Medical Costs Analysis for Smokers and Non-smokers

Analysis and Regression Models

Abstract :

The cumulative impact of excess medical care required by smokers at all ages while alive outweighs shorter life expectancy, and smokers incur higher expenditures for medical care over their lifetimes than never-smokers. This accords with the findings by Manning et al. (1989) of positive lifetime medical care costs per pack of cigarettes, but disagrees with the results found by Leu and Schaub (1983, 1985) for Swiss males. The contradictory conclusions of the analyses are undoubtedly due to a large difference in the amount of medical care used by smokers relative to neversmokers in the United States and Swiss data. Excess expenditures increase with the amount smoked among males and females so that lifetime medical costs of male heavy smokers are 47 percent higher than for neversmokers when discounted at 3 percent. Each year more than one million young people start to smoke and add an extra 9 to 10 billion (in 1990 dollars discounted at 3 percent) to the nation's health care bill over their lifetimes. Given the smoking behavior, medical care utilization and costs of care, and population size embedded in the data used in this analysis, I have concluded that in the first five years from baseline the population of smokers aged 25 and over incurs excess medical expenditures totaling 187 billion dollar, which is 2,324 dollar per smoker. The excess cost of medical care associated with cigarette smoking is 18 percent of expenditures for hospital care, physicians' services, and nursing-home care required by all persons (smokers and neversmokers) aged 25 and over. In the absence of large and rapid changes in the values of the underlying parameters, 187 billion dollar, 18 percent of medical expenditures, can be taken as the premium currently being paid every five years to provide medical care for the excess disease suffered by smokers. Even without the addition of any new smokers, the present value of the bill that will be incurred for excess medical care required by the current population of smokers over their remaining lifetimes is high. The civilian noninstitutionalized population of cigarette smokers in 1985 who are age 25 and older is expected to incur over its remaining lifetime excess medical expenditures of $501 billion, or 6,239 dollar per smoker. It is possible that future changes beyond recent historical trends in the habits of those who currently smoke, such as reductions in the amount smoked, higher rates of quitting, whether occurring fortuitously or brought about by design, may result in lower costs of smoking than estimated.

In [1]:
```
from IPython.display import Image
Image("https://cdn2.poz.com/90226_iStock-174692627.jpg_90b75fde-8eed-49b9-9b56-f917a8952787_x2.jpeg")
```

Out[1]:

PART 1 :: Analysis :

Columns:

age: age of primary beneficiary

sex: insurance contractor gender, female, male

bmi: body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9

children: number of children covered by health insurance / number of dependents

smoker: smoking

region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

charges: individual medical costs billed by health insurance

In [2]:
```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as pl
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('insurance.csv')
```

In [3]:
```
data.head()
```

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

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [5]:
```
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)
```

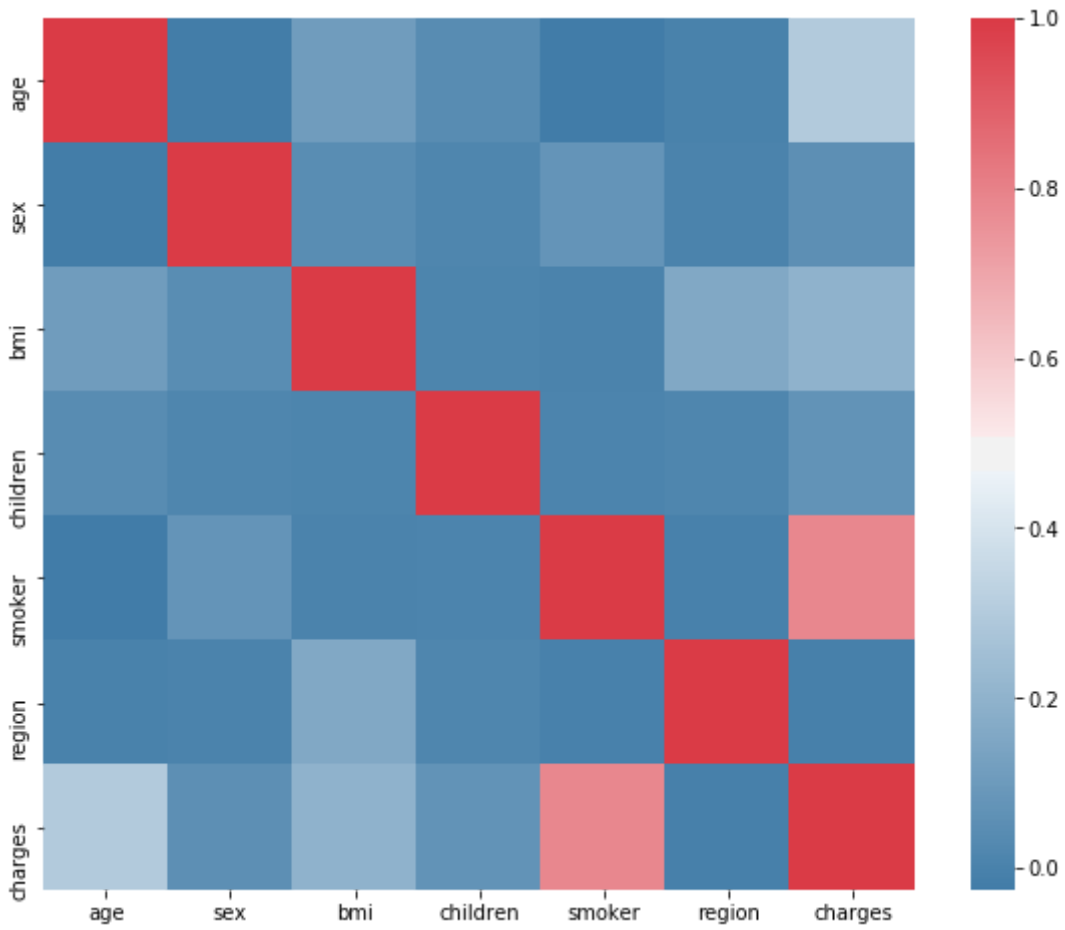
In [6]:
```
data.corr()['charges'].sort_values()
```

Out[6]:
```
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
```

Creating the Heat-Map to understand which co-relation model will be best for the dataset

In [7]:
```
f, ax = pl.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)
```

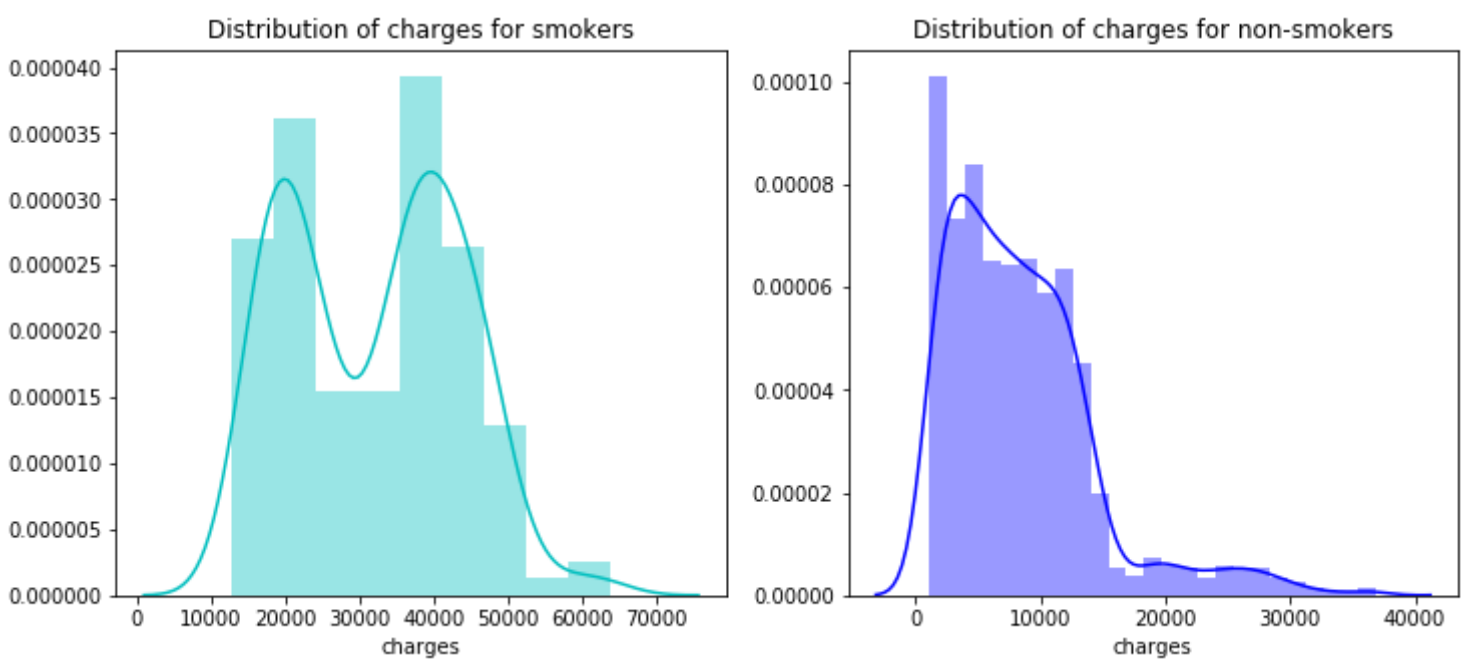
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0xbd32f08>



Distribution of Charges for Smokers and Non-smokers :

In [8]:
```
f= pl.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')
```

Out[8]: Text(0.5, 1.0, 'Distribution of charges for non-smokers')

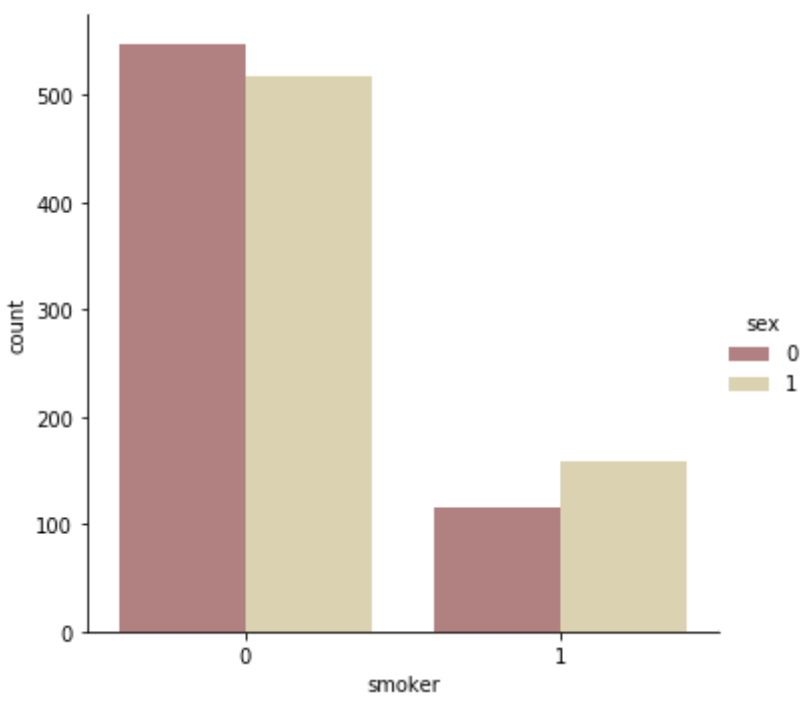


Here we can see that the distribution charges for the smokers ranges from 30000 dollar to 40000 dollar where as for non smokers it shows that it is 10000 dollars. Which provides the great difference to be noteable.

Gender wise analysis for Smokers and Non-smokers

In [9]:
```
sns.catplot(x="smoker", kind="count",hue = 'sex', palette="pink", data=data)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0xc542b88>



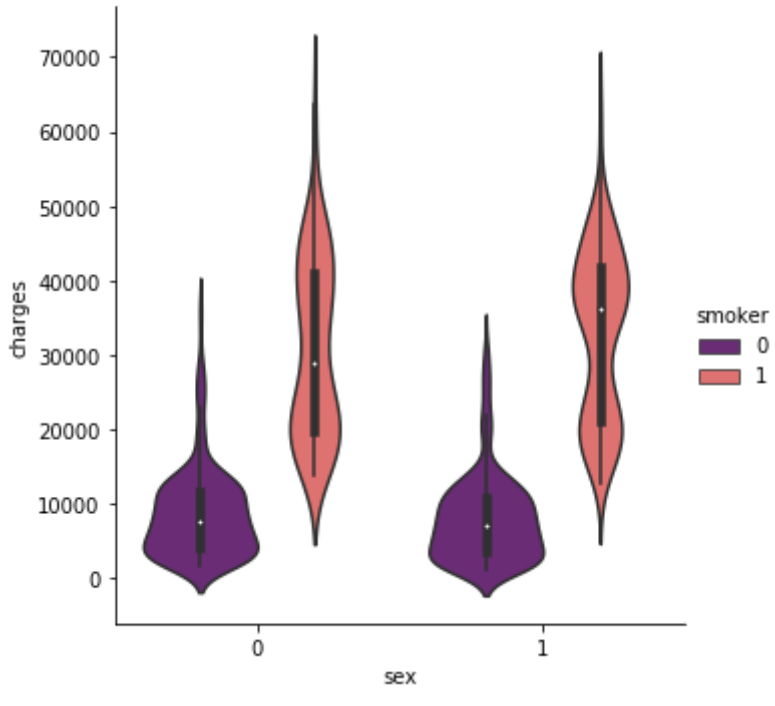
Here it shows the smoker and non smoker analysis with respect to the gender.

1st column shows the number of Non-smokers where Dark brown shows the women community and light brown shows the Men community

2nd Column shows the smokers where the Dark brown shows the Women community and the light brown shows the men community.

In [10]:
```
sns.catplot(x="sex", y="charges", hue="smoker",
            kind="violin", data=data, palette = 'magma')
```

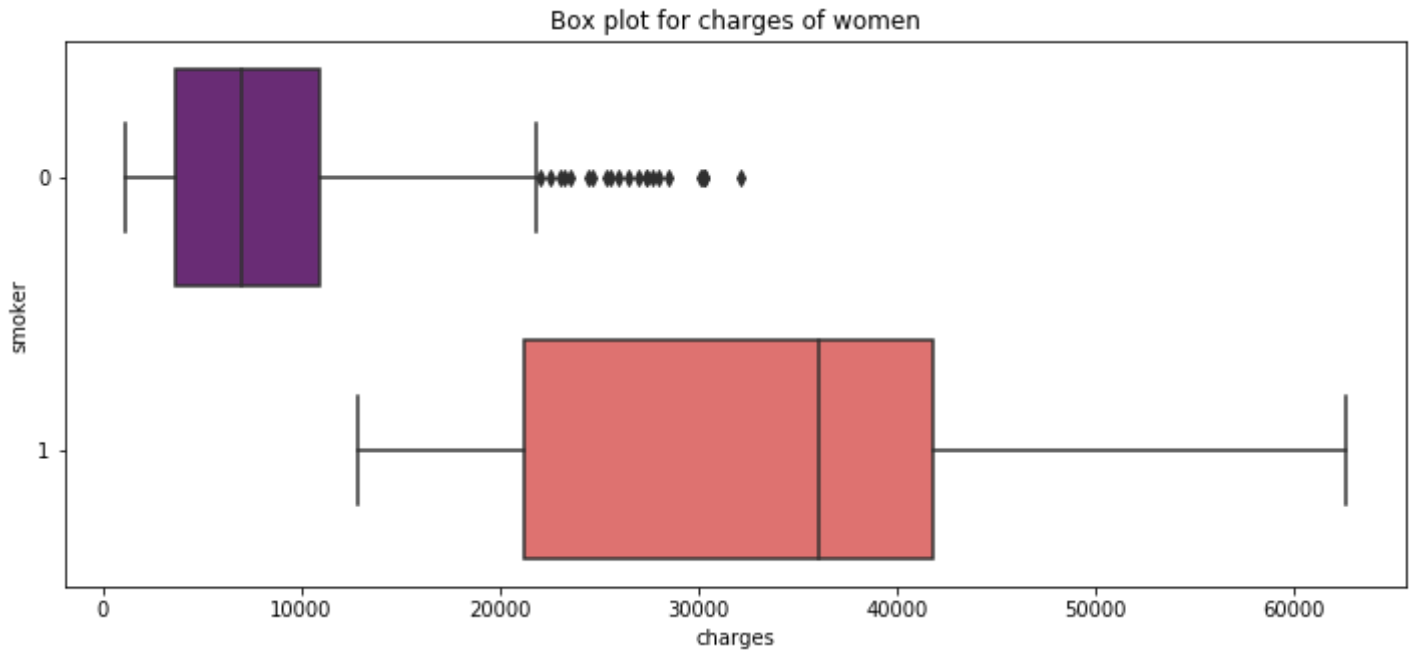
Out[10]: <seaborn.axisgrid.FacetGrid at 0xc567848>



Cost Analysis for Women Smokers and Non-smokers :

In [11]:
```
pl.figure(figsize=(12,5))
pl.title("Box plot for charges of women")
sns.boxplot(y="smoker", x="charges", data =  data[(data.sex == 1)] , orient="h", palette = 'magma')
```

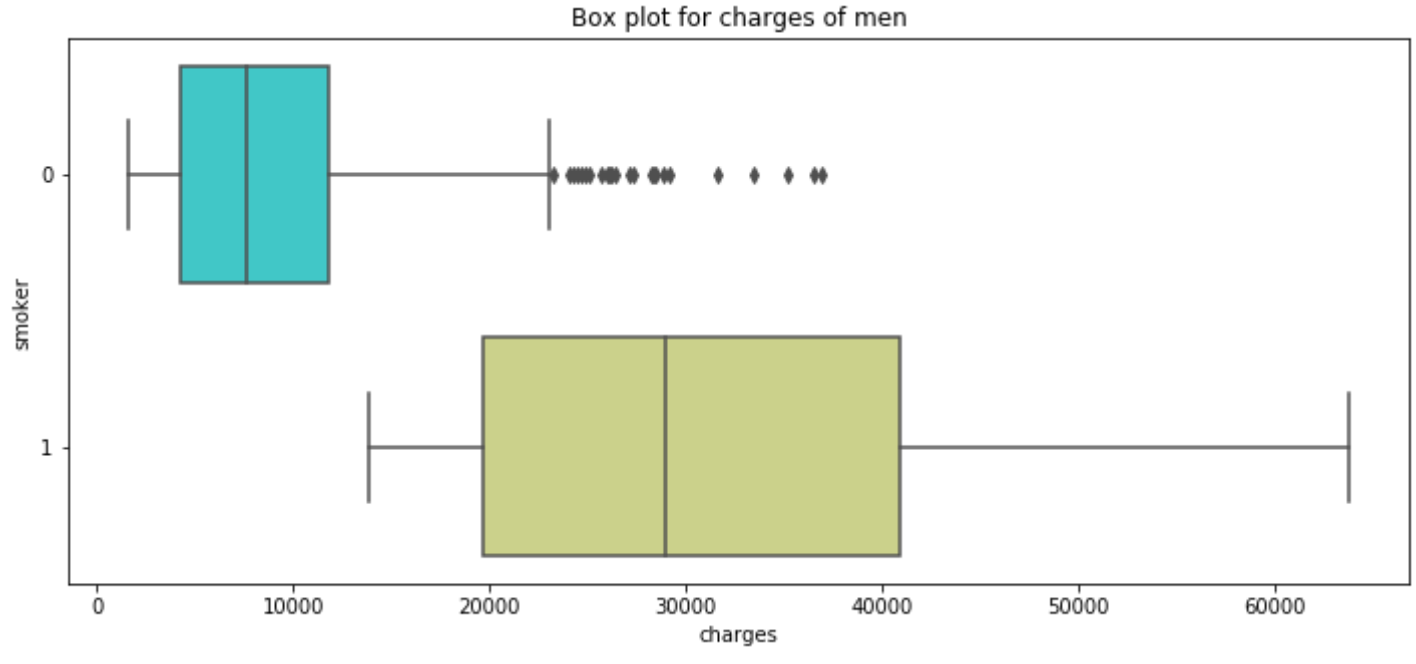
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0xbf70e08>



Cost Analysis of Men Smokers and Non-smokers :

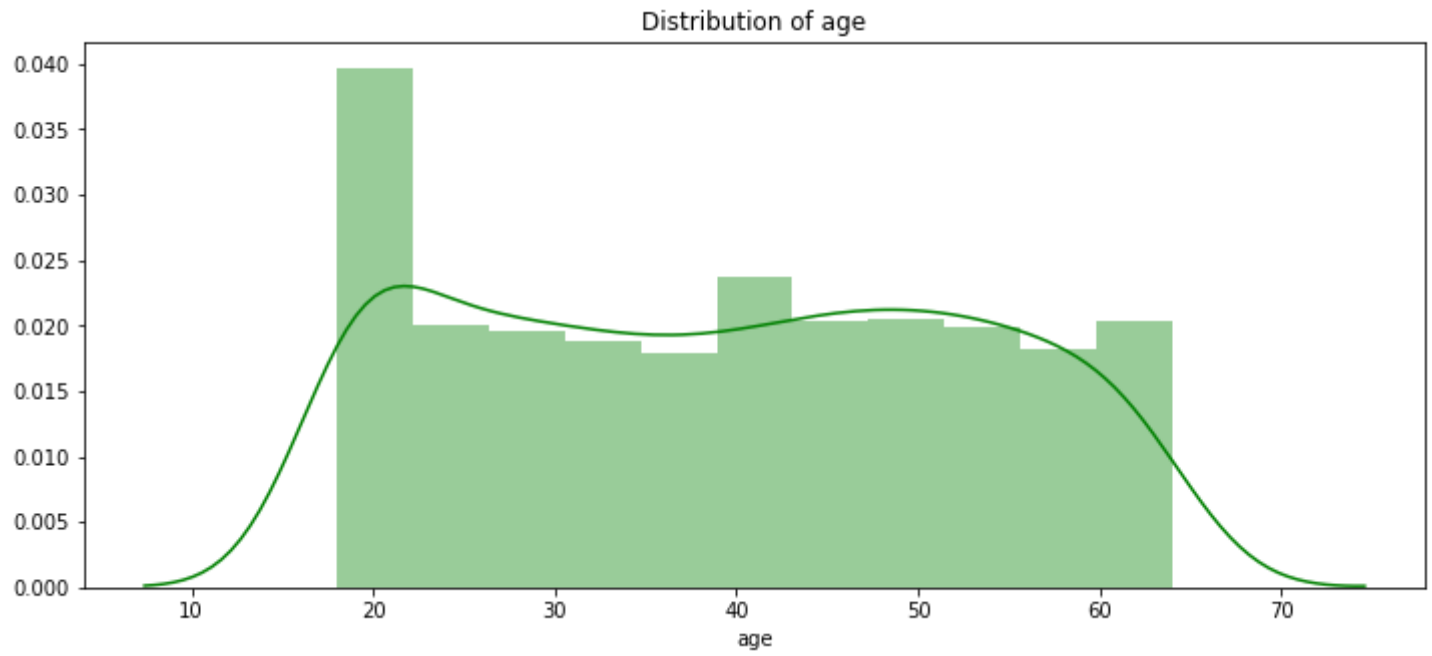
In [12]:
```
pl.figure(figsize=(12,5))
pl.title("Box plot for charges of men")
sns.boxplot(y="smoker", x="charges", data =  data[(data.sex == 0)] , orient="h", palette = 'rainbow')
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0xc7058c8>



Age Distribution for the Smokers :

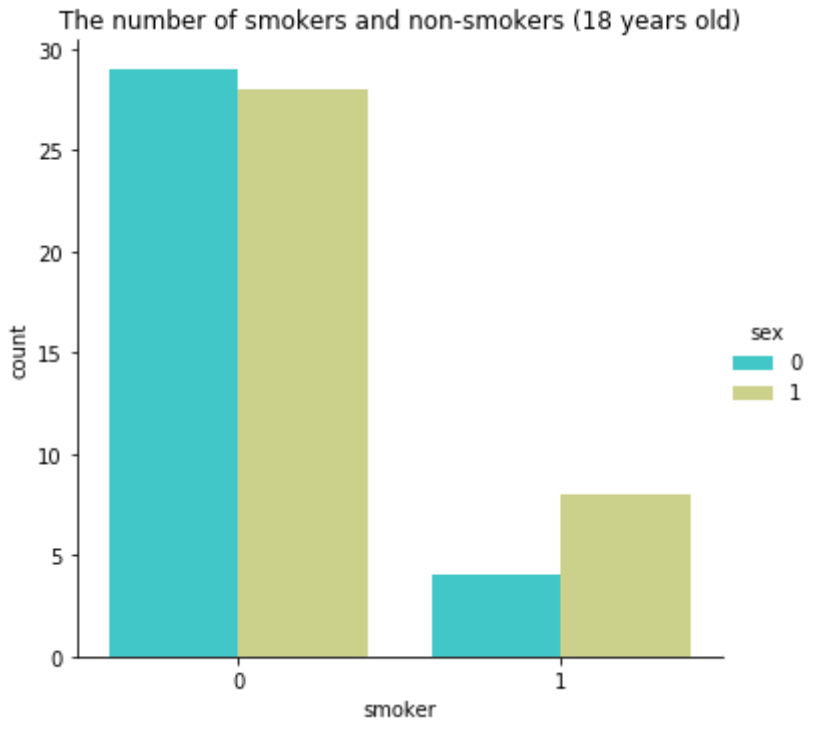
In [13]:
```
pl.figure(figsize=(12,5))
pl.title("Distribution of age")
ax = sns.distplot(data["age"], color = 'g')

```



Ratio of 18-years old Smokers and Non-smokers :

In [14]:
```
sns.catplot(x="smoker", kind="count",hue = 'sex', palette="rainbow", data=data[(data.age == 18)])
pl.title("The number of smokers and non-smokers (18 years old)")
```

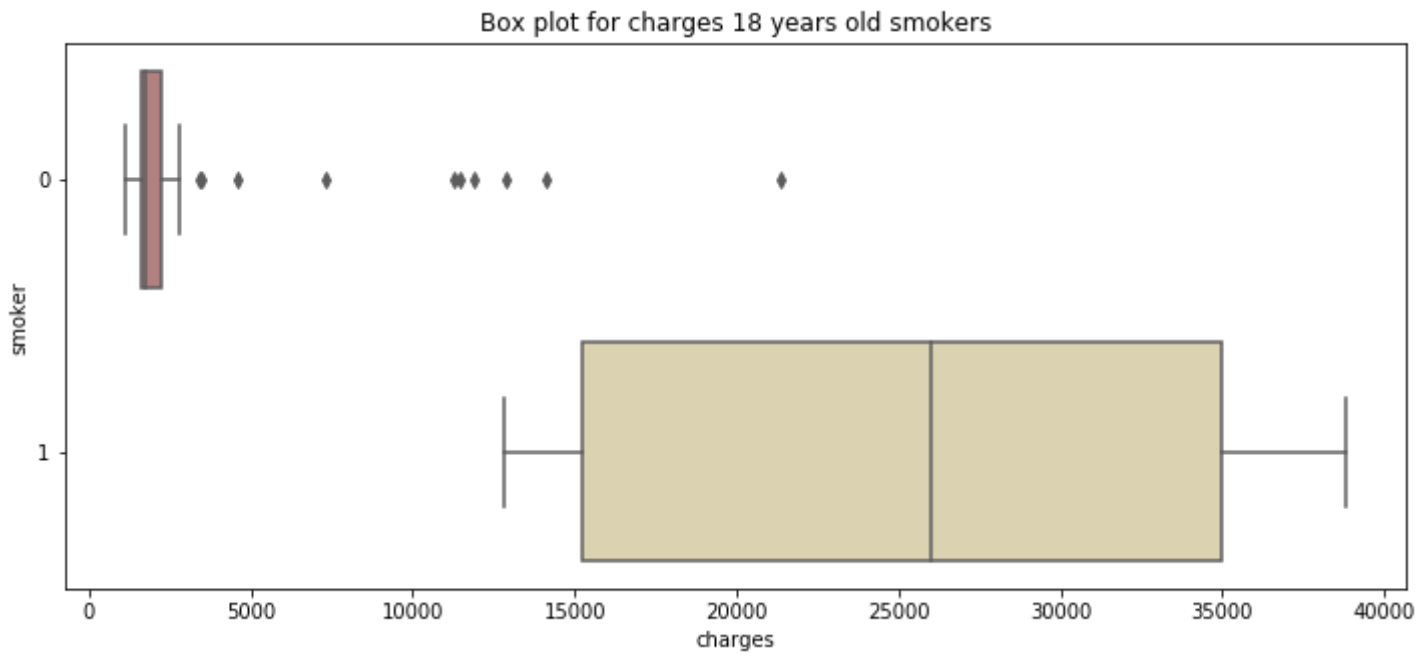
Out[14]: Text(0.5, 1, 'The number of smokers and non-smokers (18 years old)')



Charges for 18-years old Smokers and Non-smokers :

In [15]:
```
pl.figure(figsize=(12,5))
pl.title("Box plot for charges 18 years old smokers")
sns.boxplot(y="smoker", x="charges", data = data[(data.age == 18)] , orient="h", palette = 'pink')
```

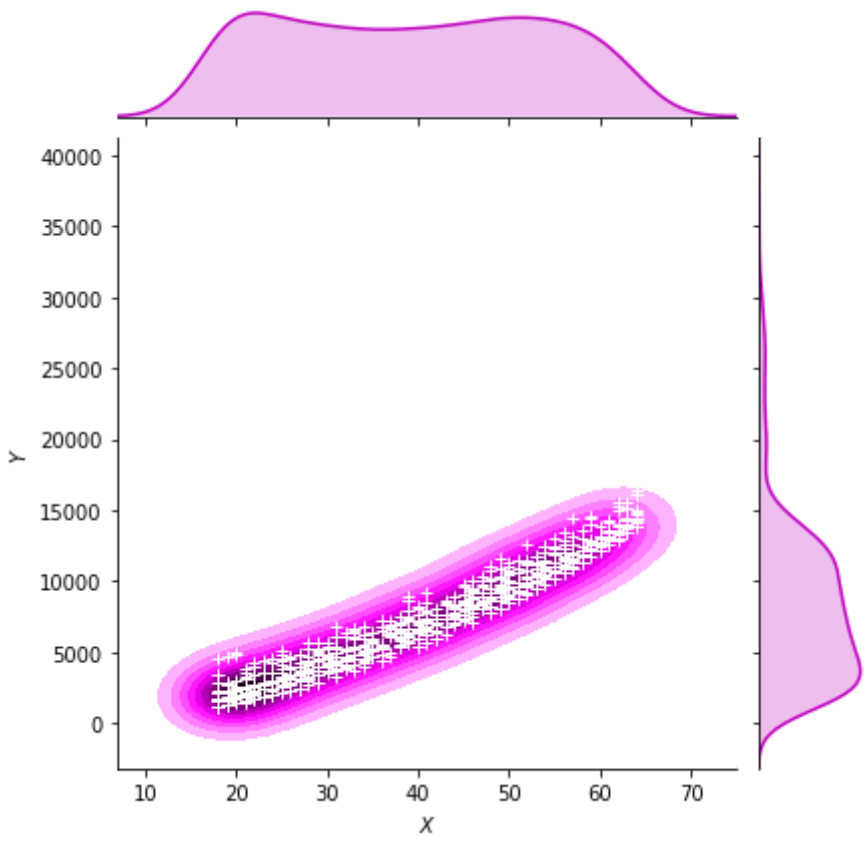
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0xc8e1908>



Jointplot : Charges and Age of the Non-smokers :

In [16]:
```
g = sns.jointplot(x="age", y="charges", data = data[(data.smoker == 0)],kind="kde", color="m")
g.plot_joint(pl.scatter, c="w", s=30, linewidth=1, marker="+")
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')
```

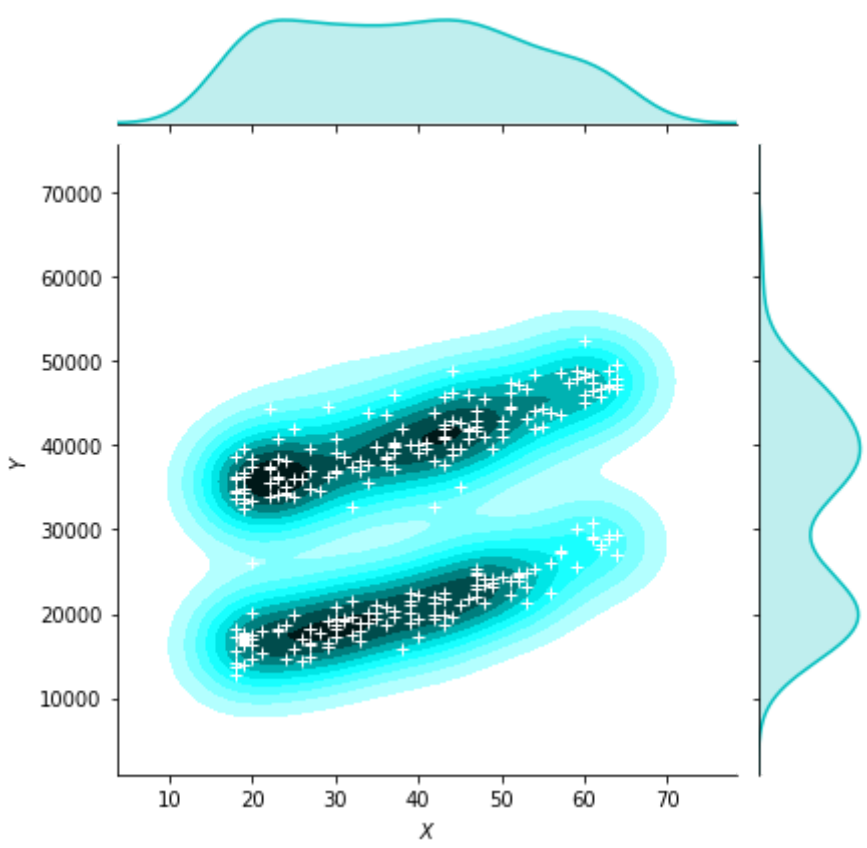
Out[16]: Text(0.5, 1, 'Distribution of charges and age for non-smokers')



Jointplot : Charges and Age of the Smokers :

In [17]:
```
g = sns.jointplot(x="age", y="charges", data = data[(data.smoker == 1)],kind="kde", color="c")
g.plot_joint(pl.scatter, c="w", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("$X$", "$Y$")
ax.set_title('Distribution of charges and age for smokers')
```

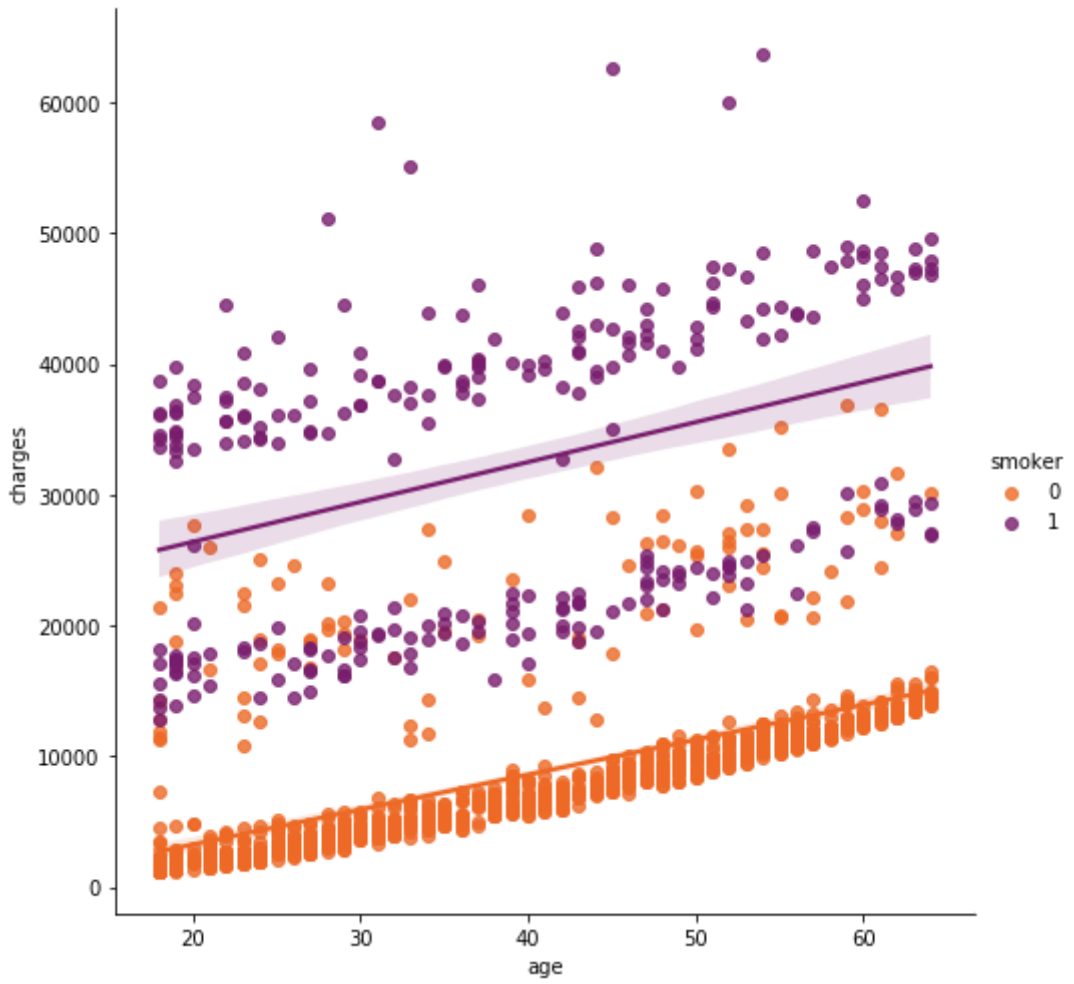
Out[17]: Text(0.5, 1, 'Distribution of charges and age for smokers')



Scatter Plot : Charges for Smokers and Non-smokers :

In [18]:
```
sns.lmplot(x="age", y="charges", hue="smoker", data=data, palette = 'inferno_r', size = 7)
ax.set_title('Smokers and non-smokers')
```

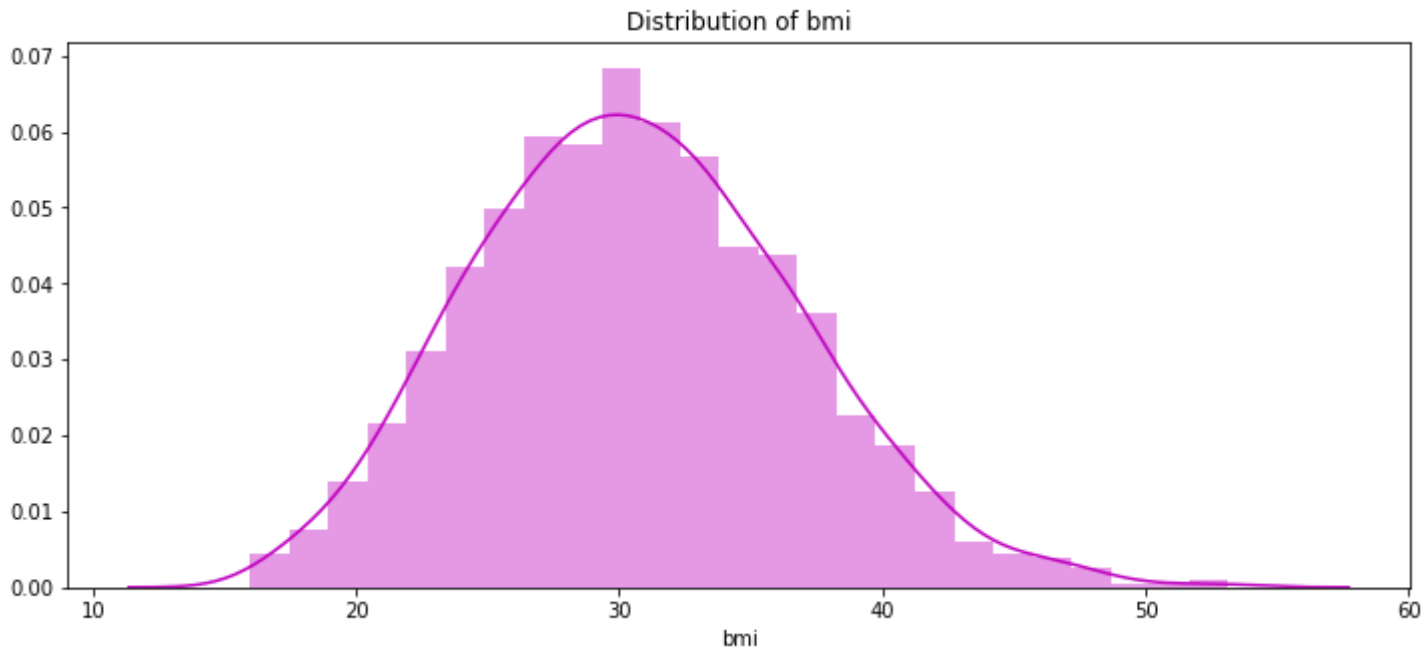
Out[18]: Text(0.5, 1, 'Smokers and non-smokers')



In non-smokers, the cost of treatment increases with age. That makes sense. So take care of your health, friends! In smoking people, we do not see such dependence. I think that it is not only in smoking but also in the peculiarities of the dataset. Such a strong effect of Smoking on the cost of treatment would be more logical to judge having a set of data with a large number of records and signs. But we work with what we have! Let's pay attention to bmi. I am surprised that this figure but affects the cost of treatment in patients. Or are we on a diet for nothing?

BMI Distribution :

In [19]:
```
pl.figure(figsize=(12,5))
pl.title("Distribution of bmi")
ax = sns.distplot(data["bmi"], color = 'm')
```

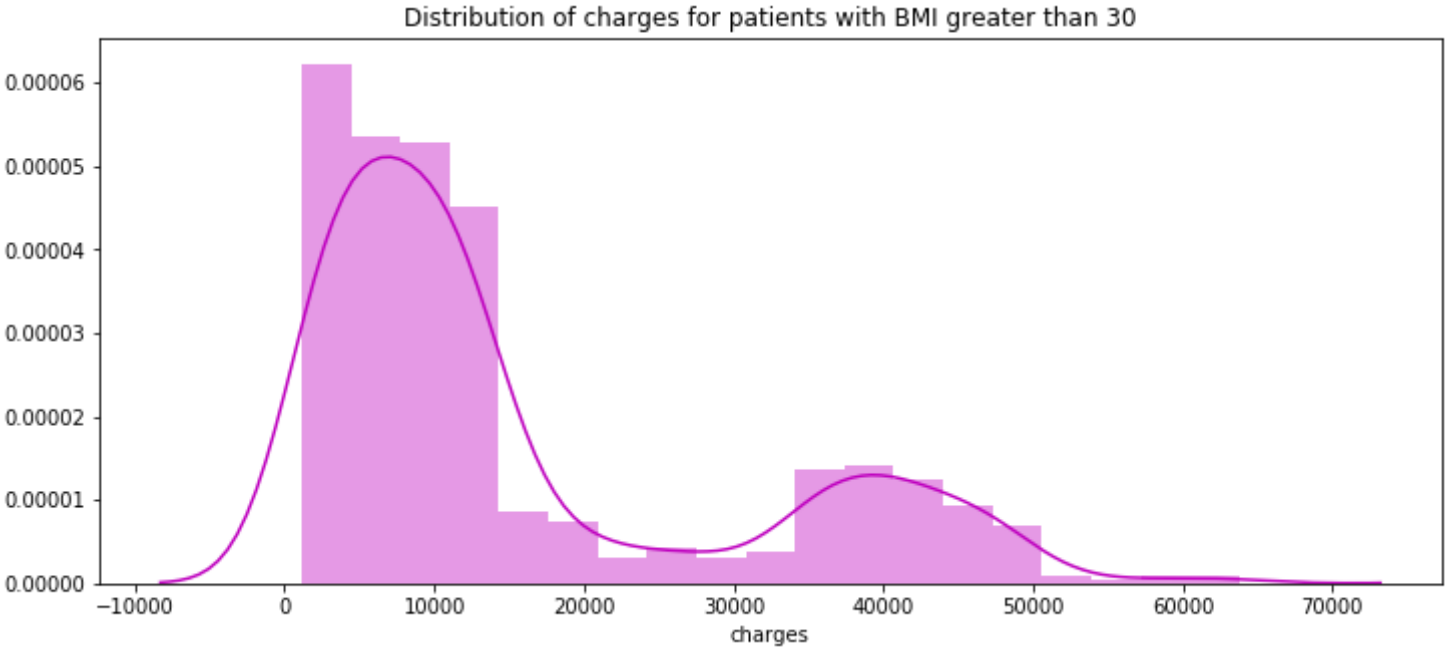


There's something insanely beautiful about this distribution, isn't there? The average BMI in patients is 30. I think I should go to the great Google with a question about this indicator.

With a value equal to 30 starts obesity. I also calculated my BMI and now I can safely eat a sandwich. Let's start to explore! First, let's look at the distribution of costs in patients with BMI greater than 30 and less than 30.

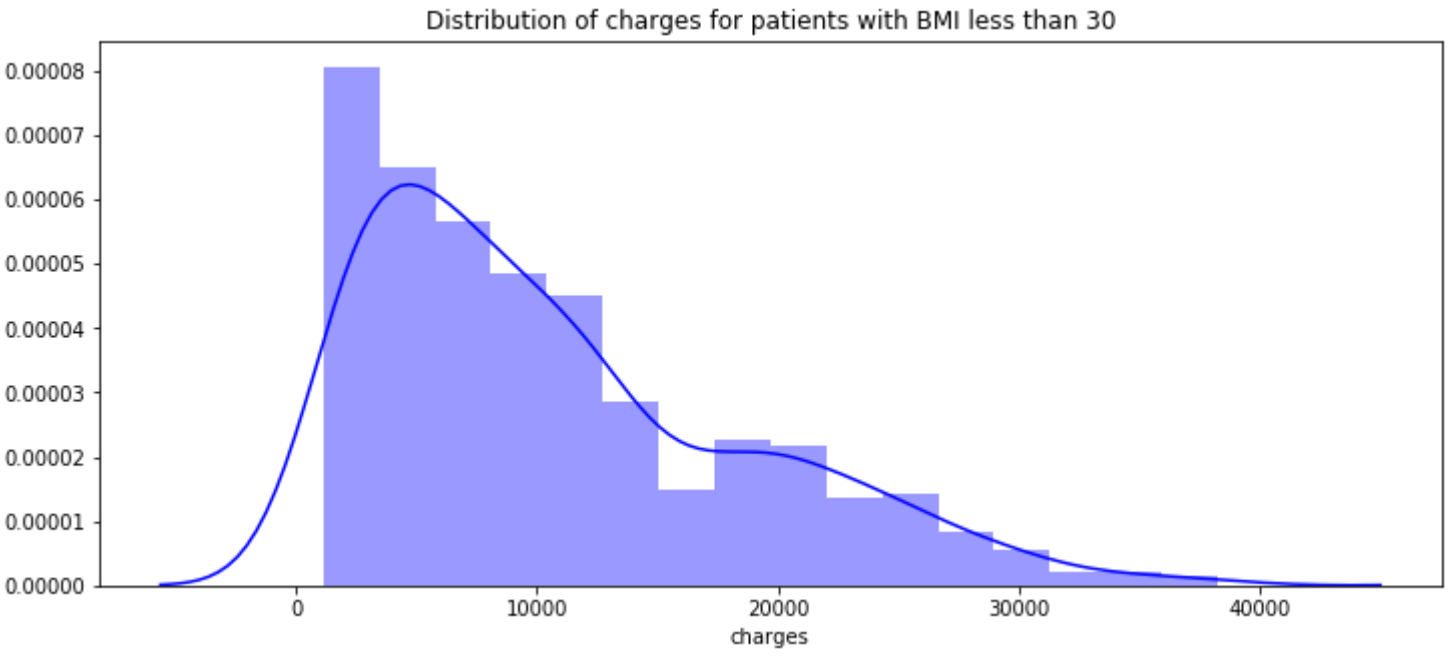
Distribution of Charges for Patients with BMI greater than 30 :

In [20]:
```
pl.figure(figsize=(12,5))
pl.title("Distribution of charges for patients with BMI greater than 30")
ax = sns.distplot(data[(data.bmi >= 30)]['charges'], color = 'm')
```



Distribution of Charges for patients with BMI less than 30 :

In [21]:
```
pl.figure(figsize=(12,5))
pl.title("Distribution of charges for patients with BMI less than 30")
ax = sns.distplot(data[(data.bmi < 30)]['charges'], color = 'b')
```

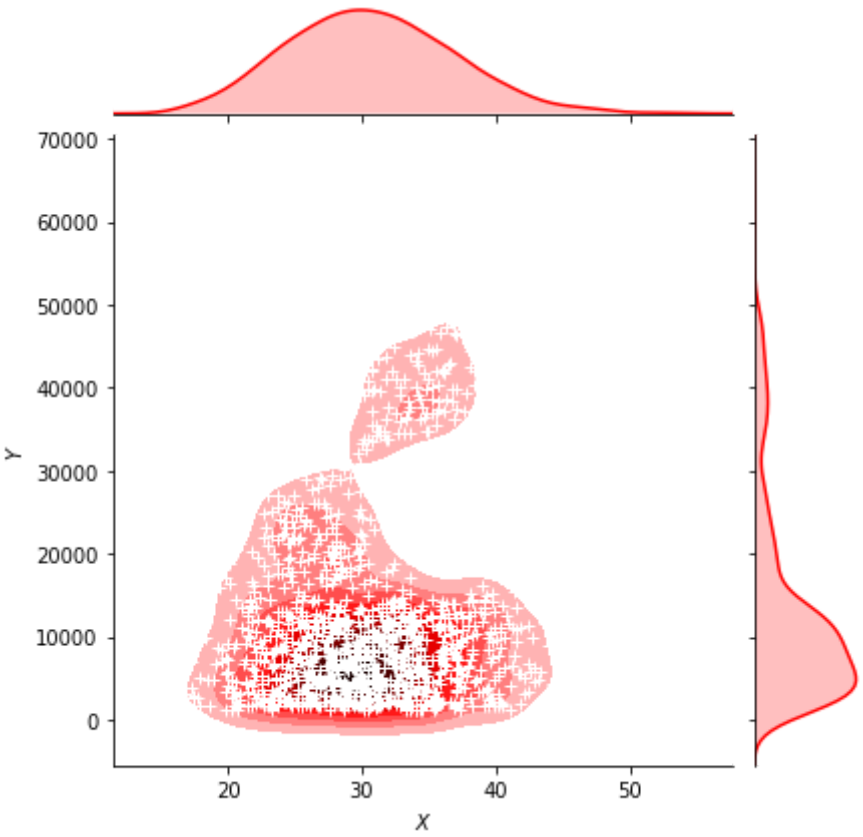


Patients with BMI above 30 spend more on treatment!

Jointplot : Distribution of BMI and Charges :

In [22]:
```
g = sns.jointplot(x="bmi", y="charges", data = data,kind="kde", color="r")
g.plot_joint(pl.scatter, c="w", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("$X$", "$Y$")
ax.set_title('Distribution of bmi and charges')
```

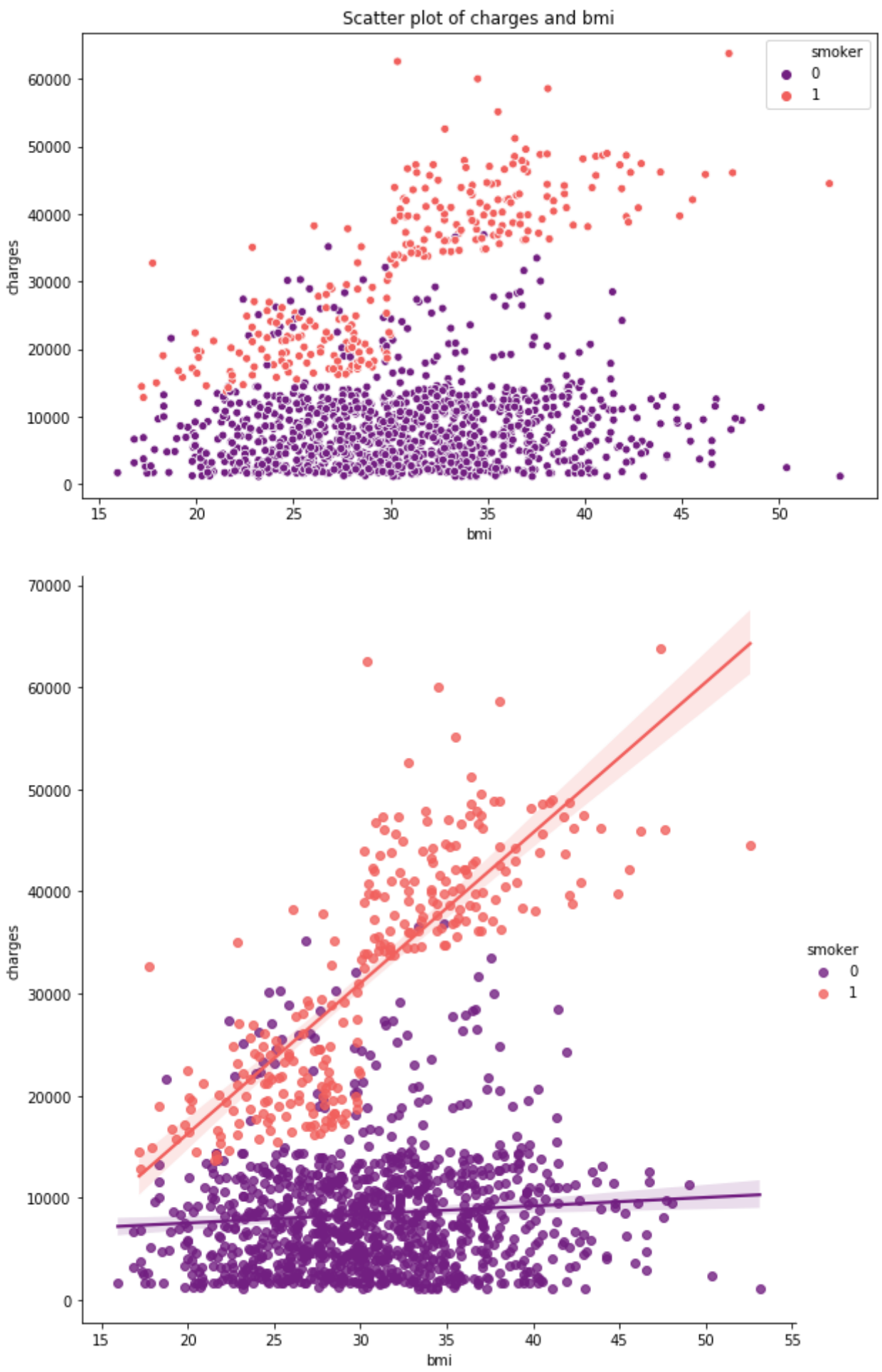
Out[22]: Text(0.5, 1, 'Distribution of bmi and charges')



Analysis on Charges of Smokers and Non-smokers using the Scatter Plot

In [23]:
```
pl.figure(figsize=(10,6))
ax = sns.scatterplot(x='bmi',y='charges',data=data,palette='magma',hue='smoker')
ax.set_title('Scatter plot of charges and bmi')

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

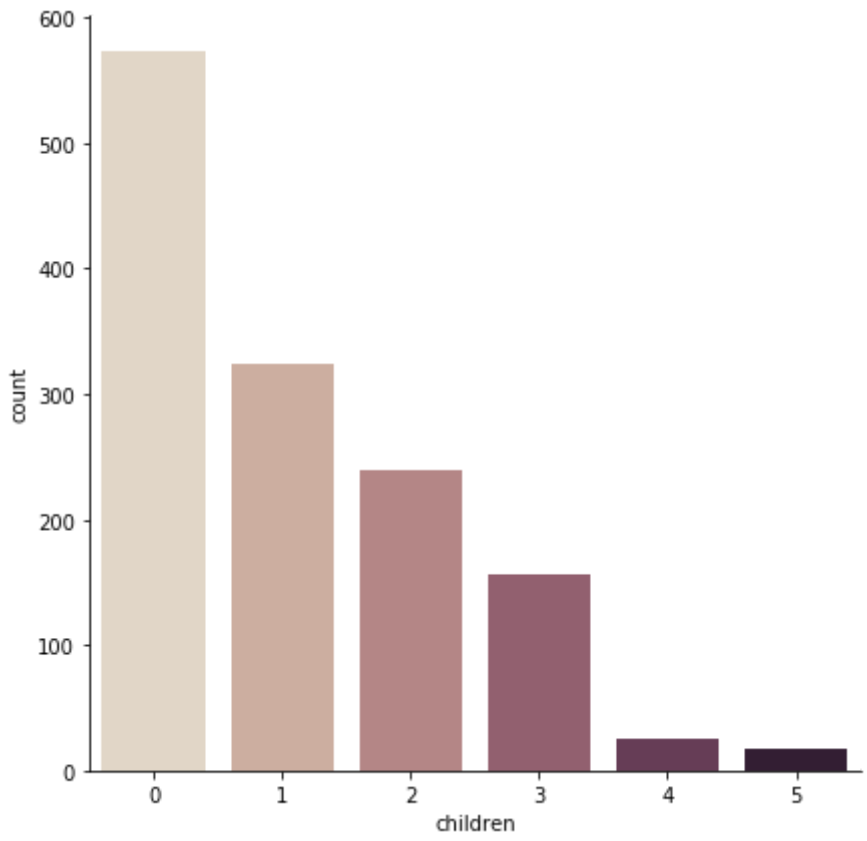
Out[23]: <seaborn.axisgrid.FacetGrid at 0xe8f7888>



Smoker Parents who have children at their home :

In [24]:
```
sns.catplot(x="children", kind="count", palette="ch:.25", data=data, size = 6)
```

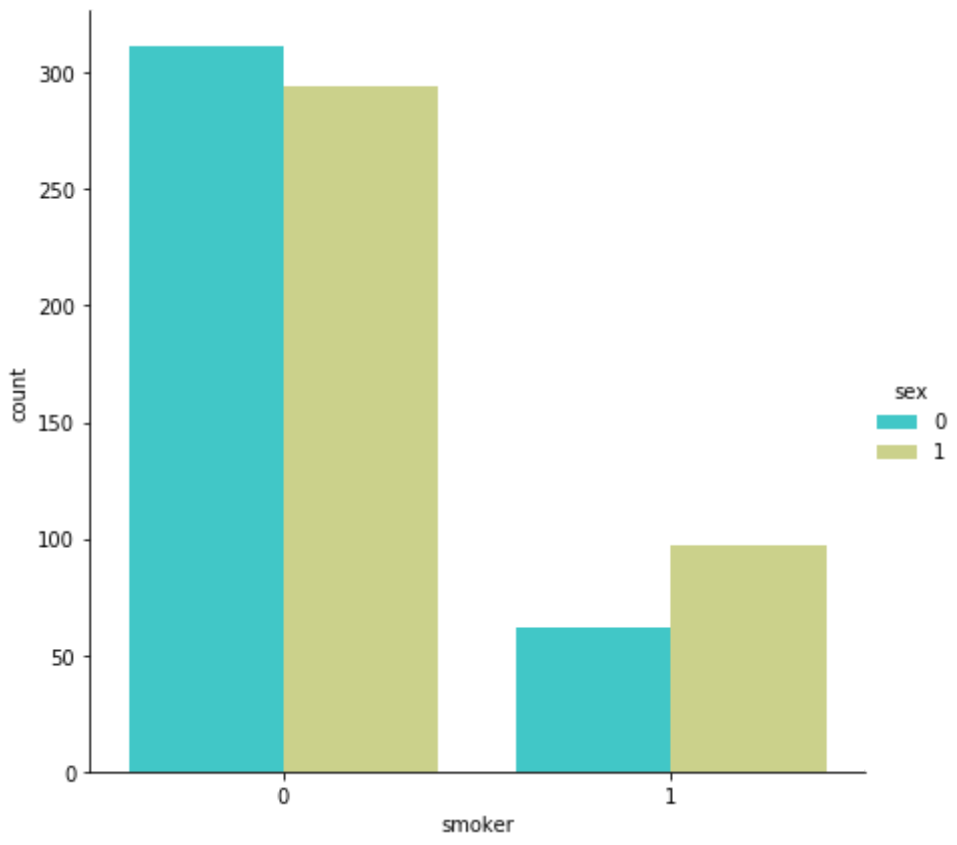
Out[24]: <seaborn.axisgrid.FacetGrid at 0xeb35308>



Distribution of Smoker and Non-smokers who have children

In [25]:
```
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')
```

Out[25]: Text(0.5, 1, 'Smokers and non-smokers who have childrens')



Health Effects of Smoking and Secondhand Smoke on Children :

1. Because their bodies are developing, infants and young children are especially vulnerable to the poisons in secondhand smoke.
2. Both babies whose mothers smoke while pregnant and babies who are exposed to secondhand smoke after birth are more likely to die from sudden infant death syndrome (SIDS) than babies who are not exposed to cigarette smoke.
3. Mothers who are exposed to secondhand smoke while pregnant are more likely to have lower birth weight babies, which makes babies weaker and increases the risk for many health problems.
4. Babies whose mothers smoke while pregnant or who are exposed to secondhand smoke after birth have weaker lungs than other babies, which increases the risk for many health problems.
5. Secondhand smoke exposure causes acute lower respiratory infections such as bronchitis and pneumonia in infants and young children.
6. Secondhand smoke exposure causes children who already have asthma to experience more frequent and severe attacks.
7. Secondhand smoke exposure causes respiratory symptoms, including cough, phlegm, wheezing, and breathlessness, among school-aged children.

8. Children exposed to secondhand smoke are at increased risk for ear infections and are more likely to need an operation to insert ear tubes for drainage.

PART 2 :: Regression Models based on the dataset :

Linear regression model :

In [26]:
```
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
```

Training the Dataset to predict :

In [27]:
```
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
```

Not bad for such a lazy implementation, even without data normalization:D After all, the data will not always be so "good". So don't forget to pre-process the data. I'll show you all this later when I try to implement my own linear regression. So don't be mad at me please :) Now let's add polynomial signs. And look at the result.

In [28]:
```
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))
```

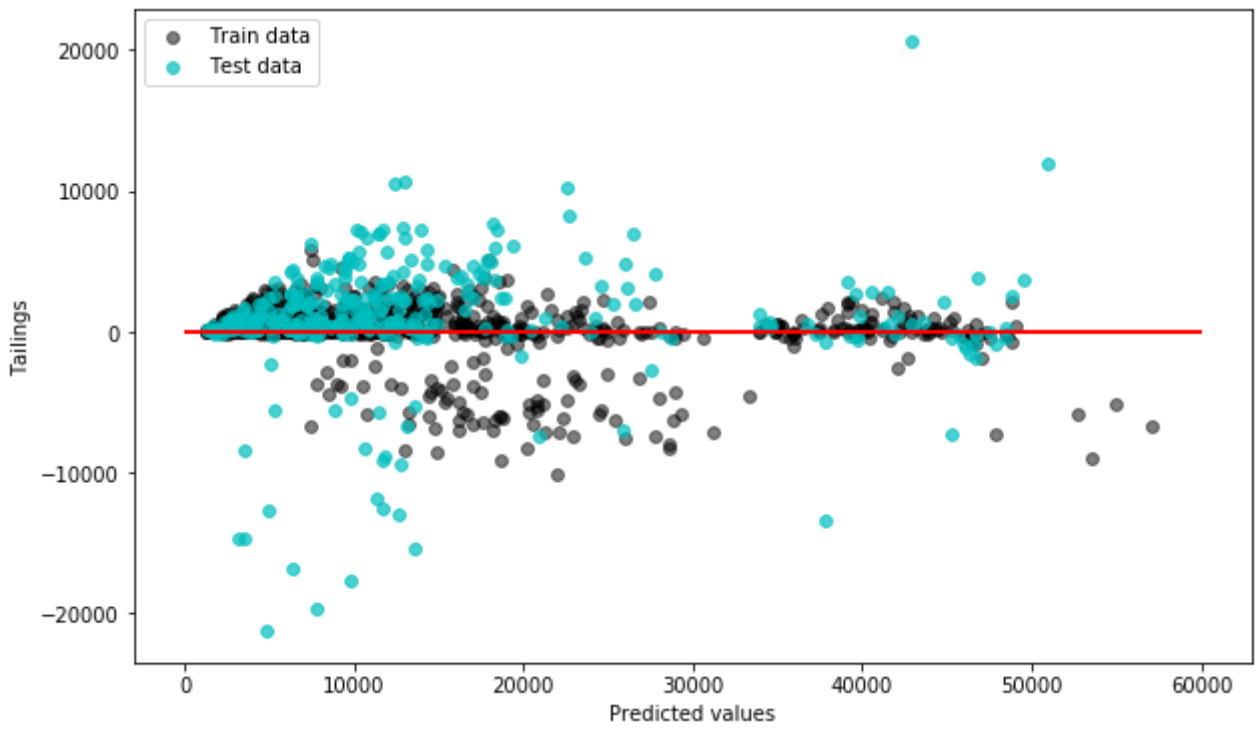
```
0.884919734414723
```

Already good. Our model predicts well the cost of treatment of patients. I think we could limit ourselves to creating two or three polynomial features, but the data set is so small, so we went the easy way. And finally try RandomForestRegressor. I've never used this algorithm in regression analysis.

In [29]:
```
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: 3729086.094, MSE test data: 19933823.142
R2 train data: 0.974, R2 test data: 0.873
```

In [30]:
```
pl.figure(figsize=(10,6))

pl.scatter(forest_train_pred,forest_train_pred - y_train,
          c = 'black', marker = 'o', s = 35, alpha = 0.5,
          label = 'Train data')
pl.scatter(forest_test_pred,forest_test_pred - y_test,
          c = 'c', marker = 'o', s = 35, alpha = 0.7,
          label = 'Test data')
pl.xlabel('Predicted values')
pl.ylabel('Tailings')
pl.legend(loc = 'upper left')
pl.hlines(y = 0, xmin = 0, xmax = 60000, lw = 2, color = 'red')
pl.show()
```



Good result. But we see a noticeable retraining of the algorithm on the training data.

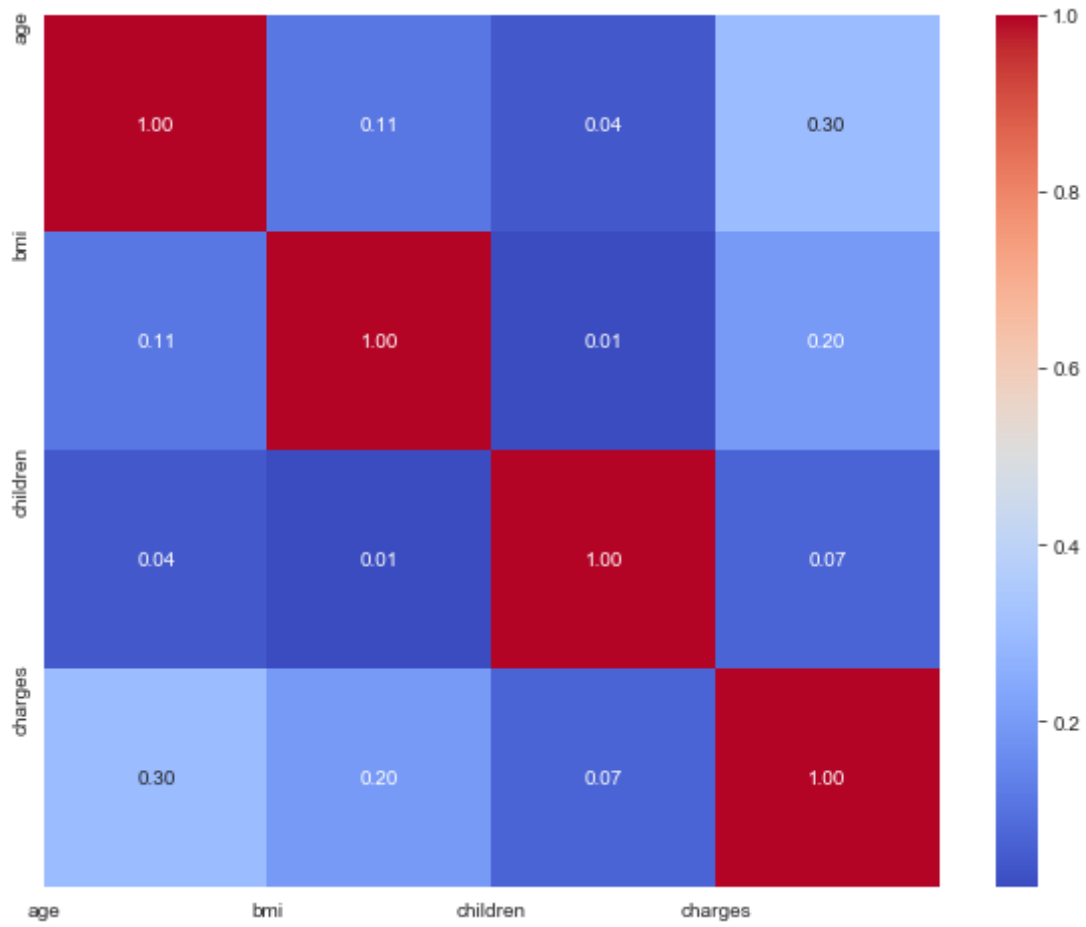
Data Visualization and Preprocessing :

In [31]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# regression models
from sklearn.preprocessing import StandardScaler # for feature scaling
from sklearn.pipeline import Pipeline # for using pipeline
from sklearn.linear_model import LinearRegression # for linear regression
from sklearn.preprocessing import PolynomialFeatures # for adding polynomial features
from sklearn.linear_model import Ridge # for ridge regression
from sklearn.linear_model import Lasso # for lasso regression
from sklearn.svm import SVR # for support vector regression
from sklearn.tree import DecisionTreeRegressor # for decisiton tree regression
from sklearn.ensemble import RandomForestRegressor # for random forest regression
# hyptertuning
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
# extra
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from collections import Counter
from IPython.core.display import display, HTML
sns.set_style('darkgrid')
```

In [32]:
```
dataset = pd.read_csv('insurance.csv')

```

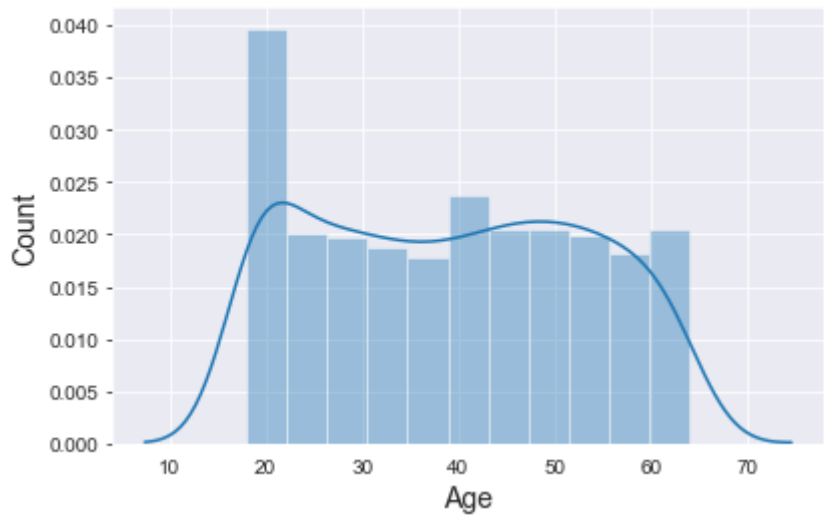
Heatmap shows the better corelation among the attributes :

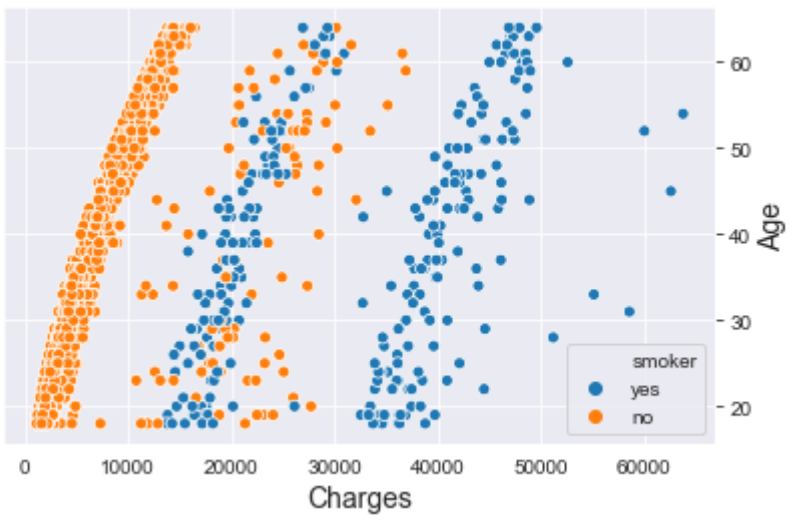
In [33]:
```
corr = dataset.corr()
#Plot figsize
fig, ax = plt.subplots(figsize=(10, 8))
#Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr, cmap='coolwarm', annot=True, fmt=".2f")
#Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
#Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
#show plot
plt.show()
```



Age v/s Count and Age v/s Charges :

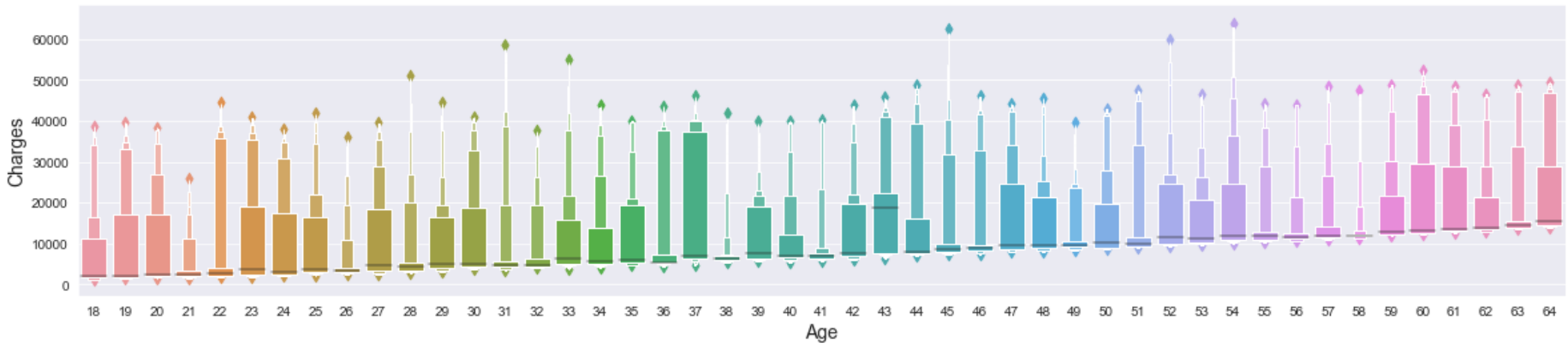
In [34]:
```
f, axes = plt.subplots(1,2,figsize=(14,4))

sns.distplot(dataset['age'], ax = axes[0])
axes[0].set_xlabel('Age', fontsize=14)
axes[0].set_ylabel('Count', fontsize=14)
axes[0].yaxis.tick_left()

sns.scatterplot(x = 'charges', y = 'age', data = dataset, hue = 'smoker', ax = axes[1])
axes[1].set_xlabel('Charges', fontsize=14)
axes[1].set_ylabel('Age', fontsize=14)
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()

plt.show()
```





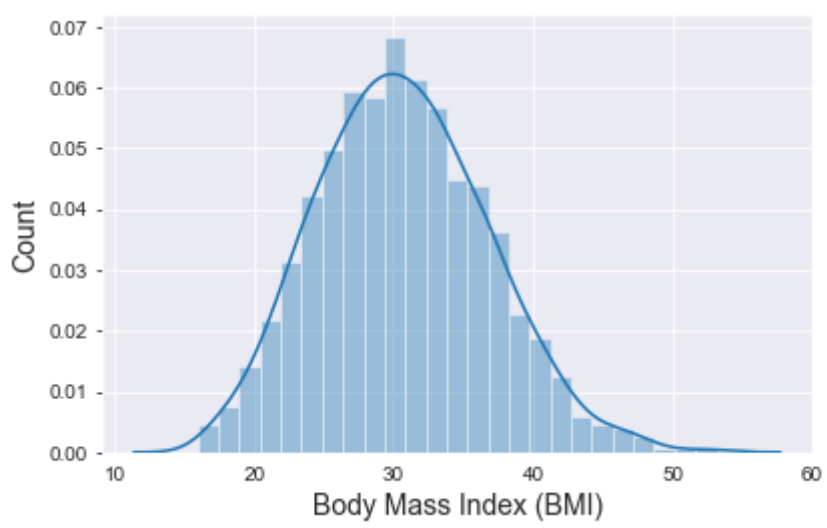
Age v/s charges [by boxenplot representation]

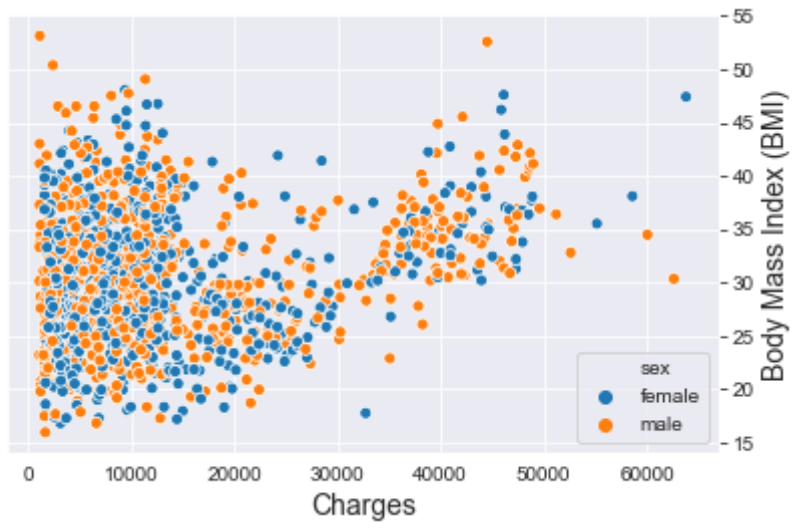
In [35]:
```
f, axe = plt.subplots(1,1,figsize=(20,4))
sns.boxenplot(x = 'age', y = 'charges', data = dataset, ax = axe)
axe.set_xlabel('Age', fontsize=14)
axe.set_ylabel('Charges', fontsize=14)
plt.show()
```



BMI v/s Count and Charge v/s BMI :

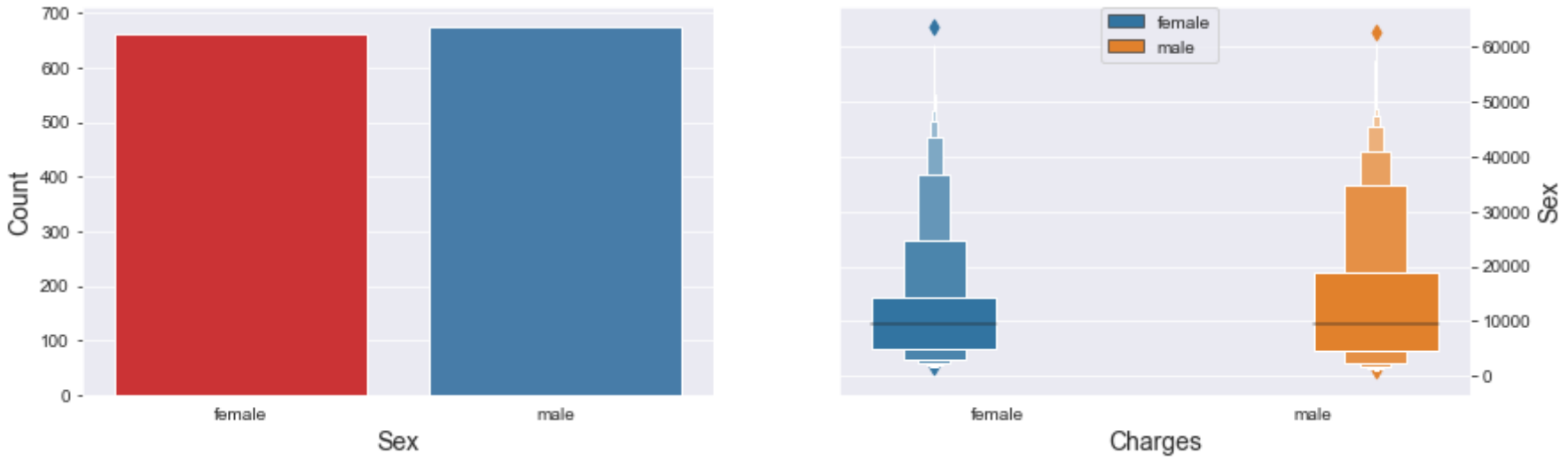
In [36]:
```
f, axes = plt.subplots(1,2,figsize=(14,4))

sns.distplot(dataset['bmi'], ax = axes[0])
axes[0].set_xlabel('Body Mass Index (BMI)', fontsize=14)
axes[0].set_ylabel('Count', fontsize=14)
axes[0].yaxis.tick_left()

sns.scatterplot(x = 'charges', y = 'bmi', data = dataset, hue = 'sex',ax = axes[1])
axes[1].set_xlabel('Charges', fontsize=14)
axes[1].set_ylabel('Body Mass Index (BMI)', fontsize=14)
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()

plt.show()
```





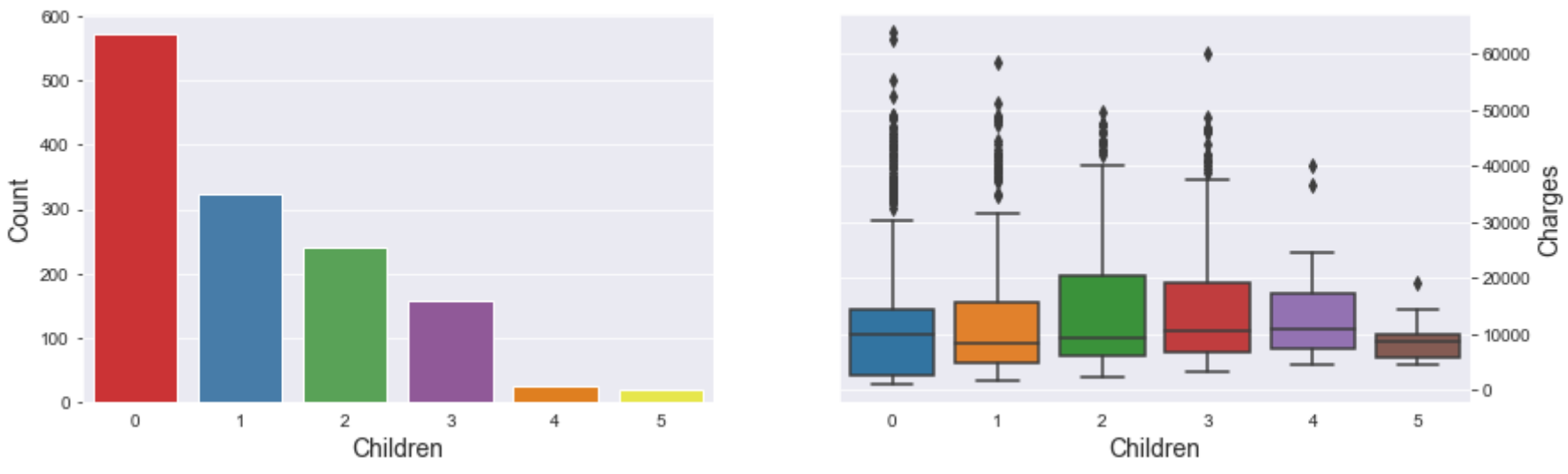
Gender Distribution and Gender v/s Charges :

In [37]:
```
sex_list = Counter(dataset['sex'])
labels = sex_list.keys()
sizes = sex_list.values()

f, axes = plt.subplots(1,2,figsize=(14,4))

sns.countplot(dataset['sex'], ax = axes[0], palette="Set1")
axes[0].set_xlabel('Sex', fontsize=14)
axes[0].set_ylabel('Count', fontsize=14)
axes[0].yaxis.tick_left()

sns.boxenplot(x = 'sex', y = 'charges', data = dataset, hue = 'sex', ax = axes[1])
axes[1].set_xlabel('Charges', fontsize=14)
axes[1].set_ylabel('Sex', fontsize=14)
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()
axes[1].legend(bbox_to_anchor=(0.6,1), loc=1, borderaxespad=0.)

plt.show()
```



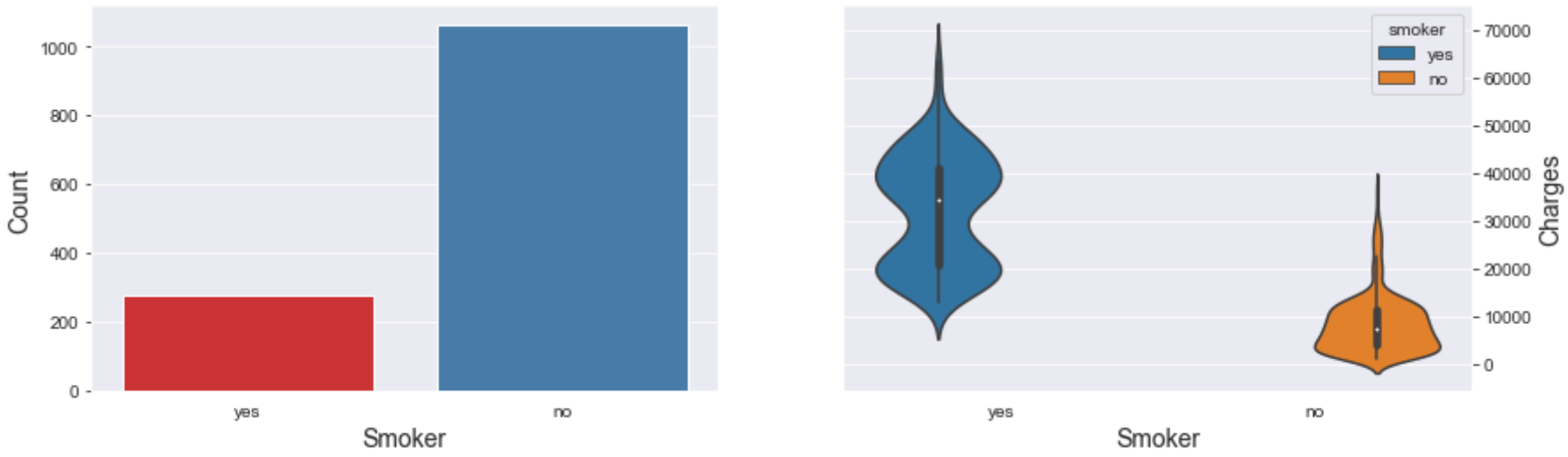
Smokers having Children :

In [73]:
```
children_list = Counter(dataset['children'])
labels = children_list.keys()
sizes = children_list.values()

f, axes = plt.subplots(1,2,figsize=(14,4))

sns.countplot(dataset['children'], ax = axes[0], palette="Set1")
axes[0].set_xlabel('Children', fontsize=14)
axes[0].set_ylabel('Count', fontsize=14)
axes[0].yaxis.tick_left()

sns.boxplot(x = 'children', y = 'charges', data = dataset, ax = axes[1])
axes[1].set_xlabel('Children', fontsize=14)
axes[1].set_ylabel('Charges', fontsize=14)
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()

plt.show()
```



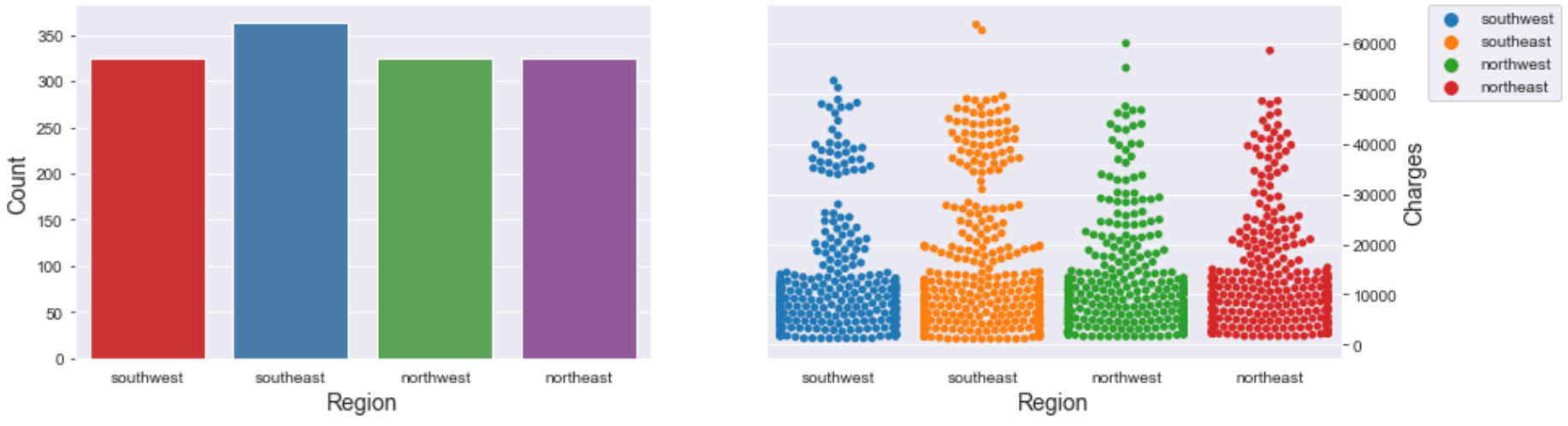
Smokers v/s Charges

In [39]:
```
smoker_list = Counter(dataset['smoker'])
labels = smoker_list.keys()
sizes = smoker_list.values()

f, axes = plt.subplots(1,2,figsize=(14,4))

sns.countplot(dataset['smoker'], ax = axes[0], palette="Set1")
axes[0].set_xlabel('Smoker', fontsize=14)
axes[0].set_ylabel('Count', fontsize=14)
axes[0].yaxis.tick_left()

sns.violinplot(x = 'smoker', y = 'charges', data = dataset, hue = 'smoker', ax = axes[1])
axes[1].set_xlabel('Smoker', fontsize=14)
axes[1].set_ylabel('Charges', fontsize=14)
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()

plt.show()
```



Region based Smokers analysis :

In [40]:
```
region_list = Counter(dataset['region'])
labels = region_list.keys()
sizes = region_list.values()

f, axes = plt.subplots(1,2,figsize=(14,4))

sns.countplot(dataset['region'], ax = axes[0], palette="Set1")
axes[0].set_xlabel('Region', fontsize=14)
axes[0].set_ylabel('Count', fontsize=14)
axes[0].yaxis.tick_left()

sns.swarmplot(x = 'region', y = 'charges', data = dataset, hue = 'region', ax = axes[1])
axes[1].set_xlabel('Region', fontsize=14)
axes[1].set_ylabel('Charges', fontsize=14)
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()
axes[1].legend(bbox_to_anchor=(1.15, 1), loc=2, borderaxespad=0.)

plt.show()
```



Creating dummy datasets and train and test the model according to the dataset and then predict the issues.

In [41]:
```
dataset = pd.get_dummies(dataset)
dataset.head()
```

Out[41]:

	age	bmi	children	charges	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwest	region_southeast	region
0	19	27.900	0	16884.92400	1	0	0	1	0	0	0	
1	18	33.770	1	1725.55230	0	1	1	0	0	0	1	
2	28	33.000	3	4449.46200	0	1	1	0	0	0	1	
3	33	22.705	0	21984.47061	0	1	1	0	0	1	0	
4	32	28.880	0	3866.85520	0	1	1	0	0	1	0	

In [42]:
```
X = dataset.drop('charges', axis = 1).values
y = dataset['charges'].values.reshape(-1,1)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)
```

In [43]:
```
print("Shape of X_train: ",X_train.shape)
print("Shape of X_test: ", X_test.shape)
print("Shape of y_train: ",y_train.shape)
print("Shape of y_test",y_test.shape)
```

```
Shape of X_train:  (1003, 11)
Shape of X_test:  (335, 11)
Shape of y_train:  (1003, 1)
Shape of y_test (335, 1)
```

Basic Linear Regression Model :

In [44]:
```
# Creating the linear regressor
regressor_linear = LinearRegression()
regressor_linear.fit(X_train, y_train)
```

Out[44]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [45]:
```
# Predicting Cross Validation Score the Test set results
cv_linear = cross_val_score(estimator = regressor_linear, X = X, y = y, cv = 10)

# Predicting R2 Score the Train set results
y_pred_linear_train = regressor_linear.predict(X_train)
r2_score_linear_train = r2_score(y_train, y_pred_linear_train)

# Predicting R2 Score the Test set results
y_pred_linear_test = regressor_linear.predict(X_test)
r2_score_linear_test = r2_score(y_test, y_pred_linear_test)

# Predicting RMSE the Test set results
rmse_linear = (np.sqrt(mean_squared_error(y_test, y_pred_linear_test)))
print("CV: ", cv_linear.mean())
print('R2_score (train): ', r2_score_linear_train)
print('R2_score (test): ', r2_score_linear_test)
print("RMSE: ", rmse_linear)
```

```
CV:  0.7445006998667603
R2_score (train):  0.7449555328228536
R2_score (test):  0.7672642952734356
RMSE:  5926.023602394469
```

Polynomial Regression - 2nd degree :

In [46]:
```
# Creating the polynomial features and regressor
poly_reg = PolynomialFeatures(degree = 2)
X_poly = poly_reg.fit_transform(X)
X_train_poly = poly_reg.fit_transform(X_train)
poly_reg.fit(X_train_poly, y_train)

regressor_poly2 = LinearRegression()
regressor_poly2.fit(X_train_poly, y_train)
```

Out[46]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [47]:
```
# Predicting Cross Validation Score the Test set results
cv_poly2 = cross_val_score(estimator = regressor_poly2, X = X_poly, y = y, cv = 10)

# Predicting R2 Score the Train set results
y_pred_poly2_train = regressor_poly2.predict(poly_reg.fit_transform(X_train))
r2_score_poly2_train = r2_score(y_train, y_pred_poly2_train)

# Predicting R2 Score the Test set results
y_pred_poly2_test = regressor_poly2.predict(poly_reg.fit_transform(X_test))
r2_score_poly2_test = r2_score(y_test, y_pred_poly2_test)

# Predicting RMSE the Test set results
rmse_poly2 = (np.sqrt(mean_squared_error(y_test, y_pred_poly2_test)))
print('CV: ', cv_poly2.mean())
print('R2_score (train): ', r2_score_poly2_train)
print('R2_score (test): ', r2_score_poly2_test)
print("RMSE: ", rmse_poly2)
```

```
CV:  0.7626012887082249
R2_score (train):  0.8395580612771243
R2_score (test):  0.8526214897539302
RMSE:  4715.730206340301
```

Ridge Regression :

In [48]:
```
steps = [
    ('scalar', StandardScaler()),
    ('poly', PolynomialFeatures(degree=2)),
    ('model', Ridge())
]

ridge_pipe = Pipeline(steps)
```

In [49]:
```
# step 1: alpha:[200, 230, 250,265, 270, 275, 290, 300, 500] -> 200
# step 2: alpha:[10,50,100,150,200] -> 50
# step 3: alpha: np.arange(30, 75, 1) -> 43

parameters =  {  'model__alpha' : [43],
                 'model__fit_intercept' : [True],
                 'model__tol' : [0.0001],
                 'model__solver' : ['auto'],
                'model__random_state': [42]
}
regressor_ridge = GridSearchCV(ridge_pipe, parameters, iid=False, cv=10)
regressor_ridge = regressor_ridge.fit(X, y.ravel())
```

In [50]:
```
print(regressor_ridge.best_score_)
print(regressor_ridge.best_params_)
```

```
0.8364285533590552
{'model__alpha': 43, 'model__fit_intercept': True, 'model__random_state': 42, 'model__solver': 'auto', 'model__tol': 0.
0001}
```

In [51]:
```
# Predicting Cross Validation Score the Test set results
cv_ridge = regressor_ridge.best_score_

# Predicting R2 Score the Test set results
y_pred_ridge_train = regressor_ridge.predict(X_train)
r2_score_ridge_train = r2_score(y_train, y_pred_ridge_train)

# Predicting R2 Score the Test set results
y_pred_ridge_test = regressor_ridge.predict(X_test)
r2_score_ridge_test = r2_score(y_test, y_pred_ridge_test)

# Predicting RMSE the Test set results
rmse_ridge = (np.sqrt(mean_squared_error(y_test, y_pred_ridge_test)))
print('CV: ', cv_ridge.mean())
print('R2_score (train): ', r2_score_ridge_train)
print('R2_score (test): ', r2_score_ridge_test)
print("RMSE: ", rmse_ridge)
```

```
CV:  0.8364285533590552
R2_score (train):  0.8427507255457396
R2_score (test):  0.8615641628327121
RMSE:  4570.420248306232
```

Lasso Regression :

In [52]:
```
steps = [
    ('scalar', StandardScaler()),
    ('poly', PolynomialFeatures(degree=2)),
    ('model', Lasso())
]

lasso_pipe = Pipeline(steps)
```

In [53]:
```
# Applying Grid Search to find the best model and the best parameters
# step 1: alpha:np.arange(0.01, 1, 0.005) -> 0.9949

parameters =  {  'model__alpha' : [0.9949],
                 'model__fit_intercept' : [True],
                 'model__tol' : [0.0001],
                 'model__max_iter' : [5000],
                'model__random_state': [42]
}
regressor_lasso = GridSearchCV(lasso_pipe, parameters, iid=False, cv=10, n_jobs = -1, verbose = 4)
regressor_lasso = regressor_lasso.fit(X, y.ravel())
```

```
Fitting 10 folds for each of 1 candidates, totalling 10 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    1.5s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    1.5s finished
```

In [54]:
```
# Predicting Cross Validation Score
cv_lasso = regressor_lasso.best_score_

# Predicting R2 Score the Test set results
y_pred_lasso_train = regressor_lasso.predict(X_train)
r2_score_lasso_train = r2_score(y_train, y_pred_lasso_train)

# Predicting R2 Score the Test set results
y_pred_lasso_test = regressor_lasso.predict(X_test)
r2_score_lasso_test = r2_score(y_test, y_pred_lasso_test)

# Predicting RMSE the Test set results
rmse_lasso = (np.sqrt(mean_squared_error(y_test, y_pred_lasso_test)))
print('CV: ', cv_lasso.mean())
print('R2_score (train): ', r2_score_lasso_train)
print('R2_score (test): ', r2_score_lasso_test)
print("RMSE: ", rmse_lasso)
```

```
CV:  0.8362960067859149
R2_score (train):  0.8429615732935353
R2_score (test):  0.8615998023225331
RMSE:  4569.83189656797
```

Support Vector Regression Model :

In [55]:
```
# Feature Scaling
sc_X = StandardScaler()
sc_y = StandardScaler()
X_scaled = sc_X.fit_transform(X)
y_scaled = sc_y.fit_transform(y.reshape(-1,1))
```

In [56]:
```
# Creating the SVR regressor
regressor_svr = SVR()
```

In [57]:
```
# Applying Grid Search to find the best model and the best parameters
parameters =  { 'kernel' : ['rbf', 'sigmoid'],
                'gamma' : [0.001, 0.01, 0.1, 1, 'scale'],
                'tol' : [0.0001],
                'C': [0.001, 0.01, 0.1, 1, 10, 100] }
regressor_svr = GridSearchCV(estimator = regressor_svr,
                             param_grid = parameters,
                             cv = 10,
                             verbose = 4,
                             iid = True,
                             n_jobs = -1)
regressor_svr = regressor_svr.fit(X_scaled, y_scaled.ravel())
```

```
Fitting 10 folds for each of 60 candidates, totalling 600 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:    2.3s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:   12.1s
[Parallel(n_jobs=-1)]: Done 430 tasks      | elapsed:   26.9s
[Parallel(n_jobs=-1)]: Done 600 out of 600 | elapsed:  1.4min finished
```

In [58]:
```
print(regressor_svr.best_params_)
print(regressor_svr.best_score_)
```

```
{'C': 10, 'gamma': 0.01, 'kernel': 'rbf', 'tol': 0.0001}
0.8415317412948141
```

In [59]:
```
# Predicting Cross Validation Score
cv_svr = regressor_svr.best_score_

# Predicting R2 Score the Train set results
y_pred_svr_train = sc_y.inverse_transform(regressor_svr.predict(sc_X.transform(X_train)))
r2_score_svr_train = r2_score(y_train, y_pred_svr_train)

# Predicting R2 Score the Test set results
y_pred_svr_test = sc_y.inverse_transform(regressor_svr.predict(sc_X.transform(X_test)))
r2_score_svr_test = r2_score(y_test, y_pred_svr_test)

# Predicting RMSE the Test set results
rmse_svr = (np.sqrt(mean_squared_error(y_test, y_pred_svr_test)))
print('CV: ', cv_svr.mean())
print('R2_score (train): ', r2_score_svr_train)
print('R2_score (test): ', r2_score_svr_test)
print("RMSE: ", rmse_svr)
```

```
CV:  0.8415317412948141
R2_score (train):  0.84341845998986
R2_score (test):  0.8610473902627702
RMSE:  4578.942852687091
```

Desicion Tree Regression :

In [60]:
```
# Creating the Decision Tree regressor
regressor_dt = DecisionTreeRegressor(random_state = 42)
```

In [61]:
```
# Applying Grid Search to find the best model and the best parameters
parameters = [ { "max_depth": np.arange(1,21),
                 "min_samples_leaf": [1, 5, 10, 20, 50, 100],
                 "min_samples_split": np.arange(2, 11),
                 "criterion": ["mse"],
                 "random_state" : [42]}
             ]
regressor_dt = GridSearchCV(estimator = regressor_dt,
                            param_grid  = parameters,
                            cv = 10,
                            verbose = 4,
                            iid = False,
                            n_jobs = -1)
regressor_dt = regressor_dt.fit(X_scaled, y_scaled)
```

```
Fitting 10 folds for each of 1080 candidates, totalling 10800 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 156 tasks      | elapsed:    0.3s
[Parallel(n_jobs=-1)]: Done 4732 tasks      | elapsed:   10.6s
[Parallel(n_jobs=-1)]: Done 10800 out of 10800 | elapsed:   27.9s finished
```

In [62]:
```
print(regressor_dt.best_params_)
print(regressor_dt.best_score_)

```

```
{'criterion': 'mse', 'max_depth': 5, 'min_samples_leaf': 10, 'min_samples_split': 2, 'random_state': 42}
0.8524782784394495
```

In [63]:
```
# Predicting Cross Validation Score
cv_dt = regressor_dt.best_score_

# Predicting R2 Score the Train set results
y_pred_dt_train = sc_y.inverse_transform(regressor_dt.predict(sc_X.transform(X_train)))
r2_score_dt_train = r2_score(y_train, y_pred_dt_train)

# Predicting R2 Score the Test set results
y_pred_dt_test = sc_y.inverse_transform(regressor_dt.predict(sc_X.transform(X_test)))
r2_score_dt_test = r2_score(y_test, y_pred_dt_test)

# Predicting RMSE the Test set results
rmse_dt = (np.sqrt(mean_squared_error(y_test, y_pred_dt_test)))
print('CV: ', cv_dt.mean())
print('R2_score (train): ', r2_score_dt_train)
print('R2_score (test): ', r2_score_dt_test)
print("RMSE: ", rmse_dt)
```

```
CV:  0.8524782784394495
R2_score (train):  0.8776418101991081
R2_score (test):  0.8776557282375658
RMSE:  4296.587785971407
```

Random Forest Regression :

In [64]:
```
# Creating the Random Forest regressor
regressor_rf = RandomForestRegressor()
```

Because of RandomSearch and GridSeach have took about 20 minutes, I didn't include these steps in the kernel's final form. But you can expand the following 2 cells and view how changed the parameters.

In [65]:
```
# Applying RandomSearch and GridSearch to find the best model and the best parameters
parameters =  { "n_estimators": [1200],
                "max_features": ["auto"],
                "max_depth": [50],
                "min_samples_split": [7],
                "min_samples_leaf": [10],
                "bootstrap": [True],
                "criterion": ["mse"],
                "random_state" : [42] }

regressor_rf = GridSearchCV(estimator = regressor_rf,
                                  param_grid = parameters,
                                  cv = 10,
                                 # verbose = 4,
                                  n_jobs = -1)
regressor_rf = regressor_rf.fit(X_scaled, y.ravel())
```

In [66]:
```
print(regressor_rf.best_params_)
print(regressor_rf.best_score_)
```

```
{'bootstrap': True, 'criterion': 'mse', 'max_depth': 50, 'max_features': 'auto', 'min_samples_leaf': 10, 'min_samples_s
plit': 7, 'n_estimators': 1200, 'random_state': 42}
0.8587777432753294
```

In [67]:
```
from sklearn.metrics import r2_score

# Predicting Cross Validation Score
cv_rf = regressor_rf.best_score_

# Predicting R2 Score the Train set results
y_pred_rf_train = regressor_rf.predict(sc_X.transform(X_train))
r2_score_rf_train = r2_score(y_train, y_pred_rf_train)

# Predicting R2 Score the Test set results
y_pred_rf_test = regressor_rf.predict(sc_X.transform(X_test))
r2_score_rf_test = r2_score(y_test, y_pred_rf_test)

# Predicting RMSE the Test set results
rmse_rf = (np.sqrt(mean_squared_error(y_test, y_pred_rf_test)))
print('CV: ', cv_rf.mean())
print('R2_score (train): ', r2_score_rf_train)
print('R2_score (test): ', r2_score_rf_test)
print("RMSE: ", rmse_rf)
```

```
CV:  0.8587777432753294
R2_score (train):  0.8909741294177032
R2_score (test):  0.8966862253629037
RMSE:  3948.3076377318935
```

Error Measurement :

In [68]:
```
models = [('Linear Regression', rmse_linear, r2_score_linear_train, r2_score_linear_test, cv_linear.mean()),
          ('Polynomial Regression (2nd)', rmse_poly2, r2_score_poly2_train, r2_score_poly2_test, cv_poly2.mean()),
          ('Ridge Regression', rmse_ridge, r2_score_ridge_train, r2_score_ridge_test, cv_ridge.mean()),
          ('Lasso Regression', rmse_lasso, r2_score_lasso_train, r2_score_lasso_test, cv_lasso.mean()),
          ('Support Vector Regression', rmse_svr, r2_score_svr_train, r2_score_svr_test, cv_svr.mean()),
          ('Decision Tree Regression', rmse_dt, r2_score_dt_train, r2_score_dt_test, cv_dt.mean()),
          ('Random Forest Regression', rmse_rf, r2_score_rf_train, r2_score_rf_test, cv_rf.mean())
         ]
```

Checking the scores of RMSE, R2_score (training), R2_score (test) and Cross Validation

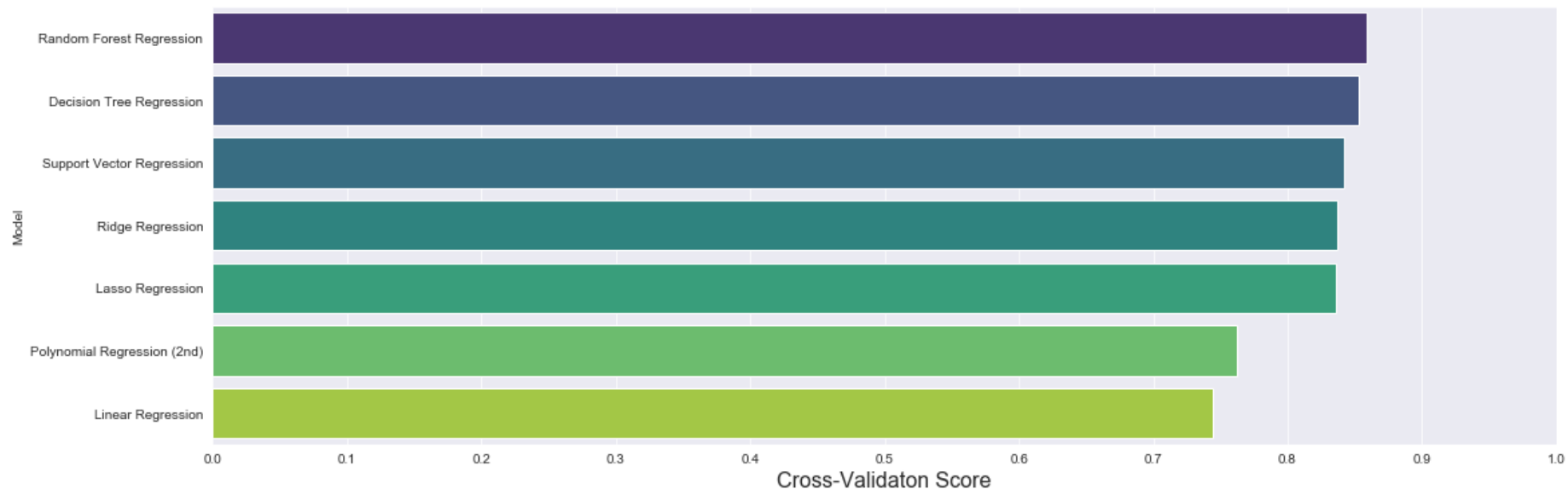
In [69]:
```
predict = pd.DataFrame(data = models, columns=['Model', 'RMSE', 'R2_Score(training)', 'R2_Score(test)', 'Cross-Valid
predict
```

Out[69]:

	Model	RMSE	R2_Score(training)	R2_Score(test)	Cross-Validation
0	Linear Regression	5926.023602	0.744956	0.767264	0.744501
1	Polynomial Regression (2nd)	4715.730206	0.839558	0.852621	0.762601
2	Ridge Regression	4570.420248	0.842751	0.861564	0.836429
3	Lasso Regression	4569.831897	0.842962	0.861600	0.836296
4	Support Vector Regression	4578.942853	0.843418	0.861047	0.841532
5	Decision Tree Regression	4296.587786	0.877642	0.877656	0.852478
6	Random Forest Regression	3948.307638	0.890974	0.896686	0.858778

Which model is better in terms of Cross Validation Score ?

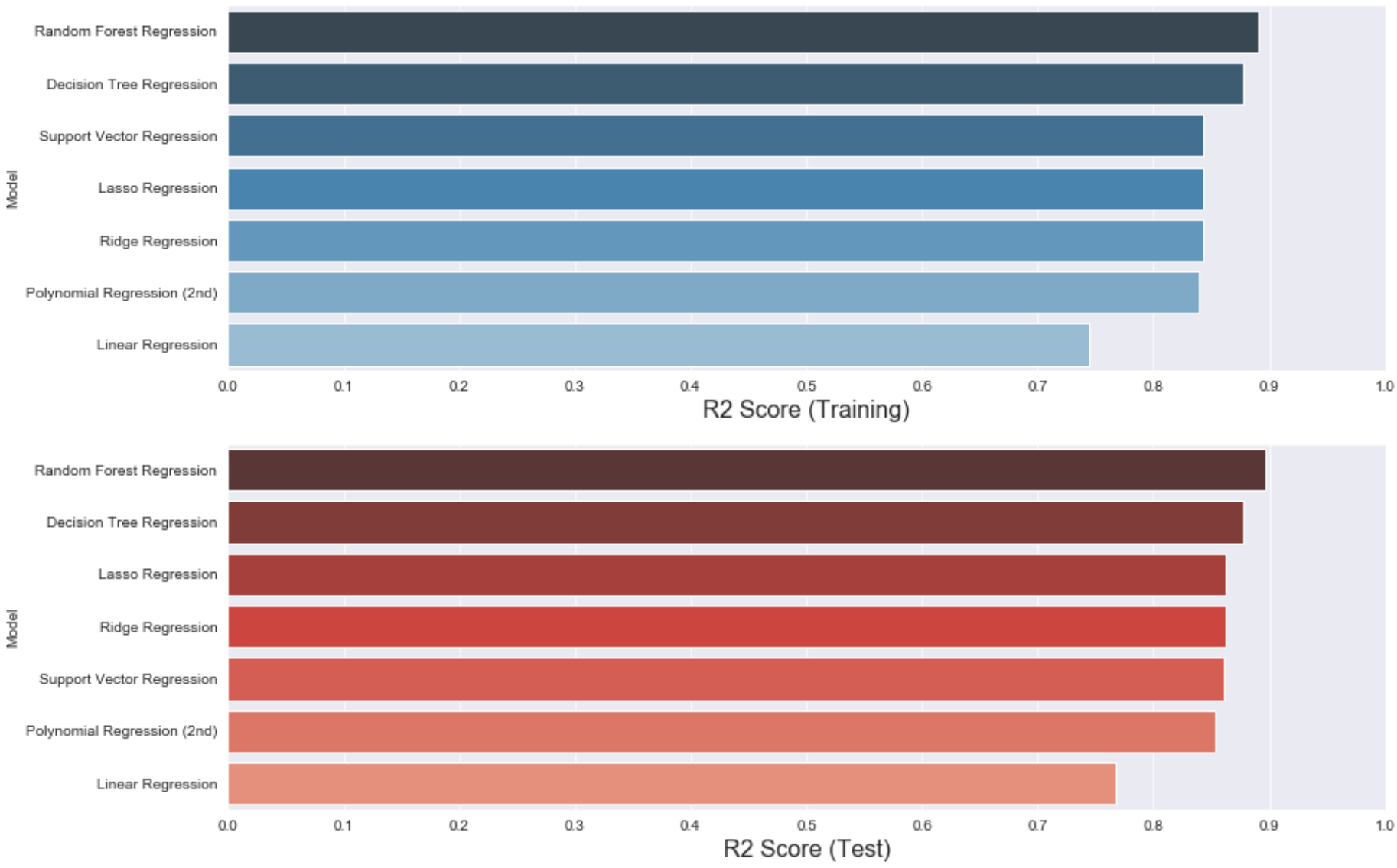
In [70]:
```
f, axe = plt.subplots(1,1, figsize=(18,6))

predict.sort_values(by=['Cross-Validation'], ascending=False, inplace=True)

sns.barplot(x='Cross-Validation', y='Model', data = predict, ax = axe, palette='viridis')
#axes[0].set(xlabel='Region', ylabel='Charges')
axe.set_xlabel('Cross-Validaton Score', size=16)
axe.set_ylabel('Model')
axe.set_xlim(0,1.0)
axe.set_xticks(np.arange(0, 1.1, 0.1))
plt.show()
```



Result :: In terms of Cross Validation scores the "RANDOM FOREST REGRESSION" provides higher accuracy

Which model is better than in terms of R2_score (Training and Test) ?

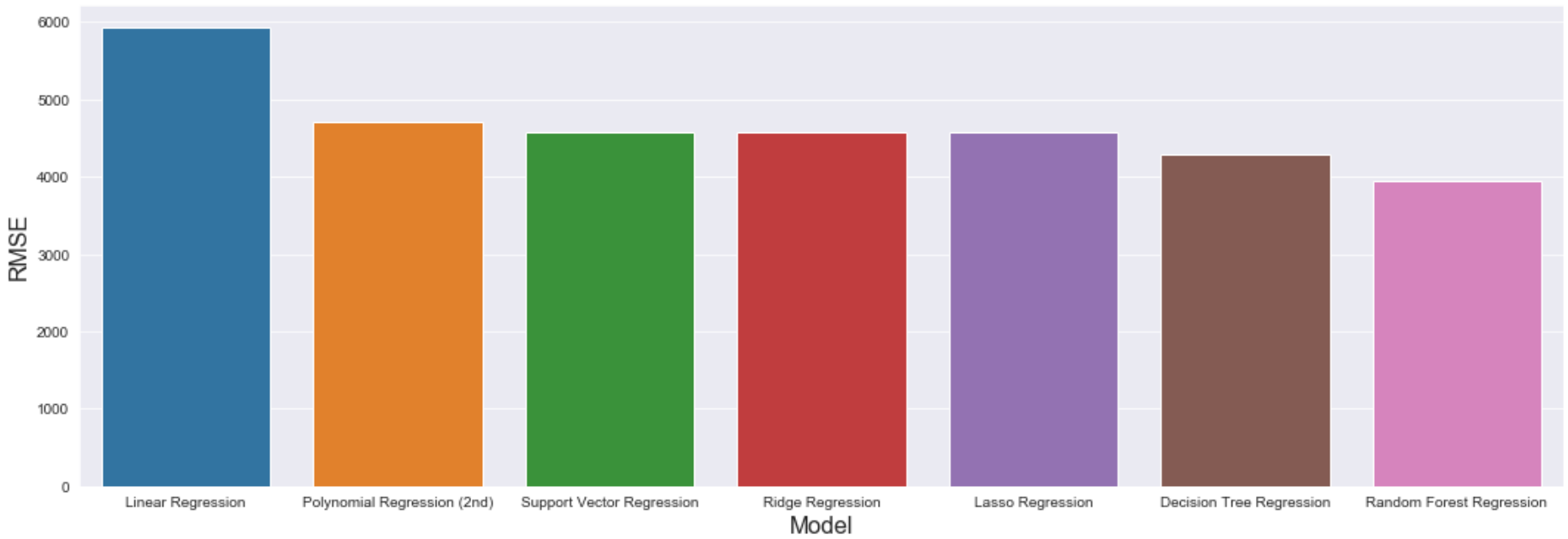
In [71]:
```
f, axes = plt.subplots(2,1, figsize=(14,10))

predict.sort_values(by=['R2_Score(training)'], ascending=False, inplace=True)

sns.barplot(x='R2_Score(training)', y='Model', data = predict, palette='Blues_d', ax = axes[0])
#axes[0].set(xlabel='Region', ylabel='Charges')
axes[0].set_xlabel('R2 Score (Training)', size=16)
axes[0].set_ylabel('Model')
axes[0].set_xlim(0,1.0)
axes[0].set_xticks(np.arange(0, 1.1, 0.1))

predict.sort_values(by=['R2_Score(test)'], ascending=False, inplace=True)

sns.barplot(x='R2_Score(test)', y='Model', data = predict, palette='Reds_d', ax = axes[1])
#axes[0].set(xlabel='Region', ylabel='Charges')
axes[1].set_xlabel('R2 Score (Test)', size=16)
axes[1].set_ylabel('Model')
axes[1].set_xlim(0,1.0)
axes[1].set_xticks(np.arange(0, 1.1, 0.1))

plt.show()
```



Result :: In terms of R2_score (for both the training and test) the "RANDOM FOREST REGRESSION" provides higher accuracy

Which model is better in terms of RMSE ?

In [72]:
```
predict.sort_values(by=['RMSE'], ascending=False, inplace=True)

f, axe = plt.subplots(1,1, figsize=(18,6))
sns.barplot(x='Model', y='RMSE', data=predict, ax = axe)
axe.set_xlabel('Model', size=16)
axe.set_ylabel('RMSE', size=16)

plt.show()
```



Result :: In terms of Cross Validation scores the "LINEAR REGRESSION" provides higher accuracy

Results : Male smokers incurred 11% more medical costs (after adjustment for age, physical functioning status, alcohol consumption, body mass index and average time spent walking) than ‘never smokers’ but for female smokers and never smokers the costs were almost the same. This difference was mainly attributable to increased use of inpatient medical care among smokers, especially in males, where per month cost of inpatient care was 33% higher in smokers. Age-group specific analysis in men showed that excess mortality and excess medical cost ratio for smokers peaked in those aged 60–69 years.

Conclusions : Smokers consume excess medical care. Among the population aged 45 years and over, about 4% of total medical costs were attributable to smoking. To pursue both better health and lower medical costs for the nation, a comprehensive programme to reduce tobacco use is needed.

KEY MESSAGES :

1. The impact of smoking on medical care use was examined in a 30-month prospective population-based cohort study in Japan (N = 43 408).
2. Male smokers incurred 11% more medical costs than ‘never smokers’ but for female smokers and never smokers the costs were almost the same.
3. This difference was mainly attributable to the increased use of inpatient medical care among smokers, especially in males, where per month cost of inpatient care was 33% higher in smokers.

Thank You! Stay Safe! :)