**Name:** Soham Belurgikar

**Roll No.:** 2019130006

**Course:** **DA (Data Analytics)**

**Assignment No.:** 1

**Part:** 3

**Name of the Assignment:** Correlation and Regression

**Problem Statement:**

The smartphone market in 2022 is filled with variety of phones catering to every person’s needs. You can buy phones from brands like Samsung, Apple, Xiaomi, buy a phone which costs as low as Rs. 1000 or as high as Rs. 179900, buy phones with colours like Black, Blue, Rose Gold etc.

The aim of this experiment is to check if there exists a correlation and/or linear regression between a pair of variables and also to determine if the correlation/regression statistics are significant.

**Implementation:**

[Dataset link](https://www.kaggle.com/devsubhash/flipkart-mobiles-dataset/)

[Colab Link](https://colab.research.google.com/drive/1ae_ckgviUNzV8-Olwj26r9EE2BzrVQ9D?usp=sharing)

**The dataset:**

The chosen dataset consists of 2647 samples with 8 attributes, namely:

* Brand - Name of the Mobile Manufacturer
* Model - Model name / number of the Mobile Phone
* Colour - Colour of the model. Missing or Null values indicate no specified colour of the model offered on the ecommerce website.
* Memory - RAM of the model (4GB, 6GB, 8GB, etc.)
* Storage - ROM of the model (32GB, 64GB, 128GB, 256GB, etc.)
* Rating - Rating of the model based on reviews (out of 5). Missing or Null values indicate there are no ratings present for the model.
* Selling Price- Selling Price/Discounted Price of the model in INR when this data was scraped. Ideally price indicates the discounted price of the model
* Original Price- Actual price of the model in INR. Missing values or null values would indicate that the product is being sold at the actual price available in the 'Price' column.

**Importing the required libraries:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

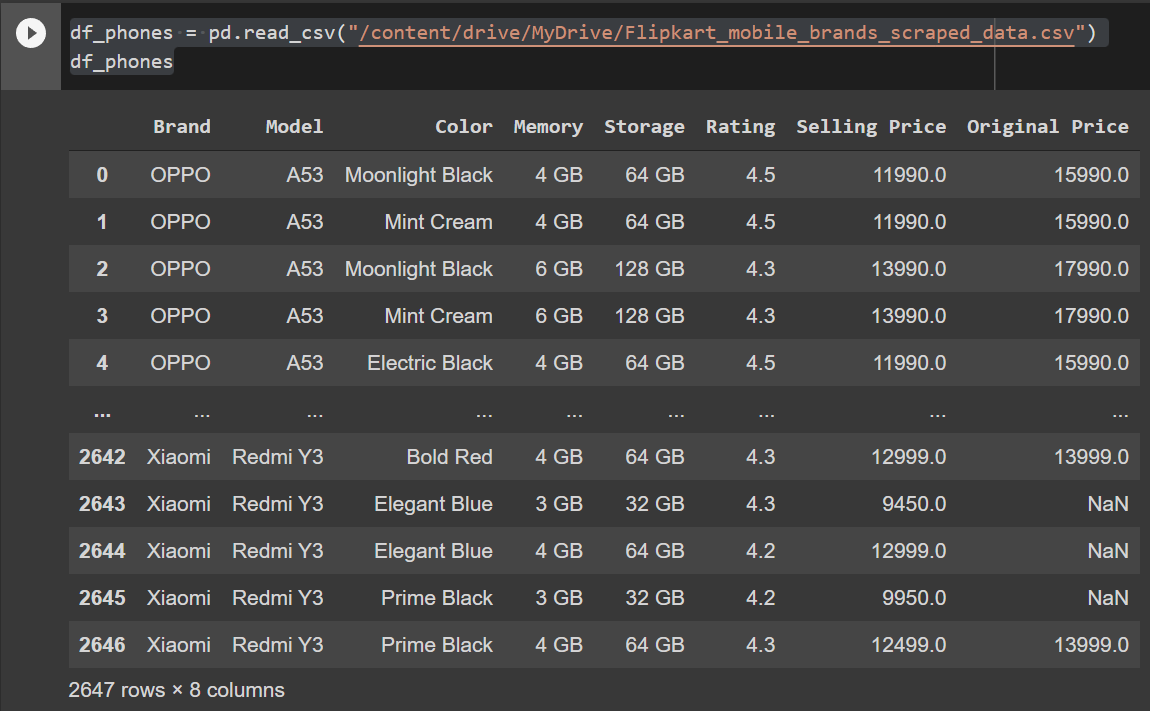
from sklearn.linear\_model import LinearRegression

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

from sklearn import metrics

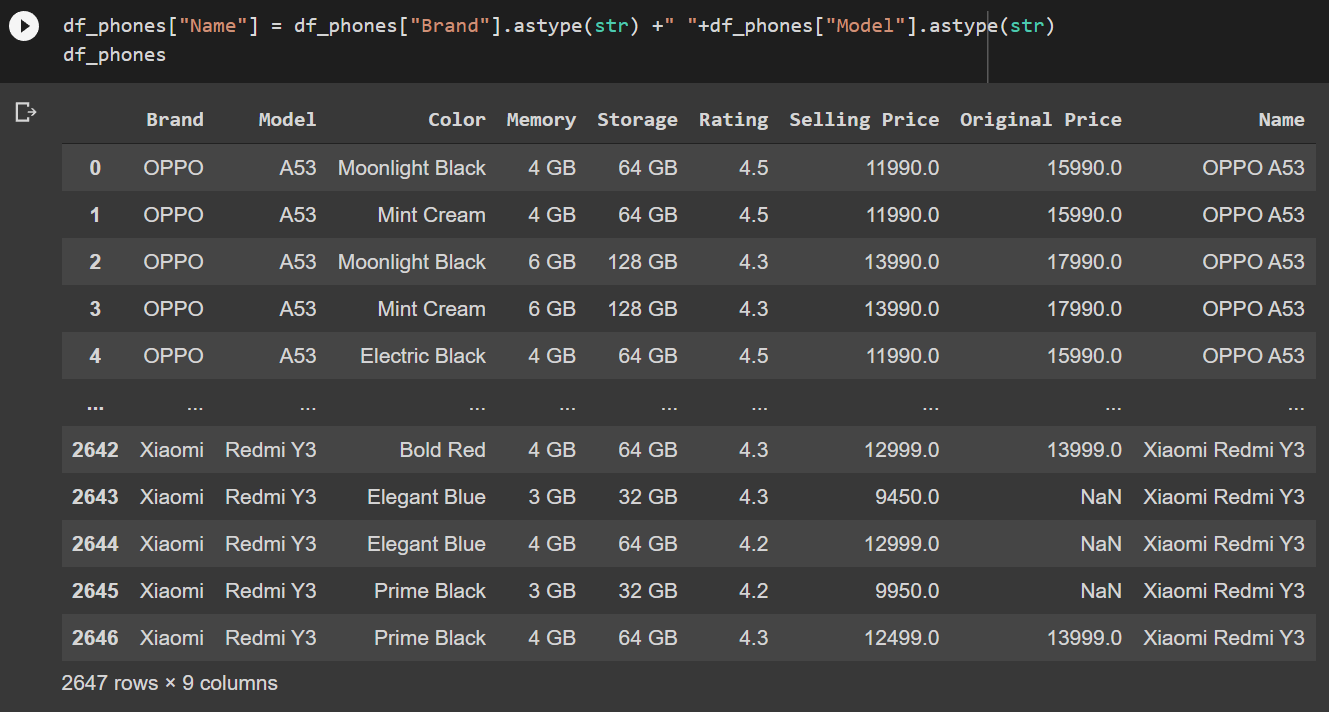
import statsmodels.api as sm

**Loading the data into the dataframe:**

****

**Adding the Name column:**

Name of the phone = Name of Brand + Name of Model



df\_phones.shape

Using .shape() we can get information about the number of rows and columns of the dataset:

(2647, 9)

So, the dataset contains 2647 rows (samples) and 9 columns (features).

**Removing duplicate rows:**

duplicate\_rows\_df = df\_phones[df\_phones.duplicated()]

print("number of duplicate rows: ", duplicate\_rows\_df.shape)

This gives us the number of rows which have the same values for every column:

number of duplicate rows: (107, 9)

So, the dataset contained 107 rows which were duplicates.

df\_phones.count()

You can also check the number of rows that each column contains using the .count() method:

Brand 2647

Model 2645

Color 2505

Memory 2605

Storage 2568

Rating 2647

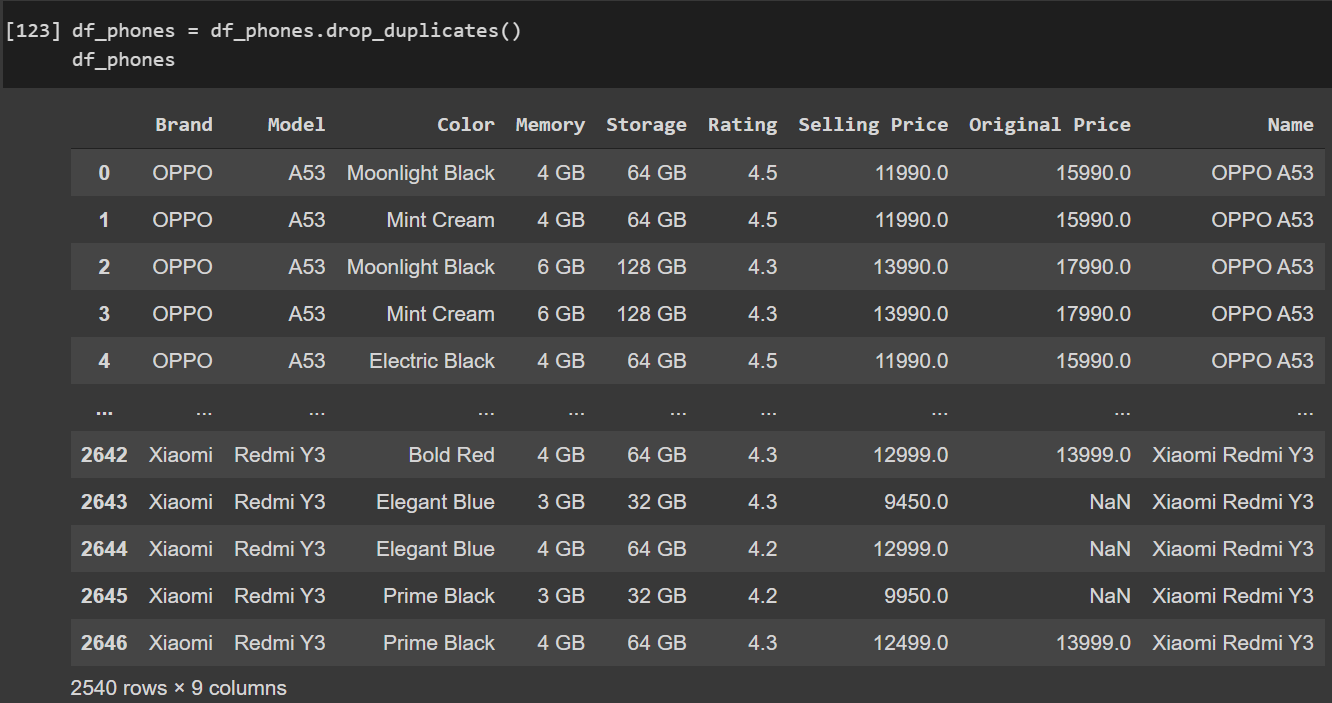
Selling Price 2644

Original Price 969

Name 2647

dtype: int64

You can delete the duplicate rows using just a simple method, i.e., .drop\_duplicates():



df\_phones.count()

Brand 2540

Model 2538

Color 2407

Memory 2501

Storage 2463

Rating 2540

Selling Price 2537

Original Price 934

Name 2540

dtype: int64

**Removing null / missing values:**

print(df\_phones.isnull().sum())

The .isnull().sum() command will return the number of values which are missing for every column:

Brand 0

Model 2

Color 133

Memory 39

Storage 77

Rating 0

Selling Price 3

Original Price 1606

Name 0

dtype: int64

We will drop lines with model unknown or missing memory information or missing storage information. Put missing value of colour to "Base". Drop lines with missing both prices else fill one with the other.

df\_phones = df\_phones.dropna(subset=["Model", "Memory","Storage"])

df\_phones["Selling Price"] = df\_phones["Selling Price"].fillna(df\_phones["Original Price"])

df\_phones["Original Price"] = df\_phones["Original Price"].fillna(df\_phones["Selling Price"])

df\_phones= df\_phones.dropna(subset=["Original Price","Selling Price"])

df\_phones["Color"] = df\_phones["Color"].fillna("Base")

print(df\_phones.isnull().sum())

Brand 0

Model 0

Color 0

Memory 0

Storage 0

Rating 0

Selling Price 0

Original Price 0

Name 0

dtype: int64

Now our dataset is free of null values.

**Converting str variables to float**

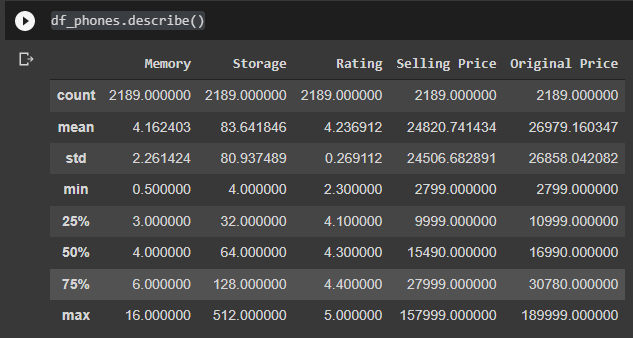
df\_phones = df\_phones[df\_phones['Memory'].str.endswith('GB')]

df\_phones['Memory'] = df\_phones['Memory'].str.replace('GB', '').astype(float)

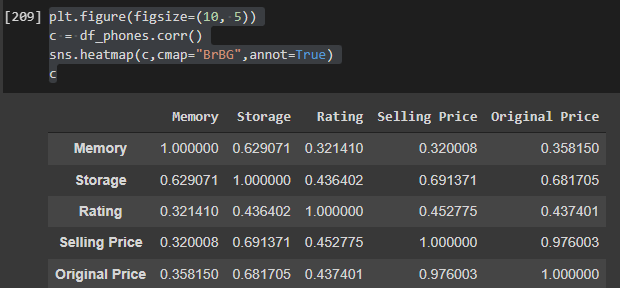
df\_phones = df\_phones[df\_phones['Storage'].str.contains('^[0-9].\*GB$')]

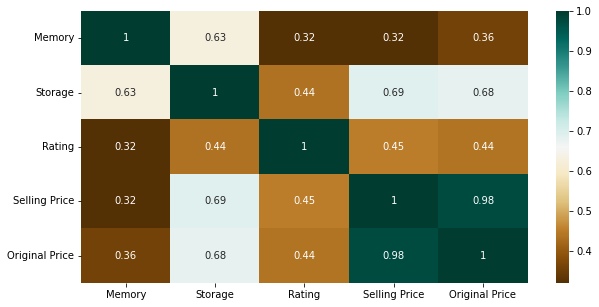
df\_phones['Storage'] = df\_phones['Storage'].str.replace('GB', '').astype(float)

df\_phones = df\_phones[df\_phones['Rating'] > 0]



**Correlation**

****

****

**Scatter plots**

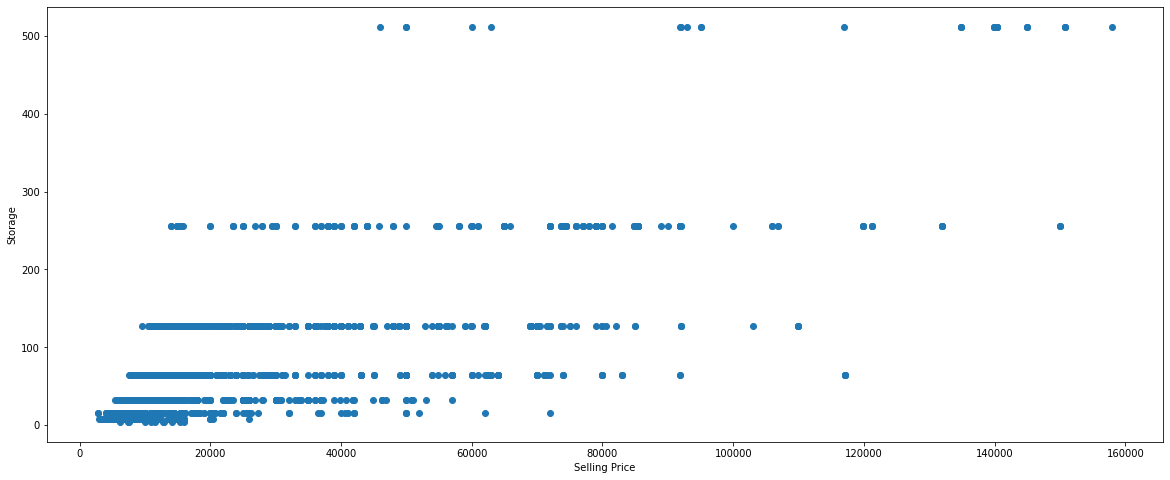
fig, ax = plt.subplots(figsize=(20, 8))

ax.scatter(df\_phones['Selling Price'], df\_phones['Storage'])

ax.set\_xlabel('Selling Price')

ax.set\_ylabel('Storage')

plt.show()

****

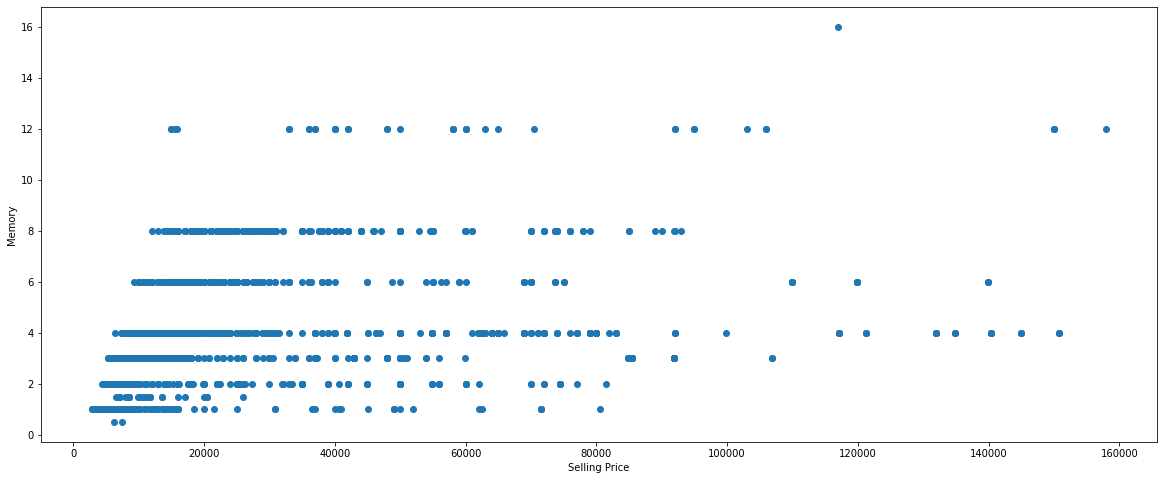
fig, ax = plt.subplots(figsize=(20, 8))

ax.scatter(df\_phones['Selling Price'], df\_phones['Memory'])

ax.set\_xlabel('Selling Price')

ax.set\_ylabel('Memory')

plt.show()

****

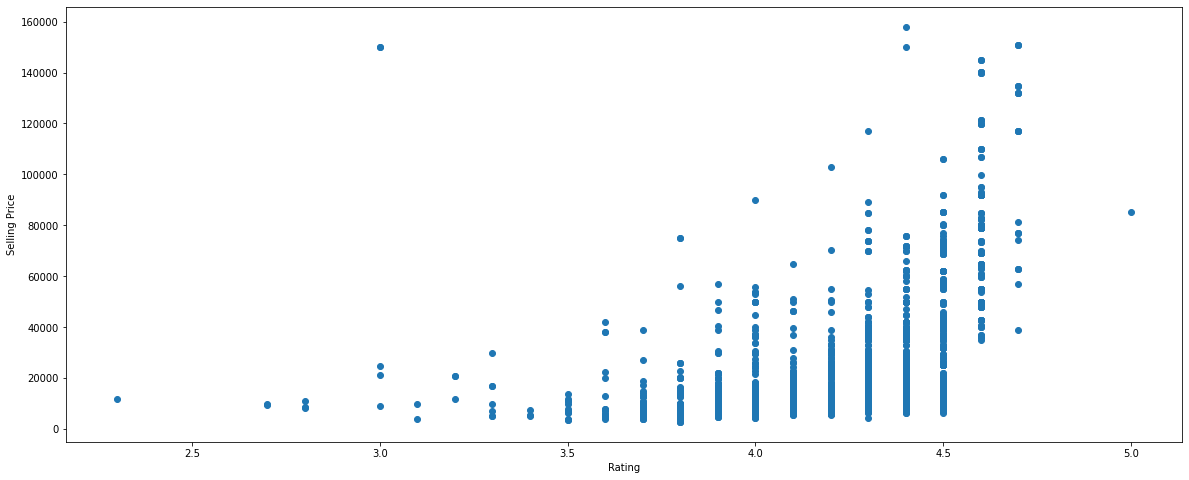
fig, ax = plt.subplots(figsize=(20, 8))

ax.scatter(df\_phones['Rating'], df\_phones['Selling Price'])

ax.set\_xlabel('Rating')

ax.set\_ylabel('Selling Price')

plt.show()

****

fig, ax = plt.subplots(figsize=(20, 8))

ax.scatter(df\_phones['Memory'], df\_phones['Storage'])

ax.set\_xlabel('Memory')

ax.set\_ylabel('Storage')

plt.show()

****

**Linear Regression statistics**

X = df\_phones['Rating'].values.reshape(-1, 1)

y = df\_phones['Selling Price'].values.reshape(-1, 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

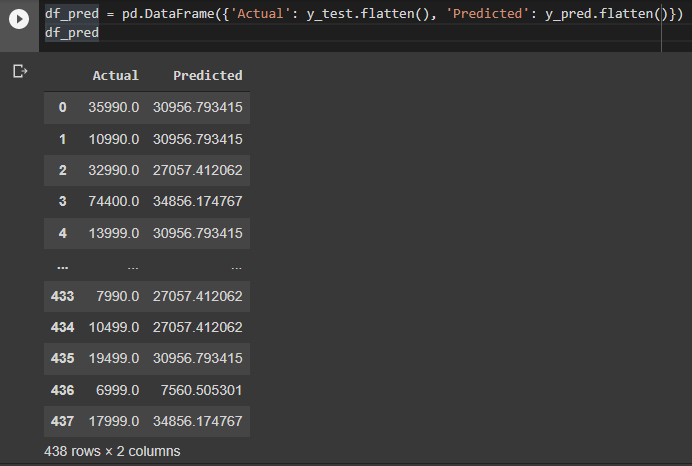
print(regressor.intercept\_)

print(regressor.coef\_)

[-140615.9860851]

[[38993.81352265]]

y\_pred = regressor.predict(X\_test)

****

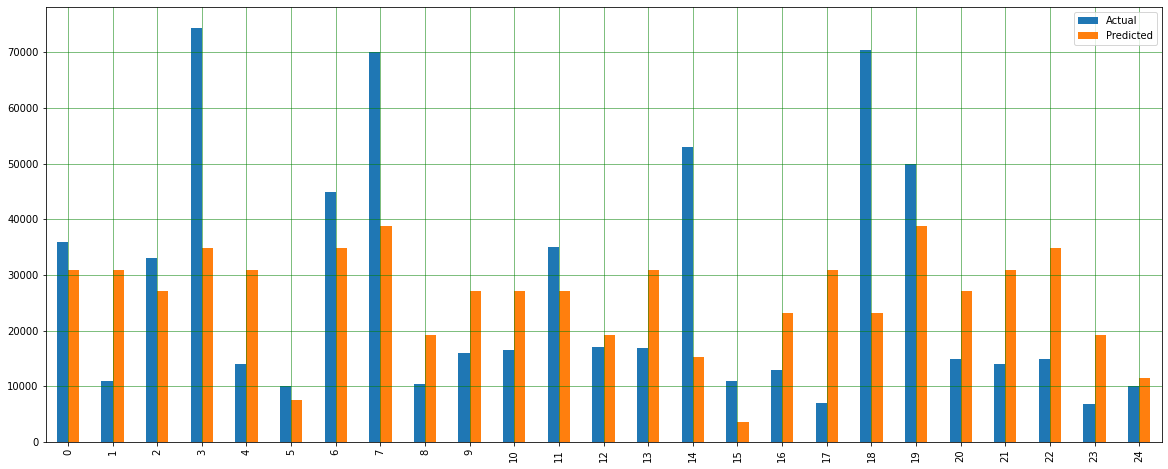
df1 = df\_pred.head(25)

df1.plot(kind='bar',figsize=(20, 8))

plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.show()

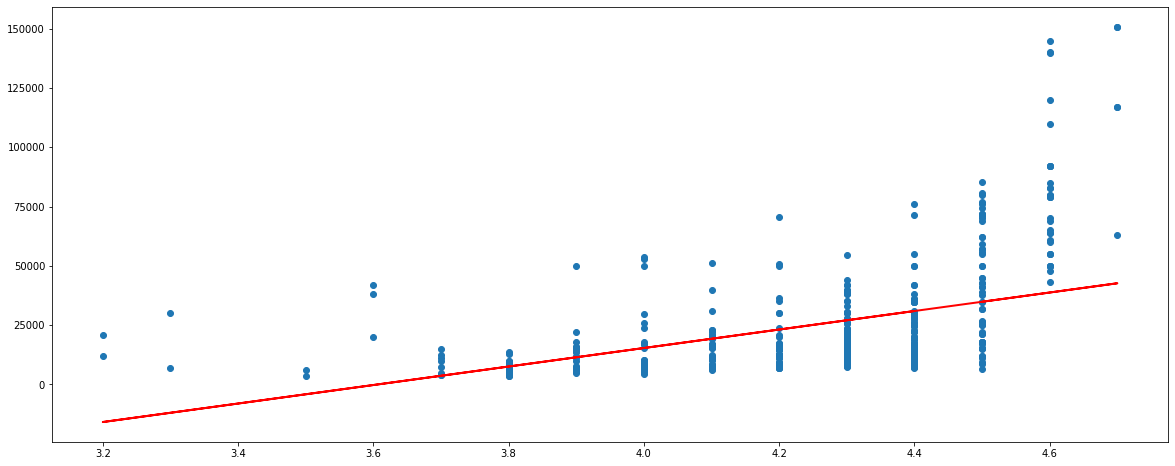
****

fig, ax = plt.subplots(figsize=(20, 8))

plt.scatter(X\_test, y\_test)

plt.plot(X\_test, y\_pred, color='red', linewidth=2)

plt.show()

****

r\_sq = regressor.score(X\_test, y\_test)

print('coefficient of determination:', r\_sq)

coefficient of determination: 0.25218898685136426

print('intercept:', regressor.intercept\_)

print('slope:', regressor.coef\_)

intercept: [-140615.9860851]

slope: [[38993.81352265]]

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

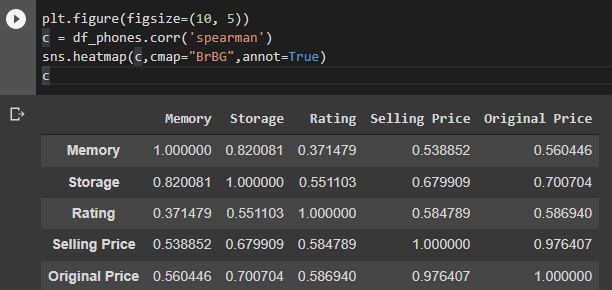
print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

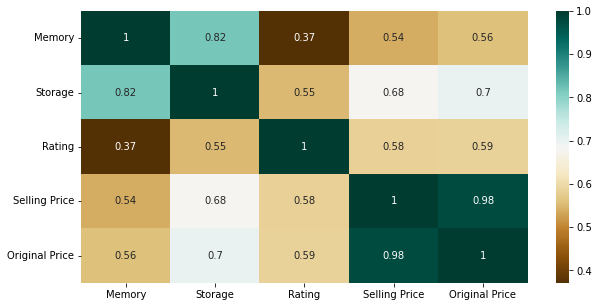
Mean Absolute Error: 15130.299756336022

Mean Squared Error: 474562981.7514144

Root Mean Squared Error: 21784.466524370397

**Spearman correlation coefficient**

****

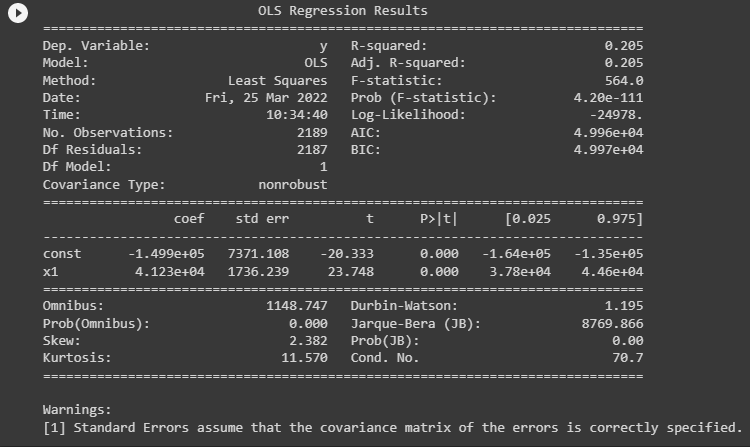
****

X2 = sm.add\_constant(X)

est = sm.OLS(y, X2)

est2 = est.fit()

print(est2.summary())

****

**Conclusion:**

* The fitted model implies that, when comparing two phones whose Rating differ by one unit, the phone with the higher Rating will, on average, have around 39000 units higher Selling Price. This difference is statistically significant, because the p-value, shown under the column labelled *P>|t|*, is less than the significance value of 0.05. This means that there is strong evidence of a linear association between the variables Selling Price and Rating.
* The other parameter to test the efficacy of the model is the *R-squared* value, which represents the percentage variation in the dependent variable (Selling Price) that is explained by the independent variable (Rating). The higher the value, the better the explain ability of the model, with the highest value being one. In our case, the R-squared value of 0.205 means that 20% of the variation in the variable Selling Price is explained by the variable Rating. This is very low, which means that the Rating will not be accurately predicted by the Selling Price of a phone.
* The Spearman correlation coefficient is also an indicator of the extent and strength of the linear relationship between the two variables. The spearman correlation coefficient for Selling Price and Rating comes out to be 0.58. This is a slight strong positive correlation between the two variables, with the highest value being one.

**References:**

[Dataset link](https://www.kaggle.com/devsubhash/flipkart-mobiles-dataset/)

[Colab link](https://colab.research.google.com/drive/1ae_ckgviUNzV8-Olwj26r9EE2BzrVQ9D?usp=sharing)