****

**Machine Learning Assignment**

**Diabetes Analysis**

**Submitted By:**

|  |  |
| --- | --- |
| **Student Name** | **Student IT NO** |
| Jayasena A.K.B.D | IT21073946 |
| Achchige R.A.N.R | IT21102264 |
| Abegunasekara P.W.A.B.N | IT21104176 |
| Padmasiri P.G.S.M | IT21078996 |

**Contents**

[**1.Introduction** 3](#_Toc164457757)

[1.1 Problem Statement 3](#_Toc164457758)

[**What is Diabetes?** 3](#_Toc164457759)

[**Symptoms** 3](#_Toc164457760)

[**Treatment and Management of Diabetes** 3](#_Toc164457761)

[**Prevention** 4](#_Toc164457762)

[**2.Methodology** 6](#_Toc164457763)

[2.1 Data Collection 6](#_Toc164457764)

[2.2 Description of Dataset 6](#_Toc164457765)

[2.3 Algorithm Selection 6](#_Toc164457766)

[**Data Cleaning and Preprocessing** 6](#_Toc164457767)

[**Exploratory data analysis** 7](#_Toc164457768)

[**Feature Selection** 7](#_Toc164457769)

[**Model construction and comparison** 8](#_Toc164457770)

[**3.Implementation** 9](#_Toc164457771)

[**4.Results and Discussions** 18](#_Toc164457772)

[K-Nearest Neighbors, Support Vector Machines, and Random Forest are the algorithms used in this code. 22](#_Toc164457773)

[The effectiveness of each method was evaluated using the subsequent metrics and 5-fold cross-validation: 22](#_Toc164457774)

[ Accuracy 22](#_Toc164457775)

[ F1 scores 22](#_Toc164457776)

[ ROC AUC results 22](#_Toc164457777)

[Discussion 22](#_Toc164457778)

[Limitations 23](#_Toc164457779)

[Future work 23](#_Toc164457780)

[**5.References** 24](#_Toc164457781)

[**6.Apendix** 24](#_Toc164457782)

# **1.Introduction**

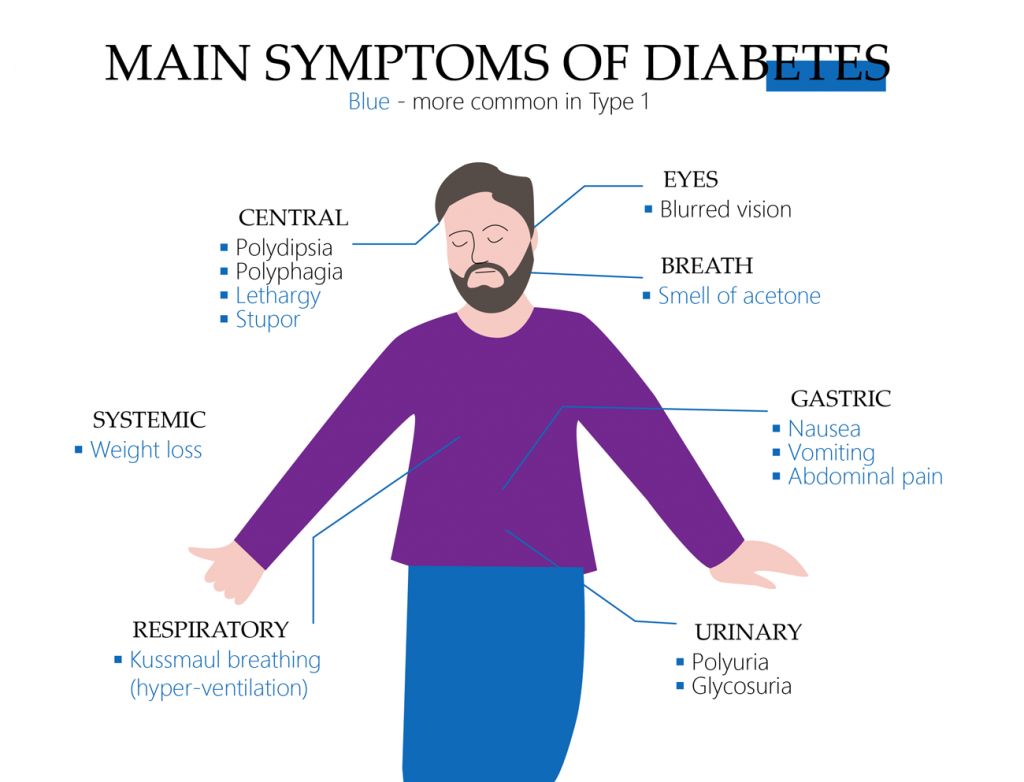
## 1.1 Problem Statement

### **What is Diabetes?**

Diabetes is a long-term medical disorder marked by high blood glucose (sugar) levels. Over time, diabetes can cause major harm to the heart, blood vessels, kidneys, eyes, nerves, and heart. Type 1, Type 2, and gestational diabetes are the three most prevalent forms of the disease.

* **Type 1** Diabetes is an autoimmune disease in which the pancreatic beta cells that produce insulin are attacked by the body, resulting in little or no insulin being produced. It is more often diagnosed in children and young adults and calls for daily insulin treatment.
* The main cause of **type 2** diabetes, which is more common, is the body's resistance to insulin or the pancreas' inability to make enough of it. It is more common in adults and has a strong correlation with obesity, while it is becoming more prevalent in younger age groups.
* Although **gestational diabetes** normally goes away after giving birth and develops throughout pregnancy, it can greatly raise the risk of type 2 diabetes in the future for both the mother and the child.

### **Symptoms**



The severity and type of the disease, as well as the length of time that blood sugar levels are elevated, all influence the symptoms of diabetes. Common symptoms include:

* Higher blood sugar levels drive the body to produce more pee, which increases the demand for fluid consumption. These two factors combined result in increased **thirst and urination**.
* **Fatigue**: A persistently low level of energy is caused by insufficient sugar entering the body's cells from the bloodstream.
* **Vision impairments**: Excessive glucose levels can harm the retina's blood vessels, resulting in vision impairments and other visual abnormalities.
* **Unexpected weight loss**: When a person has type 1 diabetes, their pancreas stops making insulin, which forces their body to burn fat and muscle for energy. This leads to an unexpected drop in weight.
* **Increased hunger**: Because insulin cannot properly carry glucose into the cells, persons with diabetes may experience extreme hunger even when they consume more.
* Diabetes impairs the body's capacity to heal and recover from infections, resulting in **slow-healing wounds or recurrent infections**.
* **Hands and foot tingling**: Nerve damage brought on by an excess of sugar in the blood can result in tingling or numbness in the extremities.

### **Treatment and Management of Diabetes**

Managing diabetes requires a comprehensive approach that includes lifestyle management, regular monitoring, medication, and sometimes insulin therapy. Key strategies include:

* **Modifications to lifestyle**: Exercise and diet are essential for controlling diabetes. It is often advised to have a diet high in fiber and low in calories, saturated fat, and carbs. Regular exercise enhances insulin sensitivity and aids in maintaining a healthy weight.
* **Blood sugar monitoring**: Those with diabetes can modify their diets, activities, and prescription regimens to keep their blood sugar levels within a normal range by regularly checking their levels.
* **Medication**: Individuals with type 2 diabetes frequently need to take medicine that helps to either produce more insulin, lessen the production of sugar, or enhance the effectiveness of insulin. Initially, metformin is usually prescribed if blood glucose levels persist, more drugs may be taken.
* **Insulin therapy**: Some individuals with type 2 diabetes and all individuals with type 1 diabetes require insulin therapy. Several types of insulin are available, including rapid-acting insulin, long-acting insulin, and intermediate options.
* **Education and support**: Individuals can better understand and manage their diabetes condition with the assistance of ongoing diabetes education, dietary counseling, and support groups.

### **Prevention**

Being the most preventable kind of diabetes, type 2 diabetes, preventive actions are especially crucial. Keeping a healthy weight through nutrition and exercise on a regular basis is the best defense against diabetes. Regular prediabetes screening can also result in early intervention and stop the progression to type 2 diabetes, especially if you have a family history or other risk factors.

# **2.Methodology**

## 2.1 Data Collection

Below is the link to the dataset,

<https://www.kaggle.com/datasets/mathchi/diabetes-data-set>

size – 768 data available in the data sheet

## 2.2 Description of Dataset

Under a number of limitations, these occurrences were selected from a larger database. Particularly, every patient at this clinic is an Indian woman from the Pima tribe, and she must be at least 21 years old.

1. Pregnancies: The total number of pregnancies the patient has had
2. Glucose: The plasma glucose concentration measured after a two-hour oral glucose tolerance test.
3. BloodPressure: The diastolic blood pressure of the patient measured in millimeters of mercury (mm Hg).
4. SkinThickness: The triceps skin fold thickness measured in millimeters. This measurement can indicate body fat percentage.
5. Insulin: The two-hour serum insulin level measured in micro-units per milliliter (mu U/ml).
6. BMI: Body Mass Index, which provides a heuristic measure of body weight based on the person's weight in relation to their height. It is expressed in kg/m².
7. Diabetes Pedigree Function: A score that represents how much the diabetes is attributed to genetics. This score can help assess the likelihood of diabetes based on family history.
8. Age: The age of the patient in years
9. Outcome: The class variable with 0 or 1 indicating negative or positive diabetes diagnosis respectively.

## 2.3 Algorithm Selection

### **Data Cleaning and Preprocessing**

We started our study by getting the dataset ready for modeling and exploration. Pre-processing and data cleaning are essential to guaranteeing that the machine learning models' predictions are accurate. The following tasks were part of the process:

* **Handling Outliers**: Outliers have the potential to distort the results for continuous variables like age, diabetes pedigree function, BMI, and insulin. To lessen the impact of severe outliers, we computed the interquartile range (IQR) for these variables and capped values at 1.5 times the IQR above the third quartile and below the first quartile.
* **Feature engineering**: In order to capture the interaction between BMI and Age, we multiplied the two variables to produce a new feature called BMI\_Age. This feature may improve the model's capacity to predict diabetes.
* **Standardization**: We applied standard scaling to numerical features to ensure they contribute equally to the analysis by bringing them onto a similar scale. For models like SVM that are sensitive to the magnitude of the input features, this is especially crucial.
* **Handling Duplicates and Missing Values**: In order to avoid including redundant data into our models, we eliminated duplicate entries. Furthermore, we verified that our dataset was complete for analysis and looked for any missing values.

### **Exploratory data analysis**

Exploratory Data Analysis was conducted to understand the distributions of various features and the relationships between them:

* **Distribution Plots**: We visualized the distribution of each numerical feature using boxplots to identify the spread and central tendencies, alongside identifying any further outliers.
* **Correlation Analysis**: Heatmaps displaying correlation coefficients between characteristics were created using correlation analysis to help visualize the connections and interdependencies between various variables.
* **Outcome Visualization**: To comprehend the class balance between diabetic and non-diabetic entries in the dataset, the distribution of the diabetes outcome variable was shown.

### **Feature Selection**

The process of feature selection entailed determining the significance and applicability of various elements within the dataset.

* **Correlation with Outcome**: Features that had a strong correlation with the outcome variable were given priority in terms of correlation. To find and keep the features that were most predictive of the outcome, we employed correlation metrics.
* **Feature Importance**: To help refine the model and concentrate on important predictors, we used the intrinsic feature importance for models such as Random Forest to determine which features have the most impact on predicting the target variable.

### **Model construction and comparison**

To determine which machine learning model was best for predicting diabetes, a number of them were created and assessed:

* **Model Selection**: We selected a range of models, including RandomForest, SVM, K-Neighbors Classifier, and Logistic Regression, that are appropriate for binary classification applications. Each model was selected based on its suitability to handle binary outcome prediction.
* **Hyperparameter Tuning**: To maximize the performance of models such as RandomForest and SVM, we used GridSearchCV to do hyperparameter tuning. For RandomForest, we explored with various parameter settings such as n\_estimators and max\_depth, for SVM, we experimented with C, kernel, and gamma.
* **Evaluation of the Models**: A number of metrics, such as accuracy, F1-score, and ROC-AUC score, were used to assess the models. This thorough assessment made it easier to compare models based on their accuracy as well as their capacity to strike a balance between precision and recall.
* **Features' Importance Visualization**: To shed light on the data and the behavior of the models, we used tree-based models to illustrate the significance of various features in generating predictions.
* **Justification why used.**

1. **Logistic Regression**

Introduction**:** Because logistic regression is so good at providing probabilities and is so simple to understand, it is frequently utilized in healthcare settings for binary classification issues like diabetes prediction.

Background: This model helps explain how variables affect the likelihood of developing diabetes by offering coefficients that provide the odds ratio for each predictor variable. It is particularly well-suited for binary outcomes. It is a mainstay in medical predictive analytics due to its resilience and simplicity.

1. **K-Nearest Neighbors (KNN)**

Introduction**:** K-Nearest Neighbors is a non-parametric method used for classification and regression. Based on similarities in case profiles within the training set, it aids in the classification of patients for diabetes prediction.

Background: KNN was selected due to its capacity to accurately represent the complicated nature of smaller datasets where similar cases can suggest similar outcomes. Since it doesn't make any assumptions about the data's form, it can be applied to real-world medical data that frequently deviates from theoretical distributions.

1. **Support Vector Machines (SVM)**

Introduction**:** SVM is favored in binary classification tasks due to its ability to model non-linear decision boundaries thanks to various kernel functions. The ability to handle large feature spaces and control overfitting through regularization are particularly valuable in complex diseases like diabetes.

.

Background: SVM is robust to outliers and effective in high dimensional spaces, which is typical in medical datasets where many variables might influence the outcome.

1. **Random Forest**

Introduction**:** Random Forest is an ensemble technique that enhances decision trees' simplicity by minimizing their propensity to overfit, which is essential for preserving the model's performance on fresh, untainted data.

Background: Random Forest is especially good at handling the complex nature of diabetes risk factors because of its capacity to handle huge datasets with numerous input features. It provides good feature importance estimations and accuracy, which are useful for analyzing the influence of different predictors.

# **3.Implementation**

**Importing the libraries.**

A computer screen shot of a program

Description automatically generated

**Read data set.**

**Describe data set.**

**A screenshot of a computer screen

Description automatically generated**

**Describe Columns.**

**A screenshot of a computer

Description automatically generated**

**Categorizing Columns.**



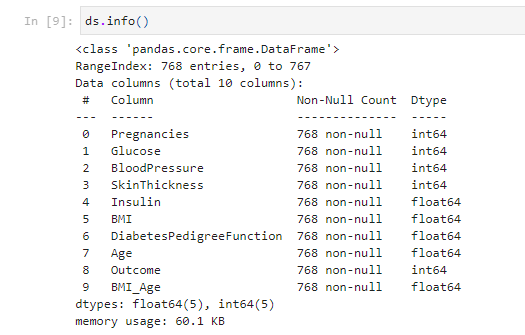
**Calculate Columns & Rows.**

A screenshot of a computer code

Description automatically generated

**Filling missing values in the dataset.**

**Before Processing.**



**After Processing.**

**A close-up of a computer code

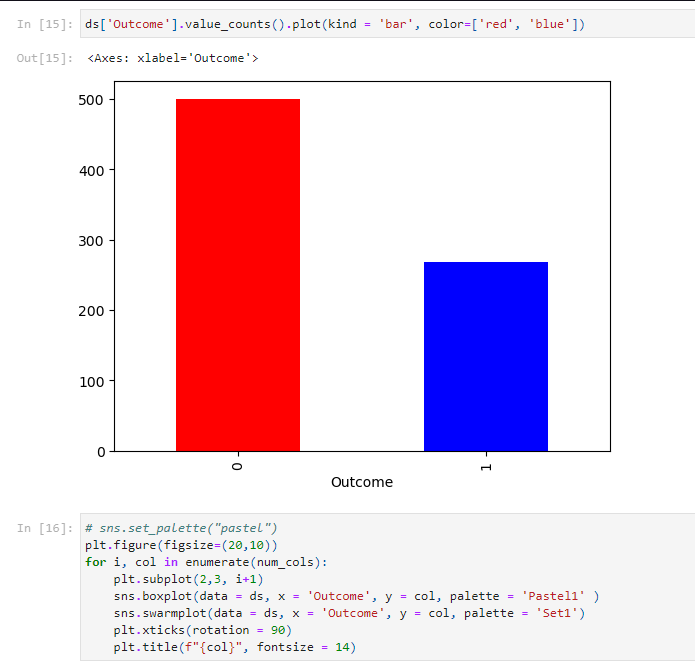
Description automatically generated**

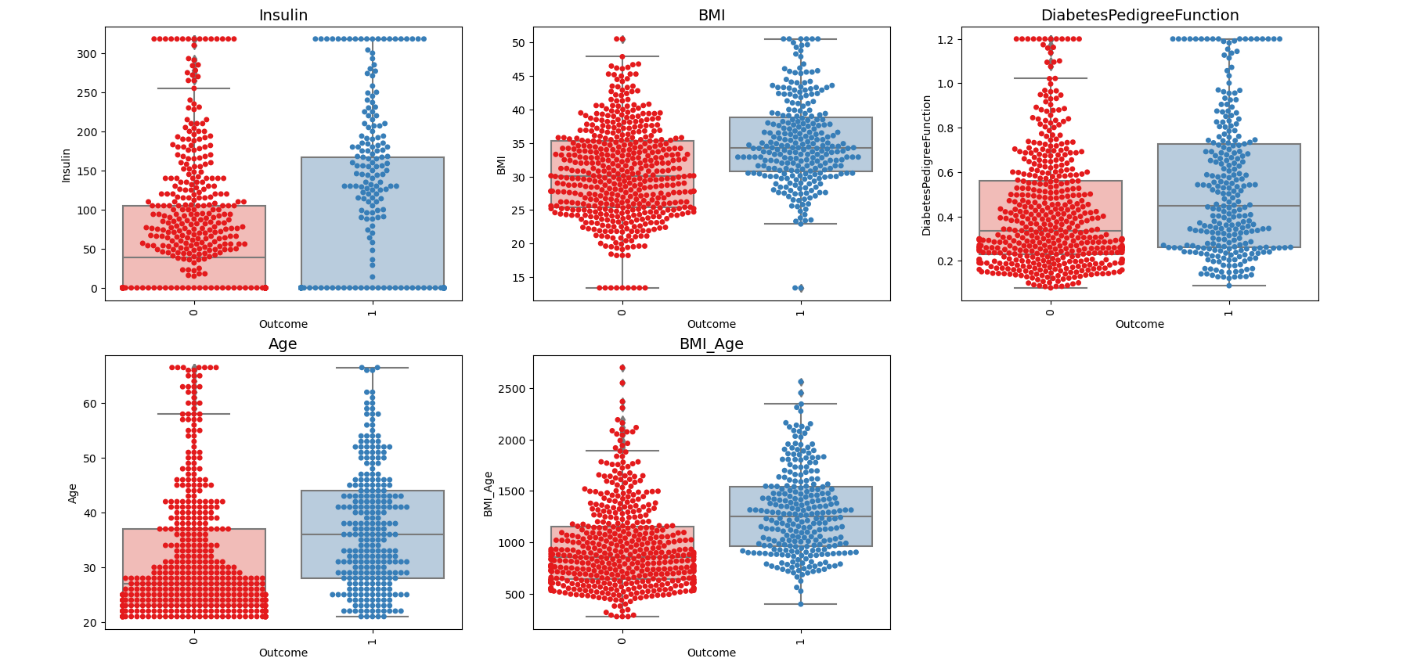
**Describe the data After Processing.**

A screenshot of a computer

Description automatically generated

**Basic Exploratory Data Analysis.**

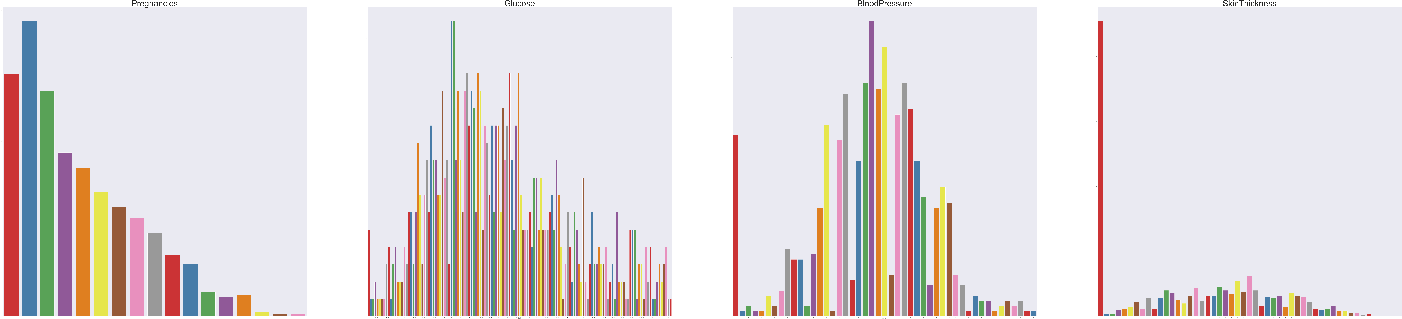




**Histograms.**

**Timeline

Description automatically generated**



Timeline

Description automatically generated

Chart, histogram

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application, scatter chart

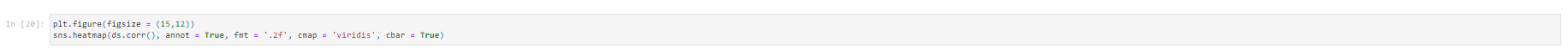
Description automatically generated

**Pair plots.**

A picture containing scatter chart

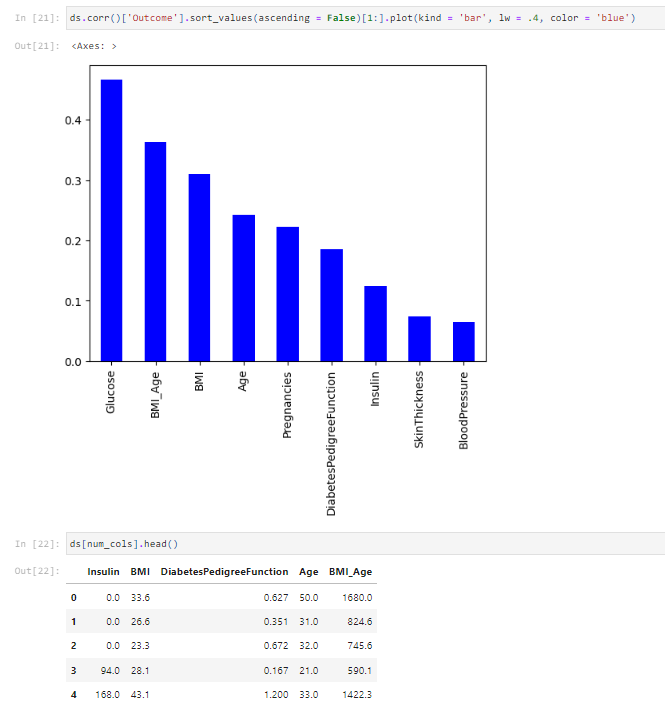
Description automatically generated

**Heatmap.**

****

**A screenshot of a computer screen

Description automatically generated**

****

**Data Pre-processing.**

**Handling Outliers**

**A screenshot of a computer screen

Description automatically generated**

**Feature Engineering**

**A close-up of a white background

Description automatically generated**

**A close-up of a computer code

Description automatically generated**

**Feature Scaling.**

**Text

Description automatically generated**

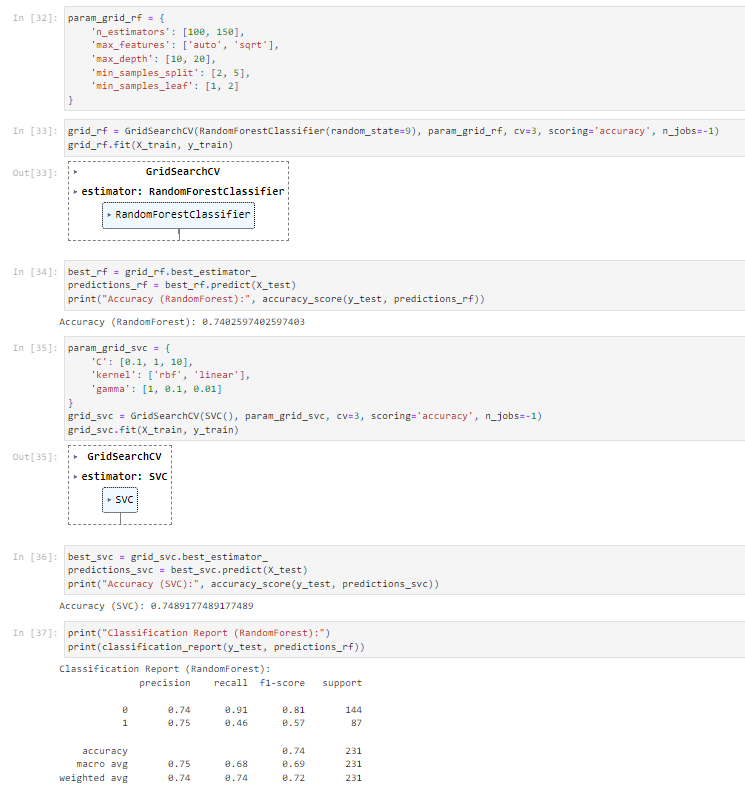
A screenshot of a computer

Description automatically generated

# **4.Results and Discussions**

Graphical user interface, text, application

Description automatically generated



A screenshot of a computer

Description automatically generated

**Confusion Matrix.**

Graphical user interface, text, application

Description automatically generated

A confusion matrix, or table, is a common way to express the efficiency with which a classification model (or "classifier") operates on a set of test data where the true values are known.

**Classification Report.**

The classification report visualization shows the model's precision, recall, F1, and support scores.

Table

Description automatically generated

Accuracy Score = The number of correctly predicted events.

Predictions made in total.

Accuracy Score = 73.59 %

## K-Nearest Neighbors, Support Vector Machines, and Random Forest are the algorithms used in this code.

## The effectiveness of each method was evaluated using the subsequent metrics and 5-fold cross-validation:

## Accuracy

## F1 scores

## ROC AUC results

The results show that all algorithms work reasonably well, with accuracy ratings ranging from 73% to 78%, F1 scores from 62% to 70%, and ROC AUC values from 79% to 84%.

The top-performing algorithms in terms of ROC AUC score were Random Forest (0.84) and SVM (0.82). Of all the methods, K-Nearest Neighbors obtained the lowest ROC AUC score (0.79).

SVM (0.77) and Random Forest (0.78) were the methods that performed best in terms of accuracy. Of all the algorithms, K-Nearest Neighbors has the lowest accuracy score (0.73).

In terms of F1 score, Random Forest (0.70) and SVM (0.68) were the top three algorithms. Of all the methods, K-Nearest Neighbors has the lowest F1 score (0.62).

Overall, the results show that Random Forest is the best algorithms.However, the individual conditions and the trade-off between different performance criteria ultimately determine which technique is used.

## Discussion

Model Effectiveness: The Random Forest model's better performance indicates that ensemble approaches, which mix several decision trees to generate conclusions, are especially good at managing the intricacies and non-linear correlations present in medical data.  
Why Feature Engineering Is Important The BMI\_Age feature's success serves as a testament to the importance of feature engineering in predictive analytics. We can extract more complex relationships from the data that are not visible when examining individual attributes by developing interaction terms.   
  
Implications for Clinical Practice: According to the research, clinical decision support systems that incorporate models such as Random Forest may be able to give early warnings based on patient data, which may help in the early detection and treatment of diabetes.

## Limitations

Despite being comprehensive, the study has a number of flaws that could affect how accurate and broadly applicable the results are. First, depending too much on a single dataset could restrict how broadly the results can be applied, since not all age or ethnic groups would be equally represented in the dataset. Moreover, even if the study's models are strong, they make the assumption that the connections found in the data are static and unchanging, which may not be the case in real-world settings that are dynamic and involve the evolution of diabetes risk variables and their interactions over time. Imputation and data capping were used to handle the handling of missing data and outliers, which has the potential to introduce bias or underestimate variability.

## Future work

Future research might concentrate on other areas in order to build on the results of this study and get over some of its shortcomings. By allowing the models to be tested and trained on a variety of demographics, multi-source data gathering could improve the volume and diversity of data, as well as the resilience and application of the models. Furthermore, in order to manage high-dimensional data and better capture non-linear correlations without requiring considerable feature engineering, the deployment of more complicated models, such deep learning, could be investigated. Sustained enhancement of the model's accuracy and dependability in predicting diabetes may potentially benefit from ongoing validation using newly available data and real-time incorporation into clinical settings.

# **5.References**

AKTURK, M. (2023, 03 03). *Diabetes Dataset*. Retrieved from Kaggle: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Bressler, N. (2023, 04 10). *How to Check the Accuracy of Your Machine Learning Model*. Retrieved from Deep Checks: https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/

*Diabetes*. (2023, 04 13). Retrieved from World Health Organization: https://www.who.int/news-room/fact-sheets/detail/diabetes

ELALFY, M. (2023, 03 25). *diabetes prediction using Machine Learning*. Retrieved from Kaggle: https://www.kaggle.com/code/mohamedelalfy4/diabetes-prediction-using-machine-learning

SUKUMARAN, A. (2023, 03 20). *Diabetes Prediction*. Retrieved from Kaggle: https://www.kaggle.com/code/anjusukumaran4/diabetes-prediction

# **6.Apendix**

# Diabetes Analysis & Prediction

# ML ASSIGNMENT

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib import cm

import seaborn as sns

import os

import warnings

warnings.filterwarnings('ignore')

# for preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split, KFold, cross\_validate , GridSearchCV

# classifiers

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, roc\_auc\_score, f1\_score, classification\_report, confusion\_matrix

# Load the dataset

ds = pd.read\_csv('diabetes.csv')

# Handling Outliers

for col in ['Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']:

    q1 = ds[col].quantile(0.25)

    q3 = ds[col].quantile(0.75)

    iqr = q3 - q1

    fence\_low = q1 - 1.5 \* iqr

    fence\_high = q3 + 1.5 \* iqr

    ds[col] = ds[col].clip(lower=fence\_low, upper=fence\_high)

# Feature Engineering - Adding an interaction term

ds['BMI\_Age'] = ds['BMI'] \* ds['Age']

ds.head()

ds.columns

cat\_cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness']

num\_cols = ['Insulin',  'BMI', 'DiabetesPedigreeFunction', 'Age', 'BMI\_Age']

ds.shape

ds.info()

ds.isnull().any().sum()

ds.describe().T

ds['Outcome'].value\_counts()

ds.duplicated().sum()

print(f"shape before removing duplicates: {ds.shape}")

ds.drop\_duplicates(inplace = True)

print(f"shape after removing duplicates: {ds.shape}")

ds['Outcome'].value\_counts().plot(kind = 'bar', color=['red', 'blue'])

# sns.set\_palette("pastel")

plt.figure(figsize=(20,10))

for i, col in enumerate(num\_cols):

    plt.subplot(2,3, i+1)

    sns.boxplot(data = ds, x = 'Outcome', y = col, palette = 'Pastel1' )

    sns.swarmplot(data = ds, x = 'Outcome', y = col, palette = 'Set1')

    plt.xticks(rotation = 90)

    plt.title(f"{col}", fontsize = 14)

plt.figure(figsize=(100,50))

for i, col in enumerate(cat\_cols):

    plt.subplot(2,4, i+1)

    sns.countplot(data = ds, x = col, palette = 'Set1')

    plt.xticks(rotation = 90)

    plt.title(f"{col}", fontsize = 40)

plt.figure(figsize=(30,15))

for i, col in enumerate(cat\_cols):

    plt.subplot(4,2, i+1)

    sns.swarmplot(data = ds, x = col, y = 'Age', hue = 'Outcome', palette = 'Set1')

    plt.xticks(rotation = 90)

    plt.title(f"{col}", fontsize = 1)

sns.pairplot(ds[['Insulin',  'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']],hue = 'Outcome',palette = 'Set1', diag\_kind='kde')

plt.figure(figsize = (15,12))

sns.heatmap(ds.corr(), annot = True, fmt = '.2f', cmap = 'viridis', cbar = True)

ds.corr()['Outcome'].sort\_values(ascending = False)[1:].plot(kind = 'bar', lw = .4, color = 'blue')

ds[num\_cols].head()

scaler = StandardScaler()

ds[num\_cols] = scaler.fit\_transform(ds[num\_cols])

# Prepare data for modeling

X = ds.drop('Outcome', axis = 1)

y = ds['Outcome'].astype(int)

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=21)

X\_train.shape, X\_test.shape

# standardize only numerical columns

X\_train[num\_cols] = scaler.fit\_transform(X\_train[num\_cols])

X\_test[num\_cols] = scaler.transform(X\_test[num\_cols])

# Tuning Hyperparameters for RandomForest using GridSearchCV

param\_grid\_rf = {

    'n\_estimators': [100, 150],

    'max\_features': ['auto', 'sqrt'],

    'max\_depth': [10, 20],

    'min\_samples\_split': [2, 5],

    'min\_samples\_leaf': [1, 2]

}

# Define the models

key = ['LogisticRegression','KNeighborsClassifier','SVC','RandomForestClassifier']

value = [LogisticRegression(random\_state=9), KNeighborsClassifier(), SVC(), RandomForestClassifier()]

models = dict(zip(key,value))

grid\_rf = GridSearchCV(RandomForestClassifier(random\_state=9), param\_grid\_rf, cv=3, scoring='accuracy', n\_jobs=-1)

grid\_rf.fit(X\_train, y\_train)

# Using the best estimator from grid search for RandomForest

best\_rf = grid\_rf.best\_estimator\_

predictions\_rf = best\_rf.predict(X\_test)

print("Accuracy (RandomForest):", accuracy\_score(y\_test, predictions\_rf))

# Tuning Hyperparameters for SVC using GridSearchCV

param\_grid\_svc = {

    'C': [0.1, 1, 10],

    'kernel': ['rbf', 'linear'],

    'gamma': [1, 0.1, 0.01]

}

grid\_svc = GridSearchCV(SVC(), param\_grid\_svc, cv=3, scoring='accuracy', n\_jobs=-1)

grid\_svc.fit(X\_train, y\_train)

# Using the best estimator from grid search for SVC

best\_svc = grid\_svc.best\_estimator\_

predictions\_svc = best\_svc.predict(X\_test)

print("Accuracy (SVC):", accuracy\_score(y\_test, predictions\_svc))

# Printing classification report for the best RandomForest model

print("Classification Report (RandomForest):")

print(classification\_report(y\_test, predictions\_rf))

# Confusion matrix for the best SVC model

cm\_svc = confusion\_matrix(y\_test, predictions\_svc)

sns.heatmap(cm\_svc, annot=True, fmt='d')

plt.title('Confusion Matrix for SVC')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

cv=KFold(5, shuffle=True, random\_state=21)

def model\_check(X, y, classifiers, cv):

    ''' A function for testing multiple classifiers and return several metrics. '''

    model\_table = pd.DataFrame()

    row\_index = 0

    for cls in classifiers:

        MLA\_name = cls.\_\_class\_\_.\_\_name\_\_

        model\_table.loc[row\_index, 'Model Name'] = MLA\_name

        cv\_results = cross\_validate(

            cls,

            X,

            y,

            cv=cv,

            scoring=('accuracy','f1','roc\_auc'),

            return\_train\_score=True,

            n\_jobs=-1

        )

        model\_table.loc[row\_index, 'Train Roc/AUC Mean'] = cv\_results['train\_roc\_auc'].mean()

        model\_table.loc[row\_index, 'Test Roc/AUC Mean'] = cv\_results['test\_roc\_auc'].mean()

        model\_table.loc[row\_index, 'Test Roc/AUC Std'] = cv\_results['test\_roc\_auc'].std()

        model\_table.loc[row\_index, 'Train Accuracy Mean'] = cv\_results['train\_accuracy'].mean()

        model\_table.loc[row\_index, 'Test Accuracy Mean'] = cv\_results['test\_accuracy'].mean()

        model\_table.loc[row\_index, 'Test Acc Std'] = cv\_results['test\_accuracy'].std()

        model\_table.loc[row\_index, 'Train F1 Mean'] = cv\_results['train\_f1'].mean()

        model\_table.loc[row\_index, 'Test F1 Mean'] = cv\_results['test\_f1'].mean()

        model\_table.loc[row\_index, 'Test F1 Std'] = cv\_results['test\_f1'].std()

        model\_table.loc[row\_index, 'Time'] = cv\_results['fit\_time'].mean()

        row\_index += 1

    model\_table.sort\_values(by=['Test F1 Mean'],

                            ascending=False,

                            inplace=True)

    return model\_table

raw\_models = model\_check(X\_train, y\_train, models.values(), cv)

raw\_models

def f\_imp(classifiers, X, y, bins):

    ''' A function for displaying feature importances'''

    fig, axes = plt.subplots(1, 2, figsize=(20, 8))

    axes = axes.flatten()

    for ax, classifier in zip(axes, classifiers):

        try:

            classifier.fit(X, y)

            feature\_imp = pd.DataFrame(sorted(

                zip(classifier.feature\_importances\_, X.columns)),

                                       columns=['Value', 'Feature'])

            sns.barplot(x="Value",

                        y="Feature",

                        data=feature\_imp.sort\_values(by="Value",

                                                     ascending=False),

                        ax=ax,

                        palette='plasma')

            plt.title('Features')

            plt.tight\_layout()

            ax.set(title=f'{classifier.\_\_class\_\_.\_\_name\_\_} Feature Impotances')

            ax.xaxis.set\_major\_locator(MaxNLocator(nbins=bins))

        except:

            continue

    plt.show()

raw\_models.columns

plt.figure(figsize = (8,5))

sns.barplot(data=raw\_models, x = 'Train Accuracy Mean', y = 'Model Name', palette = 'Set1')

plt.figure(figsize = (8,5))

sns.barplot(data=raw\_models, x = 'Test Accuracy Mean', y = 'Model Name', palette = 'Set1')

raw\_models.set\_index('Model Name', inplace = True)

plt.figure(figsize = (18,8))

raw\_models[['Train Accuracy Mean','Test Accuracy Mean' ]].plot(kind = 'barh', colormap = cm.get\_cmap('Spectral'), legend = False)

from sklearn.metrics import roc\_curve, auc, confusion\_matrix, classification\_report,accuracy\_score

lr = LogisticRegression()

lr.fit(X\_train, y\_train)

pred = lr.predict(X\_test)

print(f'Accuracy score: {round(accuracy\_score(y\_test, pred) \* 100, 2)} %')

print("train score - " + str(lr.score(X\_train, y\_train)))

print("test score - " + str(lr.score(X\_test, y\_test)))

print(classification\_report(y\_test,pred))