Upload 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.
Saving healthcare_dataset.csv to healthcare_dataset.csv
```

Load the Dataset:

```
[ ] import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    df = pd.read_csv('healthcare_dataset.csv')
```

Data Exploration:

```
df.head()
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
df.info()
df.describe()
```

```
Shape: (55500, 15)
Columns: ['Name', 'Age', 'Gender', 'Blood Type', 'Medical Condition', 'Date of Admission', 'Doctor', 'Hospital',
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55500 entries, 0 to 55499
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                55500 non-null  object
 1   Age                 55500 non-null  int64
 2   Gender              55500 non-null  object
 3   Blood Type          55500 non-null  object
 4   Medical Condition   55500 non-null  object
 5   Date of Admission   55500 non-null  object
 6   Doctor              55500 non-null  object
 7   Hospital            55500 non-null  object
 8   Insurance Provider  55500 non-null  object
 9   Billing Amount      55500 non-null  float64
 10  Room Number         55500 non-null  int64
 11  Admission Type      55500 non-null  object
 12  Discharge Date      55500 non-null  object
 13  Medication          55500 non-null  object
 14  Test Results        55500 non-null  object
dtypes: float64(1), int64(2), object(12)
memory usage: 6.4+ MB
```

	Age	Billing Amount	Room Number
count	55500.000000	55500.000000	55500.000000
mean	51.539459	25539.316097	301.134829
std	19.602454	14211.454431	115.243069
min	13.000000	-2008.492140	101.000000
25%	35.000000	13241.224652	202.000000
50%	52.000000	25538.069376	302.000000
75%	68.000000	37820.508436	401.000000
max	89.000000	52764.276736	500.000000

Check for Missing Values and Duplicates

```
print(df.isnull().sum())
print("Duplicate rows:", df.duplicated().sum())
```

```
Name                  0
Age                   0
Gender                0
Blood Type            0
Medical Condition     0
Date of Admission     0
Doctor                0
Hospital              0
Insurance Provider    0
Billing Amount        0
Room Number           0
Admission Type        0
Discharge Date        0
Medication            0
Test Results          0
dtype: int64
Duplicate rows: 534
```

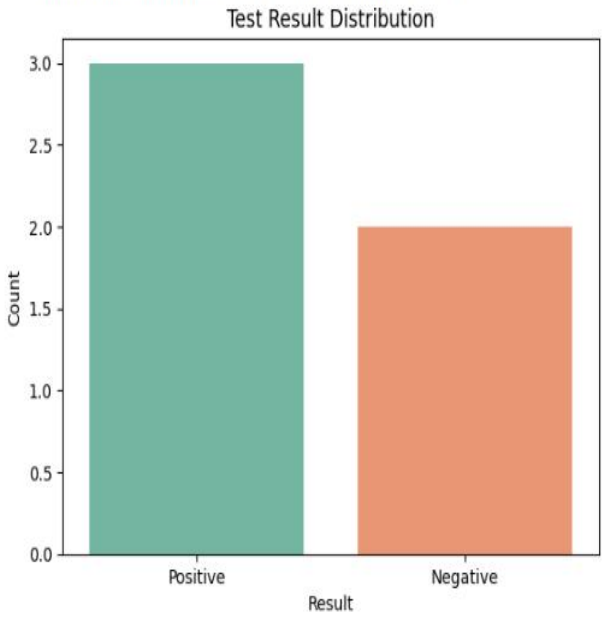
Visualize a Few Features

```
[ ] import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    df = pd.DataFrame({
        'test_result': ['Positive', 'Negative', 'Positive', 'Negative', 'Positive']
    })
    sns.countplot(data=df, x='test_result', palette='Set2')
    plt.title('Test Result Distribution')
    plt.xlabel('Result')
    plt.ylabel('Count')
    plt.show()
```

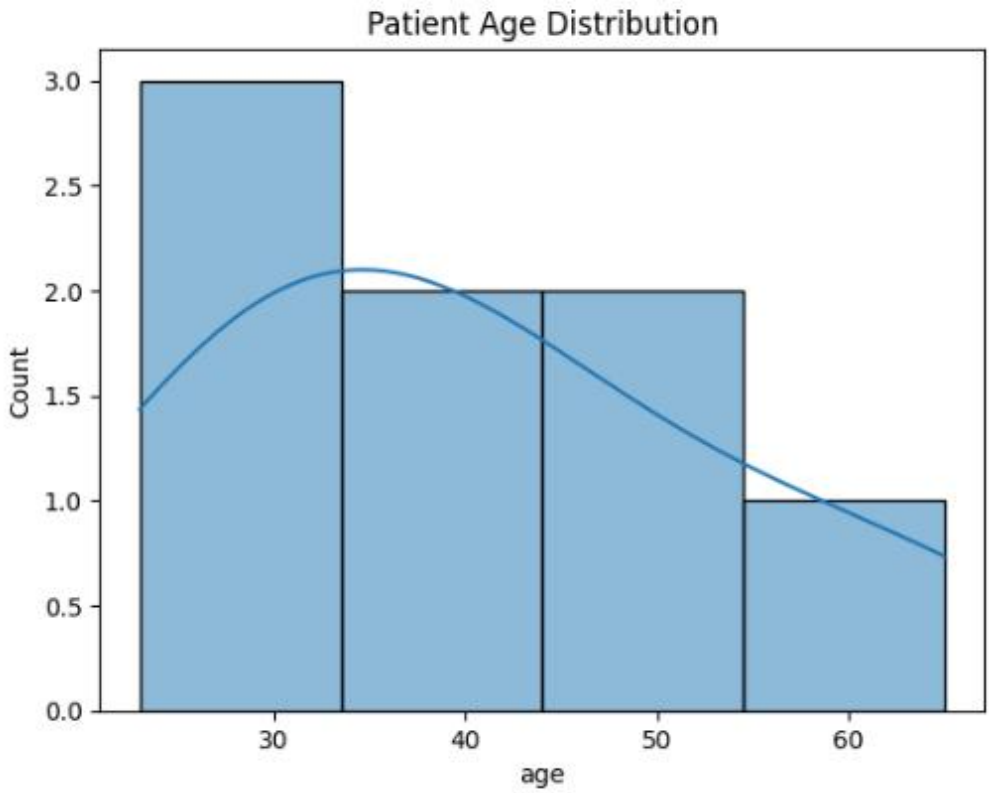
```
<ipython-input-5-85ea0559a4fa>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x='test_result', palette='Set2')
```



```
[ ] df = pd.DataFrame({'age': [23, 45, 36, 28, 65, 54, 31, 40]})
    sns.histplot(df['age'], kde=True)
    plt.title('Patient Age Distribution')
    plt.show()
```



Identify Target and Features

```
[ ] df = pd.DataFrame({
        'age': [25, 30, 45],
        'cholesterol': [200, 180, 210],
        'disease': [1, 0, 1]
    })

    target = 'disease'
    features = df.columns.drop(target)

    print("Target:", target)
    print("Features:", features)
```

```
Target: disease
Features: Index(['age', 'cholesterol'], dtype='object')
```

Convert Categorical Columns to Numerical

```
[ ] categorical_cols = df.select_dtypes(include=['object']).columns
    print("Categorical Columns:", categorical_cols.tolist())
```

```
Categorical Columns: []
```

One-Hot Encoding:

```
[ ] df_encoded = pd.get_dummies(df, drop_first=True)
```

Feature Scaling

```
[ ] from sklearn.preprocessing import StandardScaler
    Example: target = 'Outcome'
    scaler = StandardScaler()
    X = df_encoded.drop(columns=[target])
    X_scaled = scaler.fit_transform(X)
    y = df_encoded[target]
    plt.show()
```

Train-Test Split

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Model Building

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, classification_report
    model = RandomForestClassifier()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

Evaluation

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
Classification Report:
               precision    recall  f1-score   support

           1       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1
```

Make Predictions from New Input

```
[ ] new_patient = {
        'age': 45,
        'gender': 'Male',
        'blood_pressure': 130,
        'cholesterol': 200,
    }
```

Convert to DataFrame and Encode

```
[ ] new_df = pd.DataFrame([new_patient])
    df_temp = pd.concat([df.drop(target, axis=1), new_df], ignore_index=True)
    df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)
    df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop(target, axis=1).columns, fill_value=0)
    new_input_scaled = scaler.transform(df_temp_encoded.tail(1))
    predicted_disease = model.predict(new_input_scaled)
    print("Predicted Disease:", predicted_disease[0])
```

```
Predicted Disease: 1
```

Deployment-Building an Interactive App

```
[ ] !pip install gradio
    import gradio as gr
    def predict_disease(**inputs):
        input_df = pd.DataFrame([inputs])
        df_temp = pd.concat([df.drop(target, axis=1), input_df], ignore_index=True)
        df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)
        df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop(target, axis=1).columns, fill_value=0)
        scaled_input = scaler.transform(df_temp_encoded.tail(1))
        prediction = model.predict(scaled_input)
        return prediction[0]
```

```
Collecting gradio
  Downloading gradio-5.28.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.0 (from gradio)
  Downloading gradio_client-1.10.0-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.30.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.16)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.3)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.7-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
```

```
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.2)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio) (15.0.1)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.1.31)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.16.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.1 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (2.33.1)
```

Create the Gradio Interface

```
inputs = [
    gr.Number(label="Age"),
    gr.Dropdown(['Male', 'Female'], label="Gender"),
    gr.Number(label="Blood Pressure"),
    gr.Number(label="Cholesterol"),
]
gr.Interface(fn=predict_disease, inputs=inputs, outputs="text", title="AI Disease Predictor").launch()
```

```
/usr/local/lib/python3.11/dist-packages/gradio/utils.py:1018: UserWarning: Expected 0 arguments for function <function predict_disease at 0x7cb2642a5300>, received 4.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/gradio/utils.py:1026: UserWarning: Expected maximum 0 arguments for function <function predict_disease at 0x7cb2642a5300>, received 4.
  warnings.warn(
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatically setting `share=True` (you can turn this off

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://a551e18ffbacf7095a.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 directory to deploy to Hugging Face Spaces (htt
```