Regression Analysis: Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following:

a. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis

b. Bivariate analysis: Linear and logistic regression modeling

c. Multiple Regression analysis

d. Also compare the results of the above analysis for the two data sets Dataset link: https://www.kaggle.com/datasets/uciml/pimaindians-diabetes-database

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import r2_score,mean_squared_error,accuracy_score
import warnings
warnings.filterwarnings("ignore")

df=pd.read_csv(r"C:\Users\dell\Desktop\DMV and ML\ML Datasets\
diabetes.csv")

df
```

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

     DiabetesPedigreeFunction  Age  Outcome
```

```
0                        0.627   50        1
1                        0.351   31        0
2                        0.672   32        1
3                        0.167   21        0
4                        2.288   33        1
..                         ...  ...      ...
763                      0.171   63        0
764                      0.340   27        0
765                      0.245   30        0
766                      0.349   47        1
767                      0.315   23        0

[768 rows x 9 columns]
```

```
df.describe()
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness
Insulin  \
count   768.000000  768.000000     768.000000     768.000000
768.000000
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

```
# univariate_analysis = pd.DataFrame({
#     'Frequency': df.count(),
#     'Mean': df.mean(),
#     'Median': df.median(),
#     'Mode': df.mode().iloc[0],
```

```python
#     'Variance': df.var(),
#     'Standard Deviation': df.std(),
#     'Skewness': df.skew(),
#     'Kurtosis': df.kurt()
# }).T


# univariate_analysis

df.count()
```

```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

```python
df.skew()
```

```
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     2.272251
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
```

```python
df.kurt()
```

```
Pregnancies                 0.159220
Glucose                     0.640780
BloodPressure               5.180157
SkinThickness              -0.520072
Insulin                     7.214260
BMI                         3.290443
DiabetesPedigreeFunction    5.594954
Age                         0.643159
Outcome                    -1.600930
dtype: float64
```

```python
df.var()
```

```
Pregnancies                    11.354056
Glucose                      1022.248314
BloodPressure                 374.647271
SkinThickness                 254.473245
Insulin                     13281.180078
BMI                            62.159984
DiabetesPedigreeFunction        0.109779
Age                           138.303046
Outcome                         0.227483
dtype: float64
```

```
df.median()
```

```
Pregnancies                   3.0000
Glucose                     117.0000
BloodPressure                72.0000
SkinThickness                23.0000
Insulin                      30.5000
BMI                          32.0000
DiabetesPedigreeFunction      0.3725
Age                          29.0000
Outcome                       0.0000
dtype: float64
```

```
df.mode().iloc[0]
```

```
Pregnancies                  1.000
Glucose                     99.000
BloodPressure               70.000
SkinThickness                0.000
Insulin                      0.000
BMI                         32.000
DiabetesPedigreeFunction     0.254
Age                         22.000
Outcome                      0.000
Name: 0, dtype: float64
```

```
# x=df.iloc[:,:-1]
# y=df.iloc[:,-1]

x=df.drop("Outcome",axis=1)
y=df["Outcome"]

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando
m_state=100)

model = LinearRegression()
model.fit(x_train,y_train)
y_pred_linear=model.predict(x_test)
```

```
print("Linear Regression R-squared:",r2_score(y_test,y_pred_linear))
print("MSE :",mean_squared_error(y_test,y_pred_linear))

Linear Regression R-squared: 0.19996694950228422
MSE : 0.1805775391851186

logistic = LogisticRegression()
logistic.fit(x_train,y_train)
y_pred_logistic=logistic.predict(x_test)


accuracy = accuracy_score(y_test, y_pred_logistic)
print(f"Logistic Regression Accuracy: {accuracy}")

Logistic Regression Accuracy: 0.7532467532467533





# X = df[['Age', 'BMI']]
# y = df['Outcome']

# model = LinearRegression()
# model.fit(X, y)

# # You can now analyze the coefficients, make predictions, and
evaluate the model's performance.
# new_data = pd.DataFrame({'Age': [40, 45], 'BMI': [30, 35]})
# predictions = model.predict(new_data)
# print("Predictions:", predictions)

# X = df.drop(["Outcome"],axis=1)
# y = df['Outcome']

# log_model = LogisticRegression()
# log_model.fit(X, y)

# new_data = pd.DataFrame({'Pregnancies': [5, 3], 'Glucose': [120,
160], 'BloodPressure': [70, 80], 'SkinThickness': [30, 20], 'Insulin':
[0, 40], 'BMI': [25.5, 28.6], 'DiabetesPedigreeFunction': [0.5, 0.4],
'Age': [35, 28]})
# probabilities = log_model.predict_proba(new_data)
# print("Probabilities of having diabetes:",probabilities)
```

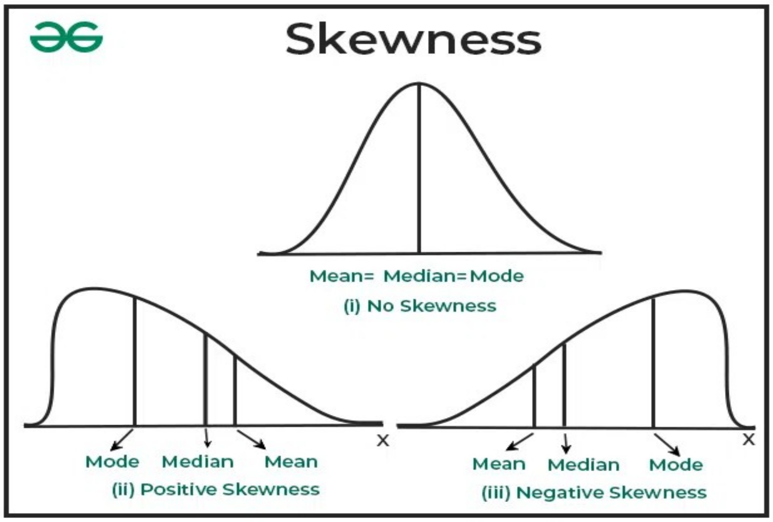
standard deviation = calculate the dispersion of the dataset relative to its mean, calculated by

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}$$

square root of variance

variance= Variance is a statistical measurement of the spread between numbers in a data set

standard deviation is the square root of the variance, and variance is the average of the squared difference of each data point from the mean.

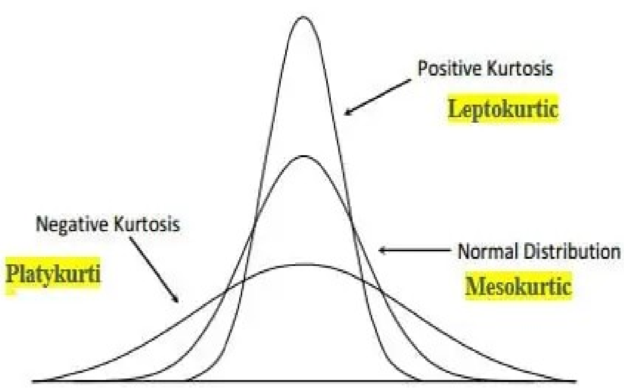


skewness= Skewness is a measure of the asymmetry of a distribution (Skewness = Mean – Mode)

Positive Skewness (Right Skew)-- Mean > Median > Mode

Negative Skewness (Left Skew)-- Mean < Median < Mode

kurtosis: statistical measure used to describe the distribution of the observed data around mean



An R-Squared value shows how well the model predicts the outcome of the dependent variable.R-Squared values range from 0 to 1. An R-Squared value of 0 means that the model explains or predicts 0% of the relationship between the dependent and independent variables.

$$R^2 = 1 - \frac{RSS}{TSS}$$

R^2 = coefficient of determination

RSS = sum of squares of residuals

TSS = total sum of squares

$$\begin{aligned}\text{r2\_score} &= 1 - \frac{\text{total\_error\_model}}{\text{total\_error\_baseline}} \\ &= 1 - \frac{\sum_{i=1}^{N} (\text{predicted}_i - \text{actual}_i)^2}{\sum_{i=1}^{N} (\text{average\_value} - \text{actual}_i)^2}\end{aligned}$$

mean_squared_error -- average squared difference between the predicted values and the actual values in the dataset.(lower the better)

$$\mathrm{MSE} = \overset{\text{Mean}}{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}\Big(\overset{\text{Error}}{\boxed{Y_i - \hat{Y}_i}}\Big)^{\overset{\text{Squared}}{\boxed{2}}}$$

Univariate Analysis For each dataset, compute the following for each numeric column:

Frequency: Count occurrences of each unique value (particularly useful for categorical data).

Mean, Median, and Mode: Central tendency measures.

Variance and Standard Deviation: Spread measures to understand data variability.

Skewness: Measures asymmetry of the data distribution. Skewness > 0 indicates a right-skewed distribution; Skewness < 0 indicates a left-skewed distribution.

Kurtosis: Indicates the "tailedness" of the distribution. Kurtosis > 0 indicates a heavy-tailed distribution, while Kurtosis < 0 suggests a light-tailed distribution.

```
x=df[["Glucose"]]
y=df["Outcome"]

model3=LinearRegression()
model3.fit(x,y)
y_pred_model3=model3.predict(x)

mse = mean_squared_error(y, y_pred_model3)
r2 = r2_score(y, y_pred_model3)
print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.17772834105264668
R-squared: 0.21769820124599804
```