Machine Learning Project

Medical Insurance Cost Prediction
Using different regression models

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted to Dr. Sushma Jain

Name – Abhishek Pandey

Roll No. - 1019180633

Batch - BS3

Index

INTRODUCTION

The goal of this project is to allow a person to get an idea of the amount needed based on their personal health situation. Later they can comply with any health insurance company and their schemes & benefits keeping in mind the predicted amount from our project. This can help a person in focusing more on the health aspect of an insurance rather than the futile part.

Health insurance is a necessity nowadays, and almost every individual is linked with a government or private health insurance company. Factors determining the amount of insurance vary from company to company. Also people in rural areas are unaware of the fact that the government of India provides free health insurance to those below the poverty line. It is a very complex method and some rural people either buy some private health insurance or do not invest money in health insurance at all. Apart from this people can be fooled easily about the amount of the insurance and may unnecessarily buy some expensive health insurance.

Our project does not give the exact amount required for any health insurance company but gives enough idea about the amount associated with an individual for his/her own health insurance.

Prediction is premature and does not comply with any particular company so it must not be only criteria in selection of a health insurance. Early health insurance amount prediction can help in better contemplation of the amount needed.

Data Description

Dataset	insurance.csv
Number of attributes	6
Number of targets	1
Type of target	Numerical
Type of attributes	Numerical & Categorical
Type of Problem	Regression

Code

Data Overview

In [1]:

```
# Importing all the necessary modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
#load and check the shape .
data = pd.read_csv("insurance.csv") data.shape
```

Out[3]:

(1338, 7)

In [4]:

```
data.head()
```

Out[4]:

	age	sex	bmi	children	smoker	region	char
0	19	female	27.900	0	yes	southwest	16884.92
1	18	male	33.770	1	no	southeast	1725.55
2	28	male	33.000	3	no	southeast	4449.46
3	33	male	22.705	0	no	northwest	21984.47
4	32	male	28.880	0	no	northwest	3866.85

In [5]:

```
#check the data
data.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: 57.6+ KB
```

In [6]:

```
#checking the null values
data.isnull().sum()
```

Out[6]:

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

In [7]:

```
data.drop_duplicates(inplace=True)
data.shape
```

Out[7]:

(1337, 7)

In [8]:

```
data.nunique()
```

Out[8]:

age	47
sex	2
bmi	548
children	6
smoker	2
region	4
charges	133
	7

dtype: int64

Data Analysis and Visualization

In [9]:

```
numerical = ['age', 'bmi', 'children']
categorial = ['sex', 'smoker', 'region'] target =
['charges']
```

In [10]:

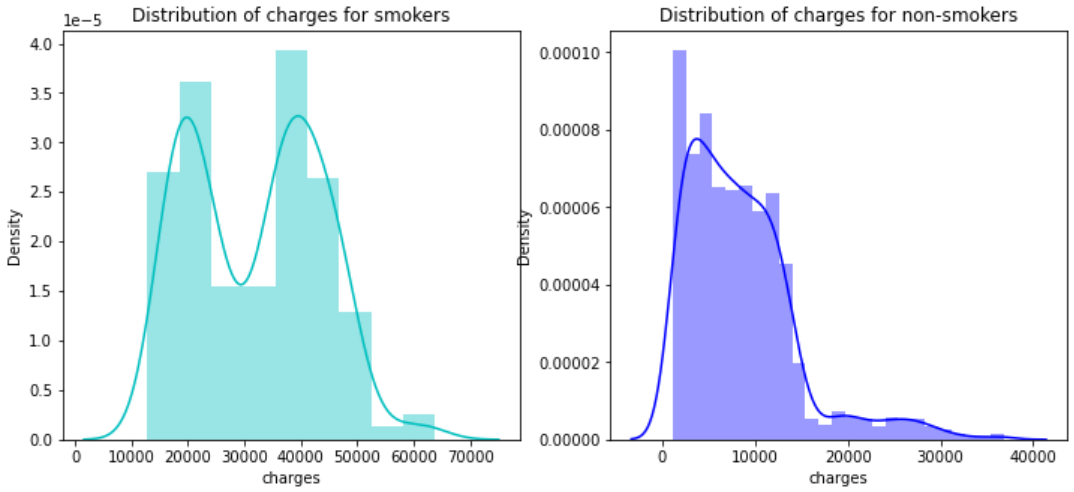
```
f= plt.figure(figsize=(12,5))

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

ax=f.add_subplot(122)
sns.distplot(data[(data.smoker == "no")]['charges'],color='b',a
ax.set_title('Distribution of charges for non-smokers')
```

Out[10]:

Text(0.5, 1.0, 'Distribution of charges for non-s mokers')

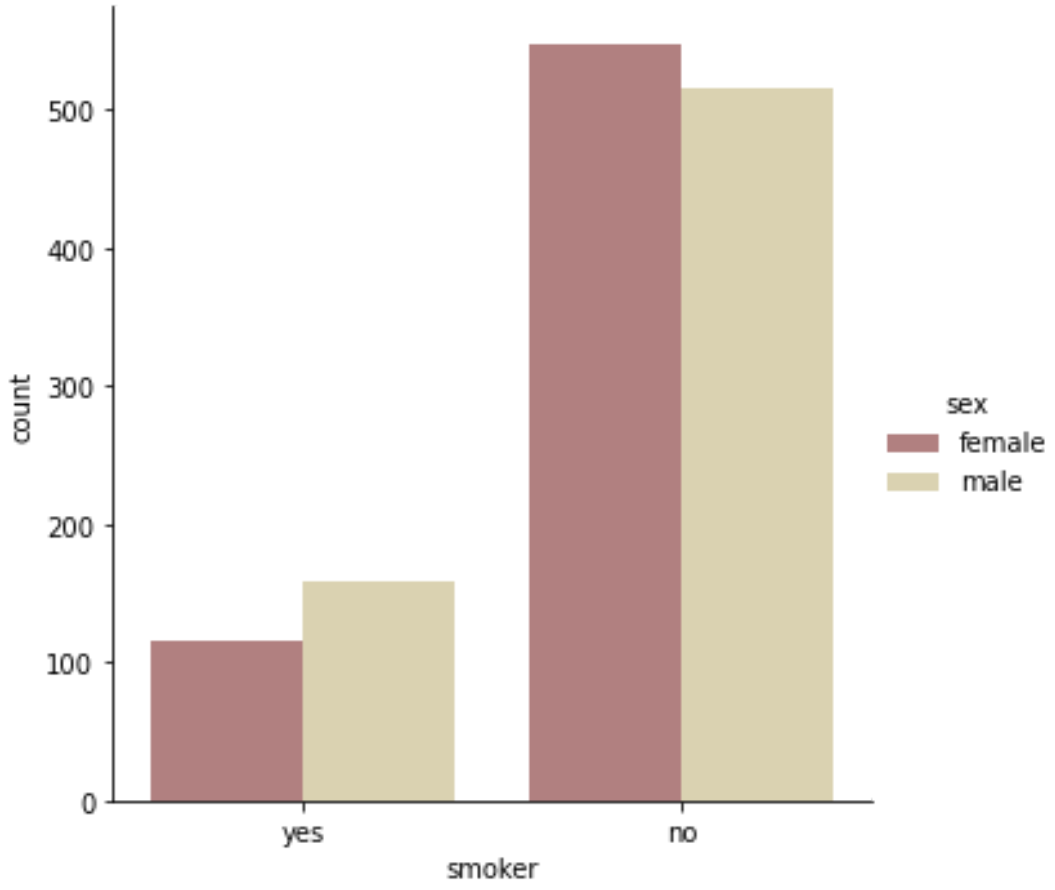


In [11]:

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

Out[11]:

<seaborn.axisgrid.FacetGrid at 0x1c9885b0>

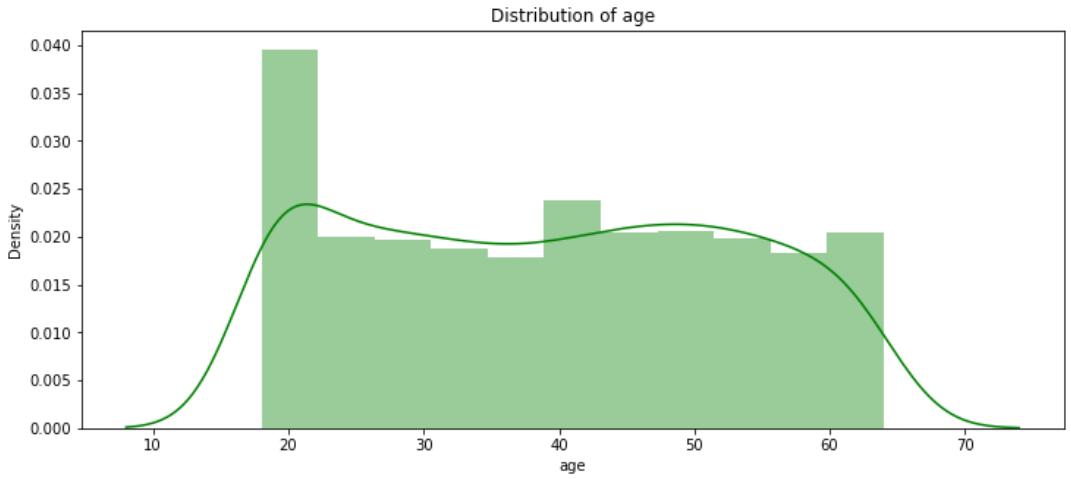


Also we can notice that more male smokers than women smokers. It can be assumed that the total cost of treatment in men will be more than in women, given the impact of smoking.

Now let's pay attention to the age of the patients. First, let's look at how age affects the cost of treatment, and also look at patients of what age more in our data set.

In [12]:

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

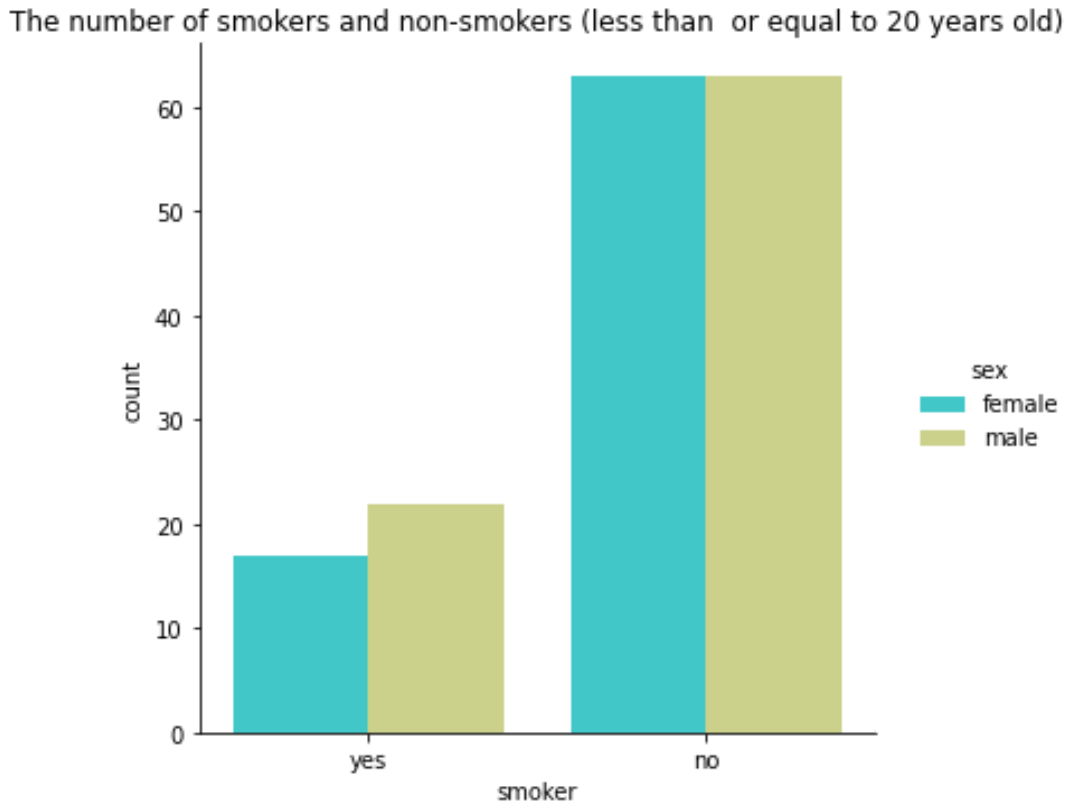


In [13]:

```
sns.catplot(x="smoker",  kind="count",hue  =  'sex',  palette="rain
plt.title("The number of smokers and non-smokers (less than      or
```

Out[13]:

Text(0.5, 1.0, 'The number of smokers and non-smo kers
(less than or equal to 20 years old)')



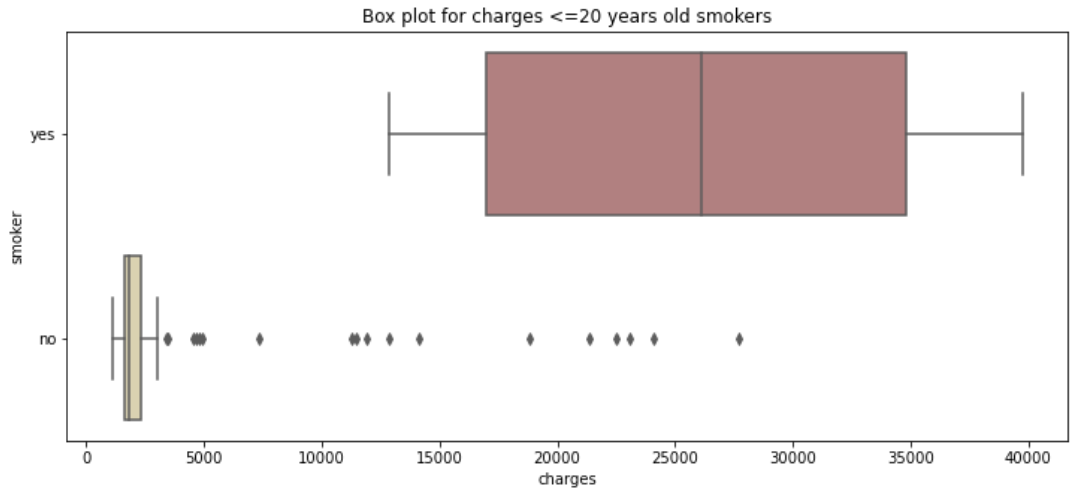
18-20 years old - a very young age. Does smoking affect the cost of treatment at this age?

In [14]:

```
plt.figure(figsize=(12,5))
plt.title("Box plot for charges <=20 years old smokers")
sns.boxplot(y="smoker", x="charges", data = data[(data.age <= 2
```

Out[14]:

<AxesSubplot:title={'center':'Box plot for charge s <=20 years old smokers'}, xlabel='charges', yla bel='smoker'>

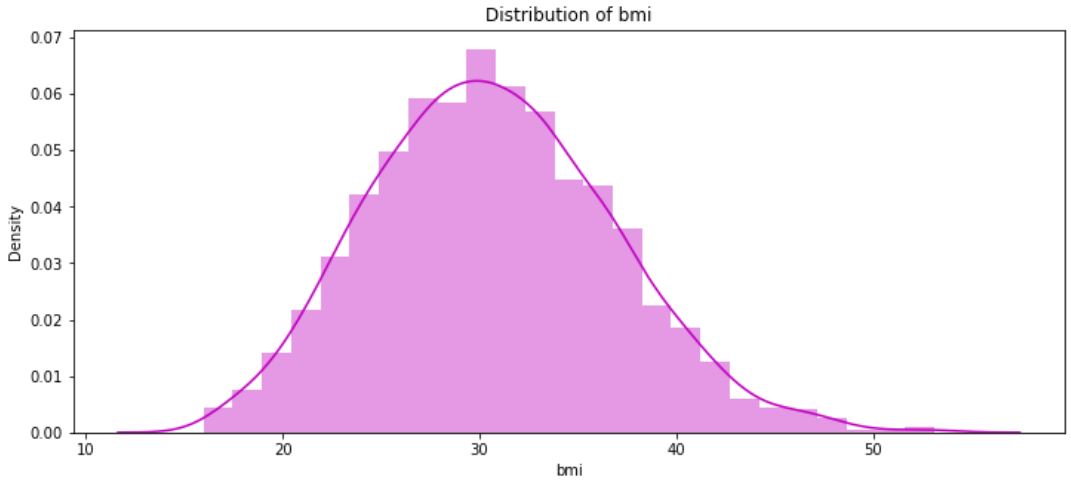


As we can see, even at the age of 20 smokers spend much more on treatment than non-smokers. Among non-smokers we are seeing some " tails." I can assume that this is due to serious diseases or accidents.

Let's pay attention to bmi.

In [15]:

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

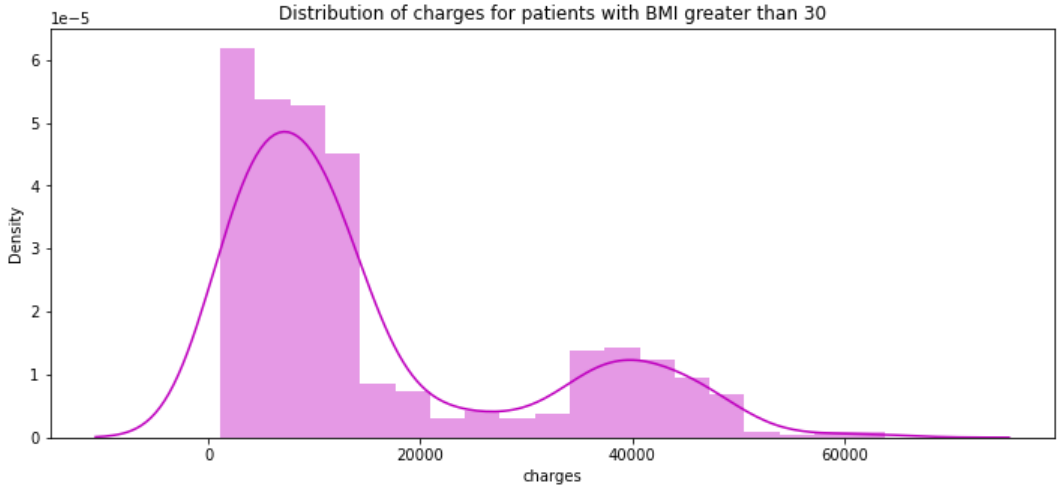


With a value equal to 30 starts obesity.

Let's look at the distribution of costs in patients with BMI greater than 30 and less than 30.

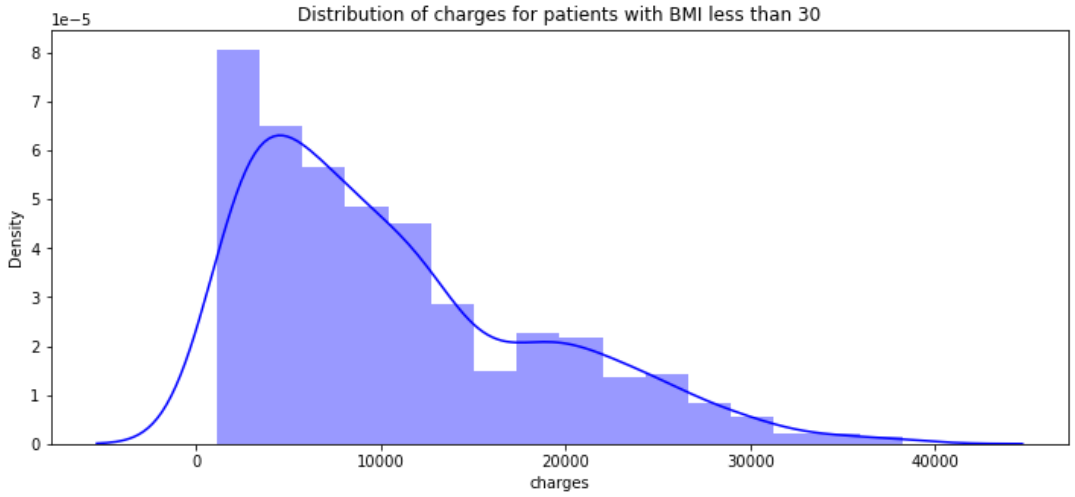
In [16]:

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



In [17]:

```
plt.figure(figsize=(12,5))
plt.title("Distribution of charges for patients with BMI less t ax =
sns.distplot(data[(data.bmi < 30)]['charges'], color = 'b'
```



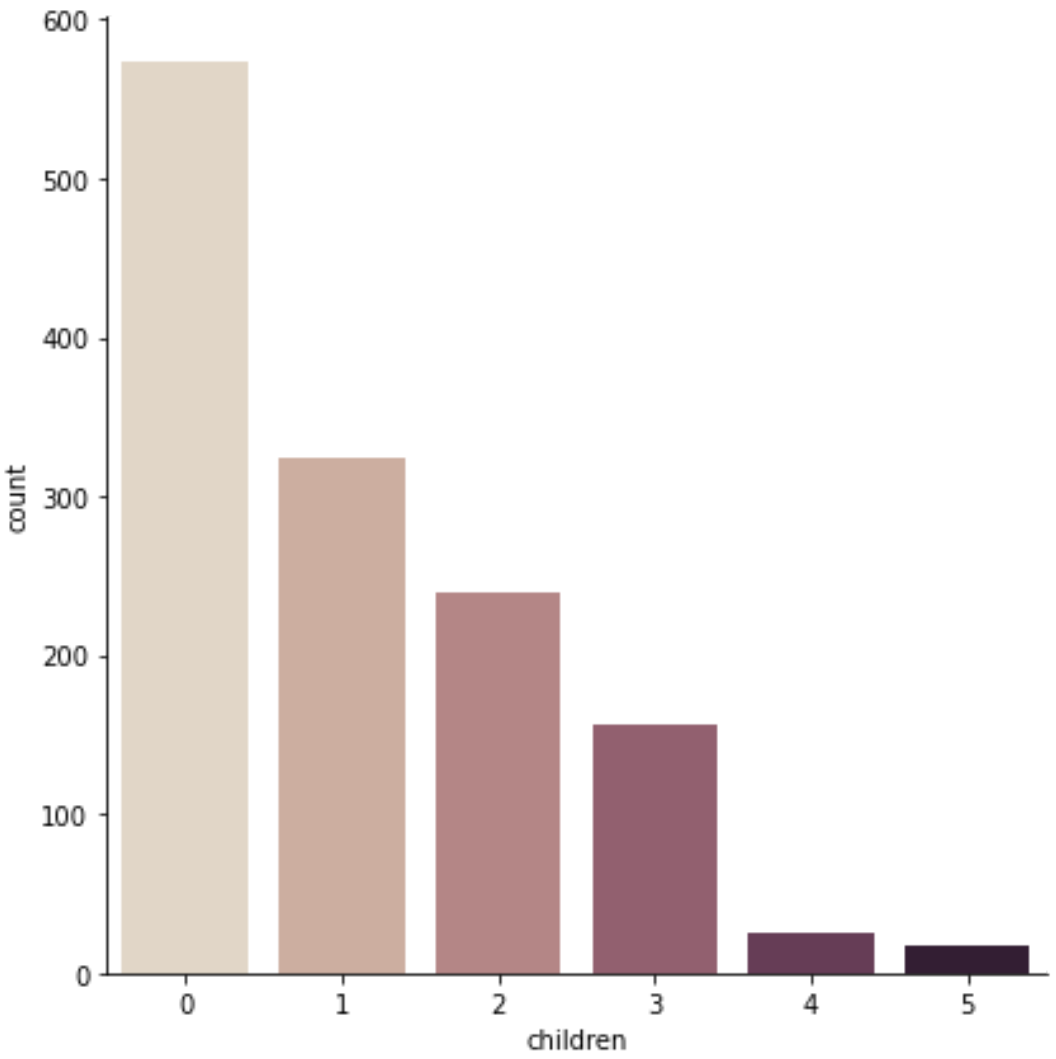
Patients with BMI above 30 spend more on treatment!

Let's pay attention to children. First, let's see how many children our patients have.

In [18]:

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

Out[18]:



<seaborn.axisgrid.FacetGrid at 0x1d5cfac0>

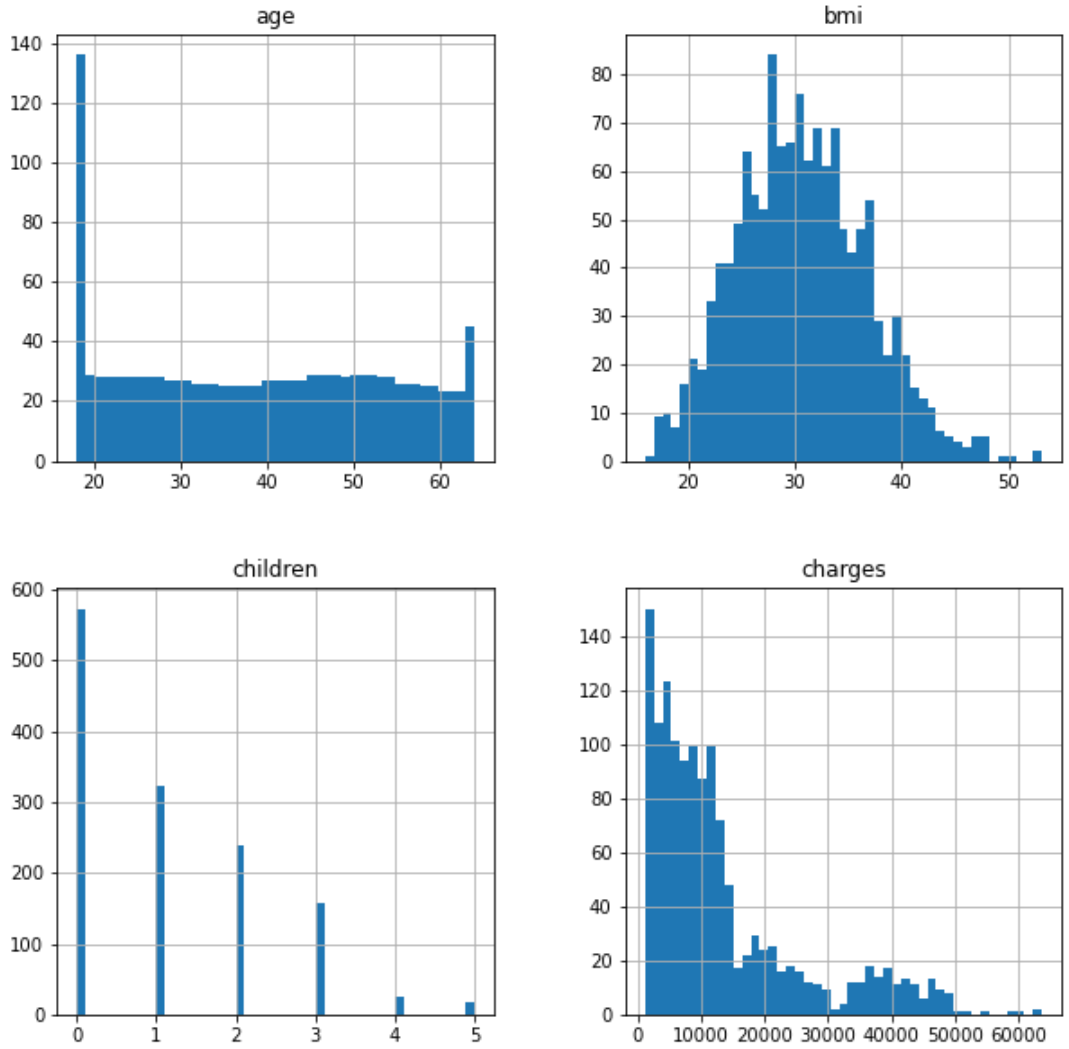
Most patients do not have children.

In [19]:

```
data[numerical + target].hist(bins=45, figsize=(10, 10))
```

Out[19]:

```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'bmi'}>],
       [<AxesSubplot:title={'center':'children'}
>,
        <AxesSubplot:title={'center':'charges'}
>]], dtype=object)
```



The bmi distribution is close to normal.

Data preprocessing

In [20]:

```
# Converting categorical variable into indicator variables.

X=data.iloc[:,0:6].values
Y=data.iloc[:,6].values

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from  sklearn.compose  import  ColumnTransformer

labelencoder_X_1=LabelEncoder()
X[:,1]=labelencoder_X_1.fit_transform(X[:,1])
labelencoder_X_2=LabelEncoder()
X[:,4]=labelencoder_X_2.fit_transform(X[:,4])
labelencoder_X_3=LabelEncoder()
X[:,5]=labelencoder_X_3.fit_transform(X[:,5])
```

In [21]:

```
df = pd.DataFrame(np.c_[X,Y], columns =['age','sex','bmi','chil
plt.figure(figsize=(10, 10))
sns.heatmap(df.corr(), annot=True)
# print(data)
```

Out[21]:

<AxesSubplot:>

In [22]:

```
# using onehot encoder for region and droping one column
onehotencoder_3 = ColumnTransformer([("region", OneHotEncoder() X
=onehotencoder_3.fit_transform(X)
X = X[:, 1:]
```

In [23]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val

X_train, X_test, y_train, y_test = train_test_split(X, Y, rando
```

In [24]:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train) X_test =
scaler.transform(X_test)
```

Now we are going to predict the cost of treatment using various **Machine Learning models**.

In [25]:

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoos
from sklearn.metrics import r2_score,mean_absolute_error,mean_s
from numpy import mean
from numpy import std
```

Regression report

In [26]:

```
R2_Score=[]
MSE=[]
MAE=[]
def regression_report(y_true, y_pred): error =

    y_true - y_pred

    metrics = [
        ('mean absolute error', mean_absolute_error(y_true, y_p ('mean
        squared error', mean_squared_error(y_true, y_pre ('r2 score',
        r2_score(y_true, y_pred)),
    ]
    R2_Score.append(r2_score(y_true, y_pred))
    MSE.append(mean_squared_error(y_true, y_pred))
    MAE.append(mean_absolute_error(y_true, y_pred))
    print('Evaluation Metrics:')
    for metric_name, metric_value in metrics:
        print(f'{metric_name:>25s}: {metric_value: >20.3f}')
```

Linear Regression

In [28]:

```
linear_regression=LinearRegression() linear_regression.fit(X_train,y_train)

# 10 fold cross validation
print("Cross validation score of training data set:",cross_val_

y_pred=linear_regression.predict(X_test)


# regression report
regression_report(y_test,y_pred)
```

```
Cross validation score of training data set: 0.73
50589896702482
Evaluation Metrics:
        mean absolute error:          4396.031
         mean squared error:      41546216.661
                   r2 score:             0.753
```

Decison tree regression

In [29]:

```
decision_tree=DecisionTreeRegressor(max_leaf_nodes=10)
decision_tree.fit(X_train,y_train)

# 10 fold cross validation
print("Cross validation score of training data set:",cross_val_

y_pred=decision_tree.predict(X_test)

# regression report
regression_report(y_test,y_pred)
```

```
Cross validation score of training data set: 0.84
43393388641238
Evaluation Metrics:
        mean absolute error:          3102.589
         mean squared error:      27730305.438
                   r2 score:             0.835
```

Random Forest Regression

In [30]:

```
random_forest=RandomForestRegressor(max_leaf_nodes=10)
random_forest.fit(X_train,y_train)

# 10 fold cross validation
print("Cross validation score of training data set:",cross_val_

y_pred=random_forest.predict(X_test)

# regression report
regression_report(y_test,y_pred)
```

Cross validation score of training data set: 0.85
30297604577524

Evaluation Metrics:

mean absolute error:	3028.090
mean squared error:	26862310.260
r2 score:	0.840

KNeighborsRegressor

In [31]:

```
knn=KNeighborsRegressor()
knn.fit(X_train,y_train)

# 10 fold cross validation
print("Cross validation score of training data set:",cross_val_

y_pred=knn.predict(X_test)

# regression report
regression_report(y_test,y_pred)
```

Cross validation score of training data set: 0.77
31001542881967

Evaluation Metrics:

mean absolute error:	3695.260
mean squared error:	33728110.656
r2 score:	0.800

Neural Network

In [32]:

```
import keras
from keras.models import Sequential # initialise nerual network
from keras.layers import Dense #          to add diferent layers in NN

# initialising the ANN
model = Sequential()

# Adding the input layer and the first hidden layer
model.add(keras.Input(shape=(8,)))
 model.add(Dense(5,       activation='swish',          kernel_initializer='unif
 model.add(Dense(5,       activation='swish',          kernel_initializer='unif
 model.add(Dense(1,       activation='swish',          kernel_initializer='unif

# compilng the ANN
# from keras import backend as K

from keras import backend as K

def root_mean_squared_error(y_true, y_pred):
                return  K.sqrt(K.mean(K.square(y_pred  -  y_true)))

model.compile(optimizer='rmsprop',  loss=root_mean_squared_error

# Fit the ANN to the training set  #
stochaistic gradient descent
model.fit(X_train,  y_train,  batch_size=1,epochs=100)

# 3- making the prediction and evaluating the model
y_predd=model.predict(X_test)

print("R2 score :--->>",r2_score(y_test,y_predd))
R2_Score.append(r2_score(y_test,  y_predd))
MSE.append(mean_squared_error(y_test,  y_predd))
MAE.append(mean_absolute_error(y_test,  y_predd))
```

```
Epoch 1/100
1069/1069  [==============================]  - 2s
2ms/step - loss: 13080.5986 - mse: 312029056.000 - mae: 13080.5986
Epoch 2/100
1069/1069 [==============================] - 2s
2ms/step - loss: 13059.7441 - mse: 311514656.00 00 - mae:
13059.7441
Epoch 3/100
1069/1069 [==============================] - 2s
2ms/step - loss: 13006.4941 - mse: 310180160.00 00 -
mae: 13006.4941
Epoch 4/100
1069/1069 [==============================] - 2s
2ms/step - loss: 12905.0498 - mse: 307676064.00 00 -
mae: 12905.0498
Epoch 5/100
069/1069 [==============================] - 2s
```

In [33]:

```
col={'R2_score':R2_Score}
print(col)
models=['Linear  Regression','Decision  Tree','Random  Forest','KN
df_R2=pd.DataFrame(data=col,index=models)
df_R2
```

```
{'R2_score': [0.7530385567240128, 0.8351639017002
683,  0.8403234892418667,  0.799511398242944, 0.809
0469834734562]}
```

Out[33]:

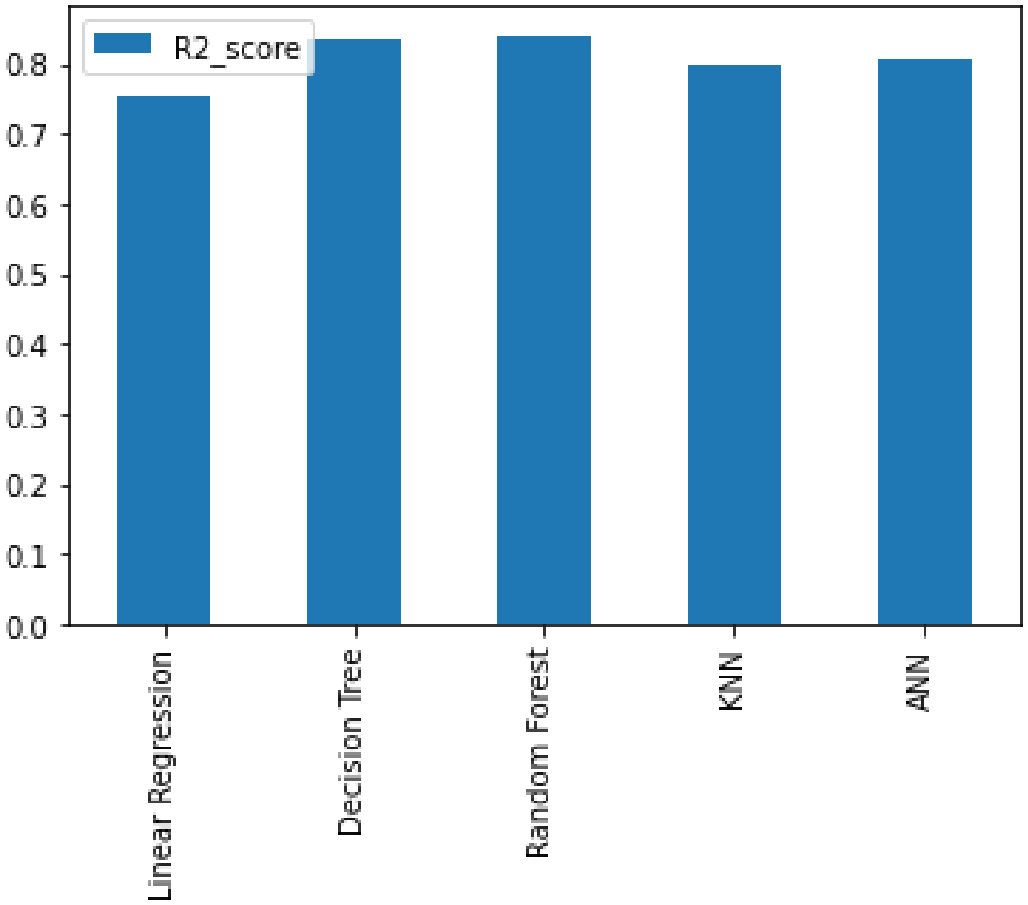
	R2_score
Linear Regression	0.753039
Decision Tree	0.835164
Random Forest	0.840323
KNN	0.799511
ANN	0.809047

In [34]:

```
df_R2.plot(kind='bar')
```

Out[34]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fdb4dec8bd0>



In [35]:

```
col={'MeanAbsoluteError':MAE}
print(col)
models=['Linear  Regression','Decision  Tree','Random  Forest','KN
df_error=pd.DataFrame(data=col,index=models)
df_error
```

{'MeanAbsoluteError': [4396.031406695098, 3102.58
8511366635, 3028.089829580364, 3695.259524910447
5, 2891.9332420990627]}

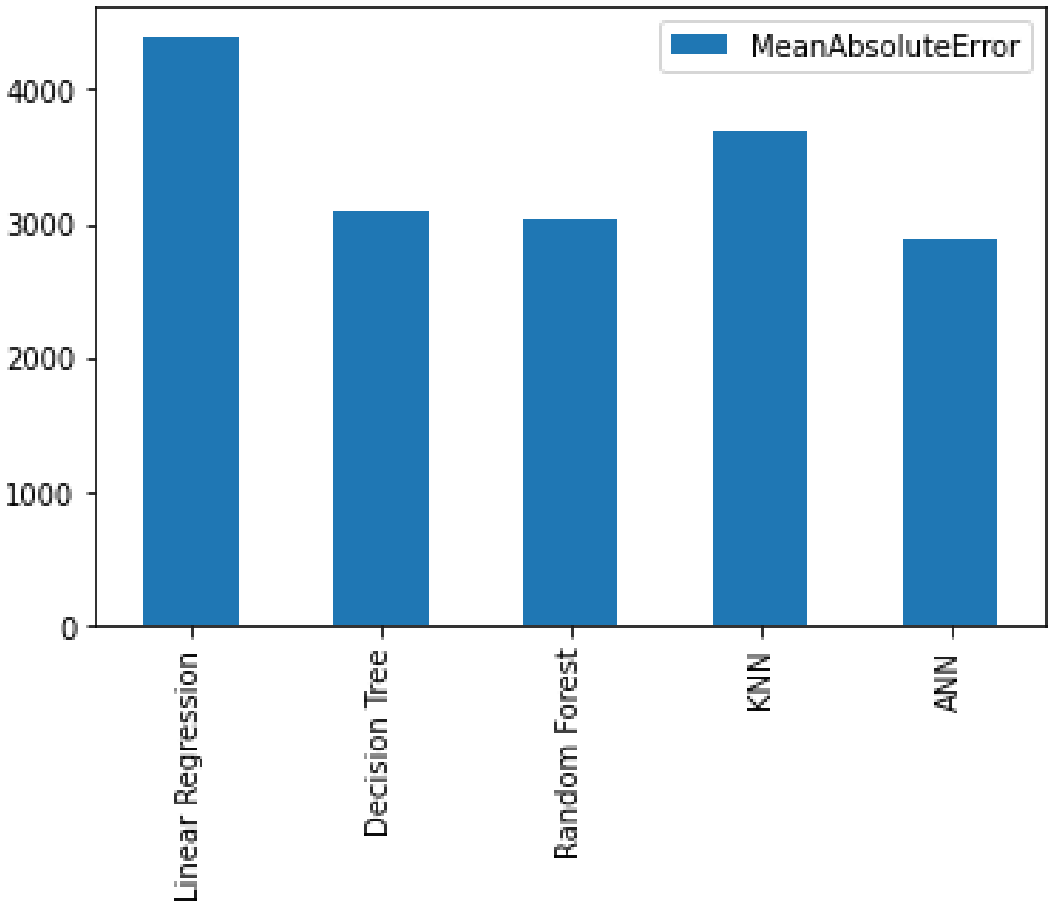
Out[35]:

	MeanAbsoluteError
Linear Regression	4396.031407
Decision Tree	3102.588511
Random Forest	3028.089830
KNN	3695.259525
ANN	2891.933242

In [36]:

```
df_error.plot(kind='bar')
```

Out[36]:



<matplotlib.axes._subplots.AxesSubplot at 0x7fdb4f122190>

Result

Random Forest Regressor is the best model for our model having an R2 score of 0.8403.