PREDICTIVE MODELING & PATIENT SEGMENTATION

ASSESSMENT OBJECTIVE

- To develop an intelligent system that applies both classification and clustering techniques to solve a problem in a particular domain.
- To identify a problem where classification can be used to predict outcomes or categorize data, and where clustering can help discover hidden patterns or segment data into meaningful group.

DOMAIN WORKED ON:

- Healthcare (Medical Domain)

Link to where dataset is generated or gotten:
https://www.kaggle.com/code/paultimothymooney/predict-diabetes-from-medical-records?select=diabetes.csv

SOFTWARE/ PYTHON LIBRARIES USED:

Jupyter Notebook is the machine used to run the codes. The following are the python Libraries used are;

- Pandas Library
- NumPy Library
- Seaborn Library
- Sklearn Libraries

OVERVIEW OF THIS TASK

Classification Models

Classification is a supervised learning technique used to predict a discrete label or category for data points based on input features. The process involves training a model on labeled data and then using the model to predict the class of new, unseen instances.

Popular Classification Models:

1. **Logistic Regression**: A statistical model that uses a logistic function to model binary or multi-class outcomes. It's simple yet effective for linear decision boundaries. (Hosmer et al., 2013)

2. **Decision Trees**: A flowchart-like structure where internal nodes represent feature tests, and leaf nodes represent outcomes. They are intuitive and handle non-linear relationships well.
3. **Random Forest**: An ensemble method that combines multiple decision trees to improve predictive accuracy and reduce overfitting.
4. **Support Vector Machines (SVM)**: A powerful algorithm that finds the hyperplane that best separates data into classes. Effective in high-dimensional spaces.
5. **Neural Networks**: Inspired by biological neural systems, they are particularly suited for large datasets and complex patterns, such as image or speech classification. (LeCun et al., 2015).

Applications:

- Disease diagnosis (e.g., diabetes classification)
- Fraud detection
- Customer segmentation for targeted marketing

Evaluation Metrics:

- Accuracy
- Precision, Recall, F1-score
- ROC-AUC for imbalanced datasets

Clustering Algorithms

Clustering is an unsupervised learning technique used to group data points into clusters based on their similarities. Unlike classification, clustering does not require labeled data.

Popular Clustering Algorithms:

1. **K-Means**: Partitions data into kkk clusters by minimizing intra-cluster variance. It's computationally efficient but sensitive to the choice of kkk. (MacQueen, 1967).
2. **Hierarchical Clustering**: Builds a tree-like structure (dendrogram) representing nested clusters. Can be agglomerative (bottom-up) or divisive (top-down).
3. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: Groups points that are closely packed together and marks outliers as noise. Works well for clusters of arbitrary shape.
4. **Gaussian Mixture Models (GMMs)**: Assumes data is generated from a mixture of several Gaussian distributions and estimates the parameters using Expectation-Maximization. (Reynolds, 2009).

5. **Spectral Clustering**: Uses graph-based techniques to identify clusters, particularly effective for non-convex shapes.

Applications:

- Market segmentation
- Image segmentation
- Social network analysis

Evaluation Metrics:

- Silhouette Score
- Davies-Bouldin Index
- Adjusted Rand Index (when ground truth is available)

KEY DIFFERENCES BETWEEN CLASSIFICATION AND CLUSTERING

1. **Label Requirement**:
   - Classification: Requires labeled data for training.
   - Clustering: Works on unlabeled data.
2. **Purpose**:
   - Classification predicts categories based on prior knowledge.
   - Clustering discovers hidden patterns and structures in data.
3. **Output**:
   - Classification assigns a predefined label.
   - Clustering identifies natural groupings.

ABOUT DATASET USED

The dataset used originates from the National Institute of Diabetes and Digestive and Kidney Diseases and downloaded from kaggle website. Its purpose is to predict diagnostically whether a patient has diabetes based on specific diagnostic measurements provided in the dataset or not. The instances were carefully selected from a larger database, with particular criteria applied. Specifically, all patients in this dataset are females of Pima Indian heritage who are at least 21 years old.

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, Glucose, blood pressure, Skin thickness, and Diabetes predgree function. It contains **768 rows of data and 9 columns.**

- **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) 268 of 768 are 1, the others are 0.

ANALYSIS REPORT

Based on this dataset (Diabetes.csv), I proceeded with the planned steps for classification (to predict Outcomes) and clustering (to segment patients). Here's the preprocessing and modeling workflow:

1. **Loading the Dataset:** This help to read the dataset into the machine. Below image shows the details information of the dataset using *.info()* after loading the data using *pd.read.csv* and *.head()* to view first 10 entries. Also, it shows there are 2 different dtypes in this dataset(diabetes.csv) which are float(2) and int(7).

```
[230]: 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
```

```
[231]: df.head(10)
```

[231]:

	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

A patient having an insulin level or skin thickness measurement of zero would indicate a significant medical issue. Therefore, it is reasonable to infer that the dataset uses zero as a placeholder for missing or null values. Upon examining the dataset, it becomes evident that up to half of the rows include columns with missing data.

2. **Handle Missing Data**: Replace zeros in specific columns (e.g.,insulin Glucose, BMI) with appropriate statistical values (like medians). Using *.isnull().sum()*, to count if there is blank entries but its clear that there are no NAN or empty rows or columns but from the info shows most columns has zero values which as replaced with median of the data.

```
[236]: #Checking for missing value
       df.isnull().sum()
```

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

Its clearly shows there are no missing values in the dataset which is cool!

Missing analysis on Diabetic dataset

```
[18]: zero_features = ['Pregnancies','Glucose','BloodPressure','SkinThickness',"Insulin",'BMI']
total_count = df['Glucose'].count()

for feature in zero_features:
    zero_count = df[df[feature]==0][feature].count()
    print('{0} 0 number of cases {1}, percent is {2:.2f} %'.format(feature, zero_count, 100*zero_count/total_count))
```

```
Pregnancies 0 number of cases 111, percent is 14.45 %
Glucose 0 number of cases 5, percent is 0.65 %
BloodPressure 0 number of cases 35, percent is 4.56 %
SkinThickness 0 number of cases 227, percent is 29.56 %
Insulin 0 number of cases 374, percent is 48.70 %
BMI 0 number of cases 11, percent is 1.43 %
```

Code to analyze the percentage of zeros in each columns

From the analyses, the number of zeros in pregnancies, glucose, BloodPressure, SkinThinkness, Insulin, and BMI are 14.45%, 0.65%, 4.56%, 29.56%, 48.70%, and 1.43% respectively.

Below is the data description showing the STD, means and other statistical analysis about the data before replacing the zeros.

```
[17]: #Checking for data description.
      df.describe()
```

[17]:

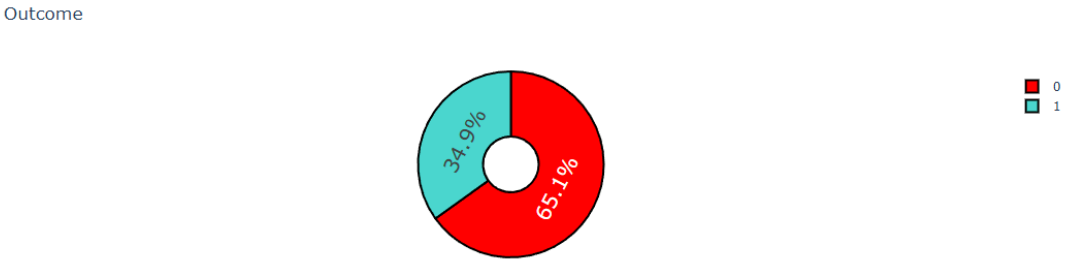
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	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%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Data Description of the diabetics dataset

Below is the data visualization that shows the analyze of outcome columns in the dataset which is the target for this dataset.

Data Visualization

```
[32]: colors = ['red', 'mediumturquoise']
labels = ['0','1']
values = df['Outcome'].value_counts()/df['Outcome'].shape[0]

# Use `hole` to create a donut-like pie chart
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.update_traces(hoverinfo='label+percent', textinfo='percent', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.update_layout(
    title_text="Outcome")
fig.show()
```

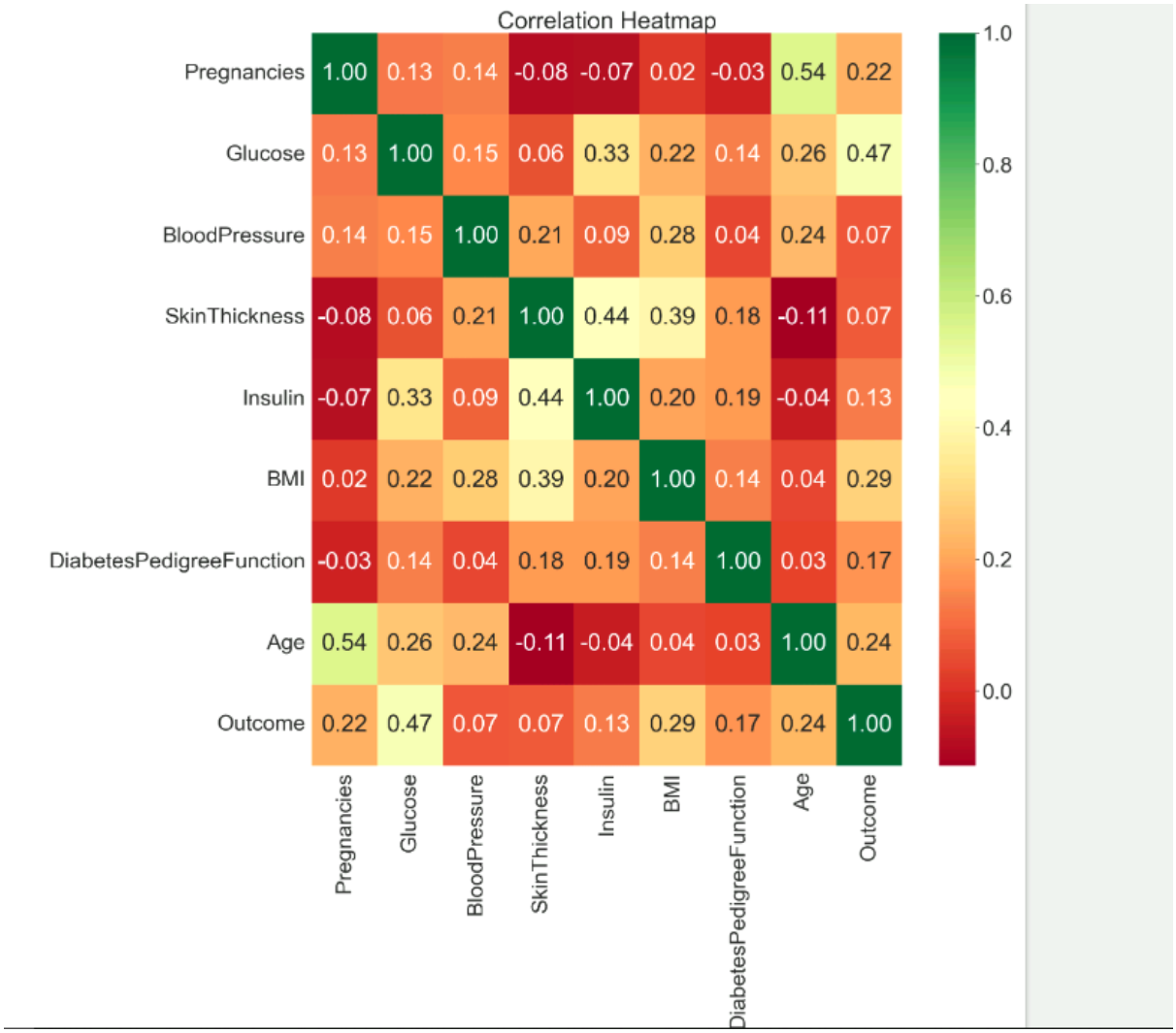


Preview of the Outcome Percentage using Donut Pie Chart

This shows that 34.9% of the persons involved in this dataset analyses has diabetics and 65.1% of the person doesnt not have diabetics.

3. **Feature Selection From Heatmap Image Analysis:** After the correlation analysis was carried out, its shows the relationship between the columns. From the image below,

- **Glucose**: Given its strong correlation with Outcome, prioritize this feature for predictive modeling.
- **Pregnancies and Age**: These are biologically relevant, so consider interactions or non-linear models to capture relationships.
- **Explore Weak Features**: Features like BloodPressure and DiabetesPedigreeFunction might still contribute indirectly or in combination with other features.



❖ **Understanding the Correlation:** The **correlation coefficient** measures the linear relationship between two variables, ranging from -1 to +1:

- **+1**: Perfect positive correlation (as one variable increases, the other also increases).
- **0**: No linear correlation.
- **-1**: Perfect negative correlation (as one variable increases, the other decreases).

The color scale in the heatmap represents the magnitude of correlation:

- **Green:** High positive correlation.
- **Yellow:** Moderate correlation.
- **Red:** Negative correlation or weak correlation.

❖ **Observations form the map**

✓ **Strong Positive Correlations**

- **Glucose and Outcome (0.47)**: A moderate-to-strong positive correlation indicates that higher glucose levels are associated with a higher likelihood of diabetes.
- **Pregnancies and Age (0.54)**: Strong positive correlation suggests that older individuals in this dataset tend to have more pregnancies.

✓ **Weak-to-Moderate Correlations**

- **BMI and Outcome (0.29)**: Indicates that higher BMI values are slightly associated with diabetes.
- **Age and Outcome (0.24)**: Suggests older individuals have a slightly higher likelihood of diabetes.
- **Insulin and SkinThickness (0.44)**: Moderate positive correlation reflects a biological relationship where higher insulin levels are often linked to greater skin thickness.

✓ **Weak or Negligible Correlations**

- **Blood Pressure and Outcome (0.07)**: Negligible correlation suggests that blood pressure may not play a significant role in predicting diabetes in this dataset.
- **DiabetesPedigreeFunction and Outcome (0.17)**: Weak correlation suggests limited direct influence of hereditary factors on diabetes in this dataset.

Implications for Analysis

- **Feature Selection**: Features with higher correlations to Outcome (e.g., Glucose, BMI, Age) are likely to be more useful for predicting diabetes. Features like BloodPressure may have less predictive value.
- **Multicollinearity**: Features with strong correlations to each other (e.g., Insulin and SkinThickness) may introduce redundancy into the model.
- **Modeling Strategy**:
    a. Classification models can use features like Glucose and BMI for higher predictive power.
    b. Clustering may benefit from normalized data to emphasize weak but meaningful relationships.

3. Standardize Features: For Scaling data for clustering and improve classification accuracy. Standardizing features is a critical step in machine learning, especially for algorithms like **K-Means clustering** and distance-based classification models (e.g., Logistic Regression, SVM, RFM etc.). The purpose is to ensure that all features contribute equally to the model, regardless of their original scale. It was split into 80:20 ratio for the train and test.

[55]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

X_train_scaled
```

[55]:
```
array([[0.05882353, 0.23776224, 0.3877551 , ..., 0.18404908, 0.22093541,
        0.05      ],
       [0.29411765, 0.48951049, 0.55102041, ..., 0.23312883, 0.15812918,
        0.31666667],
       [0.11764706, 0.34265734, 0.34693878, ..., 0.34151329, 0.06280624,
        0.06666667],
       ...,
       [0.05882353, 0.28671329, 0.46938776, ..., 0.40695297, 0.0596882 ,
        0.15      ],
       [0.58823529, 0.38461538, 0.46938776, ..., 0.19018405, 0.02538976,
        0.31666667],
       [0.23529412, 0.61538462, 0.34693878, ..., 0.23108384, 0.09042316,
        0.26666667]])
```

[47]:
```
print("Data type of X_train_scaled:", type(X_train_scaled))
```

```
Data type of X_train_scaled: <class 'numpy.ndarray'>
```

Scaled dataset for clustering

Using MinMaxScaler instead of StandardScaler is often preferred when you want to scale the data to a specific range, such as [0, 1].

- **Reasons to Standardize:**
  a. **Equal Weightage to All Features**
     - ✓ Features like Age (in years) and Glucose (typically 50–200 mg/dL) might have different scales.

- ✓ Algorithms that rely on **distance metrics** (e.g., K-Means, KNN, SVM) can get biased toward features with larger magnitudes, as they dominate distance calculations.
- ✓ Standardization ensures that each feature contributes equally.

b. **Improve Model Convergence**

- ✓ Many optimization algorithms (e.g., gradient descent) work better and converge faster when data is standardized, as they avoid large gradients caused by scale differences.
- ✓ For classification models like Random Forest Model, standardizing helps achieve better accuracy and stability.

c. **Interpretability of Clustering**

- ✓ In clustering, standardization helps form more meaningful and interpretable clusters. Without standardization, features with larger scales may drive the cluster formation, masking the influence of smaller-scale features.

d. **Compatibility with Distance-Based Metrics**

- ✓ Clustering algorithms like K-Means rely on **Euclidean distance** to assign points to clusters. If features are on different scales, the algorithm may overemphasize certain features while ignoring others.

5. Build Models and Evaluation:

- **Classification**: Train a model to predict diabetes (Outcome). Firstly, the dataset is been split into x and y (the outcome column) for model selection using *import train_test_ split* to be able to split the data into x_test, x_train and y_test, y_train . As shown in the image below;

```
[24]: from sklearn.model_selection import train_test_split
      X = df.drop(columns=["Outcome"])
      y = df["Outcome"]

[25]: X.head()
```

[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33

```
[26]: y.head()

[26]: 0    1
      1    0
      2    1
      3    0
      4    1
      Name: Outcome, dtype: int64

[27]: # Split data for classification
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

Model Selection for Classification

```
[28]: X_train
```

[28]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
353	1	90.0	62.0	12.0	43.0	27.2	0.580	24
711	5	126.0	78.0	27.0	22.0	29.6	0.439	40
373	2	105.0	58.0	40.0	94.0	34.9	0.225	25
46	1	146.0	56.0	29.0	125.0	29.7	0.564	29
682	0	95.0	64.0	39.0	105.0	44.6	0.366	22
...	...	...	...	...	...	...	...	...
451	2	134.0	70.0	29.0	125.0	28.9	0.542	23
113	4	76.0	62.0	29.0	125.0	34.0	0.391	25
556	1	97.0	70.0	40.0	125.0	38.1	0.218	30
667	10	111.0	70.0	27.0	125.0	27.5	0.141	40
107	4	144.0	58.0	28.0	140.0	29.5	0.287	37

614 rows × 8 columns

```
[29]: y_train
```

```
[29]: 353    0
      711    0
      373    0
      46     0
      682    0
            ..
      451    1
      113    0
      556    0
      667    1
      107    0
      Name: Outcome, Length: 614, dtype: int64
```

Scale the features for consistency in clustering and model training

Preview on the x_train and y_train

```
[71]: # Print summary of processed data
      print("Training set shape:", X_train_scaled.shape)
      print("Testing set shape:", X_test_scaled.shape)
      print("Dataset shape after scaling (for clustering):", X_scaled.shape)
```

```
Training set shape: (614, 8)
Testing set shape: (154, 8)
Dataset shape after scaling (for clustering): (768, 8)
```

Summary of processed data

- **Classification Results Using Random Forest Model classification**

The classification task predicts whether a patient is diabetic (Outcome = 1) or non-diabetic (Outcome = 0). Using Random Forest Model classification here are the **Key Evaluation Metrics;**

1. **Accuracy**: The percentage of correct predictions is 74% (0.74) which is not bad to consider.

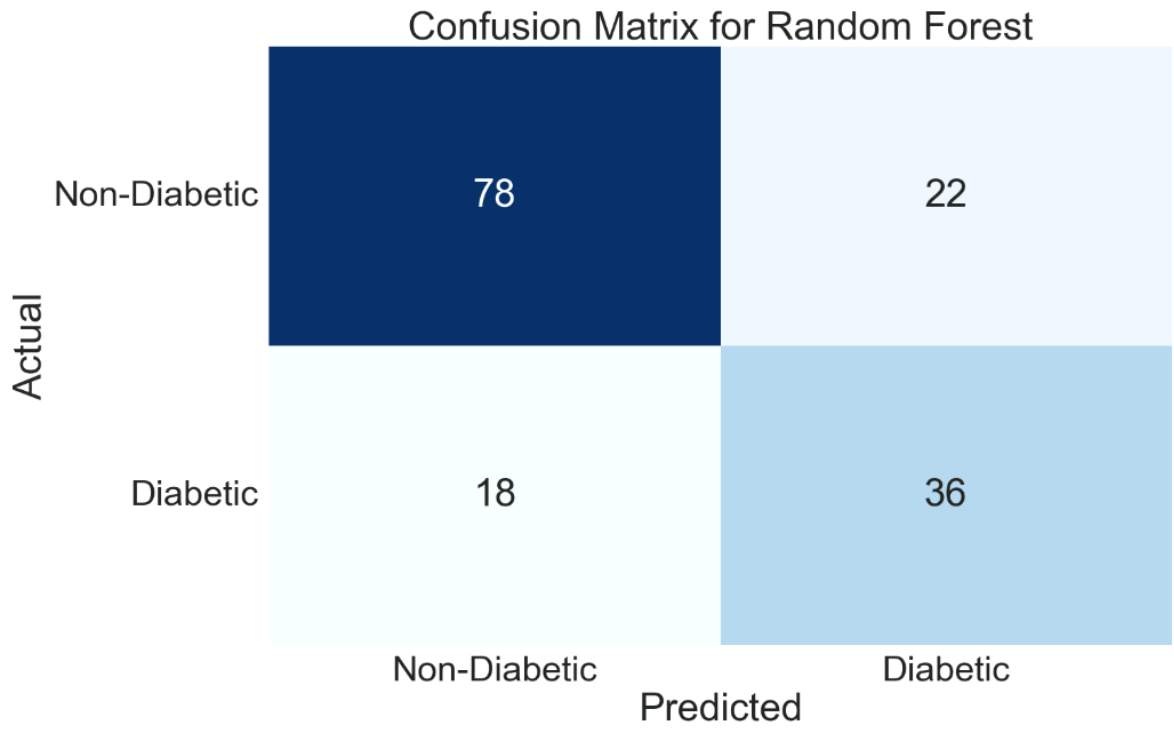
```
[63]: print("Random Forest Classification Report:")
      print(classification_report(y_test, rf_predictions))
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.78      0.80       100
           1       0.62      0.67      0.64        54

    accuracy                           0.74       154
   macro avg       0.72      0.72      0.72       154
weighted avg       0.75      0.74      0.74       154
```

```
[65]: print("Random Forest Confusion Matrix:")
      print(confusion_matrix(y_test, rf_predictions))
```

```
Random Forest Confusion Matrix:
[[78 22]
 [18 36]]
```

2. **Confusion Matrix**: This is a **confusion matrix** diagram for a classification model (in this case, Random Forest) applied to a diabetes prediction problem. It shows the performance of the model by comparing actual and predicted classifications into two categories: **Non-Diabetic** and **Diabetic**. Here's a breakdown:



Confusion Metric for RFM Classification

Structure of the Confusion Matrix

3. **Rows (Actual)**:

   Represent the true class labels:

   1. **Non-Diabetic** (Actual Non-Diabetic instances)
   2. **Diabetic** (Actual Diabetic instances)

4. **Columns (Predicted)**:

   Represent the predicted class labels made by the model:

   1. **Non-Diabetic** (Predicted as Non-Diabetic)
   2. **Diabetic** (Predicted as Diabetic)

5. **Values in the Cells**: Represents the counts of instances for each combination of actual and predicted labels.

   a. **True Positives (TP):** Correctly predicted diabetic patients is 36

   b. **True Negatives (TN):** Correctly predicted non-diabetic patients is 78

   c. **False Positives (FP):** Non-diabetic patients misclassified as diabetic is 22

   d. **False Negatives (FN):** Diabetic patients misclassified as non-diabetic is 18

6. **Classification Report**:

   Precision: Ratio of correctly predicted positives to all predicted positives for non diabetic and diabetics are 0.81 and 0.62 respectively.

   Recall: Ratio of correctly predicted positives to all actual positives for recall for non diabetic and diabetic are 0.78 and 0.67 respectively.

   F1-Score: Harmonic mean of precision and recall for non-diabetic and diabetics are 0.80 and 0.64 respectively.

[69]:
```
# Add a new column 'predicted' to the test set
X_test['rf_predicted'] = rf_predictions
X_test
```

[69]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	rf_predicted
44	7	159.0	64.0	29.0	125.0	27.4	0.294	40	1
672	10	68.0	106.0	23.0	49.0	35.5	0.285	47	0
700	2	122.0	76.0	27.0	200.0	35.9	0.483	26	0
630	7	114.0	64.0	29.0	125.0	27.4	0.732	34	0
81	2	74.0	72.0	29.0	125.0	32.3	0.102	22	0
...	...	...	...	...	...	...	...	...	...
32	3	88.0	58.0	11.0	54.0	24.8	0.267	22	0
637	2	94.0	76.0	18.0	66.0	31.6	0.649	23	0
593	2	82.0	52.0	22.0	115.0	28.5	1.699	25	0
425	4	184.0	78.0	39.0	277.0	37.0	0.264	31	1
273	1	71.0	78.0	50.0	45.0	33.2	0.422	21	0

154 rows × 9 columns

X_Test Data Prediction Outcome using RF Model from the trained data

This shows that the data are actually 74% accurate comparing with the actual dataset when checked.

- **Clustering Algorithm**

K-Means is an unsupervised learning algorithm used to partition data into K clusters. It minimizes the within-cluster variance by iteratively assigning data points to the nearest cluster center and updating the centers (MacQueen, 1967).

a. Advantages

- **Simplicity**: Easy to implement and computationally efficient.
- **Scalability**: Handles large datasets effectively.
- **Flexibility**: Can be applied to a variety of domains, including healthcare.

b. Limitations

- **Predefined Clusters**: The number of clusters (K) must be defined beforehand, which may require domain knowledge.
- **Sensitivity to Initialization**: Poor initialization can lead to suboptimal clusters.
- **Non-Convex Shapes**: Struggles with data that form non-convex clusters.

Application to the Diabetes Dataset

K-Means segments patients into groups based on features like Glucose, Age, BMI and some other columns. These clusters reveal patterns such as high-risk groups or patients with similar characteristics, aiding in targeted interventions. Apply clustering to group patients based on health metrics.

The dataset is converted from categorical values to integer value for clustering or machine learning using label encoder as shown below;

[55]:
```
# Label Encoding
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
label_encoder = LabelEncoder()
data_encoded_label = df.copy()
data_encoded_label['target_label_encoded'] = label_encoder.fit_transform(df['Outcome'])
data_encoded_label
```

[55]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	target_label_encoded
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1	1
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0	0
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32	1	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	1
...	...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0	0
764	2	122.0	70.0	27.0	125.0	36.8	0.340	27	0	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0	0
766	1	126.0	60.0	29.0	125.0	30.1	0.349	47	1	1
767	1	93.0	70.0	31.0	125.0	30.4	0.315	23	0	0

768 rows × 10 columns

LabelEncoder Processing of the dataset for clustering

[63]:
```
# K-means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
cluster = kmeans.fit_predict(df.drop('Outcome', axis=1))
df["cluster"] =  kmeans.fit_predict(df.drop('Outcome', axis=1))
df
```

[63]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	cluster
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1	1
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0	1
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32	1	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	1
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	1
...	...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0	1
764	2	122.0	70.0	27.0	125.0	36.8	0.340	27	0	1
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0	1
766	1	126.0	60.0	29.0	125.0	30.1	0.349	47	1	1
767	1	93.0	70.0	31.0	125.0	30.4	0.315	23	0	1

768 rows × 10 columns

K-Means Clustering on the dataset.

K-MEANS CLUSTERING ANALYSIS:

The algorithm has grouped the 768 rows of data into 3 clusters based on patterns in the feature space. Patients with similar feature values are assigned to the same cluster. These clusters may reflect underlying patterns or subgroups within the population, such as:

- High-risk patients.
- Moderate-risk patients.
- Low-risk patients.

For example:

Row 0: A patient with:

- Glucose = 148, BMI = 33.6, and Age = 50.
- True Outcome = 1 (diabetic).
- Assigned to **Cluster 1** by K-Means.

Row 2: A patient with:

- Glucose = 183, BMI = 23.3, and Age = 32.
- True Outcome = 1 (diabetic).

Plot Visualization Cluster of Age and Glucose

Below is the visualized view of patient cluster based on age and glucose;

```
[65]: # Visualize clusters (e.g., using Age and Glucose)
plt.figure(figsize=(8, 6))
plt.scatter(df['Age'], df['Glucose'], c=cluster, cmap='viridis', alpha=0.7)
plt.colorbar(label='Cluster')
plt.xlabel('Age')
plt.ylabel('Glucose')
plt.title('Patient Clusters')
plt.show()
```

Patient Cluster based on Glucose and Age

Interpretation of the Clusters

1. **Cluster 0 (e.g., Teal)**:
    - Represents a group of patients with moderate glucose levels and spans various age groups.
    - This cluster might indicate a group at low or moderate diabetes risk.
2. **Cluster 1 (e.g., Purple)**:
    - Patients with high glucose levels (above 150) and various ages.
    - Likely a group of patients at higher risk for diabetes.
3. **Cluster 2 (e.g., Yellow)**:
    - Patients with slightly elevated glucose levels compared to Cluster 0 but possibly lower than Cluster 1.
    - Could represent a middle-risk group.

Insights Derived from the Plot

- **Glucose Levels**: Glucose is a strong differentiator for clusters. Patients with higher glucose levels are grouped separately (e.g., Cluster 1).
- **Age Variability**: Age does not significantly differentiate clusters, as all clusters include patients from a wide age range.
- **Health Risk Segmentation**: This clustering could be useful for grouping patients into risk categories based on glucose levels, which is important for targeted healthcare interventions.

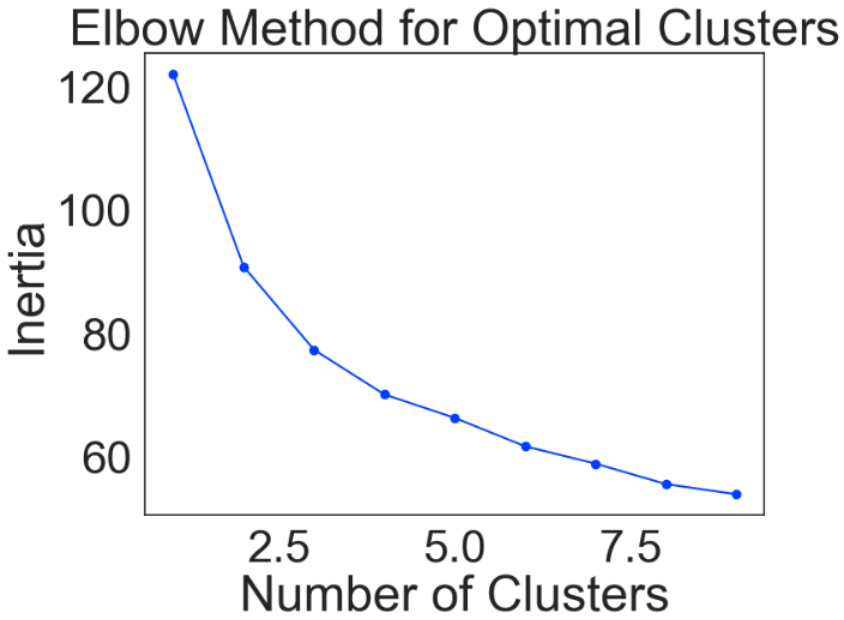
Evaluation of the clustering Method: Elbow Method

For this dataset analyses, The elbow method is a technique used to determine the optimal number of clusters for a clustering algorithm like K-Means. The graph you provided has two axes:

- **X-axis:** Number of clusters.
- **Y-axis:** Inertia (sum of squared distances of samples to their closest cluster center).

Below is the graph for the clustering analysis done on the dataset;

```python
# Plot the elbow curve
plt.figure(figsize=(8, 6))
plt.plot(cluster_range, inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal Clusters')
plt.show()
```



Key Points in the Diagram:

1. **Inertia decreases as the number of clusters increases:** This happens because as you add more clusters, data points are closer to their assigned cluster centers, reducing the inertia.
2. **The "elbow" point:** The "elbow" in the graph is where the rate of decrease in inertia slows down significantly. It marks the optimal trade-off between minimizing inertia and avoiding too many clusters. Beyond this point, adding more clusters doesn't substantially improve the fit but increases model complexity.
3. **Identifying the elbow:** In the graph, the elbow appears around 3 to 4 clusters, suggesting that this is the optimal number of clusters for the dataset.

Using this number of clusters balances simplicity and effective grouping in the data.

Reflection View

*Applying **Random Forest classification** and **clustering** techniques in healthcare has profound implications, especially when combined, as they provide complementary insights. Below is a discussion of their application in healthcare.*

Patient Diagnosis and Outcome Prediction

Random Forest Classification:

Use: Predict whether a patient has a specific disease or condition based on clinical data (e.g., diabetes, heart disease).

Example: Using patient features like glucose levels, BMI, and age to predict whether they are at risk of diabetes.

Real World Example: Diabetes Prevention Programs (DPP):

- ✓ Programs like the **CDC Diabetes Prevention Program** in the U.S. could benefit from clustering to identify high-risk communities and classification models to prioritize individuals for intervention.

Implications:

- ✓ Improves diagnostic accuracy by handling non-linear relationships and interactions between features.
- ✓ Provides feature importance insights to prioritize critical health markers, such as glucose levels.

Advantage: Robust to missing data and works well with high-dimensional data.

Clustering:

Use: Group patients into clusters based on similar risk profiles or symptoms to identify undiagnosed conditions or disease subtypes.

Example: Clustering diabetes patients into different groups based on glucose, BMI, and age to identify patients at high risk for complications.

Implications:

- ✓ Identifies hidden patterns in patient data, aiding personalized treatment.
- ✓ Enables targeting of high-risk groups for early intervention.

Combined Use of Random Forest and Clustering in Healthcare

By combining these techniques, healthcare providers can:

- ✓ Use **clustering** to segment the population and understand broader patterns.
- ✓ Apply **classification** models to make predictions for individual patients within those segments.
- ✓ Enhance decision-making for personalized medicine, resource allocation, and public health interventions.

REFERENCES

- Hastie, T., Tibshirani, R., and Friedman, J. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edn. New York: Springer.
- James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) *An Introduction to Statistical Learning with Applications in R*. New York: Springer.
- Bishop, C.M. (2006) *Pattern Recognition and Machine Learning*. New York: Springer.
- Han, J., Kamber, M., and Pei, J. (2011) *Data Mining: Concepts and Techniques*. 3rd edn. Burlington: Morgan Kaufmann.
- Hosmer, D.W., Lemeshow, S., & Sturdivant, R.X. (2013). *Applied Logistic Regression.* 3rd ed. New York: Wiley.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281–297). Berkeley: University of California Press.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.
- Reynolds, D. A. (2009). Gaussian Mixture Models. *Encyclopedia of Biometrics*, 659-663.