AI BASED DIABETES PREDICTION

TEAM MEMBER

au922321104035:S. Santhiya Devi

PHASE 3 PROJECT SUBMISSION

INTRODUCTION

In AI-based diabetes prediction, data loading and preprocessing are crucial steps to ensure the accuracy and reliability of your predictive model. Here's a step-by-step guide on how to load and preprocess data for diabetes prediction:

1. Data Collection

- First, you need to obtain a dataset that contains relevant information about individuals, such as age, gender, body mass index (BMI), blood pressure, glucose levels, and whether they have diabetes or not. You can find such datasets from sources like the UCI Machine Learning Repository or government health agencies.

2. Data Loading

- Import necessary libraries such as pandas and numpy in

Python.
```python
import pandas as pd import numpy as np
```

- Load your dataset into a pandas DataFrame. Here's an example of loading a CSV file:

```python
data = pd.read_csv('diabetes_dataset.csv')
```

PROGRAM

1. Import the important

Libraries In [1]:

```
import pandas as pd import numpy as np import matplotlib.pyplot as
plt import seaborn as sns from statistics import mean, stdev from
sklearn.model_selection import StratifiedKFold from
sklearn.preprocessing import StandardScaler from
sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score,
mean_squared_error from sklearn.linear_model import
LogisticRegression from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier from
sklearn.svm import SVC from sklearn.tree import
DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier from
sklearn.ensemble import GradientBoostingClassifier from
lightgbm import LGBMClassifier import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
```

2. Loading the Dataset

In [2]:

```
diabetes = pd.read_csv("/kaggle/input/diabetes-dataset/diabetes.csv")
```

3. Inspecting the

Dataset In [3]:

```
diabetes.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1

3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [4]:

```
diabetes.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767 Data
columns (total 9 columns):
 #   Column                         Non-Null Count  Dtype
---  ------              -------------  -----
0   Pregnancies                    768 non-null    int64
1   Glucose                        768 non-null    int64
2   BloodPressure                  768 non-null    int64
3   SkinThickness                  768 non-null    int64
4   Insulin                        768 non-null    int64
5   BMI                            768 non-null    float64
6   DiabetesPedigreeFunction       768 non-null    float64
7   Age                            768 non-null    int64    8
    Outcome              768 non-null    int64
dtypes: float64(2), int64(7) memory
usage: 54.1 KB
```

In [5]:

```
diabetes.describe() Out[5]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.00000	140.250	80.000000	32.00000	127.250	36.6000	0.626250	41.0000	1.00000

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
	0	000		0	000	00		00	0
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

3. Data Exploration

- Explore your dataset to understand its structure and characteristics. This can include checking for missing values, understanding the distribution of features, and performing descriptive statistics.

```python
# Check for missing values

print(data.isnull().sum
```

4. Data Preprocessing:

- Data preprocessing is essential to clean and transform the data for machine learning. Common preprocessing steps include:

- Handling Missing Values: You can either remove rows with missing data or impute missing values using techniques like mean, median, or machine learning-based imputation. Python data = data.dropna()

- Encoding Categorical Variables: If your dataset contains categorical variables (e.g., gender), you may need to encode them into numerical values using one-hot encoding or label encoding.

```python
data = pd.get_dummies(data, columns=['gender'],
drop_first=True)  # One-hot encoding



```

INPUT:

```
diabetes.drop_duplicates()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0

765	5	121	72	23	112	26.2	0.245	30	0
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

4.1 Outliers

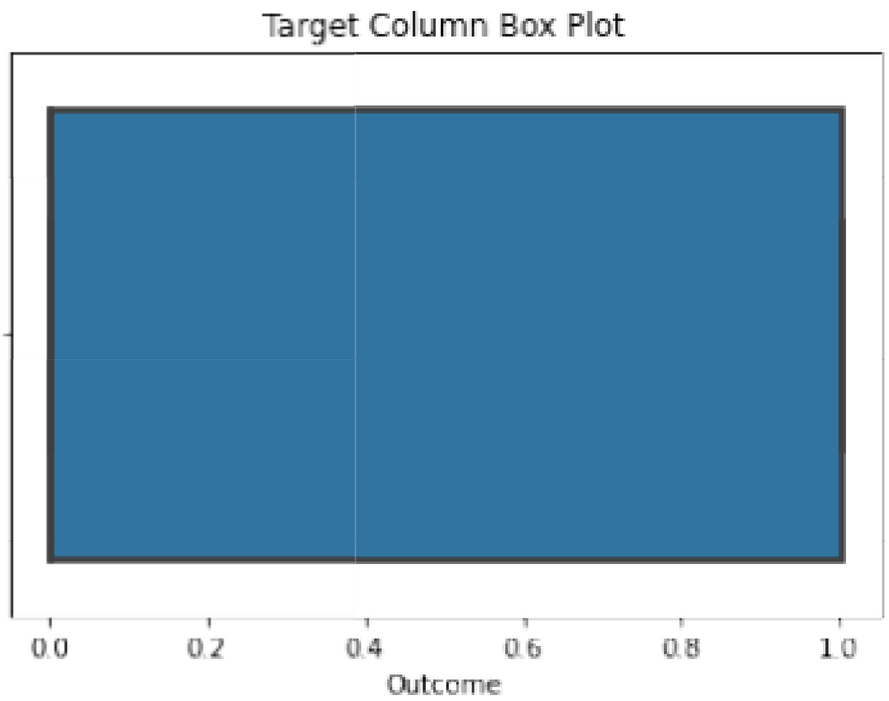
Checking for outliers using the box plot.

In [7]:

linkcode

```
#first store the features in a seperate dataframe. features =
diabetes.drop("Outcome",axis = 1).copy() #Now plot a
boxplot to identify the outliers in our features.
sns.boxplot(data = features, orient = 'h', palette = 'Set3',
linewidth = 2.5 ) plt.title("Features Box Plot") Out[7]:
```

```
Text(0.5, 1.0, 'Features Box Plot')
```



```
from scipy import stats def
removeoutliers(df=None, columns=None):
    for column in columns:
        Q1 = df[column].quantile(0.25)
Q3 = df[column].quantile(0.75)
IQR = Q3 - Q1
        floor, ceil = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
df[column] = df[column].clip(floor, ceil)
print(f"The columnn: {column}, has been treated for
outliers.\n")
```

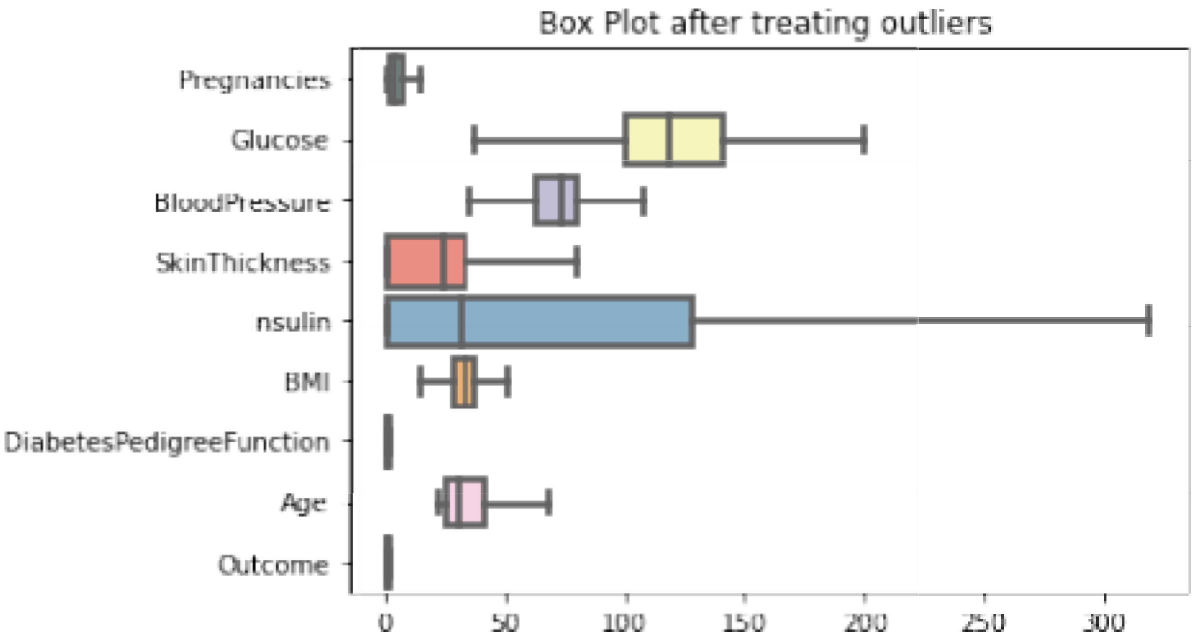
```
    return df

diabetes = removeoutliers(diabetes,[col for col in features.columns])
```

```
The columnn: Pregnancies, has been treated for outliers.

The columnn: Glucose, has been treated for outliers.

The columnn: BloodPressure, has been treated for outliers.

The columnn: SkinThickness, has been treated for outliers.

The columnn: Insulin, has been treated for outliers.

The columnn: BMI, has been treated for outliers.

The columnn: DiabetesPedigreeFunction, has been treated for outliers.

The columnn: Age, has been treated for outliers.
```

In [10]:

```
sns.boxplot(data = diabetes, orient = 'h', palette = 'Set3',
linewidth = 2.5 )

plt.title("Box Plot after treating outliers")
```

.


Box Plot after treating outliers
Pregnancies
Glucose
BloodPressure
SkinThickness
nsulin
BMI
DiabetesPedigreeFunction
Age
Outcome