Importing the Dependencies

```
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 import metrics
```

Data Collection & Analysis

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

```
# first 5 rows of the dataframe
insurance_dataset.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
# number of rows and columns
insurance_dataset.shape
```

```
(1338, 7)
```

```
# getting some informations about the dataset
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:

- Sex
- Smoker
- Region

```
# checking for missing values
insurance_dataset.isnull().sum()
```

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

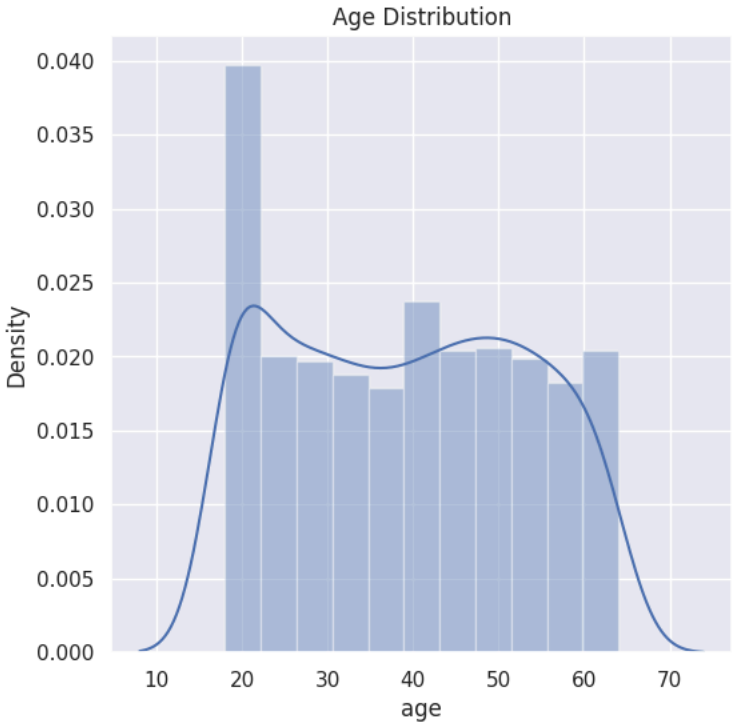
Data Analysis

```
# statistical Measures of the dataset
insurance_dataset.describe()
```

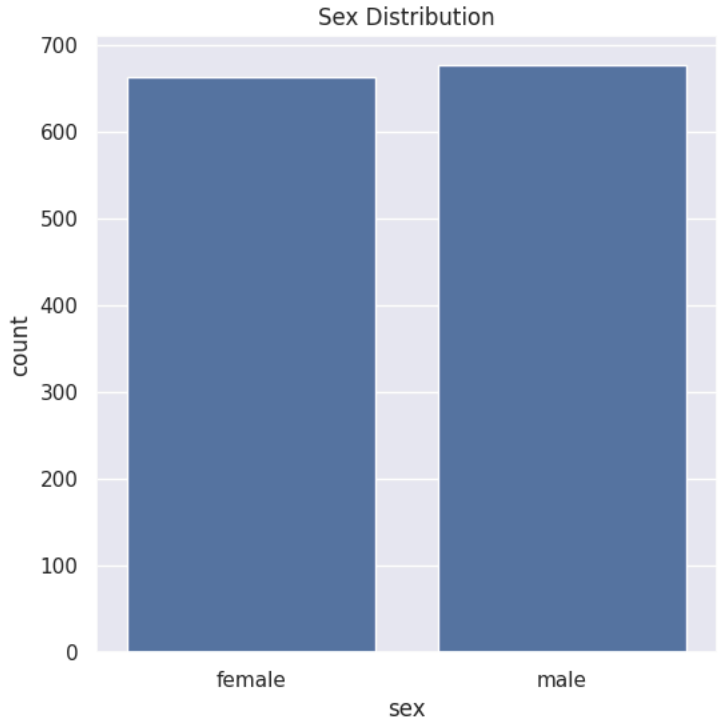
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

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

```
<ipython-input-8-28228e9c3528>: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'])
```



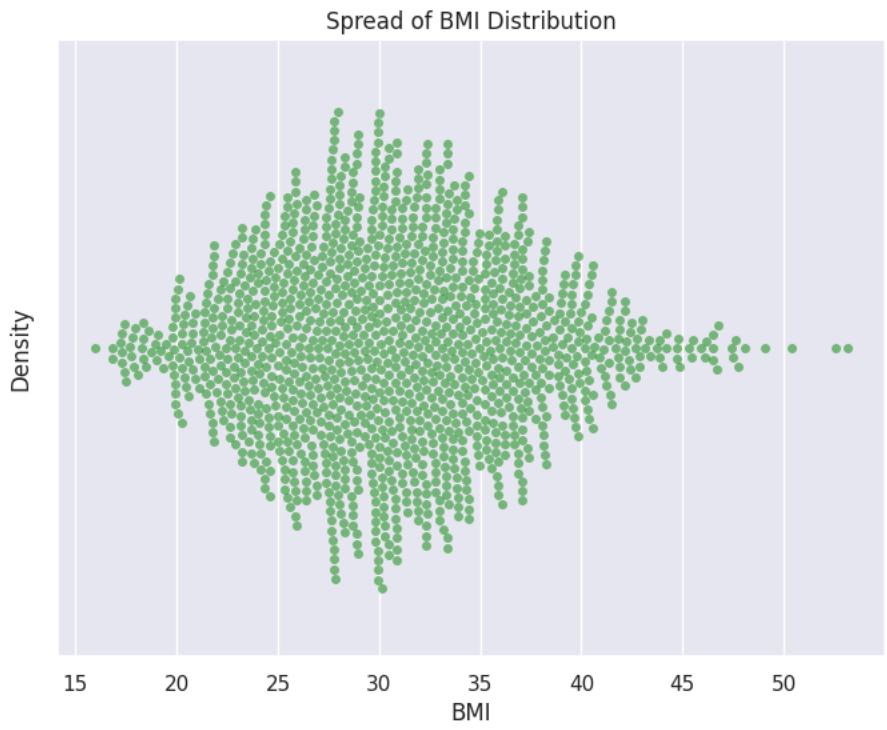

```
# Gender column
plt.figure(figsize=(6,6))
sns.countplot(x='sex', data=insurance_dataset)
plt.title('Sex Distribution')
plt.show()
```




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

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

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming insurance_dataset is your DataFrame containing the insurance data
plt.figure(figsize=(8, 6))
sns.swarmplot(x='bmi', data=insurance_dataset, color='green', alpha=0.5)
plt.title('Spread of BMI Distribution')
plt.xlabel('BMI')
plt.ylabel('Density')
plt.show()
```

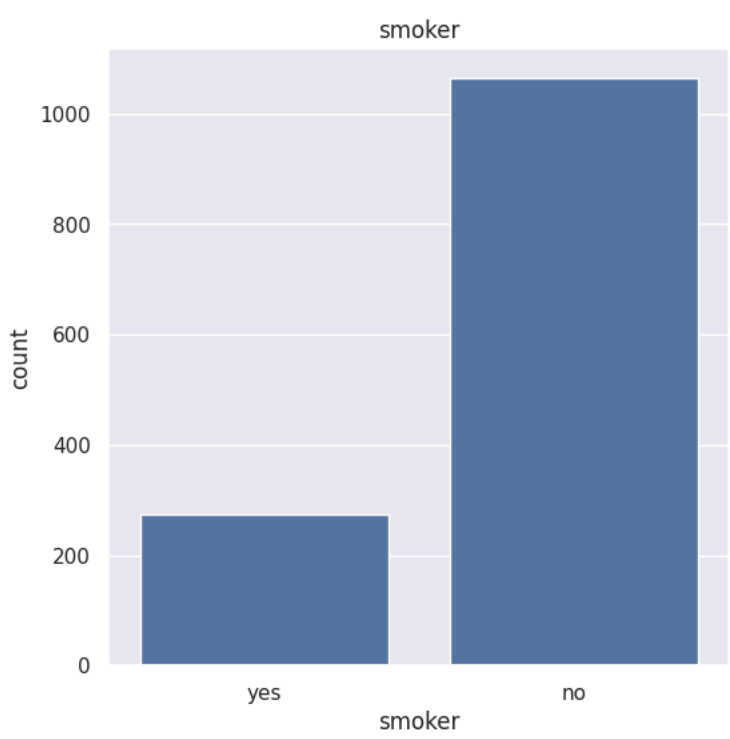


Normal BMI Range --> 18.5 to 24.9

```
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()
```



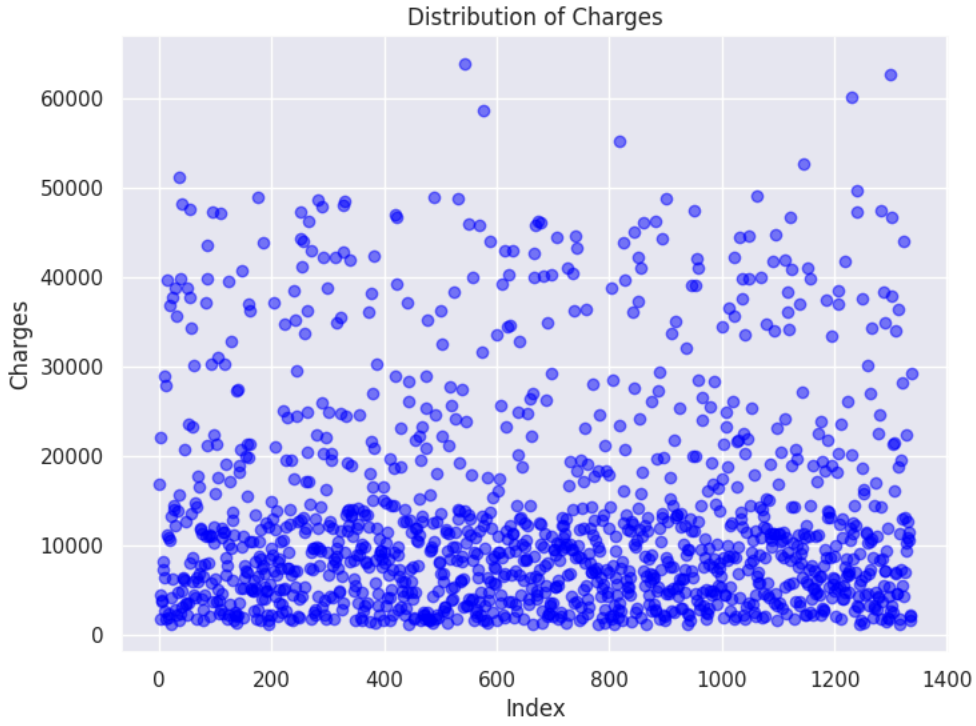

```
insurance_dataset['smoker'].value_counts()
```

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

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

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

```
import matplotlib.pyplot as plt

# Assuming insurance_dataset is your DataFrame containing the insurance data
plt.figure(figsize=(8, 6))
plt.scatter(range(len(insurance_dataset)), insurance_dataset['charges'], color='blue', alpha=0.5)
plt.title('Distribution of Charges')
plt.xlabel('Index')
plt.ylabel('Charges')
plt.show()
```



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)
```

Splitting the Features and Target

```
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 the data into Training data & Testing Data

```
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

Linear Regression

```
# loading the Linear Regression model
regressor = LinearRegression()
```

```
regressor.fit(X_train, Y_train)
```

```
LinearRegression
LinearRegression()
```

Model Evaluation

```
# 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 =regressor.predict(X_test)
```

```
# R squared value
r2_test = metrics.r2_score(Y_test, test_data_prediction)
print('R squared vale : ', r2_test)
```

```
R squared vale :  0.7447273869684076
```

Building a Predictive System

```
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 = regressor.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/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
```

```
from sklearn.linear_model import Lasso
from sklearn import metrics
import numpy as np
```

```
lasso_regressor = Lasso(alpha=0.1)
```

```
lasso_regressor.fit(X_train, Y_train)
```

```
Lasso
Lasso(alpha=0.1)
```

```
training_data_prediction = lasso_regressor.predict(X_train)
r2_train = metrics.r2_score(Y_train, training_data_prediction)
print('R squared value on training data: ', r2_train)
```

```
R squared value on training data:  0.7515056425277682
```

```
test_data_prediction = lasso_regressor.predict(X_test)
r2_test = metrics.r2_score(Y_test, test_data_prediction)
print('R squared value on test data: ', r2_test)
```

```
R squared value on test data:  0.7447271103401147
```

Start coding or generate with AI.

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

input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)

prediction = lasso_regressor.predict(input_data_reshaped)
print('The insurance cost is USD ', prediction[0])
```

```
The insurance cost is USD  3760.263524845088
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lasso was fitted
  warnings.warn(
```

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score

# Initialize and train the Gradient Boosting model
gbm_model = GradientBoostingRegressor(random_state=42)
gbm_model.fit(X_train, Y_train)

# Predict on the test set
test_data_prediction_gbm = gbm_model.predict(X_test)

# Calculate R^2 score on the test data
r2_test_gbm = r2_score(Y_test, test_data_prediction_gbm)

print('R squared value on test data (Gradient Boosting):', r2_test_gbm)
```

```
R squared value on test data (Gradient Boosting): 0.8676562807388961
```

```
# Convert input_data into a DataFrame with appropriate column names
input_data_df = pd.DataFrame([input_data], columns=X_train.columns)

# Predict insurance costs for the input data
predicted_cost_gbm = gbm_model.predict(input_data_df)

print("Predicted insurance cost (Gradient Boosting):", predicted_cost_gbm[0])
```

```
Predicted insurance cost (Gradient Boosting): 4049.422985340062
```