medical-cost-pred

October 18, 2023

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv(r"C:\Users\amits\Downloads\insurance.csv")
```

[3]:
```
df.head()
```

[3]:

	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

[4]:
```
# Data Preprocessing
```

[5]:
```
df.shape
```

[5]:
```
(1338, 7)
```

[6]:
```
df.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
```

[7]:
```
df.describe()
```

[7]:
```
                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
```

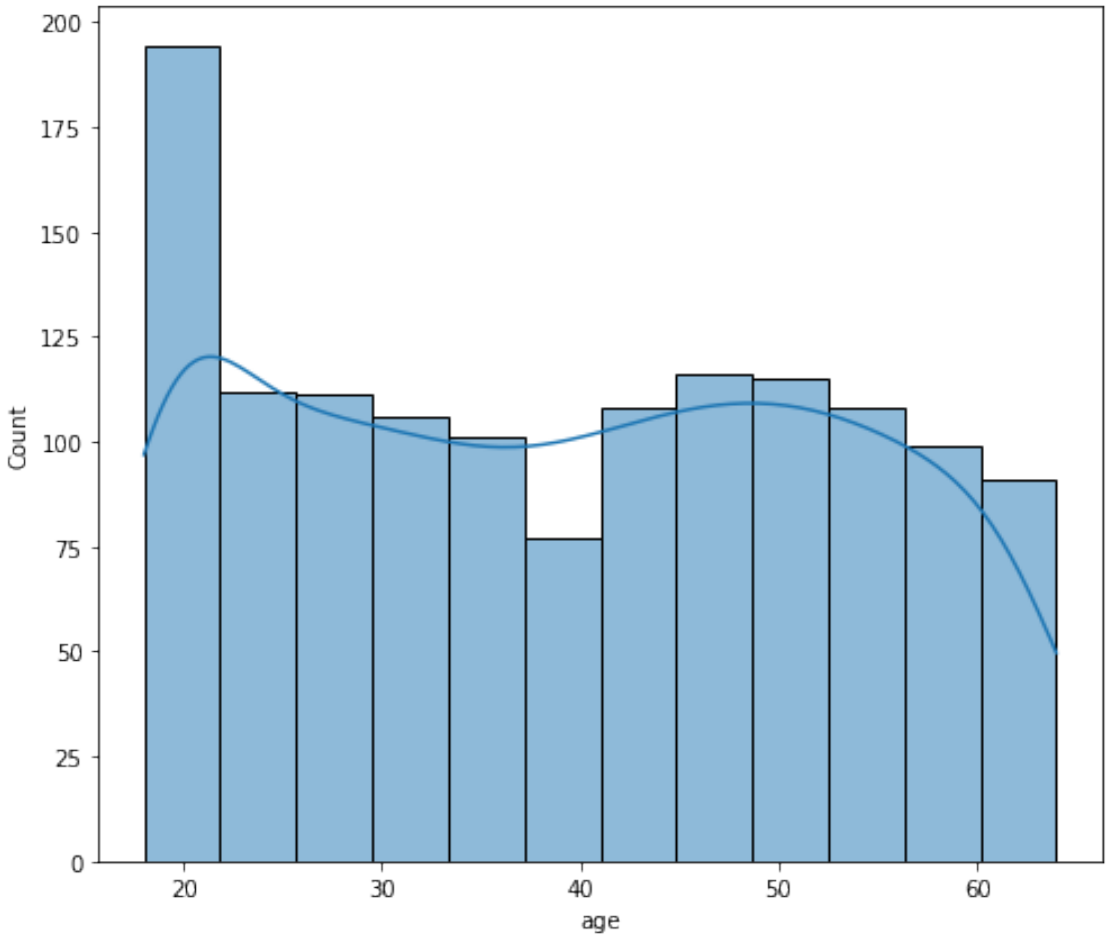
[8]:
```
df.isnull().sum()
```

[8]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

[9]:
```
# Exploratory Data Analysis
```

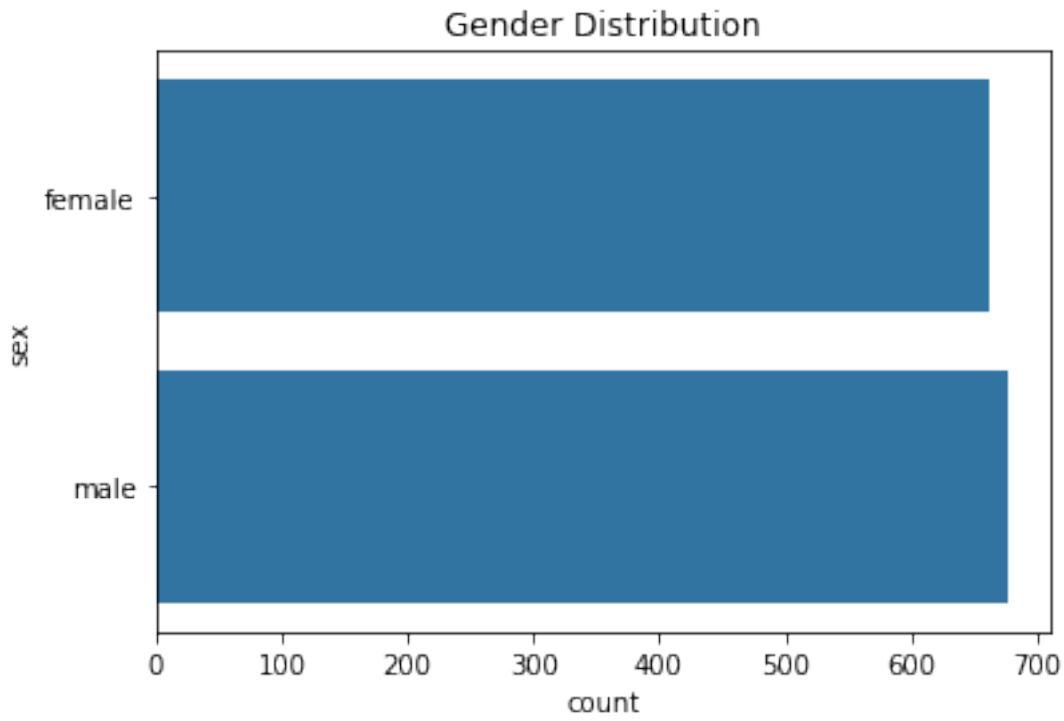
[10]:
```
plt.figure(figsize=(8,7))
sns.histplot(df['age'], kde=True)
```

[10]:
```
<Axes: xlabel='age', ylabel='Count'>
```



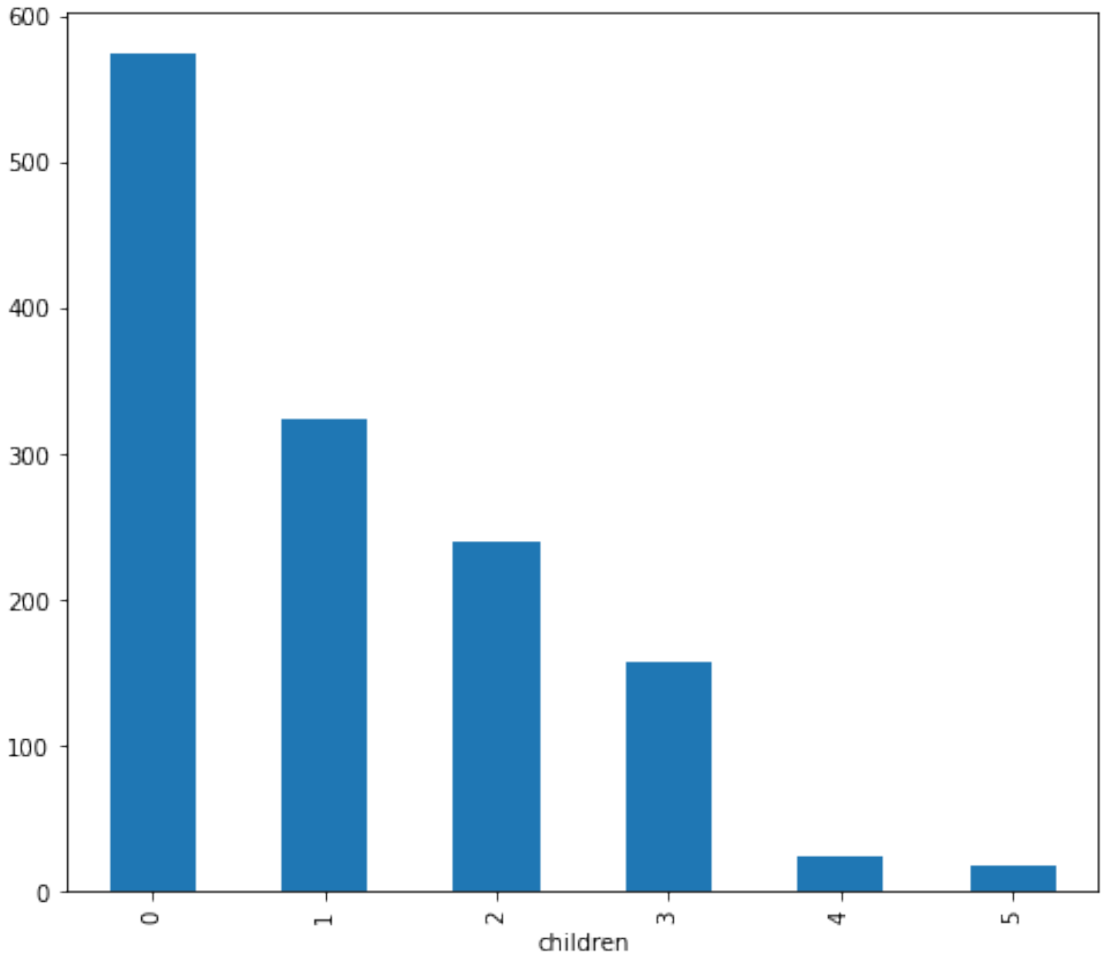
[11]:
```
sns.countplot(df['sex'])
plt.title('Gender Distribution')
```

[11]: Text(0.5, 1.0, 'Gender Distribution')



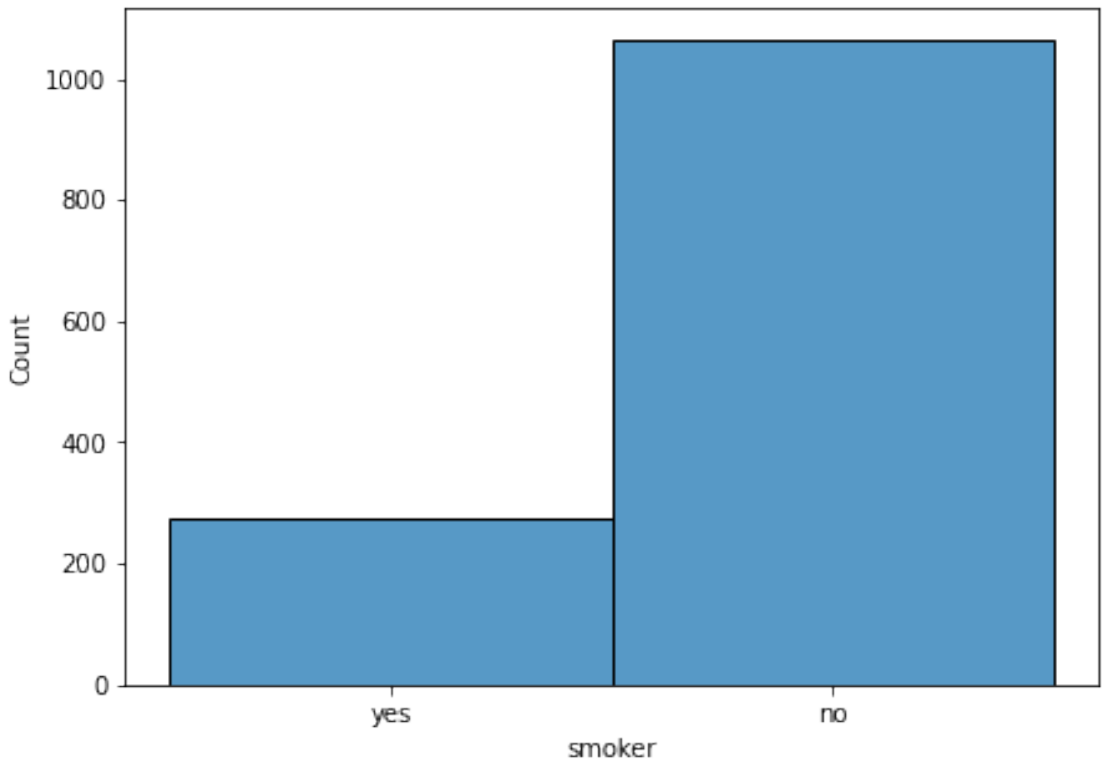
[12]:
```
df['children'].value_counts().plot(kind = 'bar', figsize = (8,7))
```

[12]: <Axes: xlabel='children'>



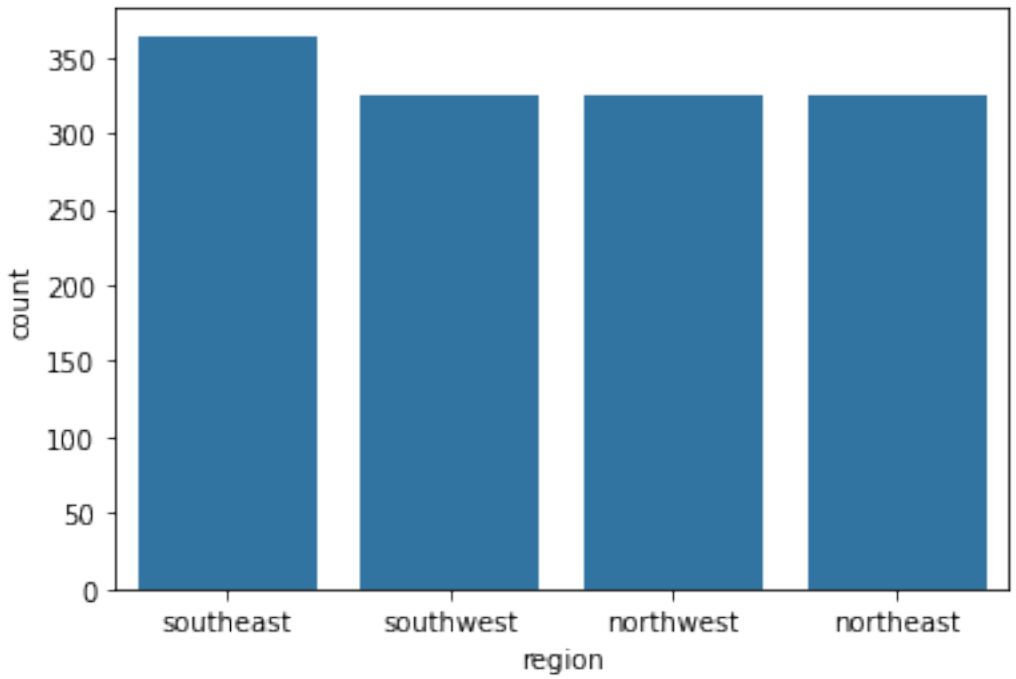
[13]:
```
plt.figure(figsize=(7,5))
sns.histplot(df['smoker'], bins=20)
```

[13]: <Axes: xlabel='smoker', ylabel='Count'>



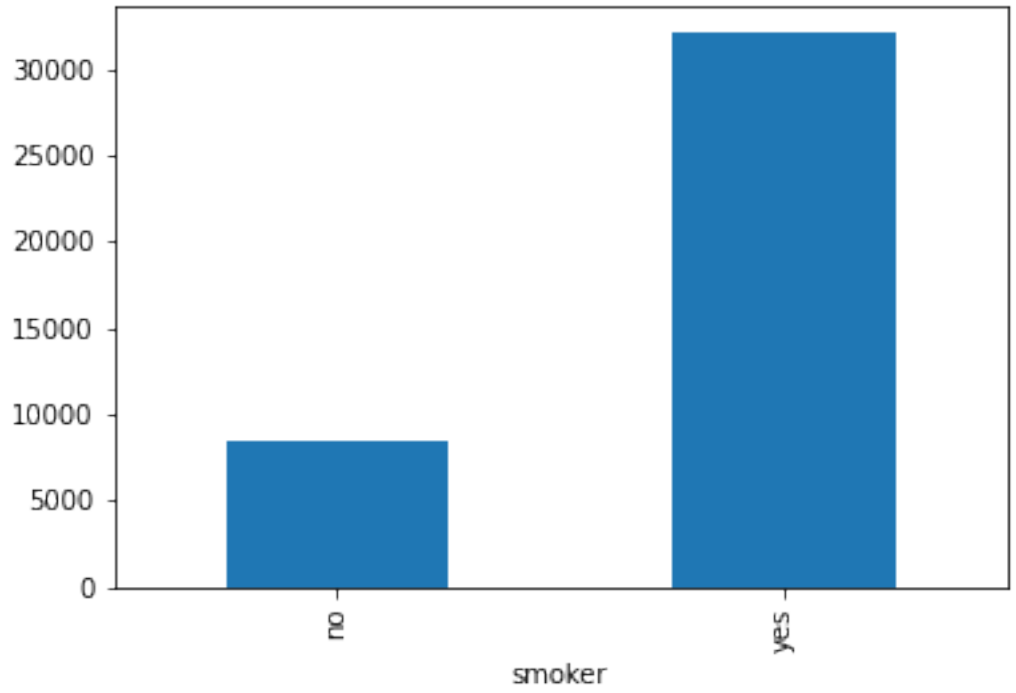
[14]:
```
sns.barplot(x = df['region'].value_counts().index, y = df['region'].
↪value_counts())
```

[14]:
```
<Axes: xlabel='region', ylabel='count'>
```



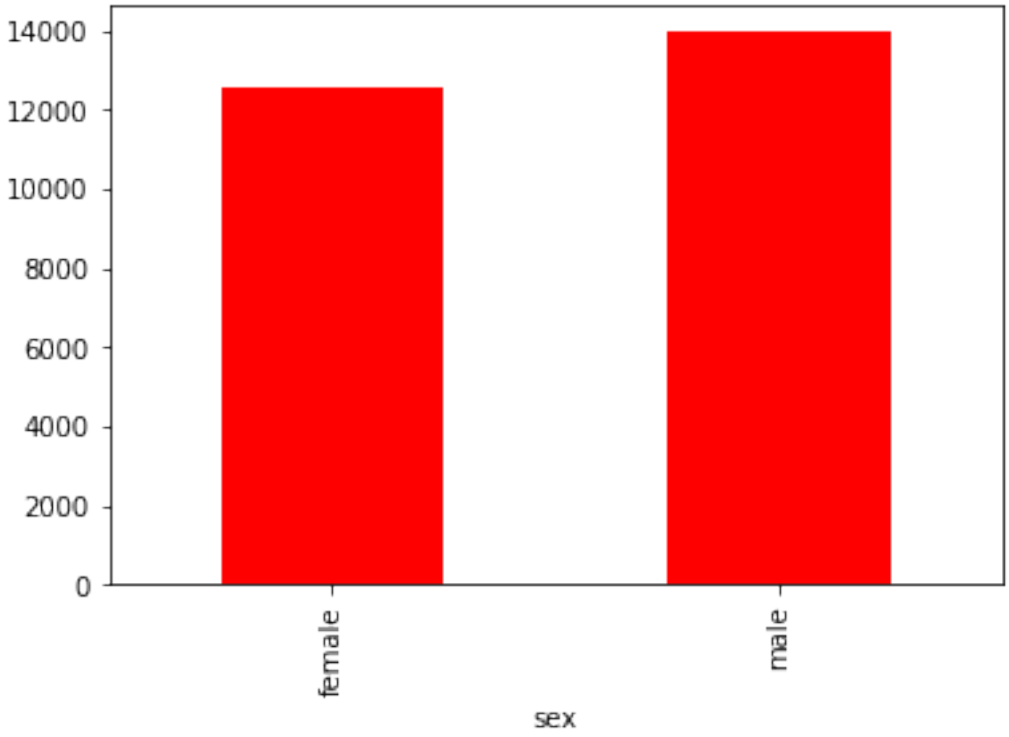
[15]:
```
df.groupby('smoker')['charges'].mean().plot(kind = 'bar')
```

[15]: <Axes: xlabel='smoker'>



[16]:
```
df.groupby('sex')['charges'].mean().plot(kind = 'bar',color = 'red')
```

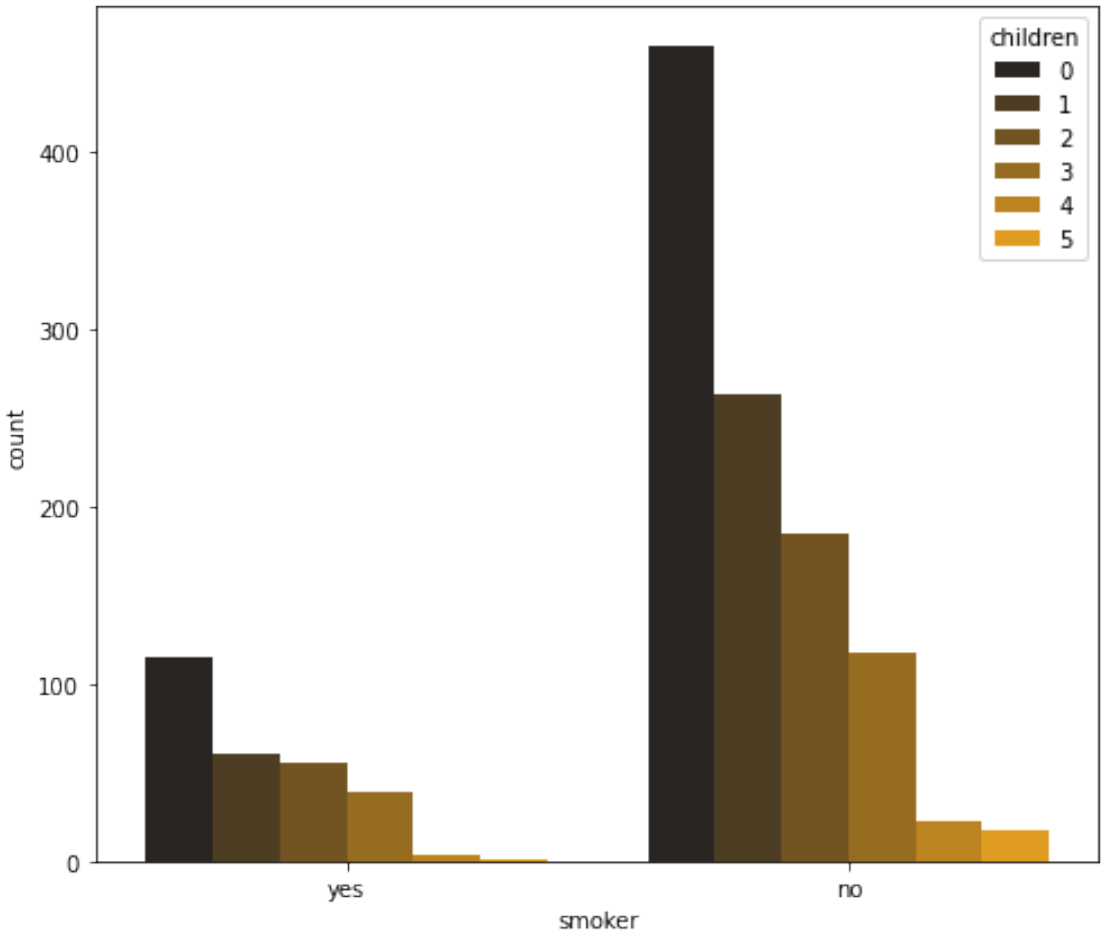
[16]: <Axes: xlabel='sex'>



[17]:
```
plt.figure(figsize = (8,7))
sns.countplot(x = df.smoker, hue = df.children, color = 'orange')
```

```
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/2618244184.py:2:
FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in
v0.14.0. Set `palette='dark:orange'` for the same effect.

  sns.countplot(x = df.smoker, hue = df.children, color = 'orange')
```

[17]: <Axes: xlabel='smoker', ylabel='count'>



[18]:
```
df.head()
```

[18]:

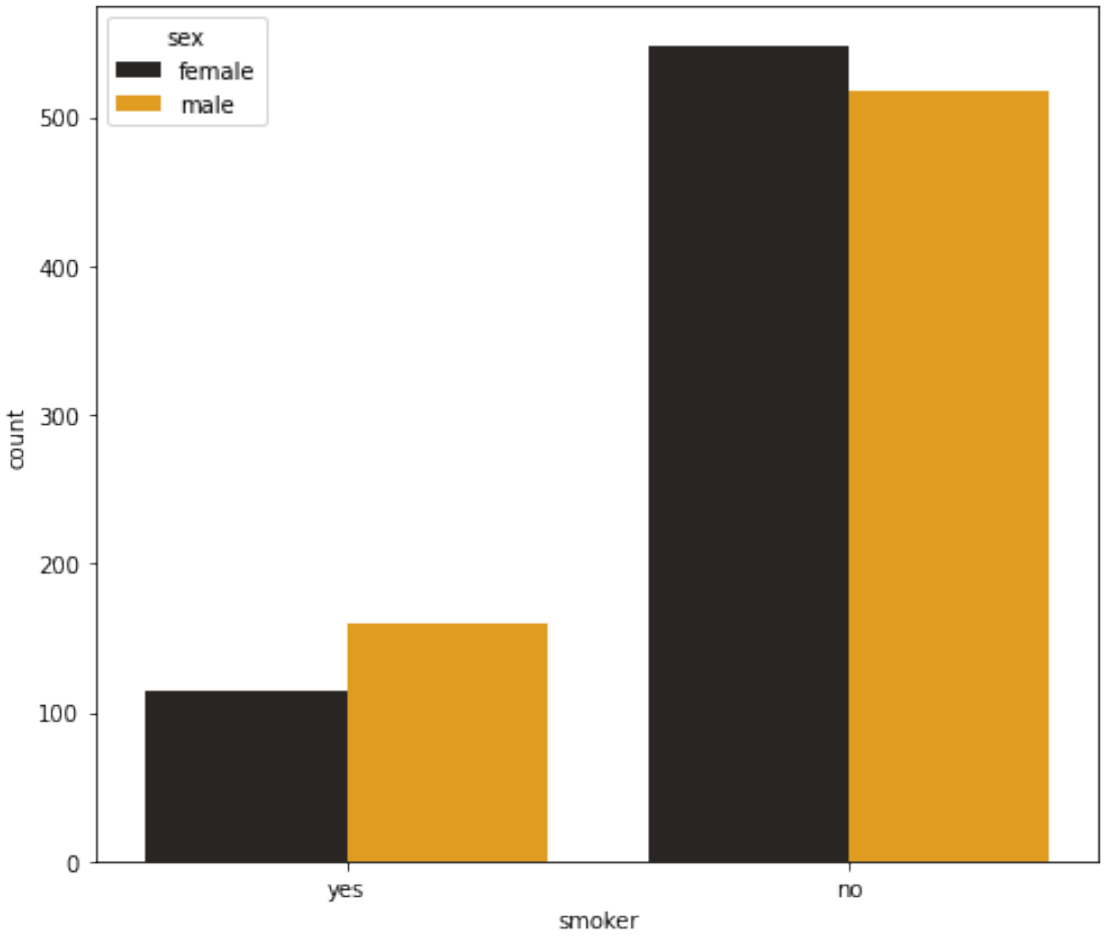
	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

[19]:
```
plt.figure(figsize = (8,7))
sns.countplot(x = df.smoker, hue = df.sex, color = 'orange')
```

```
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/1282408211.py:2:
FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in
v0.14.0. Set `palette='dark:orange'` for the same effect.
```

```
  sns.countplot(x = df.smoker, hue = df.sex, color = 'orange')
```

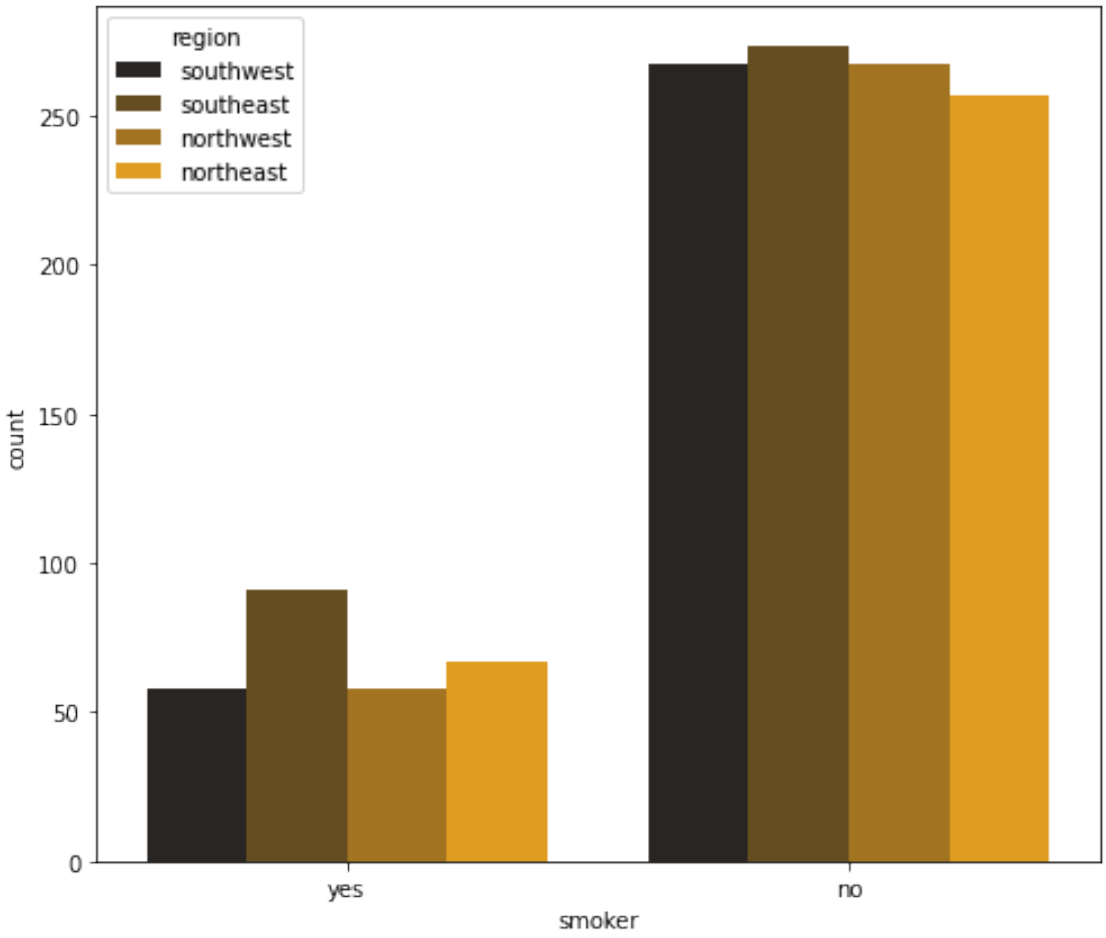
[19]: <Axes: xlabel='smoker', ylabel='count'>



[20]:
```
plt.figure(figsize = (8,7))
sns.countplot(x = df.smoker, hue = df.region, color = 'orange')
```

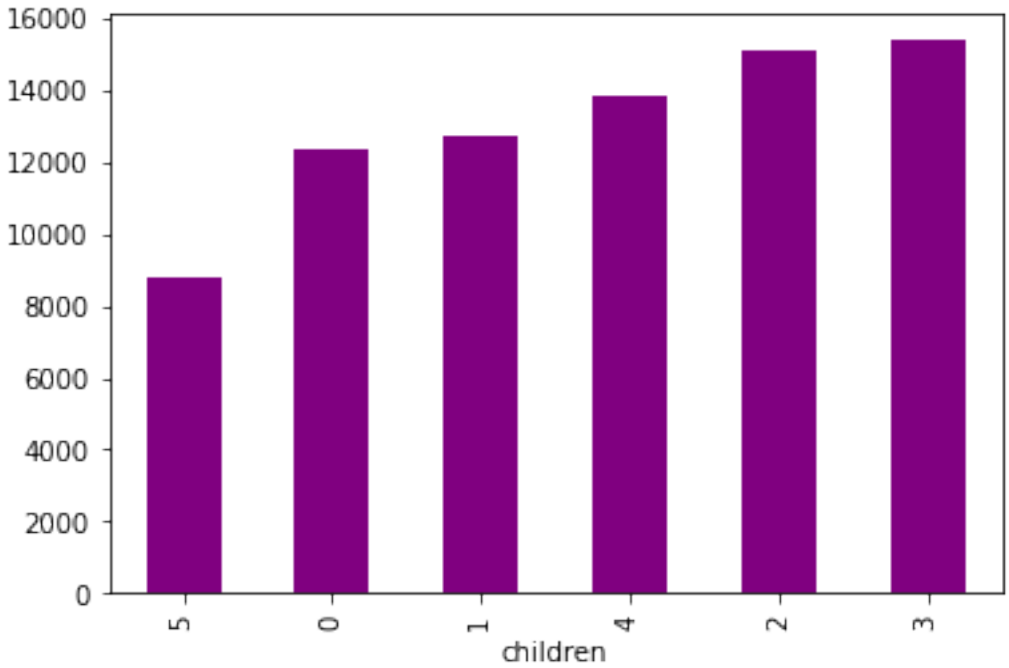
```
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/1161473306.py:2:
FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in
v0.14.0. Set `palette='dark:orange'` for the same effect.

  sns.countplot(x = df.smoker, hue = df.region, color = 'orange')
```

[20]: <Axes: xlabel='smoker', ylabel='count'>



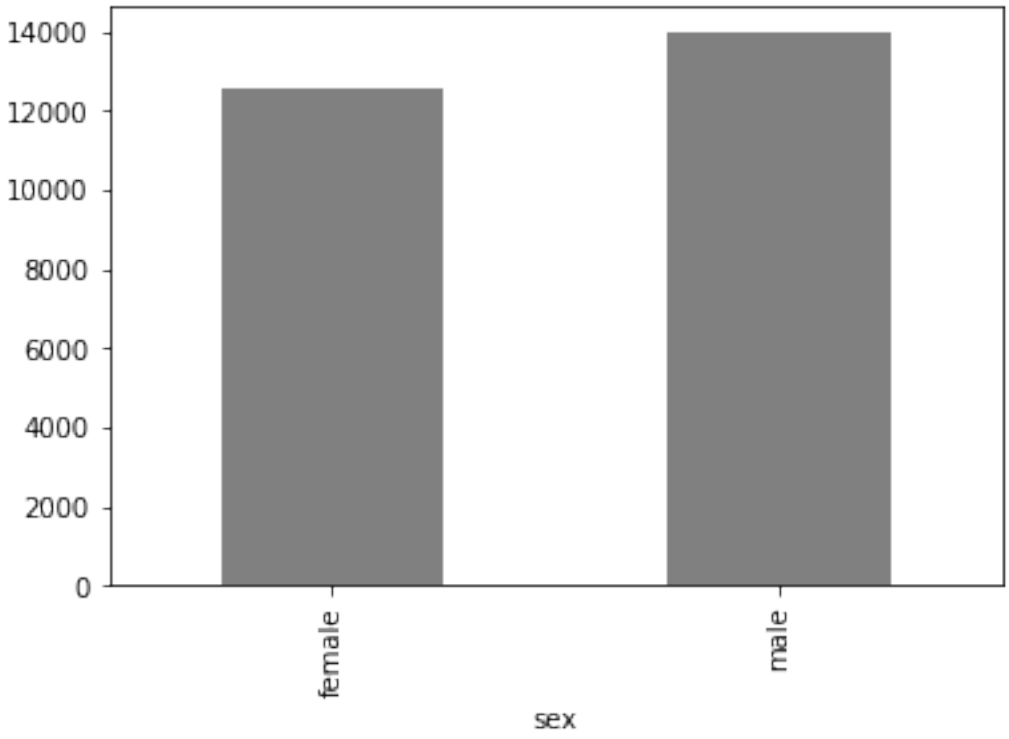
[21]:
```
df.groupby('children')['charges'].mean().sort_values(ascending =True).plot(kind
↪= 'bar', color='purple')
```

[21]: <Axes: xlabel='children'>



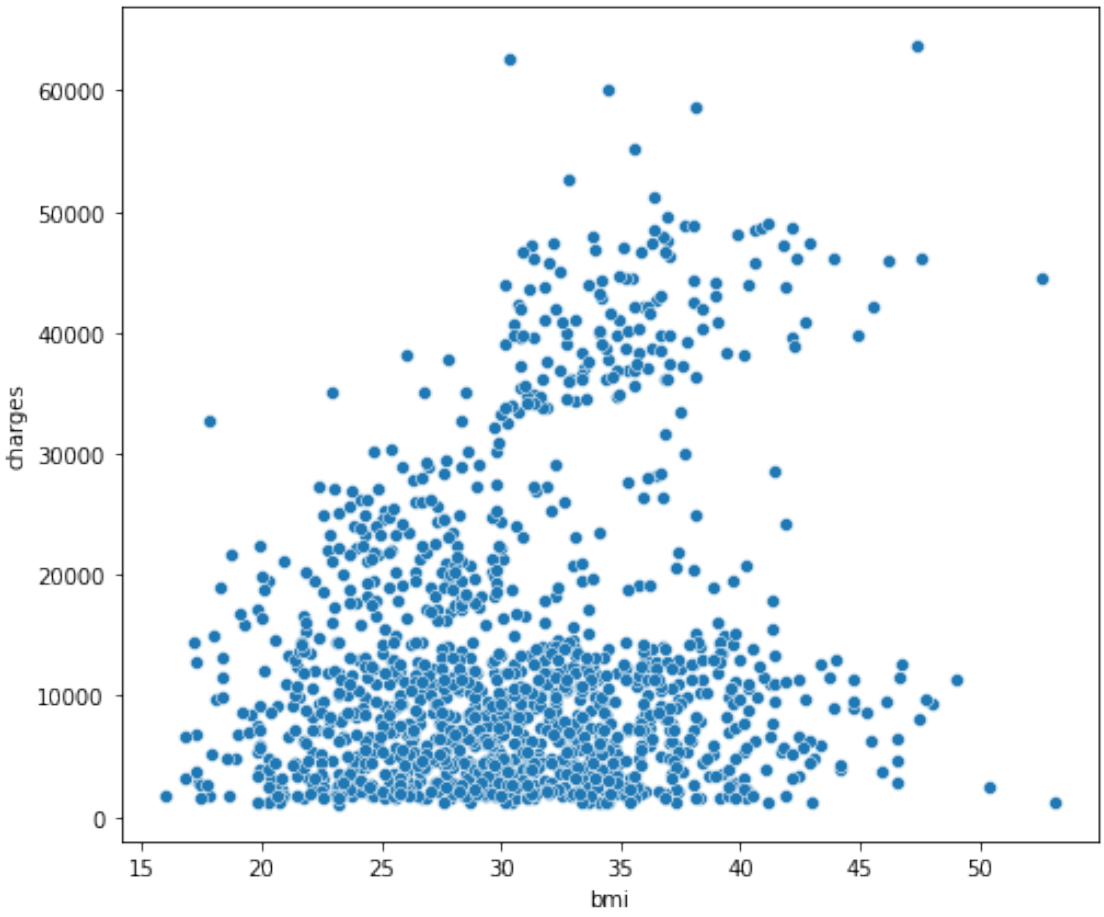
[22]:
```
df.groupby('sex')['charges'].mean().plot(kind = 'bar', color='grey')
```

[22]: <Axes: xlabel='sex'>



[23]:
```
plt.figure(figsize = (8,7))
sns.scatterplot(x = df['bmi'], y = df['charges'])
```

[23]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```



[24]:
```
# Feature Engineering
```

[25]:
```
df.head()
```

[25]:

	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

[26]:
```
# handling categorical columns
```

[27]:
```
df['sex'].unique()
```

[27]:
```
array(['female', 'male'], dtype=object)
```

[28]:
```
df['sex'] = df['sex'].map({'male':0, 'female':1})
```

[29]:
```
df['smoker'].unique()
```

[29]:
```
array(['yes', 'no'], dtype=object)
```

[30]:
```
df['smoker'] = df['smoker'].map({'yes':0, 'no':1})
```

[31]:
```
df['region'].unique()
```

[31]:
```
array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

[32]:
```
df = pd.get_dummies(columns =['region'], data =df, dtype = int)
```

[33]:
```
df.head()
```

[33]:

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

[34]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 10 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   charges           1338 non-null   float64
 6   region_northeast  1338 non-null   int32
 7   region_northwest  1338 non-null   int32
 8   region_southeast  1338 non-null   int32
 9   region_southwest  1338 non-null   int32
dtypes: float64(2), int32(4), int64(4)
memory usage: 83.8 KB
```

[35]:
```
# Correlation
```

[36]:
```
corr = df.corr()
corr
```

[36]:

	age	sex	bmi	children	smoker	charges
age	1.000000	0.020856	0.109272	0.042469	0.025019	0.299008
sex	0.020856	1.000000	-0.046371	-0.017163	0.076185	-0.057292
bmi	0.109272	-0.046371	1.000000	0.012759	-0.003750	0.198341
children	0.042469	-0.017163	0.012759	1.000000	-0.007673	0.067998
smoker	0.025019	0.076185	-0.003750	-0.007673	1.000000	-0.787251
charges	0.299008	-0.057292	0.198341	0.067998	-0.787251	1.000000
region_northeast	0.002475	0.002425	-0.138156	-0.022808	-0.002811	0.006349
region_northwest	-0.000407	0.011156	-0.135996	0.024806	0.036945	-0.039905
region_southeast	-0.011642	-0.017117	0.270025	-0.023066	-0.068498	0.073982
region_southwest	0.010016	0.004184	-0.006205	0.021914	0.036945	-0.043210

	region_northeast	region_northwest	region_southeast	region_southwest
age	0.002475	-0.000407	-0.011642	0.010016
sex	0.002425	0.011156	-0.017117	0.004184
bmi	-0.138156	-0.135996	0.270025	-0.006205
children	-0.022808	0.024806	-0.023066	0.021914
smoker	-0.002811	0.036945	-0.068498	0.036945
charges	0.006349	-0.039905	0.073982	-0.043210
region_northeast	1.000000	-0.320177	-0.345561	-0.320177
region_northwest	-0.320177	1.000000	-0.346265	-0.320829
region_southeast	-0.345561	-0.346265	1.000000	-0.346265
region_southwest	-0.320177	-0.320829	-0.346265	1.000000

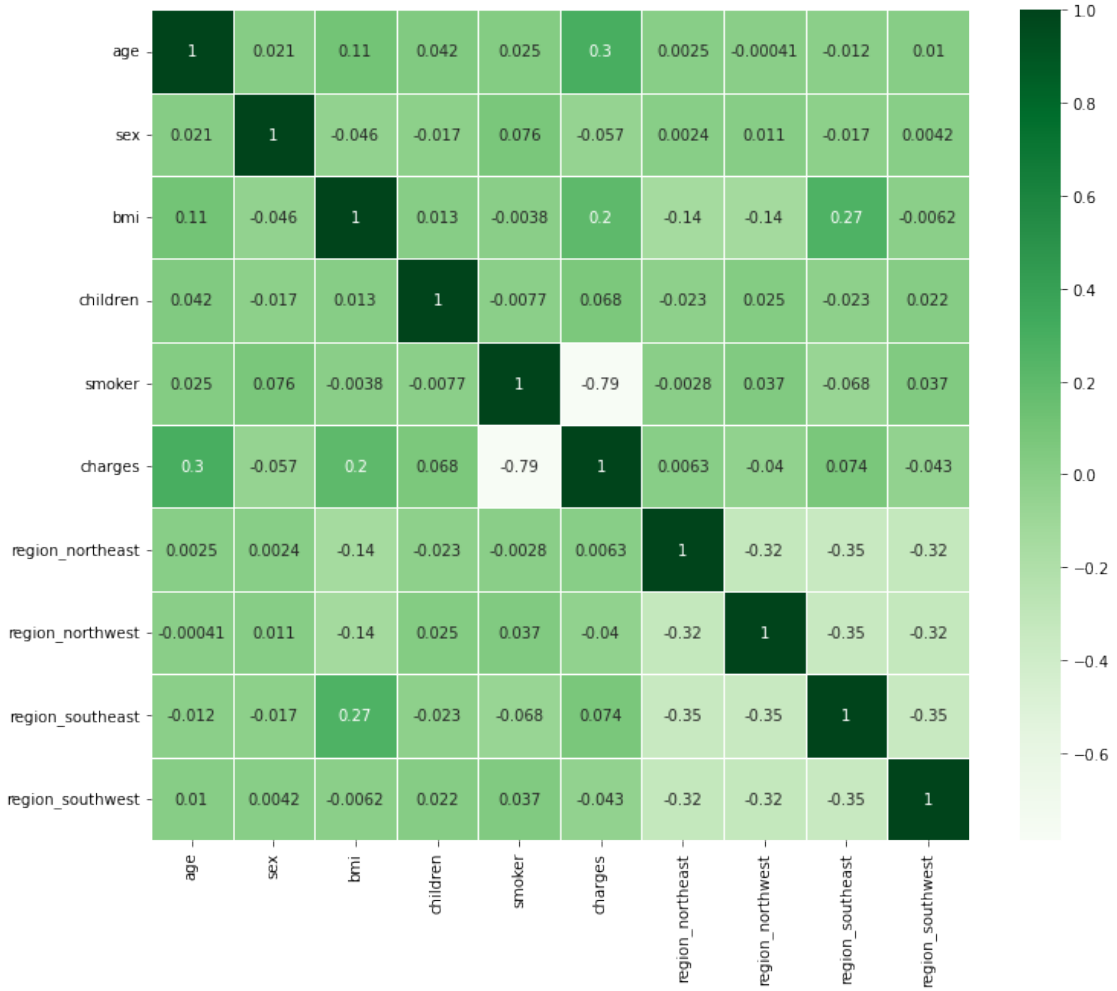
```
region_southeast          -0.346265
region_southwest           1.000000
```

[37]:
```
# matrix plot
```

[38]:
```
# Heatmap
```

[39]:
```
plt.figure(figsize = (12,10))
sns.heatmap(corr, annot =True, linewidth = 0.5, cmap='Greens')
```

[39]:
```
<Axes: >
```



[40]:
```
# train-test-split
```

[41]:
```
from sklearn.model_selection import train_test_split
```

[42]:
```
x = df.loc[:, df.columns != 'charges']
```

[43]:
```
y = df[['charges']]
```

[44]:
```
x.head()
```

[44]:

	age	sex	bmi	children	smoker	region_northeast	region_northwest	region_southeast	region_southwest
0	19	1	27.900	0	0	0	0	0	1
1	18	0	33.770	1	1	0	0	1	0
2	28	0	33.000	3	1	0	0	1	0
3	33	0	22.705	0	1	0	1	0	0
4	32	0	28.880	0	1	0	1	0	0

[45]:
```
y.head()
```

[45]:

	charges
0	16884.92400
1	1725.55230
2	4449.46200
3	21984.47061
4	3866.85520

[46]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
↪random_state =42)
```

[47]:
```
print(x_train.shape, y_train.shape)
```

```
(936, 9) (936, 1)
```

[48]:
```
print(x_test.shape, y_test.shape)
```

```
(402, 9) (402, 1)
```

[49]:
```
# Before Using linear Regression we should standardize our data
# Standardization
```

[50]:
```
from sklearn.preprocessing import StandardScaler
```

[51]:
```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
```

```
x_test_scaled = scaler.transform(x_test)
```

[52]:
```
# Linear Regression
```

[53]:
```
from sklearn.linear_model import LinearRegression
reg  = LinearRegression()
reg.fit(x_train_scaled, y_train)
y_pred = reg.predict(x_test_scaled)
```

[54]:
```
# Model Evaluation
```

[55]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

[56]:
```
mse = mean_absolute_error(y_test, y_pred)
mae =  mean_squared_error(y_test, y_pred)
print(mse, mae)
```

```
4145.450555627599 33780509.574791655
```

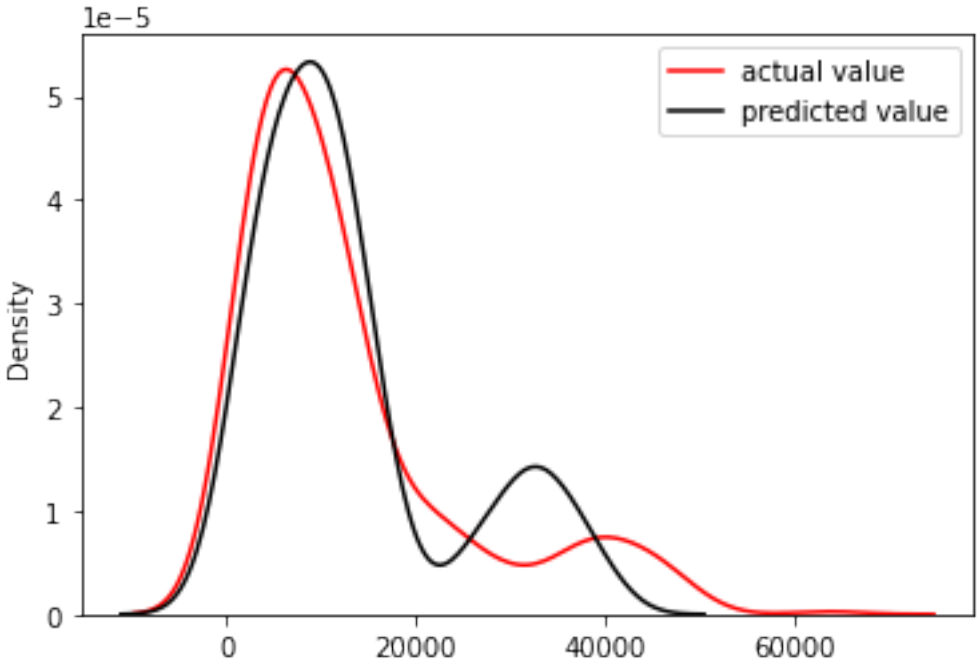
[57]:
```
r2_score(y_test, y_pred)
```

[57]:
```
0.769611805436901
```

[58]:
```
sns.distplot(y_test, label = 'actual value', color = 'red', hist = False)
sns.distplot(y_pred, label = 'predicted value', color = 'black', hist = False)
plt.legend()
plt.show()
```

```
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/1152848395.py:1: 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 `kdeplot` (an axes-level function for kernel density
plots).

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

  sns.distplot(y_test, label = 'actual value', color = 'red', hist = False)
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/1152848395.py:2: 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 `kdeplot` (an axes-level function for kernel density
plots).
```

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

  sns.distplot(y_pred, label = 'predicted value', color = 'black', hist = False)
```



[59]:
```
# Random Forest Regressor
```

[60]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()

param_grid = {
    'n_estimators': [100,200,300,400],
    'max_depth': [None, 5,10,15,17],
    'min_samples_split': [2, 5,8,10,15],
    'min_samples_leaf': [1,3, 5,8,10]

}

grid_search = GridSearchCV(rf, param_grid, cv=5, verbose=1, n_jobs = -1,␣
 ↪scoring='neg_mean_squared_error')

grid_search.fit(x_train, y_train)
```

```
grid_search.best_params_
```

```
Fitting 5 folds for each of 500 candidates, totalling 2500 fits
```

```
C:\Users\amits\anaconda3\lib\site-packages\sklearn\base.py:1152:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  return fit_method(estimator, *args, **kwargs)
```

[60]:
```
{'max_depth': 5,
 'min_samples_leaf': 10,
 'min_samples_split': 5,
 'n_estimators': 100}
```

[61]:
```
grid_search.best_estimator_
```

[61]:
```
RandomForestRegressor(max_depth=5, min_samples_leaf=10, min_samples_split=5)
```

[73]:
```
rfr = RandomForestRegressor(n_estimators= 100, max_depth=5,␣
 ↪min_samples_leaf=10, min_samples_split=5)
rfr.fit(x_train, y_train)
y_pred = rfr.predict(x_test)
```

```
C:\Users\amits\anaconda3\lib\site-packages\sklearn\base.py:1152:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  return fit_method(estimator, *args, **kwargs)
```

[74]:
```
mse = mean_absolute_error(y_test, y_pred)
mae =  mean_squared_error(y_test, y_pred)
print(mse, mae)
```

```
2504.0905027725375 18665437.71292647
```

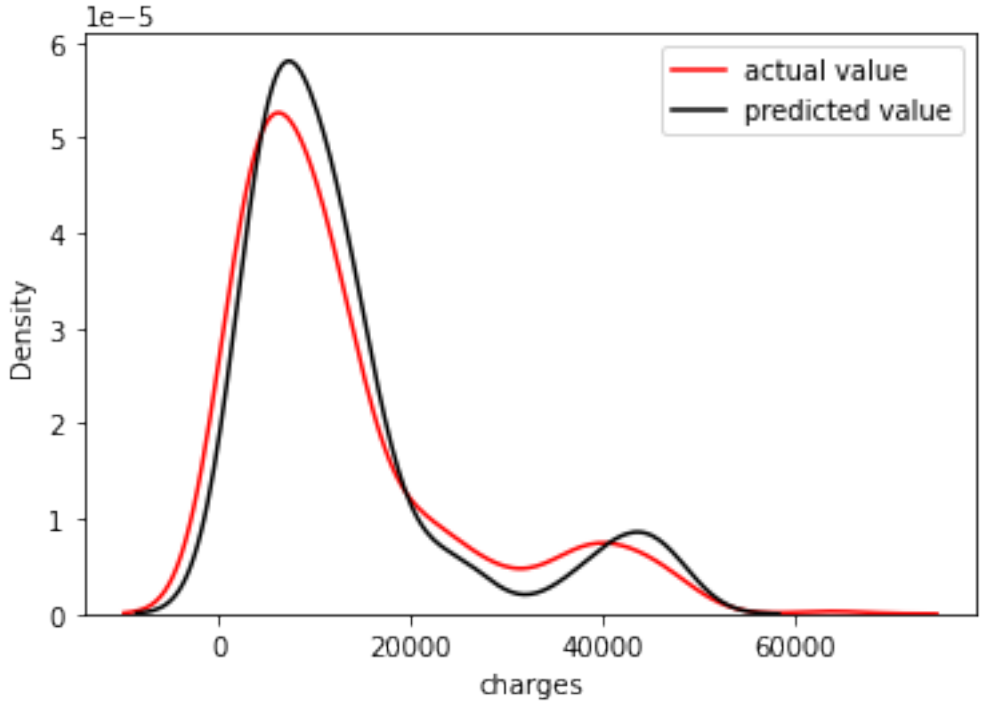
[75]:
```
r2_score(y_test, y_pred)
```

[75]:
```
0.8726988861464613
```

[76]:
```
sns.distplot(y_test, label = 'actual value', color = 'red', hist = False)
sns.distplot(y_pred, label = 'predicted value', color = 'black', hist = False)
plt.legend()
plt.xlabel('charges')
plt.show()
```

```
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/1522082191.py:1: 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 `kdeplot` (an axes-level function for kernel density
plots).

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

  sns.distplot(y_test, label = 'actual value', color = 'red', hist = False)
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/1522082191.py:2: 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 `kdeplot` (an axes-level function for kernel density
plots).

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

  sns.distplot(y_pred, label = 'predicted value', color = 'black', hist = False)
```



[77]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
rf = DecisionTreeRegressor()

param_grid = {
    'max_depth': [None, 5,10,15,17],
    'min_samples_split': [2, 5,8,10,15],
    'min_samples_leaf': [1,3, 5,8,10]

}

grid_search = GridSearchCV(rf, param_grid, cv=5, verbose=1, n_jobs = -1,␣
 ↪scoring='neg_mean_squared_error')

grid_search.fit(x_train, y_train)

grid_search.best_params_
```

```
Fitting 5 folds for each of 125 candidates, totalling 625 fits
```

[77]:
```
{'max_depth': 5, 'min_samples_leaf': 10, 'min_samples_split': 8}
```

[78]:
```
grid_search.best_estimator_
```

[78]:
```
DecisionTreeRegressor(max_depth=5, min_samples_leaf=10, min_samples_split=8)
```

[79]:
```
dtree = DecisionTreeRegressor(max_depth=5, min_samples_leaf=10,␣
 ↪min_samples_split=8)
dtree.fit(x_train, y_train)
y_pred = rfr.predict(x_test)
```

[80]:
```
r2_score(y_test, y_pred)
```

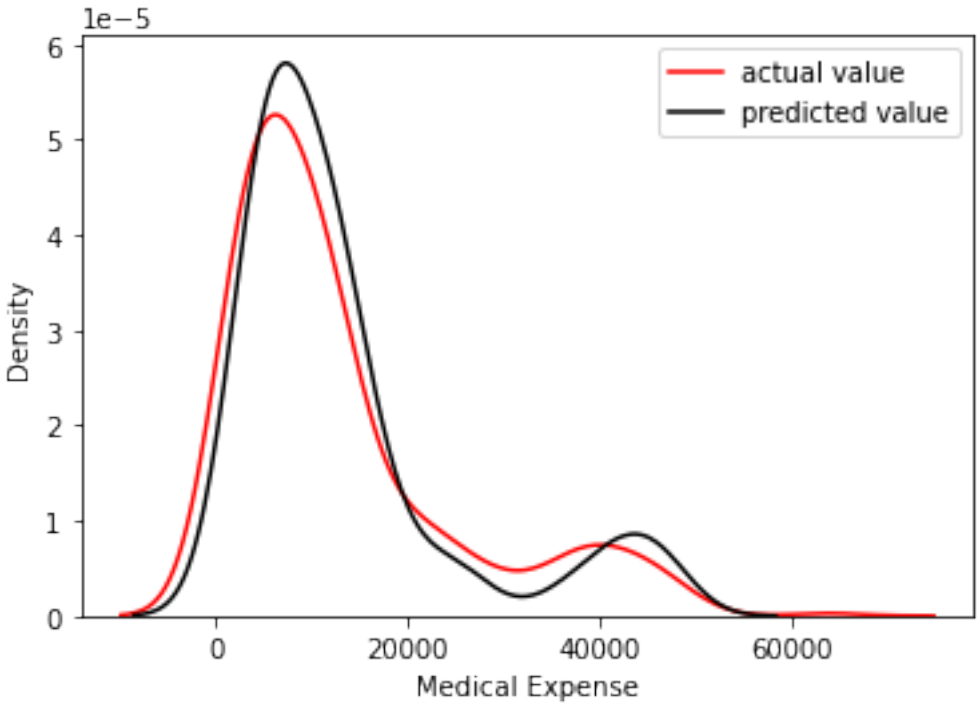
[80]:
```
0.8726988861464613
```

[81]:
```
mse = mean_absolute_error(y_test, y_pred)
mae =  mean_squared_error(y_test, y_pred)
print(mse, mae)
```

```
2504.0905027725375 18665437.71292647
```

[82]:
```
sns.distplot(y_test, label = 'actual value', color = 'red', hist = False)
sns.distplot(y_pred, label = 'predicted value', color = 'black', hist = False)
plt.legend()
plt.xlabel('Medical Expense')
plt.show()
```

```
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/3821897850.py:1: 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 `kdeplot` (an axes-level function for kernel density
plots).

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

  sns.distplot(y_test, label = 'actual value', color = 'red', hist = False)
C:\Users\amits\AppData\Local\Temp/ipykernel_21004/3821897850.py:2: 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 `kdeplot` (an axes-level function for kernel density
plots).

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

  sns.distplot(y_pred, label = 'predicted value', color = 'black', hist = False)
```



Conclusion From the above models, we can see that Decision Tree Regressor and Random Forest Regressor are giving the best results. Therefore, I can use Random Forest Regressor or decision tree regressor to predict the medical expense of patients.

Moreover, the medical expense of smokers is higher than that of non-smokers. The medical expense of older patients is higher than that of younger patients.

Thus, from the overall analysis, we can conclude that the medical expense of patients depends on their age, BMI, smoking habits.

[83]:
```
import pickle

pickle.dump(rfr, open('medical_cost_.pkl', 'wb'))
```