Introduction to Data Science (S1-22_DSECLZG532)-ASSIGNMENT

Group No 023

Group Member Names:

1. KISHAN KUMAR ** (2022DA04047)

2. SANDEEP KUMAR**** (2022DA04055)

3. MANESH KUMAR**** (2022DA04073)

4. O SATEESH KUMAR** (2022DA04116)

1. Business Understanding

Students are expected to identify a data analytics task of your choice. You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?
2. What data do you need to answer the above problem?
3. What are the different sources of data?
4. What kind of analytics task are you performing?

Score: 1 Mark in total (0.25 mark each)

```
--------------Type the answers below this line--------------
```

1.1 What is the business problem that you are trying to solve?

The business problem that we are trying to solve about diagnostically forecast if a patient has diabetes using the important diagnostic metrics contained in the data.

1.2 What data do you need to answer the above problem?

We need datasets that are made up of a variety of medical predictor factors as well as one goal variable called Outcome.

1.3 What are the different sources of data?

We took this dataset from kaggle site. Link => https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database (https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database) However, the original source for this dataset is the National Institute of Diabetes and Digestive and Kidney Diseases.These instances were chosen under a number of constraints from a larger database. In particular, Pima Indigenous women who are at least 21 years old make up the majority of the patients at this clinic.

1.4 What kind of analytics task are you performing?

We are performing below mentioned tasks :-

1. Collection (download) of Dataset and converting it into Dataframe
2. Display the column headings, statistical information, description and statistical summary of the data
3. Data Preparation :- Check for duplicate data , missing data and data inconsistencies
4. Data Preparation :- Apply techiniques to remove duplicate data , to impute or remove missing data and to remove data inconsistencies
5. Data Preparation :- Identify the target variables
6. Data Exploration using various plots
7. Data Wrangling
8. Report observations
9. Implement Machine Learning Techniques
10. Conclusions & Solutions

2. Data Acquisition

For the problem identified , find an appropriate data set (Your data set must be unique) from any public data source.

2.1 Download the data directly

In [197]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import opendatasets as od
%matplotlib inline
```

In [196]:

```
## using opendatasets library to download the dataset from below kaggle link
## ! pip install opendatasets
```

In [2]:

```
od.download("https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database")
```

```
Skipping, found downloaded files in ".\pima-indians-diabetes-database" (us
e force=True to force download)
```

2.2 Code for converting the above downloaded data into a dataframe

In [198]:

```
data=pd.read_csv("diabetes.csv")
```

In [199]:

```
data.head()
```

Out[199]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2

2.3 Confirm the data has been downloaded correctly by displaying the first 5 and last 5 records.

In [200]:

```
data.head()
```

Out[200]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2

In [201]:

```
data.tail()
```

Out[201]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

2.4 Display the column headings, statistical information, description and statistical summary of the data.

In [202]:

```
pd.DataFrame(data.describe(include="all"))
```

Out[202]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

2.5 Write your observations from the above.

1. Size of the dataset
2. What type of data attributes are there?
3. Is there any null data that has to be cleaned?

Score: 2 Marks in total (0.25 marks for 2.1, 0.25 marks for 2.2, 0.5 marks for 2.3, 0.25 marks for 2.4, 0.75 marks for 2.5)

```
--------------Type the answers below this line--------------
```

1. Size of the dataset

In [203]:

```
data.shape
```

Out[203]:

```
(768, 9)
```

- This dataset have total 768 records and 9 features in each record

In [204]:

```
print(f"Toatal No of rows in dataset is {data.shape[0]} and total number of columns are {
```

```
Toatal No of rows in dataset is 768 and total number of columns are 9
```

2. What type of data attributes are there?

In [205]:

```
data.dtypes
```

Out[205]:

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

- We have integer & float datatypes in the dataset

3. Is there any null data that has to be cleaned?

In [206]:

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

Out[206]:

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

- No null data in the dataset

3. Data Preparation

```
If input data is numerical or categorical, do 3.1, 3.2 and 3.4
If input data is text, do 3.3 and 3.4
```

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

In [207]:

```
##---------Type the code below this line-----------------##
```

checking duplicate data

In [208]:

```
data.duplicated().any()
```

Out[208]:

```
False
```

- No duplicate data in the dataset

checking missing data

In [209]:

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

Out[209]:

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

- No null data in the dataset

checking data inconsistencies

In [210]:

```
data.dtypes
```

Out[210]:

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

- We have integer & float datatypes in the dataset

In [211]:

```
data.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
```

- We have total 768 entries (0 to 767 index for records/rows)
- Each record/row having 9 attributes(columns)
- All records don't have any null and missing entry
- Out of 9 attributes 7 are integer and 2 are floating numbers.

Get all numerical coulumns

In [212]:

```
numerical = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']

numerical_column=[i for i in data.columns if str(data[i].dtype) in (numerical)]
```

Basic statistic details about the data (note only numerical columns would be displayed here unless parameter include="all")

In [213]:

```
data.describe().T
```

Out[213]:

	count	mean	std	min	25%	50%	75
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.000
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.250
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.000
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.000
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.250
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.600
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.626
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.000
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.000

- we can see for some columns they have value zero(0) ,But as to the features that doesnt make sense. Lets check out the columns which have zero but it shouldnt be there

In [214]:

```
data.isin([0]).sum()
```

Out[214]:

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

```
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
```

- > These column cant be zero we will try to impute it wherever those are zero. Best way is to do that to replace it with np.Nan

In [215]:

```
data_copy= data.copy(deep=True)
```

In [216]:

```
invalid_zero_cols= ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','BMI']

for col in invalid_zero_cols:
    data_copy[col]= data_copy[col].replace(0,np.NaN)
```

In [217]:

```
data_copy.isna().sum()
```

Out[217]:

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [218]:

```
data_withna= data_copy.copy(deep=True)
```

3.2 Apply techiniques

- to remove duplicate data
- to impute or remove missing data

- to remove data inconsistencies

In [219]:

```
##---------Type the code below this line-----------------##
```

To remove duplicate data

In [220]:

```
data_withna=data_withna.drop_duplicates()
```

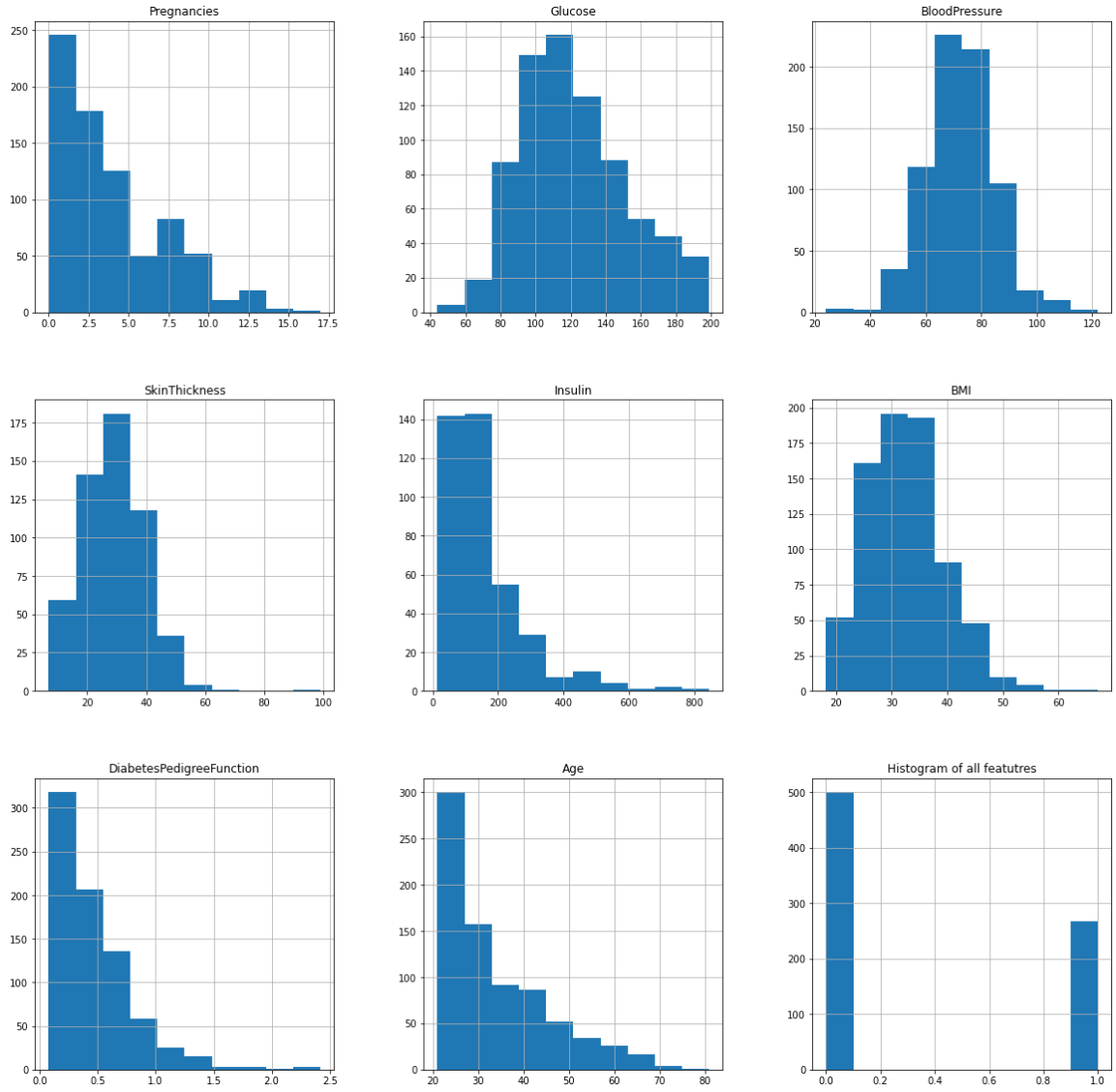
To impute or remove missing data

To fill these Nan values the data distribution needs to be understood

A better way of realizing this missingness is by visualizing the same using missingno package by drawing a nullity matrix. And as we can observe

In [221]:

```
p=data_withna.hist(figsize=(20,20))
plt.title("Histogram of all featutres");
```



For Column Glucose => check the distributuion

- For glucose column - The dustribution of Glucose seems to be normal. We know Glucose column has std as 32 (approx). Let try to impute with mean/median and see if the std changes remarkably

In [222]:

```
glucose_without_zero= data_copy["Glucose"].fillna(data_copy["Glucose"].mean())
```

In [223]:

```
glucose_without_zero.std()
```

Out[223]:

```
30.43594886720766
```

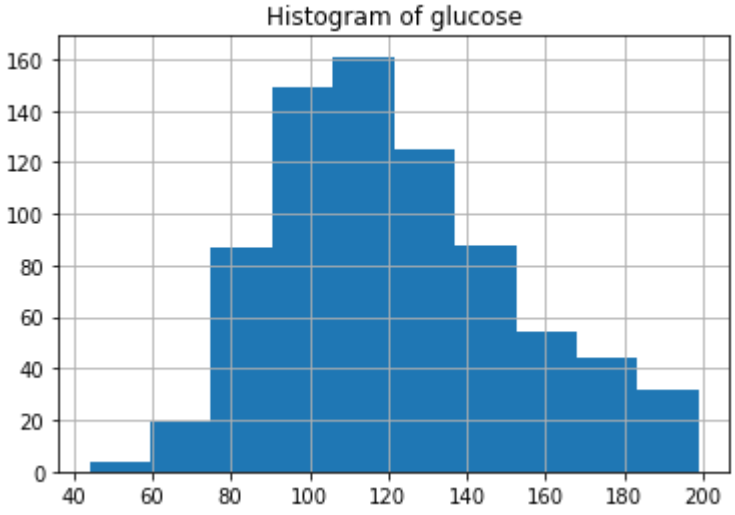
- We can see that std moved to just -1.5 SO we can consider it

In [224]:

```
glucose_without_zero.hist()
plt.title("Histogram of glucose_without_zero");
```

In [225]:

```
data_copy["Glucose"].hist()
plt.title("Histogram of glucose");
```



In [226]:

```
data_copy["Glucose"]=glucose_without_zero
```

- Not much difference after imputing median so we can change it

In [227]:

```
data_copy.isnull().sum()
```

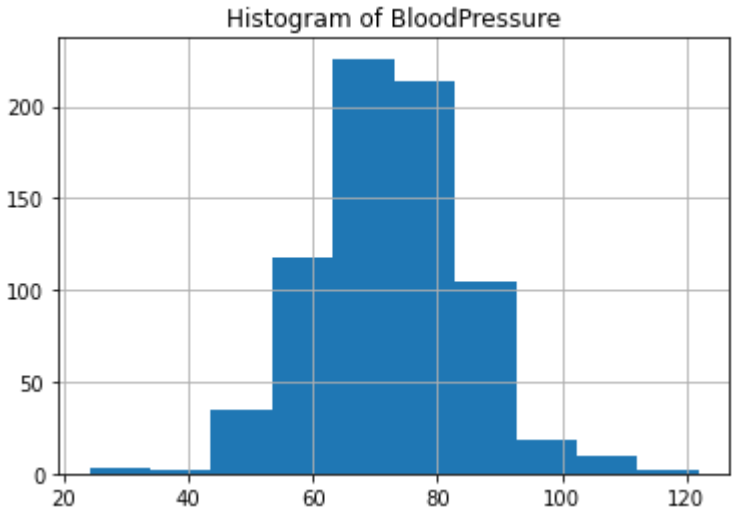
Out[227]:

```
Pregnancies                 0
Glucose                     0
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

For Column BloodPressure => check the distributuion

In [228]:

```
data_copy["BloodPressure"].hist()
plt.title("Histogram of BloodPressure");
```



- For BloodPressure column - The dustribution of BloodPressure seems to be normal. We know Glucose column has std as 12.38 (approx). Let try to impute with mean/median and see if the std changes remarkably

In [229]:

```
BloodPressure_without_zero= data_copy["BloodPressure"].fillna(data_copy["BloodPressure"].
```

In [230]:

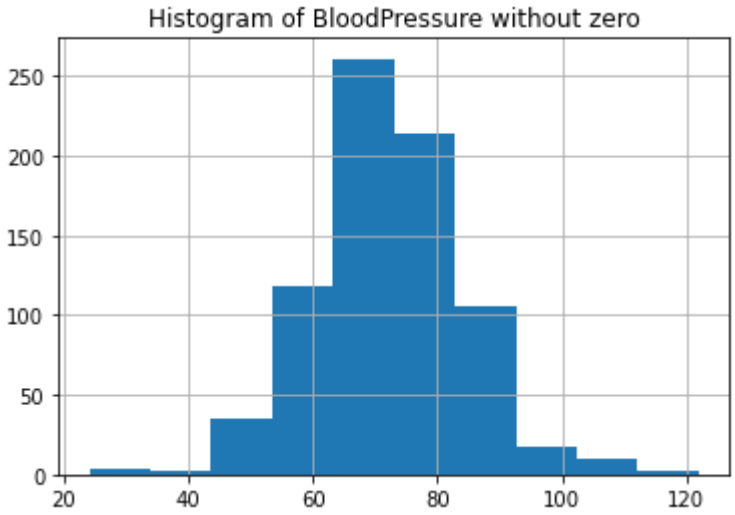
```
BloodPressure_without_zero.std()
```

Out[230]:

```
12.096346184037948
```

In [231]:

```
BloodPressure_without_zero.hist()
plt.title("Histogram of BloodPressure without zero");
```



- Its good , So just replace it

In [232]:

```
data_copy["BloodPressure"]=BloodPressure_without_zero
```

In [233]:

```
data_copy.isnull().sum()
```

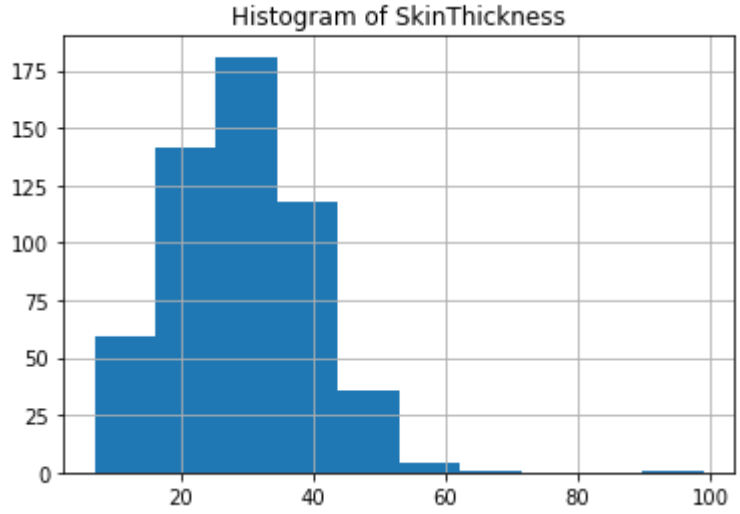
Out[233]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

For Column SkinThickness => check the distributuion

In [234]:

```
data_copy["SkinThickness"].hist()
plt.title("Histogram of SkinThickness");
```



- Its almost Normal. Lets check how much its normal with help of ggplot()

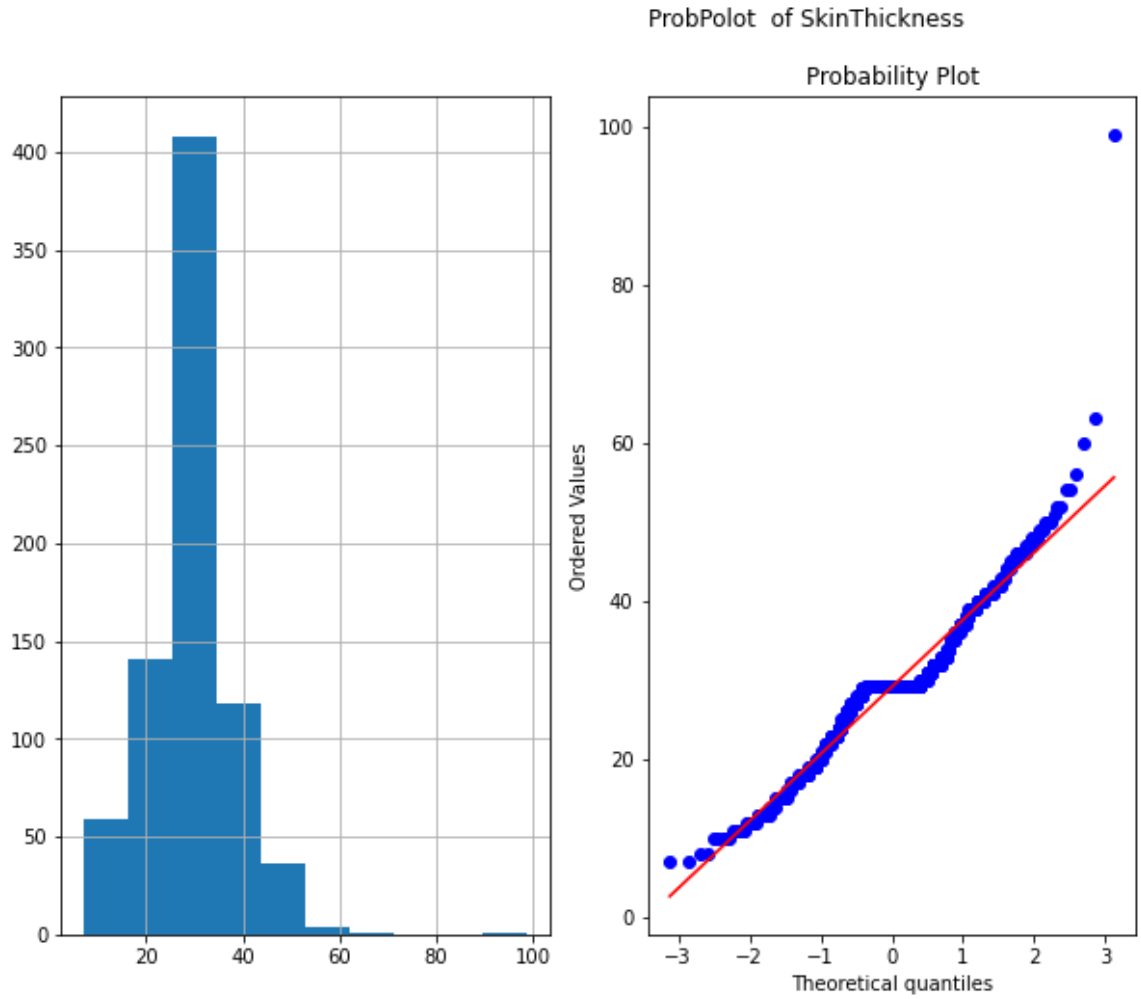
In [290]:

```
import scipy.stats as stats
import pylab

def plot_qqplot(df,feature):
    """
    function to create probplot of a feature,
    here we are trying to see if the feature is
    normally distributed or not
    """
    plt.figure(figsize=(10,8))
    plt.subplot(1,2,1)
    df[feature].hist()
    plt.subplot(1,2,2)
    stats.probplot(df[feature],dist='norm',plot=pylab)
    plt.title(f"ProbPolot  of {feature} \n \n",loc='left');
    plt.show()
```

In [291]:

```
plot_qqplot(data_copy,"SkinThickness")
#plt.title("ProbPolot  of SkinThickness");
```



- Its good as it is almost nornamlly distrubuted So we can go on and replace null with the median and check again the standard distribution as 10.95

In [256]:

```
SkinThickness_without_zero = data_copy["SkinThickness"].fillna(data_copy["SkinThickness"]
```

In [257]:

```
SkinThickness_without_zero.std()
```

Out[257]:

```
8.79094192562453
```

In [258]:

```
data_copy["SkinThickness"]= SkinThickness_without_zero
data_copy.isna().sum()
```

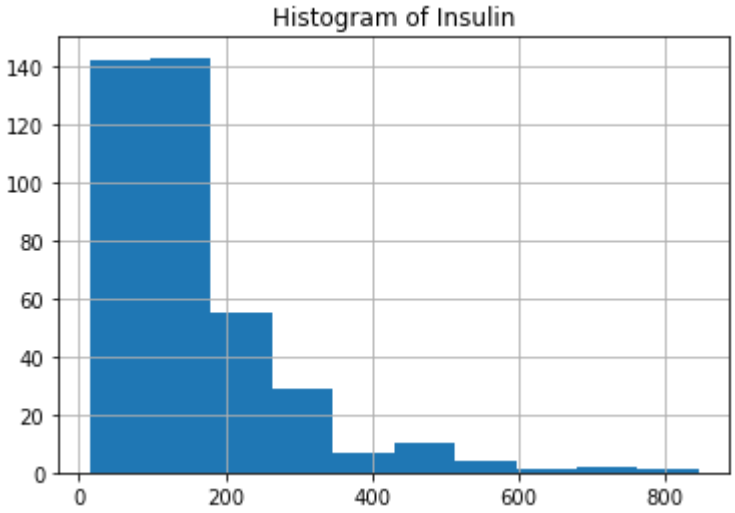
Out[258]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

For Column Insulin => check the distributuion

In [259]:

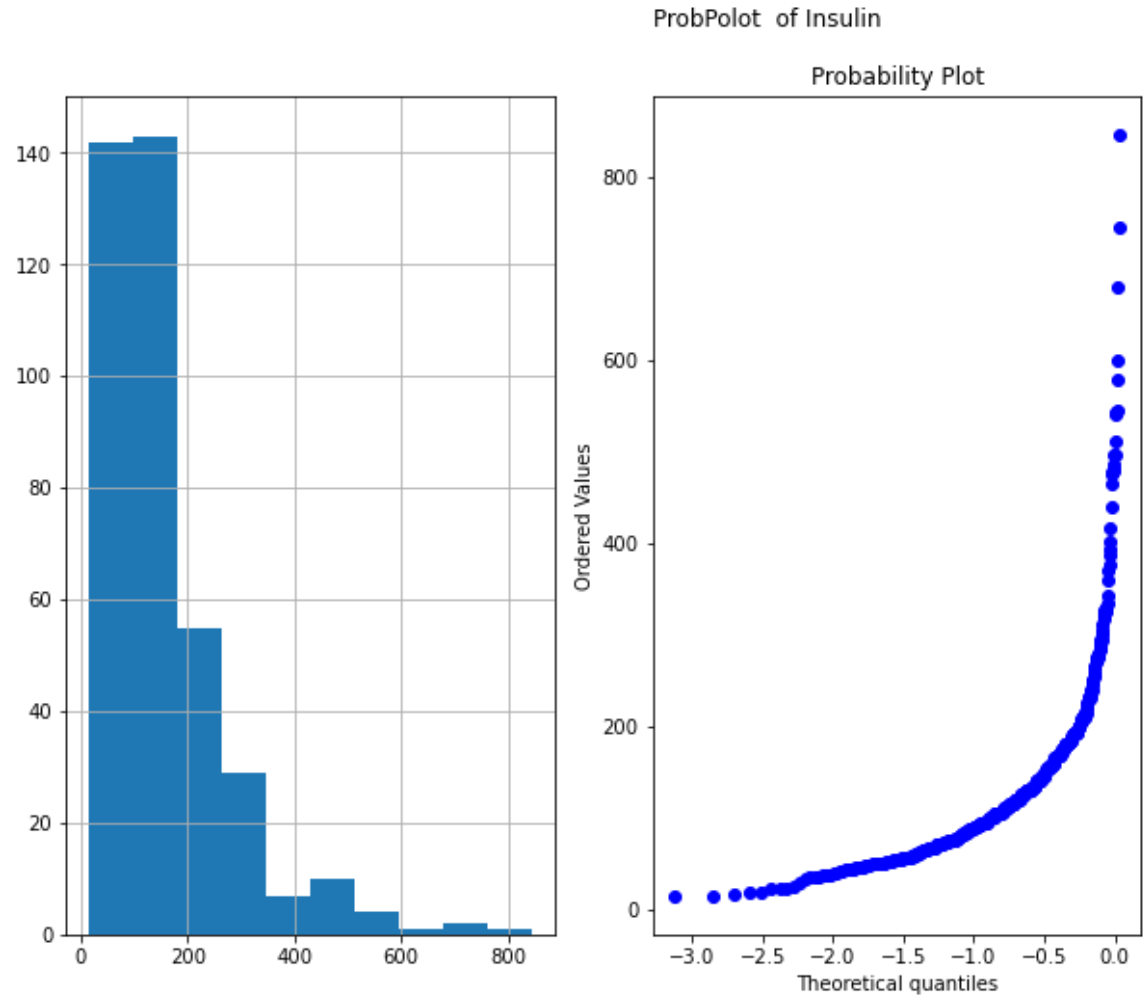
```
data_copy["Insulin"].hist()
plt.title("Histogram of Insulin");
```



- Its higly skewed

In [260]:

```
plot_qqplot(data_copy,"Insulin")
#plt.title("ProbPolot  of Insulin");
```



In [ ]:

```
## its not normally distributed but skewed
```

In [261]:

```
print(f"Std before imputing")
data_copy["Insulin"].std()
```

```
Std before imputing
```

Out[261]:

```
118.77585518724514
```

In [262]:

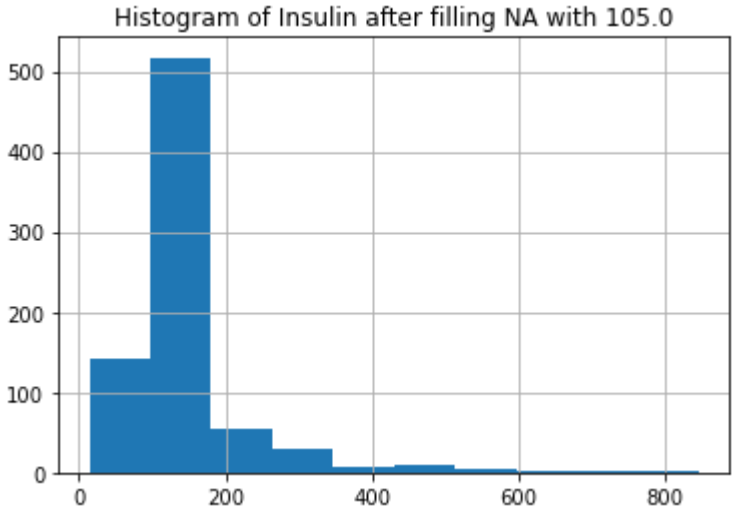
```
data_copy["Insulin"].fillna(105.0).std()
```

Out[262]:

```
88.70044279323841
```

In [263]:

```
data_copy["Insulin"].fillna(105.0).hist()
plt.title("Histogram of Insulin after filling NA with 105.0");
```



In [264]:

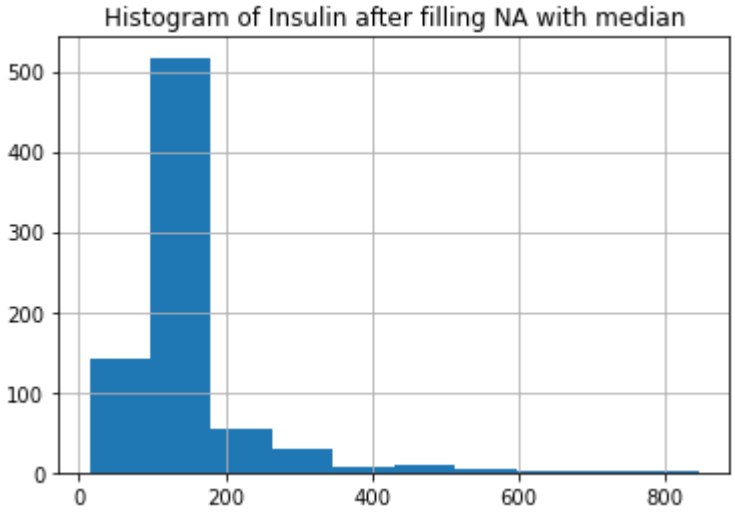
```
data_copy["Insulin"].value_counts()
```

Out[264]:

```
105.0    11
130.0     9
140.0     9
120.0     8
94.0      7
         ..
73.0      1
171.0     1
255.0     1
52.0      1
112.0     1
Name: Insulin, Length: 185, dtype: int64
```

In [265]:

```
data_copy["Insulin"].fillna(data_copy["Insulin"].median()).hist()
plt.title("Histogram of Insulin after filling NA with median");
```



In [266]:

```
data_copy["Insulin"].fillna(data_copy["Insulin"].median(),inplace=True)
```

In [267]:

```
data_copy.isnull().sum()
```

Out[267]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

For Column BMI => check the distributuion

In [268]:

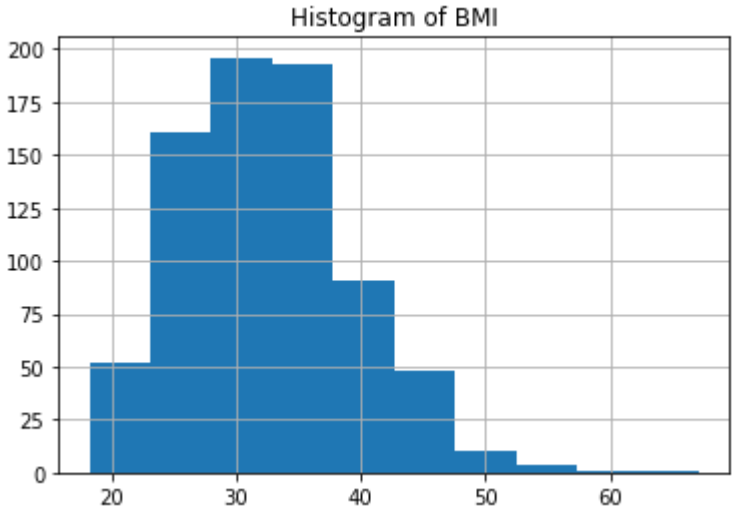
```
data_copy["BMI"].std()
```

Out[268]:

```
6.924988332105907
```

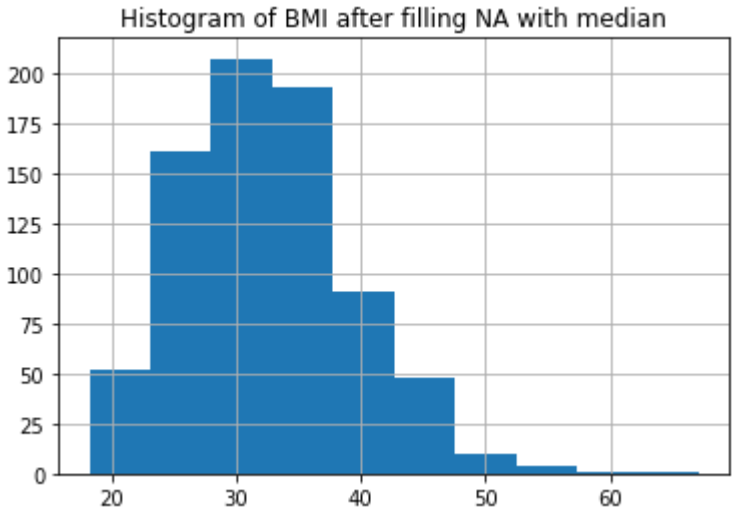
In [269]:

```
data_copy["BMI"].hist()
plt.title("Histogram of BMI");
```



In [270]:

```
data_copy["BMI"].fillna(data_copy["BMI"].median()).hist()
plt.title("Histogram of BMI after filling NA with median");
```



In [271]:

```
data_copy["BMI"].fillna(data_copy["BMI"].median()).std()
```

Out[271]:

```
6.875176818080996
```

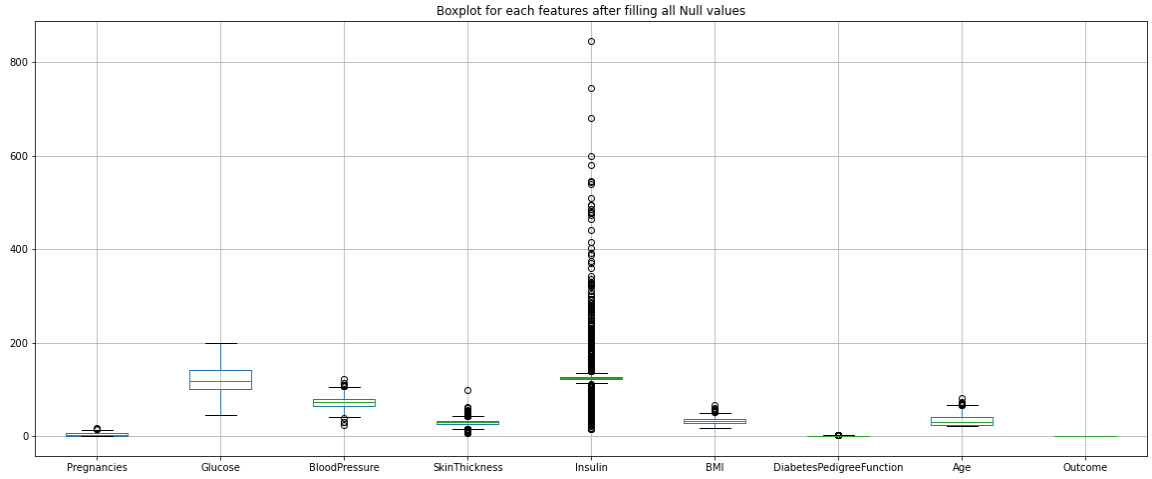
In [272]:

```
data_copy["BMI"].fillna(data_copy["BMI"].median(),inplace=True)
```

Check the boxplot

In [273]:

```
data_copy.boxplot(figsize=(20,8))
plt.title("Boxplot for each features after filling all Null values");
```



In [274]:

```
data_copy.isnull().sum()
```

Out[274]:

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

3.3 Encode categorical data

In [275]:

```
##---------Type the code below this line-----------------##
```

- No categorical data

3.4 Text data

1. Remove special characters
2. Change the case (up-casing and down-casing).
3. Tokenization — process of discretizing words within a document.

4. Filter Stop Words.

In [276]:

```
##---------Type the code below this line-----------------##
```

- No text data

3.4 Report

Mention and justify the method adopted

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present

OR for textdata

- How many tokens after step 3?
- how may tokens after stop words filtering?

If the any of the above are not present, then also add in the report below.

Score: 2 Marks (based on the dataset you have, the data prepreation you had to do and report typed, marks will be distributed between 3.1, 3.2, 3.3 and 3.4)

In [277]:

```
##---------Type the code below this line-----------------##
```

In [278]:

```
##---------Type the code below this line-----------------##
```

In [ ]:

```

```

3.5 Identify the target variables.

- Separate the data from the target such that the dataset is in the form of (X,y) or (Features, Label)
- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.
- Report the observations

Score: 1 Mark

In [279]:

```
#Features
X= data_copy.drop('Outcome',axis=1)
#Label
y=data_copy['Outcome']
```

4. Data Exploration using various plots

4.1 Scatter plot of each quantitative attribute with the target.

Score: 1 Mark

In [280]:

```
##---------Type the code below this line-----------------##
```

In [281]:

```
X.columns
```

Out[281]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age'],
      dtype='object')
```

In [282]:

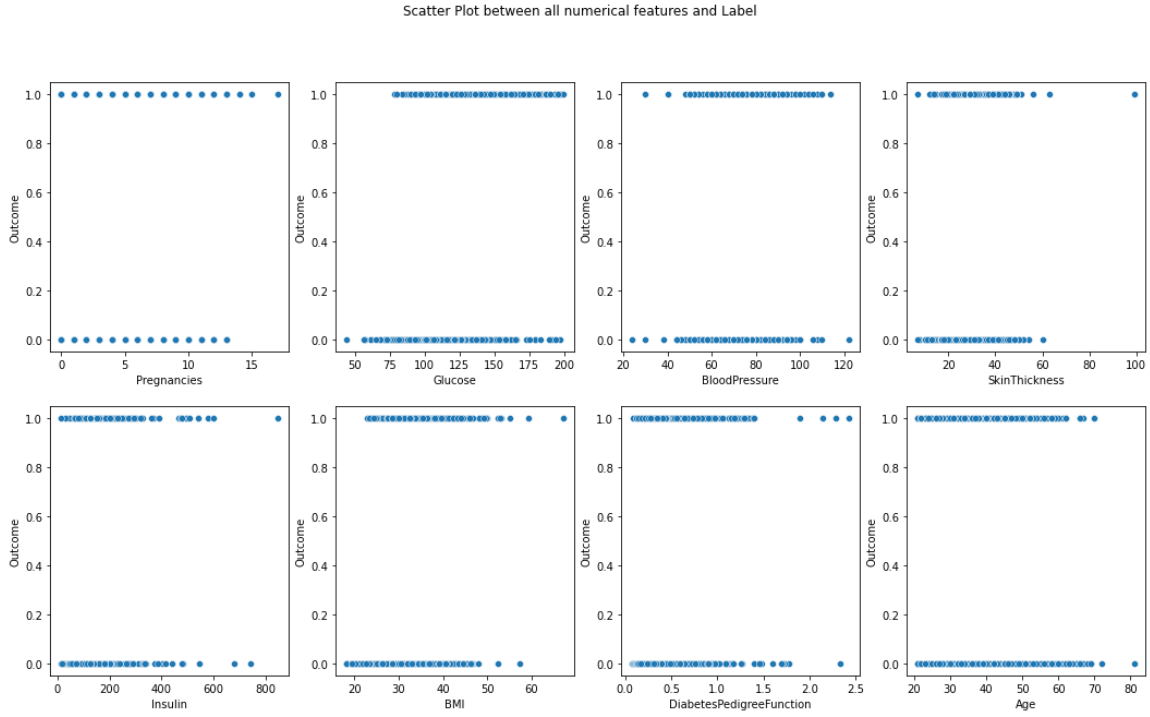
```
import seaborn as sns
fig, axes = plt.subplots(2, 4, figsize=(18, 10))

fig.suptitle('Scatter Plot between all numerical features and Label')

sns.scatterplot(ax=axes[0, 0], data=data_copy, x='Pregnancies', y='Outcome')
sns.scatterplot(ax=axes[0, 1], data=data_copy, x='Glucose', y='Outcome')
sns.scatterplot(ax=axes[0, 2], data=data_copy, x='BloodPressure', y='Outcome')
sns.scatterplot(ax=axes[0, 3], data=data_copy, x='SkinThickness', y='Outcome')
sns.scatterplot(ax=axes[1, 0], data=data_copy, x='Insulin', y='Outcome')
sns.scatterplot(ax=axes[1, 1], data=data_copy, x='BMI', y='Outcome')
sns.scatterplot(ax=axes[1, 2], data=data_copy, x='DiabetesPedigreeFunction', y='Outcome')
sns.scatterplot(ax=axes[1, 3], data=data_copy, x='Age', y='Outcome')
```

Out[282]:

```
<AxesSubplot:xlabel='Age', ylabel='Outcome'>
```



4.2 EDA using visuals

- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice of plots given, no credit will be awarded.

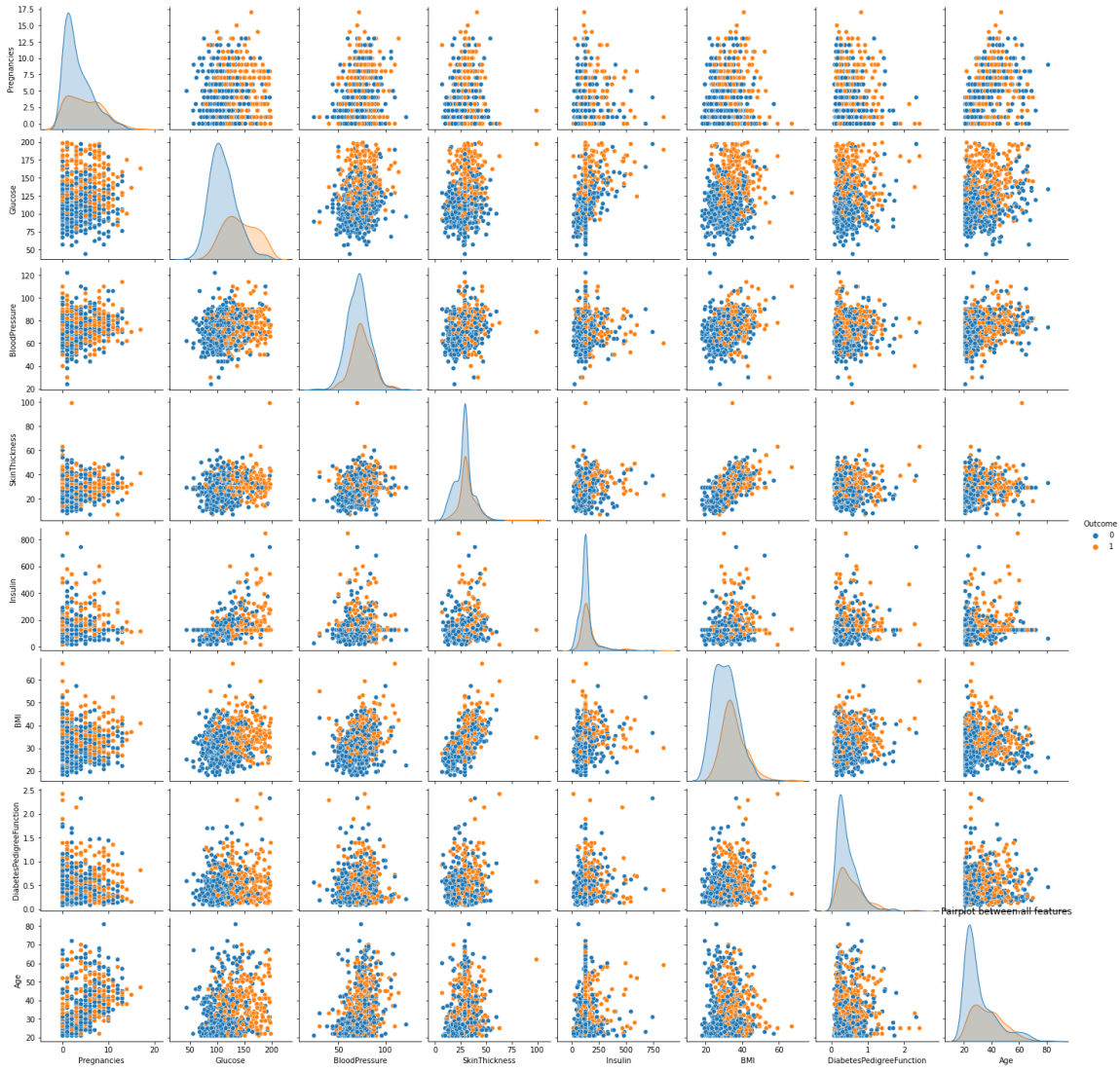
Score: 2 Marks

In [283]:

```
##---------Type the code below this line-----------------##
```

In [284]:

```
sns.pairplot(data_copy,hue="Outcome")
plt.title("Pairplot between all features");
```



Important points after EDA

- 1. All Features are continous
- 2. Non diabetic samples are more
- 3. we can see some positive correlation between some features,we can remove some columns.
- 4. Some data features are skewed we can perform some operations on that
- 5. posibilty of outliers,We can remove it.

Lets check about correlation between features

In [285]:

```
corr_matrix= data_copy.corr()
corr_matrix
```

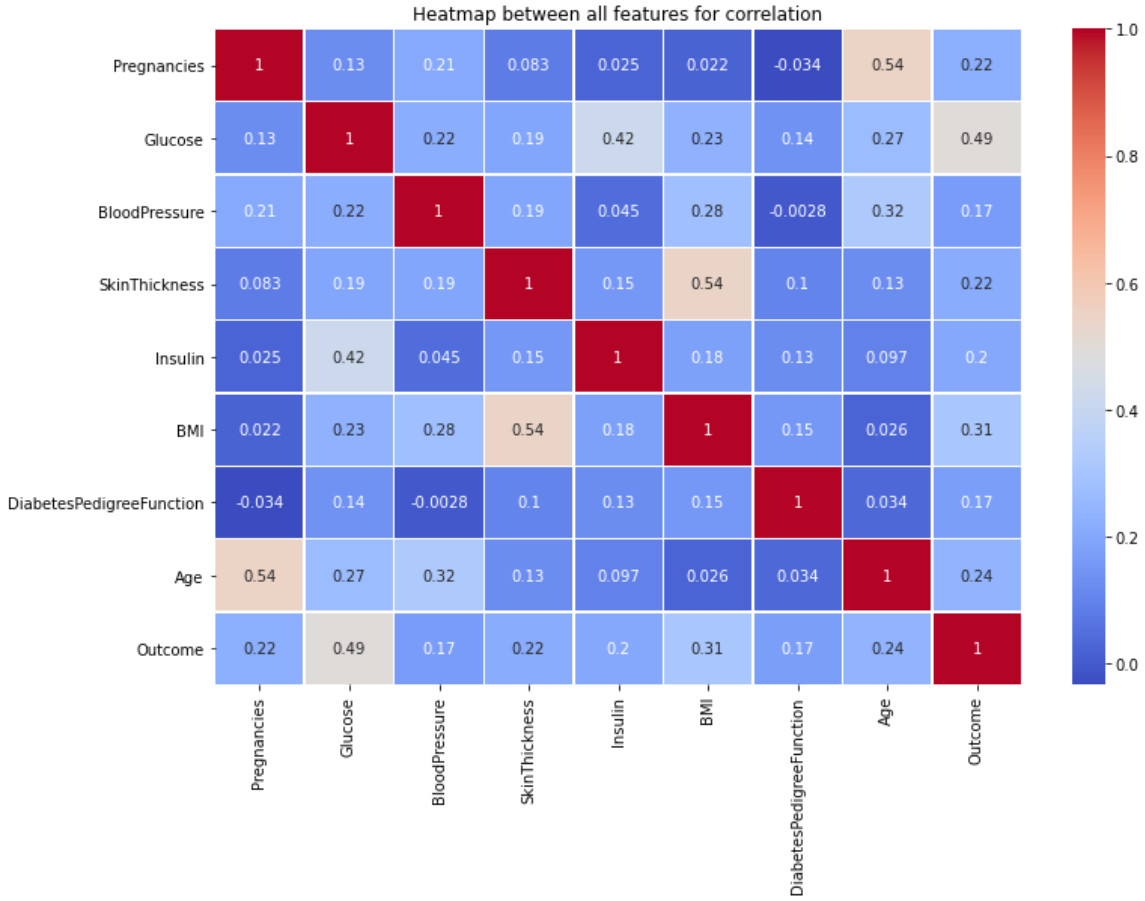
Out[285]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.127911	0.208522	0.082989	0.025047	0.
Glucose	0.127911	1.000000	0.218367	0.192991	0.419064	0.
BloodPressure	0.208522	0.218367	1.000000	0.192816	0.045087	0.
SkinThickness	0.082989	0.192991	0.192816	1.000000	0.154678	0.
Insulin	0.025047	0.419064	0.045087	0.154678	1.000000	0.
BMI	0.021559	0.231128	0.281199	0.542438	0.180241	1.
DiabetesPedigreeFunction	-0.033523	0.137060	-0.002763	0.100966	0.126503	0.
Age	0.544341	0.266534	0.324595	0.127872	0.097101	0.
Outcome	0.221898	0.492928	0.166074	0.215299	0.203790	0.

Generating heatmap to show correlation

In [286]:

```
plt.figure(figsize=(12,8))
sns.heatmap(corr_matrix,annot=True,linewidths=.5,cmap="coolwarm")
plt.title("Heatmap between all features for correlation");
```



In [287]:

```
corr_matrix["Outcome"].sort_values(ascending=False)
```

Out[287]:

```
Outcome                   1.000000
Glucose                   0.492928
BMI                       0.312038
Age                       0.238356
Pregnancies               0.221898
SkinThickness             0.215299
Insulin                   0.203790
DiabetesPedigreeFunction  0.173844
BloodPressure             0.166074
Name: Outcome, dtype: float64
```

5. Data Wrangling

5.1 Univariate Filters

Numerical and Categorical Data

- Identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring

1. Mutual Information (Information Gain)
2. Gini index
3. Gain Ratio
4. Chi-Squared test
5. Fisher Score (From the above 5 you are required to use only any **two**)

For Text data

1. Stemming / Lemmatization.
2. Forming n-grams and storing them in the document vector.
3. TF-IDF (From the above 2 you are required to use only any **two**)

Score: 3 Marks

In [288]:

```
##---------Type the code below this line-----------------##
```

5.1.1 We are using Mutual Information here.

Mutual Information :

- Estimate mutual information for a discrete target variable.
- Mutual information (MI) [1] between two random variables is a non-negative value,
- which measures the dependency between the variables.
- It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

In [173]:

```
from sklearn.feature_selection import mutual_info_classif as MIC
mi_score = MIC(X,y)
print(mi_score)
```

```
[0.00483776 0.13075605 0.00425347 0.00752573 0.03905162 0.08049082
 0.01302233 0.06458286]
```

In [174]:

```
mi_score=pd.Series(mi_score)
mi_score.index=X.columns
mi_score
```

Out[174]:

```
Pregnancies                 0.004838
Glucose                     0.130756
BloodPressure               0.004253
SkinThickness               0.007526
Insulin                     0.039052
BMI                         0.080491
DiabetesPedigreeFunction    0.013022
Age                         0.064583
dtype: float64
```

In [175]:

```
mi_score=mi_score.sort_values(ascending=False)
mi_score
```

Out[175]:

```
Glucose                     0.130756
BMI                         0.080491
Age                         0.064583
Insulin                     0.039052
DiabetesPedigreeFunction    0.013022
SkinThickness               0.007526
Pregnancies                 0.004838
BloodPressure               0.004253
dtype: float64
```

In [176]:

```
top_5byMI=list(mi_score[:5].index)
print("Top 5 features in decreasing order by Mutual information gain",top_5byMI)
```

```
Top 5 features in decreasing order by Mutual information gain ['Glucose',
'BMI', 'Age', 'Insulin', 'DiabetesPedigreeFunction']
```

In [ ]:

```

```

5.1.2 We are using chi square over here.

Chi square :

- Compute chi-squared stats between each non-negative feature and class.
- This score can be used to select the n_features features with

- the highest values for the test chi-squared statistic from X,
- which must contain only non-negative features such as booleans or frequencies (e.g., term counts in document classification),relative to the classes.

In [177]:

```
from sklearn.feature_selection import chi2
chi_score,p_values= chi2(X,y)
```

In [178]:

```
chi_score=pd.Series(chi_score,index=X.columns)
chi_score=chi_score.sort_values(ascending=False)
chi_score
```

Out[178]:

```
Insulin                     1689.711075
Glucose                     1418.705030
Age                          181.303689
Pregnancies                  111.519691
BMI                          108.766367
SkinThickness                 94.245703
BloodPressure                 42.749956
DiabetesPedigreeFunction       5.392682
dtype: float64
```

In [179]:

```
p_values=pd.Series(p_values,index=X.columns)
p_values=p_values.sort_values()
p_values
```

Out[179]:

```
Insulin                      0.000000e+00
Glucose                     1.810528e-310
Age                          2.516388e-41
Pregnancies                  4.552610e-26
BMI                          1.825876e-25
SkinThickness                2.786911e-22
BloodPressure                6.220327e-11
DiabetesPedigreeFunction     2.022137e-02
dtype: float64
```

In [180]:

```
best_5_byChisquare=p_values[:5].index
print("Top 5 features in decreasing order by Mutual information gain",best_5_byChisquare)
```

```
Top 5 features in decreasing order by Mutual information gain Index(['Insu
lin', 'Glucose', 'Age', 'Pregnancies', 'BMI'], dtype='object')
```

5.2 Report observations

Write your observations from the results of each method. Clearly justify your choice of the method.

Score 1 mark

In [181]:

```
##---------Type the code below this line-----------------##
```

6. Implement Machine Learning Techniques

Use any 2 ML algorithms

1. Classification -- Decision Tree classifier
2. Clustering -- kmeans
3. Association Analysis
4. Anomaly detection
5. Textual data -- Naive Bayes classifier (not taught in this course)

A clear justification have to be given for why a certain algorithm was chosen to address your problem.

Score: 4 Marks (2 marks each for each algorithm)

6.1 ML technique 1 + Justification

As a First technique we are using classifier which will implement a decision tree classifier.

In [182]:

```
y.value_counts()
```

Out[182]:

```
0    500
1    268
Name: Outcome, dtype: int64
```

The target seems to imbalanced and skewed towards zero class

Train test split

In [183]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,random_state=42)
```

In [184]:

```
y_test.value_counts()
```

Out[184]:

```
0    123
1     69
Name: Outcome, dtype: int64
```

Using a standard scaler to sclae the numerical values (not generaly neccesary for Tree based classifier)

In [185]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train_scaled= scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

Using Decision Tree classifier

In [186]:

```
from sklearn.tree import DecisionTreeClassifier
dtree= DecisionTreeClassifier(max_depth=8,random_state=42)
dtree.fit(X_train_scaled,y_train)
y_pred=dtree.predict(X_test_scaled)
```

In [187]:

```
from sklearn.metrics import roc_auc_score,accuracy_score,precision_score,recall_score,f1_

print(f"For decision Tree Classifier  \n \n accuracy_score :{accuracy_score(y_test,y_pred
precision_score :{precision_score(y_test,y_pred)} \n recall_score :{recall_score(y_test,y
```

```
For decision Tree Classifier

 accuracy_score :0.7083333333333334
 roc_auc_score :0.6800989749027925
 precision_score :0.5970149253731343
 recall_score :0.5797101449275363
 f1_score :0.5882352941176471
```

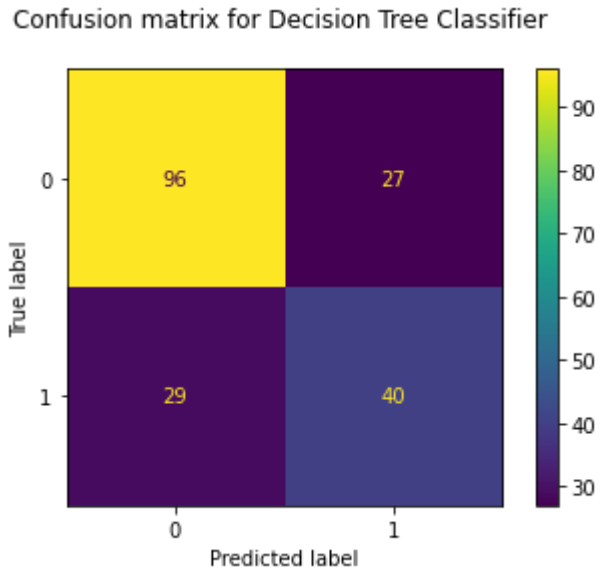
In [188]:

```
from sklearn.metrics import plot_confusion_matrix,roc_curve,auc

plot_confusion_matrix(dtree,X_test_scaled,y_test,display_labels=[0,1])
plt.title("Confusion matrix for Decision Tree Classifier \n")
```

Out[188]:

```
Text(0.5, 1.0, 'Confusion matrix for Decision Tree Classifier \n')
```



6.2 ML technique 2 + Justification

In [189]:

```
##---------Type the code below this line-----------------##
```

As a Second technique we are using K-Nearest Neighbour classifier

In [190]:

```
##---------Type the code below this line-----------------##


## using  k-nearest neighbors classifier

from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled,y_train)
y_pred=knn.predict(X_test_scaled)
```

In [191]:

```
from sklearn.metrics import roc_auc_score,accuracy_score,precision_score,recall_score

print(f"For KNeighborsClassifier  \n \n accuracy_score :{accuracy_score(y_test,y_pred)}\r
precision_score :{precision_score(y_test,y_pred)} \n recall_score :{recall_score(y_test,y
```

```
For KNeighborsClassifier

 accuracy_score :0.6822916666666666
 roc_auc_score :0.6597737716507599
 precision_score :0.5555555555555556
 recall_score :0.5797101449275363
 f1_score :0.5673758865248227
```

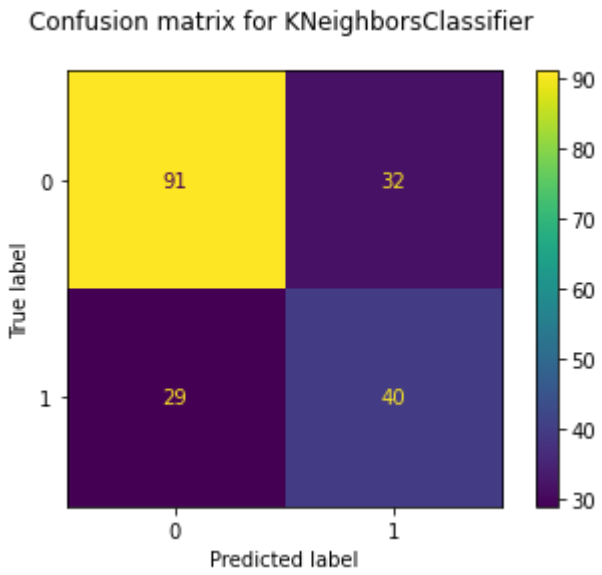
In [192]:

```
from sklearn.metrics import plot_confusion_matrix,roc_curve,auc

plot_confusion_matrix(knn,X_test_scaled,y_test,display_labels=[0,1])
plt.title("Confusion matrix for KNeighborsClassifier \n")
```

Out[192]:

```
Text(0.5, 1.0, 'Confusion matrix for KNeighborsClassifier \n')
```



7. Conclusion

Compare the performance of the ML techniques used.

Derive values for preformance study metrics like accuracy, precision, recall, F1 Score, AUC-ROC etc to compare the ML algos and plot them. A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision becomes authentic. You may use Confusion matrix, classification report, Word cloud etc as per the requirement of your application/problem.

Score 1 Mark

In [193]:

```
##---------Type the code below this line-----------------##
```

In [194]:

```
#! pip install prettytable
```

Comparing the matrices of the 2 ML technique we see that k-nearest neighbors gives better result

In [289]:

```
## SHowing and comparing result in a nice manner with pretty table library
from prettytable import PrettyTable

# Specify the Column Names while initializing the Table
metrics_comparison = PrettyTable(["metrics","DecisionTreeClassifier","KNeighborsClassifie

# Add rows
metrics_comparison.add_row(["accuracy_score", "0.666", "0.682"])
metrics_comparison.add_row(["roc_auc_score", "0.644", "0.659"])
metrics_comparison.add_row(["precision_score", "0.534", "0.555"])
metrics_comparison.add_row(["recall_score", "0.565", "0.579"])
metrics_comparison.add_row(["f1_score", "0.549", "0.567"])
print(metrics_comparison)
```

```
+-----------------+------------------------+----------------------+
|     metrics     | DecisionTreeClassifier | KNeighborsClassifier |
+-----------------+------------------------+----------------------+
|  accuracy_score |         0.666          |        0.682         |
|  roc_auc_score  |         0.644          |        0.659         |
| precision_score |         0.534          |        0.555         |
|   recall_score  |         0.565          |        0.579         |
|     f1_score    |         0.549          |        0.567         |
+-----------------+------------------------+----------------------+
```

8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

----------Type the answers below this line----------

- Checked for data duplicates, missing data and data inconsistencies.
  - Removed duplicate data
  - Ensured all data fields have right data type
  - Implemented wise data imputation as we see null values for some fields by visualizing and understanding the data distribution. Imputed with mean/median based on the data distribution.
  - Target variables are selected for the data.
- Performed efficient EDA to familiaize ourrselves with Data, extracting useful insights. As part of this,

    - We looked at various statstical details of each feature suach as **mean, std, quantiles, min and max values**.
    - Plotted *pairwise scatter plot, heat and correlations plots* to identify correlation between features, identify skewness and possiblity of outliers.
- Used Data wrangling techniques **Mutual Information gain** and **Chi Square Test** to find dependency between variables and to find top n-features respectively.
- Implemented **Decision Tree** and **KNN Classifier** models to with right train/test split,with right hyper paraeters(after tunig) and with various performance metrics.
- Compared performance metrics ***Acuracy, ROC_AUC, Precision, Recall*** and ***F1-Score*** for both models and it turned out that KNN Clasifier gives better perfromance.

- **We can see here the more important label is 1 or is diabetic**,
- **because if a patient having dibatese and predicted non-diabetic(False Negative) is more fatal than a patient not having dibatese but predicted diabetic(False Positive).So Recall score is more relavant than , Accurcay score and F1 score.***

Here KNN classifier has performed better than Decision tree classifier, With implementing just two classifier we can conclude that KNN classifier could be a better solution.

In [ ]:

##NOTE All Late Submissions will incur a penalty of -2 marks. Do ensure on time submission to avoid penalty.

Good Luck!!!