diabetes_binary,y=df.HighChol)
sns.lineplot(x=df.Diabetes_binary,y=df.BMI)
sns.lineplot(x=df.Diabetes_binary,y=df.PhysActivity)
sns.lineplot(x=df.Diabetes_binary,y=df.GenHlth)
sns.lineplot(x=df.Diabetes_binary,y=df.PhysHlth)
sns.lineplot(x=df.Diabetes_binary,y=df.DiffWalk)
sns.lineplot(x=df.Diabetes_binary,y=df.Age)

```

<Axes: xlabel='Diabetes_binary', ylabel='HighBP'>



Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.



Applying PCA in a Machine Learning Pipeline for Diabetes Prediction:

Implementing PCA (Principal Component Analysis) within a machine learning pipeline dedicated to diabetes prediction holds potential benefits for refining the model's performance. The incorporation of PCA, in conjunction with hyperparameter tuning via GridSearchCV, data preprocessing through Standard Scaler, and the employment of a classifier, can contribute to the enhancement of the machine learning model's efficacy. Notably, PCA aids in reducing the dimensionality of the data by pinpointing key features, while StandardScaler facilitates data scaling. GridSearchCV serves to optimize the classifier's hyperparameters, ensuring a finely tuned model. Subsequently, the application of an appropriate classifier to the preprocessed and dimensionality-reduced data enables the evaluation of model performance through pertinent metrics.

Dataset division into training and testing subsets is a crucial step in the process. This involves segregating the Dataset into distinct training and test sets. Firstly, the dataset is partitioned into 'x' and 'y' variables, where 'df' represents the exclusion of the target variable in the 'x' variable, and the 'y' variable encapsulates the target variable. The 'train_test_split()' function from the sklearn library is employed for the segregation of training and testing data, incorporating

parameters such as 'x,' 'y,' 'test_size,' and 'random_state.'

```
[ ] x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3,random_state=0)
    from sklearn.impute import SimpleImputer

    imputer = SimpleImputer(missing_values = 0, strategy='mean')

    x_train = imputer.fit_transform(x_train)
    x_test = imputer.fit_transform(x_test)
    x_test.shape,y_test.shape

((21208, 21), (21208,))
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project, we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Sequential Model using TensorFlow

A sequential model is created using TensorFlow and layers are added onto it. Inside the function is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model with R2_score. Uses “adam” optimiser and “binary_crossentropy” for loss function

```
[ ] model=Sequential()
    model.add(Dense(30,input_dim=21,activation='relu'))
    model.add(Dense(10,activation='relu'))
    model.add(Dense(1,activation='sigmoid'))

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

[ ] model.fit(x_train, y_train, epochs=50, batch_size=10)
```

Activity 1.2: Random Forest Classifier

A function named random forest Classifier is created and train and test data are passed as the parameters. Inside the function, random forest classifier algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model with R2_score.

```
Random Forest Classifier

[ ] rf=RandomForestClassifier(max_depth=100,n_estimators=50,random_state=0)
    rf.fit(x_train,y_train)

+ RandomForestClassifier
+ RandomForestClassifier(max_depth=100, n_estimators=50, random_state=0)

[ ] y_pred=rf.predict(x_test)
    print('Training set score: {:.4f}'.format(rf.score(x_train, y_train)))
    print('Test set score: {:.4f}'.format(rf.score (x_test, y_test)))

Training set score: 0.9952
Test set score: 0.7370
```

Activity 1.4: Decision Tree Classifier

A function named decision Tree classifier is created and train and test data are passed as the parameters. Inside the function, decision Tree classifier algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model, For evaluating the model with R2_score

```
Decision Tree Classifier

[ ] dt=DecisionTreeClassifier(max_depth=12)
    dt.fit(x_train,y_train)

+ DecisionTreeClassifier
+ DecisionTreeClassifier(max_depth=12)

[ ] y_pred=dt.predict(x_test)
    print('Training set score: {:.4f}'.format(dt.score(x_train, y_train)))
    print('Test set score: {:.4f}'.format(dt.score (x_test, y_test)))

Training set score: 0.7997
Test set score: 0.7226
```

Activity 1.3: Hyperparameter Tuning

By fine-tuning hyperparameters, we can adjust our model's efficacy for the best possible outcomes. This practice is crucial in machine learning, and it's essential to choose appropriate values for hyperparameters to achieve success.

Hyperparameter Tuning

```
grid={'penalty' : ['l1', 'l2', 'elasticnet'],
      'C' : np.logspace(-4, 4, 20),
      'solver' : ['lbfgs', 'newton-cg', 'liblinear', 'sag', 'saga'],
      'max_iter' : [100, 1000, 2500, 5000, 7500, 10000, 15000]}

logreg_cv=GridSearchCV(LogisticRegression(),grid,cv=10)

[ ] logreg_cv.fit(x_train,y_train)
```

Activity 2: Testing the model

Here we have tested with Logistic regression and SVM algorithms. With the help of predict () function.

Support Vector Classification

```
[ ] from sklearn.svm import SVC
    clf = SVC (kernel='rbf', C=1.0)
    clf.fit(x_train, y_train)
    y_pred=clf.predict(x_test)

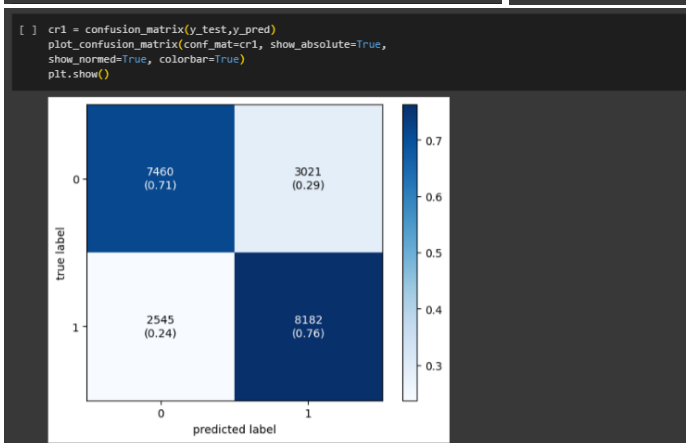
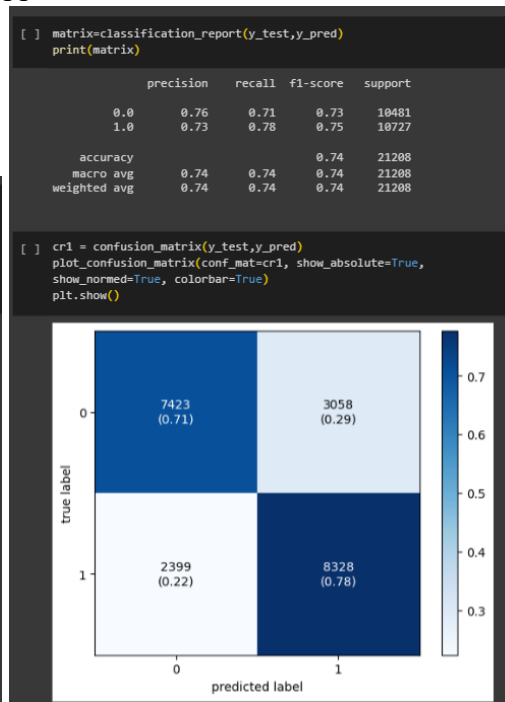
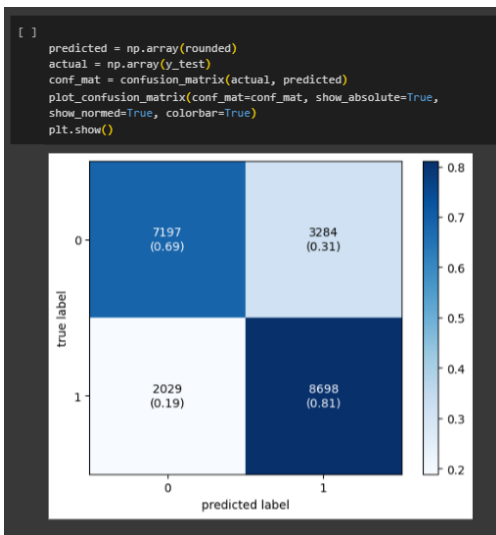
[ ] y_pred=lg.predict(x_test)
    print('Training set score: {:.4f}'.format(clf.score(x_train, y_train)))
    print('Test set score: {:.4f}'.format(clf.score (x_test, y_test)))

Training set score: 0.7661
Test set score: 0.7547
```

Milestone 5: Performance Testing

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.



Activity 1.1: Compare the model

For comparing the below two models, with their R₂ score on training and testing data. the results of models are displayed as output. From the below three models random forest regressor is performing well

```
[ ] mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error: ' + str(mse))
rmse = (mean_squared_error(y_test, y_pred))**(0.5)
print('Root Mean Squared Error: ' + str(rmse))

Mean Squared Error: 0.24976423990946814
Root Mean Squared Error: 0.49976418430042396

[ ] matrix=classification_report(y_test,y_pred)
print(matrix)
```

	precision	recall	f1-score	support
0.0	0.76	0.73	0.74	10481
1.0	0.74	0.77	0.76	10727
accuracy			0.75	21208
macro avg	0.75	0.75	0.75	21208
weighted avg	0.75	0.75	0.75	21208

```
[ ] mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error: ' + str(mse))
rmse = (mean_squared_error(y_test, y_pred))**(0.5)
print('Root Mean Squared Error: ' + str(rmse))

Mean Squared Error: 0.24976423990946814
Root Mean Squared Error: 0.49976418430042396

[ ] matrix=classification_report(y_test,y_pred)
print(matrix)
```

	precision	recall	f1-score	support
0.0	0.76	0.73	0.74	10481
1.0	0.74	0.77	0.76	10727
accuracy			0.75	21208
macro avg	0.75	0.75	0.75	21208
weighted avg	0.75	0.75	0.75	21208

Activity 2: Comparing model accuracy before & after applying hyperparameter tuning

After seeing, the results of models are displayed as output. From the three models the random forest regressor model is performing well & Hyperparameter tuning For this model (it is not required)

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
[ ]
import pickle
with open("drive/MyDrive/Diabetes_Predict4.pickle","wb") as file:
    pickle.dump(lg,file)
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Pages:

For this project create two HTML files namely

index.html

prediction.html

result.html

and save them in the templates folder. Refer this [ENTER THE LINK](#) for templates, static and python file

Activity 2.2: Build Python code:

Import the libraries in python file

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (`__name__`) as argument.

```

from flask import Flask, render_template, request
import pandas as pd
from sklearn.preprocessing import StandardScaler
import pickle
import numpy as np
import pandas as pd

df = pd.read_csv('diabetes_binary_5050split_health_indicators_BRFSS2015.csv')
app = Flask(__name__, template_folder='templates')

model = pickle.load(open('Diabetes_Predict4.pickle', 'rb'))

@app.route('/', methods=['GET', 'POST'])
def home():
    return render_template('index.html')

@app.route('/input', methods=['GET', 'POST'])
def predictions():
    if request.method == 'POST':
        Sex = float(request.form['Sex'])
        HighBP = float(request.form['HighBP'])
        HighChol = float(request.form['HighChol'])
        CholCheck = float(request.form['CholCheck'])
        Fruits = float(request.form['Fruits'])
        Veggies = float(request.form['Veggies'])
        Age = float(request.form['Age'])
        BMI = float(request.form['BMI'])
        Smoker = float(request.form['Smoker'])
        Stroke = float(request.form['Stroke'])
        HyvAlcoholConsump = float(request.form['HyvAlcoholConsump'])
        AnyHealthcare = float(request.form['AnyHealthcare'])
        DiffWalk = float(request.form['DiffWalk'])
        HeartDiseaseorAttack = float(request.form['HeartDiseaseorAttack'])
        PhysActivity = float(request.form['PhysActivity'])
        NoDocbcCost = float(request.form['NoDocbcCost'])
        GenHlth = float(request.form['GenHlth'])
        MentHlth = float(request.form['MentHlth'])
        PhysHlth = float(request.form['PhysHlth'])
        Education = float(request.form['Education'])
        Income = float(request.form['Income'])
        input_data = [HighBP, HighChol, CholCheck, BMI, Smoker, Stroke, HeartDiseaseorAttack, PhysActivity, Fruits, Veggies, HyvAlcoholConsump, AnyHealthcare, NoDocbcCost, GenHlth, MentHlth, PhysHlth, DiffWalk, Sex, Age, Education, Income]
        print(input_data)
        print(type(input_data[0]))

        X = df.drop(columns='Diabetes_binary', axis=1)
        stand = StandardScaler()
        fit_Transform = stand.fit(X)

        input_data_as_ndarray = np.array(input_data)
        input_data_resaped = input_data_as_ndarray.reshape(1, -1)
        input_data_resaped = fit_Transform.transform(input_data_resaped)
        print(input_data_resaped)
        prediction = model.predict(input_data_resaped)
        print(prediction[0])
        if (prediction[0] == 0.0):
            return render_template('result.html', result = 'Diabetes Absent', txtclr = 'green', name = __name__)
        elif (prediction[0] == 1.0):
            return render_template('result.html', result='Diabetes Present', txtclr = 'red', name=__name__)
    return render_template('predict.html')

if __name__ == "__main__":
    app.run(debug=True)

```

Activity 2.3: Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

Now, Go the web browser and write the localhost url (<http://127.0.0.1:5000>) to get the below result

Enter Your Details

Sex:

Female

Age:

70-74

How many years of education have you had?

Grades 1 through 7

What is your estimated total household income?

\$30000 - \$40000

Do you have any health care coverage?

Yes

Was there a time in the past year when you needed to see a doctor but could not because of costs?

No

Do you have high blood pressure?

Yes

Do you have high cholesterol?

Yes

When was the last time you got your cholesterol checked?

Last 5 Years

What is your daily fruit intake?

Almost never

What is your daily vegetable intake?

Almost never

How many cigarettes have you smoked in your entire life?

None

Have you ever suffered a stroke?

No

Are you a heavy drinker?

No

Do you have any difficulty walking up stairs?

Yes

Have you ever had a heart attack or suffered from heart disease?

No

Have you been physically active in the last 30 days?

No

How would you rate your general health?

Do you have any difficulty waking up stairs?

Yes

Have you ever had a heart attack or suffered from heart disease?

No

Have you been physically active in the last 30 days?

No

How would you rate your general health?

Fair

For how many days during the past 30 days was your mental health not good?

0

For how many days during the past 30 days was your physical health not good?

0

Body Mass Index:

18

Submit

Your results are:

Diabetes Present