Import the important libraries

```
import numpy as np
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 LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn import metrics
```

Import The Dataset

```
 #loading the data from csv file to a Pandas DataFrame
insurance_dataset = pd.read_csv('/content/insurance.csv')

insurance_dataset.head()
```

```
{"summary":"{\n  \"name\": \"insurance_dataset\",\n  \"rows\": 1338,\n
\"fields\": [\n    {\n      \"column\": \"age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
14,\n        \"min\": 18,\n        \"max\": 64,\n
\"num_unique_values\": 47,\n        \"samples\": [\n          21,\n
45,\n          36\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"male\",\n          \"female\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"bmi\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 6.098186911679017,\n
\"min\": 15.96,\n        \"max\": 53.13,\n
\"num_unique_values\": 548,\n        \"samples\": [\n          23.18,\
n          26.885\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"children\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 5,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"smoker\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"no\",\n          \"yes\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"region\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"southeast\",\n
\"northeast\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"charges\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 12110.011236693994,\n        \"min\": 1121.8739,\n
\"max\": 63770.42801,\n        \"num_unique_values\": 1337,\n
\"samples\": [\n          8688.85885,\n          5708.867\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"insurance_dataset"}
```

```
insurance_dataset.shape
```

```
(1338, 7)
```

```
insurance_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

Categorical Features:

1) Sex 2) Smoker 3) Region

```
insurance_dataset.isnull().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

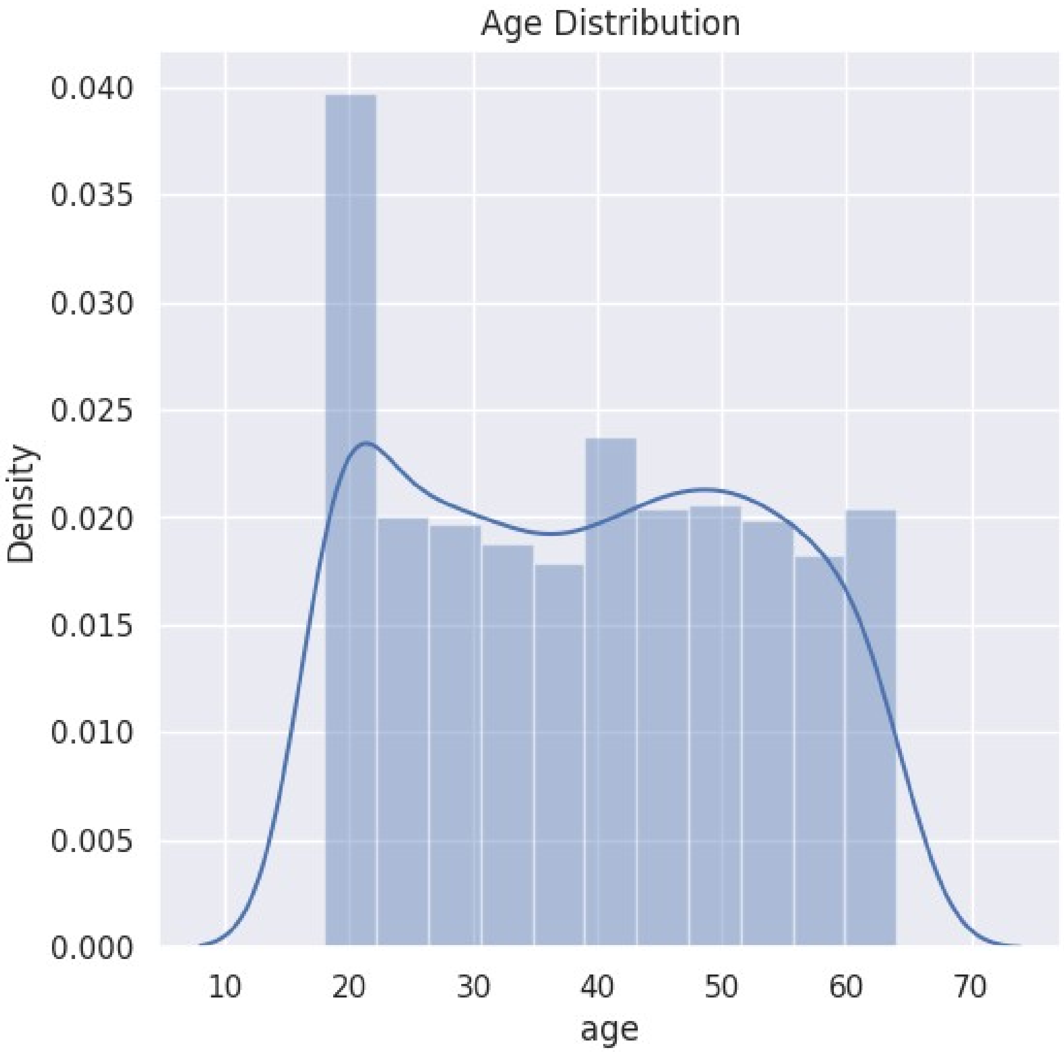
```
insurance_dataset.describe()
```

```
{"summary":"{\n  \"name\": \"insurance_dataset\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

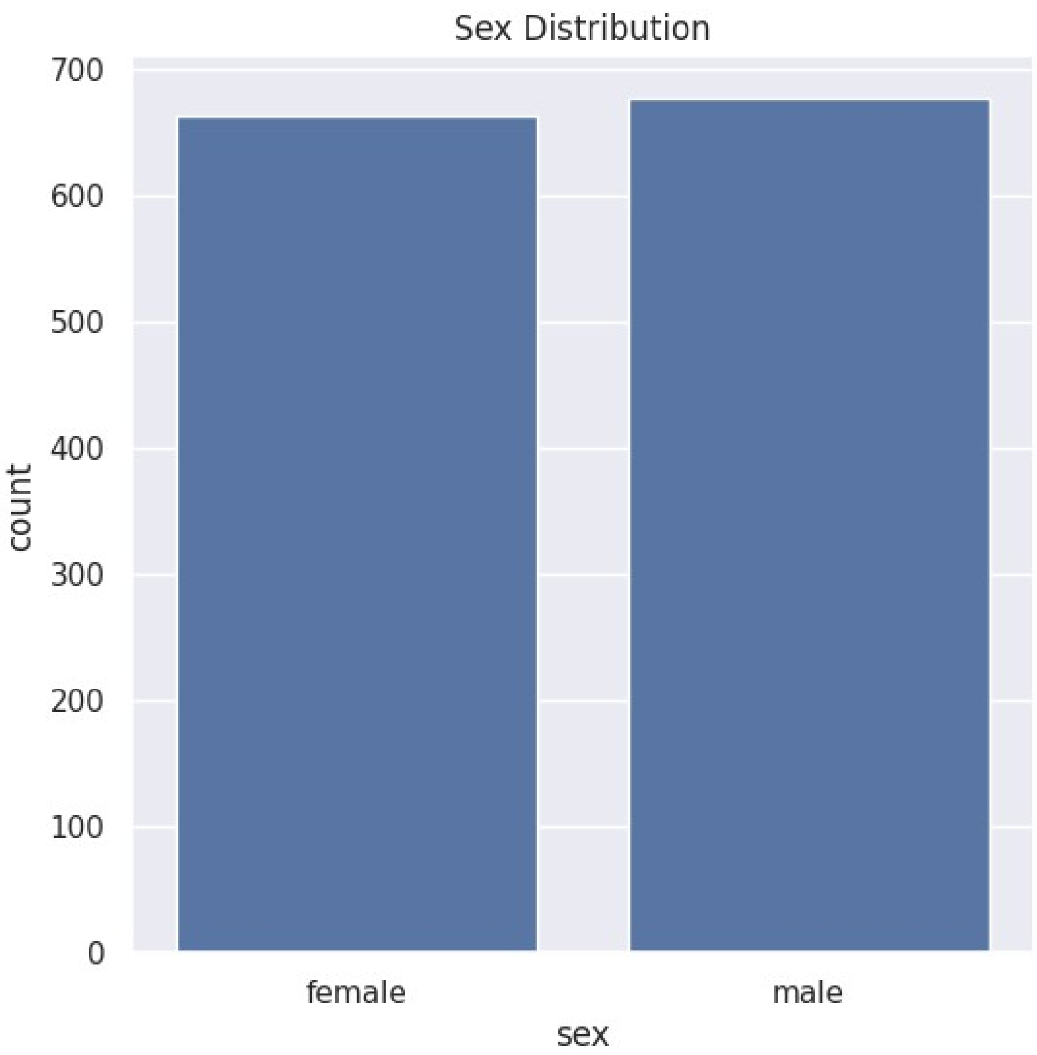
```
460.6106090399993,\n        \"min\": 14.049960379216172,\n
\"max\": 1338.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          39.20702541106129,\n          39.0,\n
1338.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"bmi\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 463.29524977918294,\n        \"min\": 6.098186911679017,\n
\"max\": 1338.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          30.66339686098655,\n          30.4,\n
1338.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"children\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 472.5368318870757,\n        \"min\":
0.0,\n        \"max\": 1338.0,\n        \"num_unique_values\": 7,\n
\"samples\": [\n          1338.0,\n          1.0949177877429,\n
2.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"charges\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 20381.922846226596,\n        \"min\": 1121.8739,\n
\"max\": 63770.42801,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          13270.422265141257,\n          9382.033,\n
1338.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```python
# distribution of age value
sns.set()
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['age'])
plt.title('Age Distribution')
plt.show()
```

```
/tmp/ipython-input-3634923312.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(insurance_dataset['age'])
```



```
# Gender column
plt.figure(figsize=(6,6))
sns.countplot(x='sex', data=insurance_dataset)
plt.title('Sex Distribution')
plt.show()
```



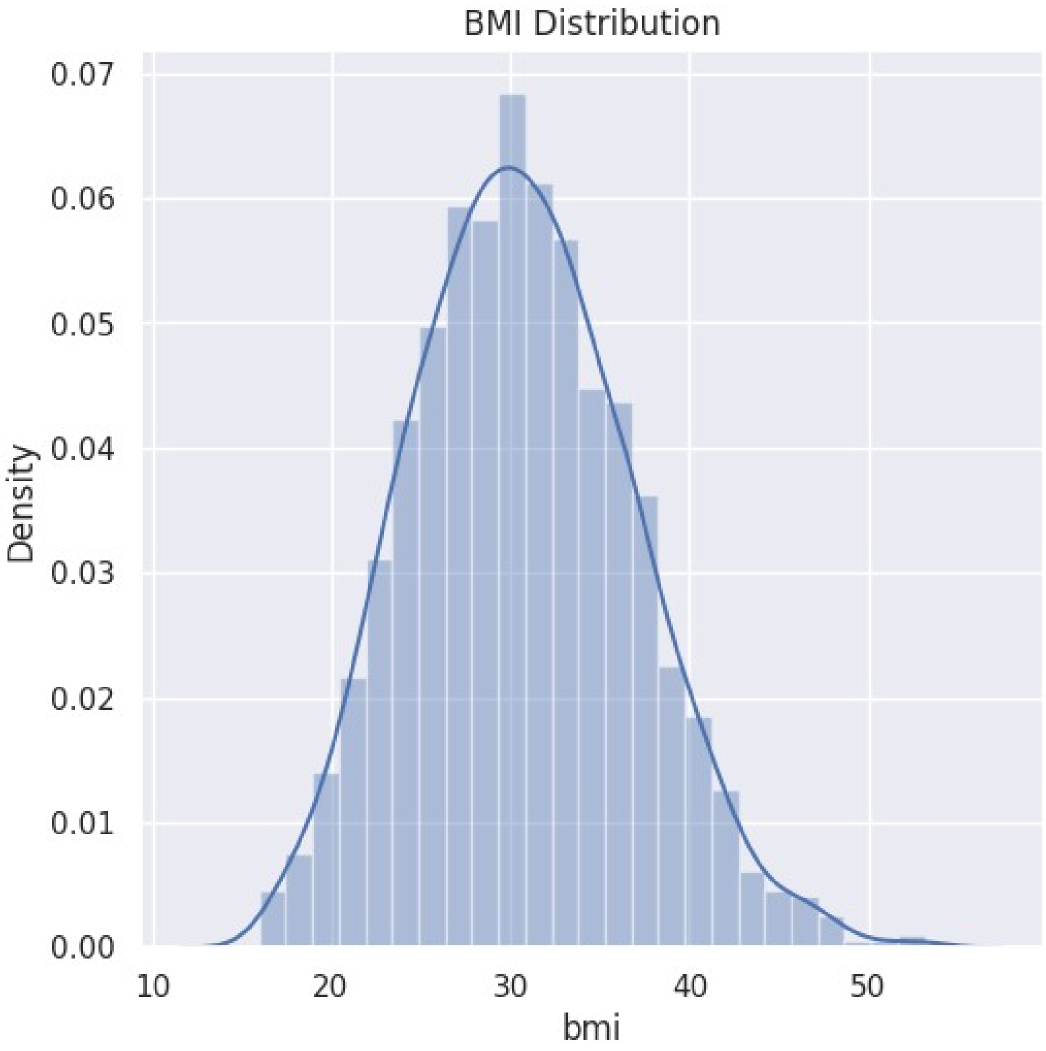
```
insurance_dataset['sex'].value_counts()
```

```
sex
male      676
female    662
Name: count, dtype: int64
```

```
# bmi distribution
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['bmi'])
plt.title('BMI Distribution')
plt.show()
```

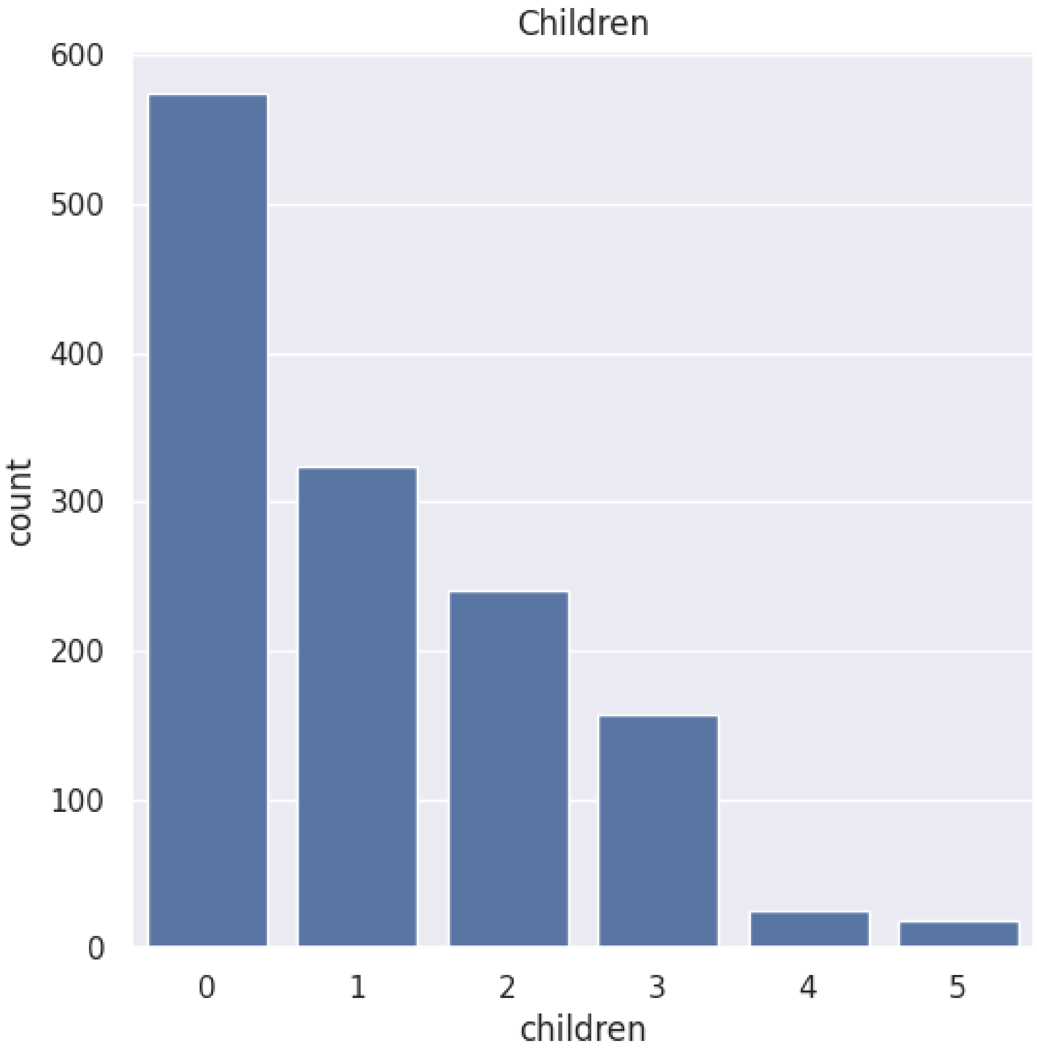
```
/tmp/ipython-input-1916795400.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(insurance_dataset['bmi'])
```



Normal BMI Range --> 18.5 to 24.9

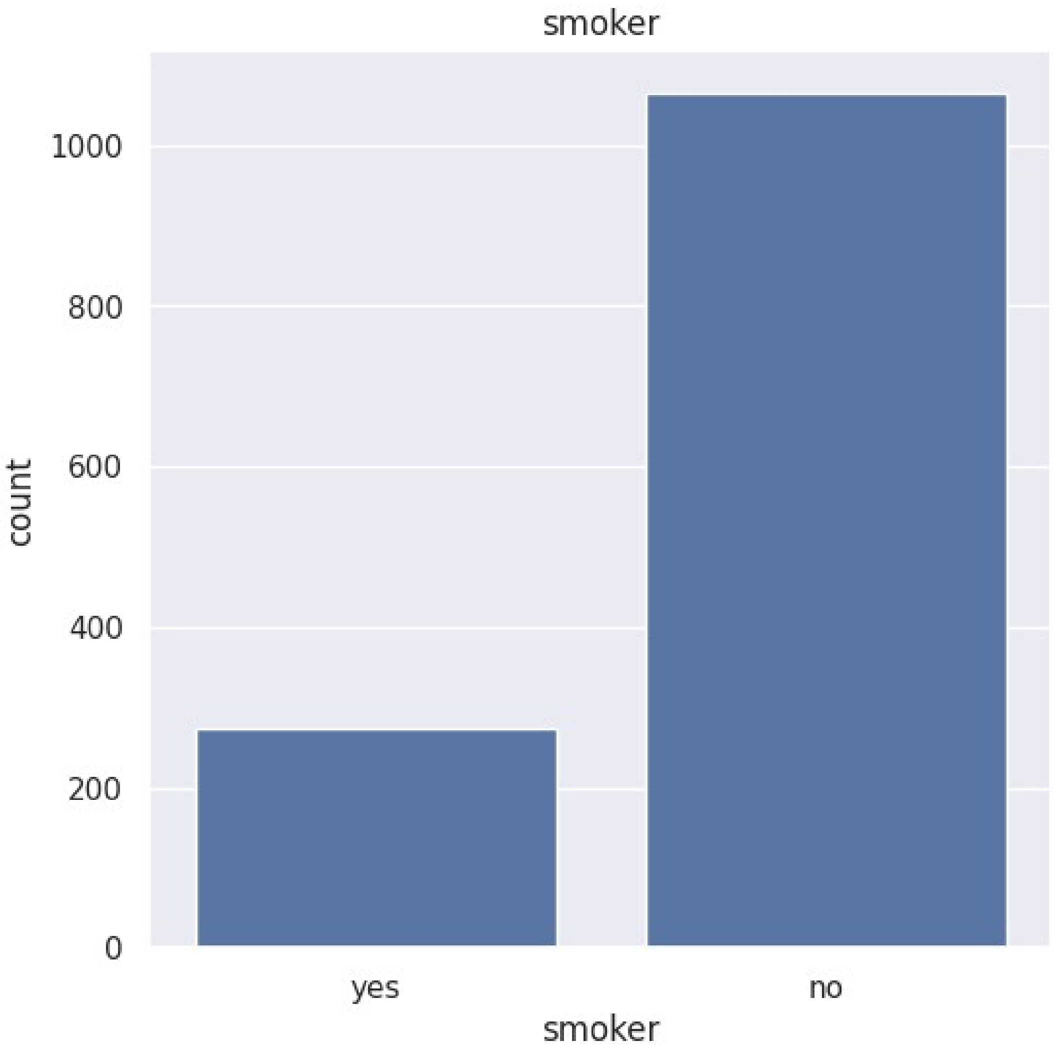
```
# children column
plt.figure(figsize=(6,6))
sns.countplot(x='children', data=insurance_dataset)
plt.title('Children')
plt.show()
```



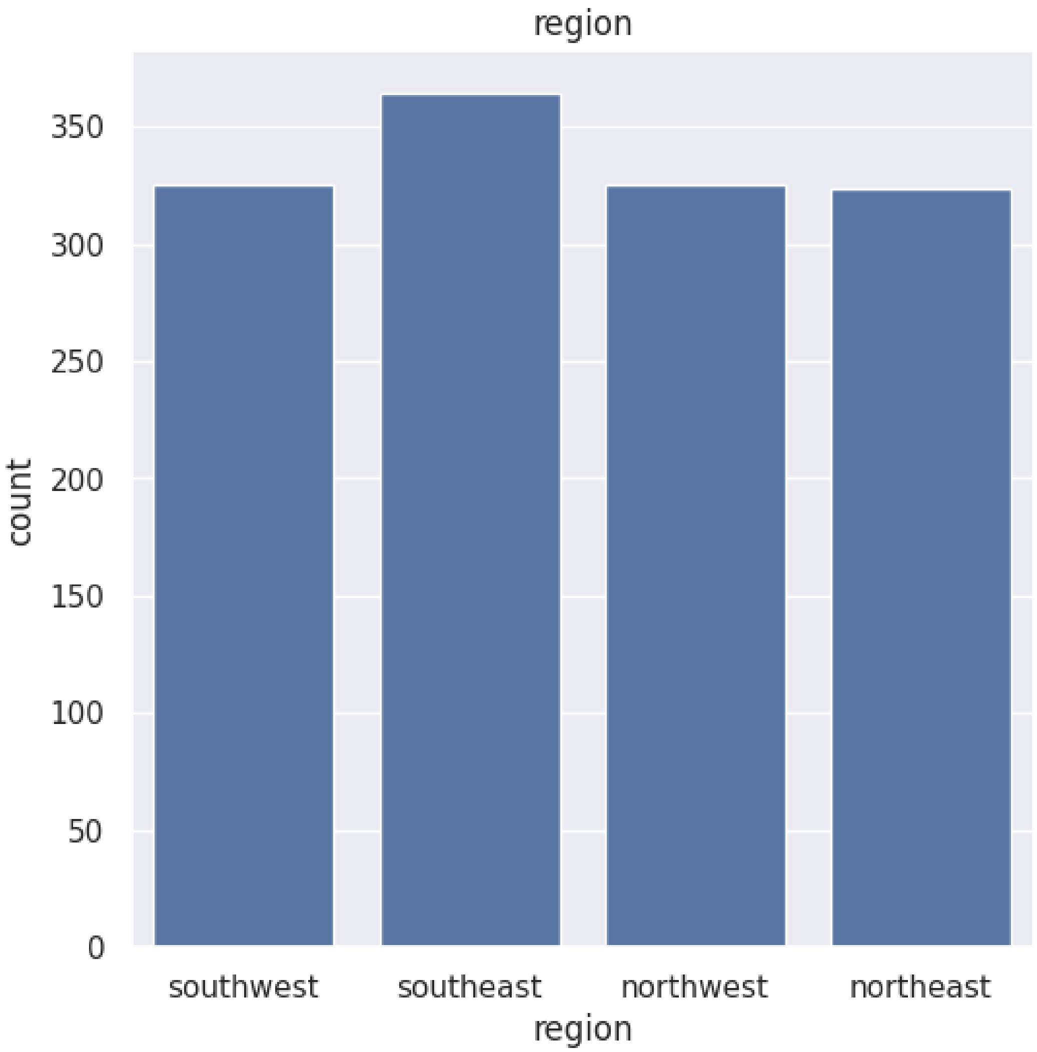
```
insurance_dataset['children'].value_counts()
```

```
children
0    574
1    324
2    240
3    157
4     25
5     18
Name: count, dtype: int64
```

```
# smoker column
plt.figure(figsize=(6,6))
sns.countplot(x='smoker', data=insurance_dataset)
plt.title('smoker')
plt.show()
```



```
insurance_dataset['smoker'].value_counts()

smoker
no     1064
yes     274
Name: count, dtype: int64

# region column
plt.figure(figsize=(6,6))
sns.countplot(x='region', data=insurance_dataset)
plt.title('region')
plt.show()
```



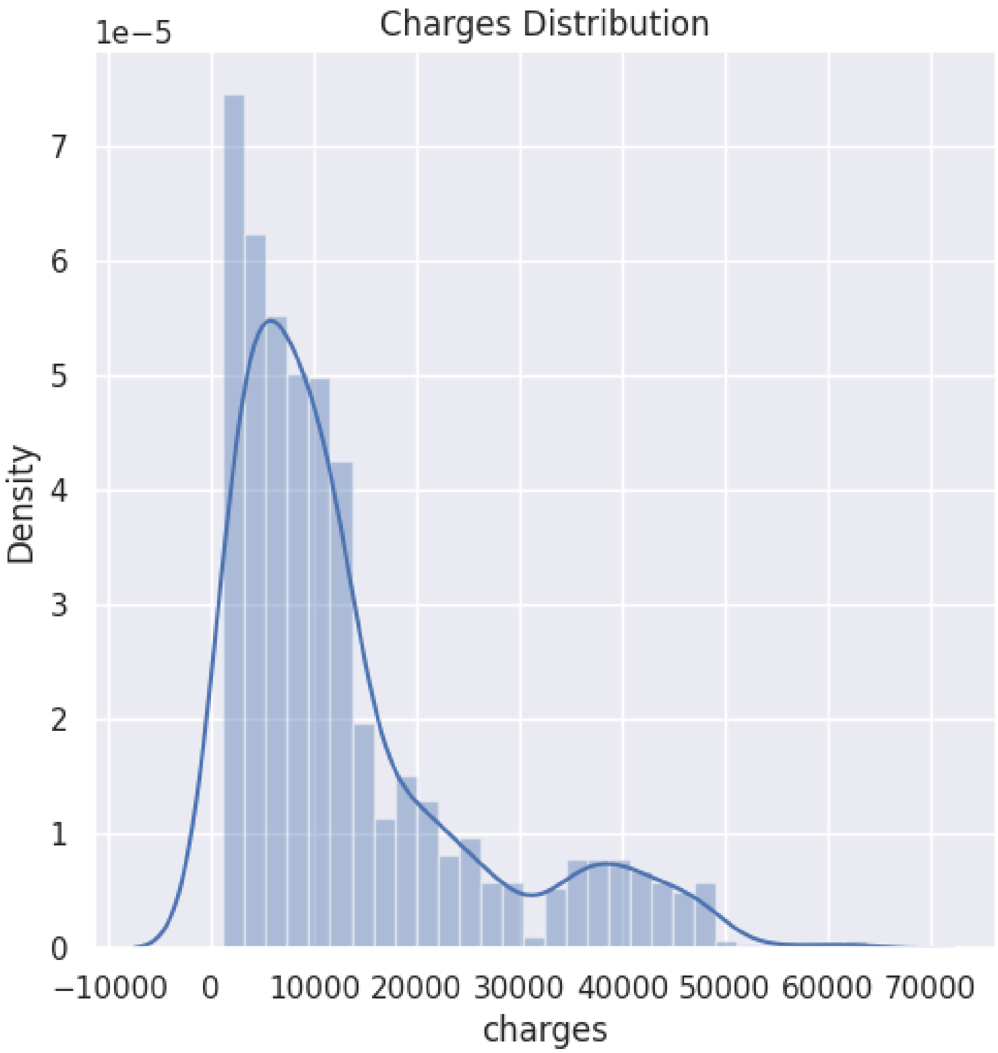
```
insurance_dataset['region'].value_counts()
```

```
region
southeast    364
southwest    325
northwest    325
northeast    324
Name: count, dtype: int64
```

```
# distribution of charges value
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['charges'])
plt.title('Charges Distribution')
plt.show()
```

```
/tmp/ipython-input-3971177022.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
```

```
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(insurance_dataset['charges'])
```



Data Pre-Processing

Encoding the categorical features

```
# encoding sex column
insurance_dataset.replace({'sex':{'male':0,'female':1}}, inplace=True)

3 # encoding 'smoker' column
insurance_dataset.replace({'smoker':{'yes':0,'no':1}}, inplace=True)

# encoding 'region' column
insurance_dataset.replace({'region':
{'southeast':0,'southwest':1,'northeast':2,'northwest':3}},
inplace=True)
```

```
/tmp/ipython-input-2871422651.py:2: FutureWarning: Downcasting
behavior in `replace` is deprecated and will be removed in a future
version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  insurance_dataset.replace({'sex':{'male':0,'female':1}},
inplace=True)
/tmp/ipython-input-2871422651.py:5: FutureWarning: Downcasting
behavior in `replace` is deprecated and will be removed in a future
version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  insurance_dataset.replace({'smoker':{'yes':0,'no':1}}, inplace=True)
/tmp/ipython-input-2871422651.py:8: FutureWarning: Downcasting
behavior in `replace` is deprecated and will be removed in a future
version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  insurance_dataset.replace({'region':
{'southeast':0,'southwest':1,'northeast':2,'northwest':3}},
inplace=True)
```

```
X = insurance_dataset.drop(columns='charges', axis=1)
Y = insurance_dataset['charges']

print(X)
```

```
      age  sex     bmi  children  smoker  region
0      19    1  27.900         0       0       1
1      18    0  33.770         1       1       0
2      28    0  33.000         3       1       0
3      33    0  22.705         0       1       3
4      32    0  28.880         0       1       3
...   ...  ...     ...       ...     ...     ...
1333   50    0  30.970         3       1       3
1334   18    1  31.920         0       1       2
1335   18    1  36.850         0       1       0
1336   21    1  25.800         0       1       1
1337   61    1  29.070         0       0       3
```

```
[1338 rows x 6 columns]

print(Y)

0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...     
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

Splitting data into training and test set

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=2)

print(X.shape, X_train.shape, X_test.shape)

(1338, 6) (1070, 6) (268, 6)
```

Model Training

1. Linear Regression

```
# loading the Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train, Y_train)
model = LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None)

# prediction on training data
training_data_prediction =regressor.predict(X_train)

# R squared value
r2_train = metrics.r2_score(Y_train, training_data_prediction)
print('R squared vale : ', r2_train)

R squared vale :  0.751505643411174

# prediction on test data
test_data_prediction_lr =lr_model.predict(X_test)
# R squared value
r2_lr = metrics.r2_score(Y_test, test_data_prediction_lr)
print('R squared vale : ', r2_lr)
```

```
mse = metrics.mean_squared_error(Y_test, test_data_prediction_lr)
rmse = np.sqrt(mse)
print(rmse)

R squared vale :  0.7447273869684076
6191.690842285236
```

Decision tree regressor

```
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, Y_train)
test_data_prediction_dt = dt_model.predict(X_test)
r2_dt = metrics.r2_score(Y_test, test_data_prediction_dt)
print(f"Decision Tree R² Score: {r2_dt:.4f}")

mse = metrics.mean_squared_error(Y_test, test_data_prediction_dt)
rmse = np.sqrt(mse)
print(rmse)

Decision Tree R² Score: 0.7166
6524.204772451173
```

Random Forest Regressor

```
rf_model = RandomForestRegressor(random_state=42, n_estimators=100)
rf_model.fit(X_train, Y_train)
test_data_prediction_rf = rf_model.predict(X_test)
r2_rf = metrics.r2_score(Y_test, test_data_prediction_rf)
print(f"Random Forest R² Score: {r2_rf:.4f}")

mse = metrics.mean_squared_error(Y_test, test_data_prediction_rf)
rmse = np.sqrt(mse)
print(rmse)

Random Forest R² Score: 0.8379
4933.692230552383

XG Boost

xgb_model = XGBRegressor(random_state=42, n_estimators=100)
xgb_model.fit(X_train, Y_train)
test_data_prediction_xgb = xgb_model.predict(X_test)
r2_xgb = metrics.r2_score(Y_test, test_data_prediction_xgb)
print(f"XGBoost R² Score: {r2_xgb:.4f}")

mse = metrics.mean_squared_error(Y_test, test_data_prediction_xg)
rmse = np.sqrt(mse)
print(rmse)
```

```
XGBoost R² Score: 0.8144
5279.090073315875

results = {
    "Linear Regression": {"R² Score": 0.7447, "RMSE": 6191.69},
    "Decision Tree":     {"R² Score": 0.7166, "RMSE": 6524.60 },
    "Random Forest":     {"R² Score": 0.8379, "RMSE": 4933.69},
    "XGBoost":           {"R² Score": 0.8144, "RMSE": 5279.09}
}

# Convert to DataFrame
results_df = pd.DataFrame(results).T  # .T to make models as rows
results_df = results_df.round(4)      # Round values for clean display

print(results_df)

                   R² Score     RMSE
Linear Regression    0.7447  6191.69
Decision Tree        0.7166  6524.60
Random Forest        0.8379  4933.69
XGBoost              0.8144  5279.09
```

Prediction of Medical Insurance on new input data

1. Linear Regession

```
input_data = (31,1,25.74,0,1,0)

# changing input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = lr_model.predict(input_data_reshaped)
print(prediction)

print('The insurance cost is USD ', prediction[0])

[3760.0805765]
The insurance cost is USD  3760.080576496057

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

1. Decision Tree

```
input_data = (31,1,25.74,0,1,0)

# changing input_data to a numpy array
```

```
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = dt_model.predict(input_data_reshaped)
print(prediction)

print('The insurance cost is USD ', prediction[0])
```

```
[3756.6216]
The insurance cost is USD  3756.6216

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
```

Random Forest

```
input_data = (31,1,25.74,0,1,0)

# changing input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = rf_model.predict(input_data_reshaped)
print(prediction)

print('The insurance cost is USD ', prediction[0])
```

```
[3729.6420035]
The insurance cost is USD  3729.6420035000065

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

XG Boost

```
input_data = (31,1,25.74,0,1,0)

# changing input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```

```python
prediction = xgb_model.predict(input_data_reshaped)
print(prediction)

print('The insurance cost is USD ', prediction[0])
```

```
[3409.0215]
The insurance cost is USD  3409.0215
```