MAJOR PROJECT

Name: Likhitha T Murthy

College: Vidyavardhaka College of Engineering

Branch: CS&E(AI&ML)

Year: 1st Year

Github Link: https://github.com/LikhithaMurthy/Major_Project

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('/content/insurance.csv')

data.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

```
data.isnull().sum()
```

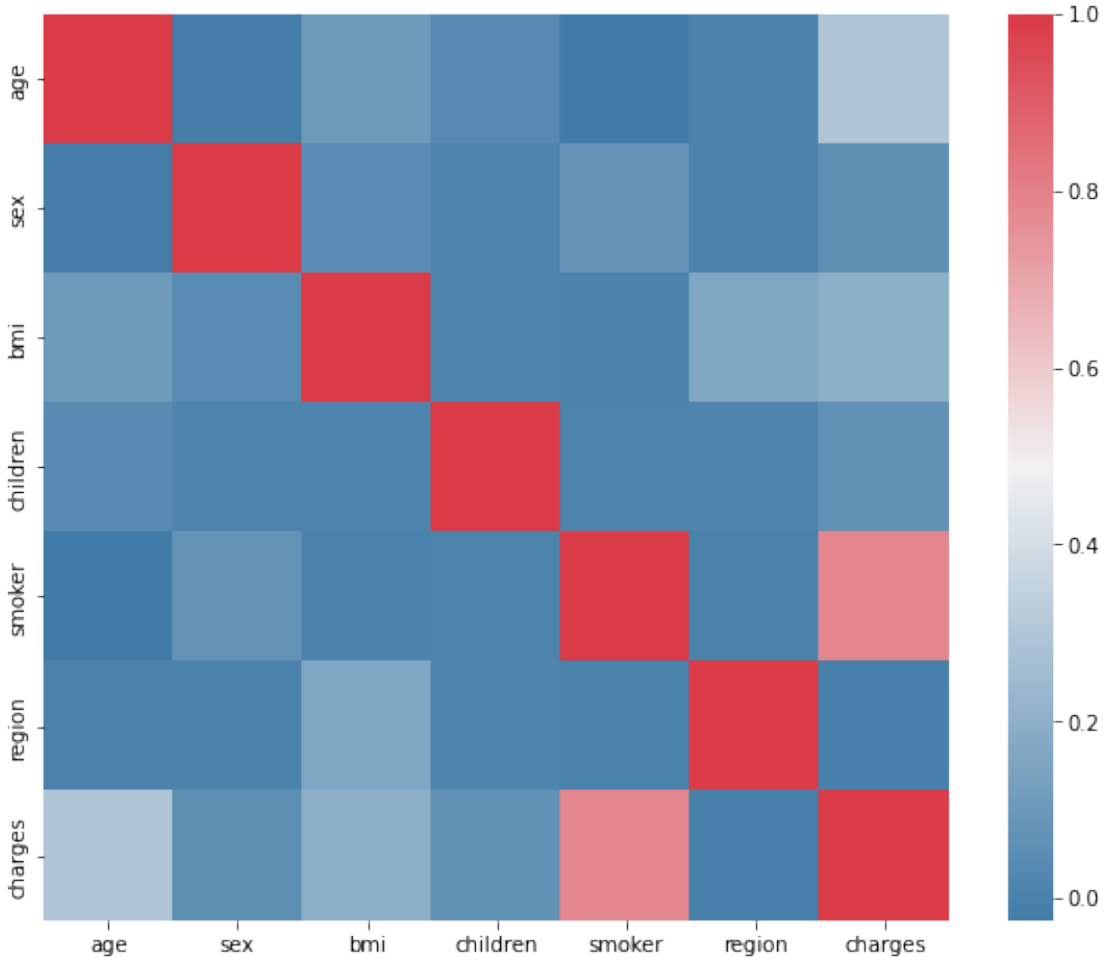
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
#sex
le = LabelEncoder()
le.fit(data.sex.drop_duplicates())
data.sex = le.transform(data.sex)
# smoker or not
le.fit(data.smoker.drop_duplicates())
data.smoker = le.transform(data.smoker)
#region
le.fit(data.region.drop_duplicates())
data.region = le.transform(data.region)

data.corr()['charges'].sort_values()
```

```
region     -0.006208
sex         0.057292
children    0.067998
bmi         0.198341
age         0.299008
smoker      0.787251
charges     1.000000
Name: charges, dtype: float64
```

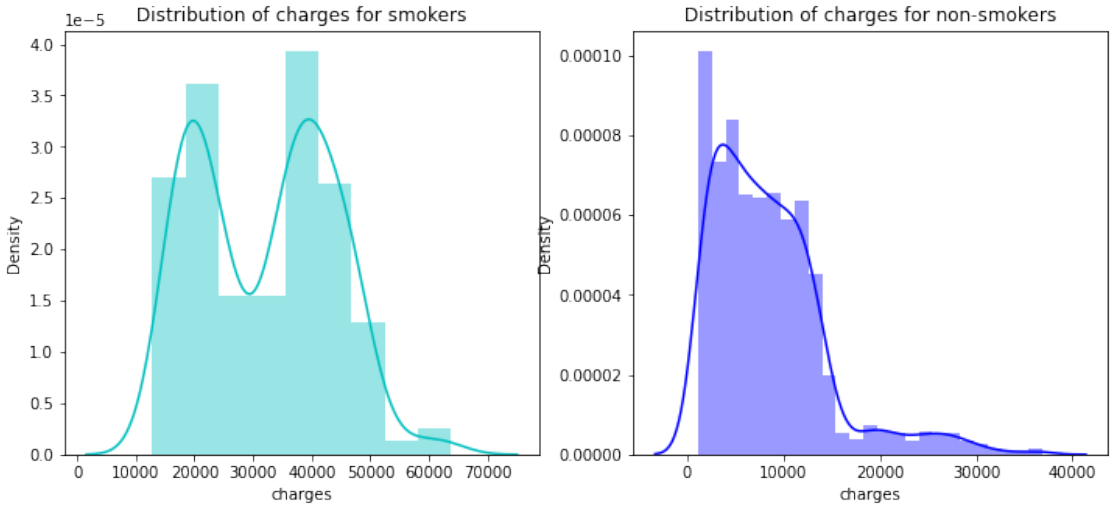
```
f, ax = plt.subplots(figsize=(10, 8))
corr = data.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(240,10,as_cmap=True),
            square=True, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0726034ed0>
```



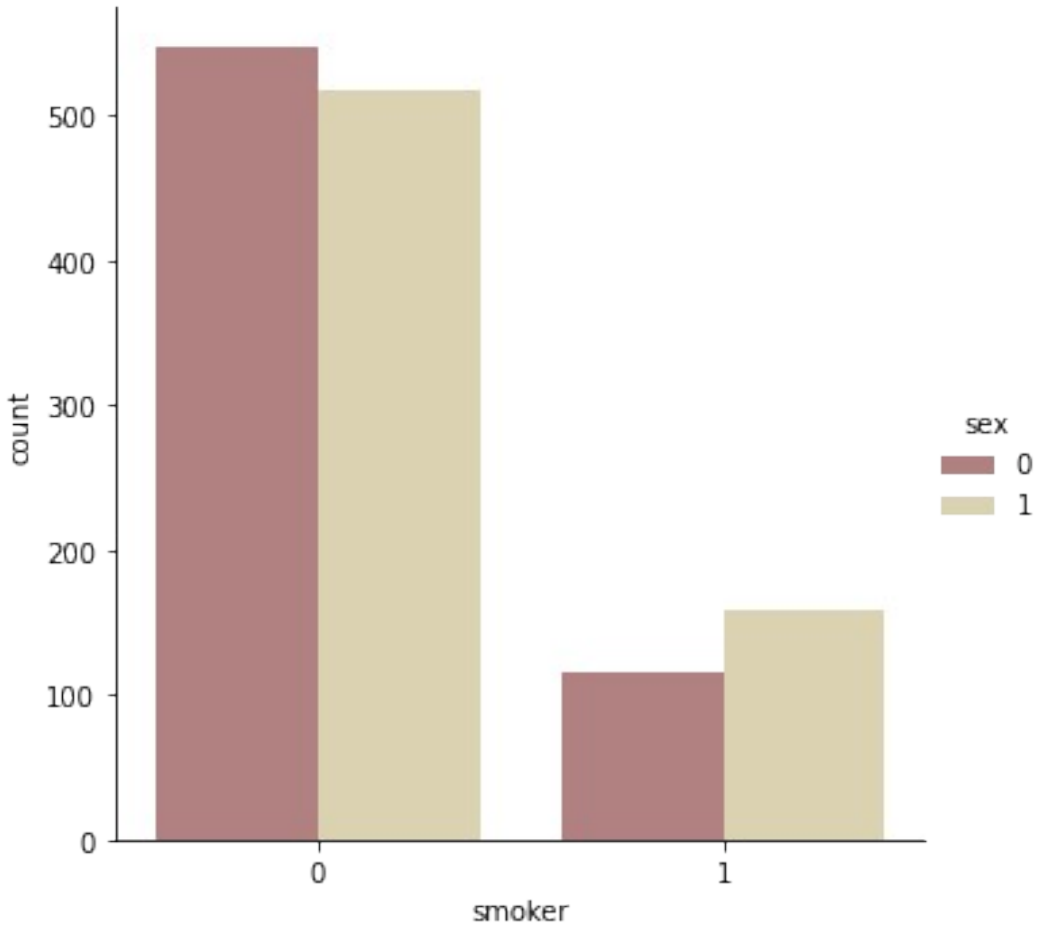

```
f= plt.figure(figsize=(12,5))

ax=f.add_subplot(121)
sns.distplot(data[(data.smoker == 1)]["charges"],color='c',ax=ax)
ax.set_title('Distribution of charges for smokers')

ax=f.add_subplot(122)
sns.distplot(data[(data.smoker == 0)]['charges'],color='b',ax=ax)
ax.set_title('Distribution of charges for non-smokers')
```

```
Text(0.5, 1.0, 'Distribution of charges for non-smokers')
```



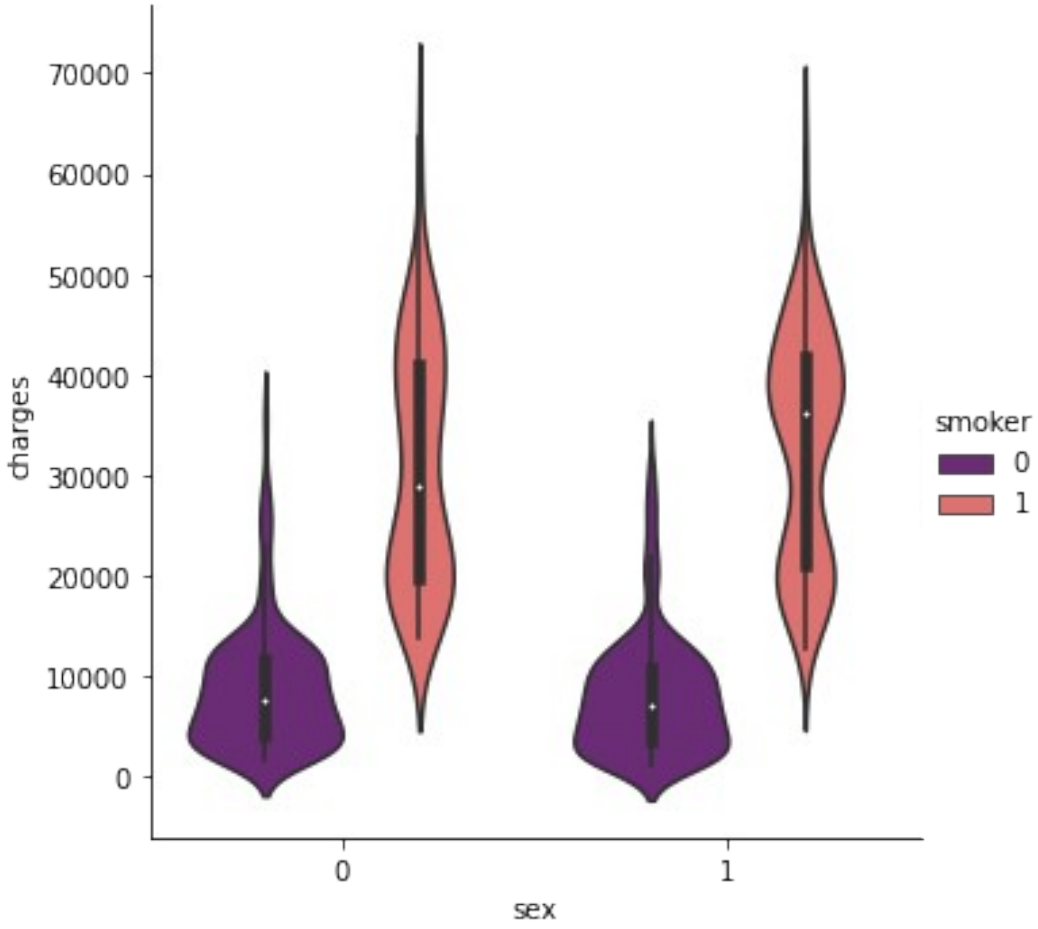

```
sns.catplot(x="smoker", kind="count",hue = 'sex', palette="pink",
data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0725e6be50>
```

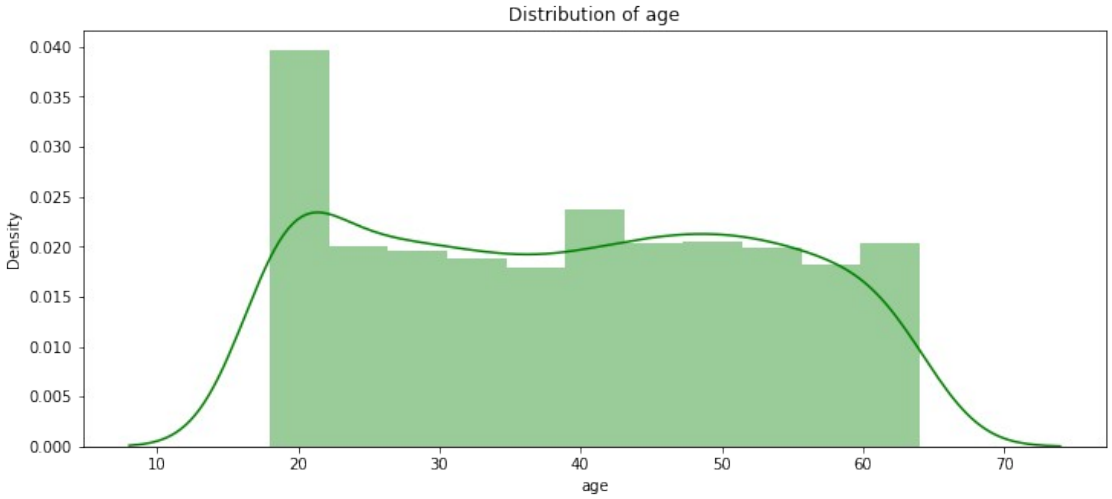


```
sns.catplot(x="sex", y="charges", hue="smoker",
            kind="violin", data=data, palette = 'magma')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0725dbca50>
```

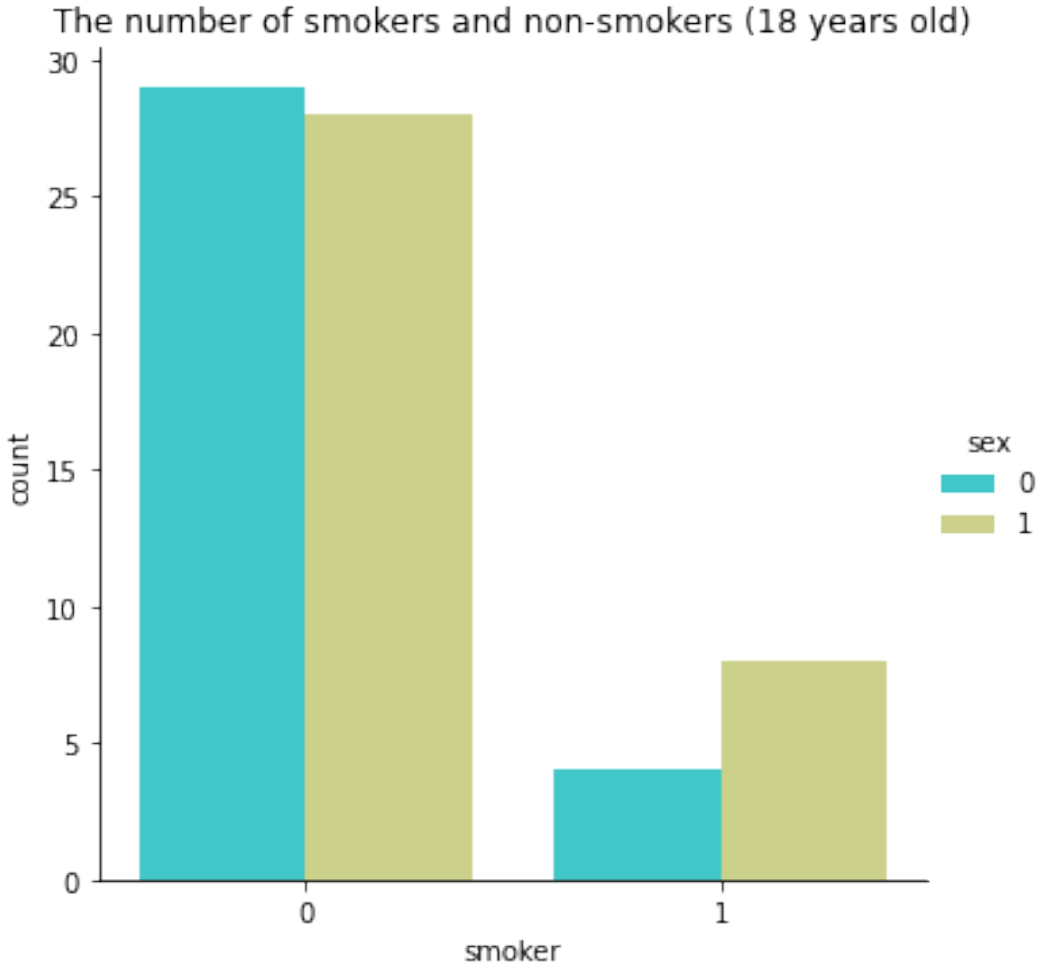



```
plt.figure(figsize=(12,5))
plt.title("Distribution of age")
ax = sns.distplot(data["age"], color = 'g')
```



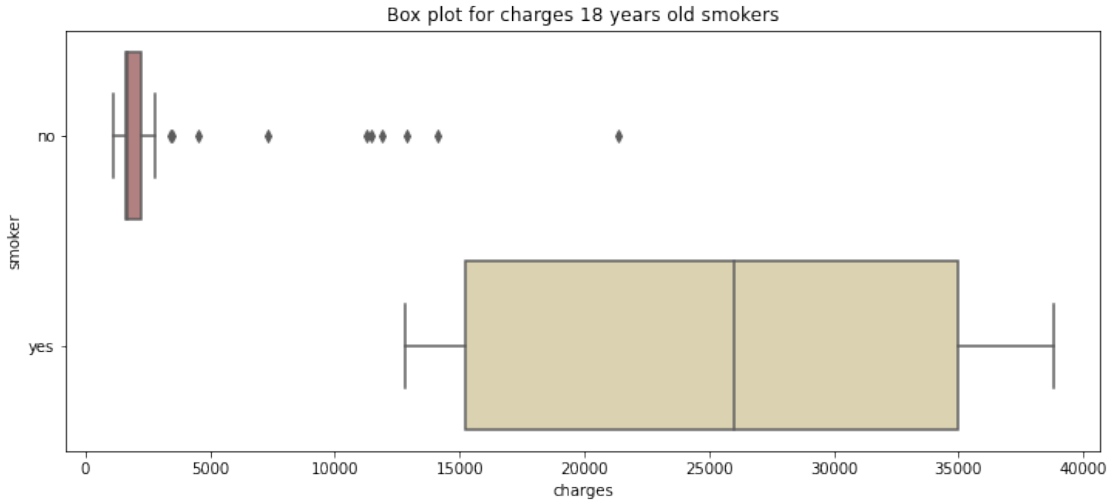

```
sns.catplot(x="smoker", kind="count",hue = 'sex', palette="rainbow",
data=data[(data.age == 18)])
plt.title("The number of smokers and non-smokers (18 years old)")
```

```
Text(0.5, 1.0, 'The number of smokers and non-smokers (18 years old)')
```



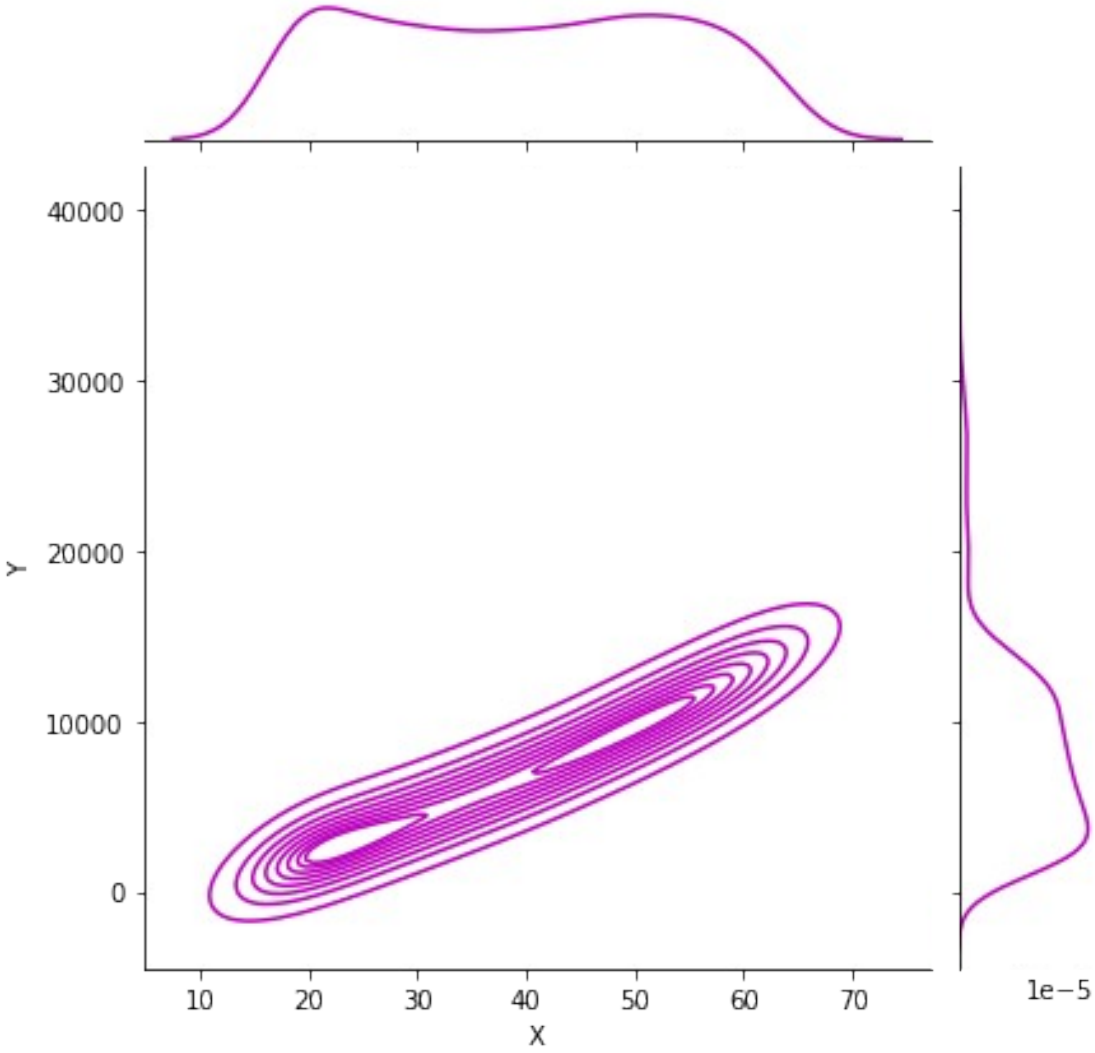
```
plt.figure(figsize=(12,5))
plt.title("Box plot for charges 18 years old smokers")
sns.boxplot(y="smoker", x="charges", data = data[(data.age == 18)] ,
orient="h", palette = 'pink')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc64678d4d0>
```



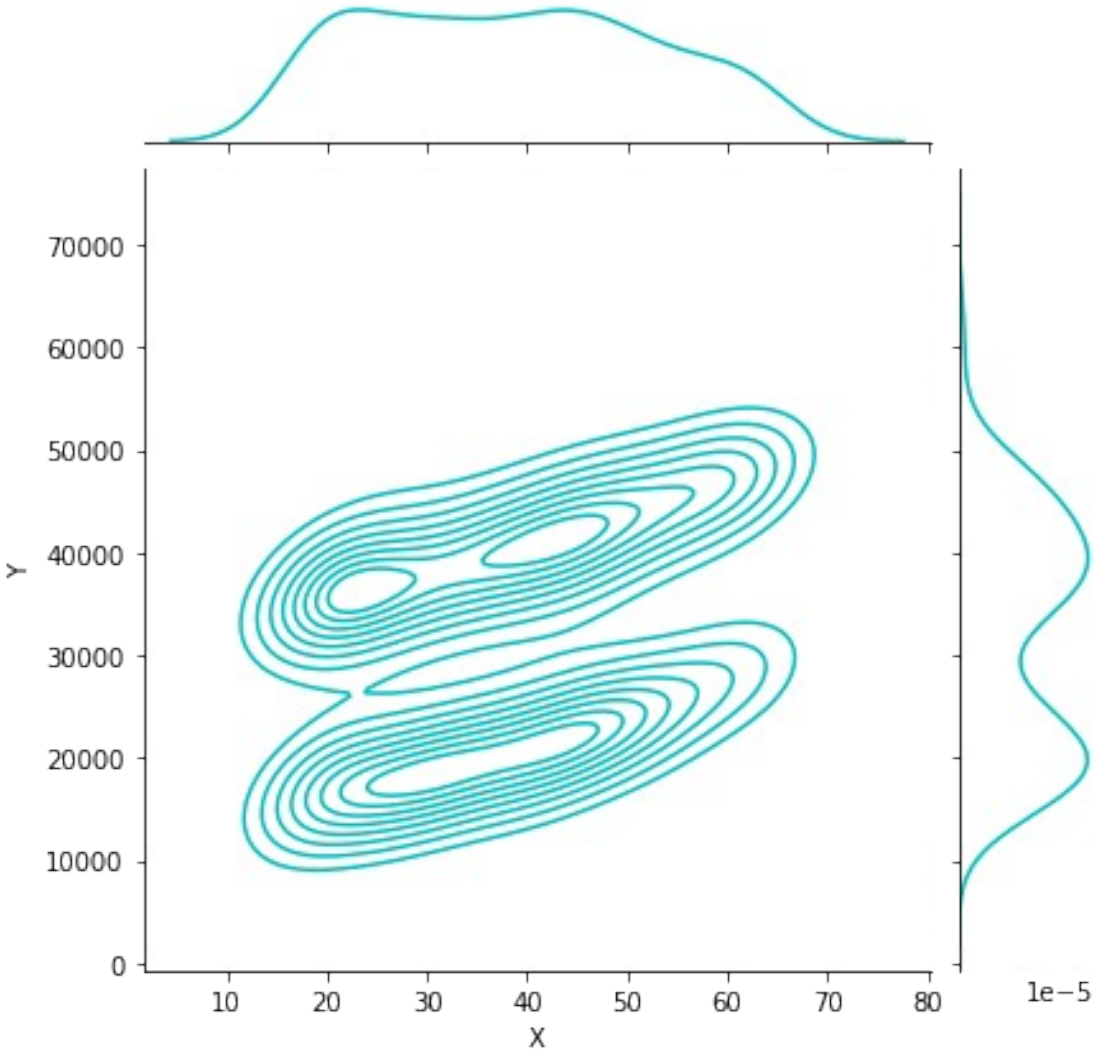

```
g = sns.jointplot(x="age", y="charges", data = data[(data.smoker ==
0)],kind="kde", color="m")
g.plot_joint(plt.scatter, c="w", s=30, linewidth=1)
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("X", "Y")
ax.set_title('Distribution of charges and age for non-smokers')
```

```
Text(0.5, 1.0, 'Distribution of charges and age for non-smokers')
```



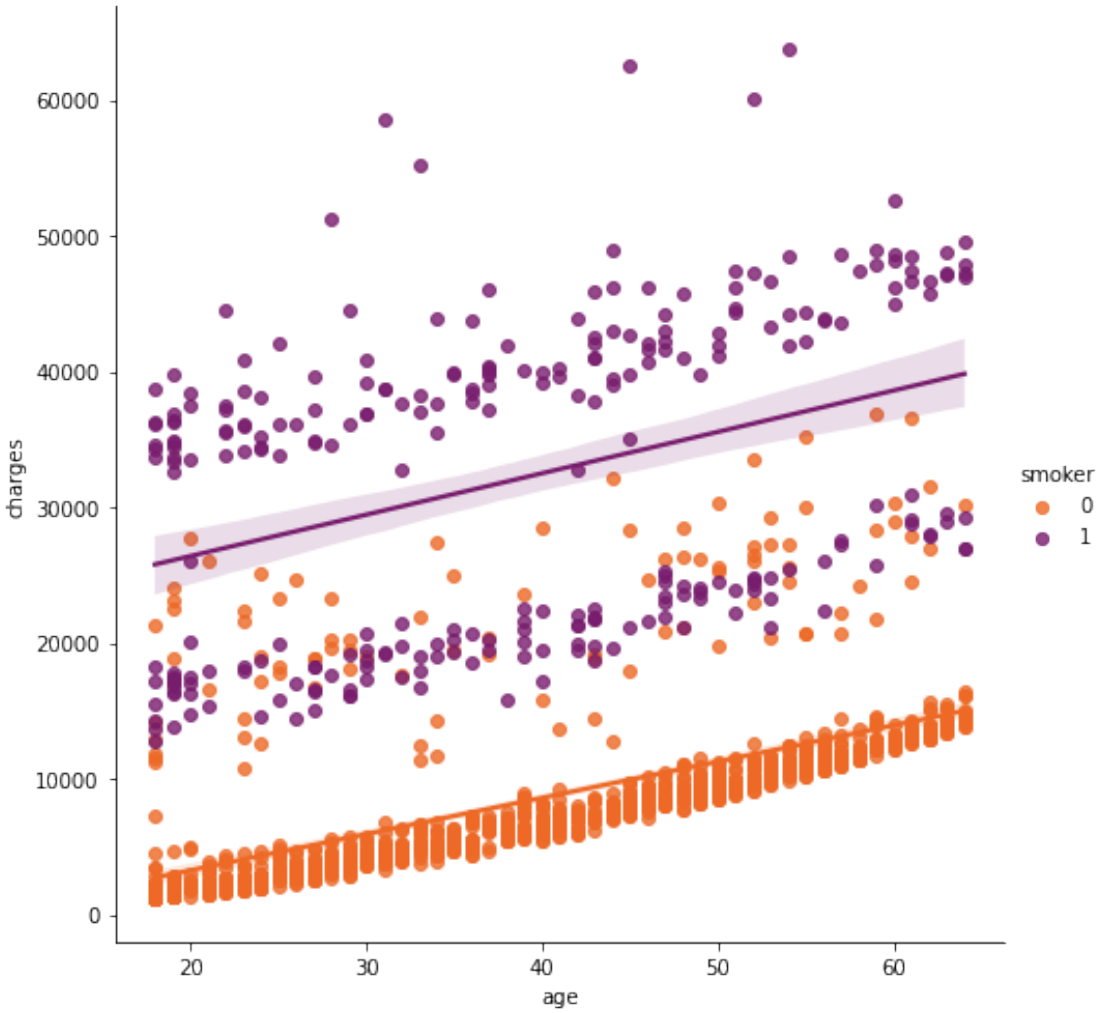

```
g = sns.jointplot(x="age", y="charges", data = data[(data.smoker ==
1)],kind="kde", color="c")
g.plot_joint(plt.scatter, c="w", s=30, linewidth=1)
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("X", "Y")
ax.set_title('Distribution of charges and age for smokers')
```

```
Text(0.5, 1.0, 'Distribution of charges and age for smokers')
```

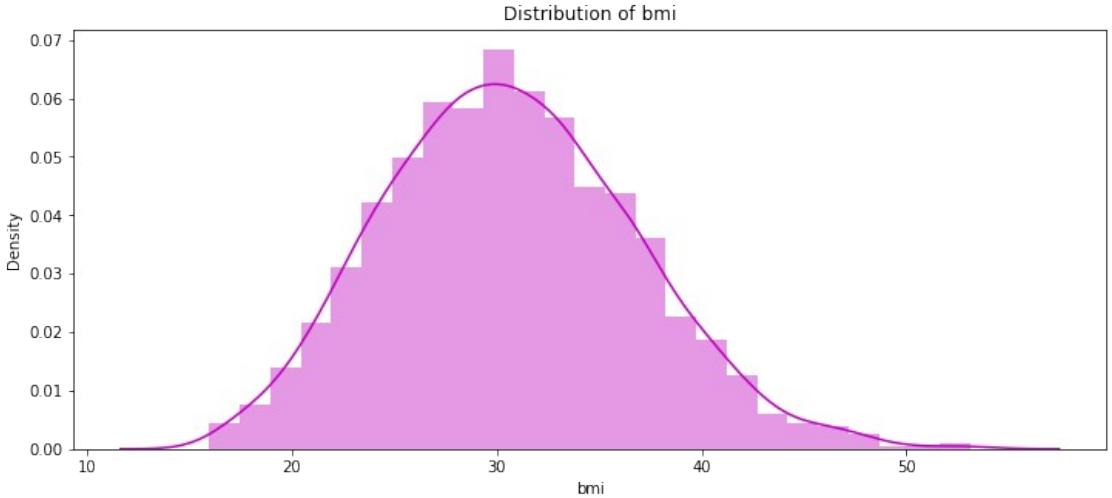



```
sns.lmplot(x="age", y="charges", hue="smoker", data=data, palette =
'inferno_r', size = 7)
ax.set_title('Smokers and non-smokers')
```

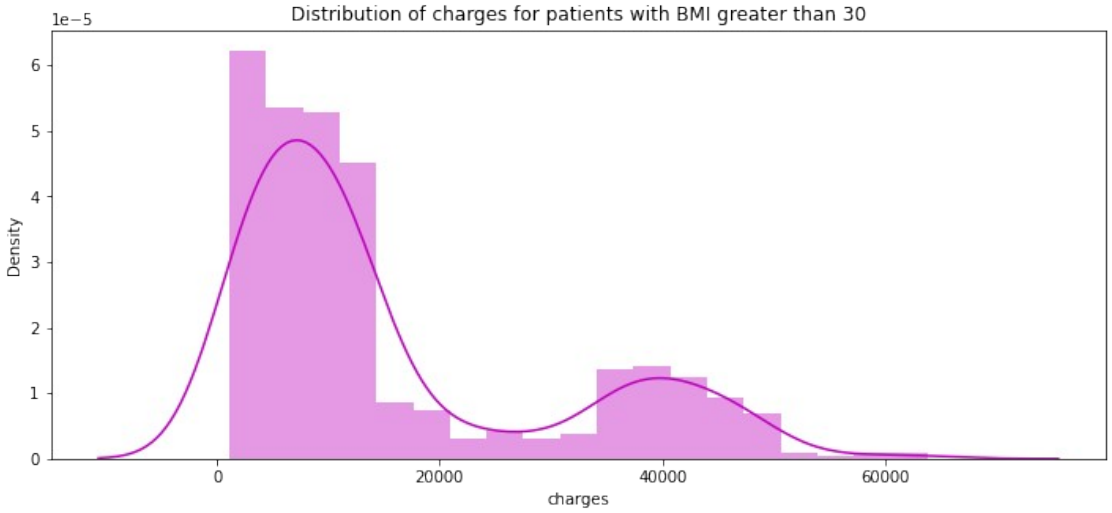
```
Text(0.5, 1.0, 'Smokers and non-smokers')
```



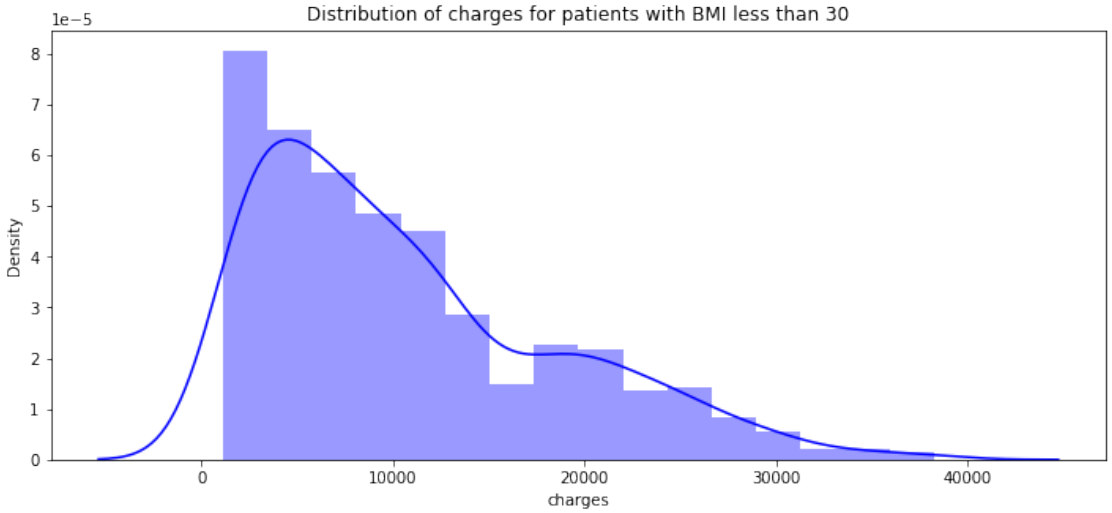

```
plt.figure(figsize=(12,5))
plt.title("Distribution of bmi")
ax = sns.distplot(data["bmi"], color = 'm')
```



```
plt.figure(figsize=(12,5))
plt.title("Distribution of charges for patients with BMI greater than
30")
ax = sns.distplot(data[(data.bmi >= 30)]['charges'], color = 'm')
```

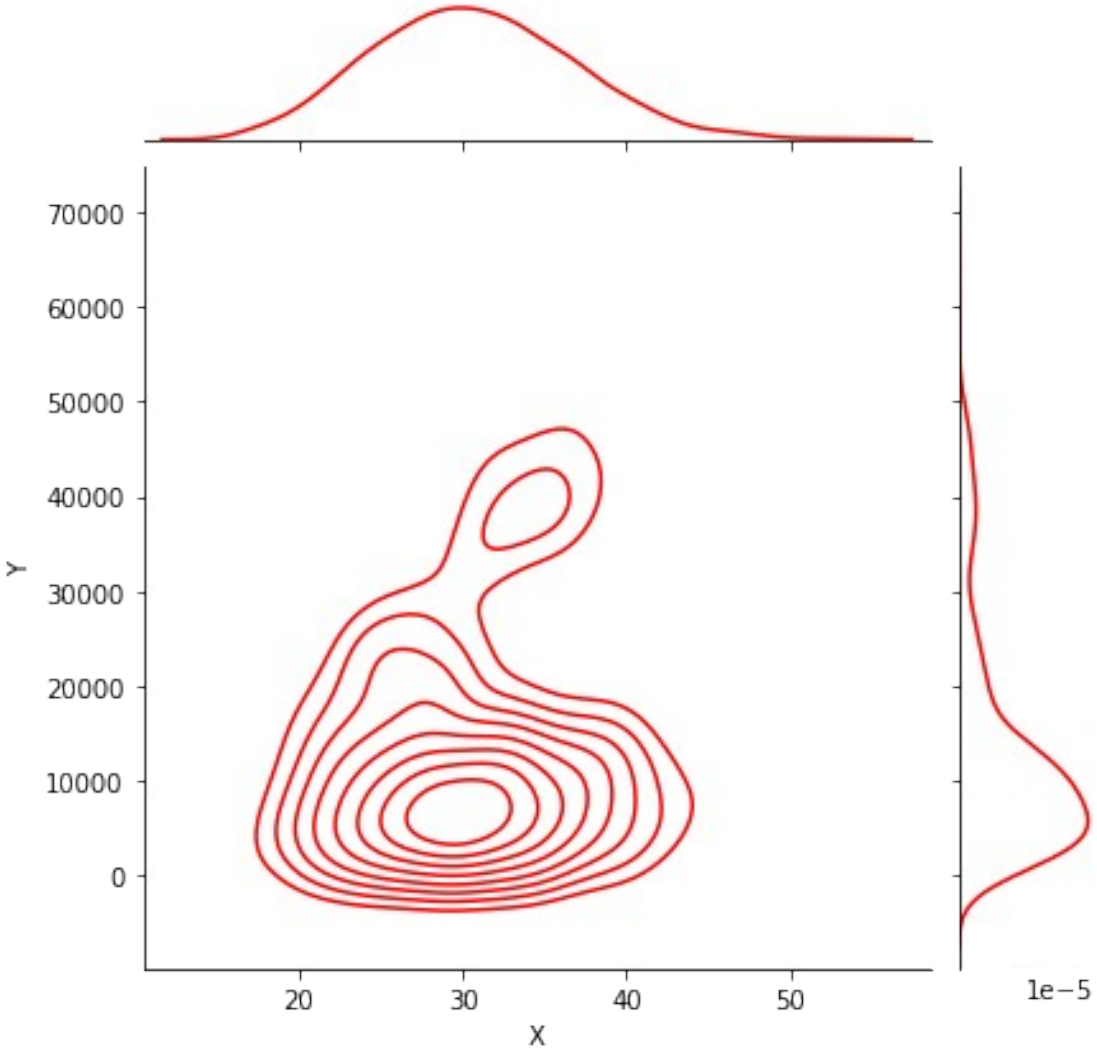



```
plt.figure(figsize=(12,5))
plt.title("Distribution of charges for patients with BMI less than
30")
ax = sns.distplot(data[(data.bmi < 30)]['charges'], color = 'b')
```



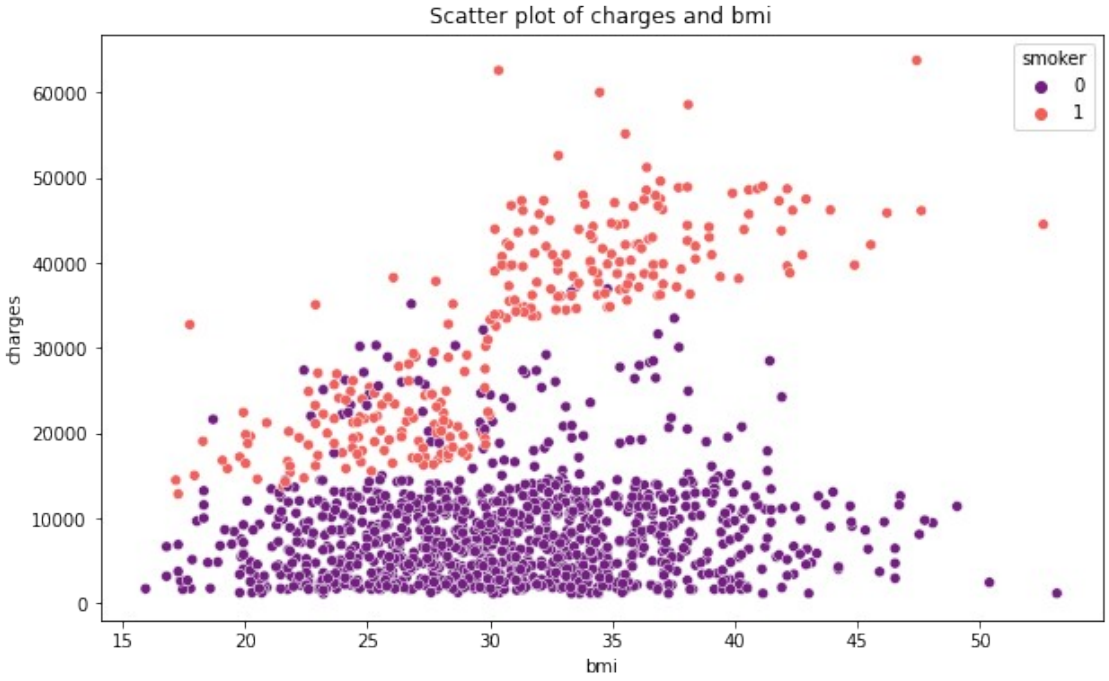

```
g = sns.jointplot(x="bmi", y="charges", data = data,kind="kde",
color="r")
g.plot_joint(plt.scatter, c="w", s=30, linewidth=1)
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("X", "Y")
ax.set_title('Distribution of bmi and charges')
```

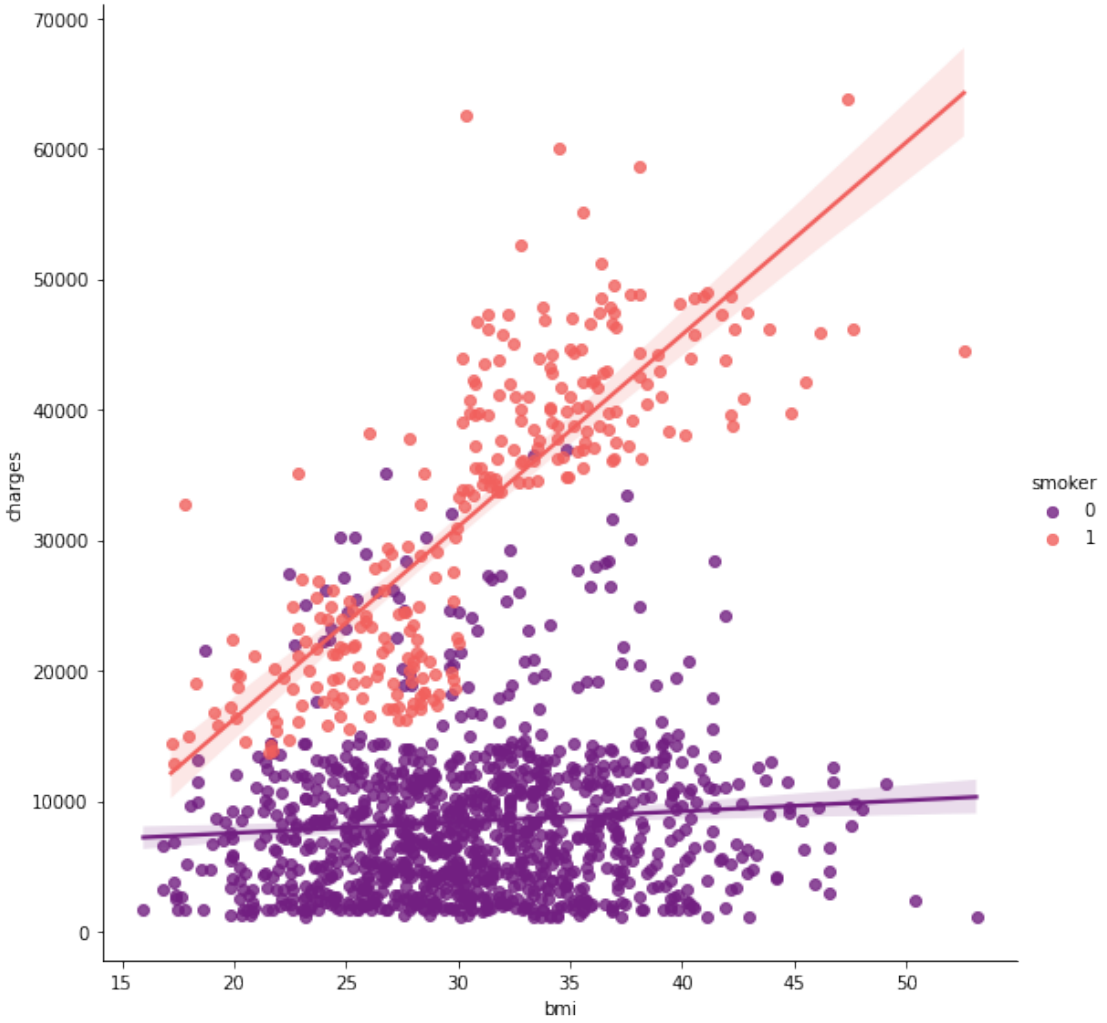
```
Text(0.5, 1.0, 'Distribution of bmi and charges')
```




```
plt.figure(figsize=(10,6))
ax =
sns.scatterplot(x='bmi',y='charges',data=data,palette='magma',hue='smo
ker')
ax.set_title('Scatter plot of charges and bmi')

sns.lmplot(x="bmi", y="charges", hue="smoker", data=data, palette =
'magma', size = 8)
```

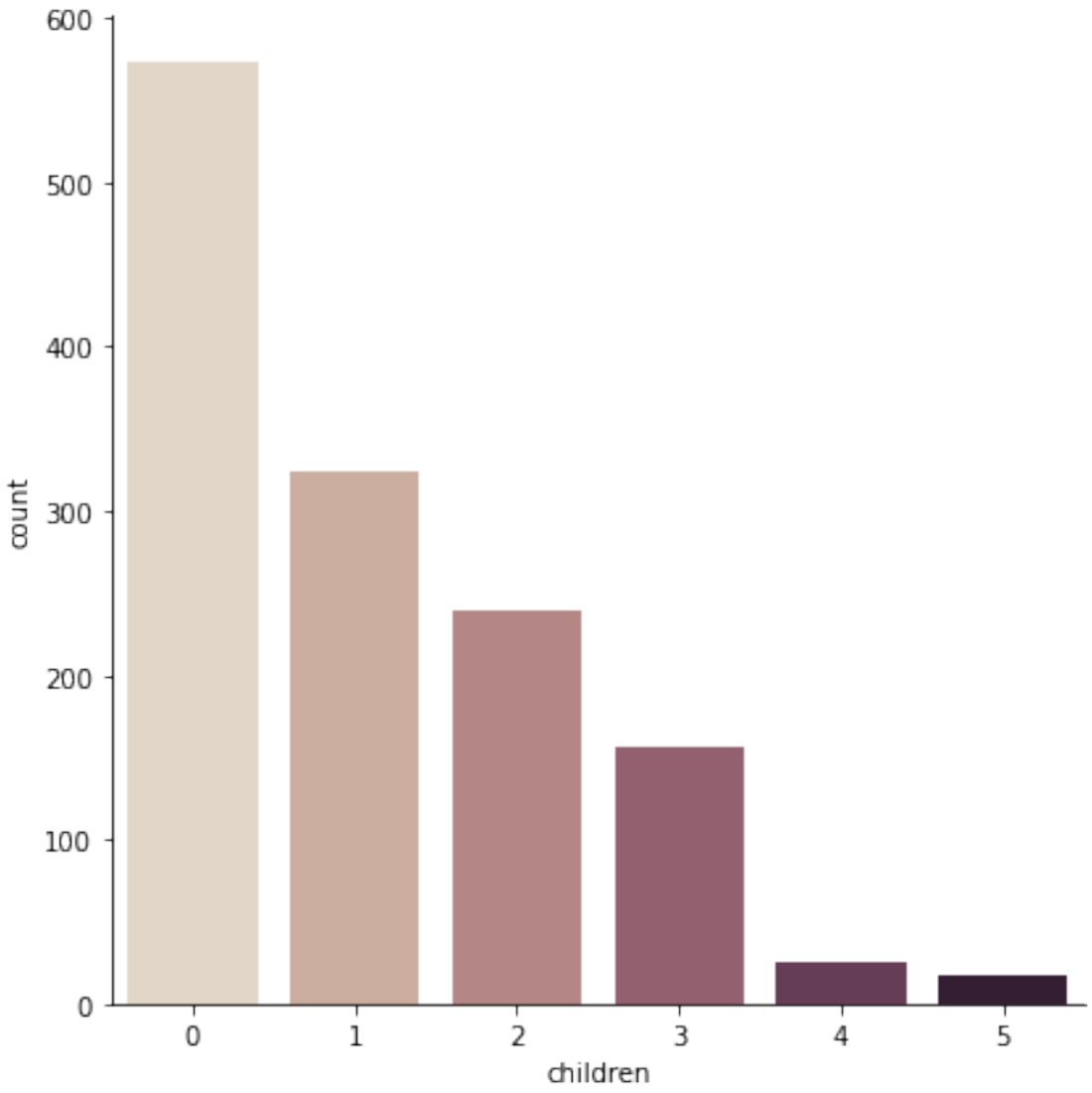
```
<seaborn.axisgrid.FacetGrid at 0x7f0725c5c6d0>
```


Scatter plot of charges and bmi
smoker
0
1
charges
bmi
60000
50000
40000
30000
20000
10000
0
15
20
25
30
35
40
45
50



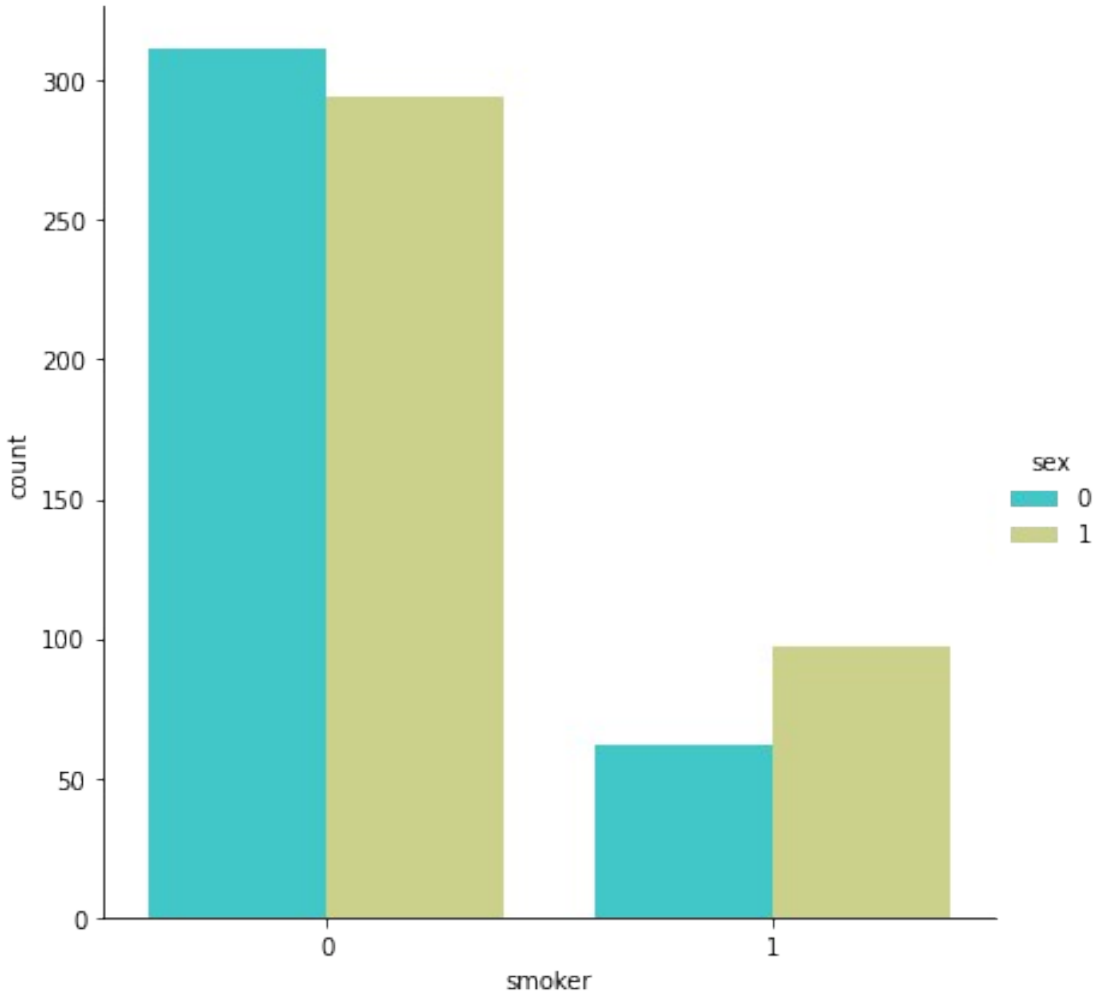

```
sns.catplot(x="children", kind="count", palette="ch:.25", data=data,
size = 6)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0725cecbd0>
```




```
sns.catplot(x="smoker", kind="count", palette="rainbow",hue = "sex",
            data=data[(data.children > 0)], size = 6)
ax.set_title('Smokers and non-smokers who have childrens')
```

```
Text(0.5, 1.0, 'Smokers and non-smokers who have childrens')
```




```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.ensemble import RandomForestRegressor

x = data.drop(['charges'], axis = 1)
y = data.charges

x_train,x_test,y_train,y_test = train_test_split(x,y, random_state =
0)
lr = LinearRegression().fit(x_train,y_train)

y_train_pred = lr.predict(x_train)
y_test_pred = lr.predict(x_test)

print(lr.score(x_test,y_test))
```

```
0.7962732059725786
```

```
X = data.drop(['charges','region'], axis = 1)
Y = data.charges



quad = PolynomialFeatures (degree = 2)
x_quad = quad.fit_transform(X)

X_train,X_test,Y_train,Y_test = train_test_split(x_quad,Y,
random_state = 0)

plr = LinearRegression().fit(X_train,Y_train)

Y_train_pred = plr.predict(X_train)
Y_test_pred = plr.predict(X_test)

print(plr.score(X_test,Y_test))
```

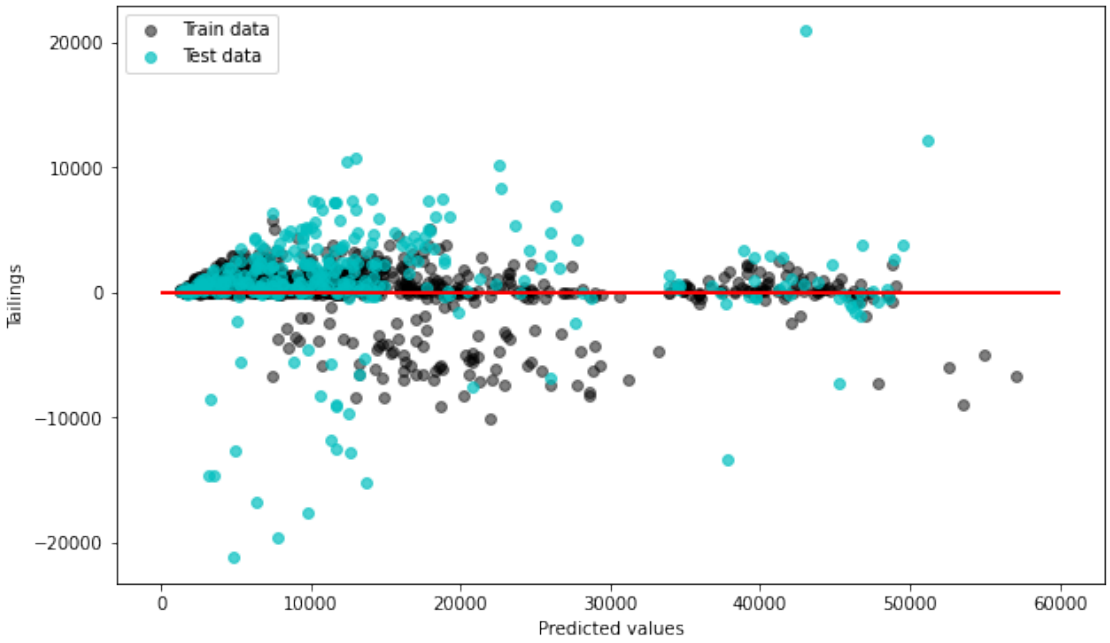
```
0.8849197344147227
```

```
forest = RandomForestRegressor(n_estimators = 100,
                               criterion = 'mse',
                               random_state = 1,
                               n_jobs = -1)
forest.fit(x_train,y_train)
forest_train_pred = forest.predict(x_train)
forest_test_pred = forest.predict(x_test)

print('MSE train data: %.3f, MSE test data: %.3f' % (
mean_squared_error(y_train,forest_train_pred),
mean_squared_error(y_test,forest_test_pred)))
print('R2 train data: %.3f, R2 test data: %.3f' % (
r2_score(y_train,forest_train_pred),
r2_score(y_test,forest_test_pred)))
```

```
MSE train data: 3746684.434, MSE test data: 19965476.411
R2 train data: 0.974, R2 test data: 0.873
```

```
plt.figure(figsize=(10,6))

plt.scatter(forest_train_pred,forest_train_pred - y_train,
          c = 'black', marker = 'o', s = 35, alpha = 0.5,
          label = 'Train data')
plt.scatter(forest_test_pred,forest_test_pred - y_test,
          c = 'c', marker = 'o', s = 35, alpha = 0.7,
          label = 'Test data')
plt.xlabel('Predicted values')
plt.ylabel('Tailings')
plt.legend(loc = 'upper left')
```

```
plt.hlines(y = 0, xmin = 0, xmax = 60000, lw = 2, color = 'red')
plt.show()
```



```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline

import os
for dirname, _, filenames in os.walk('/content/Live.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import warnings

warnings.filterwarnings('ignore')

data = '/content/Live.csv'

df = pd.read_csv(data)

# Exploratory data analysis:

df.shape  # Check shape of the dataset

(7050, 16)

df.head()  # Preview the dataset

                           status_id status_type status_published  \
0  246675545449582_1649696485147474       video   4/22/2018 6:00
1  246675545449582_1649426988507757       photo  4/21/2018 22:45
2  246675545449582_1648730588577397       video   4/21/2018 6:17
3  246675545449582_1648576705259452       photo   4/21/2018 2:29
4  246675545449582_1645700502213739       photo   4/18/2018 3:22

   num_reactions  num_comments  num_shares  num_likes  num_loves
num_wows  \
0            529           512         262        432         92
3
1            150             0           0        150          0
0
2            227           236          57        204         21
1
3            111             0           0        111          0
0
4            213             0           0        204          9
0

   num_hahas  num_sads  num_angrys  Column1  Column2  Column3  Column4

0          1         1           0      NaN      NaN      NaN      NaN
```

```
1          0         0           0      NaN      NaN      NaN      NaN

2          1         0           0      NaN      NaN      NaN      NaN

3          0         0           0      NaN      NaN      NaN      NaN

4          0         0           0      NaN      NaN      NaN      NaN
```

```
df.info() # View summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   status_id         7050 non-null   object 
 1   status_type       7050 non-null   object 
 2   status_published  7050 non-null   object 
 3   num_reactions     7050 non-null   int64  
 4   num_comments      7050 non-null   int64  
 5   num_shares        7050 non-null   int64  
 6   num_likes         7050 non-null   int64  
 7   num_loves         7050 non-null   int64  
 8   num_wows          7050 non-null   int64  
 9   num_hahas         7050 non-null   int64  
 10  num_sads          7050 non-null   int64  
 11  num_angrys        7050 non-null   int64  
 12  Column1           0 non-null      float64
 13  Column2           0 non-null      float64
 14  Column3           0 non-null      float64
 15  Column4           0 non-null      float64
dtypes: float64(4), int64(9), object(3)
memory usage: 881.4+ KB
```

```
df.isnull().sum() # Check for missing values in dataset
```

```
status_id              0
status_type            0
status_published       0
num_reactions          0
num_comments           0
num_shares             0
num_likes              0
num_loves              0
num_wows               0
num_hahas              0
num_sads               0
num_angrys             0
Column1             7050
Column2             7050
```

```
Column3             7050
Column4             7050
dtype: int64
```

```
df.drop(['Column1', 'Column2', 'Column3', 'Column4'], axis=1,
inplace=True) # Drop redundant columns
```

```
df.info() # Again view summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   status_id         7050 non-null   object
 1   status_type       7050 non-null   object
 2   status_published  7050 non-null   object
 3   num_reactions     7050 non-null   int64
 4   num_comments      7050 non-null   int64
 5   num_shares        7050 non-null   int64
 6   num_likes         7050 non-null   int64
 7   num_loves         7050 non-null   int64
 8   num_wows          7050 non-null   int64
 9   num_hahas         7050 non-null   int64
 10  num_sads          7050 non-null   int64
 11  num_angrys        7050 non-null   int64
dtypes: int64(9), object(3)
memory usage: 661.1+ KB
```

```
df.describe() # View the statistical summary of numerical variables
```

```
       num_reactions  num_comments   num_shares    num_likes
num_loves  \
count    7050.000000   7050.000000  7050.000000  7050.000000
7050.000000
mean      230.117163    224.356028    40.022553   215.043121
12.728652
std       462.625309    889.636820   131.599965   449.472357
39.972930
min         0.000000      0.000000     0.000000     0.000000
0.000000
25%        17.000000      0.000000     0.000000    17.000000
0.000000
50%        59.500000      4.000000     0.000000    58.000000
0.000000
75%       219.000000     23.000000     4.000000   184.750000
3.000000
max      4710.000000  20990.000000  3424.000000  4710.000000
657.000000

          num_wows    num_hahas     num_sads   num_angrys
```

```
count  7050.000000  7050.000000  7050.000000  7050.000000
mean      1.289362     0.696454     0.243688     0.113191
std       8.719650     3.957183     1.597156     0.726812
min       0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     0.000000     0.000000
max     278.000000   157.000000    51.000000    31.000000
```

```
# Explore status_id variable:
df['status_id'].unique() # view the labels in the variable
```

```
array(['246675545449582_1649696485147474',
       '246675545449582_1649426988507757',
       '246675545449582_1648730588577397', ...,
       '1050855161656896_1060126464063099',
       '1050855161656896_1058663487542730',
       '1050855161656896_1050858841656528'], dtype=object)
```

```
len(df['status_id'].unique()) # view how many different types of
variables are there
```

```
6997
```

```
# Explore status_published variable:
df['status_published'].unique() # view the labels in the variable
```

```
array(['4/22/2018 6:00', '4/21/2018 22:45', '4/21/2018 6:17', ...,
       '9/21/2016 23:03', '9/20/2016 0:43', '9/10/2016 10:30'],
      dtype=object)
```

```
len(df['status_published'].unique()) # view how many different types
of variables are there
```

```
6913
```

```
# Explore status_type variable:
df['status_type'].unique() # view the labels in the variable
```

```
array(['video', 'photo', 'link', 'status'], dtype=object)
```

```
len(df['status_type'].unique()) # view how many different types of
variables are there
```

```
4
```

```
# Drop status_id and status_published variable from the dataset
df.drop(['status_id', 'status_published'], axis=1, inplace=True)
```

```
df.info() # View the summary of dataset again
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
```

```
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   status_type    7050 non-null   int64
 1   num_reactions  7050 non-null   int64
 2   num_comments   7050 non-null   int64
 3   num_shares     7050 non-null   int64
 4   num_likes      7050 non-null   int64
 5   num_loves      7050 non-null   int64
 6   num_wows       7050 non-null   int64
 7   num_hahas      7050 non-null   int64
 8   num_sads       7050 non-null   int64
 9   num_angrys     7050 non-null   int64
dtypes: int64(10)
memory usage: 550.9 KB
```

```
df.head() # Preview the dataset again
```

```
   status_type  num_reactions  num_comments  num_shares  num_likes
num_loves  \
0            3            529           512         262        432
92
1            1            150             0           0        150
0
2            3            227           236          57        204
21
3            1            111             0           0        111
0
4            1            213             0           0        204
9

   num_wows  num_hahas  num_sads  num_angrys
0         3          1         1           0
1         0          0         0           0
2         1          1         0           0
3         0          0         0           0
4         0          0         0           0
```

```
# Declare feature vector and target variable:
X = df
y = df['status_type']

# Convert categorical variable into integers:

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X['status_type'] = le.fit_transform(X['status_type'])
y = le.transform(y)

X.info() #View the summary of X
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   status_type    7050 non-null   int64
 1   num_reactions  7050 non-null   int64
 2   num_comments   7050 non-null   int64
 3   num_shares     7050 non-null   int64
 4   num_likes      7050 non-null   int64
 5   num_loves      7050 non-null   int64
 6   num_wows       7050 non-null   int64
 7   num_hahas      7050 non-null   int64
 8   num_sads       7050 non-null   int64
 9   num_angrys     7050 non-null   int64
dtypes: int64(10)
memory usage: 550.9 KB
```

```
X.head() # Preview the dataset X
```

```
   status_type  num_reactions  num_comments  num_shares  num_likes
num_loves  \
0            3            529           512         262        432
92
1            1            150             0           0        150
0
2            3            227           236          57        204
21
3            1            111             0           0        111
0
4            1            213             0           0        204
9

   num_wows  num_hahas  num_sads  num_angrys
0         3          1         1           0
1         0          0         0           0
2         1          1         0           0
3         0          0         0           0
4         0          0         0           0
```

```
# Feature Scaling:

cols = X.columns

from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
X = ms.fit_transform(X)

X = pd.DataFrame(X, columns=[cols])

X.head()
```

```
  status_type num_reactions num_comments num_shares num_likes
num_loves  \
0    1.000000      0.112314     0.024393   0.076519  0.091720
0.140030
1    0.333333      0.031847     0.000000   0.000000  0.031847
0.000000
2    1.000000      0.048195     0.011243   0.016647  0.043312
0.031963
3    0.333333      0.023567     0.000000   0.000000  0.023567
0.000000
4    0.333333      0.045223     0.000000   0.000000  0.043312
0.013699

   num_wows num_hahas  num_sads num_angrys
0  0.010791  0.006369  0.019608        0.0
1  0.000000  0.000000  0.000000        0.0
2  0.003597  0.006369  0.000000        0.0
3  0.000000  0.000000  0.000000        0.0
4  0.000000  0.000000  0.000000        0.0
```

```
#12. K-Means model with two clusters
#Table of Contents

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)
```

```
KMeans(n_clusters=2, random_state=0)
```

```
# K-Means model parameters study:
kmeans.cluster_centers_
```

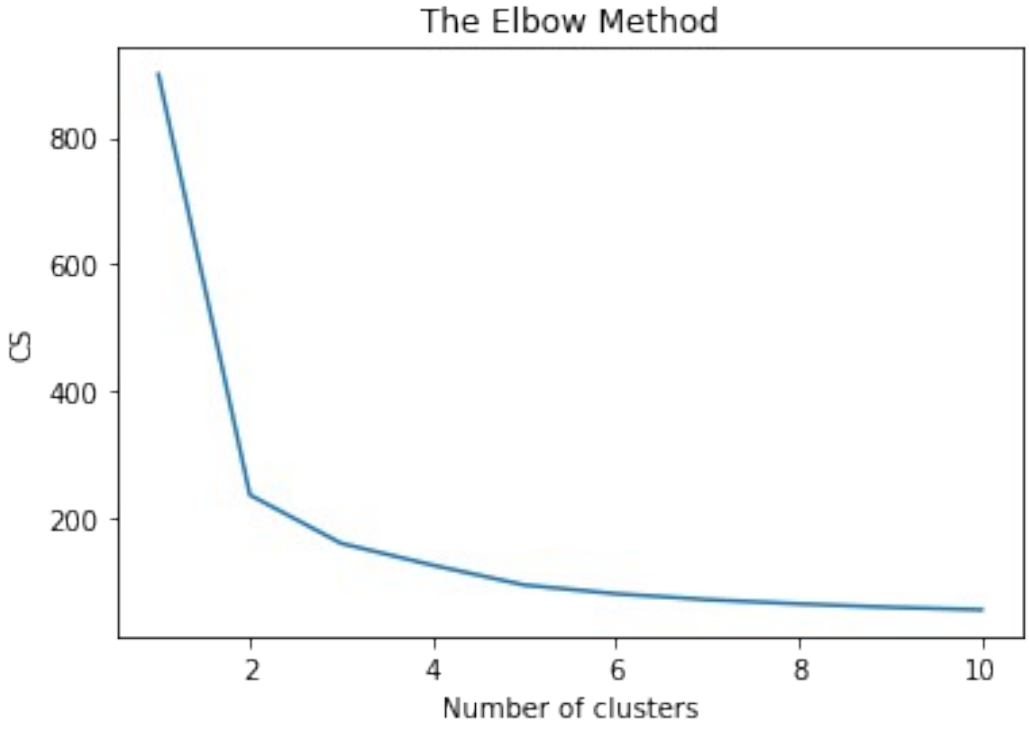
```
array([[3.28506857e-01, 3.90710874e-02, 7.54854864e-04, 7.53667113e-
04,
        3.85438884e-02, 2.17448568e-03, 2.43721364e-03, 1.20039760e-
03,
        2.75348016e-03, 1.45313276e-03],
       [9.54921576e-01, 6.46330441e-02, 2.67028654e-02, 2.93171709e-
02,
        5.71231462e-02, 4.71007076e-02, 8.18581889e-03, 9.65207685e-
03,
        8.04219428e-03, 7.19501847e-03]])
```

```
kmeans.inertia_ # inertia
```

```
237.75726404419646
```

```
# Use elbow method to find optimal number of clusters:

from sklearn.cluster import KMeans
cs = []
```

```
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter =
300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    cs.append(kmeans.inertia_)
plt.plot(range(1, 11), cs)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('CS')
plt.show()
```




```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2,random_state=0)

kmeans.fit(X)

labels = kmeans.labels_

# check how many of the samples were correctly labeled

correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." %
(correct_labels, y.size))
```

```
print('Accuracy score: {0:0.2f}'.
format(correct_labels/float(y.size)))
```

```
Result: 63 out of 7050 samples were correctly labeled.
Accuracy score: 0.01
```

```
# K-Means model with different clusters:

# K-Means model with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." %
(correct_labels, y.size))
print('Accuracy score: {0:0.2f}'.
format(correct_labels/float(y.size)))
```

```
Result: 138 out of 7050 samples were correctly labeled.
Accuracy score: 0.02
```

```
# K-Means model with 4 clusters
kmeans = KMeans(n_clusters=4, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." %
(correct_labels, y.size))
print('Accuracy score: {0:0.2f}'.
format(correct_labels/float(y.size)))
```

```
Result: 4340 out of 7050 samples were correctly labeled.
Accuracy score: 0.62
```