1) IMPORT LIBRARIES

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

2) LOAD THE DATASET

```
from google.colab import files

uploaded = files.upload()
```

Choose files   diabetes.csv
  • **diabetes.csv**(text/csv) - 23873 bytes, last modified: 15/05/2025 - 100% done
    Saving diabetes.csv to diabetes.csv

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

```
print(df.head())
```

```
   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

   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
```

3) EXPLORE THE DATA

```
print(df.info())
```

```
print(df.describe())
```

```
# Correlation heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```
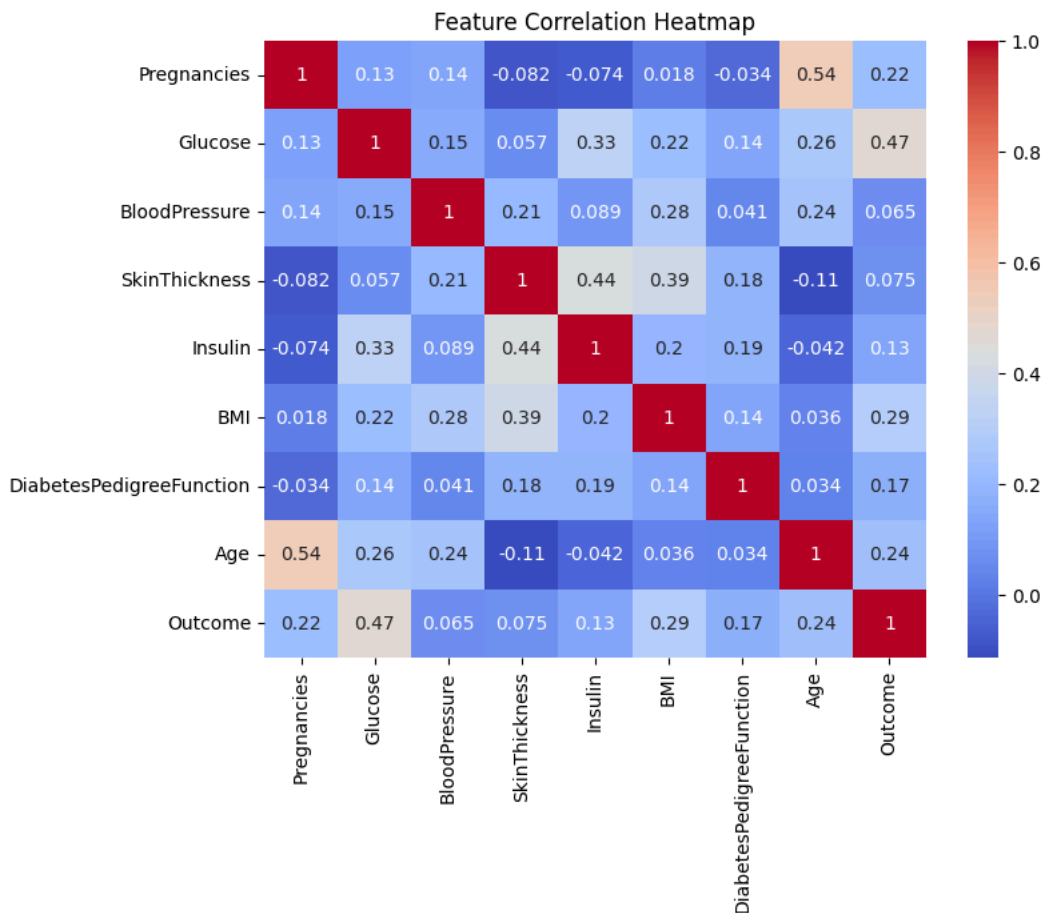
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 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
None
       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
```
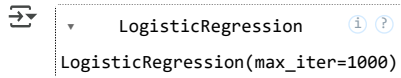

Feature Correlation Heatmap

5) SPLIT THE DATA (TRAIN/TEST)

```python
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, random_state=42)
```

6) TRAIN THE MODEL

```python
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
    ▾   LogisticRegression      ⓘ ⑦
  LogisticRegression(max_iter=1000)
```

7) MAKE PREDICTIONS

```python
# Predict on test data
y_pred = model.predict(X_test)
```

8) EVALUATE THE MODEL

```python
# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap="Blues", fmt="d")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Accuracy: 0.7467532467532467

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.79      0.80        99
           1       0.64      0.67      0.65        55

    accuracy                           0.75       154
   macro avg       0.73      0.73      0.73       154
weighted avg       0.75      0.75      0.75       154
```

Confusion Matrix