AI BASED DIABETES PREDICTION SYSTEM

TEAM NAME : Proj_224748_Team1

TOPIC : PREDICTING DIABETES USING MACHINE LEARNING

BY :

B. HEMALATHA

R. SATHYABAMA

G. GURU PRIYA

C. ROOPAVATHY

J. ABINAYA

INTRODUCTION :

- Diabetes prediction is vital in the modern world due to its potential to provide early intervention and prevention, improve health outcomes, reduce healthcare costs, and personalize care. Predictive models empower individuals to make lifestyle changes, optimize resource allocation, inform public health policies, support research, and enhance the quality of life. With the rising prevalence of diabetes and its associated burdens, accurate prediction models are crucial in healthcare and public health efforts.
- Diabetes is a prevalent and chronic metabolic disorder that affects millions of people worldwide, with significant implications for public health and individual well-being. Early detection and effective management of diabetes are crucial for preventing complications and improving patients' quality of life. In this context, machine learning has emerged as a powerful tool to assist healthcare professionals in predicting diabetes risk, diagnosing the condition, and personalizing treatment plans. This introduction explores the application of machine learning in diabetes prediction, highlighting its potential to revolutionize healthcare.
- Diabetes prediction using AI is a cutting-edge approach that leverages artificial intelligence and machine learning to assess an individual's risk of developing diabetes or aid in early diagnosis. By analyzing various health and lifestyle data, AI models can provide valuable insights to healthcare professionals and individuals, enabling early intervention and preventive measures, ultimately improving patient outcomes and reducing healthcare costs.

- Building and deploying diabetes prediction models using machine learning comes with challenges, such as the need for high-quality, diverse datasets, potential biases in data collection, and ensuring model interpretability. Ethical concerns related to patient privacy and data security also demand attention.
- Diabetes prediction using machine learning is a promising approach that offers new horizons in healthcare. It empowers healthcare professionals and individuals to make more informed decisions, ensuring early diagnosis and personalized care. As the field continues to evolve, it holds the potential to transform the landscape of diabetes management and contribute significantly to improved health outcomes for those at risk of or living with diabetes.
- **DatasetLink:** https://www.kaggle.com/datasets/mathchi/diabetes-data-set

GIVEN DATA SET :

SL. NO	PREGNANCIES	GLUCOSE	BLOOD PRESSURE	SKIN THICKNESS	INSULIN	BMI	DIABETES PEDIGREE FUNCTION	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

Diabetes prediction using machine learning is an innovative approach to assist healthcare professionals and patients in identifying individuals who are at risk of developing diabetes or managing the condition more effectively. Diabetes is a chronic metabolic disorder characterized by high blood sugar levels, and early detection and management are essential for preventing complications. Machine learning techniques offer a powerful tool for risk assessment, diagnosis, and personalized treatment plans.

AN INTRODUCTION TO DIABETES PREDICTION USING MACHINE LEARNING :

1. Diabetes Types: There are primarily two types of diabetes: Type 1 and Type 2. Type 1 is an autoimmune condition, while Type 2 is often linked to lifestyle factors like obesity and physical inactivity. Machine learning can be applied to both types for risk prediction and management.

2. Data Collection: To build an effective diabetes prediction model, a large dataset is required. This dataset typically includes information about patients' medical history, demographics, lifestyle, genetics, and most importantly, their blood glucose levels.

3. Feature Selection: Feature selection is the process of choosing relevant attributes from the dataset that have the most significant impact on diabetes prediction. Important features may include age, BMI, family history, physical activity, and diet.

4. Model Selection: Various machine learning algorithms can be applied, such as logistic regression, decision trees, random forests, support vector machines, and neural networks. The choice of model depends on the specific dataset and goals of the prediction.

5. Data Preprocessing: Data preprocessing is crucial for cleaning and transforming the data. This includes handling missing values, scaling, and encoding categorical variables.

6. Training and Testing: The dataset is split into training and testing sets to evaluate the model's performance. Cross-validation techniques may also be used to ensure robustness.

7. Model Evaluation: Various evaluation metrics like accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic (ROC-AUC) curve are used to assess the model's performance.

8. Hyperparameter Tuning: Fine-tuning the model's hyperparameters can improve its performance and generalization.

9. Diabetes Risk Assessment: Once the model is trained and validated, it can be used to assess an individual's risk of developing diabetes based on their input data.

10. Predictive Maintenance: Machine learning models can also be used for ongoing monitoring of patients with diabetes to predict complications or assess the effectiveness of their treatment plans.

11.Personalized Care: Machine learning can help tailor treatment plans and lifestyle recommendations to individual patients based on their risk profiles and other factors.

12. Challenges: Challenges in diabetes prediction using machine learning include the need for high-quality, diverse datasets, potential biases in data collection, and model interpretability.

13. Ethical Considerations: Privacy and data security are critical concerns when dealing with healthcare data. Ensuring patient consent and complying with data protection regulations is essential.

Here's a list of tools and software commonly used in the process:

Building a diabetes prediction model in AI requires a combination of tools and software for data preparation, model development, training, and evaluation. Here is a list of essential tools and software for creating a diabetes prediction model:

1. Programming Languages:

Python: Python is the most commonly used programming language for AI and machine learning. It offers a wide range of libraries and frameworks for data analysis and model development.

2. Data Manipulation and Analysis:

Pandas: Pandas is a Python library for data manipulation and analysis. It's essential for handling datasets and data preprocessing.

3. Machine Learning Libraries:

Scikit-Learn: Scikit-Learn is a comprehensive machine learning library in Python. It provides a wide range of tools for building predictive models, including classification and regression algorithms.

4. Deep Learning Frameworks:

TensorFlow: TensorFlow is an open-source deep learning framework developed by Google. It is widely used for building neural network models, which can be valuable for more complex diabetes prediction tasks.

5. Data Visualization:

Matplotlib: Matplotlib is a Python library for creating static, animated, and interactive visualizations to explore your data.

Seaborn: Seaborn is a higher-level data visualization library that works well with Pandas dataframes and is often used to create more aesthetically pleasing plots.

6. Cloud Services:

Cloud platforms like Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure provide cloud-based resources for data storage, computation, and machine learning model deployment. These platforms offer pre-configured environments for AI development.

7. AutoML Platforms:

If we prefer an automated approach, AutoML platforms like Google AutoML and Azure AutoML provide tools for building machine learning models with minimal coding. They can be useful for those without extensive data science expertise.

8. Specialized Healthcare Datasets:

Access to healthcare datasets, including diabetes-related data, is crucial for building a diabetes prediction model. These datasets may be obtained from healthcare institutions, research organizations, or open data sources.

9. Model Evaluation Tools:

We'll need tools to evaluate the performance of your diabetes prediction model. Scikit-Learn and TensorFlow provide functions for metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and more.

10. Version Control:

Git and platforms like GitHub or GitLab are essential for version control, collaboration, and tracking changes in our AI project.

11. Ethical Considerations and Compliance Tools:

If we're working with healthcare data, consider tools and guidelines for ensuring data privacy and compliance with regulations like HIPAA (Health Insurance Portability and Accountability Act).

It's important to choose the tools and software that align with our project's specific needs and our expertise in AI and machine learning. Additionally, consider the ethical and legal aspects of working with healthcare data, which may require specialized tools and practices to ensure data privacy and security.

DESIGN THINKING APPROACH FOR DIABETES PREDICTION USING AI:

1.Empathize

Stakeholder Interviews

To better understand the needs of our users, we conducted interviews with healthcare providers, data scientists, and patients to gather insights.

Key Findings:

- Healthcare providers need a tool that can aid in early diabetes diagnosis.
- Patients require personalized risk assessment and guidance for diabetes prevention.
- Data scientists emphasize the importance of accurate predictions and continuous improvement.

2.Define

Problem Statement

Our challenge is to develop an AI-based diabetes prediction system that enables healthcare providers to:

- Identify individuals at risk of diabetes early.
- Offer personalized healthcare recommendations.
- Optimize resource allocation and patient management.

3.Ideate

Brainstorming Solutions

- AI Model Development: Create a machine learning model that accurately predicts diabetes risk based on various data sources, such as medical records, lifestyle factors, and genetic information.
- User-Friendly Interface: Design an intuitive user interface for healthcare providers to input patient data and receive predictions.
- Personalized Recommendations: Develop a system that offers personalized recommendations for patients to mitigate their diabetes risk.
- Data Integration: Integrate various data sources, including electronic health records and wearable device data, to provide a comprehensive assessment.
- Continuous Improvement: Implement feedback loops and regular model updates to ensure the system remains accurate.

4.Prototype

System Prototyping

We'll create a prototype of the diabetes prediction system, including:

- A machine learning model that predicts diabetes risk.
- A user interface for healthcare providers to input data and receive predictions.
- A patient portal for personalized recommendations.
- A data integration framework.
- A continuous improvement pipeline.

5.Test and Feedback

Usability Testing

We'll conduct usability tests with healthcare providers and patients to gather feedback on the system's usability, accuracy, and user-friendliness.

6.Implement

System Development

Based on feedback and further iterations, we'll develop the final diabetes prediction system.

7.Deliver

User Training

We'll provide training to healthcare providers on how to use the system effectively.

DESIGN INTO INNOVATION :

1. Data Collection:

Gather a comprehensive dataset that includes features such as pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, age and outcome.

2.Data Preprocessing:

Clean the data by handling missing values, outliers, and encoding categorical variables. Standardize or normalize numerical features as necessary.

1.Importing Required Libraries

2. Loading the Dataset

3. Exploratory Data Analysis

a. Understanding the dataset

Head of the dataset

Shape of the data set

Types of columns

Information about data set

Summary of the data set

b. Data Cleaning

Dropping duplicate values

Checking NULL values

Checking for 0 values

4. Data Visualization

Here we are going to plot :-

Count Plot :- to see if the dataset is balanced or not

Histograms :- to see if data is normally distributed or skewed

Box Plot :- to analyse the distribution and see the outliers

Scatter plots :- to understand relationship between any two variables

Pair plot :- to create scatter plot between all the variables

5.Feature Selection

6.Handling Outliers

7.Split the Data Frame into X and Y

8.Train Test Split

1. Import Required Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
for filename in filenames:
print(os.path.join(dirname, filename))
```

2. Loading the dataset

In [2]:

```
df=pd.read_csv("../input/pima-indians-diabetes-database/diabetes.csv")
```

3. Exploratory Data Analysis

In[3]:

```
df.head() #get familier with dataset, display the top 5 data records
```

Out[3]:

SL. NO.	PREGNANCIES	GLUCOSE	BLOOD PRESSURE	SKIN THICKNESS	INSULIN	BMI	DIABETES PEDIGREE FUNCTION	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

In [4]:

```
df.shape
```

Out[4]:

```
(768, 9)
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Pregnancies', 'Glucose', 'Blood Pressure', 'Skin Thickness',
'Insulin' ,'BMI', 'Diabetes Pedigree Function', 'Age', 'Outcome'],
dtype='object')
```

In [6]:

```
df.dtypes
```

Out[6]:

```
Pregnancies                 int64
Glucose                     int64
Blood Pressure              int64
Skin Thickness              int64
Insulin                     int64
BMI                       float64

dtype: object
```

In [7]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#        Column                  Non-Null Count        Dtype
---       ------                  --------------        -----
0        Pregnancies             768 non-null          int64
1        Glucose                 768 non-null          int64
2        Blood Pressure          768 non-null          int64
3        Skin Thickness          768 non-null          int64
4        Insulin                 768 non-null          int64
5        BMI                      768 non-null         float64
6        Diabetes Pedigree
           Function              768 non-null          float64
7        Age                     768 non-null          int64
8       Outcome                  768 non-null          int64

dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [8]:

```
df.describe()
# count :- the number of NoN-empty rows in a feature.
# mean :- mean value of that feature.
# std :- Standard Deviation Value of that feature.
```

 # min :- minimum value of that feature.
 # max :- maximum value of that feature.
 # 25%, 50%, and 75% are the percentile/quartile of each features.

Out[8]:

	PREGNANCIES	GLUCOSE	BLOOD PRESSURE	SKIN THICKNESS	INSULIN	BMI	DIABETES PEDIGREE FUNCTION	AGE	OUTCOME
COUNT	768.000000	768.000000	768.000000	768.000000	768.000000	768.0000000	768.000000	768.000000	768.000000
MEAN	3.845052	120.894531	69.105496	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
STD	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
MIN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
MAX	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE SELECTION, MODEL TRAINING AND EVALUTION

FEATURE SELECTION:

Feature selection is the process of identifying and selecting the most relevant features from a dataset to improve the performance of a machine learning model. Feature selection in diabetes prediction involves choosing the most relevant patient characteristics from dataset to build an accurate model for predicting diabetes. This process helps improve model performance, interpretability, and resource efficiency in healthcare by focusing on the most important data points.

MODEL TRAINING:

Model training is the process of feeding the selected features to a machine learning algorithm and allowing it to learn the relationship between the features and the target variable (i.e. diabetes prediction). Once the model is trained, it can be used to predict the diabetes of new people, given their features.

MODEL EVALUATION:

Model evaluation is the process of assessing the performance of a machine learning or statistical model by comparing its predictions to actual outcomes. It involves using various metrics, data splitting techniques, and visualizations to determine how well the model is working. Model evaluation in diabetes prediction involves assessing the performance of a machine learning or statistical model specifically designed to predict diabetes. This process includes using various metrics to measure the model's accuracy, precision, recall, and other relevant criteria. The goal is to determine how well the model can predict diabetes in real-world scenarios. Model evaluation helps ensure that the model is effective, reliable, and suitable for its intended purpose, such as early diagnosis or risk assessment for diabetes.

FEATURE SELECTION :

1.Identify the target variable : This is the variable that we want to predict, such as diabetes prediction.

2. Explore the data: This will help to understand the relationships between the different features and the target variable.

3.Feature Engineering: Creating new variables is important for models. But we need to create a logical new variable. For this data set, some new variables were created according to BMI, Insulin and glucose variables.

```
NewBMI = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2", "Obesity 3"], dtype = "category")
df["NewBMI"] = NewBMI
df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]
df.loc[(df["BMI"] > 24.9) & (df["BMI"] <= 29.9), "NewBMI"] = NewBMI[2]
df.loc[(df["BMI"] > 29.9) & (df["BMI"] <= 34.9), "NewBMI"] = NewBMI[3]
df.loc[(df["BMI"] > 34.9) & (df["BMI"] <= 39.9), "NewBMI"] = NewBMI[4]
df.loc[(df["BMI"] > 39.9 ,"NewBMI"] = NewBMI[5]
df.head()
```

SL. NO.	PREGNANCIES	GLUCOSE	BLOOD PRESSURE	SKIN THICKNESS	INSULIN	BMI	DIABETES PEDIGREE FUNCTION	AGE	OUT-COME	NEW BMI
0	6	148	72.0	35.0	169.5	33.6	0.627	50	1	OBESITY 1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	OVERWEIGHT
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	NORMAL
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	OVERWEIGHT
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	OBESITY 3

```
def set_insulin(row):
if row["Insulin"] >= 16 and row["Insulin"] <= 166:
return "Normal"
else:
return "Abnormal"
df = df.assign(NewInsulinScore=df.apply(set_insulin, axis=1))
df.head()
```

SL NO	PREGNANCIES	GLUCOSE	BLOOD PRESSURE	SKIN THICKNESS	INSULIN	BMI	DIABETES PEDIGREE FUNCTION	AGE	OUTCOME	NEW BMI	NEW INSULIN SCORE
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	OBESITY 1	ABNORMAL
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	OVER WEIGHT	NORMAL
2	8	183.0	64.0	32.0	168.5	23.2	0.672	32	1	NORMAL	ABNORMAL
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	OVER WEIGHT	NORMAL
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	OBESITY 3	ABNORMAL

```
NewGlucose = pd.Series(["Low", "Normal", "Overweight", "Secret", "High"], dtype = "category")
df["NewGlucose"] = NewGlucose
df.loc[df["Glucose"] <= 70, "NewGlucose"] = NewGlucose[0]
df.loc[(df["Glucose"] > 70) & (df["Glucose"] <= 99), "NewGlucose"] = NewGlucose[1]
df.loc[(df["Glucose"] > 99) & (df["Glucose"] <= 126), "NewGlucose"] = NewGlucose[2]
df.loc[df["Glucose"] > 126 ,"NewGlucose"] = NewGlucose[3]
df.head()
```

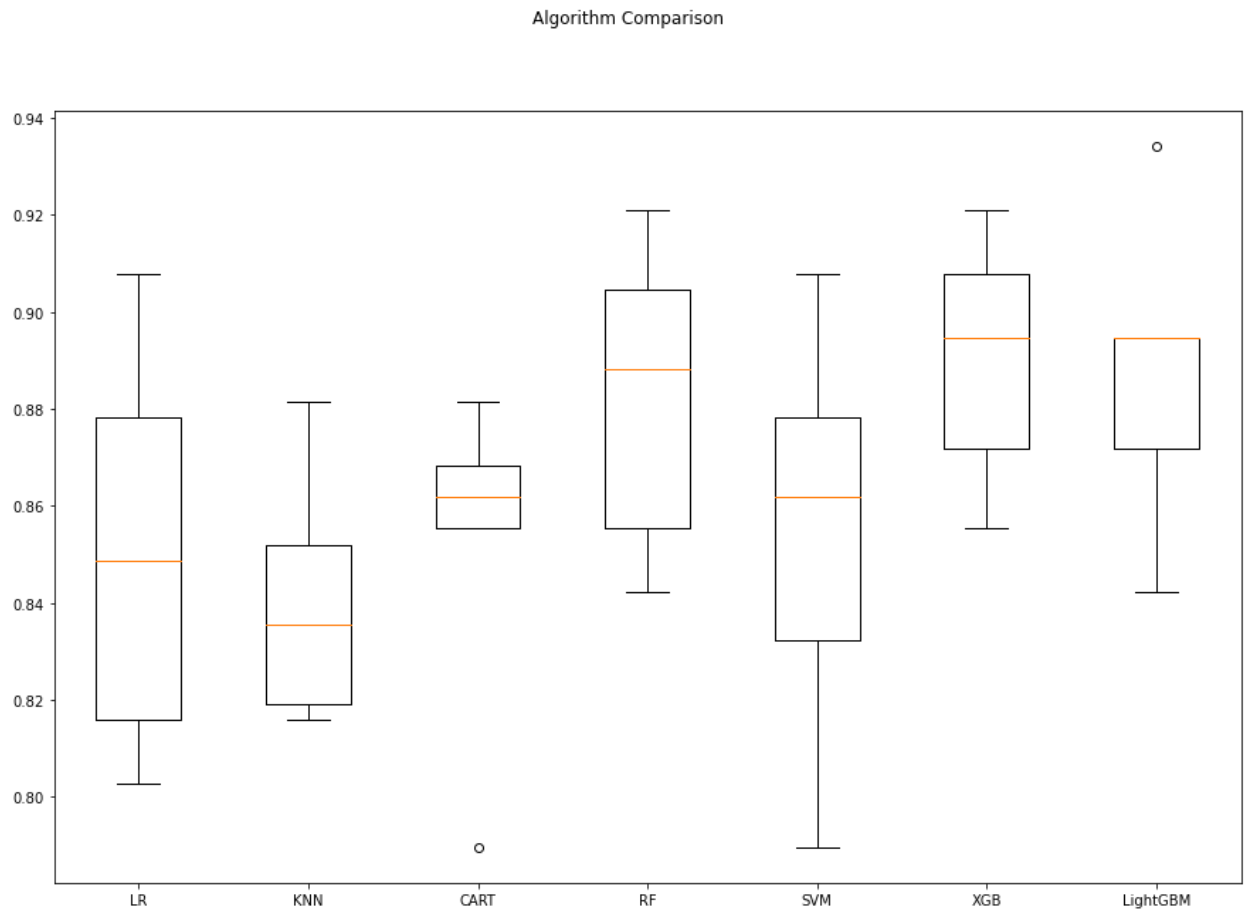
PRE NANCIES	GLUCOSE	BLOOD PRESSURE	SKIN THICKNESS	INSULIN	BMI	DIABETES PEDIGREE FUNCTION	AGE	OUTC OME	NEW BMI	NEW INSULIN SCORE	NEW GLUCOSE
6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	OBESITY 1	ABNORMAL	SECRET
1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	OVER WEIGHT	NORMAL	NORMAL
8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	NORMAL	ABNORMAL	SECRET
1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	OVER WEIGHT	NORMAL	NORMAL
0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	OBESITY 3	ABNORMAL	SECRET

MODEL TRAINING :

The process aims to create a model that can predict diabetes status based on input data, such as a patient's characteristics, and it often involves fine-tuning and testing to improve accuracy.

1.Base Model :

```
models = []
models.append(('LR', LogisticRegression(random_state = 12345)))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state = 12345)))
models.append(('RF', RandomForestClassifier(random_state = 12345)))
models.append(('SVM', SVC(gamma='auto', random_state = 12345)))
models.append(('XGB', GradientBoostingClassifier(random_state = 12345)))
models.append(("LightGBM", LGBMClassifier(random_state = 12345)))
results = []
names =[]
for name, model in models:
kfold = KFold(n_splits = 10, random_state = 12345)
cv_results = cross_val_score(model, X, y, cv = 10,
scoring= "accuracy")
results.append(cv_results)
names.append(name)
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)
fig = plt.figure(figsize=(15,10))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
LR: 0.848684 (0.036866)
KNN: 0.840789 (0.023866)
CART: 0.857895 (0.024826)
RF: 0.881579 (0.026316)
SVM: 0.853947 (0.036488)
XGB: 0.890789 (0.020427)
LightGBM: 0.885526 (0.024298)
```



2. Random Forests Tuning :

When using Random Forest for diabetes prediction, it's essential to tune the model's hyperparameters to achieve the best possible performance. To tune the hyperparameters for diabetes prediction, we can use techniques like grid search or random search. These methods systematically explore different hyperparameter combinations and evaluate the model's performance using cross-validation. The goal is to identify the hyperparameter settings that produce the best accuracy and reliability in predicting diabetes status based on the given features.

rf_params = {"n_estimators" :[100,200,500,1000],

```
"max_features": [3,5,7],
"min_samples_split": [2,5,10,30],
"max_depth": [3,5,8,None]}
rf_model = RandomForestClassifier(random_state = 12345)
gs_cv = GridSearchCV(rf_model,
rf_params,
cv = 10,
n_jobs = -1,
verbose = 2).fit(X, y)
Fitting 10 folds for each of 192 candidates, totalling 1920 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done 33 tasks | elapsed: 10.4s
[Parallel(n_jobs=-1)]: Done 154 tasks | elapsed: 48.2s
[Parallel(n_jobs=-1)]: Done 357 tasks | elapsed: 1.9min
[Parallel(n_jobs=-1)]: Done 640 tasks | elapsed: 3.5min
[Parallel(n_jobs=-1)]: Done 1005 tasks | elapsed: 5.7min
[Parallel(n_jobs=-1)]: Done 1450 tasks | elapsed: 8.4min
[Parallel(n_jobs=-1)]: Done 1920 out of 1920 | elapsed: 11.4min finished
gs_cv.best_params_
{'max_depth': 8,
'max_features': 7,
'min_samples_split': 2,
'n_estimators': 500}
```

MODEL TUNING

1) Random Forests Tuning

```
In [59]:
rf_params = {"n_estimators" :[100,200,500,1000],
        "max_features": [3,5,7],
        "min_samples_split": [2,5,10,30],
```

```
        "max_depth": [3,5,8,None]}
```

In [60]:

```
rf_model = RandomForestClassifier(random_state = 12345)
```

In [61]:

```
gs_cv = GridSearchCV(rf_model,
                     rf_params,
                     cv = 10,
                     n_jobs = -1,
                     verbose = 2).fit(X, y)
```

```
Fitting 10 folds for each of 192 candidates, totalling 1920 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   10.4s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   48.2s
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done 1005 tasks      | elapsed:  5.7min
[Parallel(n_jobs=-1)]: Done 1450 tasks      | elapsed:  8.4min
[Parallel(n_jobs=-1)]: Done 1920 out of 1920 | elapsed: 11.4min finished
```

In [62]:

```
gs_cv.best_params_
```

Out[62]:

```
{'max_depth': 8,
 'max_features': 7,
 'min_samples_split': 2,
 'n_estimators': 500}
```

1.1) Final Model Installation

In [63]:

```
rf_tuned = RandomForestClassifier(**gs_cv.best_params_)
```

In [64]:

```
rf_tuned = rf_tuned.fit(X,y)
```

In [65]:

```
cross_val_score(rf_tuned, X, y, cv = 10).mean()
```

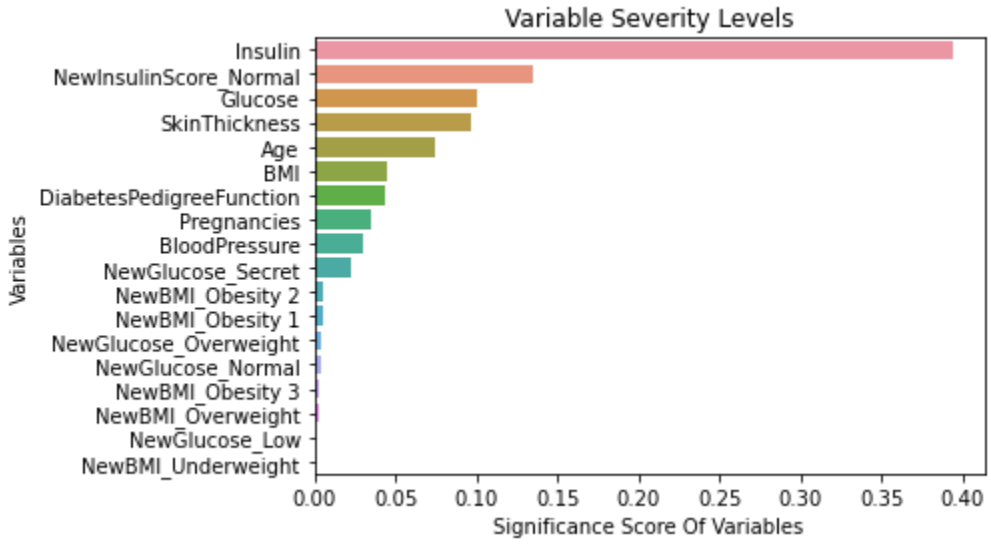
Out[65]:

```
0.8921052631578947
```

In [66]:

```
feature_imp = pd.Series(rf_tuned.feature_importances_,
feature_imp = pd.Series(rf_tuned.feature_importances_,
                        index=X.columns).sort_values(ascending=False)

sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Significance Score Of Variables')
plt.ylabel('Variables')
plt.title("Variable Severity Levels")
plt.show()
```



LIGHTGBM TUNING

In [67]:

```
lgbm = LGBMClassifier(random_state = 12345)
```

In [68]:

```
lgbm_params = {"learning_rate": [0.01, 0.03, 0.05, 0.1, 0.5],
               "n_estimators": [500, 1000, 1500],
```

```
            "max_depth":[3,5,8]}
```

In [69]:

```
gs_cv = GridSearchCV(lgbm,
                     lgbm_params,
                     cv = 10,
                     n_jobs = -1,
                     verbose = 2).fit(X, y)
```

```
Fitting 10 folds for each of 45 candidates, totalling 450 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:    3.7s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   25.3s
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:   57.4s
[Parallel(n_jobs=-1)]: Done 450 out of 450 | elapsed:  1.1min finished
```

In [70]:

```
gs_cv.best_params_
```

Out[70]:

```
{'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 1000}
```

2.1) Final Model Installation

In [71]:

```
lgbm_tuned = LGBMClassifier(**gs_cv.best_params_).fit(X,y)
```

In [72]:

```
cross_val_score(lgbm_tuned, X, y, cv = 10).mean()
```

Out[72]:

```
0.8960526315789474
```

In [73]:

```
feature_imp = pd.Series(lgbm_tuned.feature_importances_,
                        index=X.columns).sort_values(ascending=False)

sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Significance Score Of Variables')
```

```
plt.ylabel('Variables') xgb_params = {
    "learning_rate": [0.01, 0.1, 0.2, 1],
    "min_samples_split": np.linspace(0.1, 0.5, 10),
    "max_depth":[3,5,8],
    "subsample":[0.5, 0.9, 1.0],
    "n_estimators": [100,1000]}
```

In [76]:

```
xgb_cv_model  = GridSearchCV(xgb,xgb_params, cv = 10, n_jobs = -1, ver
bose = 2).fit(X, y)
```

```
Fitting 10 folds for each of 720 candidates, totalling 7200 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   42.3s
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done 1005 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done 1450 tasks      | elapsed:  8.4min
[Parallel(n_jobs=-1)]: Done 1977 tasks      | elapsed: 11.6min
[Parallel(n_jobs=-1)]: Done 2584 tasks      | elapsed: 14.8min
[Parallel(n_jobs=-1)]: Done 3273 tasks      | elapsed: 19.3min
[Parallel(n_jobs=-1)]: Done 4042 tasks      | elapsed: 23.6min
[Parallel(n_jobs=-1)]: Done 4893 tasks      | elapsed: 28.7min
[Parallel(n_jobs=-1)]: Done 5824 tasks      | elapsed: 34.5min
[Parallel(n_jobs=-1)]: Done 6837 tasks      | elapsed: 40.9min
[Parallel(n_jobs=-1)]: Done 7200 out of 7200 | elapsed: 43.2min finished
```

In [77]:

```
xgb_cv_model.best_params_
```

Out[77]:

```
{'learning_rate': 0.1,
 'max_depth': 5,
 'min_samples_split': 0.1,
 'n_estimators': 100, 'subsample': 1.0}
```

3.1) Final Model Installation

In [78]:

```
xgb_tuned = GradientBoostingClassifier(**xgb_cv_model.best_params_).
fit(X,y)
```

In [79]:

```
cross_val_score(xgb_tuned, X, y, cv = 10).mean()
```

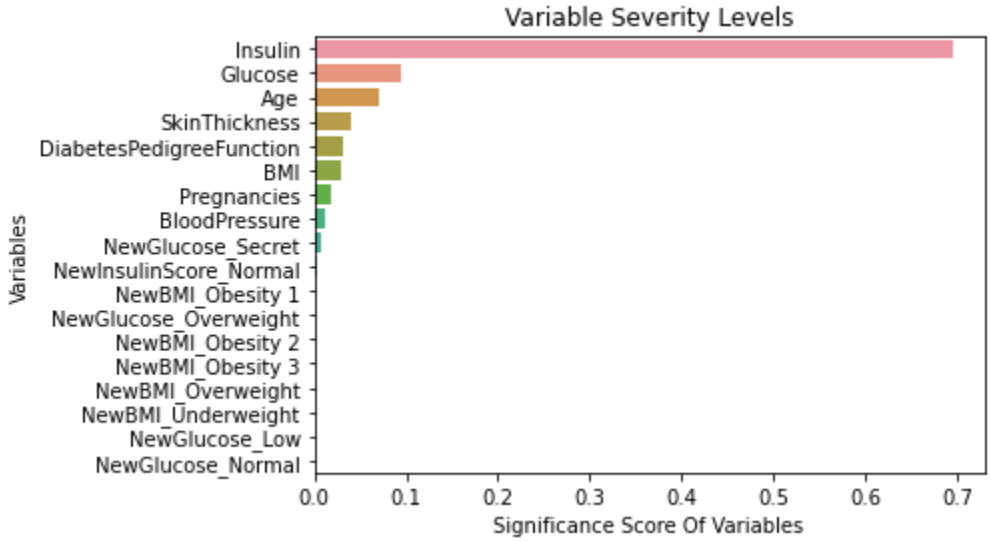
Out[79]:

```
0.9013157894736843
```

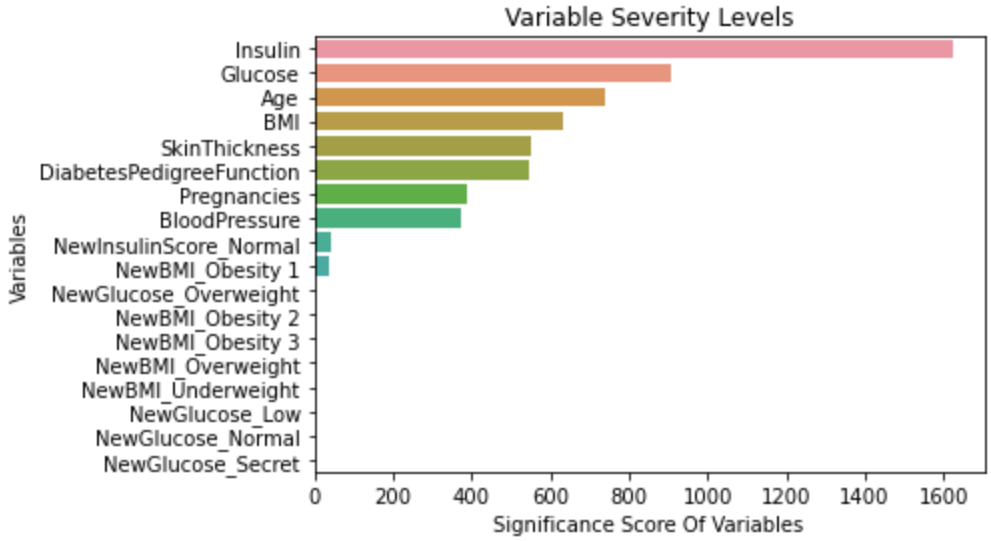
In [80]:

```
feature_imp = pd.Series(xgb_tuned.feature_importances_,
                        index=X.columns).sort_values(ascending=False)

sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Significance Score Of Variables')
plt.ylabel('Variables')
plt.title("Variable Severity Levels")
plt.show()
```




```
plt.title("Variable Severity Levels")
plt.show()
```



3) XGBOOST TUNING

In [74]:

```
xgb = GradientBoostingClassifier(random_state = 12345)


xgb_params = {
    "learning_rate": [0.01, 0.1, 0.2, 1],
    "min_samples_split": np.linspace(0.1, 0.5, 10),
    "max_depth":[3,5,8],
    "subsample":[0.5, 0.9, 1.0],
    "n_estimators": [100,1000]}
```

In [76]:

```
xgb_cv_model  = GridSearchCV(xgb,xgb_params, cv = 10, n_jobs = -1, ver
bose = 2).fit(X, y)
```

```
Fitting 10 folds for each of 720 candidates, totalling 7200 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   42.3s
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done 1005 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done 1450 tasks      | elapsed:  8.4min
```

```
[Parallel(n_jobs=-1)]: Done 1977 tasks      | elapsed: 11.6min
[Parallel(n_jobs=-1)]: Done 2584 tasks      | elapsed: 14.8min
[Parallel(n_jobs=-1)]: Done 3273 tasks      | elapsed: 19.3min
[Parallel(n_jobs=-1)]: Done 4042 tasks      | elapsed: 23.6min
[Parallel(n_jobs=-1)]: Done 4893 tasks      | elapsed: 28.7min
[Parallel(n_jobs=-1)]: Done 5824 tasks      | elapsed: 34.5min
[Parallel(n_jobs=-1)]: Done 6837 tasks      | elapsed: 40.9min
[Parallel(n_jobs=-1)]: Done 7200 out of 7200 | elapsed: 43.2min finished
```

In [77]:

```
xgb_cv_model.best_params_
```

Out[77]:

```
{'learning_rate': 0.1,
 'max_depth': 5,
 'min_samples_split': 0.1,
 'n_estimators': 100,
 'subsample': 1.0}
```

3.1) Final Model Installation

In [78]:

```
xgb_tuned = GradientBoostingClassifier(**xgb_cv_model.best_params_).
fit(X,y)
```

In [79]:

```
cross_val_score(xgb_tuned, X, y, cv = 10).mean()
```

Out[79]:

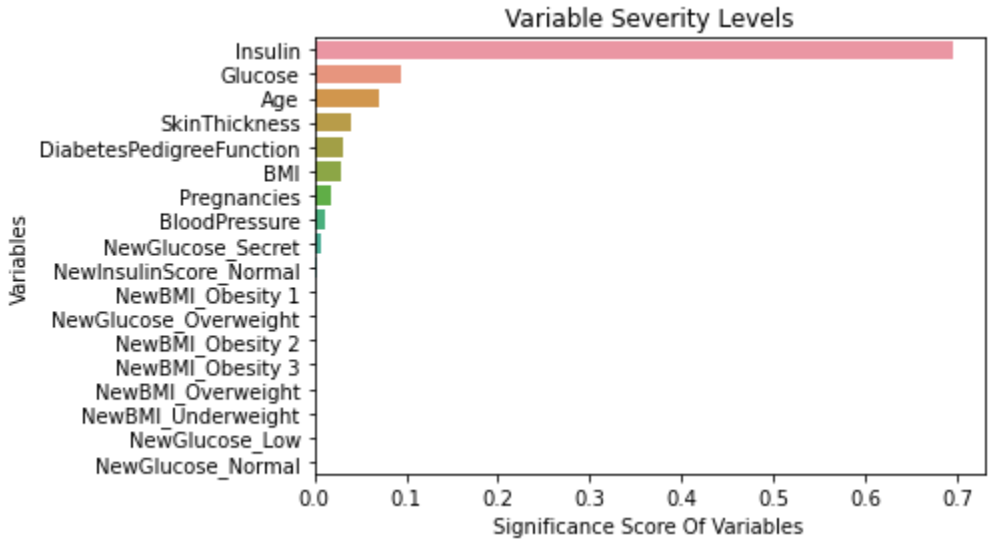
```
0.9013157894736843
```

In [80]:

```
feature_imp = pd.Series(xgb_tuned.feature_importances_,
                        index=X.columns).sort_values(ascending=False)

sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Significance Score Of Variables')
plt.ylabel('Variables')
```

```
plt.title("Variable Severity Levels")
plt.show()
```



COMPARISON OF FINAL MODELS

In [81]:

```
models = []

models.append(('RF', RandomForestClassifier(random_state = 12345, max
_depth = 8, max_features = 7, min_samples_split = 2, n_estimators =
 500)))
models.append(('XGB', GradientBoostingClassifier(random_state = 12345,
learning_rate = 0.1, max_depth = 5, min_samples_split = 0.1, n_estimators
= 100, subsample = 1.0)))
models.append(("LightGBM", LGBMClassifier(random_state = 12345, learn
ing_rate = 0.01,  max_depth = 3, n_estimators = 1000)))

# evaluate each model in turn
results = []
names = []
```

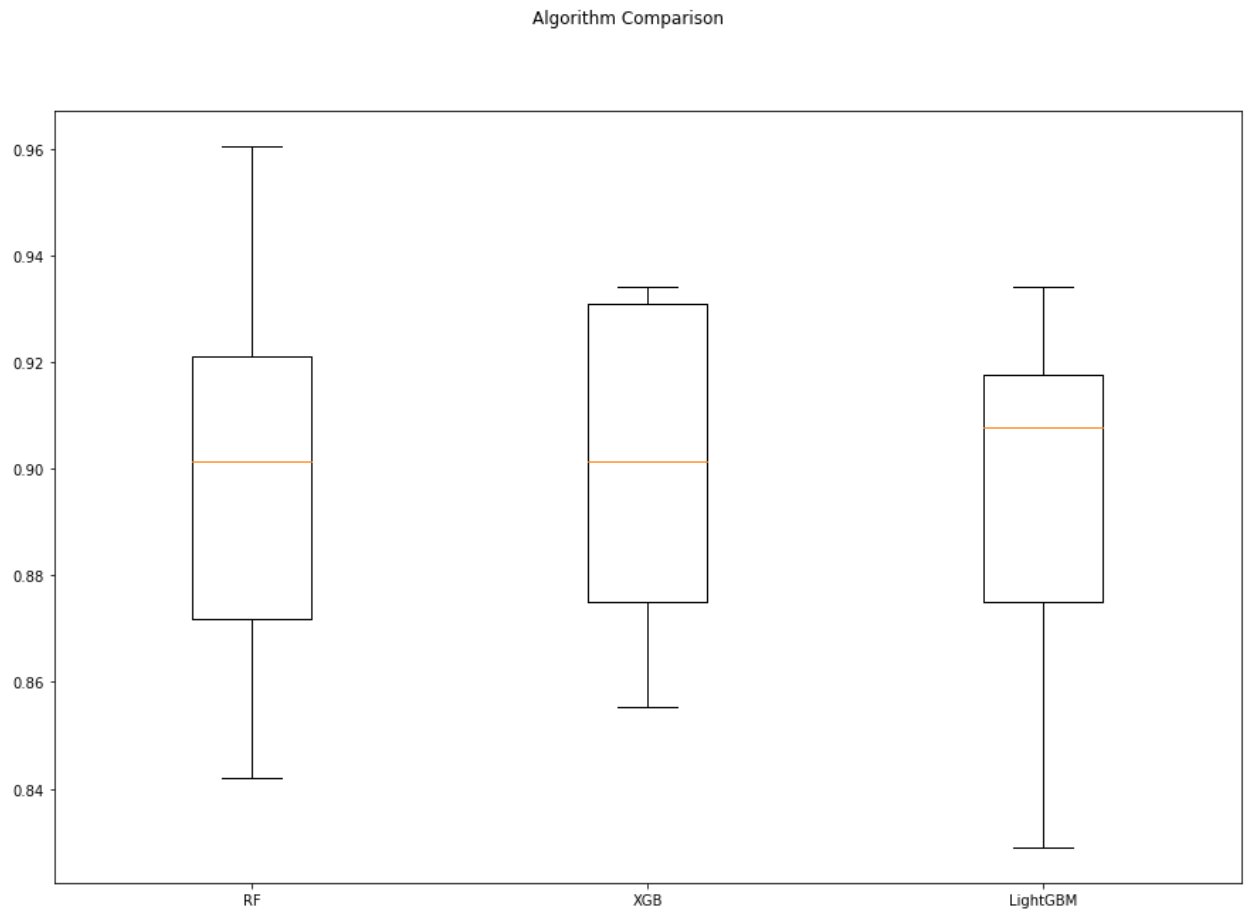
In [82]:

```
for name, model in models:

        kfold = KFold(n_splits = 10, random_state = 12345)
        cv_results = cross_val_score(model, X, y, cv = 10, scoring=
"accuracy")
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)

# boxplot algorithm comparison
fig = plt.figure(figsize=(15,10))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

RF: 0.897368 (0.034211)
XGB: 0.901316 (0.028373)
LightGBM: 0.896053 (0.033000)



REPORTING :

The aim of this study was to create classification models for the diabetes data set and to predict whether a person is sick by establishing models and to obtain maximum validation scores in the established models. The work done is as follows:

1) Diabetes Data Set read.

2) With Exploratory Data Analysis: The data set's structural data were checked. The types of variables in the dataset were examined. Size information of the dataset was accessed. The 0 values in the data

set are missing values. Primarily these 0 values were replaced with NaN values. Descriptive statistics of the data set were examined.

3) Data Preprocessing section; df for: The NaN values missing observations were filled with the median values of whether each variable was sick or not. The outliers were determined by LOF and dropped. The X variables were standardized with the rubost method..

4) During Model Building: Logistic Regression, KNN, SVM, CART, Random Forests, XGBoost, LightGBM like using machine learning models Cross Validation Score were calculated. Later Random Forests, XGBoost, LightGBM hyperparameter optimizations optimized to increase Cross Validation value.

5) Result: The model created as a result of XGBoost hyperparameter optimization became the model with the lowest Cross Validation Score value. (0.90)

ADVANTAGES :

1.Early Detection:

AI can identify individuals at risk of developing diabetes even before clinical symptoms appear, allowing for early intervention and proactive management.

2.Personalized Risk Assessment:

These systems provide tailored risk assessments based on an individual's medical history, lifestyle, and genetic factors. This personalized approach helps healthcare providers offer more targeted guidance and recommendations.

3.Efficient Healthcare Resource Allocation:

AI can assist healthcare providers in efficiently allocating resources by focusing on individuals at higher risk, allowing for optimized appointment scheduling and patient management.

4.Cost Savings:

Early detection and prevention can lead to significant cost savings in healthcare. Preventing or delaying the onset of diabetes reduces the need for expensive long-term treatments and complications management.

5.Improved Clinical Decision-Making:

Healthcare providers can use AI predictions to make more informed clinical decisions, enabling more accurate diagnoses and treatment plans.

6.Patient Engagement:

AI-based predictions can engage patients by providing them with insights into their health risks. This motivation can lead to better adherence to recommended treatment plans and lifestyle changes.

7.Population Health Management:

AI can assess the diabetes risk of an entire population, facilitating the development of targeted public health initiatives and interventions.

8.Telehealth Integration:

AI predictions can be integrated into telehealth platforms, allowing for remote monitoring and virtual consultations, particularly valuable for individuals in remote areas or with limited access to healthcare facilities.

9.Reduction in Complications:

By identifying individuals at risk early and providing proper management, AI-based systems can help reduce the risk of complications associated with diabetes, such as cardiovascular diseases, neuropathy, and retinopathy.

10.Data-Driven Research:

AI systems generate large datasets that can be used for diabetes research, including identifying trends and risk factors. This data can inform ongoing research and public health policies.

11.Continuous Improvement:

AI models can adapt and improve over time as more data becomes available, leading to more accurate and reliable predictions.

12.Empowering Patients:

Patients gain a better understanding of their health status and can take more active roles in their well-being, making informed choices regarding their health.

13.Global Relevance:

Diabetes is a worldwide health challenge, and AI-based prediction models can be adapted and applied globally, providing a valuable tool for addressing this public health issue.

14.Innovation and Technological Progress:

The development and deployment of AI-based systems for diabetes prediction drive technological innovation in healthcare, fostering the advancement of data science and AI in the medical field.

CONCLUSION :

- AI-based prediction systems for diabetes represent a transformative and highly promising approach to the prevention and management of this prevalent chronic disease. These systems leverage the power of artificial intelligence and machine learning to provide early detection, personalized risk assessment, and data-driven insights for both healthcare providers and individuals at risk of or living with diabetes.
- The advantages of AI-based prediction systems, including early detection, cost savings, efficient resource allocation, and improved clinical decision-making, underscore their potential to significantly enhance healthcare outcomes and reduce the burden of diabetes on individuals and healthcare systems. These systems empower patients to take control of their health and engage in informed decision-making, fostering a more proactive and patient-centric approach to healthcare.
- Furthermore, the global relevance of AI-based diabetes prediction models makes them an invaluable tool in addressing a worldwide public health challenge. Their adaptability and potential for continuous improvement through data-driven research and feedback loops position them as a cornerstone of innovation and technological progress in healthcare.

- As AI-based prediction systems for diabetes continue to evolve and become more accessible, they hold the promise of revolutionizing the way we approach diabetes prevention and management. By fostering early intervention, personalized care, and data-driven decision-making, these systems have the potential to usher in a new era of precision medicine, ultimately leading to improved health outcomes and a brighter future in the fight against diabetes.