```
import pandas as pd
import matplotlib.pyplot as plt
```

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

```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Out
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	

```
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
```

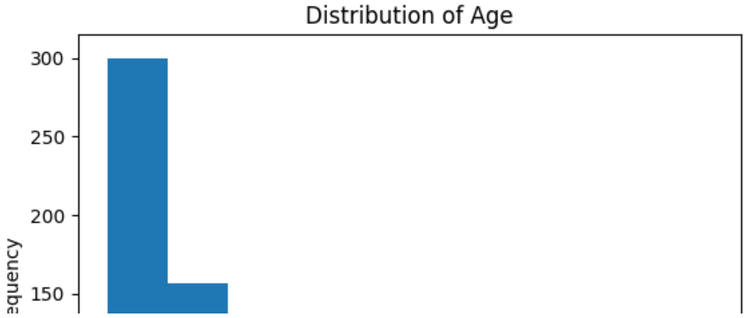
```
df.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

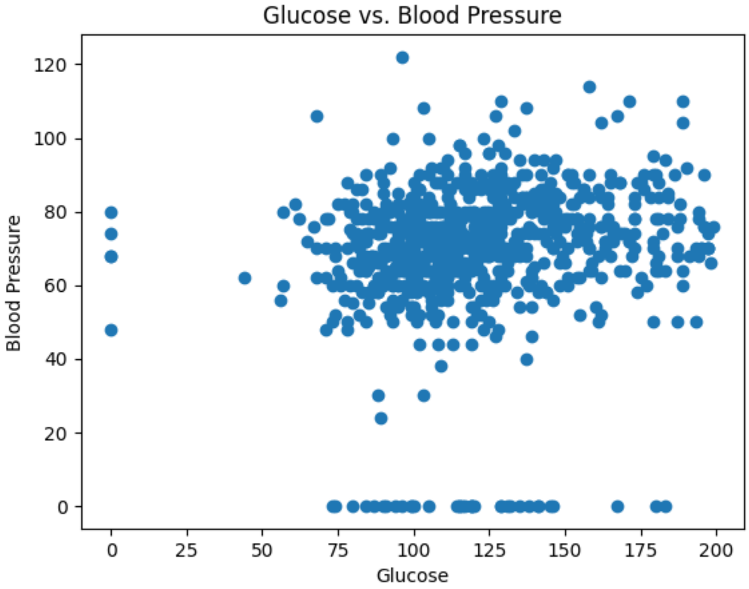
```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.

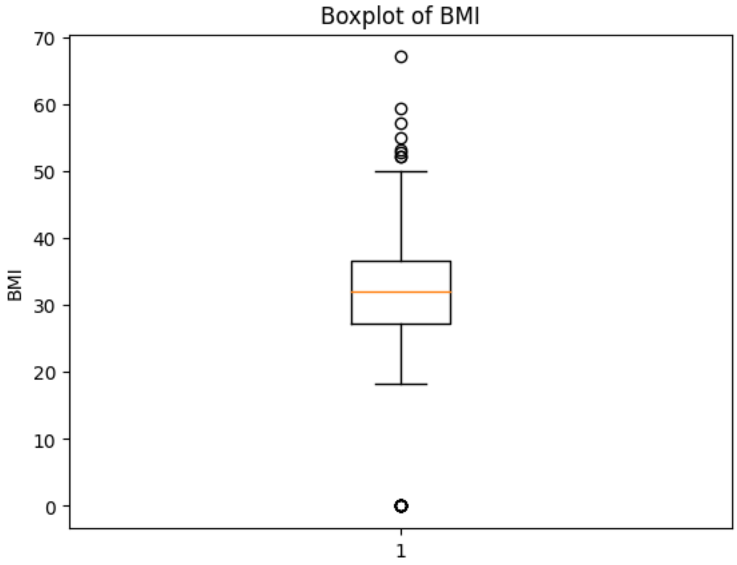
```
plt.hist(df['Age'], bins=10)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Age')
plt.show()
```



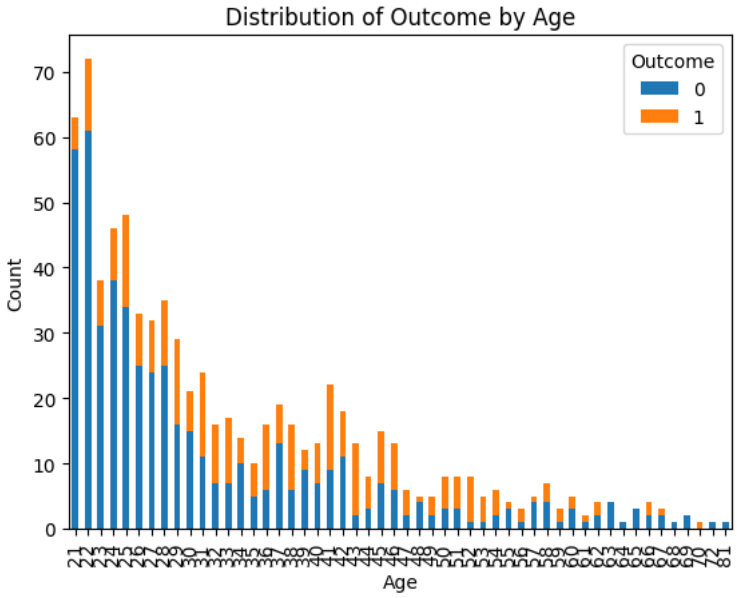
```
plt.scatter(df['Glucose'], df['BloodPressure'])
plt.xlabel('Glucose')
plt.ylabel('Blood Pressure')
plt.title('Glucose vs. Blood Pressure')
plt.show()
```



```
plt.boxplot(df['BMI'])
plt.ylabel('BMI')
plt.title('Boxplot of BMI')
plt.show()
```



```
age_outcome_counts = df.groupby('Age')['Outcome'].value_counts().unstack()
age_outcome_counts.plot(kind='bar', stacked=True)
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Distribution of Outcome by Age')
plt.legend(title='Outcome')
plt.show()
```



```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
x=df.drop(["Outcome"],axis=1)
y=df.Outcome
```

```
model = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=34)
```

```
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8116883116883117
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.16, random_state=0)
```

```
RF = RandomForestClassifier(random_state=42)
```

```
RF.fit(X_train, y_train)
```

```
y_pred = RF.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

```
0.8048780487804879
```

```
import pandas as pd


def make_prediction():

    pregnancies = int(input("Enter the number of pregnancies: "))
    glucose = float(input("Enter the glucose level: "))
    blood_pressure = float(input("Enter the blood pressure: "))
    skin_thickness = float(input("Enter the skin thickness: "))
    insulin = float(input("Enter the insulin level: "))
    bmi = float(input("Enter the BMI: "))
    diabetes_pedigree = float(input("Enter the diabetes pedigree function: "))
    age = int(input("Enter the age: "))


    data = pd.DataFrame([[pregnancies, glucose, blood_pressure, skin_thickness, insulin, bmi, diabetes_pedigree, age]],
                        columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
                                 'BMI', 'DiabetesPedigreeFunction', 'Age'])


    prediction = model.predict(data)


    if prediction[0] == 0:
        print("The model predicts that the person does not have diabetes.")
    else:
        print("The model predicts that the person has diabetes.")


make_prediction()
```

```
import joblib
joblib.dump(model, 'Diabetes.joblib')
```

```
     ['Diabetes.joblib']
```

```
import pickle

# Assuming you have trained your model and stored it in the 'model' variable
# Save the model as a pickle file
with open('your_model.pkl', 'wb') as file:
    pickle.dump(model, file)
```

```
from google.colab import files

# Download the pickle file
files.download('your_model.pkl')
```