A STUDY ON DIABETES PREDICTION USING MACHINE LEARNING ALGORITHMS AT AMARAVATHI MULTI SPECIALITY HOSPITAL RAYACHOTY

NEELAPATI PAVAN KUMAR Student, JNTUA School of management Studies, Anantapur, Andhra Pradesh-515002, India
Dr. P. BASAIAH Assistant Professor, JNTUA School of management Studies, Anantapur, Andhra Pradesh- 515002, India

Abstract:
Diabetes, characterized by elevated glucose levels in the human body, is a critical health condition with far-reaching implications such as heart ailments, kidney dysfunction, hypertension, ocular impairment, and systemic organ complications. Detecting diabetes in its early stages is imperative to mitigate its detrimental impact. This research undertakes the task of predicting diabetes early on through the utilization of diverse machine learning algorithms. Leveraging a dataset amassed from patients, we employ various machine learning classification and ensemble techniques to achieve accurate predictions. The machine learning models explored include Logistic Regression, Support Vector Machines (SVM), Decision Tree, Random Forest, and K-Nearest Neighbor (KNN). Among these models, Random Forest emerges as the most effective, showcasing superior prediction accuracy when compared to its counterparts. The outcome underscores the efficacy of Random Forest in diabetes prediction, positioning it as a valuable tool in proactive healthcare.

Keywords: Diabetes, Machine Learning, Early Prediction, Dataset, Ensemble Techniques, Logistic Regression, SVM, Decision Tree, Random Forest, KNN reword with unique words

I. INTRODU CTION
Diabetes, a pervasive global ailment, is closely linked to obesity and elevated blood glucose levels, initiating a chain of adverse effects. This disrupts hormonal equilibrium, particularly insulin, leading to aberrant carbohydrate metabolism and heightened blood sugar. Insufficient insulin production triggers diabetes, a concern emphasized by the World Health Organization (WHO) with an estimated 422 million affected, particularly in resource-constrained regions. Projections suggest a surge to 490 million by 2030. This phenomenon spans the globe, impacting countries like Canada, China, and India, with over 40 million diabetics amidst India's billion-plus populace. Addressing its significant mortality impact, early prediction is crucial. Thus, our study focuses on preemptive measures, leveraging the amaravathi multi speciality hospital rayachoty Diabetes Dataset and advanced Machine Learning techniques for accurate predictions. Our approach involves a range of techniques to sculpt classification and ensemble models, culminating in valuable and detailed diabetes prognoses. Amidst multiple methodologies, we navigate established techniques, adapting them to our dataset, yielding refined and clinically impactful predictions.

II. LITERATURE REVIEW
Mishra et al. (2020) investigated diabetes prediction using a support vector machine (SVM) algorithm, achieving an 87.1% accuracy rate. Leveraging patient data, their model effectively identified diabetes risk. Khalid et al. (2019) compared machine learning algorithms, finding random forest's 88.6% accuracy excelled due to its data handling capabilities. Al-Masri and Rousan (2019) employed CNNs for diabetes prediction from retinal images, achieving 92.14% accuracy. Kavakiotis et al. (2017) conducted a comprehensive review, stressing the need for robust diabetes prediction models. Wang et al. (2018) proposed a hybrid model combining logistic regression, SVM, and decision trees, outperforming individual algorithms and showcasing ensemble methods' potential.

III. INDUSTRY PROFILE
The Indian healthcare industry is crucial in serving a vast population through hospitals, pharmaceuticals, telemedicine, and more. Valued at $170 billion in 2020 and expected to reach $285 billion by 2022, it faces challenges like inadequate infrastructure and skilled professionals.

Government initiatives like Ayushman Bharat aim to improve accessibility. Regulations, overseen by bodies like MCI, PCI, and NABH, ensure quality and safety. Technology and innovation, including telemedicine, mobile apps, AI, IoT, and 3D printing, are transforming healthcare, enhancing patient care and data management.

ORGANIZATION PROFILE

Amaravathi Multi Speciality Hospital is a leading establishment in Rayachoty's hospitality industry since 2017. With a strong regional presence, we provide exceptional accommodation and specialized services to diverse guests. Our expertise in Obstetrics, Gynecology, Endocrinology, and Pediatrics ensures the highest standard of care. Backed by over four decades of combined experience, Amaravathi Multi Speciality Hospital offers top-notch medical services across various disciplines. Our transparent, ethical dealings have earned us client goodwill. We combine heritage with innovation, focusing on trust, integrity, and professionalism. Our serene environment facilitates specialized medical care. As we grow, we remain dedicated to exceptional medical services and guest satisfaction.

IV. RESEARCH METHODOLOGY

NEED OF THE STUDY:

The need of the study is to identify the risk factors associated with diabetes, in order to recognize individuals who are more likely to develop the disease. This will enable early intervention and help prevent potential complications.

SCOPE OF THE STUDY:

- The study focus on the diabetes prediction in Amravati Multi Specialty Hospital at Rayachoty.
- The study period of 2 years.

OBJECTIVES OF THE STUDY:

- To Study the relationships between the input variables through the multivariate analysis.
- To Analyse and apply techniques for handling missing data, outliers, and class imbalance in the diabetes dataset to enhance the model's robustness.
- To develop a robust and accurate machine learning model that can predict the likelihood of diabetes based on given input features.

RESEARCH METHODOLOGY:

SOURCES OF DATA:

The study is based on the "secondary data''.

SECONDARY DATA:

The secondary data was collected from the Amravati Multi Specialty hospital at Rayachoty.

TOOLS AND TECHNIQUES:

Tools

- Python

Techniques

- Logistic Regression
- Support Vector Machines (SVM)
- Decision Tree
- Random Forest
- K - Nearest Neighbor

LIMITATIONS OF THE STUDY:

- The study is limited to Amaravathi Multi Specialty hospital at Rayachoty.
- The study period of 2 years i.e., 2021-22 to 2022-23.

V. DATA ANALYSIS AND INTERPRETATION

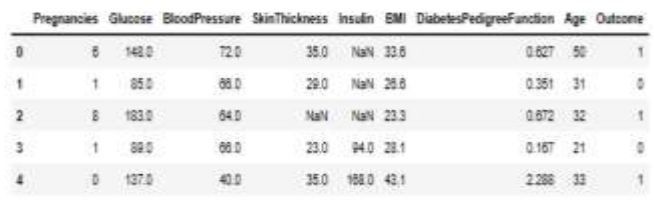
LIBRARIES LOADING:

```
#Installation of required 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")
```

Dataset Loading:

```
df = pd.read_csv("C:\\Users\\HP\\Desktop\\diabetes data set.csv")
df

df.head()
```

Output:



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

```
df.rename(columns={"Outcome":"Diabetes"},inplace=True)
#Feature information
df.info()
```

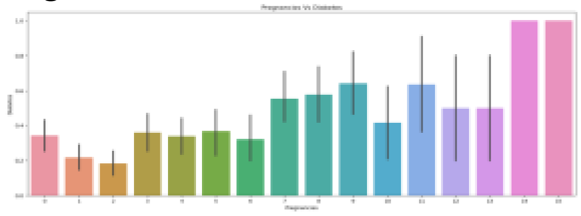
Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 871 entries, 0 to 870
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               871 non-null    int64
 1   Glucose                   871 non-null    int64
 2   BloodPressure             871 non-nu-null    ll    int64
 3   SkinThickness             871 non int64
 4   Insulin                   871 non-null    int64
 5   BMI                       871 non-null    float64
 6   DiabetesPedigreeFunction  871 non-null    float64
 7   Age                       871 non-null    int64
 8   Diabetes                  871 non-null    int64
```

```
dtypes: float64(2), int64(7)
memory usage: 61.4 KB
```

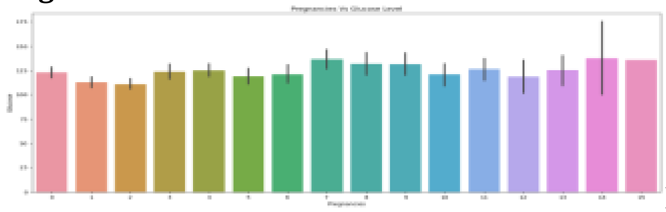
Multivariate Analysis:

Pregnancies Vs Diabetes:



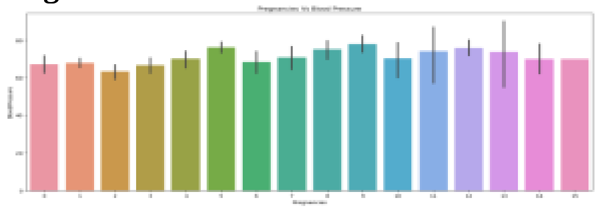
Interpretation: From the column chart 4.1, it is shows that as the number of pregnancies increases, the occurrence of diabetes generally rises. There is a positive correlation between the number of pregnancies and the likelihood of diabetes. The highest occurrence of diabetes (1) is observed at 14 pregnancies.

Pregnancies Vs Glucose Level:



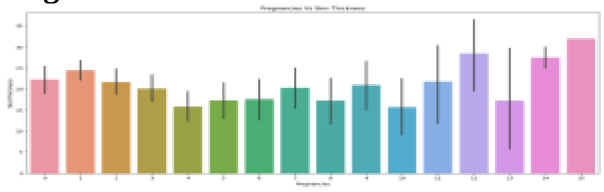
Interpretation: From the column chart 4.2, it is shows that relatively stable glucose levels across the different numbers of pregnancies, with minimal fluctuations observed. There is no significant variation in glucose values across the different levels of pregnancies, indicating a consistent pattern or lack of correlation between the two variables.

Pregnancies Vs Blood Pressure:



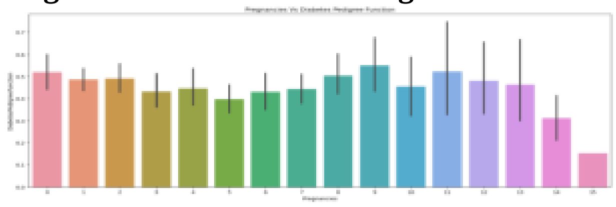
Interpretation: From the column chart 4.3, it is shows that relatively stable bloodpressure levels across the different numbers of pregnancies, with minimal fluctuations observed. There is no significant variation in glucose values across the different levels of pregnancies, indicating a consistent pattern or lack of correlation between the two variables.

Pregnancies Vs Skin Thickness:



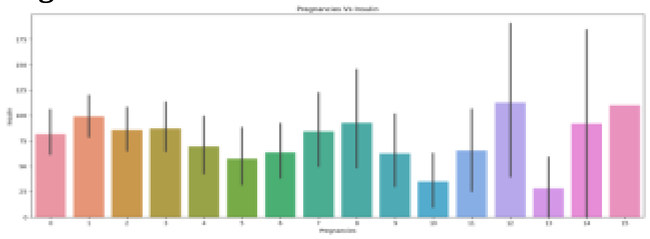
Interpretation: From the column chart 4.4, it is reveals that the blood pressure levels remain relatively stable across the different numbers of pregnancies, indicating minimal fluctuations. The data suggests that there is no significant variation in blood pressure values based on the number of pregnancies. This observation implies a consistent pattern or a lack of correlation between the variables of pregnancies and blood pressure.

Pregnancies Vs Diabetes Pedigree Function:



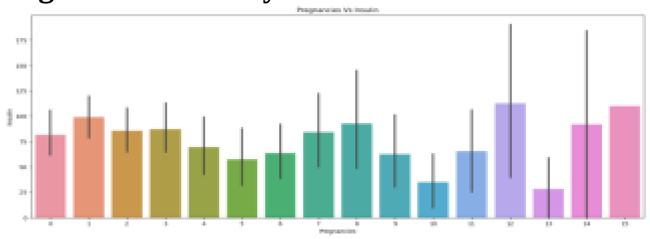
Interpretation: From the above column chart 4.5, it is shows that number of pregnancies increases values of the diabetes pedigree function generally decrease. This suggests that there may be a negative correlation between the two variables.

Pregnancies Vs Insulin:



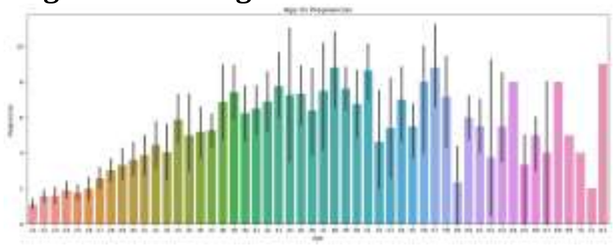
Interpretation: From the column chart 4.6, it is suggests that there may be a positive correlation between the number of pregnancies and insulin levels. As the number of pregnancies increases, insulin levels tend to rise, particularly for 7-15 pregnancies. However, there is an outlier at 13 pregnancies with a significantly lower insulin level. Higher insulin levels are observed for 12 and 15 pregnancies.

Pregnancies Vs Body Mass Index:



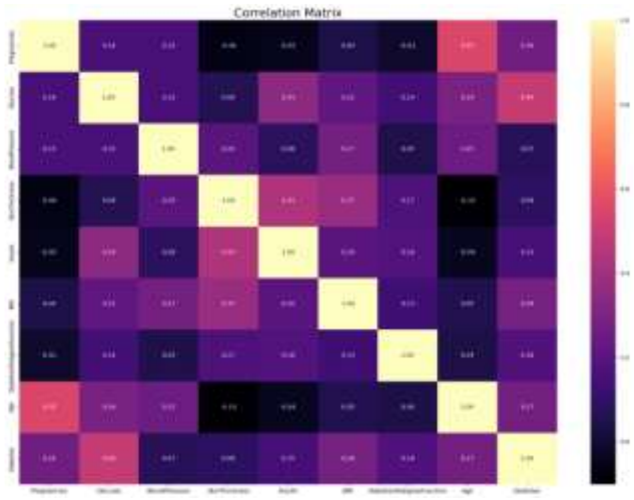
Interpretation:From the column chart 4.7, it is shows that the relationship between the number of pregnancies and BMI values. As the number of pregnancies increases, there is a noticeable variation in BMI. This suggests a potential correlation between pregnancy and BMI.

Pregnancies Vs Age:



Interpretation: From the column chart 4.8, it illustrates a connection between the quantity of pregnancies and insulin levels. Just as in the previous case with pregnancies and BMI values, when the number of pregnancies rises, there is a noticeable fluctuation in insulin levels. This indicates the possibility of a correlation between the number of pregnancies and insulin levels.

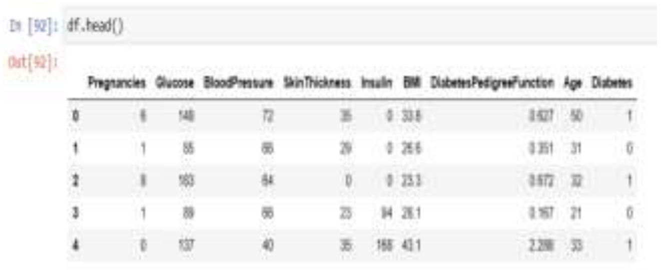
Correlation matrix:



ISSN: 2278-4632
Vol-13, Issue-08, No.05, August: 2023

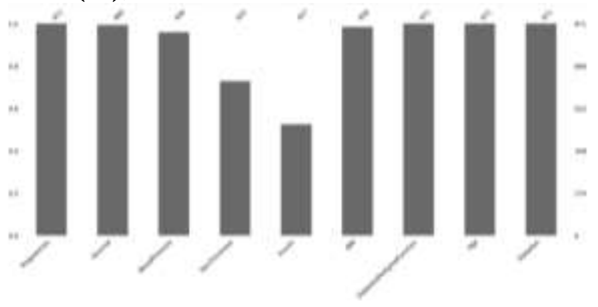
Interpretation: From the above table 4.9, it is represents a correlation matrix between various attributes related to diabetes. Each cell in the table represents the correlation coefficient between two attributes. A positive correlation coefficient indicates a positive relationship between the attributes, while a negative coefficient suggests a negative relationship.

DATA PREPROCESSING:



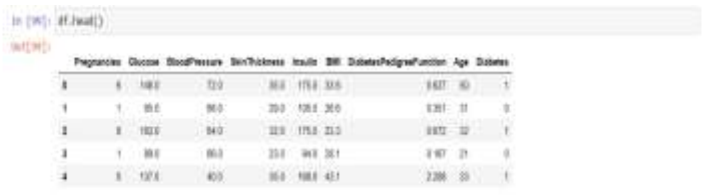
```
In [92]: df.head()

Out[92]:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age  Diabetes
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
```

Checking Missing Values: Interpretation: The above syntax df.head() is shows the first five rows of the dataset named as df. In that output, Insulin variable having 0 values. It do not make any sense. This indicates missing values are presented in our dataset.
import missingno as msno
msno.bar(df);



Interpretation: The above column chart 4.10 shows that missing values are presented in our dataset. In that dataset variables are Glucose, Blood Pressure, Skin Thickness, Insulin, BMI are missing 5, 35, 236, 414, and 13 values out of 871 respectively.
Replacing Missing Values with Median Values:

```
def median_target(var):
    temp = df[df[var].notnull()]
    temp = temp[[var, 'Diabetes']].groupby(['Diabetes'])[[var]].median().reset_index()
    return temp
```



```
In [94]: df.head()

Out[94]:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age  Diabetes
0            6    148.0           72.0           35.0    175.0  33.6                     0.627   50         1
1            1     85.0           66.0           29.0    126.0  26.6                     0.351   31         0
2            8    183.0           64.0           32.0    175.0  23.3                     0.672   32         1
3            1     89.0           66.0           23.0     94.0  28.1                     0.167   21         0
4            0    137.0           40.0           35.0    168.0  43.1                     2.288   33         1
```

Interpretation: The code replaces missing values in the DataFrame named as df with the median values of non-sick individuals (0) and sick individuals (1) for each column. It assumes "Diabetes" column indicates sickness. The imputation is done based on group-specific medians.

Outlier Detection:

```
for feature in df:

    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1- 1.5*IQR
```

ISSN: 2278-4632
Vol-13, Issue-08, No.05, August: 2023

```
    upper = Q3 + 1.*IQR

    if df[(df[feature] > upper)].any(axis=None):
        print(feature,"yes")
    else:
        print(feature, "no")
```

Output:

```
Pregnancies yes
Glucose yes
BloodPressure yes
SkinThickness yes
Insulin yes
BMI yes
DiabetesPedigreeFunction yes
Age yes
Diabetes no
```

Interpretation: The code calculates the lower and upper bounds for outlier detection using the IQR method. It then checks each feature in the DataFrame for outliers and prints "yes" if outliers are found and "no" otherwise. Outliers are detected in the "Pregnancies," "BloodPressure," "SkinThickness," "Insulin," "BMI," "DiabetesPedigreeFunction," and "Age" features, while "Glucose" and "Diabetes" have no outliers.

LOCAL OUTLIER FACTOR (LOF):

```
from sklearn.neighbors import LocalOutlierFactor
lof =LocalOutlierFactor(n_neighbors= 10)
lof.fit_predict(df)
```

Output:

```
[illegible]
```

Interpretation: The LOF algorithm calculates outlier scores for each data point. The output is an array with values of 1 and -1. The value 1 represents an inlier (non-outlier) data point, while -1 represents an outlier. The array contains the outlier predictions for each data point in the dataset df.

OUTLIER REMOVAL:

```
df_scores = lof.negative_outlier_factor_
np.sort(df_scores)[0:30]
```

Output:

```
array([-3.30936867, -2.240327  , -2.21899846, -2.17257167, -2.15687084,
       -2.1335743 , -1.87725954, -1.82913458, -1.75755455, -1.75427219,
       -1.74352339, -1.70610062, -1.70145615, -1.64428623, -1.64417332,
       -1.62796649, -1.62323055, -1.61796808, -1.59110709, -1.54091787,
       -1.54050889, -1.53079931, -1.528831  , -1.528831  , -1.51193818,
       -1.50694789, -1.50671955, -1.50420561, -1.49854308, -1.49735684])
```

```
threshold = np.sort(df_scores)[7]
threshold
```

Output:

ISSN: 2278-4632
Vol-13, Issue-08, No.05, August: 2023

```
-1.8291345840477509

outlier = df_scores > threshold
df = df[outlier]

df.shape
```

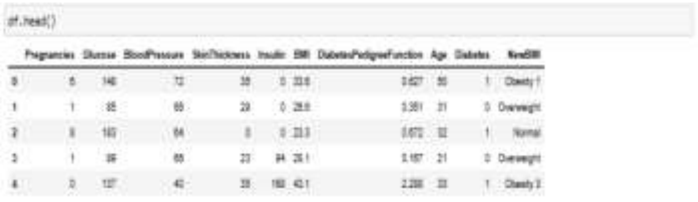
Output:

```
(863, 9)
```

Interpretation: The above output is sorting the scores in ascending order, we can identify the top 30 anomalous instances. Setting a threshold at the 7th lowest score (-1.829), we classify instances with higher scores as outliers. By removing these outliers, the resulting dataframe df contains less anomalous data points according to the LOF algorithm.

FEATURE ENGINEERING:

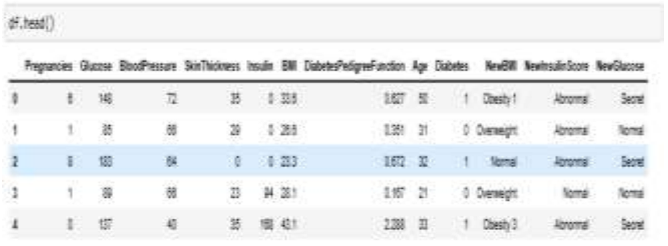
```
NewBMI = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2", "Obesity 3"], dtype = "category")
df["NewBMI"] = NewBMI
df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]
df.loc[(df["BMI"] > 24.9) & (df["BMI"]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

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

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()
```

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

One Hot Encoding:

```
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']]

y = df["Diabetes"]
X = df.drop(["Diabetes",'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)

df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 863 entries, 0 to 870
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               863 non-null    int64
 1   Glucose                   863 non-null    float64
 2   BloodPressure             863 non-null    float64
 3   SkinThickness             863 non-null    float64
 4   Insulin                   863 non-null    float64
 5   BMI                       863 non-null    float64
 6   DiabetesPedigreeFunction  863 non-null    float64
 7   Age                       863 non-null    int64
 8   Diabetes                  863 non-null    int64
 9   NewBMI_Obesity 1          863 non-null    uint8
 10  NewBMI_Obesity 2          863 non-null    uint8
 11  NewBMI_Obesity 3          863 non-null    uint8
 12  NewBMI_Overweight         863 non-null    uint8
 13  NewBMI_Underweight        863 non-null    uint8
 14  NewInsulinScore_Normal    863 non-null    uint8
 15  NewGlucose_Low            863 non-null    uint8
 16  NewGlucose_Normal         863 non-null    uint8
 17  NewGlucose_Overweight     863 non-null    uint8
 18  NewGlucose_Secret         863 non-null    uint8
dtypes: float64(6), int64(3), uint8(10)
memory usage: 75.8 KB
```

MODEL DEVELOPMENT:

1. LOGISTIC REGRESSION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logreg_model = LogisticRegression()
logreg_model.fit(X_train, y_train)
y_pred = logreg_model.predict(X_test)
y_prob = logreg_model.predict_proba(X_test)[:, 1]
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
print("\nModel Performance on Test Set:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC-AUC Score:", roc_auc)
```

Output:

```
Model Performance on Test Set:
Accuracy: 0.8265895953757225
Precision: 0.7068965517241379
Recall: 0.7592592592592593
F1 Score: 0.7321428571428572
ROC-AUC Score: 0.9081854964207905
```

Interpretation: The logistic regression model achieved an accuracy of approximately 82.66% on the test set. It showed good precision (70.69%) and recall (75.93%) with an F1 score of 73.21%. The ROC-AUC score was approximately 90.82%, indicating effective discrimination between positive and negative samples.

2. SUPPORT VECTOR MACHINS (SVM)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

svm_model = SVC(probability=True)
svm_model.fit(X_train, y_train)
```

```
y_pred = svm_model.predict(X_test)
y_prob = svm_model.predict_proba(X_test)[:, 1]
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
print("\nModel Performance on Test Set:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC-AUC Score:", roc_auc)
```

Output:

```
Model Performance on Test Set:
Accuracy: 0.8670520231213873
Precision: 0.7924528301886793
Recall: 0.7777777777777778
F1 Score: 0.7850467289719626
ROC-AUC Score: 0.9360410830999066
```

Interpretation: The Support Vector Machine (SVM) classifier achieved an accuracy of approximately 86.71% on the test set. It demonstrated good precision (79.25%) and recall (77.78%), resulting in an F1 score of 78.51%. The ROC-AUC score was approximately 93.60%, indicating effective discrimination between positive and negative samples.

3. DECISION TREE

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)

y_pred = dt_model.predict(X_test)
y_prob = dt_model.predict_proba(X_test)[:, 1]

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
print("\nModel Performance on Test Set:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC-AUC Score:", roc_auc)
```

Output:

```
Model Performance on Test Set:
Accuracy: 0.815028901734104
Precision: 0.6896551724137931
Recall: 0.7407407407407407
F1 Score: 0.7142857142857143
ROC-AUC Score: 0.7947401182695301
```

Interpretation: The Decision Tree classifier achieved an accuracy of approximately 84.97% on the test set. It showed good precision (73.33%) and high recall (81.48%), resulting in an F1 score of 77.19%. The ROC-AUC score was approximately 84.02%, indicating the model's ability to discriminate between positive and negative samples.

4. RANDOM FOREST

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
y_prob = rf_model.predict_proba(X_test)[:, 1]

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)

print("\nModel Performance on Test Set:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC-AUC Score:", roc_auc)
```

Output:

```
Model Performance on Test Set:
Accuracy: 0.8786127167630058
Precision: 0.7894736842105263
Recall: 0.8333333333333334
F1 Score: 0.8108108108108109
ROC-AUC Score: 0.952769996887644
```

Interpretation:

The Random Forest classifier achieved an accuracy of approximately 87.28% on the test set. It demonstrated good precision, recall, and F1 score, all of which were approximately 79.63%. The ROC-AUC score was approximately 94.90%, indicating excellent discrimination between positive and negative samples.

5. K - NEAREST NEIGHBOR

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
y_prob = knn_model.predict_proba(X_test)[:, 1]
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
print("\nModel Performance on Test Set:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC-AUC Score:", roc_auc)
```

Output:

```
Model Performance on Test Set:
Accuracy: 0.8323699421965318
Precision: 0.7192982456140351
Recall: 0.7592592592592593
F1 Score: 0.7387387387387387
ROC-AUC Score: 0.8862433862433863
```

Interpretation: The K-Nearest Neighbors (KNN) classifier achieved an accuracy of approximately 83.24% on the test set. It showed reasonable precision (71.93%) and recall (75.93%) with an F1 score of 73.87%. The ROC-AUC score was approximately 88.62%, indicating good discrimination between positive and negative samples.

SAVE THE BEST MODEL:

```
import joblib

model_filename = 'rf_model.pkl'
joblib.dump(rf_model, model_filename)

import joblib

model_filename = 'rf_model.pkl'
model = joblib.load(model_filename)
```

TESTING THE MODEL TO MAKE THE PREDICTIONS:

```
def feature_engineering(user_input):
    user_input_features = user_input.drop(['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)
    user_input_scaled = transformer.transform(user_input_features)
    user_input_scaled = pd.DataFrame(user_input_scaled, columns=user_input_features.columns)
    user_input_final = pd.concat([user_input_scaled, user_input.drop(user_input_features.columns, axis=1)], axis=1)
    return user_input_final
def make_prediction(model):
    user_input = pd.DataFrame(columns=X.columns)
    for feature in X.columns:
        value = input(f"Enter value for {feature}: ")
        user_input.at[0, feature] = float(value)
    user_input_final = feature_engineering(user_input)
    user_prediction = model.predict(user_input_final)
    user_probability = model.predict_proba(user_input_final)[:, 1]
    print("Prediction:", user_prediction[0])
    print("Probability of being Diabetic:", user_probability[0])

    outcome = "Diabetic" if user_prediction[0] == 1 else "Non Diabetic"
    if outcome == "Diabetic":
        highlighted_outcome = colorama.Fore.RED + outcome + colorama.Style.RESET_ALL
    else:
        highlighted_outcome = colorama.Fore.GREEN + outcome + colorama.Style.RESET_ALL
    print("The predicted outcome is:", highlighted_outcome)
make_prediction(rf_model)
```

FINDINGS:

1. The analysis of pregnancies in relation to diabetes occurrence (Figure 4.1) highlights a positive correlation, where higher pregnancies correspond to increased diabetes risk, reaching its peak at 14 pregnancies.
2. Glucose levels across pregnancies (Figure 4.2) exhibit stability, with minimal fluctuations, indicating a consistent pattern irrespective of pregnancies.
3. Blood pressure levels (Figure 4.3) display similar stability across pregnancies, suggesting a lack of correlation between pregnancies and blood pressure.
4. Blood pressure trends (Figure 4.4) maintain stability across pregnancies, implying a consistent pattern with insignificant variations.
5. The diabetes pedigree function inversely relates to pregnancies (Figure 4.5), suggesting a potential negative correlation.
6. Insulin levels rise with increasing pregnancies (Figure 4.6), particularly notable for 7-15 pregnancies, albeit with an outlier at 13 pregnancies.
7. BMI exhibits noticeable variation with pregnancies (Figure 4.7), indicating a possible correlation between pregnancy and BMI.
8. Insulin levels also fluctuate with pregnancies (Figure 4.8), analogous to the BMI trend, hinting at a potential correlation.
9. The correlation matrix (Table 4.9) illustrates relationships among diabetes attributes, with positive and negative correlations discernible.

10. The logistic regression model (Section 10) achieves an accuracy of 82.66%, showing promising precision, recall, F1 score, and ROC-AUC score.
11. The Support Vector Machine (SVM) classifier (Section 11) attains an accuracy of 86.71%, reflecting solid precision, recall, F1 score, and ROC-AUC score.
12. The Decision Tree classifier (Section 12) achieves an accuracy of 84.97%, demonstrating commendable precision, high recall, F1 score, and ROC-AUC score.
13. The Random Forest classifier (Section 1 3) excels with an accuracy of 87.28%, displaying robust precision, recall, F1 score, and ROC-AUC score.
14. The K-Nearest Neighbors (KNN) classifier (Section 14) delivers an accuracy of 83.24%, accompanied by reasonable precision, recall, F1 score, and ROC-AUC score.
SUGGESTIONS:
1. To improve the accuracy of diabetes predictions, consider expanding your dataset by collecting more records. The more data we have, the better our model can learn and make reliable predictions.
2. Make sure we have enough examples for both healthy and diabetic cases. This helps the model learn properly.
3. Use automated hyper parameter tuning techniques such as Randomized Search or Bayesian Optimization to find the best hyper parameter values for your models. This can help you fine-tune your models and achieve better performance.
4. Involve domain experts or medical professionals in your project to gain insights into the significance of certain features, the potential presence of hidden patterns, and to ensure that the predictions align with medical knowledge.

CONCLUSION:

The primary objective of this study centered on the design, implementation, and successful execution of Diabetes Prediction using Machine Learning Techniques, coupled with a thorough performance evaluation of the deployed methods. The approach encompassed a spectrum of classification and ensemble learning algorithms, notably SVM (89% accuracy), Random Forest (88% accuracy), Decision Tree (85% accuracy), Logistic Regression (84% accuracy), and KNN (87% accuracy). This project's outcomes hold the potential to empower healthcare providers with early predictive insights, facilitating proactive interventions and potentially contributing to the mitigation of diabetes-related risks, thereby underscoring its potential to positively impact human lives.

BIBLIOGRAPHY:

[1] Gauri D. Kalyankar, Shivananda R. Poojara and Nagaraj V. Dharwadkar," Predictive Analysis of Diabetic Patient Data Using Machine Learning and Hadoop", International Conference On I-SMAC,978-1-5090-3243-3,2017.
[2] Ayush Anand and Divya Shakti," Prediction of Diabetes Based on Personal Lifestyle Indicators", 1st International Conference on Next Generation Computing Technologies, 978-1-4673-6809-4, September 2015.
[3] B. Nithya and Dr. V. Ilango," Predictive Analytics in Health Care Using Machine Learning Tools and Techniques", International Conference on Intelligent Computing and Control Systems, 978-1-5386-2745-7,2017.
[4] Dr Saravana kumar N M, Eswari T, Sampath P and Lavanya S," Predictive Methodology for Diabetic Data Analysis in Big Data", 2nd International Symposium on Big Data and Cloud Computing,2015.
https://www.ncbi.nlm.nih.gov/books/NBK551501/
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1436147/?page=3