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()
```

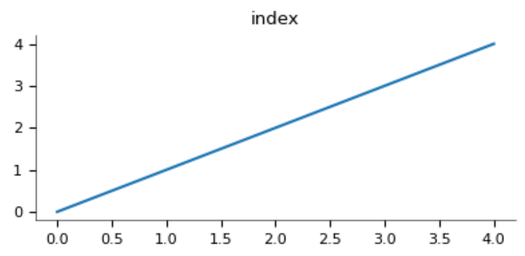
1 to 5 of 5 entries Filter

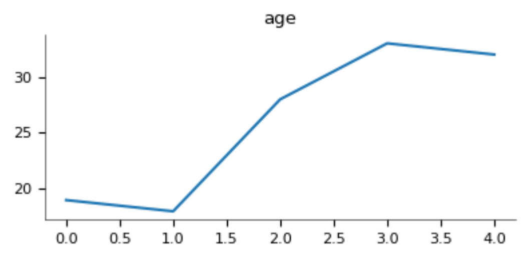
index	age	sex	bmi	children	smoker	region	charges
0	19	female	27.9	0	yes	southwest	16884.924
1	18	male	33.77	1	no	southeast	1725.5523
2	28	male	33.0	3	no	southeast	4449.462
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.88	0	no	northwest	3866.8552

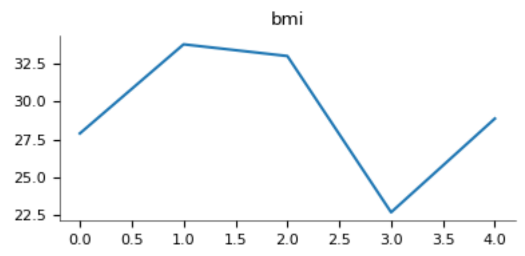
Show 25 per page

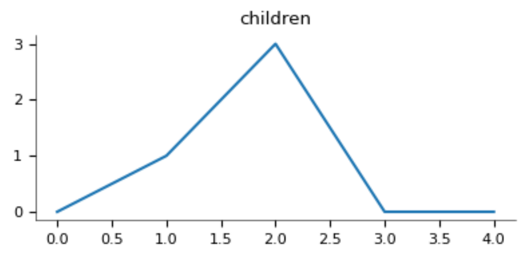
Like what you see? Visit the data table notebook to learn more about interactive tables.

Values

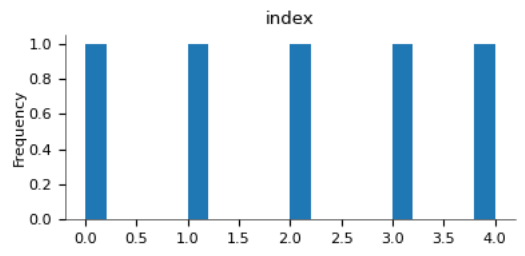


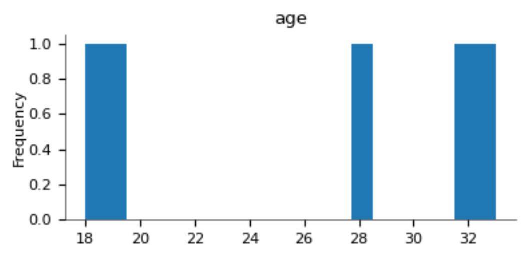


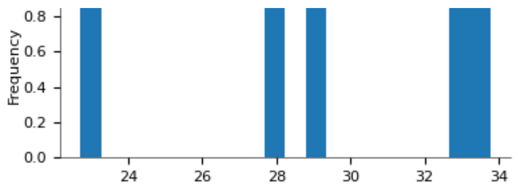


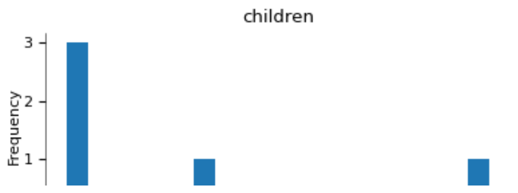


Distributions









```
# 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()
```

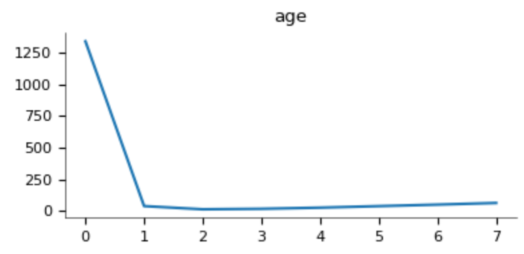
1 to 8 of 8 entries Filter

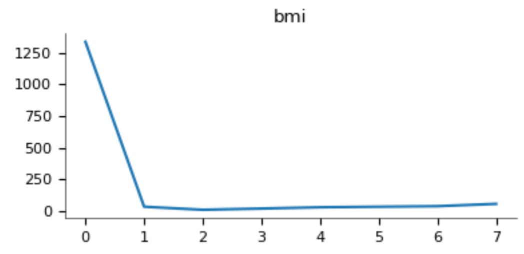
index	age	bmi	children	charges
count	1338.0	1338.0	1338.0	1338.0
mean	39.20702541106129	30.66339686098655	1.0949177877429	13270.422265141257
std	14.049960379216154	6.098186911679014	1.205492739781914	12110.011236694001
min	18.0	15.96	0.0	1121.8739
25%	27.0	26.29625	0.0	4740.28715
50%	39.0	30.4	1.0	9382.033
75%	51.0	34.69375	2.0	16639.912515
max	64.0	53.13	5.0	63770.42801

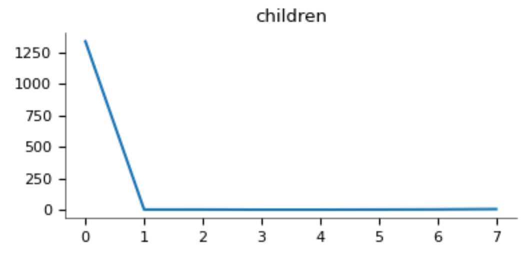
Show 25 per page

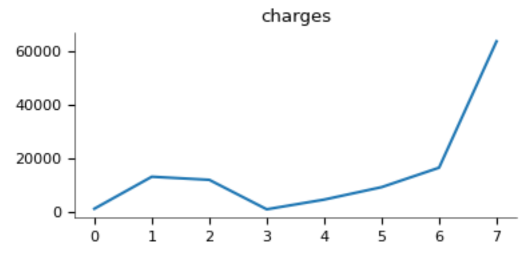
Like what you see? Visit the data table notebook to learn more about interactive tables.

Values

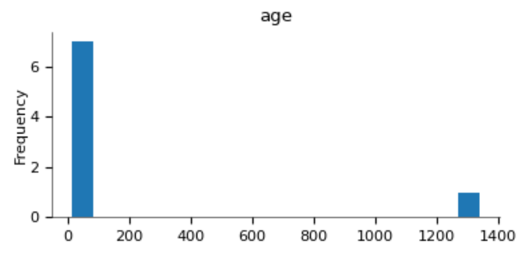


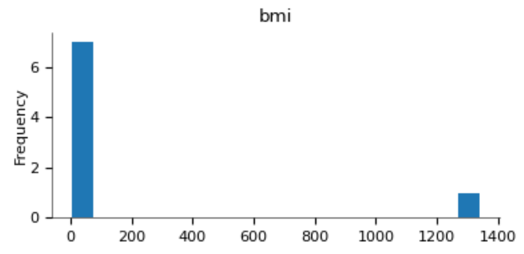


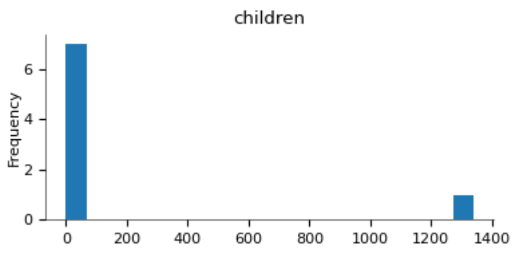


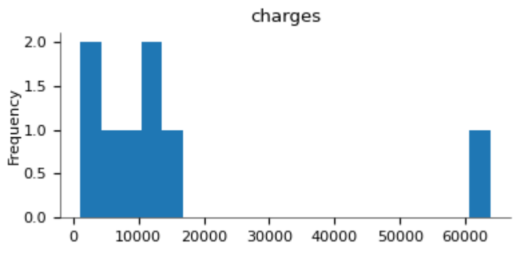


Distributions

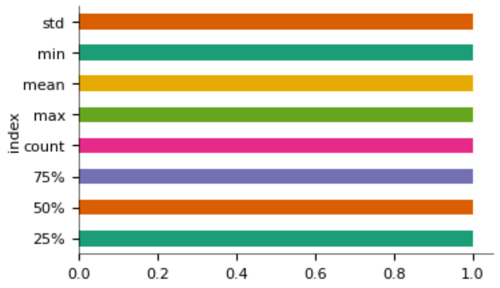




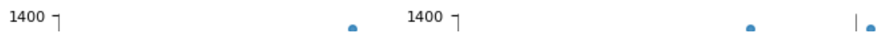




Categorical distributions



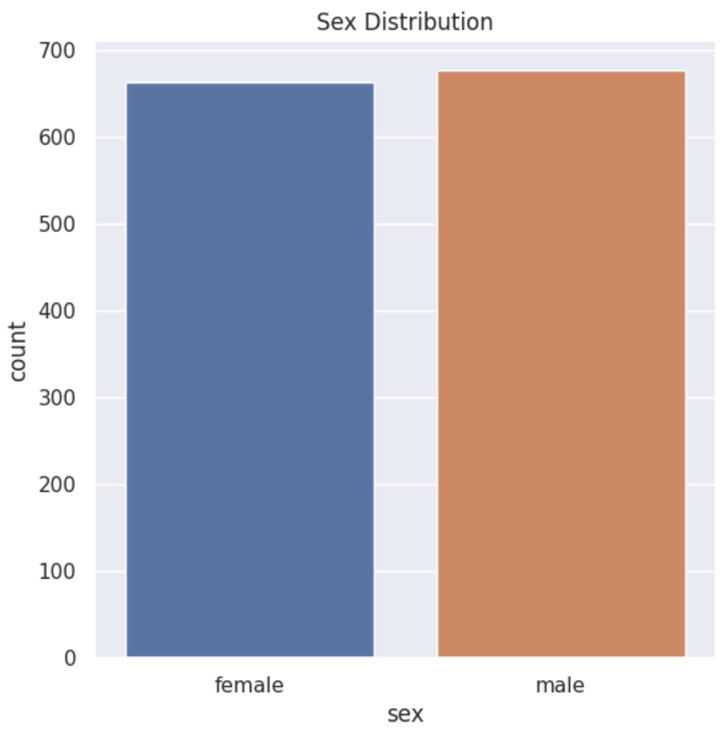
2-d distributions



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

```
    <ipython-input-14-28228e9c3528>:4: UserWarning:

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

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



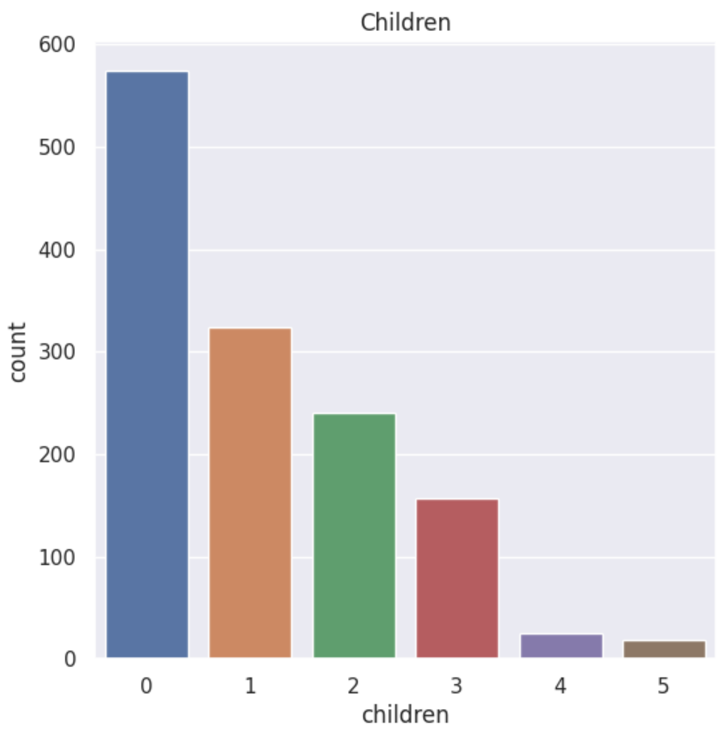
```
insurance_dataset['sex'].value_counts()

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

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

```
<ipython-input-17-81b69896b0d5>: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

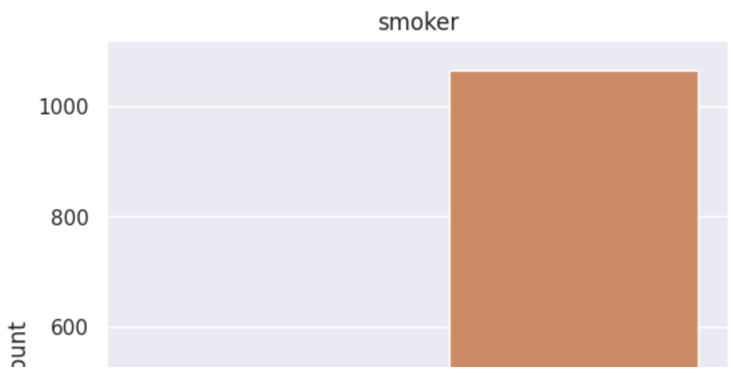
```
# children column
plt.figure(figsize=(6,6))
sns.countplot(x='children', data=insurance_dataset)
plt.title('Children')
plt.show()
```



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

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

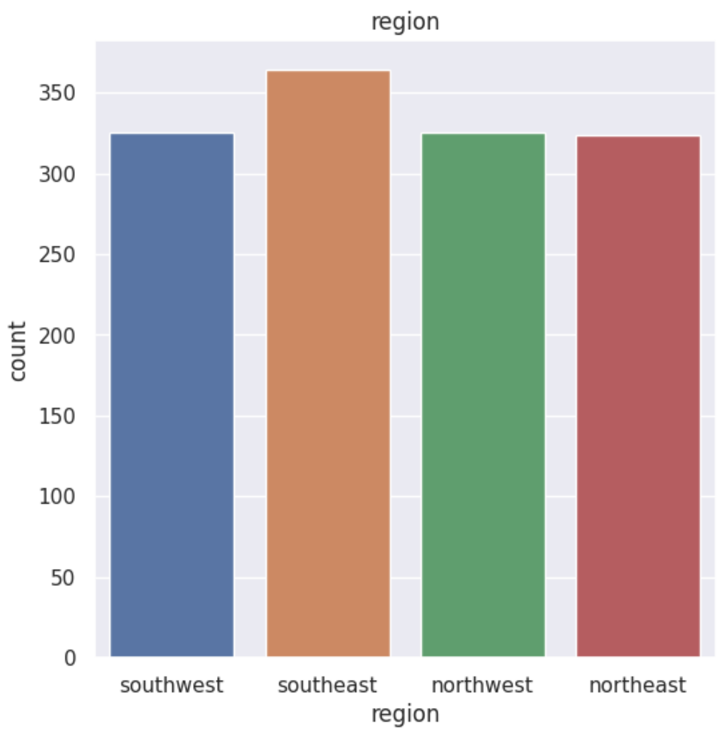
```
# smoker column
plt.figure(figsize=(6,6))
sns.countplot(x='smoker', data=insurance_dataset)
plt.title('smoker')
plt.show()
```



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

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

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

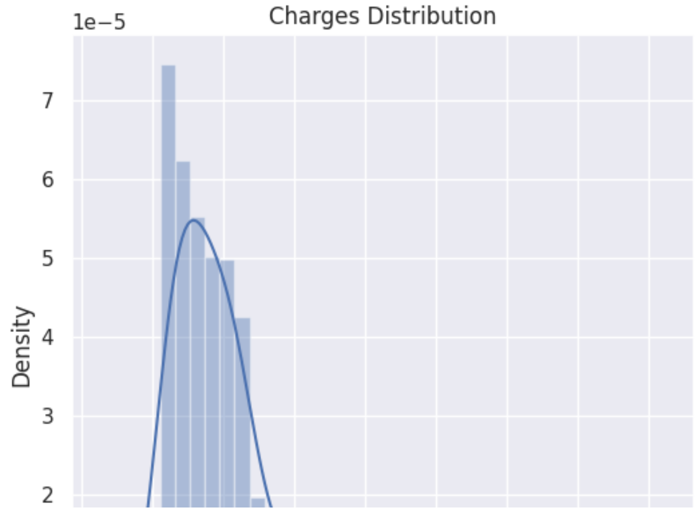


```
insurance_dataset['region'].value_counts()
```

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

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

```
<ipython-input-24-a2fe9b394a51>: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)
```

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)
# prediction on test data
test_data_prediction =regressor.predict(X_test)
print("*************************")
# R squared value
r2_test = metrics.r2_score(Y_test, test_data_prediction)
print('R squared vale : ', r2_test)
```

```
R squared vale :  0.751505643411174
*************************
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(
```

✓ 0s completed at 3:16 PM