DATA MINING AND ITS APPLICATIONS IN HEALTHCARE

An Introduction to Data Mining in the Healthcare Sector

What is Data Mining?

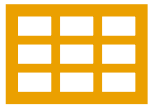
Data mining is the process of discovering patterns and useful information from large datasets.

It uses statistics, machine learning, and AI techniques to analyze data and extract valuable insights. This process helps in making informed decisions, predicting trends, and identifying relationships within the data.

DATA MINING PROCESS

1. Data Collection: Medical records, lab results, imaging data, wearable devices

2. Data Preprocessing: Cleaning and organizing data



3. Data Transformation: Converting data into a suitable format

4. Data Mining Techniques: Applying algorithms to extract insights

5. Interpretation & Evaluation: Understanding results

Take Note

Different data mining processing models will have different steps, though the general process is usually pretty similar. For example, the Knowledge Discovery Databases model has nine steps, the CRISP-DM model has six steps, and the SEMMA process model has five steps.

WHAT ARE KDD, CRISP-DM AND SEMMA?

Methodologies or **frameworks** developed to guide the **data mining** or **data science** process — meaning how you take raw data and turn it into useful knowledge or models.

KDD – KNOWLEDGE DISCOVERY DATA

- **What it is?:** A broad process of discovering useful knowledge from data.
- **Stages:** Typically includes:

Data selection → Data preprocessing → Data transformation → Data mining (actual pattern finding) → Interpretation/evaluation

- **What does it focus?:** It sees data mining as just one step in a bigger process of "knowledge discovery".
- **Example:**

Hospital patient records ⟶ You clean it, select only diabetic patients, transform it into useful features, mine for patterns like "people over 60 are more at risk," and finally conclude insights.

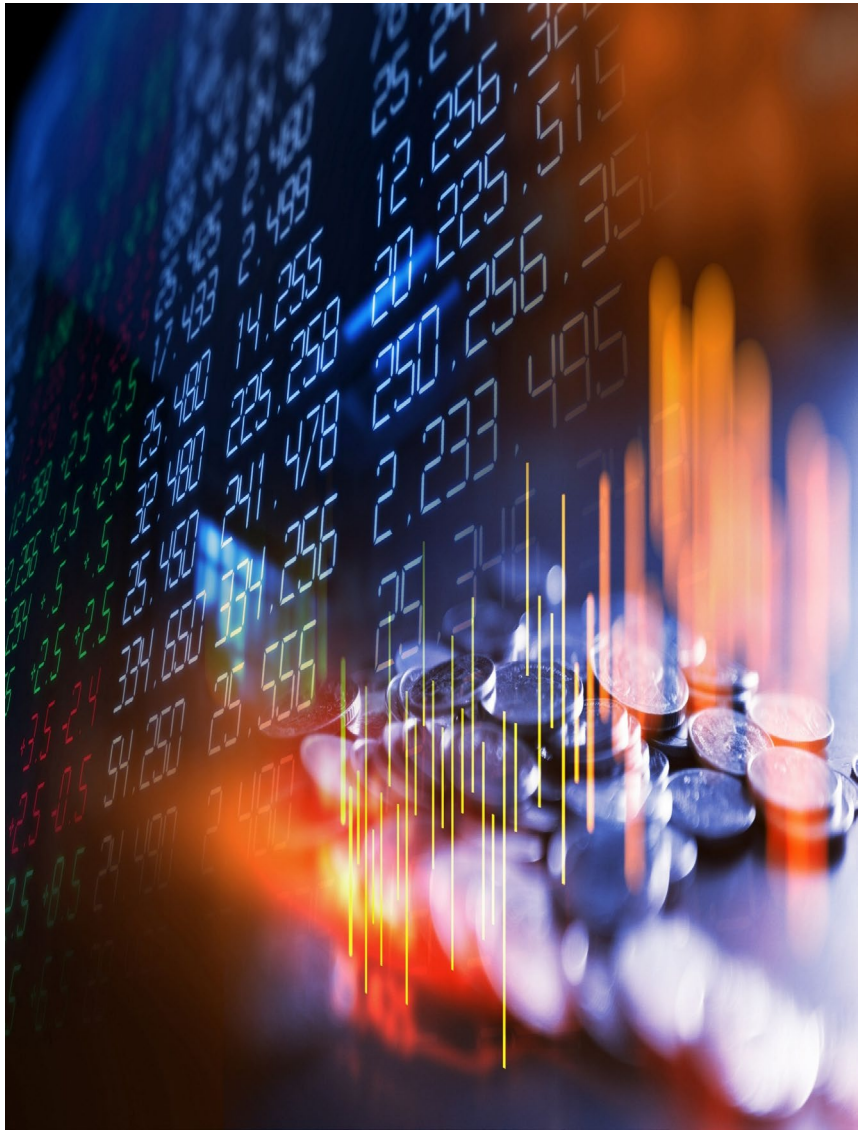
CRISP-DM (CROSS-INDUSTRY STANDARD PROCESS FOR DATA MINING)

- **What it is?:** A structured, practical guide used across industries for doing data mining projects.
- **Stages:**

Business Understanding → Data Understanding → Data Preparation → Modeling → Evaluation → Deployment

- **What does it focus?:** Very business-oriented. It's about solving a business problem using data, not just analyzing data for fun.
- **Example:**

If a supermarket wants to predict which customers will leave, CRISP-DM guides you: first understand the business problem, gather and prepare customer data, build predictive models, evaluate them, and finally deploy the model into action.



SEMMA (Sample, Explore, Modify, Model, Assess)

- **What it is?:** A data-driven approach developed by SAS Institute for building predictive models.
- **Stages:**

Sample the data → Explore the data → Modify the data → Model the data → Assess the model

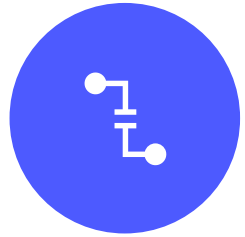
- **What does it focus?:**
  - More technical and model-focused than business-focused.
  - It mainly cares about data preparation and modeling (less emphasis on understanding business).
- **Example:**

You have a telecom company data → You sample a subset, explore patterns, modify variables (e.g., normalize or transform features), build models (e.g., decision trees, neural nets), and assess their accuracy.

DATA MINING FRAMEWORK

DATA INTEGRATION: COMBINING DATA FROM MULTIPLE SOURCES

DATA SELECTION: FILTERING RELEVANT DATA



DATA PREPROCESSING: HANDLING MISSING VALUES, NORMALIZATION

DATA TRANSFORMATION: CONVERTING RAW DATA INTO FEATURES

PATTERN EVALUATION: EXTRACTING USEFUL KNOWLEDGE

KNOWLEDGE PRESENTATION: VISUALIZING INSIGHTS

KEY DATA MINING TECHNIQUES

- Classification: Predicting disease diagnosis
- Clustering: Grouping similar patient cases
- Association Rule Mining: Finding relationships in medical data
- Regression Analysis: Predicting patient readmission rates
- Anomaly Detection: Identifying unusual patterns
- Neural Networks, Decision Trees, SVMs

CLASSIFICATION

WHAT IS CLASSIFICATION?

- Classification is a type of problem in data mining, machine learning and statistics where the goal is to **predict a label or category** for a given input based on its features.
- Unlike regression, which deals with continuous values, classification assigns observations to discrete categories or labels.
- **Category of Classification**: Binary Classification, Multiclass Classification and Multilabel Classification
- **Popular algorithms**: Decision Trees, Random Forests, Logistic Regression, Support Vector Machines (SVM), Neural Networks.

EXAMPLE OF CLASSIFICATION PROBLEM IN HEALTHCARE

1. Binary Classification

Problem: Predict whether a patient will **develop heart disease** within 5 years.

Classes: Yes (will develop heart disease), **No** (will not develop heart disease)

Input: Cholesterol levels, smoking status, physical activity, blood pressure, etc.

Output: Yes or No prediction.

EXAMPLE OF CLASSIFICATION PROBLEM

2. Multiclass Classification

Problem: Predict the stage of cancer (based on imaging and blood markers).

Classes: Stage I, Stage II, Stage III, Stage IV

Input: MRI features, tumor markers, patient's age, etc.

Output: One predicted cancer stage.

EXAMPLE OF CLASSIFICATION PROBLEM

3. Multilabel Classification

Problem: Predict complications after surgery.

Possible complications (labels):
Wound infection
Blood clot (deep vein thrombosis)
Pneumonia
Kidney failure

✅ A patient can have none, one, or several complications at the same time.

LOGISTIC REGRESSION ANALYSIS – BINARY CLASSIFICATION

Why Linear Regression is Not Suitable for Classification?

*Although linear regression is commonly used for prediction, it **is not ideal for binary outcomes**.*

1. Predicted values not bounded

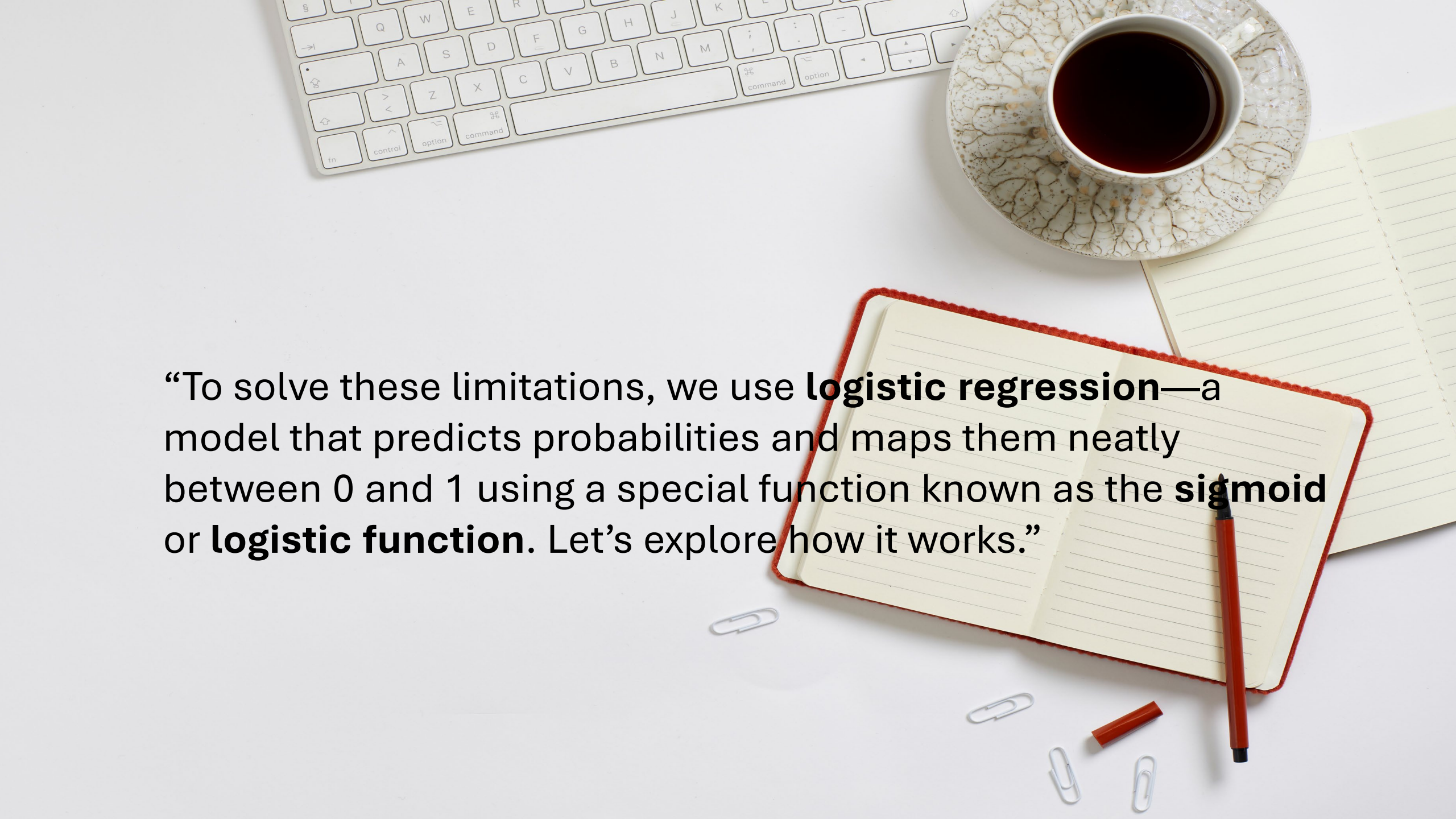
- Linear regression can output any real number: from $-\infty$ to $+\infty$.
- But probabilities must be between 0 and 1.
- This can lead to **illogical predictions** (e.g., a -0.3 probability of having a disease).

2. Violation of assumptions

- Linear regression assumes **homoscedasticity** (constant variance) and **normality of residuals**, which are **not valid** when Y is binary.

3. Poor decision boundaries

- In a binary setting, we need a model that **decides clearly** between two classes.
- Logistic regression uses a **sigmoid function** to create a **non-linear, S-shaped boundary**, better for probability-based decision making.



“To solve these limitations, we use **logistic regression**—a model that predicts probabilities and maps them neatly between 0 and 1 using a special function known as the **sigmoid** or **logistic function**. Let’s explore how it works.”

CONCEPT OF LOGISTIC REGRESSION

Logistic regression is a statistical method used to **predict the probability** that a given input belongs to a **particular class**.

Specifically, in **binary classification**, the goal is to estimate the probability that the dependent variable Y equals **1** (i.e., the event of interest), given a set of independent variables X.

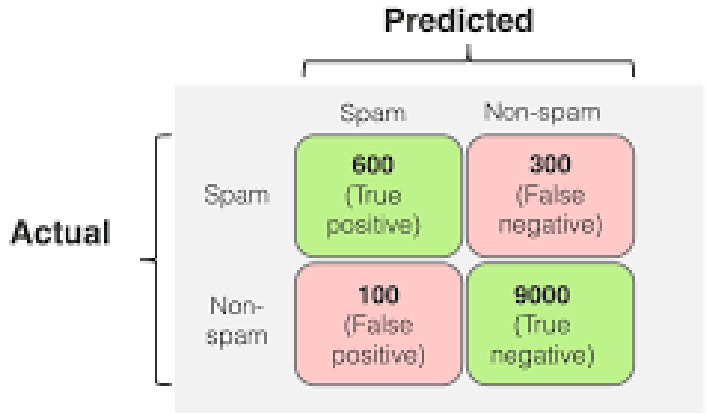
$$P(Y = 1 \mid X) = \text{some value between 0 and 1}$$

LOGISTIC REGRESSION MODEL EVALUATION

Once you've trained a logistic regression model, you need to **evaluate how well it performs**, especially on **unseen data**. Since logistic regression is typically used for **classification**, we use **classification metrics** rather than regression metrics.

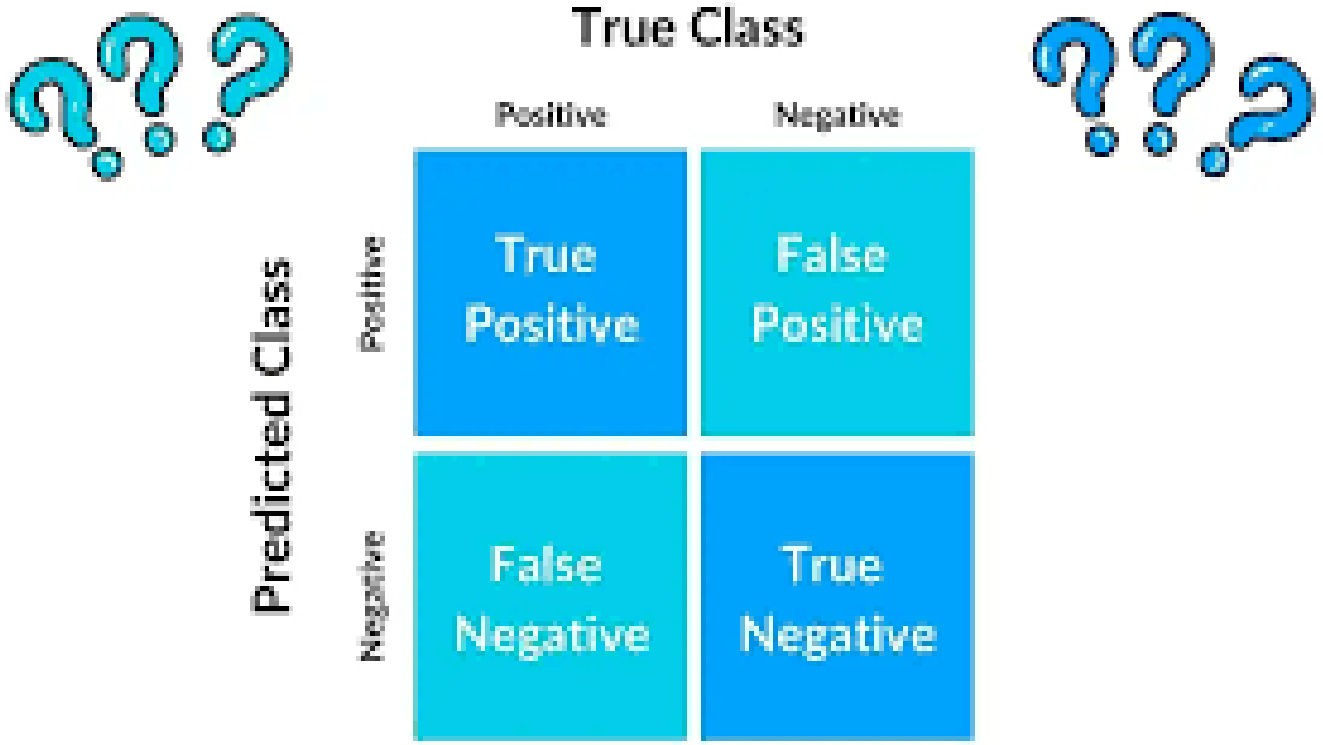
The Metrices are:

1. Confusion Matrix
2. Accuracy
3. Precision and Recall
4. F1 Score
5. ROC Curve and AUC



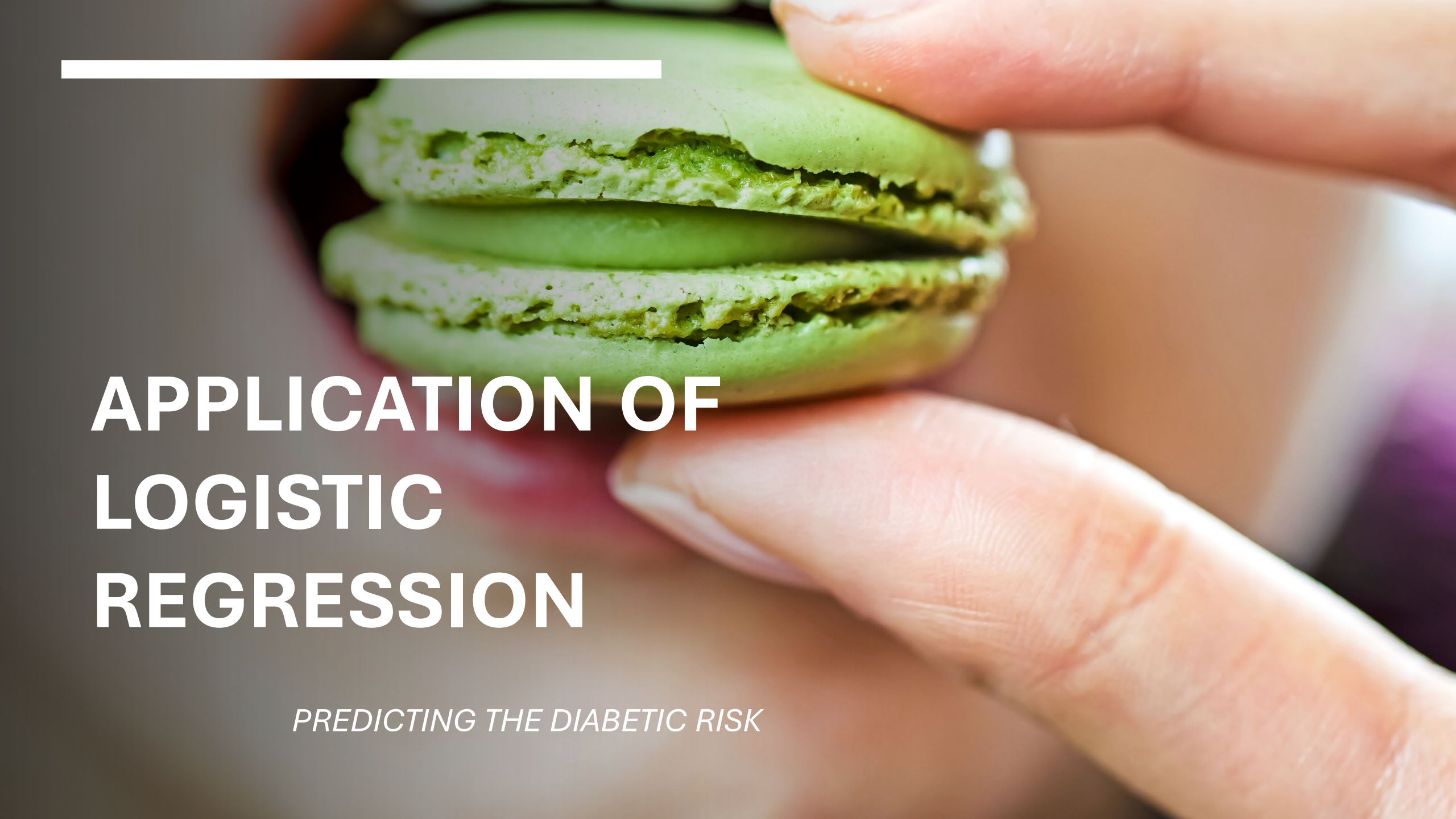
CONFUSION MATRIX

Summarizes the results of predictions compared to actual values.



BINARY CLASSIFICATION

Metric	In Simple Words	When to Focus On It
Accuracy	How often it's correct	When classes are balanced
Precision	Of predicted Yes, how many are correct	When false positives are bad (e.g. alarms)
Recall	Of real Yes, how many did we find?	When false negatives are dangerous (e.g. diseases)
F1-Score	Balance of precision and recall	When you want a fair trade-off
AUC	How well the model separates classes	When comparing different models



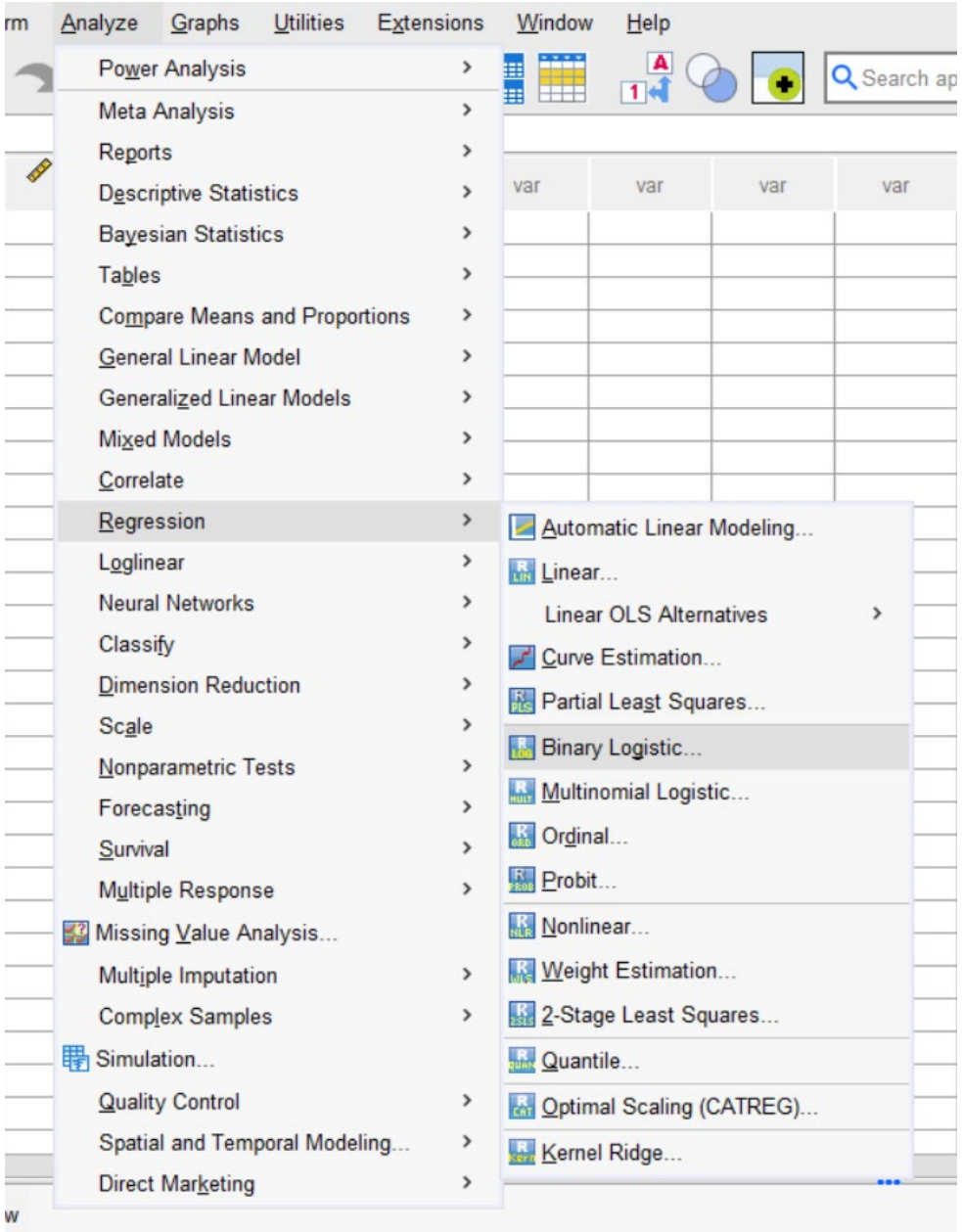
APPLICATION OF LOGISTIC REGRESSION

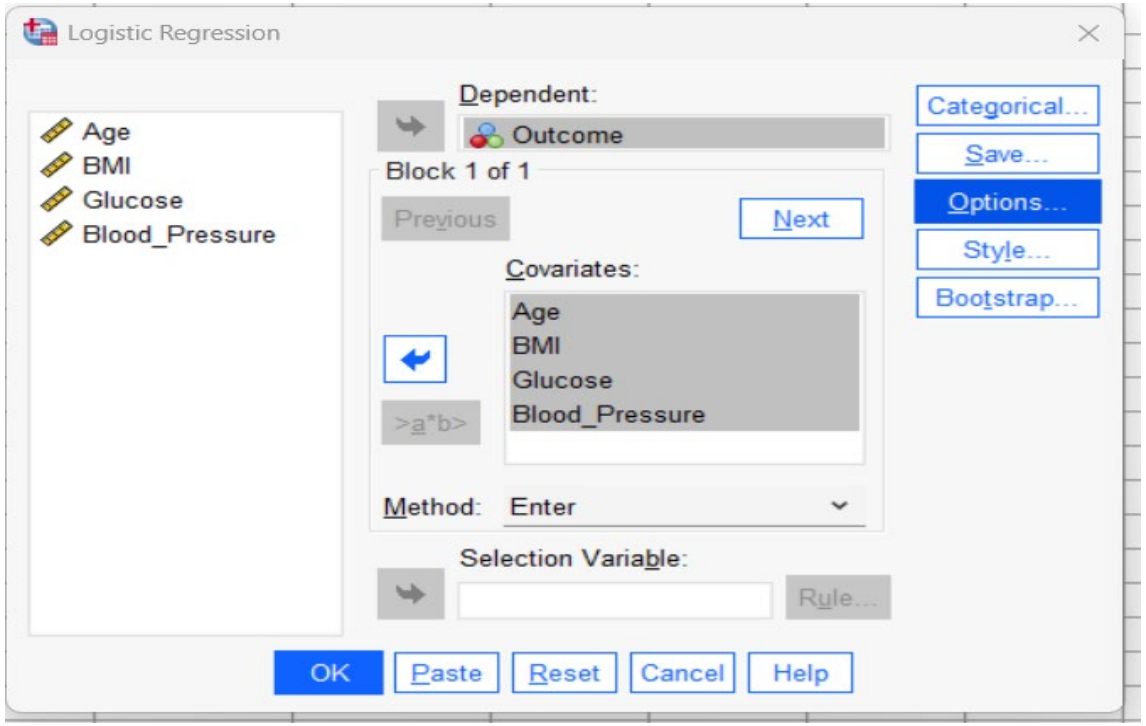
PREDICTING THE DIABETIC RISK

- **DATA SET:** healthcare_noisy_dataset.xlsx
- **FEATURES:** Age, BMI, Glucose Level, Blood Pressure
- **ANALYSIS:** Predicting number of patient with Diabetic. (YES, NO)
- **METHOD:** Logistic Regression

1

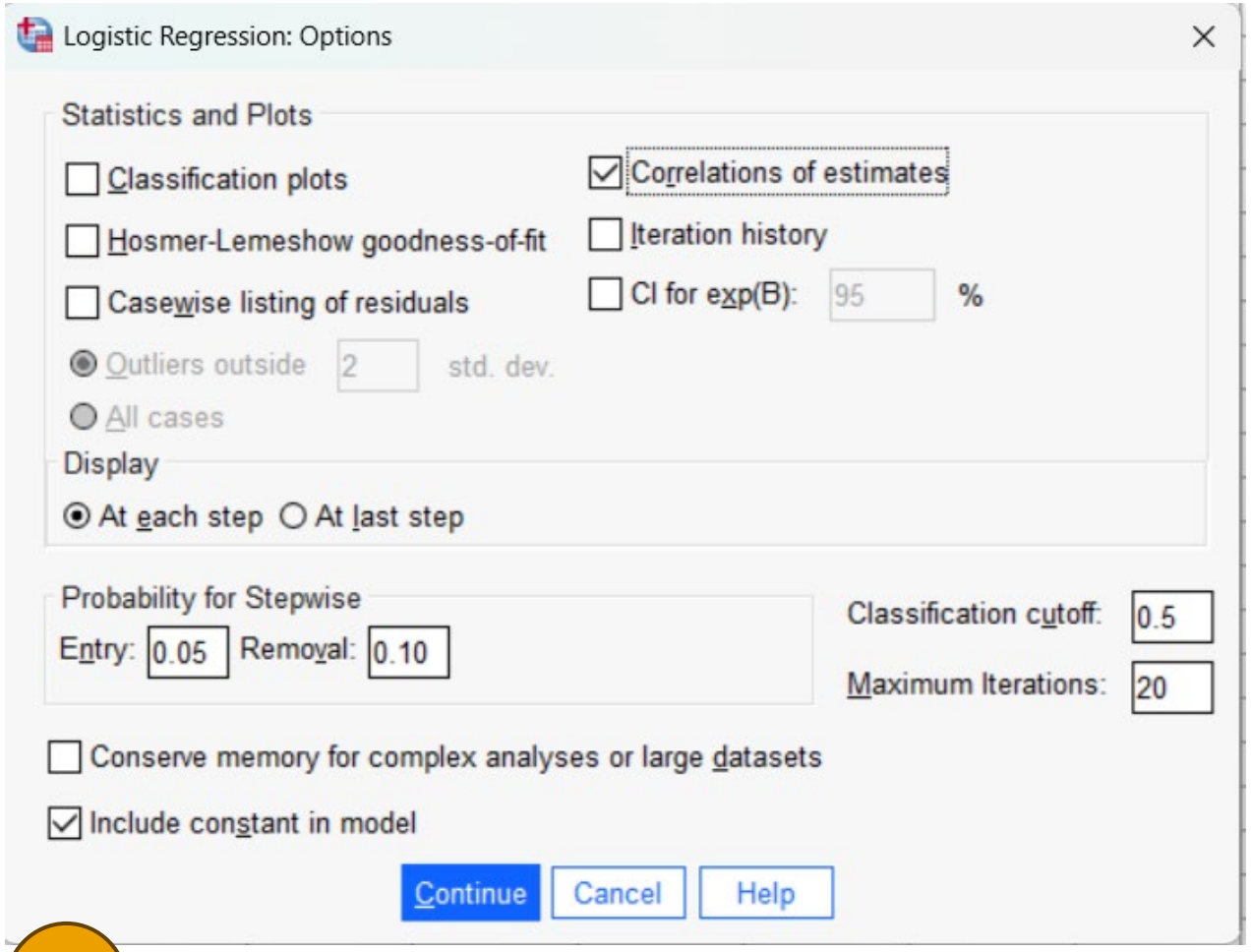
Analyze → Regression → Binary Logistic





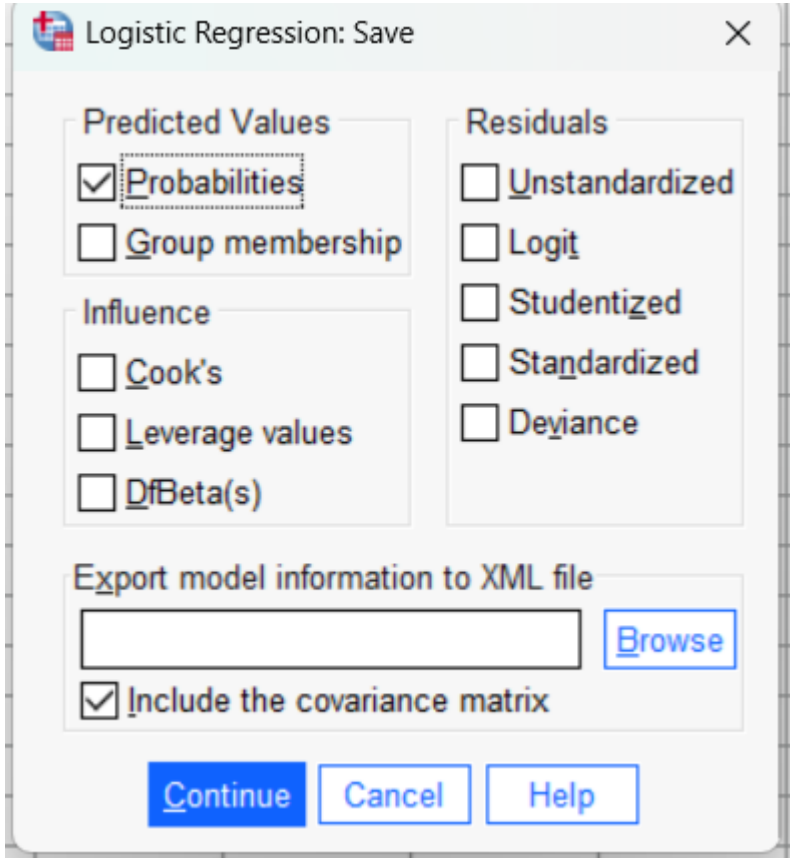
2

Variable Selection
Make sure Dependent variable should be value 0 and 1



3

Click **Options** (optional) to:
• Get **confidence intervals**, **classification plots**, **Hosmer-Lemeshow test**, etc.



4

Click **Save** (optional) to:
• Get **probabilities (Useful input for ROC)**

Interpret the Output (LOGISTIC REGRESSION)
Here's how to read the important ones:

📌 ***Variables in the Equation*** **(Table)**
• **B (coefficients)**: The effect size of each predictor.
• **Exp(B)**: The **odds ratio**. If Exp(B) > 1, the variable increases the odds of the outcome.
• **Sig. (p-value)**: If < 0.05 → the variable is **statistically significant**.

📌 ***Model Summary***
• **-2 Log Likelihood**: Lower is better.
• **Cox & Snell R^2 / Nagelkerke R^2**: Similar to R^2 in linear regression (how well model explains outcome).

📌 ***Classification Table***
• Shows **overall accuracy**: how many cases were correctly classified (e.g., 85%).

📌 ***Hosmer-Lemeshow Test***
• If **p > 0.05**, model fit is **good**
(null hypothesis is predicted = observed).

RESULT AND SUMMARY - SPSS

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	67.514[a]	.247	.475

a. Estimation terminated at iteration number 7 because parameter estimates changed by less than .001.

Classification Table[a]

			Predicted		
			Outcome		Percentage Correct
	Observed		0	1	
Step 1	Outcome	0	9	9	50.0
		1	3	129	97.7
	Overall Percentage				92.0

a. The cut value is .500

BMI and Blood_Pressure is NOT the predictor.

Age and Glucose is the predictor in predicting the Diebetic.

Model performance is moderate with Pseudo R^2 = 0.475

Model performance Accuracy = 92%.

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Age	.069	.021	10.542	1	.001	1.072
	BMI	.084	.053	2.494	1	.114	1.087
	Glucose	.054	.014	15.007	1	<.001	1.056
	Blood_Pressure	.022	.016	1.853	1	.173	1.022
	Constant	-11.837	3.415	12.015	1	<.001	.000

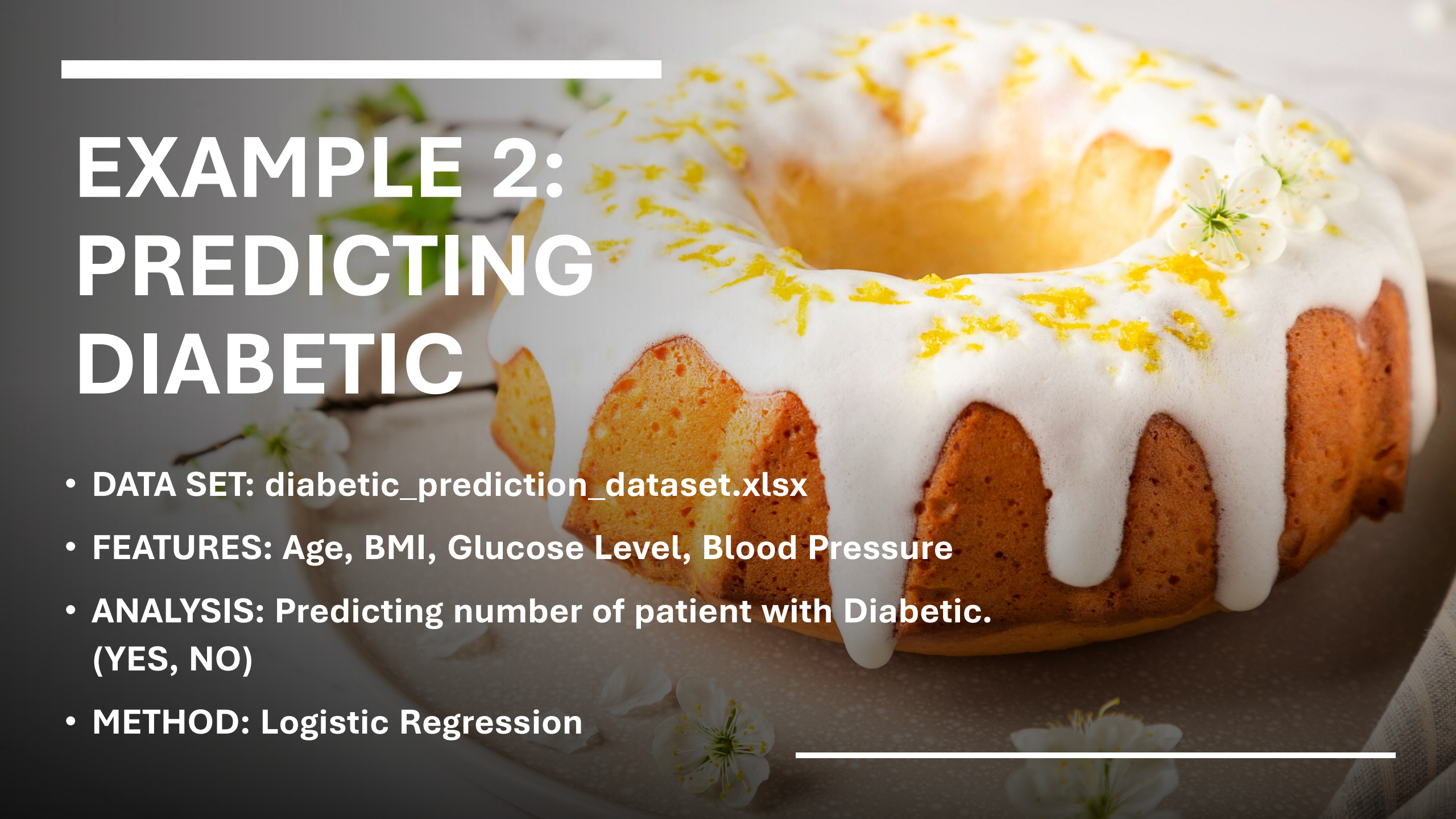
a. Variable(s) entered on step 1: Age, BMI, Glucose, Blood_Pressure.

True Negatives (TN): 9 — correctly predicted as no diabetes

False Positives (FP): 9 — wrongly predicted as having diabetes

False Negatives (FN): 3 — missed predictions of diabetes

True Positives (TP): 129 — correctly predicted as having diabetes



EXAMPLE 2: PREDICTING DIABETIC

- DATA SET: diabetic_prediction_dataset.xlsx
- FEATURES: Age, BMI, Glucose Level, Blood Pressure
- ANALYSIS: Predicting number of patient with Diabetic. (YES, NO)
- METHOD: Logistic Regression

RESULT AND SUMMARY - SPSS

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	102.970[a]	.576	.879

a. Estimation terminated at iteration number 10 because parameter estimates changed by less than .001.

Classification Table[a]

			Predicted		
			Outcome		Percentage Correct
	Observed		0	1	
Step 1	Outcome	0	376	12	96.9
		1	13	99	88.4
	Overall Percentage				95.0

a. The cut value is .500

BMI is the dominant predictor (coefficient is much larger).

Age, Glucose, and Blood Pressure still contribute but with smaller impacts.

Model performance is excellent: Pseudo $R^2 = 0.879$

Model performance Accuracy = 95%.

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Age	.105	.021	25.670	1	<.001	1.111
	BMI	1.349	.196	47.446	1	<.001	3.853
	Glucose	.062	.011	30.476	1	<.001	1.064
	Blood_Pressure	.058	.013	19.089	1	<.001	1.059
	Constant	-67.381	9.688	48.372	1	<.001	.000

a. Variable(s) entered on step 1: Age, BMI, Glucose, Blood_Pressure.

True Negatives (TN): 376 — correctly predicted as no diabetes

False Positives (FP): 12 — wrongly predicted as having diabetes

False Negatives (FN): 13 — missed predictions of diabetes

True Positives (TP): 99 — correctly predicted as having diabetes

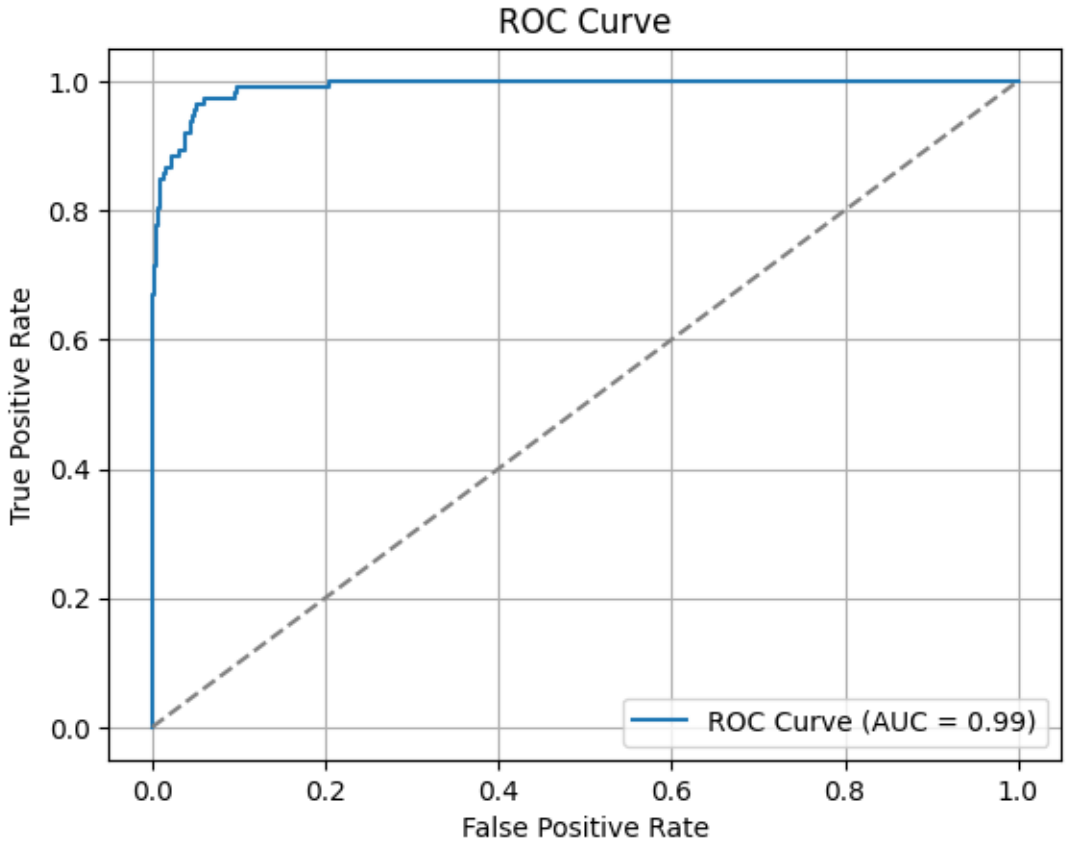
PYTHON CODE & RESULT – without learning

```
# STEP 1: Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,
classification_report, roc_curve, roc_auc_score

# STEP 2: Upload your Excel file in Colab
from google.colab import files
uploaded = files.upload()

# STEP 3: Load the dataset
df = pd.read_excel(next(iter(uploaded)))

# STEP 4: Define features and target
X = df[['Age', 'BMI', 'Glucose', 'Blood_Pressure']]
y = df['Outcome']

# STEP 5: Fit logistic regression on full dataset
model = LogisticRegression(max_iter=1000)
model.fit(X, y)

# STEP 6: Predict outcomes using the same data (like SPSS does)
y_pred = model.predict(X)
y_proba = model.predict_proba(X)[:, 1]

# STEP 7: Confusion matrix & classification report
conf_matrix = confusion_matrix(y, y_pred)
class_report = classification_report(y, y_pred)
roc_auc = roc_auc_score(y, y_proba)

print("Confusion Matrix:\n", conf_matrix)
print("\nClassification Report:\n", class_report)
print("ROC AUC Score:", round(roc_auc, 2))
```

```
# STEP 8: Plot ROC curve
fpr, tpr, thresholds = roc_curve(y, y_proba)
plt.figure()
plt.plot(fpr, tpr, label=f"ROC Curve (AUC =
{roc_auc:.2f})")
plt.plot([0, 1], [0, 1], linestyle="--", color='grey')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.grid(True)
plt.show()
```

PYTHON CODE & RESULT – without learning

```
Confusion Matrix:
 [[376  12]
 [ 13  99]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       388
           1       0.89      0.88      0.89       112

    accuracy                           0.95       500
   macro avg       0.93      0.93      0.93       500
weighted avg       0.95      0.95      0.95       500

ROC AUC Score: 0.99
```




MACHINE
LEARNING

WHAT IS MACHINE LEARNING

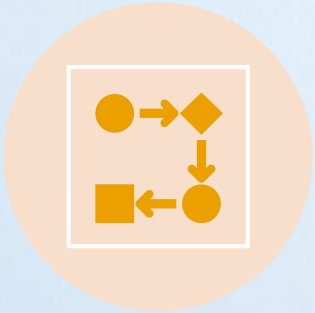
Definition:
Machine Learning is a method where computers learn patterns from data to make decisions or predictions **without being directly programmed**.

In simple terms:
It's like training a junior doctor — after seeing many patient cases, they learn how to diagnose new patients on their own.

HOW DOES MACHINE LEARNING WORKS?

Collect Data (e.g., patient records, lab results)

Train the Model (use past data to learn patterns)



Test the Model (see how well it works on new data)

Use the Model (make predictions in real-time)

SIMPLE ML MODEL YOU MIGHT HEAR

Model	How it works	Healthcare Example
Logistic Regression	Calculates risk based on factors	Predicts if a patient will get a disease
Decision Tree	Follows yes/no questions	Used in triage or diagnosis pathways
Neural Networks	Works like a simplified brain	Reads X-rays or MRIs to detect issues
k-Means Clustering	Groups data into similar clusters	Segments patients for targeted care

PYTHON CODE & RESULT – logistic regression with learning

```
# STEP 1: Import libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
from google.colab import files

# STEP 2: Upload your Excel file
uploaded = files.upload()

# STEP 3: Load the dataset
df = pd.read_excel(next(iter(uploaded)))

# STEP 4: Define features and target variable
X = df[['Age', 'BMI', 'Glucose', 'Blood_Pressure']]
y = df['Outcome']

# STEP 5: Split the data (70% training, 30% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# STEP 6: Train the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
# STEP 7: Predict on test data
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]


# STEP 8: Evaluation metrics
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_proba)

print("🔹 Confusion Matrix:\n", conf_matrix)
print("\n🔹 Classification Report:\n", class_report)
print("🔹 ROC AUC Score:", round(roc_auc, 2))

# STEP 9: Plot ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
plt.figure()
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], linestyle="--", color='grey')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve (Test Data)")
plt.legend()
plt.grid(True)
plt.show()
```

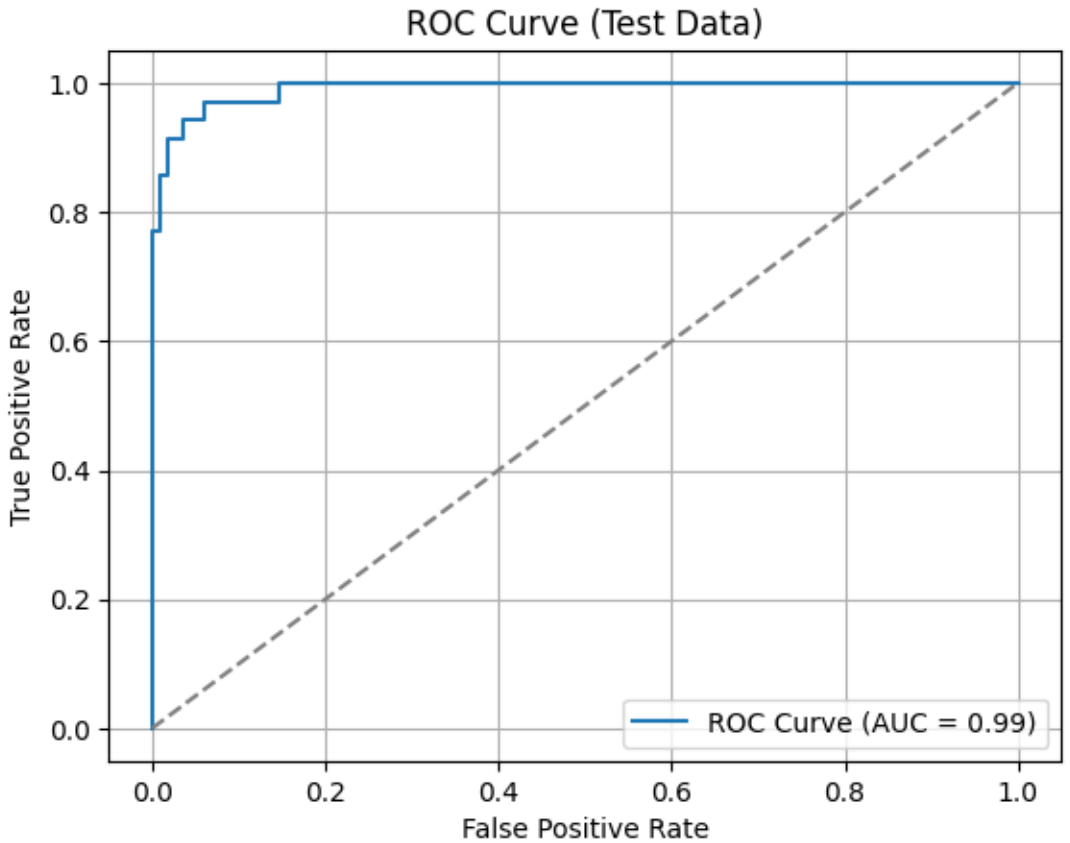
- Confusion Matrix:

```
[[112   3]
[  3  32]]
```

- Classification Report:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	115
1	0.91	0.91	0.91	35
accuracy			0.96	150
macro avg	0.94	0.94	0.94	150
weighted avg	0.96	0.96	0.96	150

- ROC AUC Score: 0.99



MODEL COMPARISON: LOGISTIC REGRESSION VS RANDOM FOREST VS SUPORT VECTOR MACHINE (SVM) AND K-NEAREST NEIGHBOUR (KNN)

PYTHON CODE

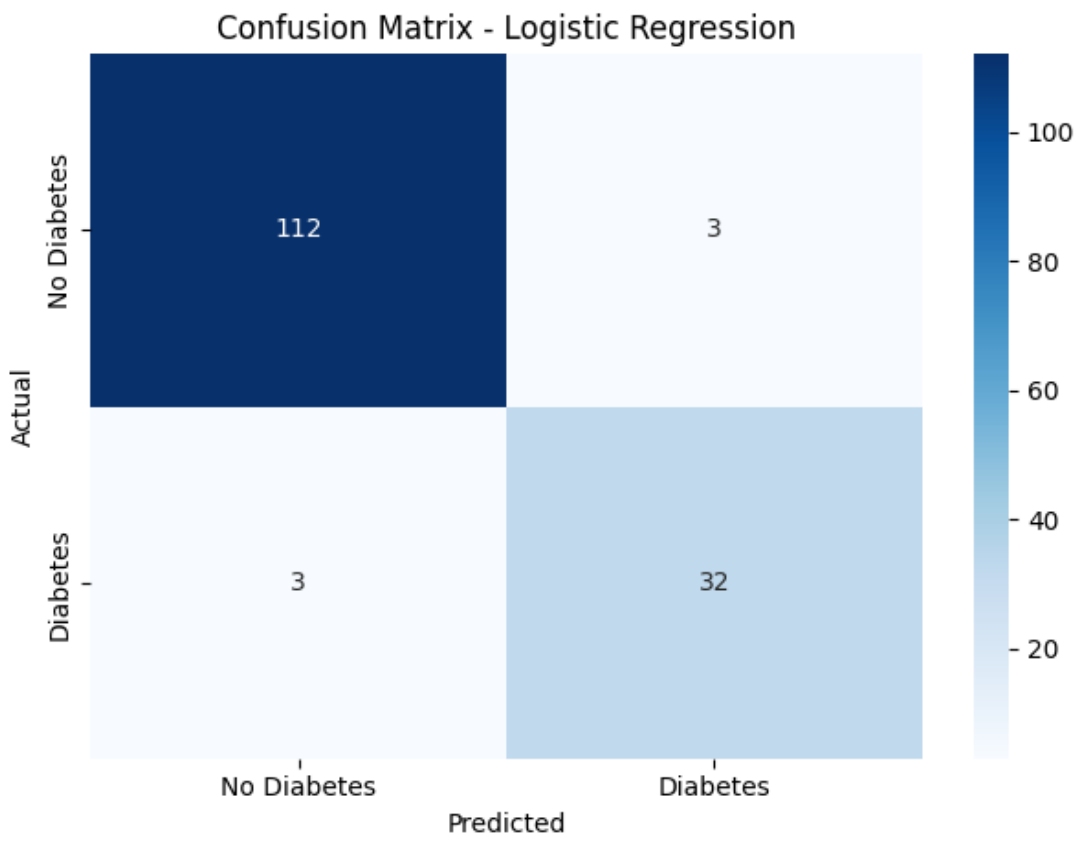
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, roc_curve, auc, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# STEP 2: Upload your Excel file in Colab
from google.colab import files
uploaded = files.upload()

# STEP 3: Load the dataset
df = pd.read_excel(next(iter(uploaded)))

# Define features and target
X = df[['Age', 'BMI', 'Glucose', 'Blood_Pressure']]
y = df['Outcome']

# Split the data (70% training, 30% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Define models
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Support Vector Machine": SVC(probability=True),
    "K-Nearest Neighbors": KNeighborsClassifier()
}
```

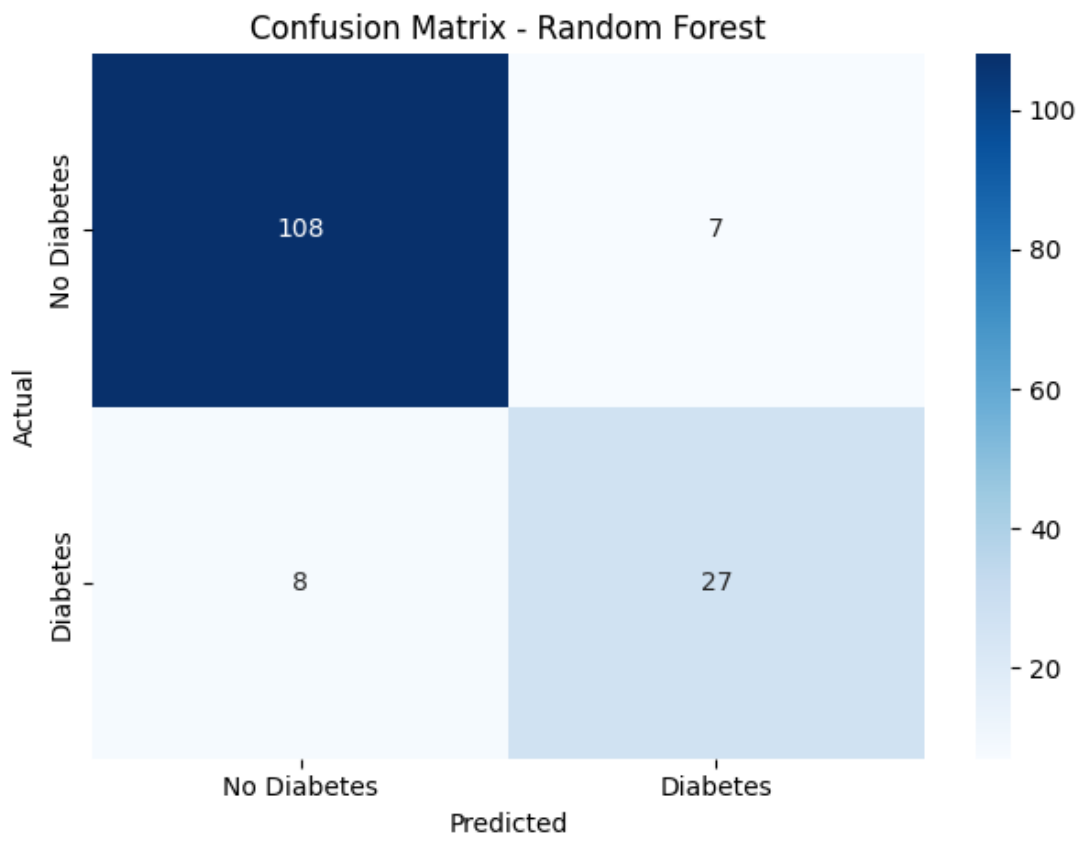
PYTHON CODE

```
# Train and evaluate models
results = []
plt.figure(figsize=(10, 7))

for name, model in models.items():
    model.fit(X_train, y_train)
    y_proba = model.predict_proba(X_test)[:, 1]
    y_pred = model.predict(X_test)

    # Accuracy
    acc = accuracy_score(y_test, y_pred)
    results.append({"Model": name, "Accuracy": round(acc, 3)})

    # ROC Curve
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc:.3f})')

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure()
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['No Diabetes', 'Diabetes'],
                yticklabels=['No Diabetes', 'Diabetes'])
    plt.title(f'Confusion Matrix - {name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.tight_layout()
    plt.show()
```

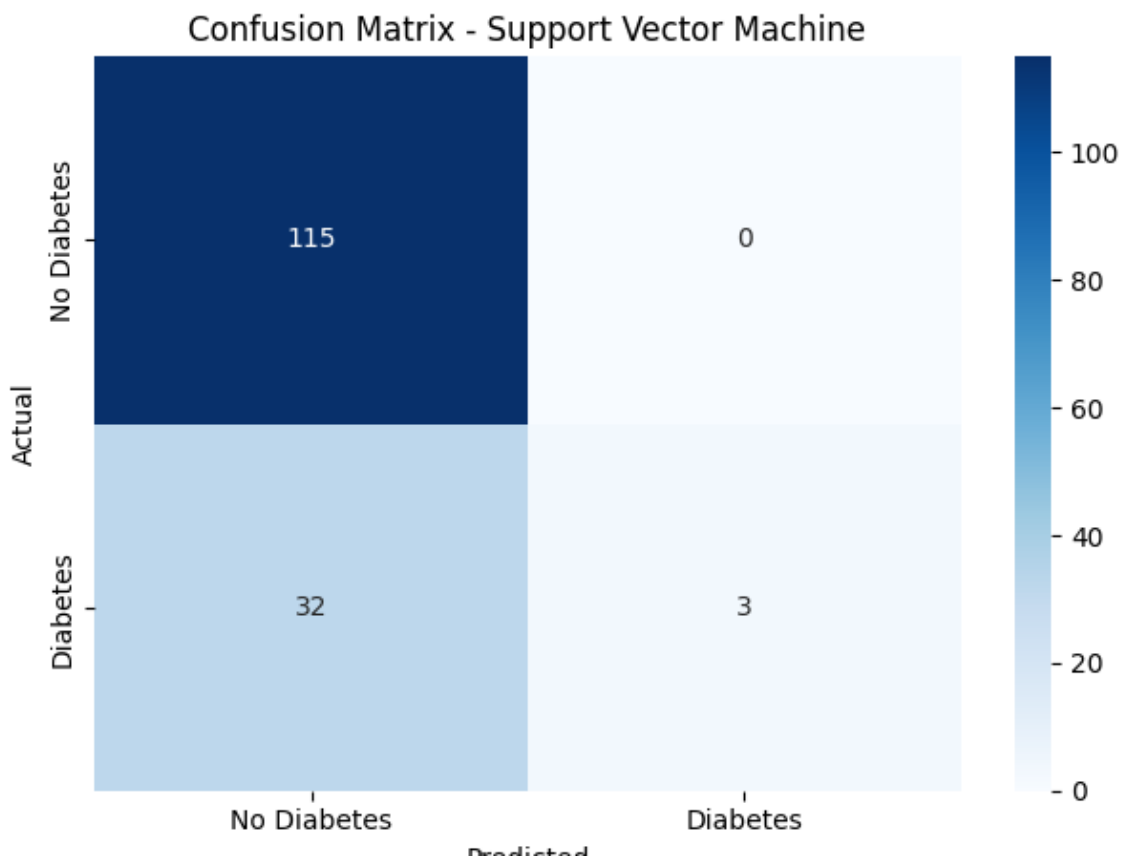
PYTHON CODE

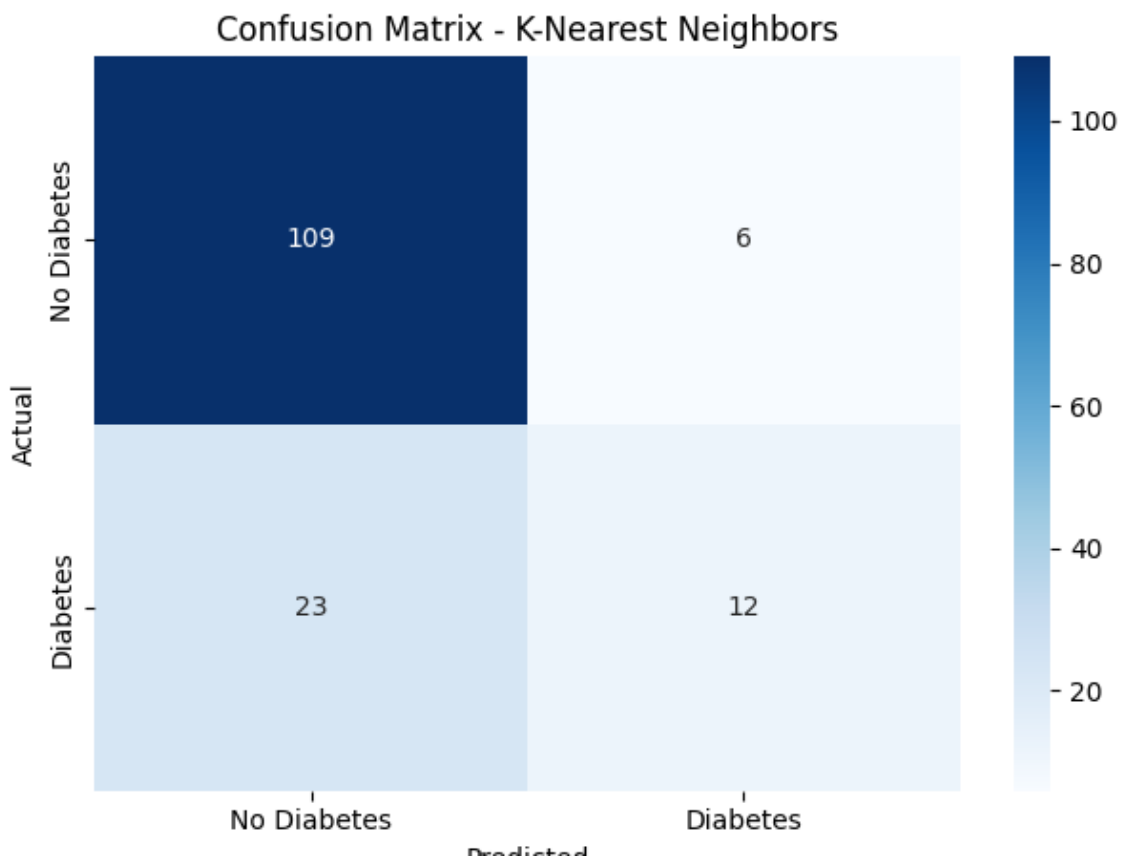
```
# Display results
results_df = pd.DataFrame(results)
print("Model Comparison Results:")
print(results_df)

# Finalize ROC plot
plt.figure(figsize=(10, 7))
for name, model in models.items():
    model.fit(X_train, y_train)
    y_proba = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc:.3f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend(loc='lower right')
plt.tight_layout()
plt.show()
```





OUTPUT AND RESULT – CONFUSION MATRIX

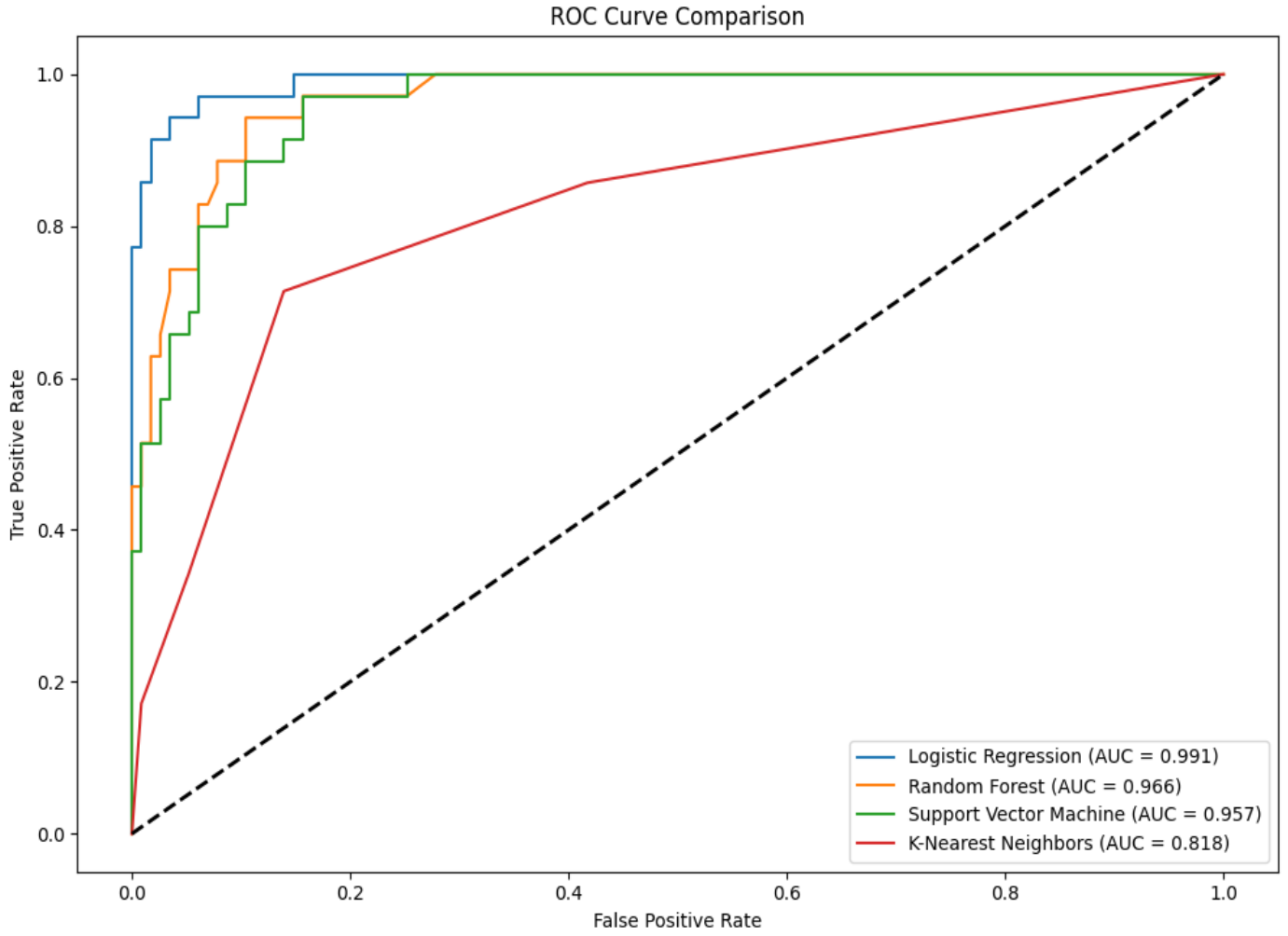




OUTPUT AND RESULT – CONFUSION MATRIX

Model Comparison Results:

	Model	Accuracy
0	Logistic Regression	0.960
1	Random Forest	0.900
2	Support Vector Machine	0.787
3	K-Nearest Neighbors	0.807



OUTPUT AND RESULT – ROC ANALYSIS

FINAL TAKEAWAYS: WHAT YOU SHOULD REMEMBER ABOUT MACHINE LEARNING

1. **Machine Learning is a Tool, Not a Replacement**
   ML doesn't replace doctors, nurses, or healthcare staff. It helps them do their job better. Think of it like a stethoscope or MRI machine—tools that enhance, not replace, human expertise.

2. **Your Clinical Judgement is Still Essential**
   Even if a model says "this patient is high-risk," it's up to the healthcare professional to interpret the result in the context of the patient's symptoms, history, and values. ML provides a second opinion, not a final verdict.

3. **ML is Only as Good as the Data**
   If the model is trained on poor, biased, or outdated data, its recommendations can be wrong or unfair. Healthcare professionals must ensure the data reflects diverse and real-world populations.

4. **You Don't Need to Be a Programmer to Benefit from ML**
   You don't need to write code or know complex math. What matters is understanding **what the model does**, **why it's being used**, and **how to interpret its outputs**. This allows you to collaborate better with data scientists and use ML tools effectively.

THANK YOU