DOMAIN NAME : ARTIFICIAL INTELLIGENCE

PROJECT NAME: DIABETES PREDICTION SYSTEM USING AI

SUBMITTED BY:

- SATHI SRI.C
- RAJESHWARI.M
- MARIAMMAL.M
- RATHNA SRI.R.K
- SELVAESWARI.S

We will be predicting that whether the patient has diabetes or not on the basis of the features we will provide to our machine learning model, and for that, we will be using the famous Pima Indians Diabetes Database

- .PURPOSE:

Diabetes can affect almost every part of your body. Therefore, you will need to manage your blood glucose levels, also called blood sugar. Managing your blood glucose, as well as your blood pressure and cholesterol, can help prevent the health problems that can occur when you have diabetes.

PROBLEM STATEMENT:

The problem statement is to determine which algorithm is more accurate in predicting diabetes. The methodology involves implementing the SVM and decision tree algorithms on the dataset and evaluating their performance using metrics such as accuracy, precision, and recall.

DESIGN THINKING:

He SVM algorithm performs better than the decision tree algorithm, with an accuracy of 76.6% compared to 75%. This work concludes that the SVM algorithm is more accurate in predicting diabetes and can be a valuable tool for early detection and management of the disease.

[11/1, 9:53 PM]

PHASES OF DEVELOPMENT:

There are 5 phases to develop the diabetes prediction system

The phase are

[11/1, 10:19 PM]

1. Data analysis: Here one will get to know about how the data analysis

Part is done in a data science life cycle.

2. Exploratory data analysis: EDA is one of the most important steps in

The data science project life cycle and here one will need to know that

How to make inferences from the visualizations and data analysis

3. Model building: Here we will be using 4 ML models and then we will

Choose the best performing model.

4. Saving model: Saving the best model using pickle to make the

Prediction from real data.

[11/1, 10:19 PM]

Details about the dataset:

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)

DiabetesPedigreeFunction: Diabetes pedigree function

Age: Age (years)

Outcome: Class variable (0 or 1)

) Exploratory Data Analysis

#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”)

#Reading the dataset

Df = pd.read_csv(“../input/pima-indians-diabetes-database/diabetes.csv”)

# The first 5 observation units of the data set were accessed.

Df.head()
```

Diabetes Prediction using Machine Learning

Diabetes, is a group of metabolic disorders in which there are high blood sugar levels over a prolonged period. Symptoms of high blood sugar include frequent urination, increased thirst, and increased hunger. If left untreated, diabetes can cause many complications. Acute complications can include diabetic ketoacidosis, hyperosmolar hyperglycemic state, or death. Serious long-term complications include cardiovascular disease, stroke, chronic kidney disease, foot ulcers, and damage to the eyes.

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Objective

We will try to build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not?

Details about the dataset:

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)

DiabetesPedigreeFunction: Diabetes pedigree function

Age: Age (years)

Outcome: Class variable (0 or 1)

Number of Observation Units: 768

Variable Number: 9

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

1) Exploratory Data Analysis

#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”)

#Reading the dataset

Df = pd.read_csv(“../input/pima-indians-diabetes-database/diabetes.csv”)

# The first 5 observation units of the data set were accessed.

Df.head()

Examined. It consists of 768 observation units and 9 variables.

Df.shape

(768, 9)

#Feature information

Df.info()

<class ‘pandas.core.frame.DataFrame’>

RangeIndex: 768 entries, 0 to 767
```

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Pregnancies	768 non-null	int64
1	Glucose	768 non-null	int64
2	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

Descriptive statistics of the data set accessed.

Df.describe([0.10,0.25,0.50,0.75,0.90,0.95,0.99]).T

	Count	mean	std	min	10%	25%	50%	75%	90%	95%	99%	max
Pregnancies	768.0	3.845052	3.369578	0.000	0.000	1.00000	3.0000	6.00000	9.0000	10.00000	13.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	85.000	99.00000	117.0000	140.25000	167.0000	181.00000	196.00000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	54.000	62.00000	72.0000	80.00000	88.0000	90.00000	106.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.000	0.00000	23.0000	32.00000	40.0000	44.00000	51.33000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.000	0.00000	30.5000	127.25000	210.0000	293.00000	519.90000	846.00
BMI	768.0	31.992578	7.884160	0.000	23.600	27.30000	32.0000	36.60000	41.5000	44.39500	50.75900	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.165	0.24375	0.3725	0.62625	0.8786	1.13285	1.69833	2.42

Age 768.0 33.240885 11.760232 21.000 22.000 24.00000 29.0000
41.00000 51.0000 58.00000 67.00000 81.00

Outcome 768.0 0.348958

The distribution of the Outcome variable was examined.

Df[“Outcome”].value_counts()*100/len(df)

0 65.104167
1 1 34.895833

Name: Outcome, dtype: float64

The classes of the outcome variable were examined.

Df.Outcome.value_counts()

0 500
1 1 268

Name: Outcome, dtype: int64

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()))

Max Age: 81 Min Age: 21

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])

<matplotlib.axes._subplots.AxesSubplot at 0x7f77b83d5950>

Df.groupby(“Outcome”).agg({“Pregnancies”:”mean”})

Pregnancies

Outcome

0 3.298000

1 4.865672

Df.groupby(“Outcome”).agg({“Age”:”mean”})

Age

Outcome

0 31.190000

1 37.067164

Df.groupby(“Outcome”).agg({“Age”:”max”})

Age

Outcome

0 81

1 70

Df.groupby(“Outcome”).agg({“Insulin”: “mean”})

Insulin

Outcome

0 68.792000

1 100.335821

Df.groupby(“Outcome”).agg({“Insulin”: “max”})

Insulin

Outcome

0 744

1 846

Df.groupby(“Outcome”).agg({“Glucose”: “mean”})

Glucose

Outcome

0 109.980000

1 141.257463

Df.groupby(“Outcome”).agg({“Glucose”: “max”})

Glucose

Outcome

0 197

1 199

Df.groupby(“Outcome”).agg({“BMI”: “mean”})

BMI

Outcome

0 30.304200

1 35.142537

The distribution of the outcome variable in the data was examined and visualized.

F,ax=plt.subplots(1,2,figsize=(18,8))

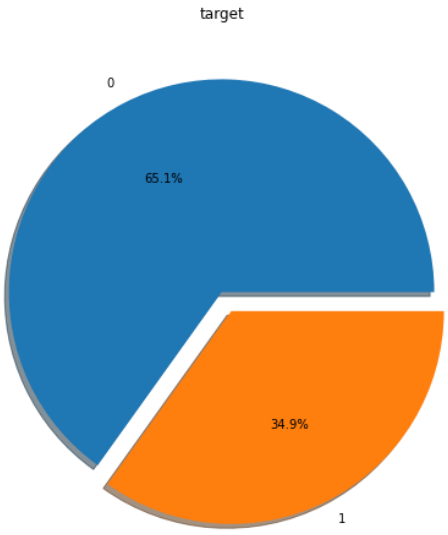
Df[‘Outcome’].value_counts().plot.pie(explode=[0,0.1],autopct=’%1.1f%%’,ax=ax[0],shadow=True)

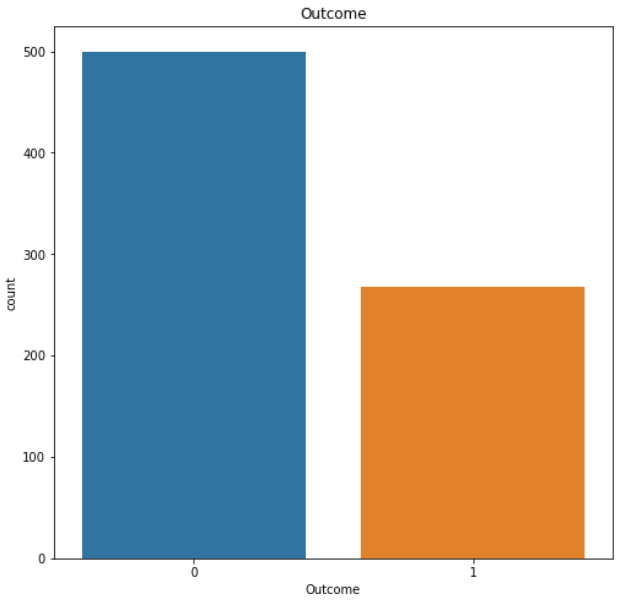
Ax[0].set_title(‘target’)

Ax[0].set_ylabel(‘’)

Sns.countplot(‘Outcome’,data=df,ax=ax[1])

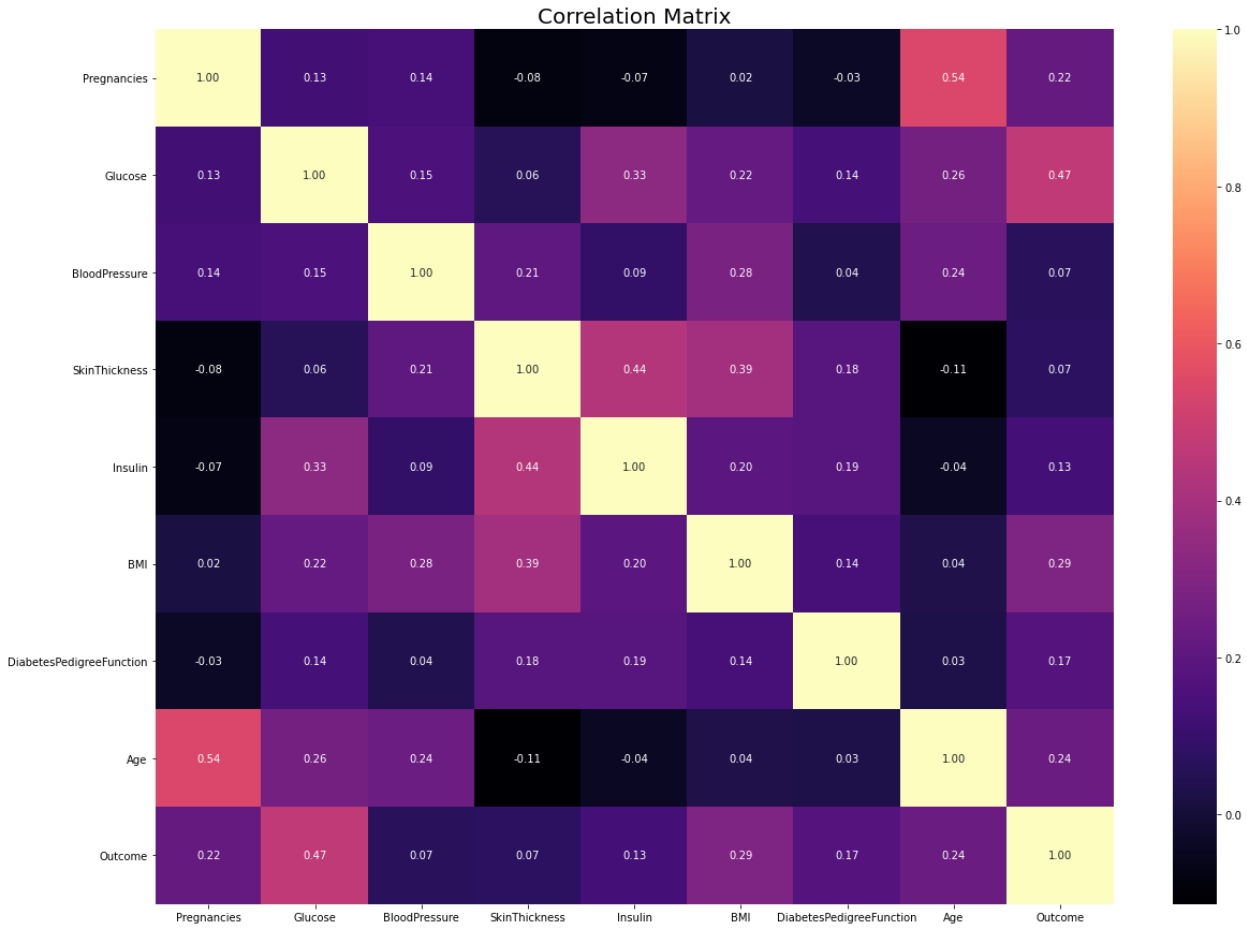
0.074752





Correlation matrix graph of the data set

F, ax = plt.subplots(figsize= [20,15])



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

Ax.set_title("Correlation Matrix", fontsize=20)

Plt.show()

) Data Preprocessing

2.1) Missing Observation Analysis

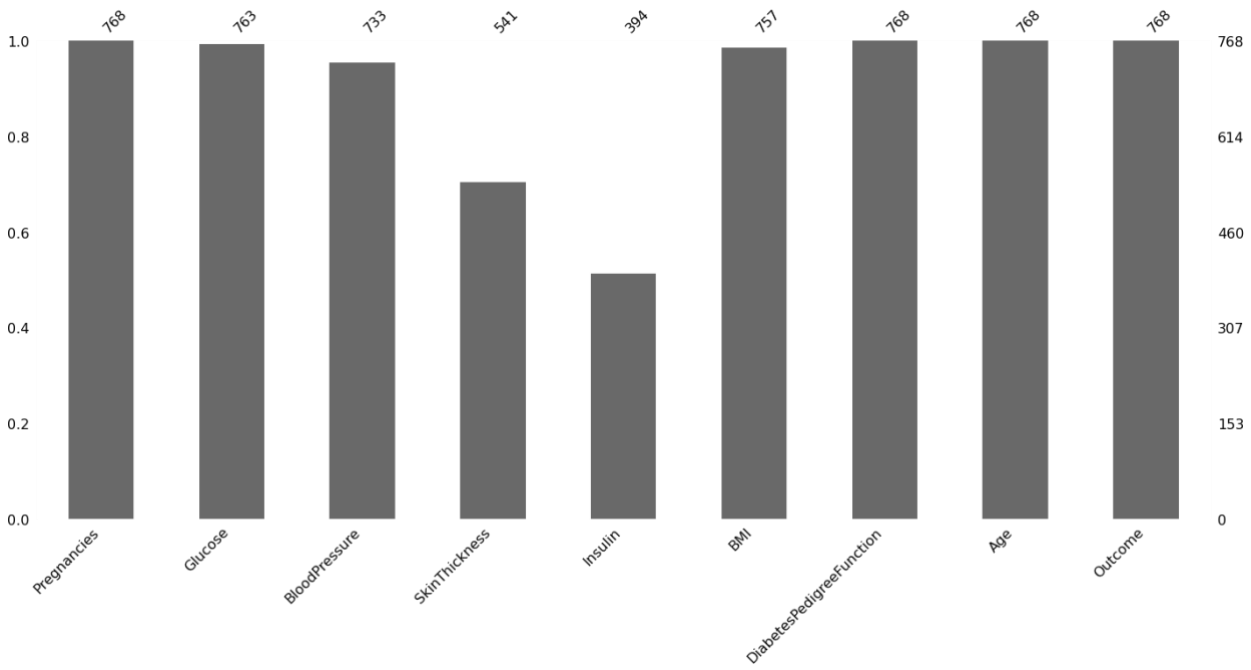
We saw on df.head() that some features contain 0, it doesn't make sense here and this indicates missing value Below we replace 0 value by NaN:

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

Df.head()

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

```
# Now, we can look at where are missing values
Df.isnull().sum()
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
Dtype: int64
# Have been visualized using the missingno library for the visualization of missing observations.
# Plotting
Import missingno as msno
Msno.bar(df);
```



The missing values will be filled with the median values of each variable.

```
Def median_target(var):
    Temp = df[df[var].notnull()]
    Temp = temp[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset_index()
    Return temp
```

The values to be given for incomplete observations are given the median value of people who are not sick and the median values of people who are sick.

```
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()
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0

2	8	183.0	64.0	32.0	169.5	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

Missing values were filled.

```
Df.isnull().sum()

Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
Dtype: int64
```

2.2) Outlier Observation Analysis¶

In the data set, there were asked whether there were any outlier observations compared to the 25% and 75% quarters.

It was found to be an outlier observation.

```
For feature in df:
Q1 = df[feature].quantile(0.25)
   Q3 = df[feature].quantile(0.75)
   IQR = Q3-Q1
   Lower = Q1- 1.5*IQR
   Upper = Q3 + 1.5*IQR
   If df[(df[feature] > upper)].any(axis=None):
      Print(feature,"yes")
   Else:
      Print(feature, "no")
```

Pregnancies yes

Glucose no

BloodPressure yes

SkinThickness yes

Insulin yes

BMI yes

DiabetesPedigreeFunction yes

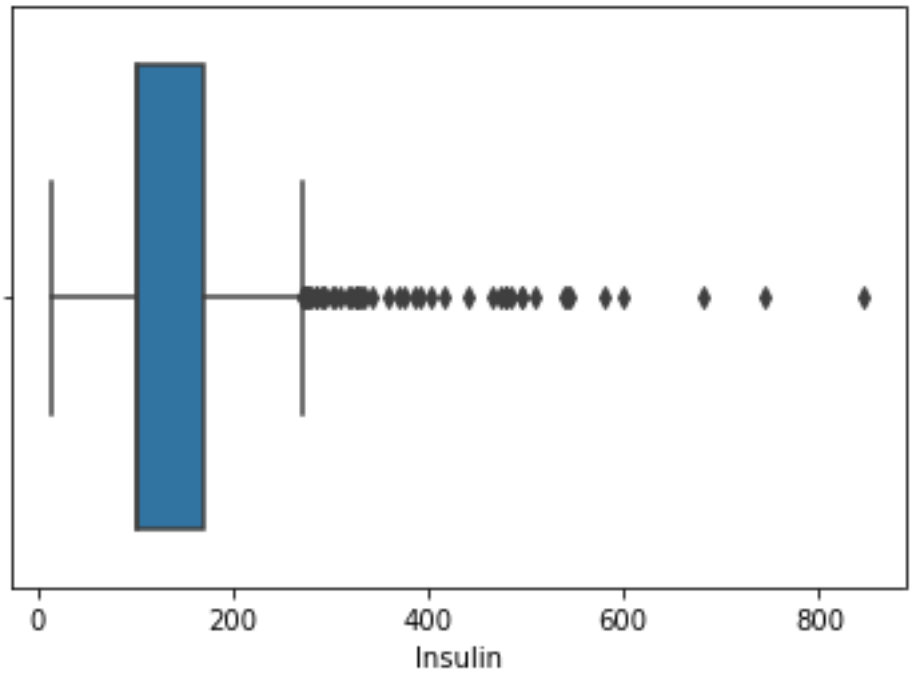
Age yes

Outcome no

The process of visualizing the Insulin variable with boxplot method was done. We find the outlier observations on the chart.

Import seaborn as sns

Sns.boxplot(x = df[“Insulin”]);



We conduct a stand alone observation review for the Insulin variable

#We suppress contradictory values

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”]);

LOF)¶

# We determine outliers between all variables with the LOF method

From sklearn.neighbors import LocalOutlierFactor

Lof =LocalOutlierFactor(n_neighbors= 10)

Lof.fit_predict(df)

Array([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1, -1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
```

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

-1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, -1,

```
Df_scores = lof.negative_outlier_factor_

Np.sort(df_scores)[0:30]

Array([-3.05893469, -2.37289269, -2.15297995, -2.09708735, -2.0772561 ,
       -1.95255968, -1.86384019, -1.74003158, -1.72703492, -1.71674689,
       -1.70343883, -1.6688722 , -1.64296768, -1.64190437, -1.61620872,
       -1.61369917, -1.60057603, -1.5988774 , -1.59608032, -1.57027568,
       -1.55876022, -1.55674614, -1.51852389, -1.50843907, -1.50280943,
       -1.50160698, -1.48391514, -1.4752983 , -1.4713427 , -1.47006248])

#We choose the threshold value according to lof scores

Threshold = np.sort(df_scores)[7]

Threshold

-1.740031580305444

#We delete those that are higher than the threshold

Outlier = df_scores > threshold

Df = df[outlier]

# The size of the data set was examined.

Df.shape

(760, 9)
```

2) Feature Engineering

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

```
# According to BMI, some ranges were determined and categorical variables were assigned.

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]

= 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	Outcome	NewBMI
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	Obesity 1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	Overweight
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	Normal
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	Overweight
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	Obesity 3

```
# A categorical variable creation process is performed according to the insulin value.

Def set_insulin(row):

   If row[“Insulin”] >= 16 and row[“Insulin”] <= 166:

      Return “Normal”

   Else:

      Return “Abnormal”

# The operation performed was added to the dataframe.

Df = df.assign(NewInsulinScore=df.apply(set_insulin, axis=1))


Df.head()
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	NewBMI	NewInsulinScore

0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	Obesity 1	Abnormal
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	Overweight	Normal
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	Normal	Abnormal
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	Overweight	Normal
4	0	137.0	40.0	35.0							

Some intervals were determined according to the glucose variable and these were assigned categorical variables.

```
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	Outcome	NewBMI	NewInsulinScore	NewGlucose
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	Obesity 1	Abnormal
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	Overweight	Normal
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	Normal	Abnormal
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	Overweight	Normal
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	Obesity 3	Abnormal

4) One Hot Encoding

Categorical variables in the data set should be converted into numerical values. For this reason, these transformation processes are performed with Label Encoding and One Hot Encoding method.

Here, by making One Hot Encoding transformation, categorical variables were converted into numerical values. It is also protected from the Dummy variable trap.

```
Df = pd.get_dummies(df, columns =["NewBMI","NewInsulinScore", "NewGlucose"], drop_first = True)
```

```
Df.head()
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	NewBMI_Obesity 1	NewBMI_Obesity 2	NewBMI_Obesity 3	NewBMI_Overweight	NewBMI_Underweight	NewInsulinScore_Normal	NewGlucose_Low	NewGlucose_Normal	NewGlucose_Overweight	NewGlucose_Secret
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	1	0	0	0	0	0	0	0	0
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	0	0	0	0	0	0	0	0	0
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	0	0	0	0	0	0	0	0	0
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	0	0	0	0	0	0	0	0	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	0	0	0	1	0	0	0	0	1

```
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']]
```

3.1) Final Model Installation

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

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

0.9013157894736843

```
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()

7) Comparison of Final Models

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 = []

For name, model in models:

) Reporting

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

1) Diabetes Data Set read.

2) With Exploratory Data Analysis; The data set’s structural data were checked. The types of variables in the dataset were examined. Size information of the dataset was accessed. The 0 values in the data set are missing values. Primarily these 0 values were replaced with NaN values. Descriptive statistics of the data set were examined.

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

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

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