1. Understanding data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Create dataframe

```
df = pd.read_csv("diabetes.csv")
df.head()
```

`df.columns` → Return column name / features

`df.dtypes` → Data types of each column

Missing values in given dataset.

```
df.isnull().sum()
```

```
df.head(20)
```

⇒ There are many zeros like in blood pressure, insulin, skin thickness, BMI columns, which are not possible (problem).

* Data Imputation of 0's in every feature
* Size of the data

```
df.shape
```

* Target column: Outcome [0, 1] (Binary classification Task)
* As the target column is available in the dataset, supervised machine learning algorithm
* Records: 768

⇒ Data does not contain any null values but having some wierd zeros.

2. EDA : Feature Selection and Data Imputation

Correlation coeff. between the features

```
df.corr()
```

Heat map

```
plt.figure(figsize=(15,15))
ax = sns.heatmap(df.corr(), annot=True)
plt.savefig('correlation_coeff.jpg')
plt.show()
```

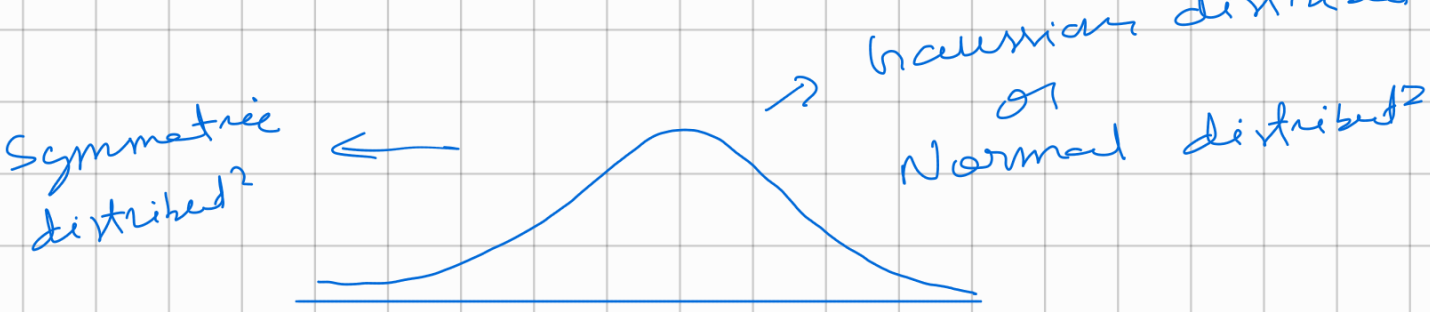
To add value in heat map

⇒ No features are highly correlated so we cannot remove any feature.

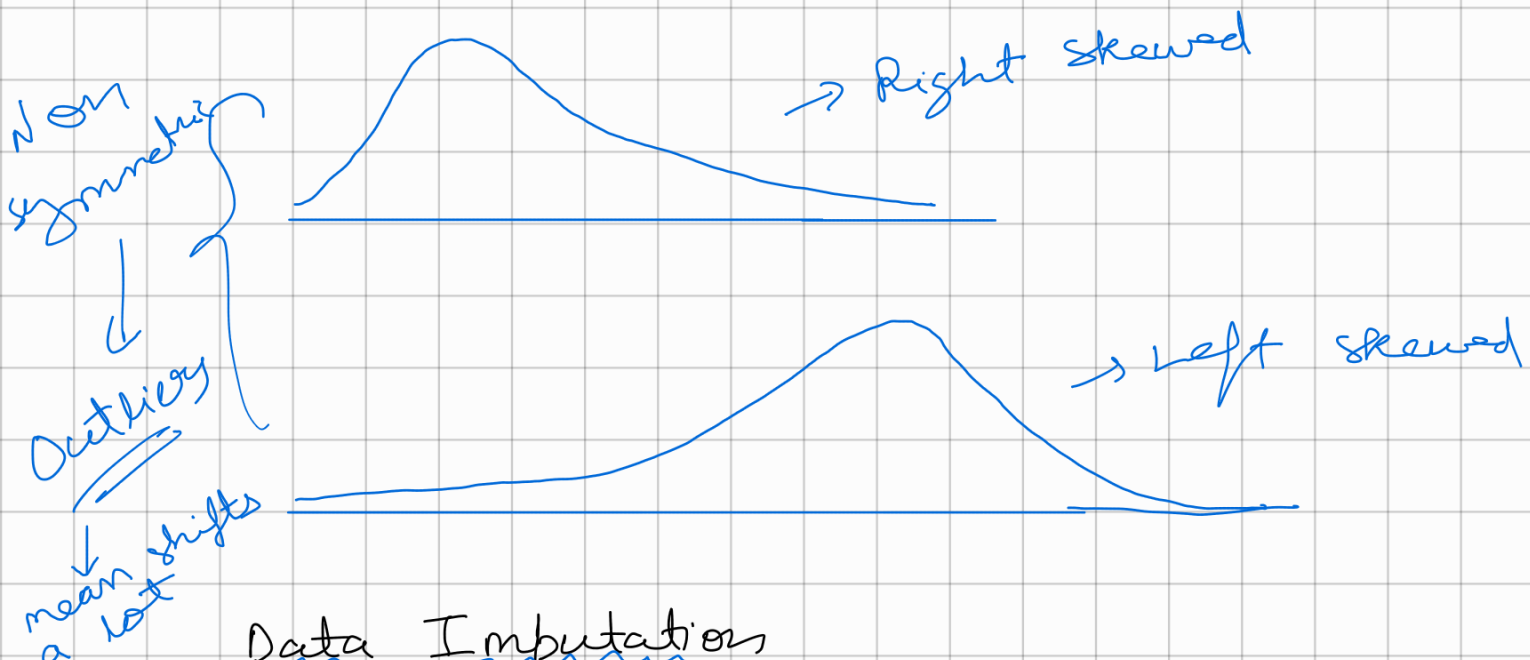
Descriptive Statistics of the given data

```
df.describe()
```

Gaussian distributed

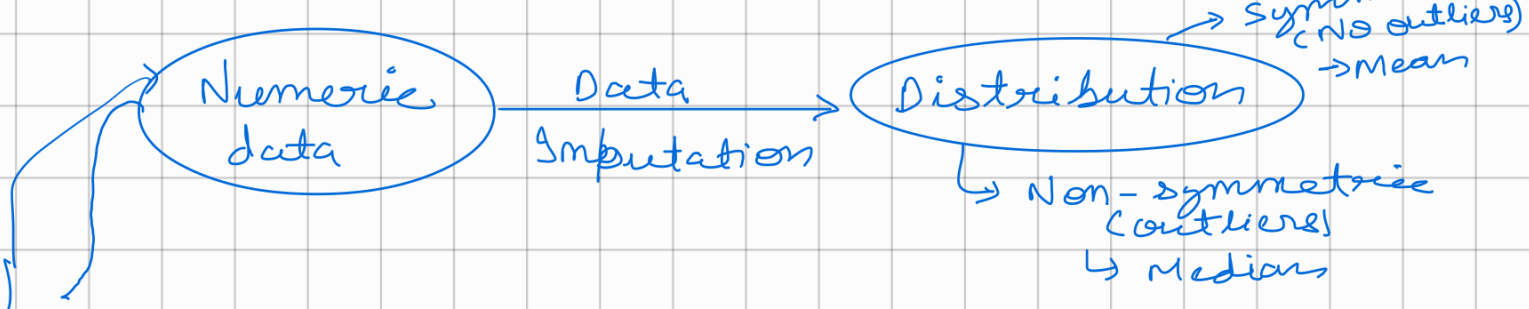


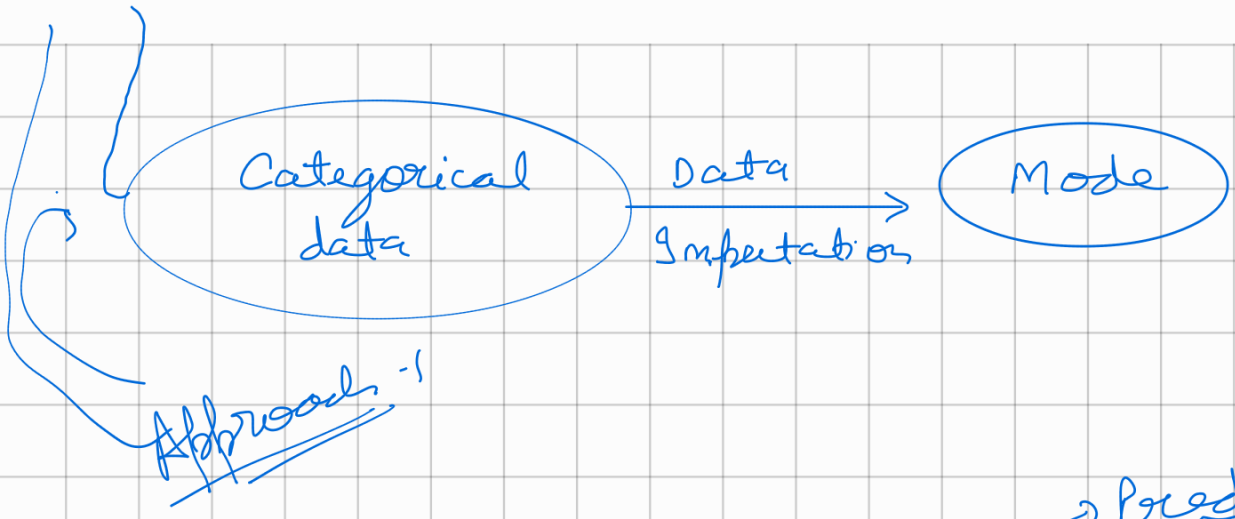
Standard Normal Distribut^n
↳ $\mu = 0$, $\sigma = 1$



Data Imputation

median is more robust to outliers.





Approach 2 → Model ↳ KNN → Predictions ↳ Data Imputation

Distribution

```
sns.distplot(df.Pregnancies)  # median
sns.distplot(df.BloodPressure)  # mean
sns.distplot(df.Insulin)  # median
```

Data imputation of insulin → median

Insuline → Right skewed distribution

```
df['Insulin'] = df['Insulin'].replace(0,
                df['Insulin'].median())
```

Blood Preser → mean
Pregnancies → median
skin thickness →

Data imputation :- Data imputation is the process of replacing missing data values in a dataset with substituted values, a crucial step in data preprocessing to ensure data integrity and enable effective analysis.

Why Impute Data?

i) Handling Missing Data :- Real-world datsets often contain missing values due to various reasons, such as data entry errors, incomplete data recording, or sensor failures.

ii) Maintaining Data Integrity :- Ignoring or deleting rows with missing values can

lead to a significant reduction in dataset size and potentially introduce bias, impacting the reliability of analysis.

iii) Enabling Analysis :- Many statistical and machine learning algorithms cannot handle missing data, making imputation necessary for data analysis and model training

Common Imputation Techniques:

i) Mean / Median / Mode Imputation :- Replacing missing values with the mean, median or mode of the non-missing values in the same column.

ii) Constant Value Imputation :- Replacing missing values with a constant value, such as 0 or a specific placeholder.

iii) Regression Imputation :- Using a regression model to predict missing values based on other variables in the dataset.

iv) K-Nearest Neighbors (KNN) Imputation :- Using the values of similar data points (neighbors) to estimate missing values.

v) Multiple Imputation :- Generating multiple imputed datasets and analyzing them separately, allowing for uncertainty in the imputed values to be reflected in the analysis.

Considerations When Choosing an Imputation Method:

i) Type of Missing Data :- Understanding the reason for missing

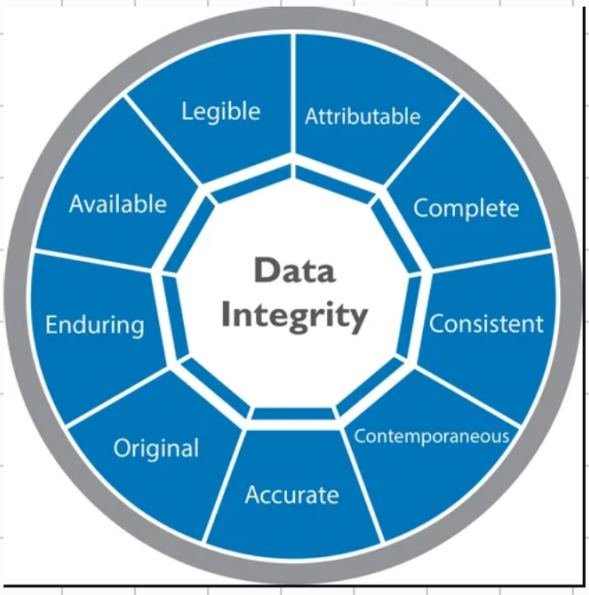
values is important, as different imputat^n methods are suitable for different types of missing data.
eg:- Missing Completely at Random (MCAR), Missing at Random (MAR), Missing Not at Random (MNAR).

ii) Data Characteristics :-
The nature of the data (eg., numerical, categorical) and its distribution can influence the choice of imputation method.

iii) Potential Bias :- Imputation methods can introduce bias, so it's important to choose a method that minimize bias and preserves the integrity of the data.

iv) Computational cost :- Some imputation methods, like multiple imputation, can be computationally expensive.

⇒ Data integrity is the accuracy, consistency, and reliability of data over its entire lifecycle. It can also refer to the process of ensuring that data is valid and accurate.



EDA : Outlier Detection and Normalization

X → input features
Y → target value

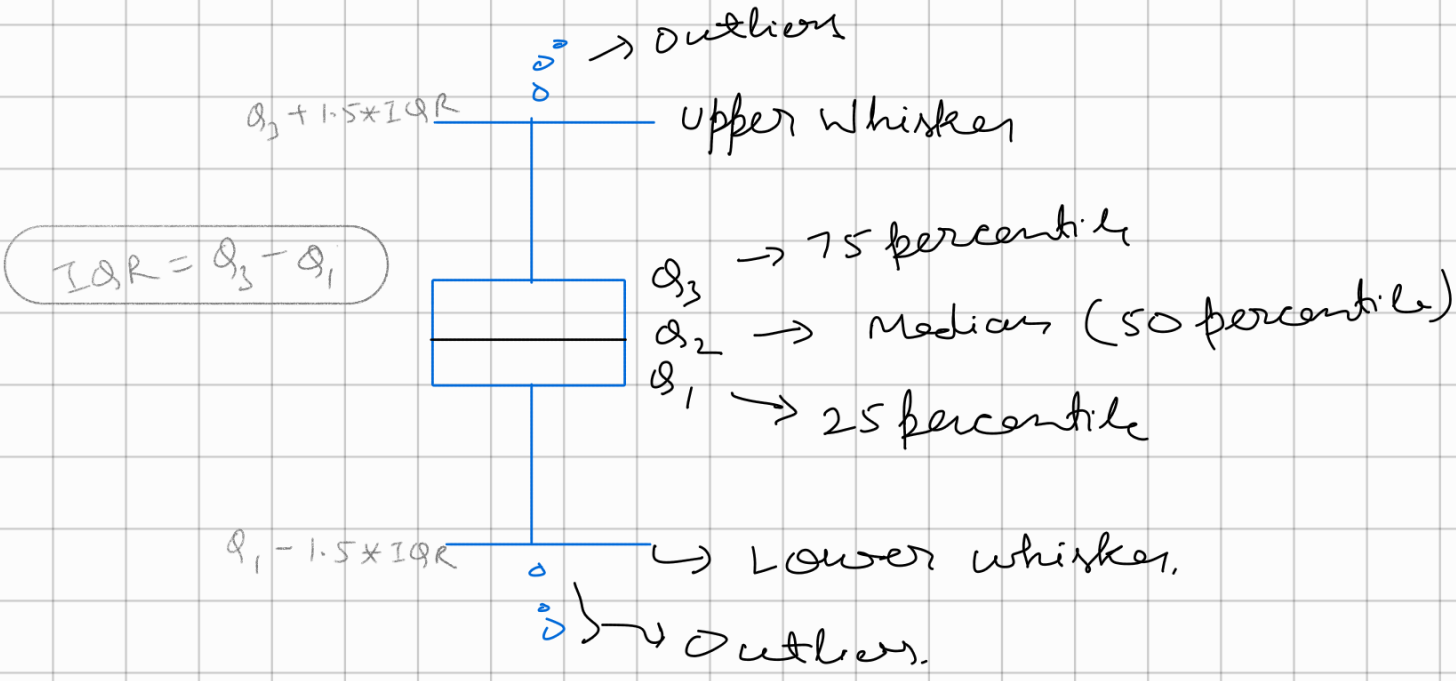
```
X = df.drop(columns = "Outcome", axis =1)
```

↳ Deleted the outcome column for X.

```
Y = df["Outcome"]
```

1. Outlier Detection

```
fig, ax = plt.subplots(figsize =(15, 15))
sns.boxplot(data =X , ax =ax)
plt.savefig("BoxPlot.jpg")
```




```
cols = ['Pregnancies', 'Glucose', 'BloodPressure',
        'SkinThickness', 'Insulin', 'BMI',
        'DiabetesPedigreeFunction', 'Age']
for col in cols:
    Q1 = X[col].quantile(0.25)
    Q3 = X[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    mask = (X[col] >= lower_bound) &
           (X[col] <= upper_bound)
```

```
X_outlier_detection = X[mask]
Y_outlier_detection = Y[mask]
```

} After Removing outlier.

2. Standardization / Normalization

↳ To reduce biasness ($\mu=0$, $\sigma=1$)

```
from sklearn.preprocessing import
    StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_outlier_detection)
```

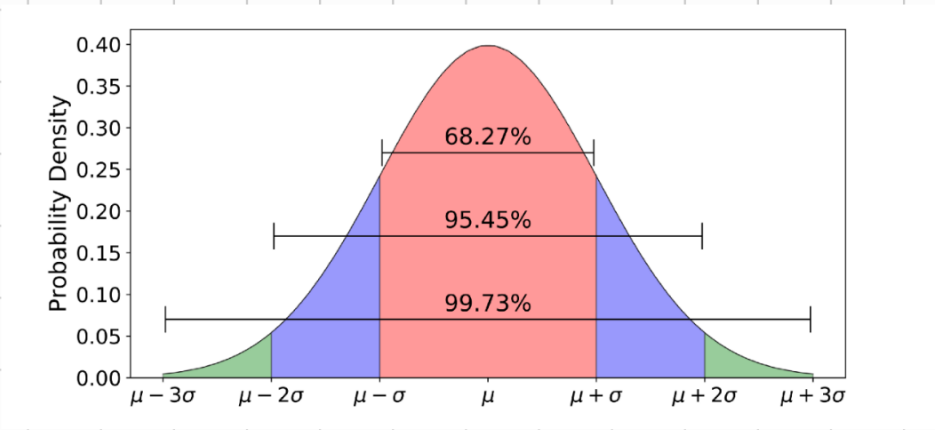
⇒ After this outliers are found

```
X_scaled = pd.DataFrame(X_scaled)

X_scaled.describe()

Y_outlier_detection.value_counts()
```

↳ Outcome
0 : 493
1 : 263
} ⇒ Biased

EDA :- Outlier Handling Techniques

Normal / Gaussian Distribution




```
cols = ['Pregnancies', 'Glucose', 'BloodPressure',
        'SkinThickness', 'Insulin', 'BMI',
        'DiabetesPedigreeFunction', 'Age']

X_scaled = pd.DataFrame(X_scaled, columns=cols)
X_scaled.describe()

fig, ax = plt.subplots(figsize=(15,15))
sns.boxplot(data = X_scaled, ax=ax)
plt.savefig('boxplot_1.jpg')
```

95 percentile

```
q = X_scaled["Insulin"].quantile(0.95)
datanew = X_scaled[X_scaled['Insulin']
                   <q]
```

```
fig, ax = plt.subplots(figsize=(15,15))
sns.boxplot(data = datanew , ax = ax)
plt.savefig('boxplot2.jpg')
```

6. Model Training : Splitting data and Handling of Imbalanced Data

Sklearn test train split

Model Training

Splitting of data into training and testing

```
from sklearn.model_selection import
train_test_split

X_train, X_test, y_train, y_test =
    train_test_split(X_scaled, y_cleaned,
    test_size = 0.33, random_state =42)
```

33%

Result will be same everytime

```
y_train.value_counts()
```

⇒ Outcome

0	293
1	135

} Data Imbalance

Data Imbalancing

- → Oversampling :- Minority class and increase that number to the majority class.
- → Undersampling :- Majority class and decrease that number to the minority class
- → SMOTE :- Synthetic data and

increase the number of samples to the majority class.

Imblearn documentation

SMOTE Technique

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, Y_train_resampled = smote.fit_resample(X_train, Y_train)

# Check resampled class distribution
print("\nResampled class distribution:")
print(pd.Series(Y_train_resampled).value_counts())
```

7. Model Training : Implementation of Logistic Regression

sklearn logistic regression

```
from sklearn.linear_model import LogisticRegression
classification = LogisticRegression()
classification.fit(X_train_resampled, Y_train_resampled)
```

8. Model Prediction and Evaluation

Model Prediction

```
Y_predictions = classification.predict(X_test)
print(Y_predictions)
```

Model Evaluation

sklearn accuracy score

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, Y_predictions)
```

Prediction is 75.943%.

Classification report

sklearn classification report

Health care :- Recall is very important metric

```
from sklearn.metrics import classification_report
target_names = ['Non-Diabetic', 'Diabetic']
print(classification_report(Y_test, Y_predictions,
                            target_names = target_names))
```

9. Saving the Model

```
import pickle
pickle.dump(classification,
            open("classification_model.pkl", "wb"))
```

Use saved file

```
classification_model = pickle.load(open(
    "classification_model.pkl", "rb"))

classification_model.predict(X_test)
```

10 Project Outcome