Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

Data Collection and Analysis

In [2]:
```
# Loading data from CSV file to Pandas Dataframe

df = pd.read_csv('insurance.csv')
```

In [3]:
```
# First 5 rows of the Dataframe

df.head()
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
0	19	0	27.900	0	1	3	16884.92400	1
1	18	1	33.770	1	0	2	1725.55230	1
2	28	1	33.000	3	0	2	4449.46200	0
3	33	1	22.705	0	0	1	21984.47061	0
4	32	1	28.880	0	0	1	3866.85520	1

In [4]:
```
# Number of rows and columns

df.shape
```

Out[4]:
```
(1338, 8)
```

In [5]:
```
# Getting Information about Dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             1338 non-null   int64  
 1   sex             1338 non-null   int64  
 2   bmi             1338 non-null   float64
 3   children        1338 non-null   int64  
 4   smoker          1338 non-null   int64  
 5   region          1338 non-null   int64  
 6   charges         1338 non-null   float64
 7   insuranceclaim  1338 non-null   int64  
dtypes: float64(2), int64(6)
memory usage: 83.8 KB
```

In [6]:
```
# checking for Missing values

df.isnull().sum()
```

Out[6]:
```
age               0
sex               0
bmi               0
children          0
smoker            0
region            0
charges           0
insuranceclaim    0
dtype: int64
```

In [7]:
```
# Statical Information of dataset

df.describe()
```

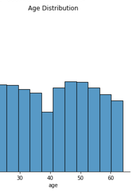
Out[7]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	1.515695	13270.422265	0.585202
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237	0.492871
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900	0.000000
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150	0.000000
50%	39.000000	1.000000	30.400000	1.000000	0.000000	2.000000	9382.033000	1.000000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515	1.000000
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010	1.000000

In [19]:
```
# Checking for Age distribution

plt.figure(figsize=(5,6))
sns.displot(df['age'])
plt.title('Age Distribution')
plt.show()
```

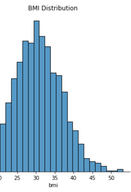
```
<Figure size 360x432 with 0 Axes>
```



In [20]:
```
# Checking for BMI distribution

plt.figure(figsize=(5,6))
sns.displot(df['bmi'])
plt.title('BMI Distribution')
plt.show()
```

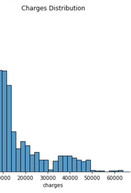
```
<Figure size 360x432 with 0 Axes>
```



In [21]:
```
# Checking for Charges distribution

plt.figure(figsize=(5,6))
sns.displot(df['charges'])
plt.title('Charges Distribution')
plt.show()
```

```
<Figure size 360x432 with 0 Axes>
```



Splitting Feature & Target

In [8]:
```
X = df.drop(['insuranceclaim'],axis=1)
y = df['insuranceclaim']
```

In [9]:
```
# Feature

X
```

Out[9]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	0	1	10600.54830
1334	18	0	31.920	0	0	0	2205.98080
1335	18	0	36.850	0	0	2	1629.83350
1336	21	0	25.800	0	0	3	2007.94500
1337	61	0	29.070	0	1	1	29141.36030

1338 rows × 7 columns

In [10]:
```
# Target

y
```

Out[10]:
```
0       1
1       1
2       0
3       0
4       1
       ..
1333    0
1334    1
1335    1
1336    0
1337    1
Name: insuranceclaim, Length: 1338, dtype: int64
```

Splitting data into Training & Tetsing Data

In [11]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.30,random_state=42)
```

In [12]:
```
# Printing shape

print("X_train :",X_train.shape)
print("X_test :",X_test.shape)
print("y_train :",y_train.shape)
print("y_test :",y_test.shape)
```

```
X_train : (936, 7)
X_test : (402, 7)
y_train : (936,)
y_test : (402,)
```

Model Training

In [13]:
```
# Loading the Decision Tree Classification model

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
```

In [14]:
```
# Fitting the model

dt.fit(X_train,y_train)
```

Out[14]:
```
DecisionTreeClassifier()
```

Model Evaluation

In [15]:
```
# Checking Accuracy on Training Data

acc1 = dt.score(X_train,y_train)
print("Accuracy on Training data :",acc1)
```

```
Accuracy on Training data : 1.0
```

In [16]:
```
# Prediction on Testing Data

y_pred = dt.predict(X_test)
print(y_pred)
```

```
[0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 0 1 1 0 1 0 0 1 0
 1 1 1 0 1 1 0 0 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 0 1 0 1 0 1 0 1 0
 1 1 0 0 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 0 1 1 1 1
 1 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1
 0 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 0 1 0
 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 1 1 1
 0 1 1 1 0 1 1 1 0 0 1 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0
 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0
 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1
 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0 0 0
 1 1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0]
```

In [17]:
```
# Checking Accuracy on Testing Data

acc2 = dt.score(X_test,y_test)
print("Accuracy on Tetsing data :",acc2)
```

```
Accuracy on Tetsing data : 0.9751243781094527
```

In [18]:
```
# I used Logistic Regression model on Same dataset and I got the Accuracy:0.815020398
```