1) IMPORT LIBRARIES

```
!pip install gradio
```

Show hidden output

```
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.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

2) LOAD THE DATASET

```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
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

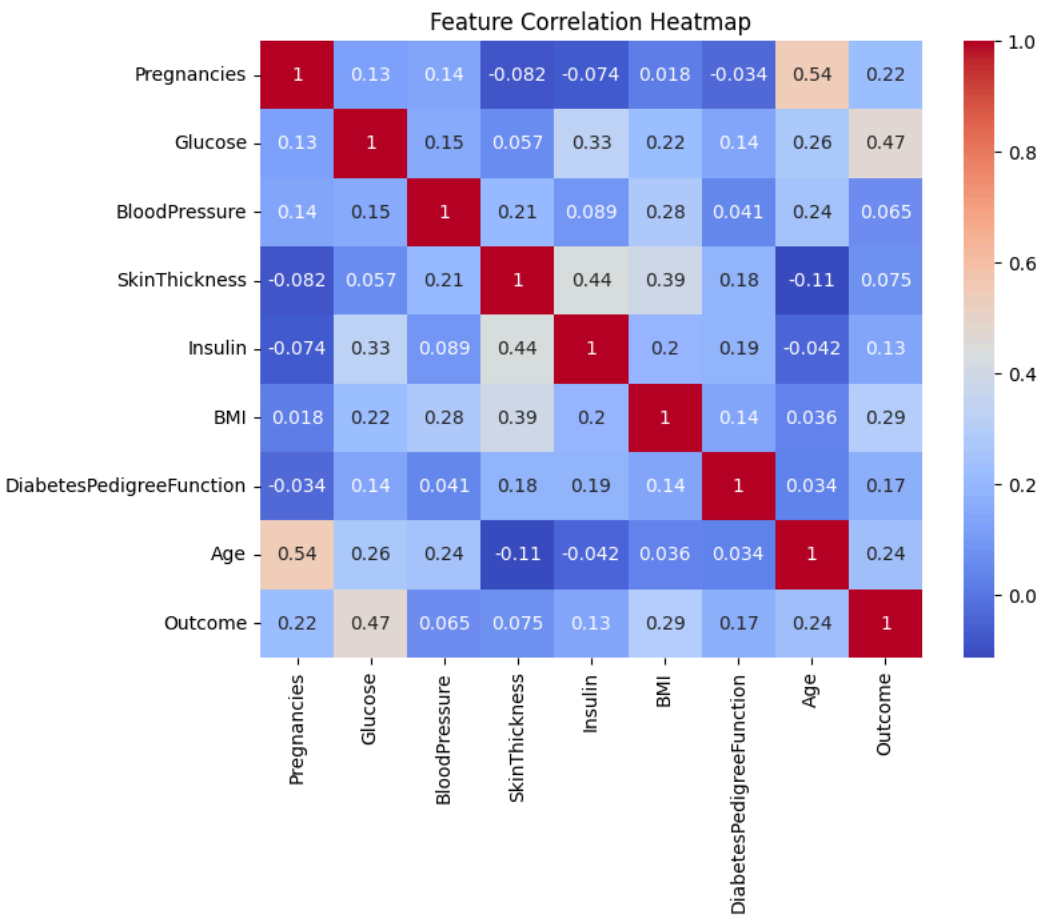
```
print(df.info())
```

```
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
```



5) SPLIT THE DATA (TRAIN/TEST)

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

X = df[['Age', 'Glucose', 'BloodPressure']]
y = df['BMI']


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Training features:\n", X_train)
print("Test features:\n", X_test)
print("Training labels:\n", y_train)
print("Test labels:\n", y_test)
```

```
Training features:
      Age  Glucose  BloodPressure
60    21       84              0
618   50      112             82
346   22      139             46
294   65      161             50
231   46      134             80
..   ...      ...            ...
71    26      139             64
106   27       96            122
270   38      101             86
435   29      141              0
102   21      125             96

[614 rows x 3 columns]
Test features:
      Age  Glucose  BloodPressure
668   43       98             58
324   21      112             75
624   21      108             64
690   34      107             80
473   50      136             90
..   ...      ...            ...
355   49      165             88
534   24       77             56
344   57       95             72
296   29      146             70
462   39       74             70

[154 rows x 3 columns]
Training labels:
 60      0.0
618    28.2
346    28.7
294    21.9
231    46.2
       ...
71     28.6
106    22.4
270    45.6
435    42.4
102    22.5
Name: BMI, Length: 614, dtype: float64
Test labels:
 668    34.0
324    35.7
624    30.8
690    24.6
473    29.9
       ...
355    30.4
534    33.3
344    36.8
296    28.0
462    35.3
Name: BMI, Length: 154, dtype: float64
```

6) REGRESSION MODEL

6) [A] LOGISTIC REGRESION MODEL

```
# Train logistic regression model
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)

# Predict
log_preds = log_model.predict(X_test)

# Evaluate
print("Logistic Regression Accuracy:", accuracy_score(y_test, log_preds))
print("Classification Report:\n", classification_report(y_test, log_preds))
```

```
Logistic Regression 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
```

6) [B] RANDOM FOREST MODEL

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

rf_preds = rf_model.predict(X_test)
```

```
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
print("Classification Report:\n", classification_report(y_test, rf_preds))
```

```
Random Forest Accuracy: 0.7207792207792207
Classification Report:
                precision    recall  f1-score   support

           0       0.79      0.78      0.78        99
           1       0.61      0.62      0.61        55

    accuracy                           0.72       154
   macro avg       0.70      0.70      0.70       154
weighted avg       0.72      0.72      0.72       154
```

7) MAKE PREDICTIONS

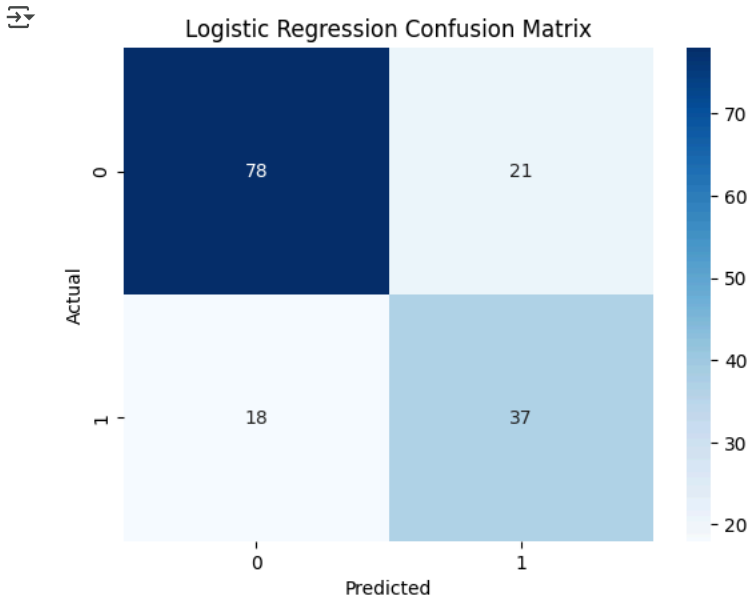
```
# Predict on test data
y_pred = model.predict(X_test)
```

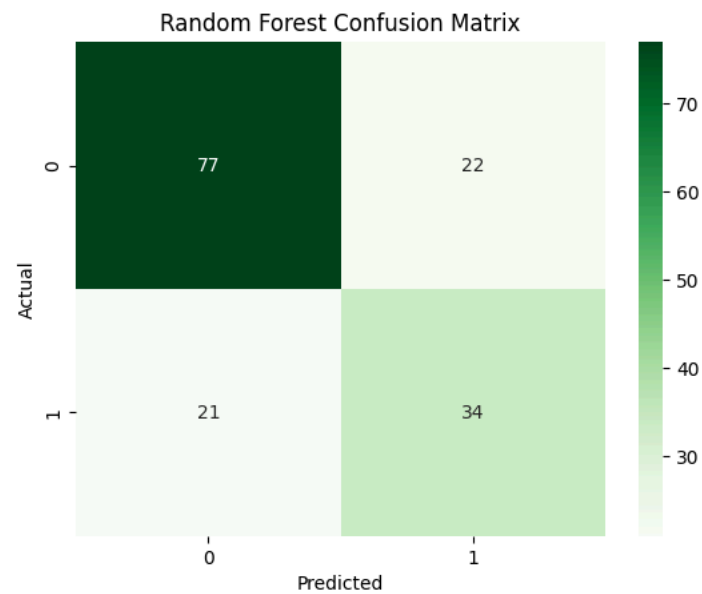
8) EVALUATE THE MODEL

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

# Random Forest Confusion Matrix
```

```
sns.heatmap(confusion_matrix(y_test, rf_preds), annot=True, fmt='d', cmap='Greens')
plt.title("Random Forest Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```





Build The Model

```
import gradio as gr
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
```

```
# Sample data and model (for demo purposes)
X, y = make_classification(n_samples=500, n_features=5, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)

# Prediction function
```

```
def predict_health_risk(age, gender, bmi, blood_pressure, glucose):
    gender_val = 1 if gender == "Male" else 0
    features = np.array([[age, gender_val, bmi, blood_pressure, glucose]])
    pred = model.predict(features)[0]
    return "High Risk" if pred == 1 else "Low Risk"

# Gradio Interface
iface = gr.Interface(
    fn=predict_health_risk,
    inputs=[
        gr.Number(label="Age"),
        gr.Radio(["Male", "Female"], label="Gender"),
        gr.Number(label="BMI"),
        gr.Number(label="BloodPressure"),
        gr.Number(label="Glucose"),
    ],
    outputs=gr.Text(label="Risk Prediction"),
    title="AI-Powered Health Risk Predictor",
    description="Enter patient details to predict health risk using AI.",
    theme="default"
)

# Launch app
iface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatically

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://36587948784983e87a.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working dir
```

AI-Powered Health Risk Predictor

Enter patient details to predict health risk using AI.

Age

0

Gender

Male Female

BMI

0

BloodPressure

0

Glucose

Risk Prediction

Flag