In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split as holdout
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import cross_val_score
from sklearn import datasets
from sklearn import metrics
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df = pd.read_csv('medical_cost_insurance.csv')
```

In [3]:
```
df
```

Out[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
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

In [4]:
```
x = df.drop('charges', axis=1)
y = df['charges']
```

In [5]:
```
model = RandomForestRegressor(n_estimators=100, random_state=42)
```

In [6]:
```
df.head()
```

Out[6]:

	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

In [7]:
```
print('Number of rows:', df.shape[0]," ",'Number of columns:', df.shape[1])
```

```
Number of rows: 1338   Number of columns: 7
```

In [8]:
```
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
```

In [9]:
```
df.describe()
```

Out[9]:

	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

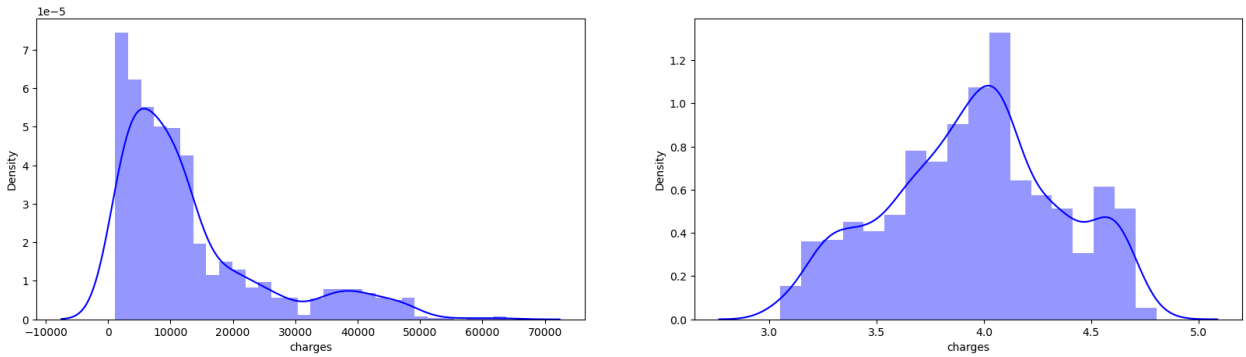
In [10]:
```
df.describe(include=['object'])
```

Out[10]:

	sex	smoker	region
count	1338	1338	1338
unique	2	2	4
top	male	no	southeast
freq	676	1064	364

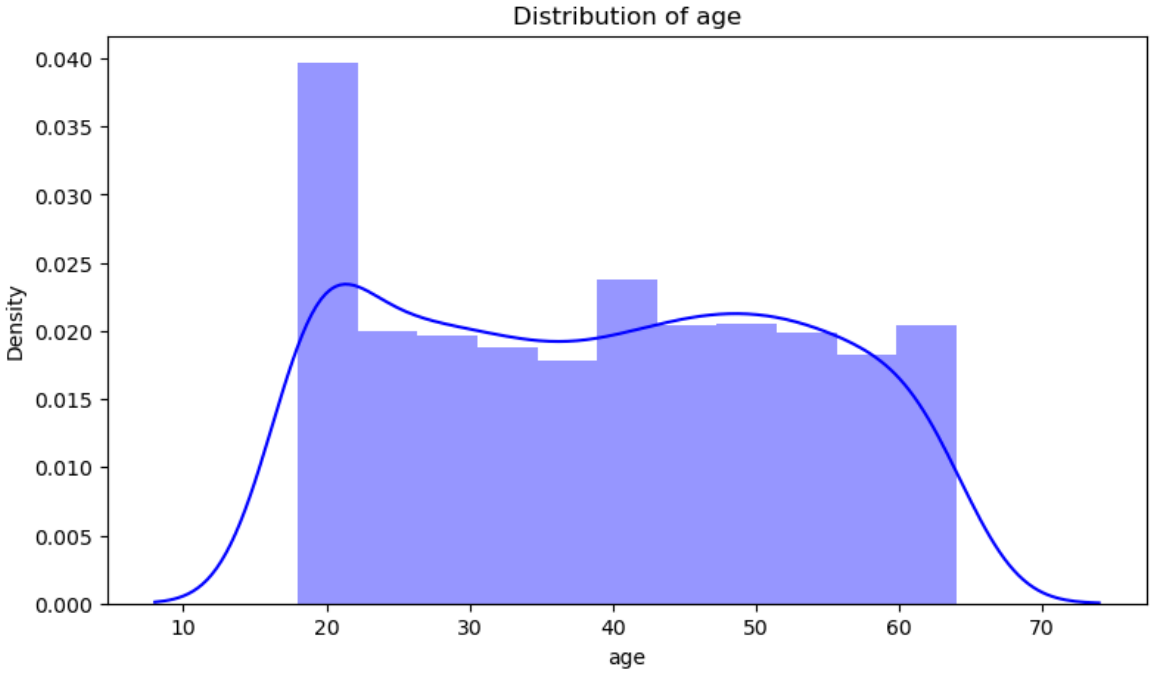
In [11]:
```
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
sns.distplot(df.charges, color = 'b')
plt.subplot(1,2,2)
sns.distplot(np.log10(df.charges), color = 'b')
```

Out[11]:
```
<Axes: xlabel='charges', ylabel='Density'>
```



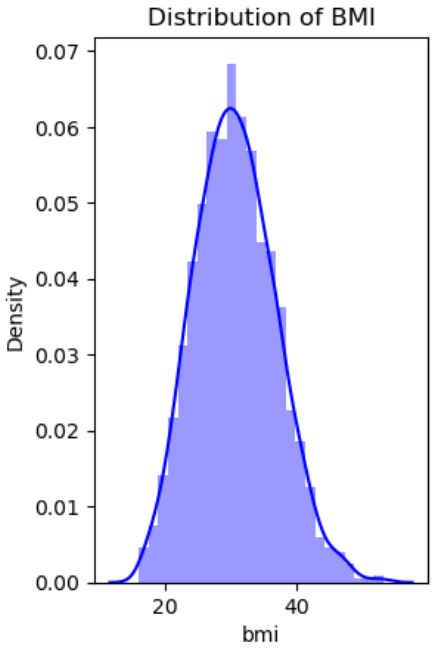
In [16]:
```
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
sns.distplot(df.age, color = 'b').set_title('Distribution of age')
```

Out[16]: Text(0.5, 1.0, 'Distribution of age')

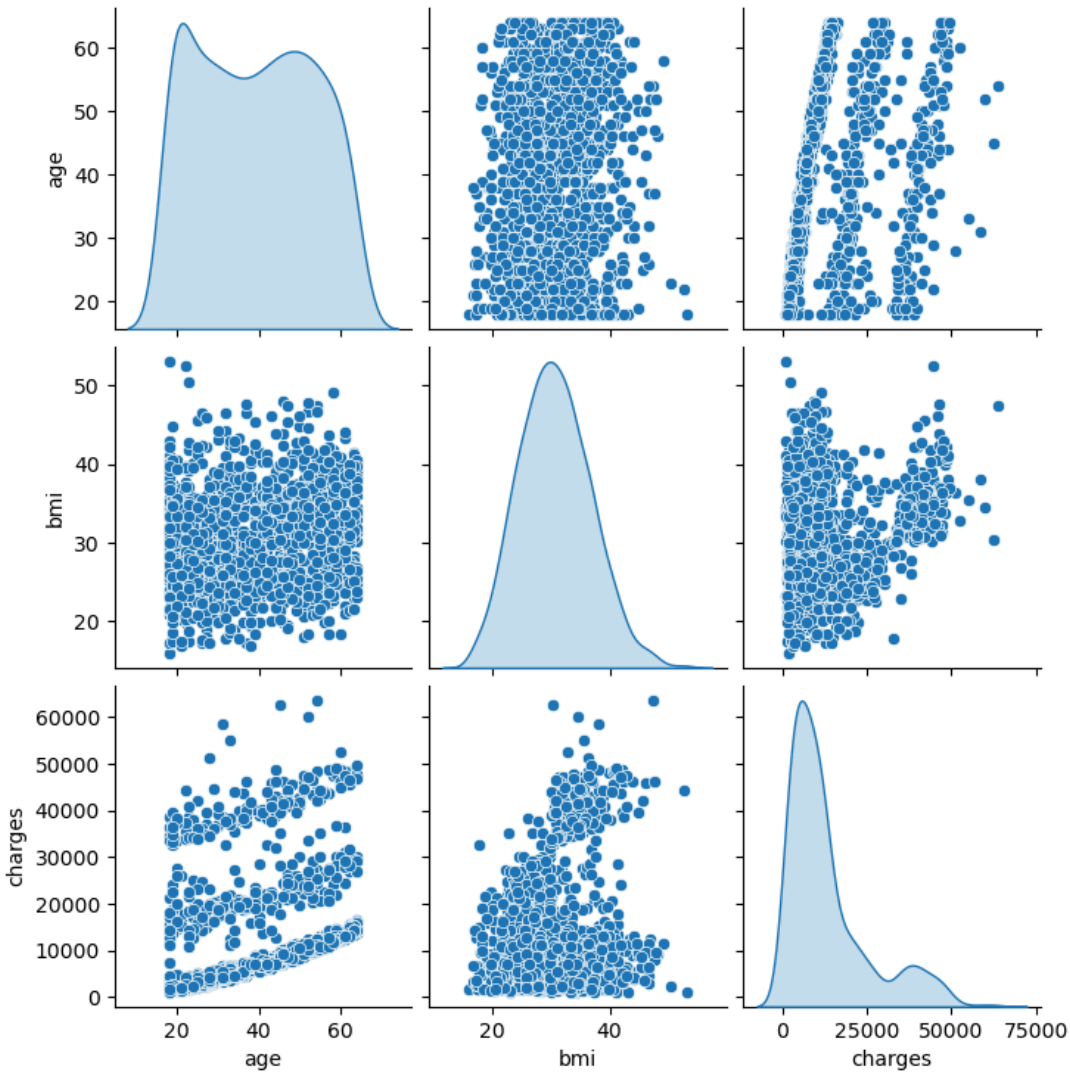


In [17]:
```
plt.subplot(1,2,2)
sns.distplot(df.bmi, color = 'b').set_title('Distribution of BMI')
```

Out[17]: Text(0.5, 1.0, 'Distribution of BMI')

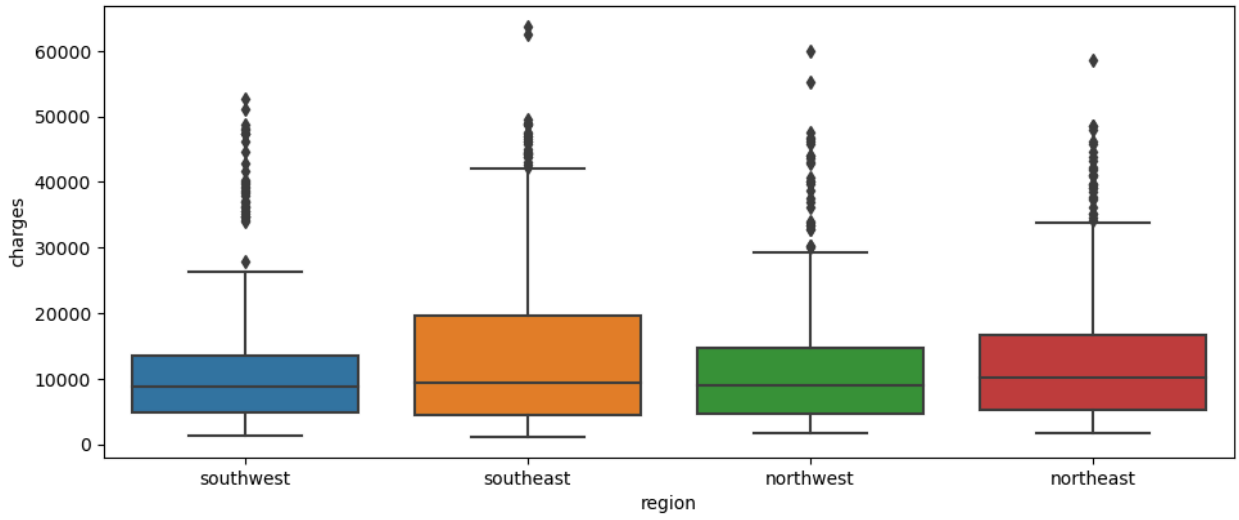


In [18]:
```
df_num = df[['age','bmi','charges']]
sns.pairplot(df_num, diag_kind = 'kde')
plt.show()
```



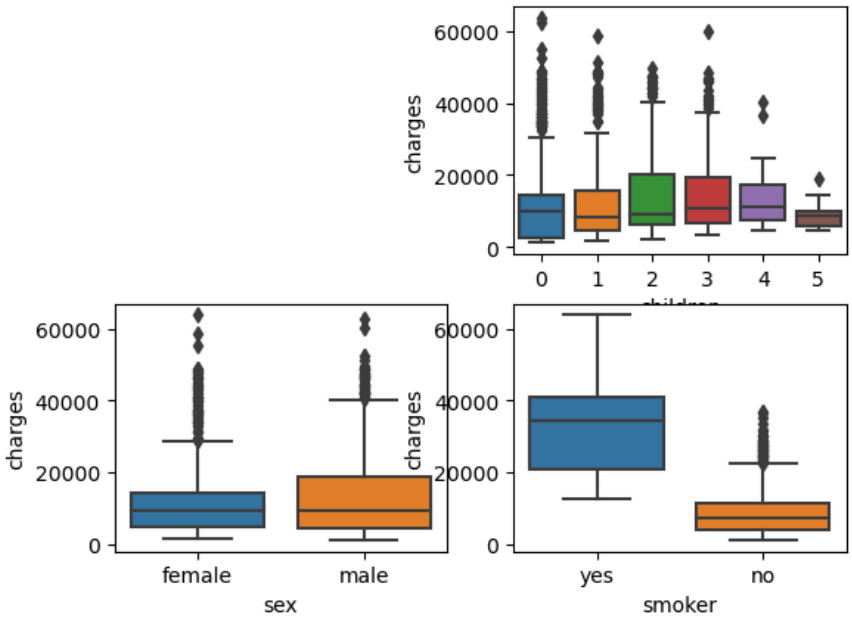
In [19]:
```
plt.figure (figsize = (25,10))
plt.subplot(2,2,1)
sns.boxplot(x = 'region', y = 'charges', data = df)
```

Out[19]: <Axes: xlabel='region', ylabel='charges'>

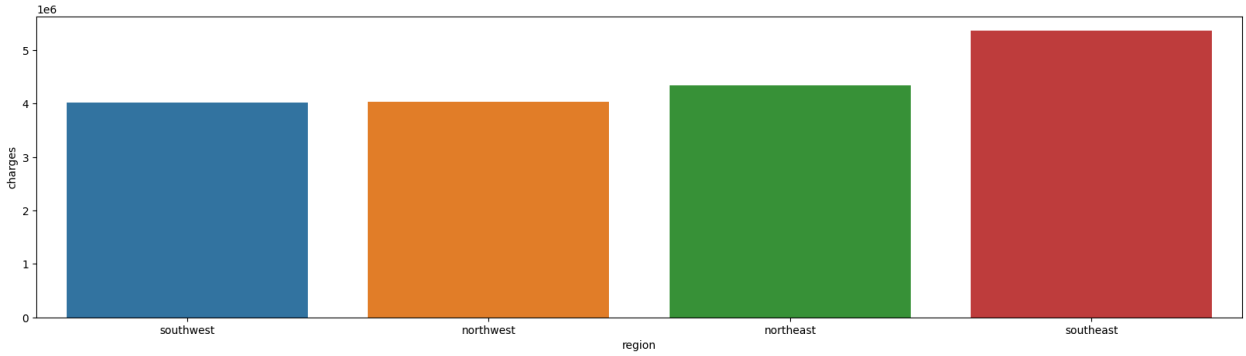


In [20]:
```
plt.subplot(2,2,2)
sns.boxplot(x = 'children', y = 'charges', data = df)
plt.subplot(2,2,3)
sns.boxplot(x = 'sex', y = 'charges', data = df)
plt.subplot(2,2,4)
sns.boxplot(x = 'smoker', y = 'charges', data = df)
```

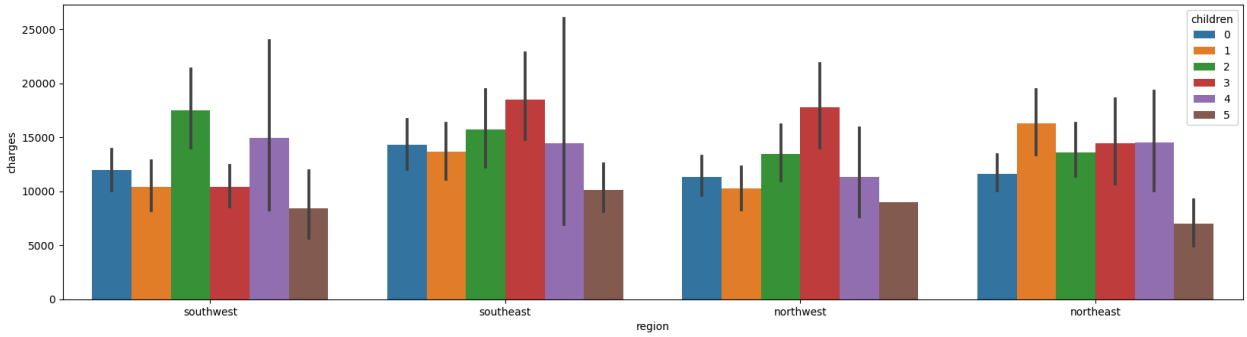
Out[20]: <Axes: xlabel='smoker', ylabel='charges'>



In [21]:
```
charges = df['charges'].groupby(df.region).sum().sort_values(ascending = True)
plt.figure(figsize=(20,5))
ax = sns.barplot(x = charges.index, y = charges)
```

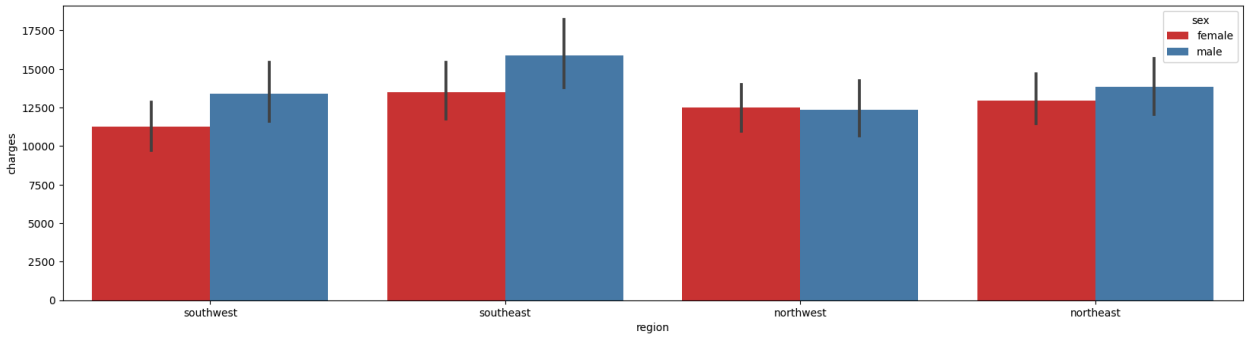


In [22]:
```
plt.figure(figsize=(20,5))
ax = sns.barplot(x = 'region', y = 'charges', hue = 'children', data = df)
```

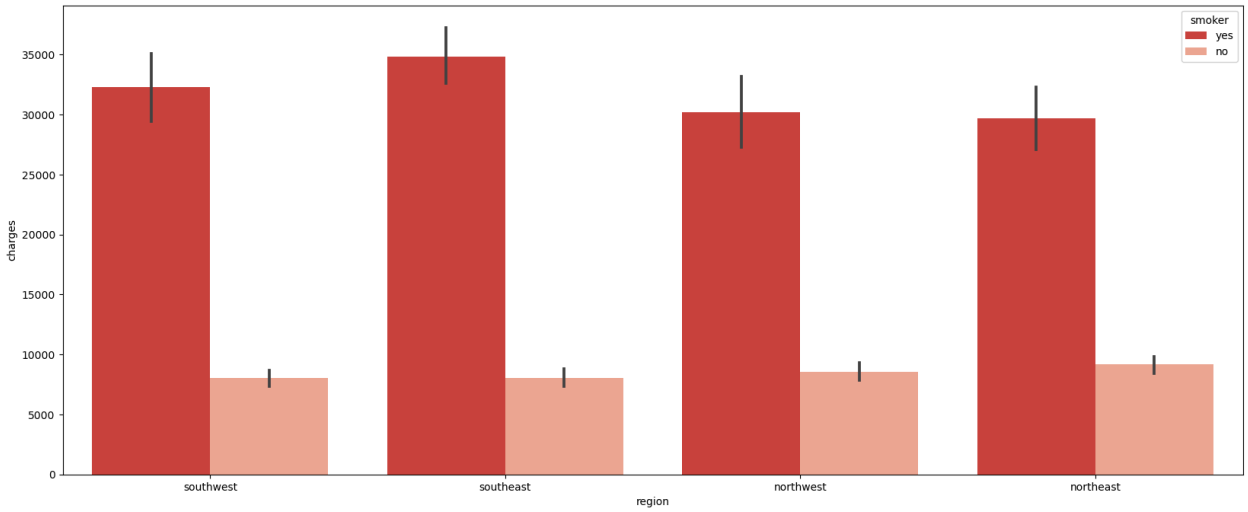


In [23]:
```
plt.figure(figsize=(20,5))
sns.barplot(x = 'region', y = 'charges', hue = 'sex', data = df, palette = 'Set1')
```

Out[23]: <Axes: xlabel='region', ylabel='charges'>

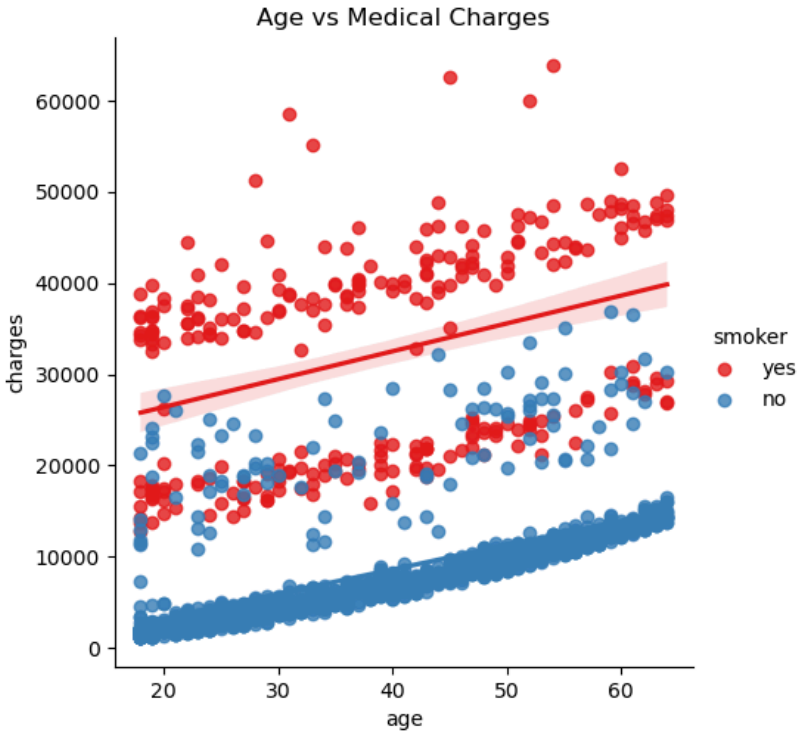


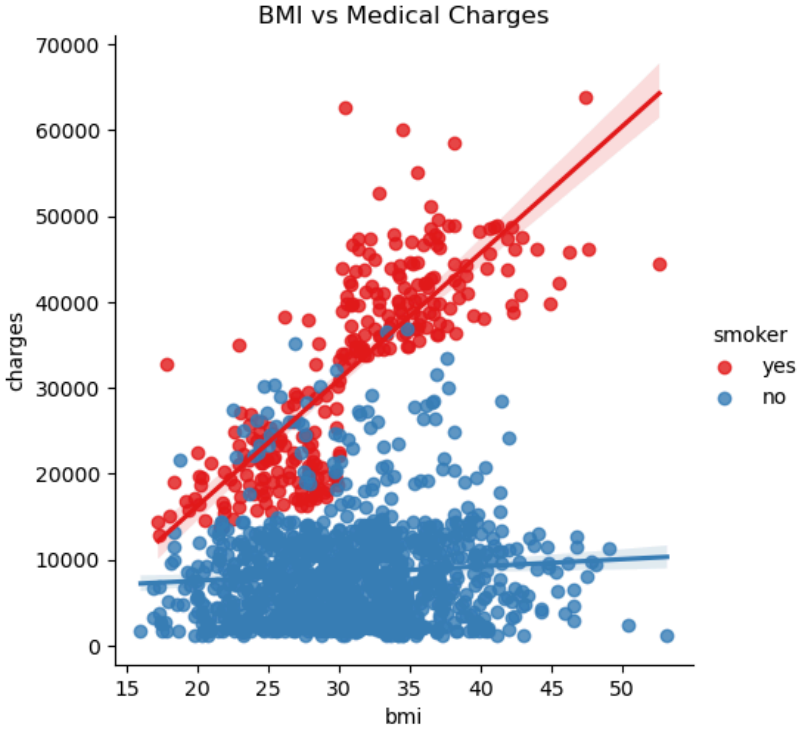
In [24]:
```
f, ax = plt.subplots(1, 1, figsize = (20, 8))
ax = sns.barplot(x = 'region', y = 'charges', hue = 'smoker', data = df, palette = 'Reds_r')
```

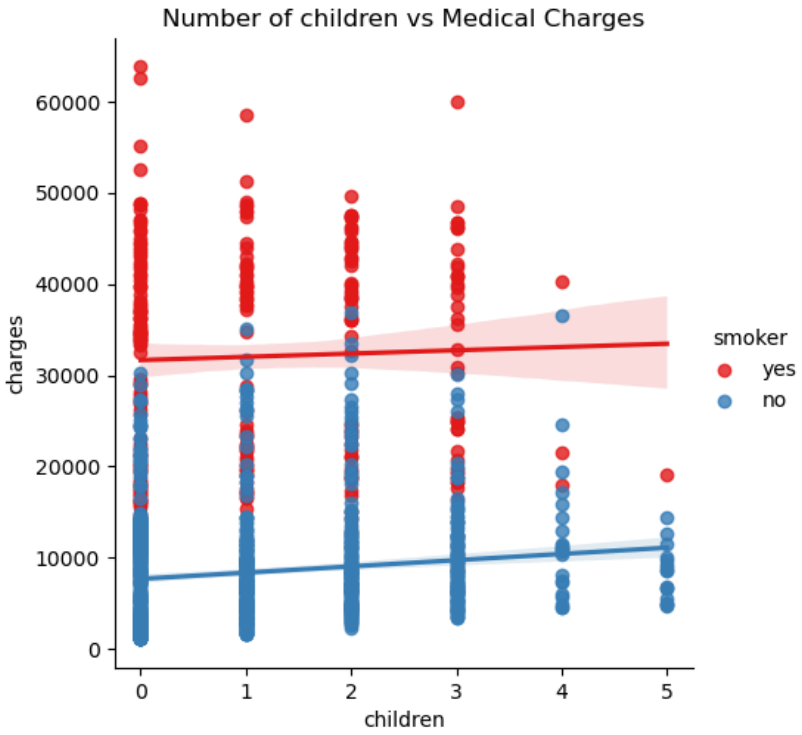


In [25]:
```
ax = sns.lmplot(x = 'age', y = 'charges', data = df, hue = 'smoker', palette = 'Set1')
plt.title('Age vs Medical Charges')
ax = sns.lmplot(x = 'bmi', y = 'charges', data = df, hue = 'smoker', palette = 'Set1')
plt.title('BMI vs Medical Charges')
ax = sns.lmplot(x = 'children', y = 'charges', data = df, hue = 'smoker', palette = 'Set1')
plt.title('Number of children vs Medical Charges')
```

Out[25]:
```
Text(0.5, 1.0, 'Number of children vs Medical Charges')
```







In [26]:
```
df[['region','sex','smoker']] = df[['region','sex','smoker']].astype('category')
df.dtypes
```

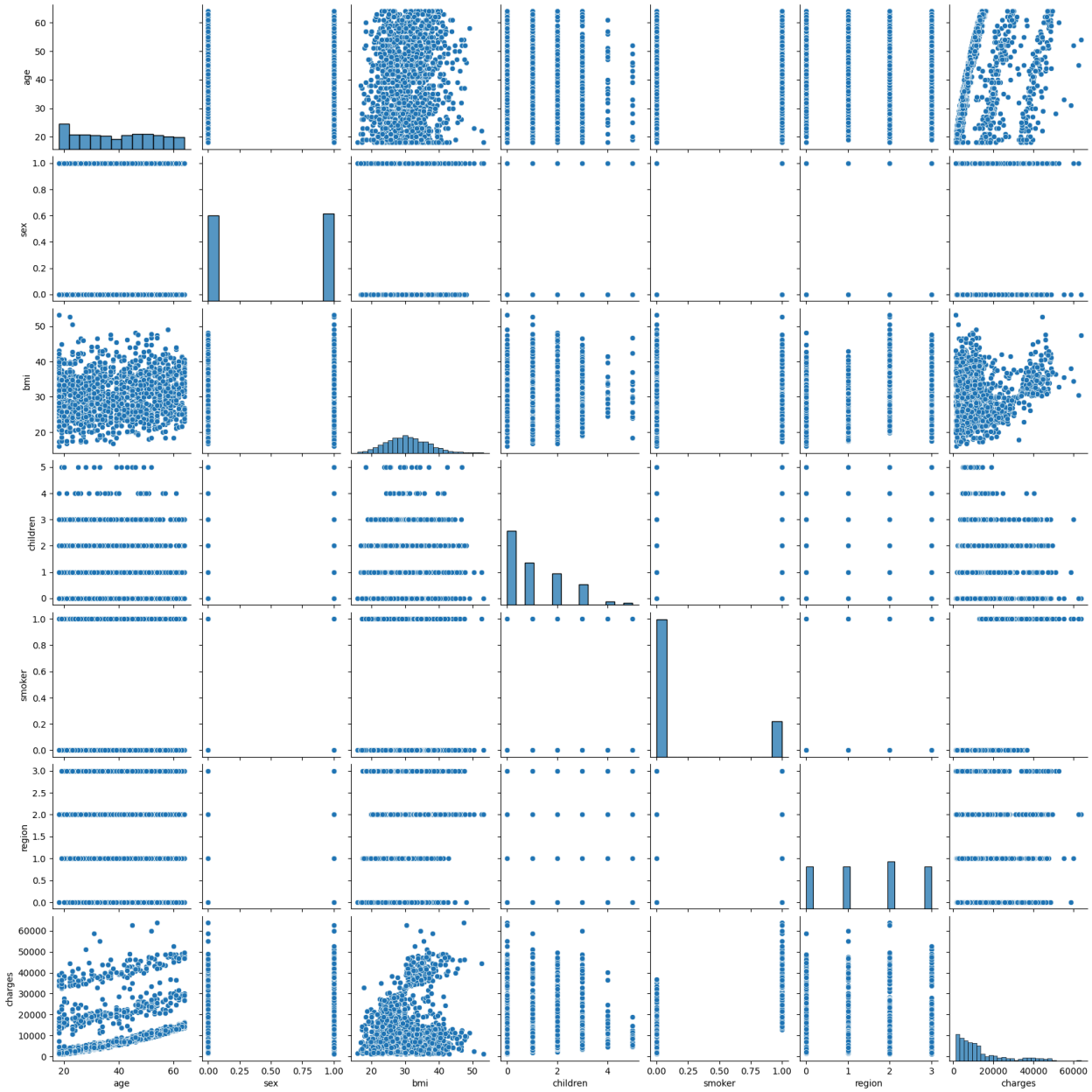
Out[26]:
```
age            int64
sex         category
bmi          float64
children       int64
smoker      category
region      category
charges      float64
dtype: object
```

In [27]:
```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(df.region.drop_duplicates())
df.region = label.transform(df.region)
label.fit(df.sex.drop_duplicates())
df.sex = label.transform(df.sex)
label.fit(df.smoker.drop_duplicates())
df.smoker = label.transform(df.smoker)
df.dtypes
```

Out[27]:
```
age            int64
sex            int32
bmi          float64
children       int64
smoker         int32
region         int32
charges      float64
dtype: object
```

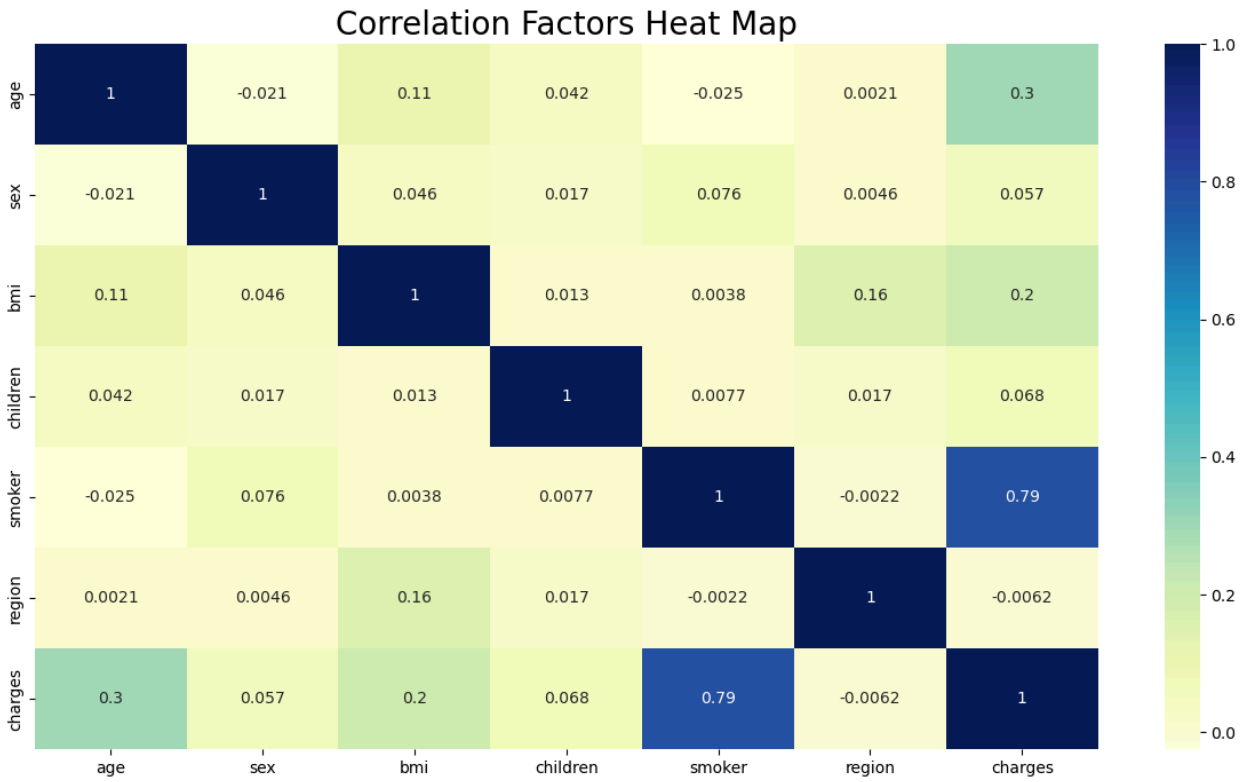
In [28]:
```
sns.pairplot(df)
```

Out[28]: <seaborn.axisgrid.PairGrid at 0x2877d850890>



In [29]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.corr(), annot = True, cmap = 'YlGnBu').set_title('Correlation Factors Heat Map', size = '20')
```

Out[29]: Text(0.5, 1.0, 'Correlation Factors Heat Map')



In [30]:
```
from sklearn.model_selection import train_test_split as holdout
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

In [ ]:

In [ ]:

In [31]:
```
features = df.drop(['charges'], axis = 1)
targets = df['charges']
```

In [ ]:

In [ ]:

In [32]:
```
x_train, x_test, y_train, y_test = holdout(features, targets, test_size = 0.2, random_state = 0)
```

In [33]:
```
Lin_reg_model = LinearRegression()
Lin_reg_model.fit(x_train, y_train)
print('Intercept:', Lin_reg_model.intercept_)
print('Coefficients:', Lin_reg_model.coef_)
```

```
Intercept: -11661.98390882442
Coefficients: [  253.99185244   -24.32455098   328.40261701   443.72929547
 23568.87948381  -288.50857254]
```

In [34]:
```
Lin_reg_model_train_pred = Lin_reg_model.predict(x_train)
Lin_reg_model_test_pred = Lin_reg_model.predict(x_test)
```

In [35]:
```
Lin_reg_model_train_mse = mean_squared_error(y_train, Lin_reg_model_train_pred)
Lin_reg_model_test_mse = mean_squared_error(y_test, Lin_reg_model_test_pred)
print('MSE train data: {:.3}, \nMSE test data: {:.3}\n'.format(Lin_reg_model_train_mse, Lin_reg_model_test_mse)
```

```
MSE train data: 3.77e+07,
MSE test data: 3.18e+07
```

In [36]:
```
print('RMSE train data: {:.3}, \nRMSE test data: {:.3}\n'.format(
    np.sqrt(np.absolute(Lin_reg_model_train_mse)),
    np.sqrt(np.absolute(Lin_reg_model_train_mse))))
print('R2 train data: {:.3}, \nR2 test data: {:.3}\n'.format(
    r2_score(y_train,Lin_reg_model_train_pred),
    r2_score(y_test,Lin_reg_model_test_pred)))
print('Model Score:', Lin_reg_model.score(x_test, y_test))
```

```
RMSE train data: 6.14e+03,
RMSE test data: 6.14e+03

R2 train data: 0.737,
R2 test data: 0.8

Model Score: 0.7998747145449958
```

In [37]:
```
from sklearn.ensemble import RandomForestRegressor as rfr
```

In [47]:
```
from sklearn.model_selection import GridSearchCV
```

In [49]:
```
plt.figure(figsize = (6, 4))
```

Out[49]:
```
<Figure size 600x400 with 0 Axes>

<Figure size 600x400 with 0 Axes>
```

In [58]:
```
from sklearn.preprocessing import PolynomialFeatures
features = df.drop(['charges','sex','region'], axis = 1)
target = df.charges
```

In [59]:
```
pol = PolynomialFeatures (degree = 2)
x_pol = pol.fit_transform(features)
x_train, x_test, y_train, y_test = holdout(x_pol, target, test_size = 0.2, random_state = 0)
```

In [60]:
```
Pol_reg = LinearRegression()
Pol_reg.fit(x_train, y_train)
```

Out[60]:
```
▾ LinearRegression

LinearRegression()
```

In [61]:
```
y_train_predic = Pol_reg.predict(x_train)
y_test_predic = Pol_reg.predict(x_test)
print('Intercept:', Pol_reg.intercept_)
print('Coefficients:', Pol_reg.coef_)
```

```
Intercept: -5325.881705252052
Coefficients: [ 0.00000000e+00 -4.01606591e+01  5.23702019e+02  8.52025026e+02
 -9.52698471e+03  3.04430186e+00  1.84508369e+00  6.01720286e+00
  4.20849790e+00 -9.38983382e+00  3.81612289e+00  1.40840670e+03
 -1.45982790e+02 -4.46151855e+02 -9.52698471e+03]
```

In [62]:
```
print('\nModel Accuracy Score:', (Pol_reg.score(x_test, y_test))*100)
```

```
Model Accuracy Score: 88.12595703345232
```

In [63]:
```
targets.head()
```

Out[63]:
```
0    16884.92400
1     1725.55230
2     4449.46200
3    21984.47061
4     3866.85520
Name: charges, dtype: float64
```

In [64]:
```
y_test_predic = Pol_reg.predict(x_test)
```

In [65]:
```
final_values = pd.DataFrame({'Actual values': y_test, 'Predicted values': y_test_predic})
final_values
```

Out[65]:

	Actual values	Predicted values
578	9724.53000	12101.156323
610	8547.69130	10440.782266
569	45702.02235	48541.022951
1034	12950.07120	14140.067522
198	9644.25250	8636.235727
...	...	...
1084	15019.76005	16712.196281
726	6664.68595	8654.565461
1132	20709.02034	12372.050609
725	40932.42950	41465.617268
963	9500.57305	10941.780705

268 rows × 2 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]: