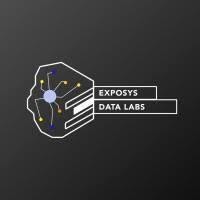
**Exposys Data Labs**

Bengaluru, Karnataka, 560064



Internship report on

# DIABETES PREDICTION

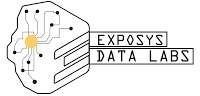
By

**SHREENIITHI D**

**KGiSL Institute of Technology**

Under the guidance of

**Exposys Data Labs**



# Abstract

Diabetes is a sort of chronic illness that affects people of all ages more frequently.Early disease prediction can assist a person in making the required preparations and lifestyle changes to either avoid the disease from occurring or to control the disease (for those who already have the disease).

Diabetes is a category of metabolic illnesses characterized by consistently raised blood sugar levels. Frequent urination, increased thirst, and increased hunger are all signs of elevated blood sugar. Diabetes can lead to a wide range of consequences if ignored. Hyperosmolar hyperglycemia, diabetic ketoacidosis, and even deaths are examples of acute complications. Cardiovascular illness, stroke, chronic kidney disease, foot ulcers, and eye impairment are examples of serious long-term consequences.The National Institute of Diabetes and Digestive and Kidney Diseases is the original source of this dataset. Based on specific diagnostic metrics present in the dataset, the dataset's goal is to diagnostically predict whether a patient has diabetes or not.

Data science has an emerging topic called machine learning that studies how machines learn from experience. The goal of this study is to create a system that, by combining the findings of several machine learning approaches, can more accurately perform early diabetes prediction for a patient. K nearest neighbour, Logistic Regression, Random Forest, Support Vector Machine, and Decision Tree are some of the techniques employed. Each algorithm's accuracy is calculated along with the model's accuracy. The model for predicting diabetes is then chosen from those with good accuracy.

# Table of Contents

**Chapter Name Page no**

[Abstract](#_TOC_250018) **[2](#_TOC_250018)**

1. [Introduction 4-5](#_TOC_250017)
   1. [Data Science 4](#_TOC_250016)
   2. [Machine Learning 5](#_TOC_250015)
2. [Existing Methods 6](#_TOC_250014)-7
   1. Issues in Existing Systems 7
3. [Proposed method 8-9](#_TOC_250013)
   1. XG Boost Hyperparameter optimization 8

3.1.1 Hyperparameter Optimization Techniques for XGBoost 8

* 1. Algorithm 9

1. Methodology 10-13
   1. [Diabetes Data Set reading 10](#_TOC_250007)
   2. [Exploratory Data Analysis 11](#_TOC_250006)
   3. [Data Preprocessing section 11](#_TOC_250005)
   4. During Model Building 13
   5. Result 13
2. Implementation 14-22
   1. Source Code 14-22
3. Conclusion 23
4. References 24

# 1. Introduction

## Data Science

In the general field of data science, significant knowledge and insights are drawn from vast multifaceted datasets. The process of gathering, organizing, analyzing, and interpreting data to find patterns, trends, and correlations is at the heart of data science. The solutions to issues, increased comprehension of phenomena, or the ability to foresee and forecast can all be done with these insights.The data science workflow typically consists of the following basic steps:

1. **Problem Definition:** The problem or question that has to be addressed must be clearly defined and expressed in such a way that it can be addressed using data.
2. **Data Acquisition:** It is the process of gathering pertinent data from a variety of sources, including files, databases, and APIs. To ensure that the data is in an appropriate format for analysis, this may require cleaning and preparing the data.
3. **Exploratory Data Analysis (EDA):** It is the process of examining and visualizing data to better understand its features, spot trends and outliers, and uncover connections between variables. EDA aids in developing hypotheses and revealing insights.
4. **Feature engineering** **:** It is the process of choosing or producing the most relevant features (input variables) out of the data that is available. To increase the predictive value of the variables, this step usually involves modifying or merging them.
5. **Model development :** It is the process of constructing predictive models or spotting patterns in data using statistical and machine learning methods. Depending on the nature of the issue, this may require the use of techniques like regression, classification, clustering, or deep learning.
6. **Model evaluation:** Using the right evaluation criteria, determining the effectiveness and correctness of the produced models. This aids in deciding on the optimal model and pinpointing potential improvement areas.
7. **Deployment and Communication:**Implementing the models into operational systems or leveraging the conclusions from the research to guide decision-making are examples of deployment and communication. Clear and actionable communication of the findings and results to the stakeholders is essential.

## Machine Learning

## 

Machine learning algorithms can analyze vast amounts of data to identify patterns and make predictions about future outcomes. In the context of diabetes, this means that machine learning models can be trained on large datasets of patient information to predict who is at risk of developing the disease and what interventions are most likely to be effective for individual patients.

One example of this is the development of a machine learning model that can predict the onset of type 2 diabetes up to five years in advance. By analyzing data from electronic health records, including factors such as age, weight, blood pressure, and cholesterol levels, the model was able to accurately identify patients at high risk of developing diabetes and recommend lifestyle interventions and medication to prevent or delay the onset of the disease.

This has the potential to significantly improve patient outcomes and reduce healthcare costs associated with diabetes.One of the biggest challenges in data collection and analysis for machine learning is ensuring that the data is accurate and reliable. There are many potential sources of error, including incomplete or incorrect data, biased samples, and measurement errors. It is important to carefully validate and clean the data before using it to train machine learning models. Despite these challenges, the potential benefits of machine learning for diabetes prediction are significant. By accurately identifying individuals who are at high risk of developing diabetes, healthcare providers can intervene early with preventative measures such as lifestyle changes and medication. This can improve patient outcomes and reduce the overall burden of diabetes on society.Recent studies have demonstrated the effectiveness of machine learning algorithms in predicting the onset of diabetes, allowing for early intervention and improved patient outcomes. One promising area of research involves the use of wearable devices and mobile apps to collect real-time data on blood sugar levels, physical activity, and other relevant metrics. By analyzing this data using machine learning techniques, researchers can develop personalized risk profiles and treatment plans for individuals at high risk of developing diabetes.

# 2. Existing Methods

There are various methods and approaches that have been used for diabetes prediction. Here are some of the existing methods commonly employed in diabetes prediction:

**Logistic Regression:** Logistic regression is a simple yet effective method for binary classification problems like diabetes prediction. It models the probability of a binary outcome using a linear combination of predictor variables.

**Support Vector Machines (SVM):** SVM is a powerful classification algorithm that finds the optimal hyperplane to separate classes. It can handle both linear and non-linear relationships in data.

**Random Forest:** Random Forest is an ensemble learning technique that combines multiple decision trees to improve predictive accuracy and control overfitting. It's capable of handling complex interactions in data.

**Gradient Boosting:** Gradient Boosting algorithms like XGBoost, LightGBM, and CatBoost are popular for their ability to create strong predictive models by combining multiple weak learners (trees) in an iterative manner.

**Neural Networks:** Deep learning techniques like artificial neural networks can capture intricate patterns in data and automatically learn relevant features for prediction tasks. They can be effective for diabetes prediction when dealing with large and complex datasets.

**K-Nearest Neighbors (KNN):** KNN is a simple instance-based learning algorithm that makes predictions based on the class of k-nearest neighbors to a given data point.

**Naive Bayes:** Naive Bayes is a probabilistic algorithm that's based on the Bayes' theorem and the assumption of independence between features. It's commonly used for text classification but can also be adapted for diabetes prediction.

**Ensemble Methods:** Techniques like stacking, where multiple models are combined to make predictions, can further improve prediction accuracy.

**Feature Selection and Engineering:** Choosing relevant features and creating new features based on domain knowledge can significantly enhance model performance.

**Hyperparameter Tuning:** Optimizing the hyperparameters of a model, such as learning rates, regularization strengths, and number of trees, can lead to improved performance.

It's important to note that the effectiveness of these methods can vary depending on the dataset, the amount of available data, feature engineering, and domain-specific knowledge. Often, a combination of these methods, along with careful preprocessing and feature engineering, can yield the best results in diabetes prediction

## 2.1 Issues in Existing System

Existing diabetes prediction systems can have several issues and challenges that need to be addressed to ensure accurate and reliable predictions. Some common issues include:

**1. Imbalanced Data:** Diabetes datasets often have an imbalance between the number of positive and negative cases, which can lead to biased models. Techniques like oversampling, undersampling, or using specialized algorithms designed for imbalanced data are necessary.

**2. Data Quality and Missing Values:** Incomplete or erroneous data can affect the performance of prediction models. Proper data preprocessing, imputing missing values, and handling outliers are crucial.

**3. Feature Selection and Engineering:** Choosing relevant features and creating new ones can be challenging. Improper feature selection can lead to poor model performance or overfitting.

**4. Model Overfitting:** Complex models can overfit the training data, resulting in poor generalization to new data. Regularization techniques and careful model selection can mitigate this issue.

**5. Hyperparameter Tuning:** Finding the optimal hyperparameters for a model can be time-consuming and resource-intensive. Automated hyperparameter tuning methods like grid search, random search, or Bayesian optimization are essential.

# 3.Proposed Method

## **XG Boost Hyperparameter optimization**

XGBoost (Extreme Gradient Boosting) is a popular machine learning algorithm known for its high performance in various tasks. Optimizing its hyperparameters is crucial for achieving the best possible model performance. While I can't provide you with a diagram, I can guide you through the process of hyperparameter optimization for XGBoost using a combination of techniques.

3.1.1 **Hyperparameter Optimization Techniques for XGBoost:**

**Grid Search:**Grid search involves specifying a range of values for each hyperparameter you want to tune. The algorithm then exhaustively searches through all possible combinations of hyperparameters to find the best set.

**Random Search:**Random search randomly samples hyperparameters from predefined ranges. This approach is more efficient than grid search because it doesn't try every possible combination.

**Bayesian Optimization:**Bayesian optimization uses a probabilistic model to predict which hyperparameters are most likely to improve the model's performance. It iteratively updates the model based on the results of previous evaluations.

**Gradient-Based Optimization:**Some libraries, like Optuna, use gradient-based optimization methods to efficiently search for the best hyperparameters.

**Genetic Algorithms:**Genetic algorithms use concepts from biological evolution to search for optimal hyperparameters.

**Neural Architecture Search (NAS):**NAS techniques, like AutoML, use neural networks to search for optimal hyperparameters.

## 

## 

CHOOSE OPTIMIZATION TECHNIQUE

DEFINE SEARCH

SPACE

EVALUATE PERFORMANCE OF MODEL

PERFORM OPTIMIZATION ITERATIONS

SELECT BEST HYPERPARAMS

TRAIN AND EVALUATE FINAL MODEL

## Algorithm

## Step 1:Start by defining a search space for hyperparameters.

## **Step 2:**Then choose an optimization technique based on the available resources and desired optimization speed.

## **Step 3:**Perform multiple optimization iterations, evaluating the performance of different

## hyperparameter sets.

## **Step 4:**Finally, select the best hyperparameters and train the final XGBoost model using those settings.

# 4.Methodology

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 :** The diabetes dataset is read as a csv file.

**2) 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:** 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)

## **4.1 Diabetes Data Set reading:**

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

* **Pregnancies**: Number of times pregnant
* **Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
* **BloodPressure**: Diastolic blood pressure (mm Hg)
* **SkinThickness**: Triceps skin fold thickness (mm)
* **Insulin**: 2-Hour serum insulin (mu U/ml)
* **BMI**: Body mass index (weight in kg/(height in m)^2)
* **Diabetes Pedigree Function**: Diabetes pedigree function
* **Age**: Age (years)
* **Outcome**: Class variable (0 or 1)

**Number of Observation Units: 768**

**Variable Number: 9**

**4.2 Exploratory Data Analysis:**

****Load and Inspect Data:**Load your dataset into your chosen programming environment (Python, R, etc.).Display basic information such as the number of rows and columns, data types, and the first few rows of data.**

****Summary Statistics**:Calculate summary statistics for numerical variables (mean, median, standard deviation, etc.).Observe the range and distribution of values in each feature.**

****Missing Data:**Identify missing values in your dataset.Determine the percentage of missing values for each feature.Decide how to handle missing values: impute them, drop rows or columns, etc.**

****Data Visualization:**Create various types of visualizations to understand the distribution and relationships within the data:**

* ****Histograms:** Visualize the frequency distribution of numerical variables.**
* ****Box plots:** Display the distribution of data and identify outliers.**
* ****Scatter plots:** Explore relationships between pairs of numerical variables.**
* ****Bar plots:** Visualize the distribution of categorical variables.**
* ****Heatmaps:** Display correlations between numerical variables.**
* ****Pair plots:** Create scatter plots for multiple pairs of numerical variables.**

****Correlation Analysis:**Calculate the correlation matrix of numerical variables.Visualize correlations using a heatmap to identify strong relationships.**

****Categorical Variables:**Analyze the distribution of categorical variables.Create bar plots or pie charts to visualize the frequency of different categories.**

**4.3 Data Preprocessing section**

**Data Cleaning**

**Handle missing values:** Decide whether to remove rows or impute missing values based on the percentage of missing data and the nature of the problem.

**Remove duplicate rows:** Check for and remove duplicated data points to avoid biasing your model.

**Handle outliers**: Identify and handle outliers that might negatively affect model performance.

**Feature Scaling**: Scale numerical features to the same range (e.g., using Min-Max scaling or Standardization) to prevent some features from dominating others during modeling.

**One-Hot Encoding**: Convert categorical variables into binary vectors to make them suitable for most machine learning algorithms.

**Label Encoding:** Convert categorical variables into numeric labels when the order of categories matters.

**Feature Engineering**: Create new features that might be more informative or relevant for the problem. This can involve mathematical operations, combining existing features, or extracting relevant information.

**Normalization and Standardization:**Normalize numerical features to a common scale (usually between 0 and 1) to prevent features with larger values from dominating the model.

Standardize features to have zero mean and unit variance, which can improve convergence and performance for some algorithms.

**Feature Selection:**Select the most relevant features to include in your model to reduce dimensionality and potentially improve model performance.Use techniques like feature importance scores, correlation analysis, and domain knowledge to guide feature selection.

**Data Splitting:**Split your preprocessed data into training, validation, and test sets to evaluate your model's performance on unseen data. Common splits are 70-15-15 or 80-20.

**Feature Scaling on Validation and Test Sets:**

When scaling your features, use the scaling parameters (mean and standard deviation) computed from the training set to also transform the validation and test sets. This ensures consistency between different datasets.

**4.4 During Model Building:**

**Cross-Validation Score Calculation:**

For each model (Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Classification and Regression Trees (CART), Random Forests, XGBoost, LightGBM), a common practice is to use k-fold cross-validation.In k-fold cross-validation, the dataset is divided into 'k' equally sized folds.For each fold, the model is trained on 'k-1' folds and tested on the remaining fold.This process is repeated 'k' times, each time using a different fold as the test set, and the average performance metric (e.g., accuracy, F1-score) across all 'k' folds is calculated. This is the cross-validation score.

**Hyperparameter Optimization:**After calculating the initial cross-validation score, the next step is to optimize the model's hyperparameters. Hyperparameters are settings that are not learned during training but affect how the model is trained and the complexity of the model.

For Random Forests, XGBoost, and LightGBM, these hyperparameters include the number of trees, maximum depth of trees, learning rate, regularization parameters, and more.

Hyperparameter optimization aims to find the best combination of hyperparameters that maximizes the cross-validation score.Techniques like grid search, random search, and more advanced methods like Bayesian optimization can be used for hyperparameter tuning.

**Impact on Cross-Validation Score:**Hyperparameter optimization aims to fine-tune the model to improve its generalization performance.By finding optimal hyperparameters, you're adjusting the model's behavior to better fit the data, which can lead to better cross-validation scores.

For example, finding the right tree depth, learning rate, or number of estimators can significantly impact the model's bias-variance trade-off, resulting in better performance on unseen data.

**4.5 Result:**

**Evaluation and Model Selection:**

After hyperparameter optimization, you can compare the cross-validation scores of different models (Random Forests, XGBoost, LightGBM) and select the model with the highest cross-validation score as your final model.This final model should generalize well to new data and can be further evaluated on a separate test set to get an estimate of its performance on unseen data.

# Implementation

## 5.1 Source Code

#Installing necessary Libraries

import numpy as np

import pandas as pd

import statsmodels.api as sm

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import scale, StandardScaler

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score

from sklearn.metrics import confusion\_matrix, accuracy\_score, mean\_squared\_error, r2\_score, roc\_auc\_score, roc\_curve, classification\_report

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

from lightgbm import LGBMClassifier

from sklearn.model\_selection import KFold

import warnings

warnings.simplefilter(action = "ignore")

#../input/pima-indians-diabetes-database/diabetes.csv

df = pd.read\_csv("diabetes.csv")

df.head()

df.info()

df.describe()

df.shape

# The histagram of the Age variable was reached.

df["Age"].hist(edgecolor = "black");

print("Max Age: " + str(df["Age"].max()) + " Min Age: " + str(df["Age"].min()))

# Histogram and density graphs of all variables were accessed.

fig, ax = plt.subplots(4,2, figsize=(16,16))

sns.distplot(df.Age, bins = 20, ax=ax[0,0])

sns.distplot(df.Pregnancies, bins = 20, ax=ax[0,1])

sns.distplot(df.Glucose, bins = 20, ax=ax[1,0])

sns.distplot(df.BloodPressure, bins = 20, ax=ax[1,1])

sns.distplot(df.SkinThickness, bins = 20, ax=ax[2,0])

sns.distplot(df.Insulin, bins = 20, ax=ax[2,1])

sns.distplot(df.DiabetesPedigreeFunction, bins = 20, ax=ax[3,0])

sns.distplot(df.BMI, bins = 20, ax=ax[3,1])

df.corr()

f, ax = plt.subplots(figsize= [10,15])

sns.heatmap(df.corr(), annot=True, fmt=".2f", ax=ax, cmap = "magma" )

ax.set\_title("Correlation Matrix", fontsize=10)

plt.show()

df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

df.head()

df.isnull().sum()

import missingno as msno

msno.bar(df);

def median\_target(var):

temp = df[df[var].notnull()]

temp = temp[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset\_index()

return temp

columns = df.columns

columns = columns.drop("Outcome")

for i in columns:

median\_target(i)

df.loc[(df['Outcome'] == 0 ) & (df[i].isnull()), i] = median\_target(i)[i][0]

df.loc[(df['Outcome'] == 1 ) & (df[i].isnull()), i] = median\_target(i)[i][1]

df.head()

import seaborn as sns

sns.boxplot(x = df["Insulin"]);

Q1 = df.Insulin.quantile(0.25)

Q3 = df.Insulin.quantile(0.75)

IQR = Q3-Q1

lower = Q1 - 1.5\*IQR

upper = Q3 + 1.5\*IQR

df.loc[df["Insulin"] > upper,"Insulin"] = upper

import seaborn as sns

sns.boxplot(x = df["Insulin"]);

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()

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()

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 = pd.get\_dummies(df, columns =["NewBMI","NewInsulinScore", "NewGlucose"], drop\_first = True)

categorical\_df = df[['NewBMI\_Obesity 1','NewBMI\_Obesity 2', 'NewBMI\_Obesity 3', 'NewBMI\_Overweight','NewBMI\_Underweight',

'NewInsulinScore\_Normal','NewGlucose\_Low','NewGlucose\_Normal', 'NewGlucose\_Overweight', 'NewGlucose\_Secret']]

categorical\_df.head()

y = df["Outcome"]

X = df.drop(["Outcome",'NewBMI\_Obesity 1','NewBMI\_Obesity 2', 'NewBMI\_Obesity 3', 'NewBMI\_Overweight','NewBMI\_Underweight',

'NewInsulinScore\_Normal','NewGlucose\_Low','NewGlucose\_Normal', 'NewGlucose\_Overweight', 'NewGlucose\_Secret'], axis = 1)

cols = X.columns

index = X.index

from sklearn.preprocessing import RobustScaler

transformer = RobustScaler().fit(X)

X = transformer.transform(X)

X = pd.DataFrame(X, columns = cols, index = index)

X = pd.concat([X,categorical\_df], axis = 1)

X.head()

y.head()

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 = []

from sklearn.model\_selection import KFold, cross\_val\_score

import matplotlib.pyplot as plt

results = []

names = []

# Define the models

for index, (name, model) in enumerate(models):

kfold = KFold(n\_splits=10)

cv\_results = cross\_val\_score(model, X, y, cv=kfold, scoring="accuracy")

results.append(cv\_results)

names.append(name)

msg = 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, labels=names)

plt.show()

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)

gs\_cv.best\_params\_

rf\_tuned = RandomForestClassifier(\*\*gs\_cv.best\_params\_)

rf\_tuned = rf\_tuned.fit(X,y)

cross\_val\_score(rf\_tuned, X, y, cv = 10).mean()

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()

lgbm = LGBMClassifier(random\_state = 12345)

lgbm\_params = {"learning\_rate": [0.01, 0.03, 0.05, 0.1, 0.5],

"n\_estimators": [500, 1000, 1500],

"max\_depth":[3,5,8]}

gs\_cv = GridSearchCV(lgbm,

lgbm\_params,

cv = 10,

n\_jobs = -1,

verbose = 2).fit(X, y)

gs\_cv.best\_params\_

lgbm\_tuned = LGBMClassifier(\*\*gs\_cv.best\_params\_).fit(X,y)

cross\_val\_score(lgbm\_tuned, X, y, cv = 10).mean()

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')

plt.title("Variable Severity Levels")

plt.show()

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]}

xgb\_cv\_model = GridSearchCV(xgb,xgb\_params, cv = 10, n\_jobs = -1, verbose = 2).fit(X, y)

xgb\_cv\_model.best\_params\_

xgb\_tuned = GradientBoostingClassifier(\*\*xgb\_cv\_model.best\_params\_).fit(X,y)

cross\_val\_score(xgb\_tuned, X, y, cv = 10).mean()

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()

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, learning\_rate = 0.01, max\_depth = 3, n\_estimators = 1000)))

# evaluate each model in turn

results = []

names = []

from sklearn.model\_selection import KFold, cross\_val\_score

import matplotlib.pyplot as plt

results = []

names = []

# Define the models

for name, model in models:

kfold = KFold(n\_splits=10, shuffle=True)

cv\_results = cross\_val\_score(model, X, y, cv=kfold, 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()

# Conclusion

In conclusion, we have learned about the potential of data science and machine learning in predicting diabetes. By utilizing techniques such as the XGBoosting algorithm and data preprocessing/feature selection, we can improve our accuracy in predicting this disease. It is important to continue researching in this field to further enhance our understanding and predictive abilities.Machine learning has the potential to revolutionize diabetes prediction and improve patient outcomes. By analyzing large amounts of data from various sources, machine learning algorithms can identify patterns and predict the likelihood of an individual developing diabetes with high accuracy.

Through the use of data science and machine learning, we have the opportunity to make a significant impact on the lives of those affected by diabetes. By accurately predicting the disease, we can take preventative measures and provide better treatment options.**The model created as a result of XGBoost hyperparameter optimization became the model with the lowest Cross Validation Score value. (0.90)**

# References

1. <https://blog.cambridgespark.com/hyperparameter-tuning-in-xgboost-4ff9100a3b2f>
2. <https://www.kaggle.com/code/prashant111/a-guide-on-xgboost-hyperparameters-tuning>
3. <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>
4. <https://www.youtube.com/watch?v=ZsM2z0pTbnk>
5. <https://www.javatpoint.com/diabetes-prediction-using-machine-learning>
6. <https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7>
7. <https://www.w3schools.com/python/python_ml_scatterplot.asp>
8. <https://www.youtube.com/watch?v=TbundVfb8ow&t=3s>
9. <https://machinelearningmastery.com/seaborn-data-visualization-for-machine-learning/>
10. https://www.kaggle.com/code/residentmario/using-missingno-to-diagnose-data-sparsity