INDEX

PROBLEM STATEMENT

You are tasked with developing a machine learning model to predict whether an individual is at risk of having diabetes based on a set of relevant features and historical data. Diabetes is a chronic medical condition that affects millions of people worldwide, and early detection can significantly improve the quality of life for those at risk.

SOLUTION-

Diabetes is a chronic health condition that affects millions of people worldwide. Early detection and prediction of diabetes can significantly contribute to effective management and prevention of complications. In this context, the goal is to develop a binary classification model that predicts whether a person is likely to have diabetes based on certain features.

1. Data Preprocessing:

- Handling Missing Data: Identify and handle missing data appropriately, either by imputation or removal of incomplete records.

- Feature Scaling: Standardize or normalize numerical features to ensure that they have a similar scale.

- Categorical Variables: Encode categorical variables using techniques like one-hot encoding.

2. Exploratory Data Analysis (EDA):

- Understand the distribution of each feature.

- Explore correlations between features.

- Visualize the distribution of the target variable (diabetes) to understand class imbalances.

3. Feature Engineering:

- Create new features that might provide additional insights.

- Consider interactions between features.

- Use domain knowledge to guide feature selection.

4. Model Selection:

- Choose appropriate binary classification algorithms such as Logistic Regression, Random Forest, Support Vector Machines, or Neural Networks.

- Split the dataset into training and testing sets to evaluate model performance.

5. Model Training:

- Train the chosen model using the training dataset.

- Fine-tune hyperparameters using techniques like grid search or random search.

6. Model Evaluation:

- Evaluate the model on the testing dataset using metrics such as accuracy, precision, recall, F1 score, and area under the ROC curve (AUC-ROC).

- Address class imbalances using techniques like oversampling, undersampling, or using class weights.

7. Model Interpretability:

- Understand the importance of different features in making predictions.

- Utilize techniques like feature importance plots or SHAP (SHapley Additive exPlanations) values.

8. Validation and Cross-Validation:

- Perform k-fold cross-validation to ensure the model's robustness.

- Validate the model's performance on different subsets of the data.

9. Optimization and Iteration:

- Iterate on the model based on feedback from validation results.

- Consider ensemble methods for improving model performance.

10. Deployment:

- Once satisfied with the model's performance, deploy it to a production environment.

- Implement a monitoring system to track the model's performance over time.

Software used-

Jupyter notebook-

Jupyter Notebook is an open-source web application that allows you to create and share interactive documents that contain live code, equations, visualizations, and narrative text. It was originally developed as an interactive computing environment for Python, but now supports over 40 different programming languages including R, Julia, and MATLAB.

Jupyter Notebook allows you to create documents called "notebooks" that consist of a series of cells, each of which can contain code, markdown, or raw text. These cells can be executed individually or together in a specific order, allowing you to explore and analyze data, create visualizations, and share your work with others.

One of the key advantages of Jupyter Notebook is its ability to display output directly inline with the code that generated it. This means that you can see the results of your code immediately, making it easier to explore and understand your data.

Jupyter Notebook also provides a wide range of tools and extensions for data analysis, machine learning, and scientific computing. It supports popular libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn, making it a powerful tool for data scientists and researchers.

Finally, Jupyter Notebook is highly customizable and extensible, with a large and active community that contributes to its development and maintenance. It can be run locally on your computer or in the cloud using services such as Microsoft Azure, Amazon Web Services, or Google Cloud Platform.

PROJECT DESCRIPTION-

Classification

Supervised machine learning techniques involve training a model to operate on a set of features and predict a label using a dataset that includes some already-known label values. You can think of this function like this, in which y represents the label we want to predict and X represents the vector of features the model uses to predict it.

y=f([x1,x2,x3,...])

Classification is a form of supervised machine learning in which you train a model to use the features (the x values in our function) to predict a label (y) that calculates the probability of the observed case belonging to each of a number of possible classes, and predicting an appropriate label. The simplest form of classification is binary classification, in which the label is 0 or 1, representing one of two classes; for example, "True" or "False"; "Internal" or "External"; "Profitable" or "Non-Profitable"; and so on.

Binary Classification

In this notebook, we will focus on an example of binary classification, where the model must predict a label that belongs to one of two classes. In this exercise, we'll train a binary classifier to predict whether or not a patient should be tested for diabetes based on some medical data.

ANN-

Artificial Neural Networks (ANNs) are computational models inspired by the intricate architecture and functioning of the human brain. Consisting of layers of interconnected nodes, or artificial neurons, ANNs excel at learning patterns and relationships within data. Input features pass through the network, and with the aid of weights, biases, and activation functions, predictions are generated. These networks are particularly effective in complex tasks such as image and speech recognition, natural language processing, and medical diagnosis. Training ANNs involves adjusting weights and biases through forward and backward propagation, minimizing prediction errors. Despite their remarkable capabilities, challenges such as overfitting and interpretability remain, driving ongoing research to enhance the efficiency and applicability of these neural networks across various domains.

Logistic Regression-

Logistic Regression is a statistical method used for binary classification problems, where the outcome variable is categorical and has two classes (e.g., 0 or 1, True or False, Yes or No). Despite its name, logistic regression is a classification algorithm rather than a regression algorithm. Here's a concise overview of logistic regression:

Formulation:

Logistic regression models the relationship between one or more independent variables and the probability of a particular outcome occurring. The logistic function (sigmoid function) is employed to ensure the output lies between 0 and 1.

Training:

The model is trained using a method called Maximum Likelihood Estimation (MLE) or optimization algorithms like gradient descent. The goal is to maximize the likelihood of observing the given set of outcomes.

Decision Boundary:

Logistic regression produces a decision boundary that separates the two classes. Instances falling on one side of the boundary are predicted as one class, and those on the other side as the other class.

Supervised Learning-

Definition:

Supervised learning is a category of machine learning where the algorithm learns a mapping from input features to a target output variable based on a labeled dataset. In supervised learning, the algorithm is provided with a set of input-output pairs, and the goal is to learn a mapping that can accurately predict the output for new, unseen inputs.

Key Components:

1. Input Features
2. Output Variable (Target)
3. Training Data
4. Model or Algorithm

DATASET –

	A	B	C	D	E	F	G	H	I	J
1	Pregnancie	Glucose	BloodPres	SkinThickn	Insulin	BMI	DiabetesPe	Age	Outcome	
2	6	148	72	35	0	33.6	0.627	50	1	
3	1	85	66	29	0	26.6	0.351	31	0	
4	8	183	64	0	0	23.3	0.672	32	1	
5	1	89	66	23	94	28.1	0.167	21	0	
6	0	137	40	35	168	43.1	2.288	33	1	
7	5	116	74	0	0	25.6	0.201	30	0	
8	3	78	50	32	88	31	0.248	26	1	
9	10	115	0	0	0	35.3	0.134	29	0	
10	2	197	70	45	543	30.5	0.158	53	1	
11	8	125	96	0	0	0	0.232	54	1	
12	4	110	92	0	0	37.6	0.191	30	0	
13	10	168	74	0	0	38	0.537	34	1	
14	10	139	80	0	0	27.1	1.441	57	0	
15	1	189	60	23	846	30.1	0.398	59	1	
16	5	166	72	19	175	25.8	0.587	51	1	
17	7	100	0	0	0	30	0.484	32	1	
18	0	118	84	47	230	45.8	0.551	31	1	
19	7	107	74	0	0	29.6	0.254	31	1	
20	1	103	30	38	83	43.3	0.183	33	0	
21	1	115	70	30	96	34.6	0.529	32	1	
22	3	126	88	41	235	39.3	0.704	27	0	
23	8	99	84	0	0	35.4	0.388	50	0	
24	7	196	90	0	0	39.8	0.451	41	1	
25	9	119	80	35	0	29	0.263	29	1	
26	11	143	94	33	146	36.6	0.254	51	1	
27	10	125	70	26	115	31.1	0.205	41	1	
28	7	147	76	0	0	39.4	0.257	43	1	
29	1	97	66	15	140	23.2	0.487	22	0	
30	13	145	82	19	110	22.2	0.245	57	0	
31	5	117	92	0	0	34.1	0.337	38	0	
32	5	109	75	26	0	36	0.546	60	0	
33	3	158	76	36	245	31.6	0.851	28	1	
34	3	88	58	11	54	24.8	0.267	22	0	
35	6	92	92	0	0	19.9	0.188	28	0	
36	10	122	78	31	0	27.6	0.512	45	0	

diabetes

FIG:1

Dataset Information:

The predictive model for diabetes was trained on a dataset comprising information from 769 individuals. The dataset was carefully curated to ensure diversity and representation of the target population. The inclusion of a substantial number of individuals in the training dataset enhances the model's ability to generalize patterns and relationships, making it robust for real-world applications.

Dataset Composition:

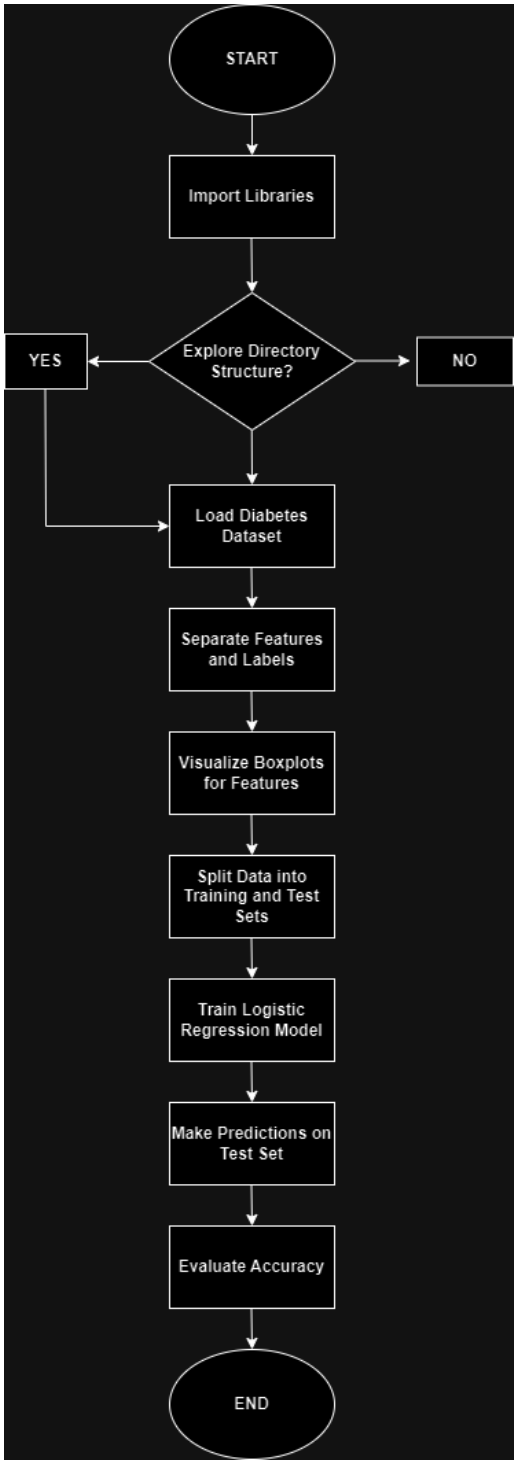
The dataset consists of relevant features such as age, BMI (Body Mass Index), blood pressure, insulin levels, family history, and other health indicators. This diverse set of features is essential for capturing the complexity of factors contributing to diabetes risk.

Data Preprocessing:

Prior to model training, meticulous data preprocessing was conducted, including the handling of missing values, feature scaling, and encoding categorical variables. These steps aimed to ensure the quality and uniformity of the dataset, providing a solid foundation for training a reliable predictive model.

FLOWCHART OF PREDICTING DIBETIES USING BINARY CLASSIFICATION

Fig:2



ALGORITHM –

Step 1: Import Necessary Libraries

Import the required libraries, such as NumPy, Pandas, Matplotlib, and scikit-learn.

Step 2: Load the Dataset

Load the diabetes dataset, which likely contains information about various health indicators and whether individuals have diabetes (the target variable).

Step 3: Explore the Dataset

Identify the features and target variable in the dataset.

Visualize boxplots for features grouped by the outcome (presence or absence of diabetes). This step helps in understanding the distribution of features for different outcomes.

Step 4: Data Splitting

Split the dataset into features (X) and labels (y).

Use the train_test_split function to split the dataset into training and testing sets. A common split might be 70% for training and 30% for testing.

Step 5: Model Training

Choose a machine learning algorithm; in this case, a logistic regression model.

Set hyperparameters, such as the regularization parameter.

Train the logistic regression model using the training data.

Step 6: Make Predictions

Use the trained model to make predictions on the test set.

Obtain the predicted labels for the test set.

Step 7: Model Evaluation

Evaluate the performance of the model using various metrics:

Calculate and print the accuracy of the model on the test set.

Generate a classification report, which includes precision, recall, and F1-score for each class.

Calculate and print overall precision and recall.

Step 8: ROC Curve and AUC

Calculate the predicted probabilities for each class on the test set.

Compute the ROC curve using the true positive rate and false positive rate.

Plot the ROC curve to visualize the model's performance in differentiating between classes.

Calculate and print the Area Under the Curve (AUC) to quantify the model's discrimination ability.

Step 9: Further Analysis (Optional)

Explore additional analyses, such as confusion matrices or feature importance, for a more in-depth understanding of the model's behavior.

Step 10: Conclusion and Reporting

Summarize the findings, including model performance metrics and any insights gained from the analysis.

PROGRAM-

```
# Step 1: Import necessary libraries
import numpy as np
import pandas as pd

# Step 2: Explore the directory structure
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Step 3: Load the diabetes dataset
diabetes = pd.read_csv('C:/Users/deshm/diabetes.csv')

# Display the first few rows of the dataset
diabetes.head()

# Step 4: Separate features and labels
features = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
label = 'Outcome'
X, y = diabetes[features].values, diabetes[label].values

# Display features and labels for the first few patients
for n in range(0, 4):
    print("Patient", str(n+1), "\n  Features:", list(X[n]), "\n  Label:", y[n])

# Step 5: Visualize boxplots for features by Outcome
from matplotlib import pyplot as plt
%matplotlib inline

for col in features:
    diabetes.boxplot(column=col, by='Outcome', figsize=(6, 6))
    plt.title(col)
plt.show()

# Step 6: Split data 70%-30% into training set and test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)

# Display the number of training and test cases
print('Training cases: %d\nTest cases: %d' % (X_train.shape[0], X_test.shape[0]))
```

```
# Step 7: Train the logistic regression model
from sklearn.linear_model import LogisticRegression

# Set regularization rate
reg = 0.01

# Train a logistic regression model on the training set
model = LogisticRegression(C=1/reg, solver="liblinear").fit(X_train, y_train)
print(model)

# Step 8: Make predictions on the test set
predictions = model.predict(X_test)
print('Predicted labels: ', predictions)
print('Actual labels:    ', y_test)

# Step 9: Evaluate accuracy
from sklearn.metrics import accuracy_score

print('Accuracy: ', accuracy_score(y_test, predictions))

# Step 10: Display classification report
from sklearn.metrics import classification_report

print(classification_report(y_test, predictions))

# Step 11: Calculate overall precision and recall
from sklearn.metrics import precision_score, recall_score

print("Overall Precision:", precision_score(y_test, predictions))
print("Overall Recall:", recall_score(y_test, predictions))

# Step 12: Calculate predicted probabilities
y_scores = model.predict_proba(X_test)
print(y_scores[:10,])

# Step 13: Plot ROC curve
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_scores[:, 1])

# Plot ROC curve
fig = plt.figure(figsize=(6, 6))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()

# Step 14: Calculate and print AUC
from sklearn.metrics import roc_auc_score

auc = roc_auc_score(y_test, y_scores[:, 1])
print('AUC: ' + str(auc))
```

RESULT & DISSCUSION-

```
In [5]: diabetes = pd.read_csv('C:/Users/deshm/diabetes.csv')
        diabetes.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Fig : 3

This is the dataset which read by the function using pd.read.

```
In [6]: # Separate features and labels
        features = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
        label = 'Outcome'
        X, y = diabetes[features].values, diabetes[label].values

        for n in range(0,4):
            print("Patient", str(n+1), "\n  Features:",list(X[n]), "\n  Label:", y[n])

Patient 1
  Features: [6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0]
  Label: 1
Patient 2
  Features: [1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0]
  Label: 0
Patient 3
  Features: [8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0]
  Label: 1
Patient 4
  Features: [1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0]
  Label: 0
```

Fig: 4

This program reads the dataset and returns the desirable outcome with conditions which have been applied in the code.

```
In [10]: predictions = model.predict(X_test)
         print('Predicted labels: ', predictions)
         print('Actual labels:    ' ,y_test)

Predicted labels:  [1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 1
 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0
 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
 0 1 1 1 0 0 0 0 0]
Actual labels:     [1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0
 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0
 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1
 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0
 1 1 0 0 1 1 0 0 0]
```

Fig: 5

This program returns the predicted and the actual output of the dataset

```
In [11]: from sklearn.metrics import accuracy_score

         print('Accuracy: ', accuracy_score(y_test, predictions))

         Accuracy:  0.7792207792207793
```

Fig: 6

This line of code returns the accuracy of the program.

CONCLUSION-

The provided program is for a binary classification task to predict diabetes based on certain health indicators.

1. Data Loading and Exploration:

- The dataset was loaded from a CSV file containing information on various health indicators and the presence or absence of diabetes.

2. Data Visualization:

- Boxplots were used to visualize the distribution of different health indicators for individuals with and without diabetes.

3. Data Splitting:

- The dataset was split into training and testing sets, with 70% of the data used for training the model and 30% for testing its performance.

4. Model Training:

- A logistic regression model was trained on the training set to learn the relationship between the health indicators and the presence or absence of diabetes.

5. Model Evaluation:

- The trained model was evaluated on the testing set using various metrics.

- The accuracy of the model on the test set was calculated, indicating the proportion of correctly predicted instances.

6. Classification Report:

- A detailed classification report was generated, including precision, recall, and F1-score for each class (presence and absence of diabetes).

7. Precision and Recall:

- Overall precision and recall were calculated, providing insights into the model's ability to correctly identify positive cases and avoid false positives.

8. ROC Curve and AUC:

- The Receiver Operating Characteristic (ROC) curve was plotted to visualize the trade-off between true positive rate and false positive rate.

- The Area Under the Curve (AUC) was calculated, providing a single metric to quantify the model's discrimination ability.

9. Conclusion:

- The program demonstrated the process of loading, exploring, and preparing a dataset for a binary classification task.

- A logistic regression model was trained to predict diabetes based on health indicators.

- Model performance was evaluated using various metrics, including accuracy, precision, recall, and AUC.

The program is a comprehensive example of building and evaluating a binary classification model for predicting diabetes, providing insights into the model's performance and its ability to make accurate predictions on unseen data.

REFERENCE-

- https://www.kaggle.com/datasets?tags=14201-Binary+Classification
- https://www.ibm.com/topics/supervised-learning
- https://www.geeksforgeeks.org/understanding-logistic-regression/
- https://www.javatpoint.com/machine-learning-models
- https://app.diagrams.net/
- https://jupyter.org/

Appendix-

- https://github.com/arunvdeshmukh/predicting_diabetes_using_Binary_classification