

**NAME**: Muhammad Shoaib Qureshi

**ID**  : F2019065034

**COURSE:** DATA SCIENCE TECHNOLOGIES

**SUBJECT**: PROJECT DOCUMENTATION

**TO**  : MAZHAR JAVED AWAN

**PROJECT DOCUMENTATION**

Thi**s** documentation is on Data Science Project that I built. I have selected a Pakistan Medical Insurance data set and did some data analysis, applied some pandas and numpy operations, also plotted many charts by using matpolit libraries and seaborns to compare different labels features of data set. After doing all these things I finally Build a Logistic Regression Model by creating dummy variables.

First I inserted libraries that I have to use to build the model and do some data analysis stuff.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

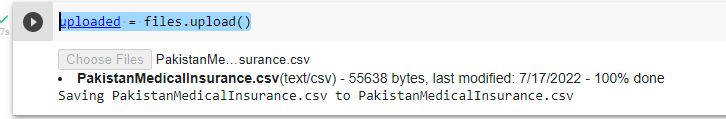
import warnings

warnings.filterwarnings('ignore')

And the we upload the file on the google colab.

from google.colab import files

uploaded = files.upload()

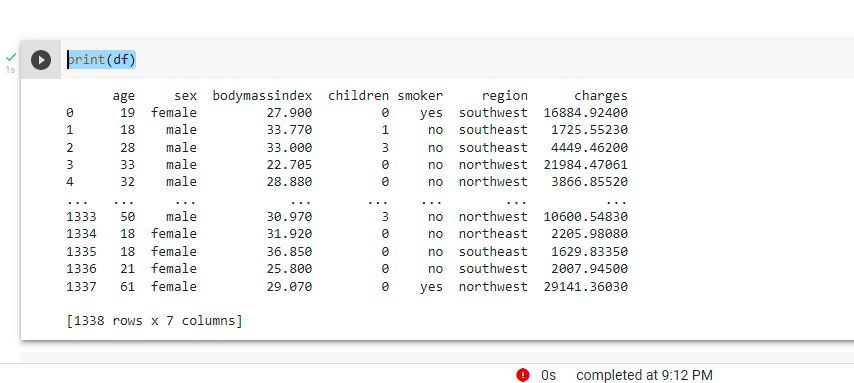


# Then Read the file into a DataFrame: df

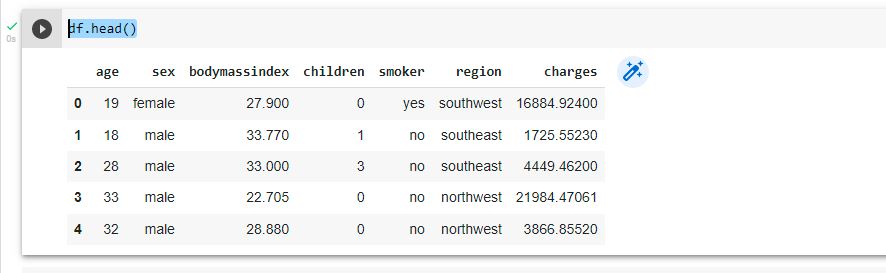
df = pd.read\_csv('/content/PakistanMedicalInsurance.csv')

**Classification of Data**

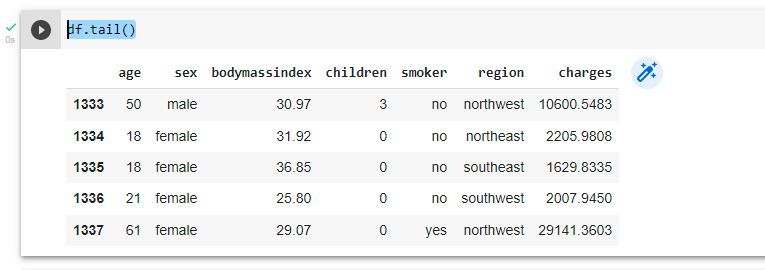
print(df)



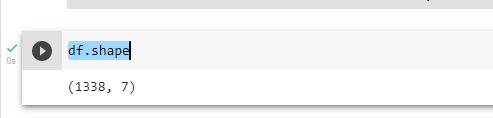
df.head()



df.tail()



df.shape



df.describe()

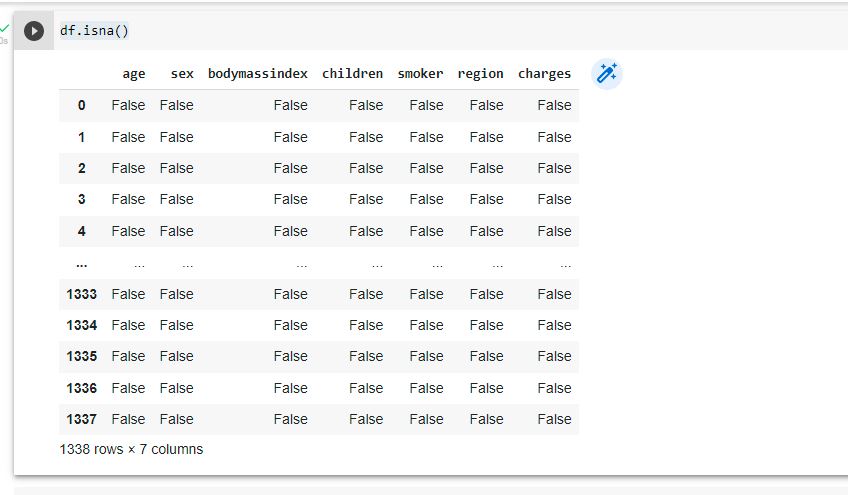


df.columns



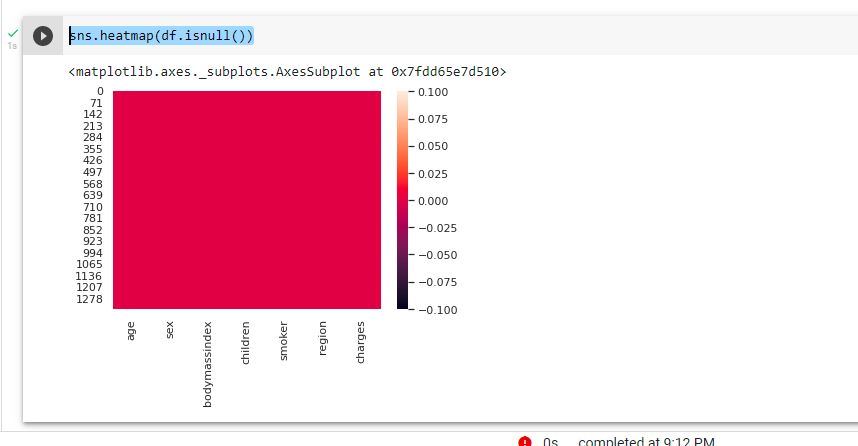
# ****Analyzing missing values and noisy data****

df.isna()



Then we plot a Heatmap to check if our data set has missing values:

sns.heatmap(df.isnull())



As from above commands it can be seen that there are not any empty cells so no need to do interpolaion.

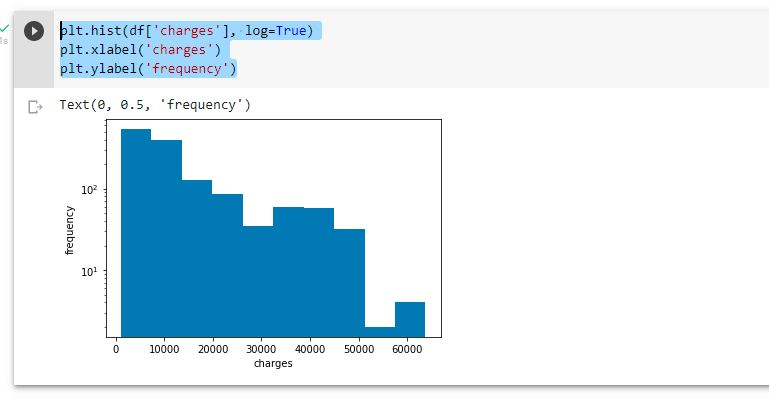
# Frequency count for Categorical Data

Create the Histogram plot

plt.hist(df['charges'], log=True)

plt.xlabel('charges')

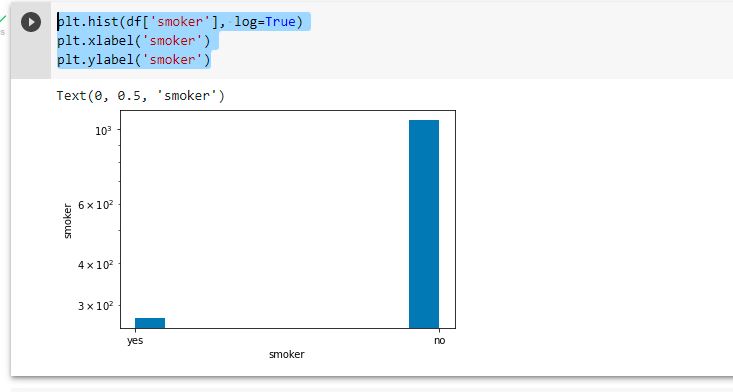
plt.ylabel('frequency')



plt.hist(df['smoker'], log=True)

plt.xlabel('smoker')

plt.ylabel('smoker')

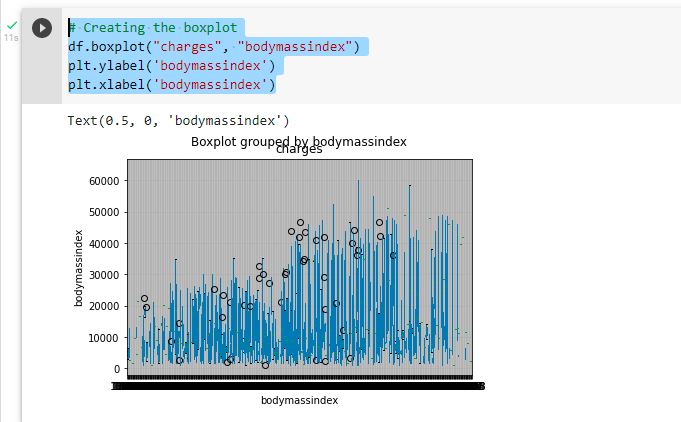


# Creating the boxplot

df.boxplot("charges", "bodymassindex")

plt.ylabel('bodymassindex')

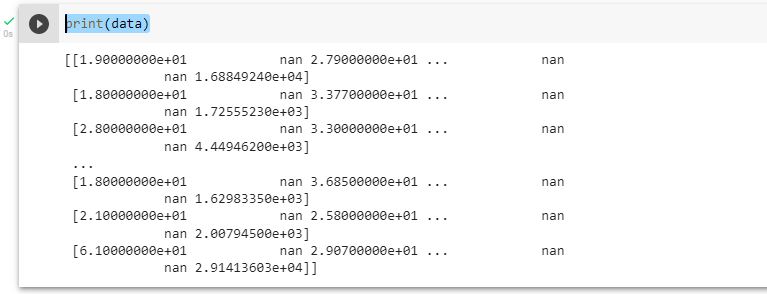
plt.xlabel('bodymassindex')



# Numpy operations

data = np.genfromtxt('/content/PakistanMedicalInsurance.csv',delimiter=',', skip\_header= True,)

print(data)



#splitting data into column arrays using numpy

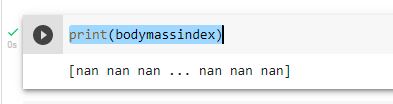
charges = data[:,0]

bodymassindex = data[:,1]

print(charges)

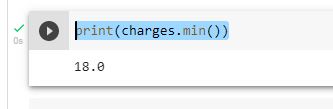


print(bodymassindex)

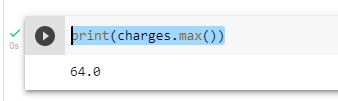


#some basic observations

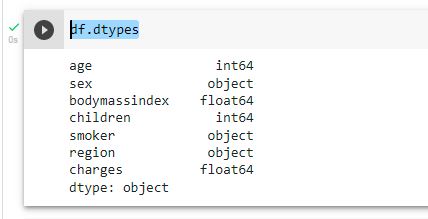
print(charges.min())



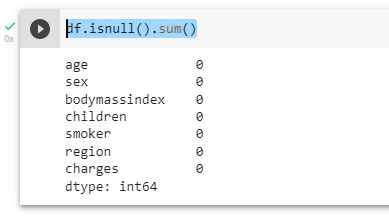
print(charges.max())



df.dtypes

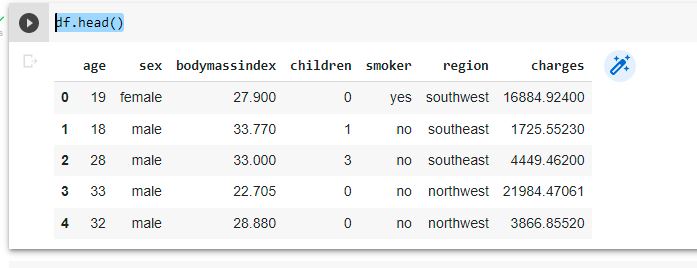


df.isnull().sum()

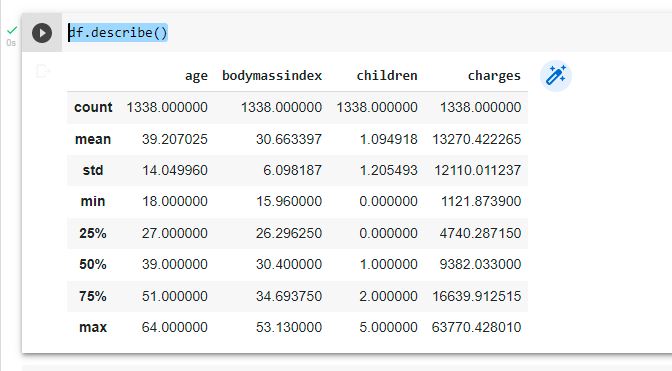


# ****Pandas operations****

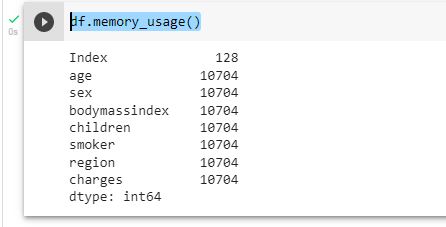
df.head()



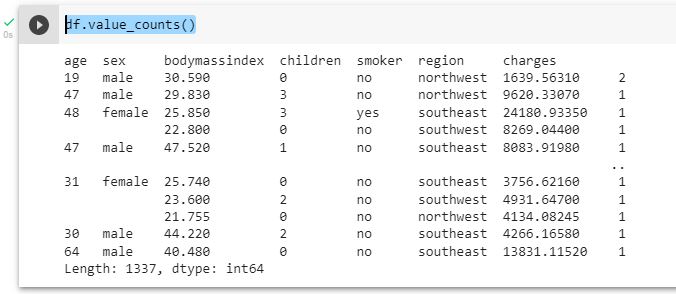
df.describe()



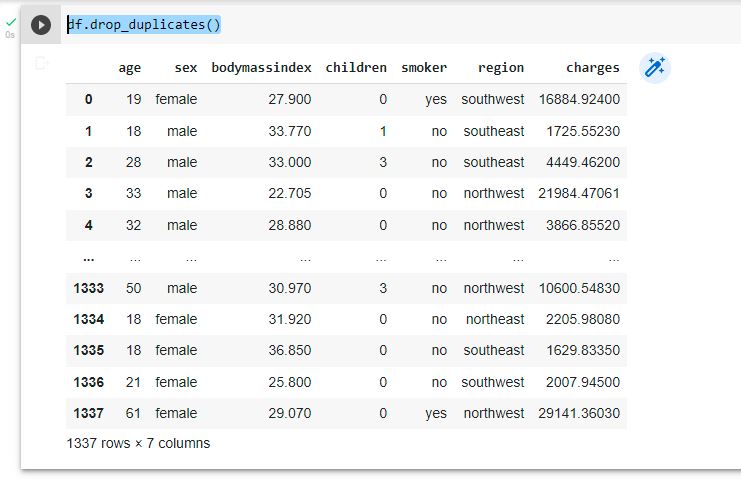
df.memory\_usage()



df.value\_counts()



df.drop\_duplicates()



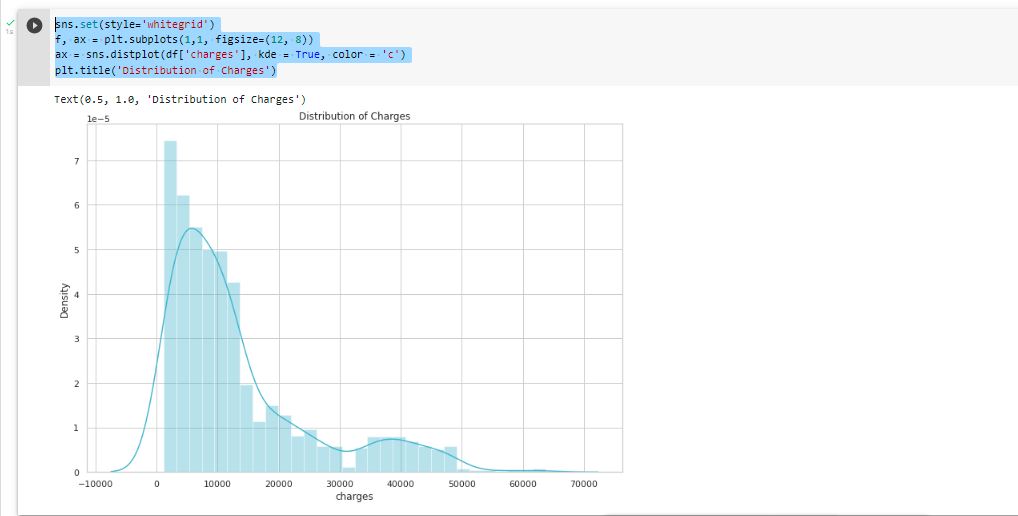
# ****Seaborn Graph plote****

sns.set(style='whitegrid')

f, ax = plt.subplots(1,1, figsize=(12, 8))

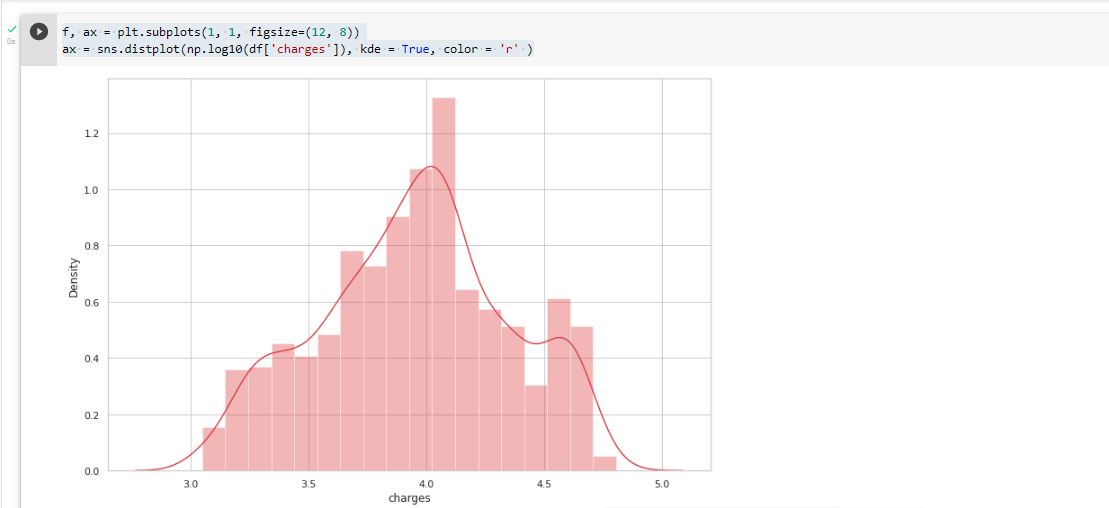
ax = sns.distplot(df['charges'], kde = True, color = 'c')

plt.title('Distribution of Charges')



f, ax = plt.subplots(1, 1, figsize=(12, 8))

ax = sns.distplot(np.log10(df['charges']), kde = True, color = 'r' )

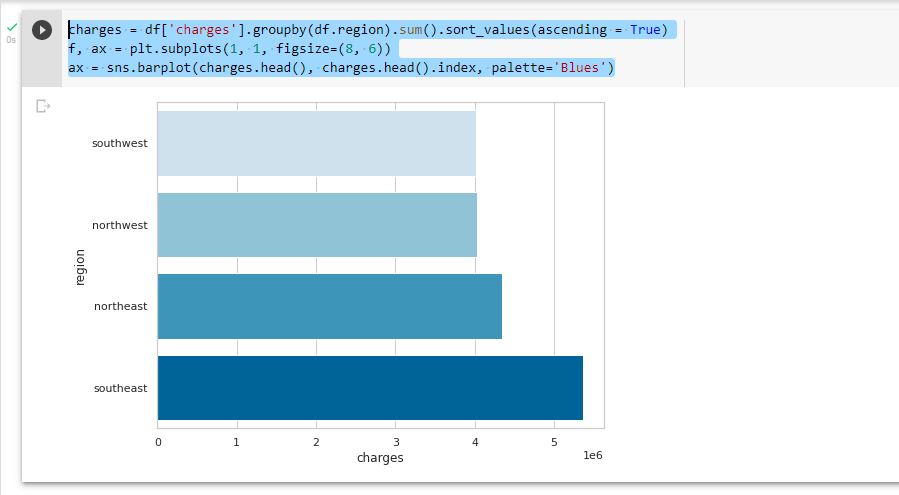


# ****REGIONAL CHARGES:****

charges = df['charges'].groupby(df.region).sum().sort\_values(ascending = True)

f, ax = plt.subplots(1, 1, figsize=(8, 6))

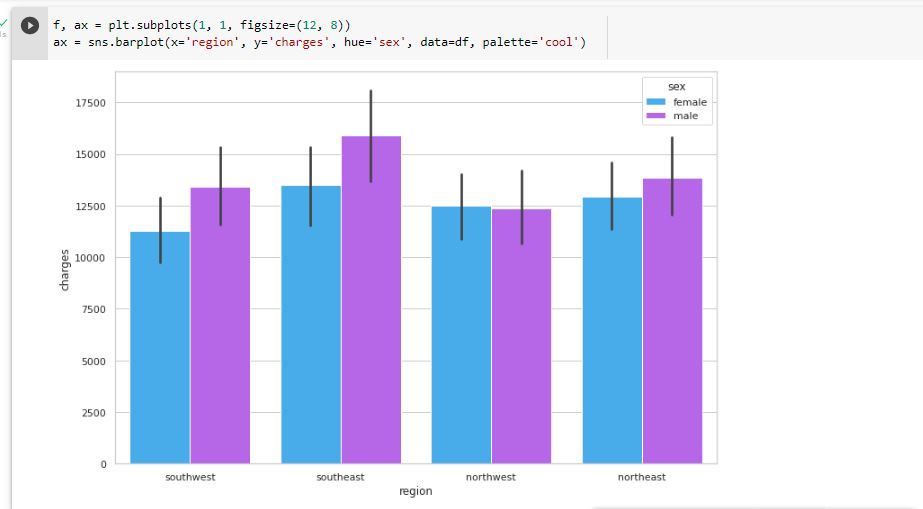
ax = sns.barplot(charges.head(), charges.head().index, palette='Blues')



**So High number of Charges in east and less in West:**

f, ax = plt.subplots(1, 1, figsize=(12, 8))

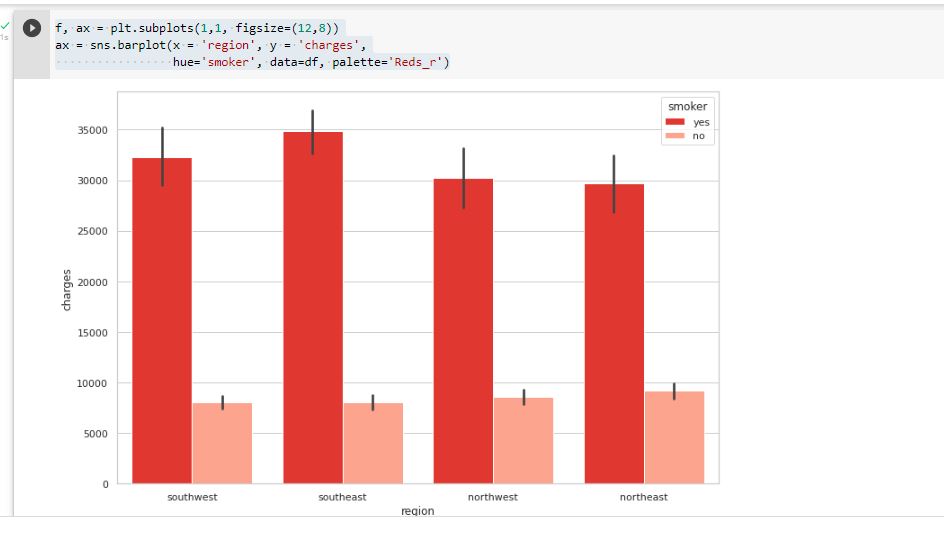
ax = sns.barplot(x='region', y='charges', hue='sex', data=df, palette='cool')



f, ax = plt.subplots(1,1, figsize=(12,8))

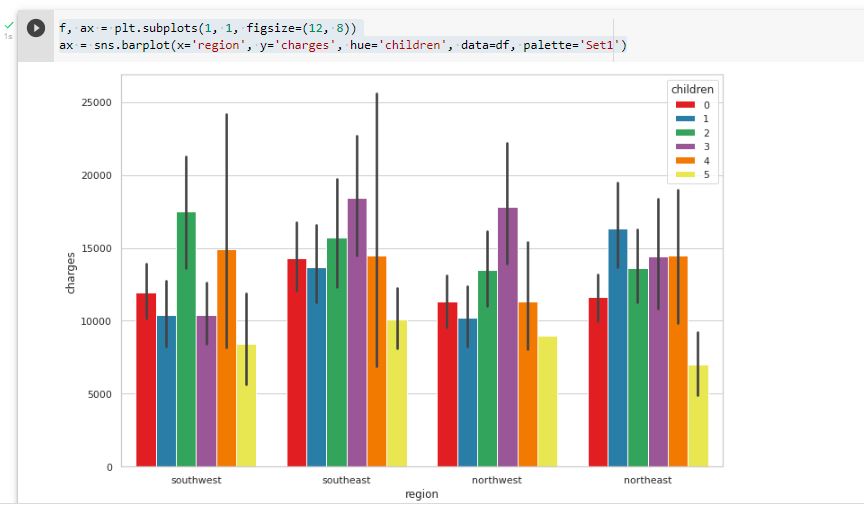
ax = sns.barplot(x = 'region', y = 'charges',

                 hue='smoker', data=df, palette='Reds\_r')



f, ax = plt.subplots(1, 1, figsize=(12, 8))

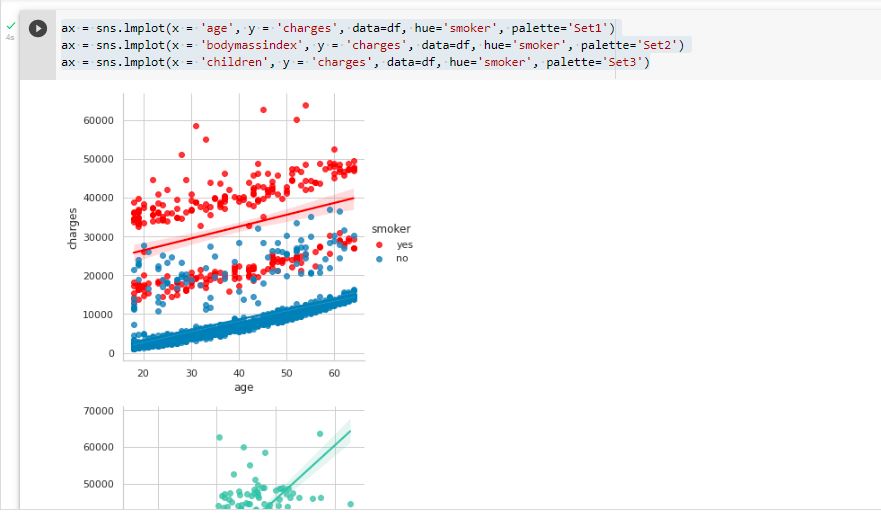
ax = sns.barplot(x='region', y='charges', hue='children', data=df, palette='Set1')

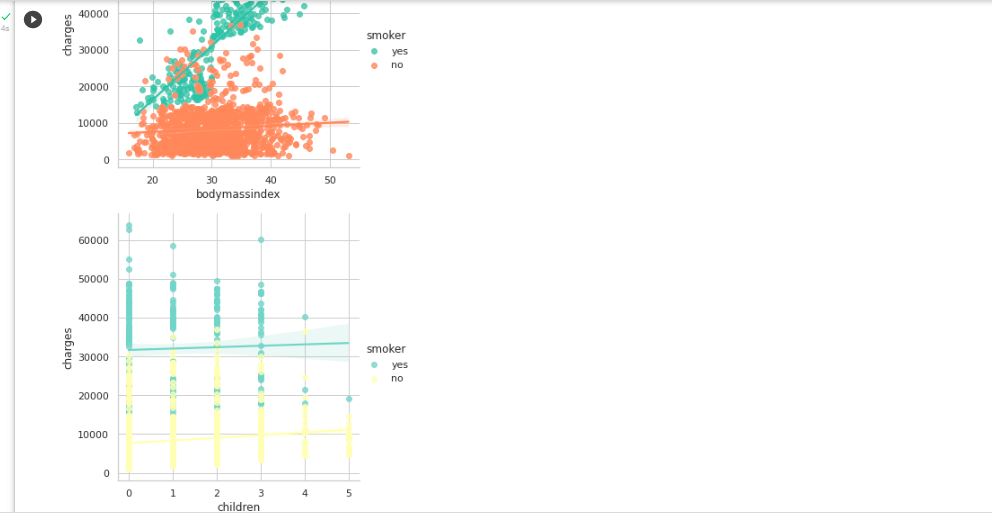


ax = sns.lmplot(x = 'age', y = 'charges', data=df, hue='smoker', palette='Set1')

ax = sns.lmplot(x = 'bodymassindex', y = 'charges', data=df, hue='smoker', palette='Set2')

ax = sns.lmplot(x = 'children', y = 'charges', data=df, hue='smoker', palette='Set3')

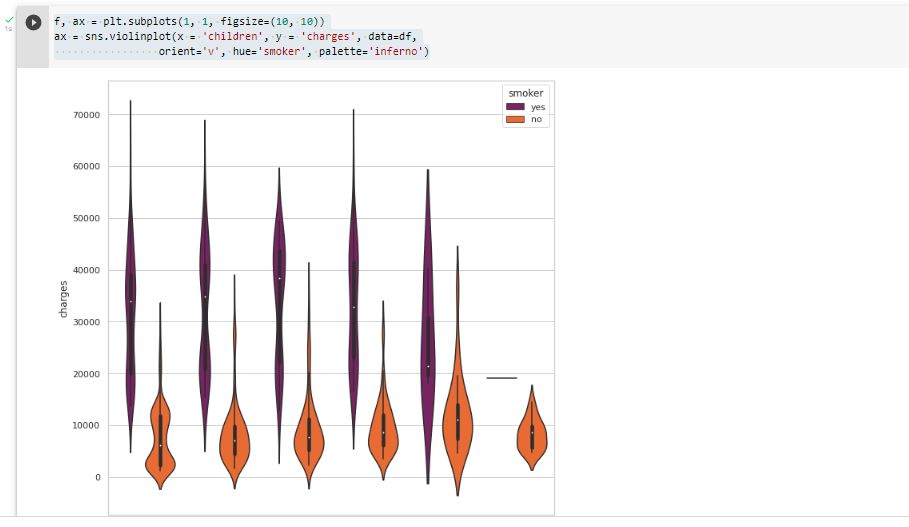




f, ax = plt.subplots(1, 1, figsize=(10, 10))

ax = sns.violinplot(x = 'children', y = 'charges', data=df,

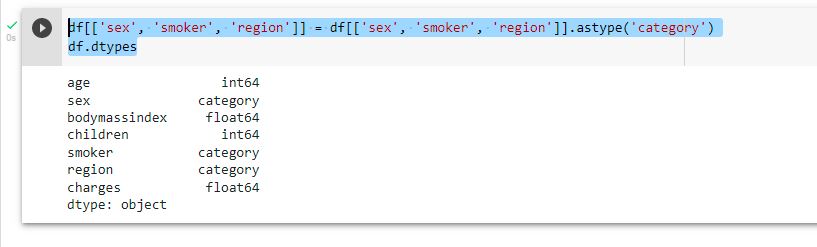
                 orient='v', hue='smoker', palette='inferno')



**NOW CONVERTING TYPE OF OBJECTS INTO A CATEGORICAL DATA:**

df[['sex', 'smoker', 'region']] = df[['sex', 'smoker', 'region']].astype('category')

df.dtypes



# Machine learning models

from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()

label.fit(df.sex.drop\_duplicates())

df.sex = label.transform(df.sex)

label.fit(df.smoker.drop\_duplicates())

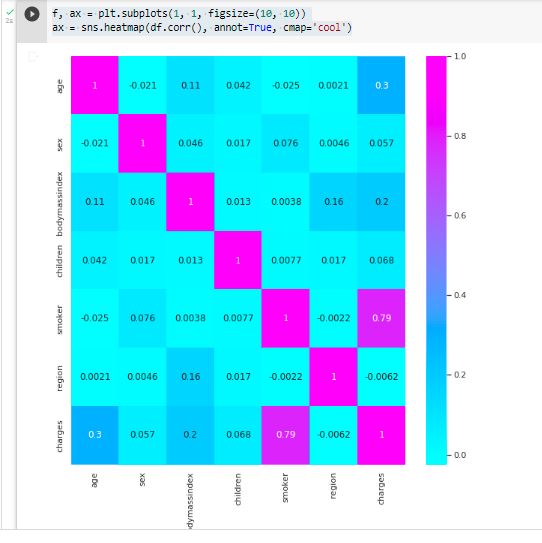
df.smoker = label.transform(df.smoker)

label.fit(df.region.drop\_duplicates())

df.region = label.transform(df.region)

f, ax = plt.subplots(1, 1, figsize=(10, 10))

ax = sns.heatmap(df.corr(), annot=True, cmap='cool')



# LinearRegression model

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split as holdout

from sklearn import metrics

x = df.drop(['charges'], axis = 1)

y = df['charges']

x\_train, x\_test, y\_train, y\_test = holdout(x, y, test\_size=0.2, random\_state=0)

Lin\_reg = LinearRegression()

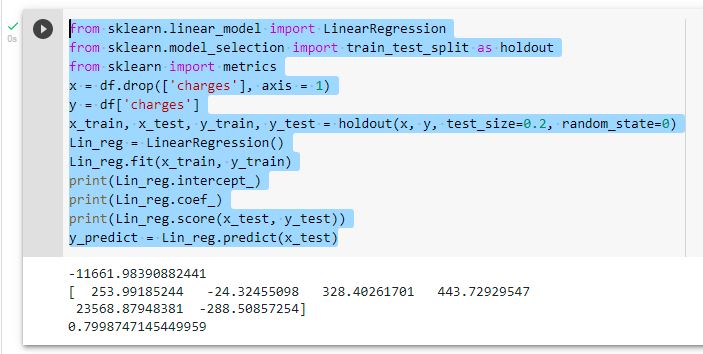
Lin\_reg.fit(x\_train, y\_train)

print(Lin\_reg.intercept\_)

print(Lin\_reg.coef\_)

print(Lin\_reg.score(x\_test, y\_test))

y\_predict = Lin\_reg.predict(x\_test)



df



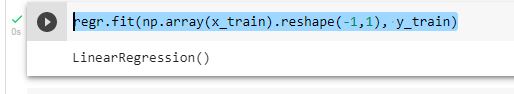
df.columns.values



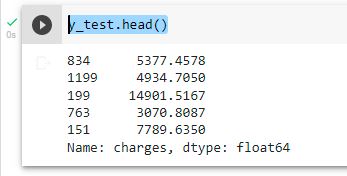
x\_train, x\_test, y\_train, y\_test = train\_test\_split(df['age'], df['charges'], test\_size = 0.2)

regr = LinearRegression()

regr.fit(np.array(x\_train).reshape(-1,1), y\_train)

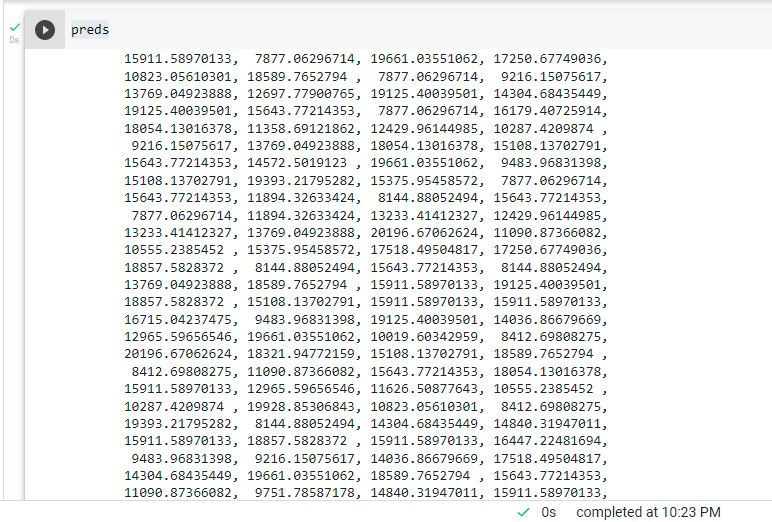


y\_test.head()



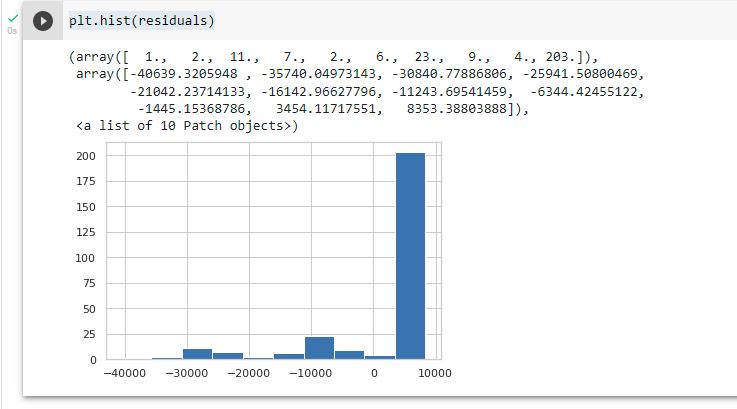
preds = regr.predict(np.array(x\_test).reshape(-1,1))

preds



residuals = preds - y\_test

plt.hist(residuals)



# Random Forest Regressor model

#classification reports

from sklearn.ensemble import RandomForestRegressor as rfr

x = df.drop(['charges'], axis=1)

y = df.charges

Rfr = rfr(n\_estimators = 100, criterion = 'mse',

                              random\_state = 1,

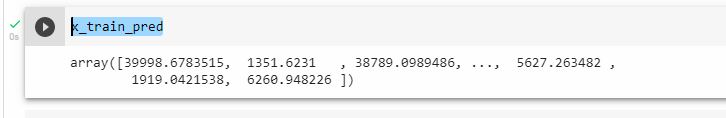
                              n\_jobs = -1)

Rfr.fit(x\_train,y\_train)

x\_train\_pred = Rfr.predict(x\_train)

x\_test\_pred = Rfr.predict(x\_test)

x\_train\_pred



x\_test\_pred

