In [ ]:
```
# The Contents Of the project
.....................................................................................
.....................................................................................
  #-----Importing Libraries
  #-----Loading data
  #-----Information about data
  #-----Statistics
  #-----Vizualization
  #-----Preprocessing and Explanatory Data Analysis
  #-----Vizualizing the difference between Actual and Predicted prices for Train Dat
  #-----Applying Linear model to Test Data
  #-----Vizualizing the difference between Actual and Predicted prices for Test Data
.....................................................................................
```

Importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Reading and Loading the Dataset

In [2]:
```
import os
```

In [3]:
```
os.getcwd()
```

Out[3]:
```
'C:\\Users\\morem\\Downloads'
```

In [4]:
```
Insurance=pd.read_csv('Insurance.csv')
```

In [5]:
```
Insurance.shape
```

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

Information on Data

In [6]:
```
Insurance.head(7)
```

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

	age	sex	bmi	children	smoker	region	charges
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960

In [7]:
```
Insurance.tail(7)
```

Out[7]:

	age	sex	bmi	children	smoker	region	charges
1331	23	female	33.40	0	no	southwest	10795.93733
1332	52	female	44.70	3	no	southwest	11411.68500
1333	50	male	30.97	3	no	northwest	10600.54830
1334	18	female	31.92	0	no	northeast	2205.98080
1335	18	female	36.85	0	no	southeast	1629.83350
1336	21	female	25.80	0	no	southwest	2007.94500
1337	61	female	29.07	0	yes	northwest	29141.36030

In [8]:
```
Insurance.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
```

Data Analysis

In [9]:
```
#Categorical Features present in the dataset (features with object type are called c
```

In [10]:
```
Insurance.isnull().sum()
```

Out[10]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

Statistics

In [11]:
```
#There are no missing Values in the dataset
```

In [12]:
```
Insurance.describe()
```

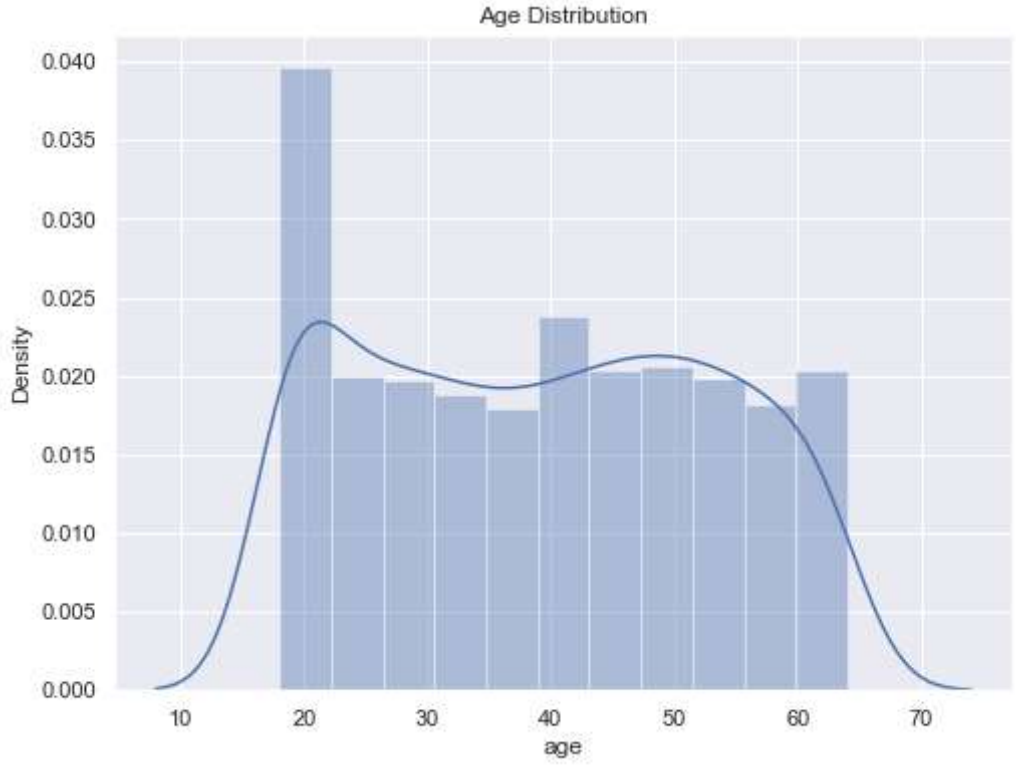
Out[12]:

	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

Visualization

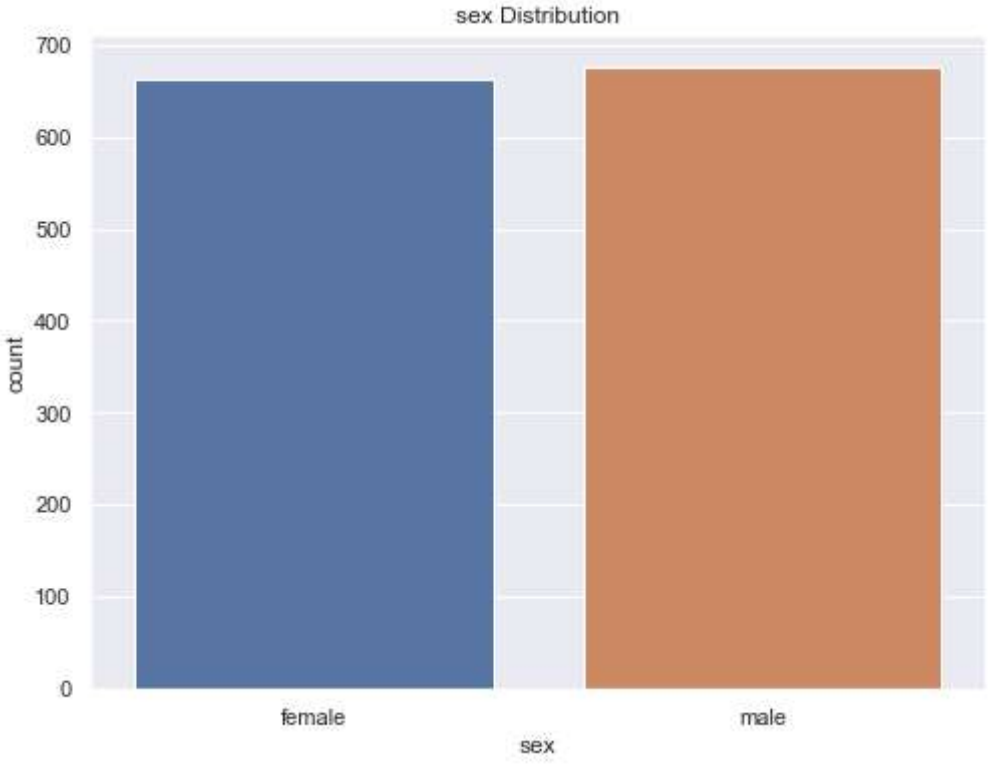
In [13]:
```
#Checking data Distribution in age
```

In [14]:
```
sns.set()
plt.figure(figsize=(8,6))
sns.distplot(Insurance['age'])
plt.title('Age Distribution')
plt.show()
```



In [15]:
```
#Checking data Distribution in gender
```

In [16]:
```
sns.set()
plt.figure(figsize=(8,6))
sns.countplot(x='sex',data=Insurance)
plt.title('sex Distribution')
plt.show()
```

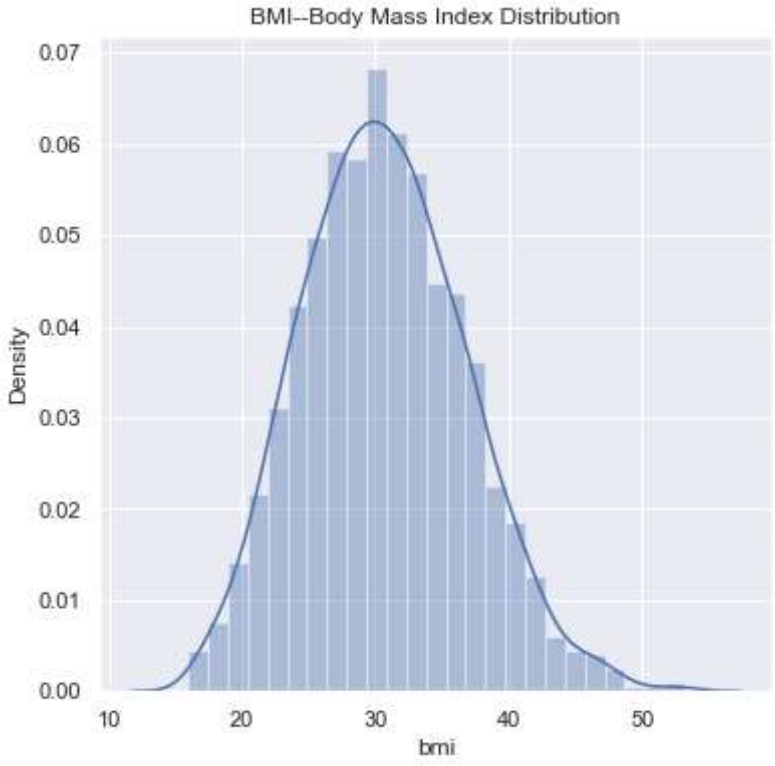


In [17]:
```
Insurance['sex'].value_counts()
```

Out[17]:
```
male      676
female    662
Name: sex, dtype: int64
```

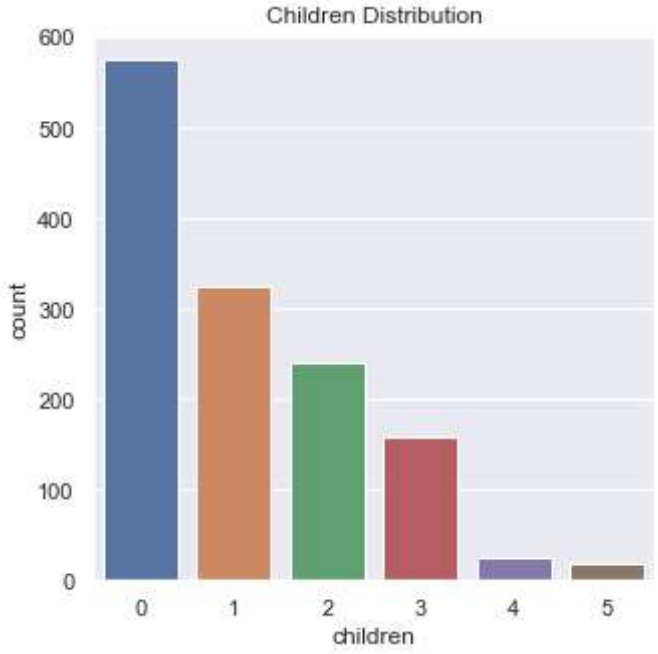
In [18]:
```
#Checking data Distribution in bmi
```

In [19]:
```
sns.set()
plt.figure(figsize=(6,6))
sns.distplot(Insurance['bmi'])
plt.title('BMI--Body Mass Index Distribution')
plt.show()
```



In [20]:
```
# Normal bmi range--->>> 18.5 to 24.9
```

In [21]:
```
sns.set()
plt.figure(figsize=(5,5))
sns.countplot(x='children',data=Insurance)
plt.title('Children Distribution')
plt.show()
```



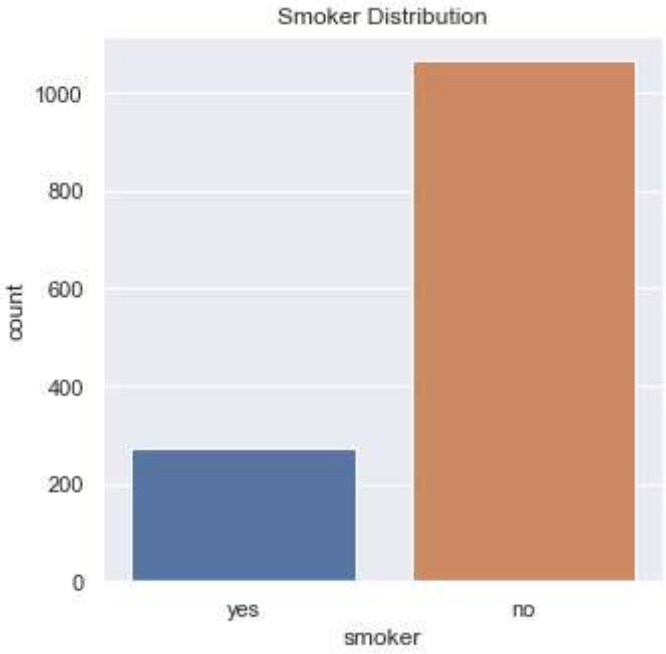
In [22]:
```
Insurance['children'].value_counts()
```

Out[22]:
```
0    574
1    324
2    240
3    157
```

```
4     25
5     18
Name: children, dtype: int64
```

In [23]:

```
plt.figure(figsize=(5,5))
sns.countplot(Insurance['smoker'])
plt.title('Smoker Distribution')
plt.show()
```



In [24]:

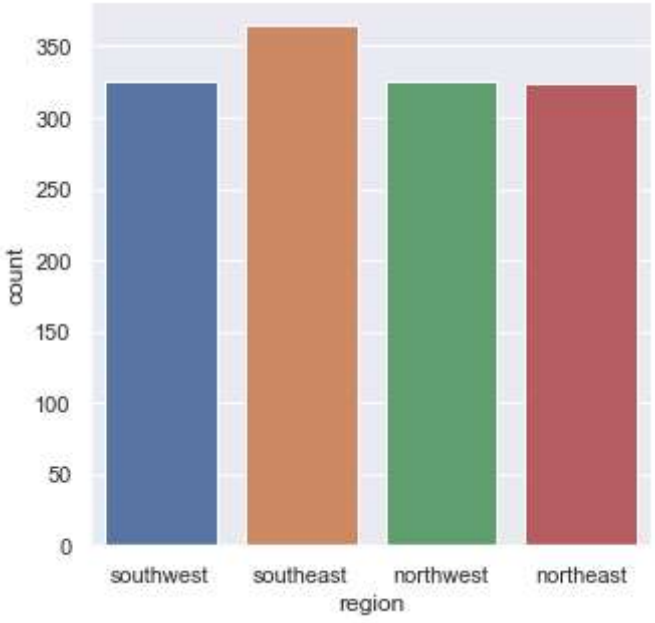
```
Insurance['smoker'].value_counts()
```

Out[24]:

```
no     1064
yes     274
Name: smoker, dtype: int64
```

In [25]:

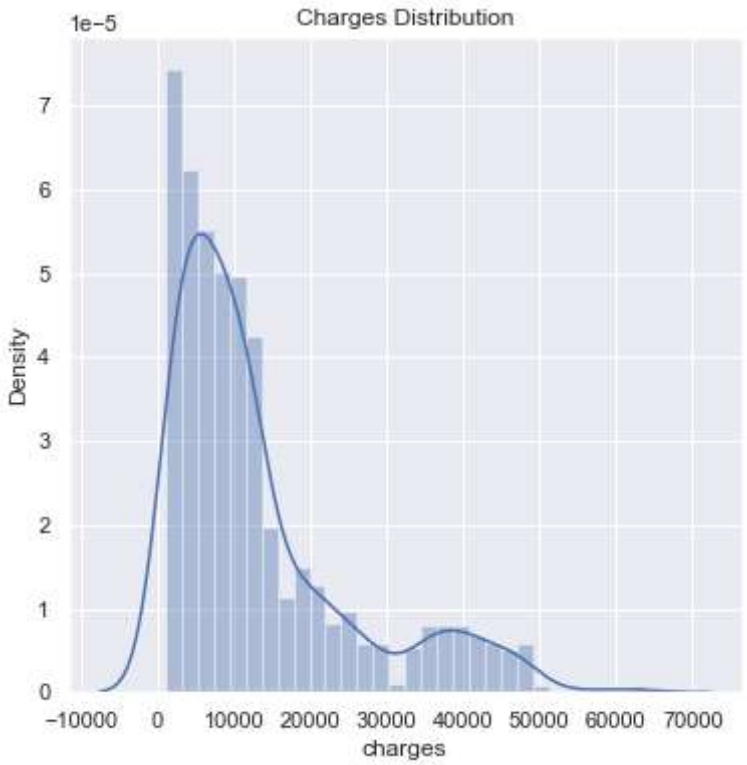
```
sns.set()
plt.figure(figsize=(5,5))
sns.countplot(Insurance['region'])
plt.show()
```



In [26]:
```
Insurance['region'].value_counts()
```

Out[26]:
```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [27]:
```
sns.set()
plt.figure(figsize=(6,6))
sns.distplot(Insurance['charges'])
plt.title('Charges Distribution')
plt.show()
```



Data Pre processing

LABEL ENCODING

In [28]:
```
# #Label Encoding ----Encoding the Categorical data
#which are ---sex,smoker,region
```

In [29]:
```
#Encoding sex column
Insurance.replace({'sex':{'male':0,'female':1}},inplace=True)
```

In [30]:
```
Insurance['sex'].head(5)
```

Out[30]:
```
0    1
1    0
2    0
3    0
```

```
4    0
Name: sex, dtype: int64
```

In [31]:
```
#Encoding smoker column
Insurance.replace({'smoker':{'yes':0,'no':1}},inplace=True)
```

In [32]:
```
Insurance['smoker'].head(5)
```

Out[32]:
```
0    0
1    1
2    1
3    1
4    1
Name: smoker, dtype: int64
```

In [33]:
```
#Encoding region column
Insurance.replace({'region':{'southeast':0,'southwest':1,'northeast':2,'northwest':3
```

In [34]:
```
Insurance['region'].head(5)
```

Out[34]:
```
0    1
1    0
2    0
3    3
4    3
Name: region, dtype: int64
```

In [35]:
```
Insurance
```

Out[35]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	0	1	16884.92400
1	18	0	33.770	1	1	0	1725.55230
2	28	0	33.000	3	1	0	4449.46200
3	33	0	22.705	0	1	3	21984.47061
4	32	0	28.880	0	1	3	3866.85520
...	...	...	...	...	...	...	...
1333	50	0	30.970	3	1	3	10600.54830
1334	18	1	31.920	0	1	2	2205.98080
1335	18	1	36.850	0	1	0	1629.83350
1336	21	1	25.800	0	1	1	2007.94500
1337	61	1	29.070	0	0	3	29141.36030

1338 rows × 7 columns

In [36]:
```
X=Insurance.drop(columns='charges',axis=1)
Y=Insurance['charges']
```

In [37]:
```
print(X)
```

```
      age  sex     bmi  children  smoker  region
0      19    1  27.900         0       0       1
1      18    0  33.770         1       1       0
2      28    0  33.000         3       1       0
3      33    0  22.705         0       1       3
4      32    0  28.880         0       1       3
...   ...  ...     ...       ...     ...     ...
1333   50    0  30.970         3       1       3
1334   18    1  31.920         0       1       2
1335   18    1  36.850         0       1       0
1336   21    1  25.800         0       1       1
1337   61    1  29.070         0       0       3

[1338 rows x 6 columns]
```

In [38]:
```
print(Y)
```

```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
            ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

In [39]:
```
#Splitting the Data into Train and Test Data
from sklearn.model_selection import train_test_split
```

In [40]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=2)
```

In [41]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(1338, 6) (1070, 6) (268, 6)
```

In [42]:
```
#Importing Linear Regression Model --Trainig the Model
```

In [43]:
```
#Linear Regression
from sklearn.linear_model import LinearRegression
```

In [44]:
```
LR=LinearRegression()
```

For Train Data

In [45]:
```
LR.fit(X_train,Y_train)                                    # Fitting the Model
```

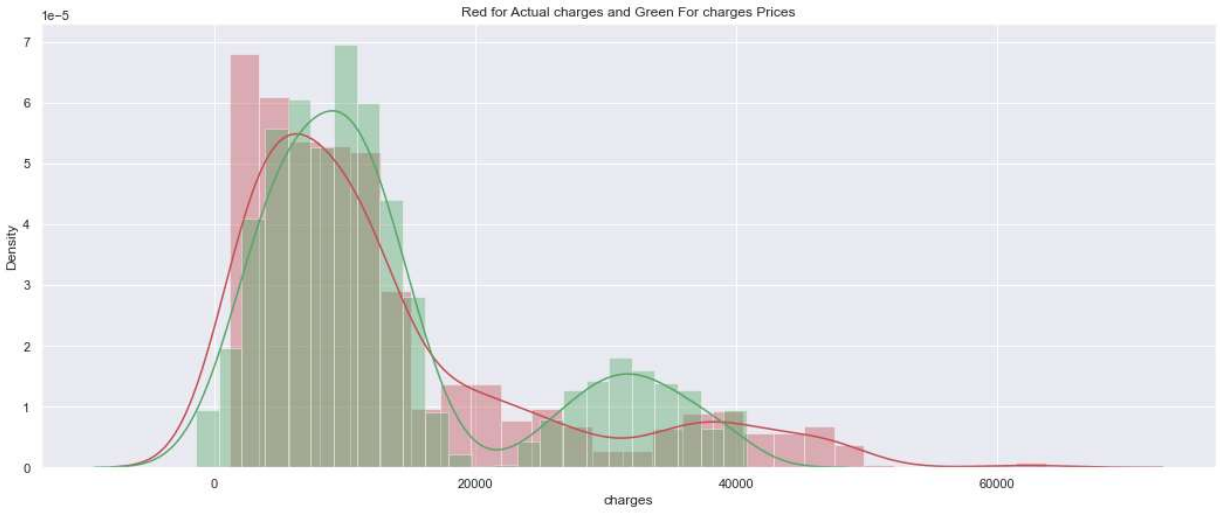
Out[45]:
```
LinearRegression()
```

In [46]:
```
#Model evaluation
```

In [47]:
```
train_data_prediction=LR.predict(X_train)
```

Visualizing difference in actual charges and predicted Charges for Train Data

In [48]:
```
plt.figure(figsize=(18,7))
sns.distplot( Y_train,color='r')
sns.distplot(train_data_prediction,color='g')
plt.title('Red for Actual charges and Green For charges Prices')
plt.show()
```



In [49]:
```
from sklearn import metrics
```

In [50]:
```
print('R^2:',metrics.r2_score(Y_train,train_data_prediction))
print('Adjusted R^2:',1 - (1-metrics.r2_score(Y_train,train_data_prediction))*(len(Y
print('MAE:',metrics.mean_absolute_error(Y_train, train_data_prediction))
print('MSE:',metrics.mean_squared_error(Y_train, train_data_prediction))
print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_train, train_data_prediction)))
```

```
R^2: 0.751505643411174
Adjusted R^2: 0.7501030412102963
MAE: 4150.500304883778
MSE: 36174978.42709207
RMSE: 6014.563860089281
```

For Test Data

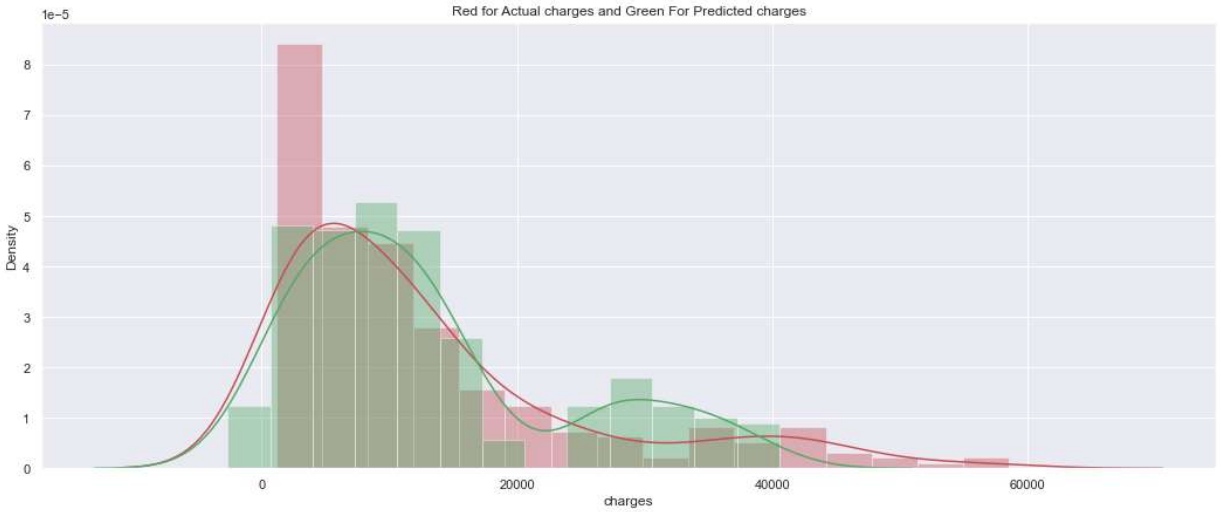
In [51]:
```
LR.fit(X_test,Y_test)
```

Out[51]:
```
LinearRegression()
```

In [52]:
```
test_data_prediction=LR.predict(X_test)
```

Visualizing difference in actual charges and predicted Charges for Test Data

In [53]:
```
plt.figure(figsize=(18,7))
sns.distplot( Y_test,color='r')
sns.distplot(test_data_prediction,color='g')
plt.title('Red for Actual charges and Green For Predicted charges')
plt.show()
```



In [54]:
```
print('R^2:',metrics.r2_score(Y_test,test_data_prediction))
print('Adjusted R^2:',1 - (1-metrics.r2_score(Y_test,test_data_prediction))*(len(Y_t
print('MAE:',metrics.mean_absolute_error(Y_test, test_data_prediction))
print('MSE:',metrics.mean_squared_error(Y_test, test_data_prediction))
print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test, test_data_prediction)))
```

```
R^2: 0.7504536759876399
Adjusted R^2: 0.7447169788839075
MAE: 4295.167910265434
MSE: 37477057.03936295
RMSE: 6121.850785453934
```

As the Root square Score of both train and test data is 0.75 we can conclude that the model shows negligible signs of overfitting

In [ ]:

In [ ]:

Importing and performing Gradient Boosting Regressor For test Data

In [57]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

In [58]:
```
gradientregressor = GradientBoostingRegressor(n_estimators = 3, learning_rate = 1.0
```

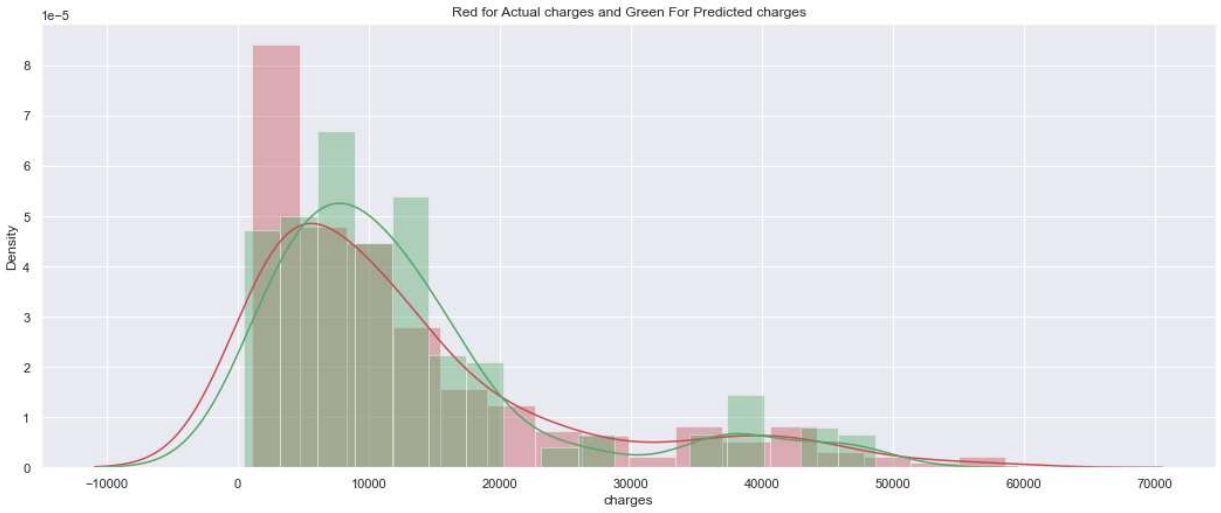
In [60]:
```
model = gradientregressor.fit(X_train, Y_train)

G_Pred = model.predict(X_test)
```

In [61]:
```
print('R^2:',metrics.r2_score(Y_test,G_Pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(Y_test,G_Pred))*(len(Y_test)-1)/(len(Y
print('MAE:',metrics.mean_absolute_error(Y_test, G_Pred))
print('MSE:',metrics.mean_squared_error(Y_test, G_Pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test, G_Pred)))
```

```
R^2: 0.8501517165141571
Adjusted R^2: 0.8467069283880457
MAE: 2699.57396612246
MSE: 22504329.36520198
RMSE: 4743.872823464177
```

In [80]:
```
plt.figure(figsize=(18,7))
sns.distplot( Y_test,color='r')
sns.distplot(G_Pred,color='g')
plt.title('Red for Actual charges and Green For Predicted charges')
plt.show()
```



Importing and performing Random Forest Regressor For test Data

In [63]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [64]:
```
RFR = RandomForestRegressor()
```

In [66]:
```
RFR.fit(X_test, Y_test)
```

Out[66]:
```
RandomForestRegressor()
```

In [69]:
```
# Model prediction on train data
```

```
RFR_pred = RFR.predict(X_test)
```

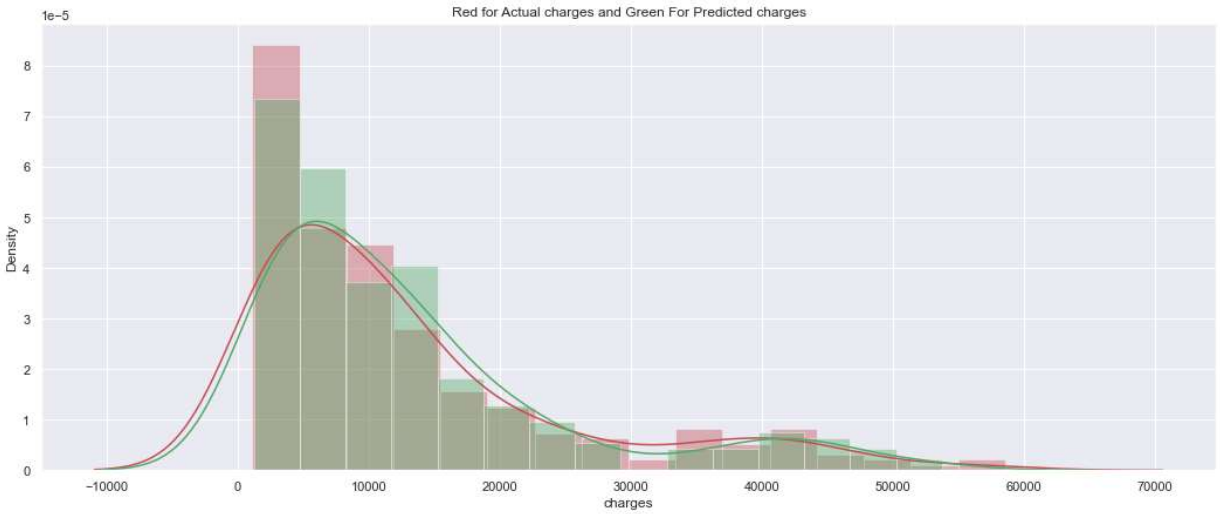
In [72]:

```
# Model Evaluation
A=print('R^2:',metrics.r2_score(Y_test,RFR_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(Y_test,RFR_pred))*(len(Y_test)-1)/(len
print('MAE:',metrics.mean_absolute_error(Y_test, RFR_pred))
print('MSE:',metrics.mean_squared_error(Y_test, RFR_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test, RFR_pred)))
```

```
R^2: 0.9753787422597736
Adjusted R^2: 0.974812736334711
MAE: 1060.8471548256218
MSE: 3697639.243388012
RMSE: 1922.9246587913974
```

In [79]:

```
plt.figure(figsize=(18,7))
sns.distplot( Y_test,color='r')
sns.distplot(RFR_pred,color='g')
plt.title('Red for Actual charges and Green For Predicted charges')
plt.show()
```



In [97]:

```
models = pd.DataFrame({
    'Model': ['Linear Regression', 'Random Forest', 'Gradient Boost'],
    'R-squared Score': [ 0.7504536759876399*100,0.9753787422597736*100,0.85015171651
models
```

Out[97]:

	Model	R-squared Score
0	Linear Regression	75.045368
1	Random Forest	97.537874
2	Gradient Boost	85.015172

In [ ]:

Prediciting Cost for New customers

In [ ]:

In [85]:
```
Data={'age':21,'sex':1,'bmi':25.30,'children':0,'smoker':1,'region':2}
```

In [86]:
```
DF=pd.DataFrame(Data,index=[0])
DF
```

Out[86]:

	age	sex	bmi	children	smoker	region
0	21	1	25.3	0	1	2

In [84]:
```
new_pred=RFR.predict(DF)
print("Medical Insurance For New Customer is : ",new_pred[0])
```

```
Medical Insurance For New Customer is :  7532.5944684999995
```

As random Forest Regressor is having highest R^2 score we will implement the random Forest Regressor for Prediciting Prices

In [ ]:
```

```