**App Rating Prediction**

Project 1

DESCRIPTION

**Objective**: Make a model to predict the app rating, with other information about the app provided.

**Problem Statement:**

Google Play Store team is about to launch a new feature wherein, certain apps that are promising, are boosted in visibility. The boost will manifest in multiple ways including higher priority in recommendations sections (“Similar apps”, “You might also like”, “New and updated games”). These will also get a boost in search results visibility.  This feature will help bring more attention to newer apps that have the potential.

**Domain**: General

**Analysis to be done**: The problem is to identify the apps that are going to be good for Google to promote. App ratings, which are provided by the customers, is always a great indicator of the goodness of the app. The problem reduces to: predict which apps will have high ratings.

**Content:**Dataset: Google Play Store data (“googleplaystore.csv”)

**Fields in the data** –

* App: Application name
* Category: Category to which the app belongs
* Rating: Overall user rating of the app
* Reviews: Number of user reviews for the app
* Size: Size of the app
* Installs: Number of user downloads/installs for the app
* Type: Paid or Free
* Price: Price of the app
* Content Rating: Age group the app is targeted at - Children / Mature 21+ / Adult
* Genres: An app can belong to multiple genres (apart from its main category). For example, a musical family game will belong to Music, Game, Family genres.
* Last Updated: Date when the app was last updated on Play Store
* Current Ver: Current version of the app available on Play Store
* Android Ver: Minimum required Android version

**Steps to perform**:

1. Load the data file using pandas.

import numpy as np

import pandas as pd

import re

import matplotlib.pyplot as plt

import seaborn as sns

my\_data = pd.read\_csv('googleplaystore.csv')

my\_data.shape

(10841, 13)

my\_data.head(5)

1. Check for null values in the data. Get the number of null values for each column.

my\_data.isna().any()

App False

Category False

Rating True

Reviews False

Size False

Installs False

Type True

Price False

Content Rating True

Genres False

Last Updated False

Current Ver True

Android Ver True

dtype: bool

1. Drop records with nulls in any of the columns.

my\_data.dropna(axis=0, inplace=True)

1. Variables seem to have incorrect type and inconsistent formatting. You need to fix them:
2. Size column has sizes in Kb as well as Mb. To analyze, you’ll need to convert these to numeric.
   1. Extract the numeric value from the column
   2. Multiply the value by 1,000, if size is mentioned in Mb

def convert\_to\_int(value):

digits = re.findall('(\d+)', value)

result = 0 # default value in case size is NaN

if len(digits) > 0:

result = int(digits[0])

return result

def convert\_to\_kb(size):

new\_size = convert\_to\_int(size)

if 'M' in size:

new\_size = new\_size \* 1000

return new\_size

my\_data['New\_Size'] = my\_data['Size'].apply(convert\_to\_kb)

1. Reviews is a numeric field that is loaded as a string field. Convert it to numeric (int/float).

my\_data['Reviews'] = pd.to\_numeric(my\_data['Reviews'])

1. Installs field is currently stored as string and has values like 1,000,000+.
   1. Treat 1,000,000+ as 1,000,000
   2. remove ‘+’, ‘,’ from the field, convert it to integer

def cleanup\_installs(installs):

installs = installs.replace(',','')

return convert\_to\_int(installs)

my\_data['New\_installs'] = my\_data['Installs'].apply(cleanup\_installs)

1. Price field is a string and has $ symbol. Remove ‘$’ sign, and convert it to numeric.

def cleanup\_price(price):

if price == 'Everyone':

price = '0'

new\_price = price.replace('$','')

return float(new\_price)

my\_data['New\_Price'] = my\_data['Price'].apply(cleanup\_price)

5. Sanity checks:

1. Average rating should be between 1 and 5 as only these values are allowed on the play store. Drop the rows that have a value outside this range.

incorrect\_ratings = my\_data['Rating'] > 5

my\_data.drop(my\_data[incorrect\_ratings].index, axis=0, inplace = True)

1. Reviews should not be more than installs as only those who installed can review the app. If there are any such records, drop them.

incorrect\_reviews = my\_data['Reviews'] > my\_data['New\_installs']

my\_data.drop(my\_data[incorrect\_reviews].index,axis=0, inplace = True)

1. For free apps (type = “Free”), the price should not be >0. Drop any such rows.

incorrect\_price = (my\_data['Type'] == 'Free') & (my\_data['New\_Price'] > 0)

my\_data.drop(my\_data[incorrect\_price].index, axis=0, inplace = True)

5. Performing univariate analysis:

* Boxplot for Price

plt.figure(figsize = [25,8])

plt.boxplot(my\_data.New\_Price, vert=False)

plt.xlabel("Price",fontsize=20)

plt.ylabel("",fontsize=20)

# 

Are there any outliers? Think about the price of usual apps on Play Store.

*From the BoxPlot we can definitely see that the Prices above 50$ are obvious outliers.*

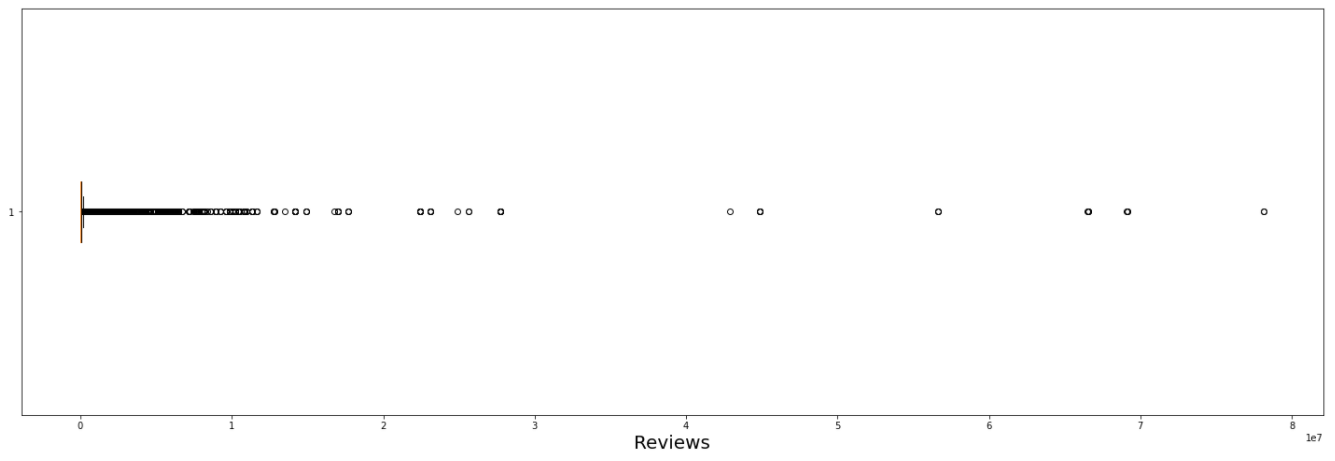
* Boxplot for Reviews

plt.figure(figsize = [25,8])

plt.boxplot(my\_data.Reviews,vert=False)

plt.xlabel("Reviews",fontsize=20)

plt.ylabel("",fontsize=20)



Are there any apps with very high number of reviews? Do the values seem right?

*Some apps have very high number of reviews and are outliers.*

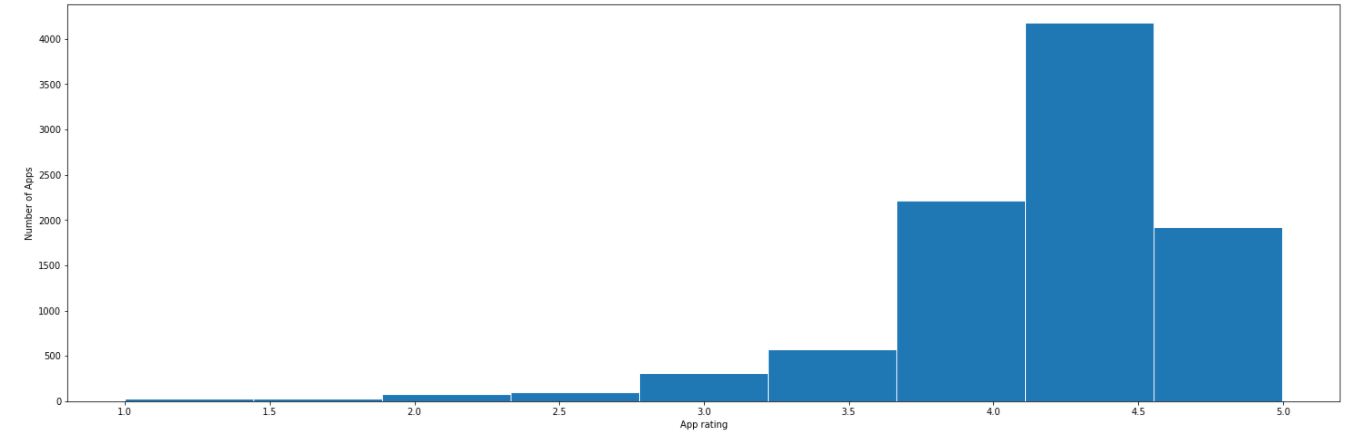
* Histogram for Rating

plt.figure(figsize = [25,8])

my\_data['Rating'].plot.hist(bins=9,rwidth=0.99)

plt.xlabel('App rating')

plt.ylabel('Number of Apps')



How are the ratings distributed? Is it more toward higher ratings?

*Majority of the apps have higher ratings (most between 4.0 to 5)*

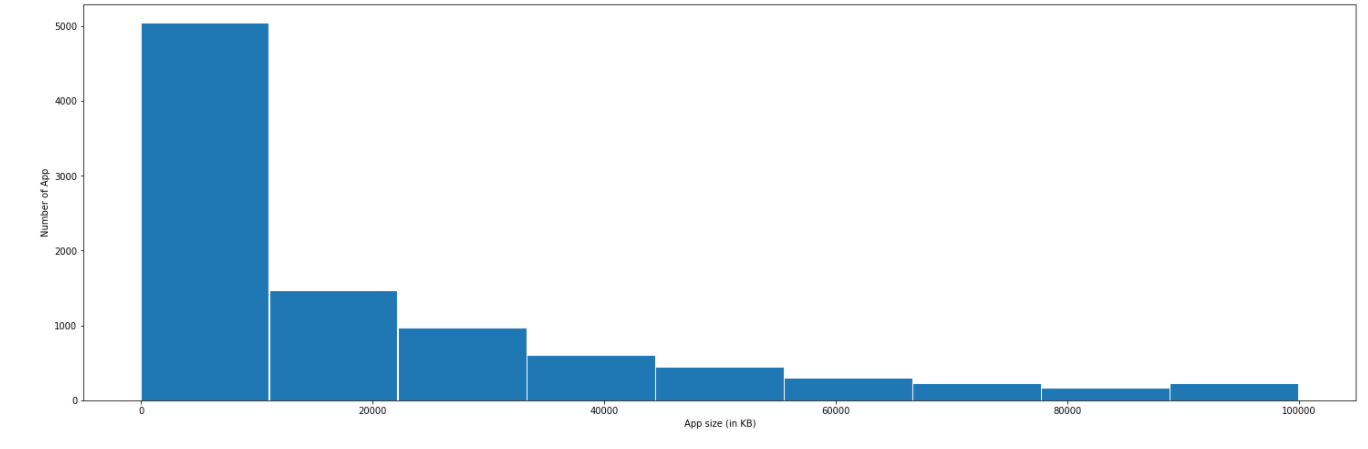
* Histogram for Size

plt.figure(figsize = [25,8])

my\_data['New\_Size'].plot.hist(bins=9,rwidth=0.99)

plt.xlabel('App size (in KB)')

plt.ylabel('Number of App')



Note down your observations for the plots made above. Which of these seem to have outliers?

*Most apps' size are below 20,000.*

6. Outlier treatment:

1. Price: From the box plot, it seems like there are some apps with very high price. A price of $200 for an application on the Play Store is very high and suspicious!
   1. Check out the records with very high price
      1. Is 200 indeed a high price?

*It seems that price between 0 and 50. We can assume that price above $50 is outlier.*

* 1. Drop these as most seem to be junk apps

my\_data.drop(my\_data[my\_data.New\_Price>200].index,axis=0,inplace=True)

1. Reviews: Very few apps have very high number of reviews. These are all star apps that don’t help with the analysis and, in fact, will skew it. Drop records having more than 2 million reviews.

\_data.drop(my\_data[my\_data.Reviews>2000000].index,axis=0,inplace=True)

1. Installs:  There seems to be some outliers in this field too. Apps having very high number of installs should be dropped from the analysis.
   1. Find out the different percentiles – 10, 25, 50, 70, 90, 95, 99

my\_data['New\_installs'].quantile([0.1, 0.25, 0.5, 0.70, 0.9, 0.95, 0.99])

0.10 1000.0

0.25 10000.0

0.50 500000.0

0.70 1000000.0

0.90 10000000.0

0.95 10000000.0

0.99 100000000.0

Name: New\_installs, dtype: float64

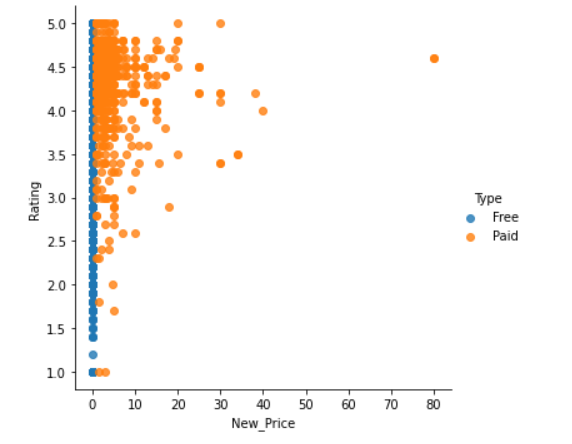
* 1. Decide a threshold as cutoff for outlier and drop records having values more than that - *10,000,000*

my\_data.drop(my\_data[my\_data.New\_installs>10000000].index,axis=0,inplace=True)

7. Bivariate analysis: Let’s look at how the available predictors relate to the variable of interest, i.e., our target variable rating. Make scatter plots (for numeric features) and box plots (for character features) to assess the relations between rating and the other features.

1. Make scatter plot/joinplot for Rating vs. Price

sns.lmplot(x="New\_Price",y="Rating",hue='Type',data=my\_data,fit\_reg=False)

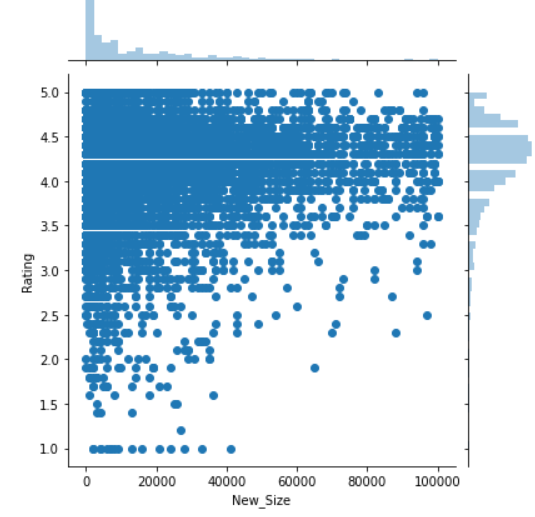


What pattern do you observe? Does rating increase with price?

*There is some correlation between price and rating but not very strong. Majority of the paid apps have a higher rating.*

1. Make scatter plot/joinplot for Rating vs. Size

sns.jointplot(x='New\_Size',y='Rating',data=my\_data,kind='scatter')

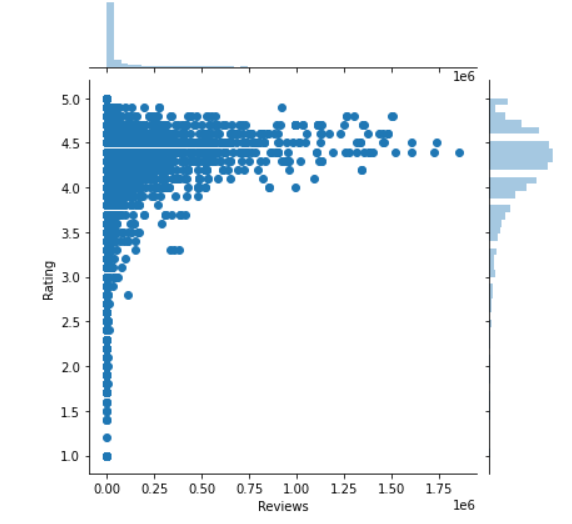


Are heavier apps rated better?

*Entries seem evenly distributed throughout sizes. The percentage of large apps having high ratings(>4 and rated better).*

1. Make scatter plot/joinplot for Rating vs. Reviews

sns.jointplot(x='Reviews',y='Rating',data=my\_data,kind='scatter')



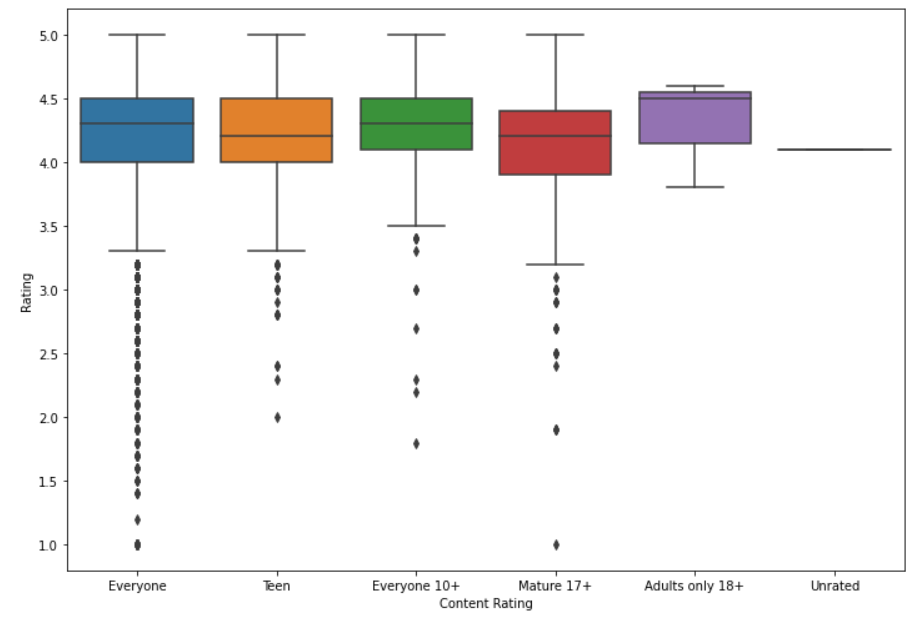
Does more review mean a better rating always?

*For higher number of reviews, the rating is always greater than 3.5*

1. Make boxplot for Rating vs. Content Rating

plt.figure(figsize=(12,8.27))

sns.boxplot(x="Content Rating",y="Rating",data=my\_data)



Is there any difference in the ratings? Are some types liked better?

*No visible changes between categories in relation to ratings.*

1. Make boxplot for Ratings vs. Category

plt.figure(figsize=(25,8.27))

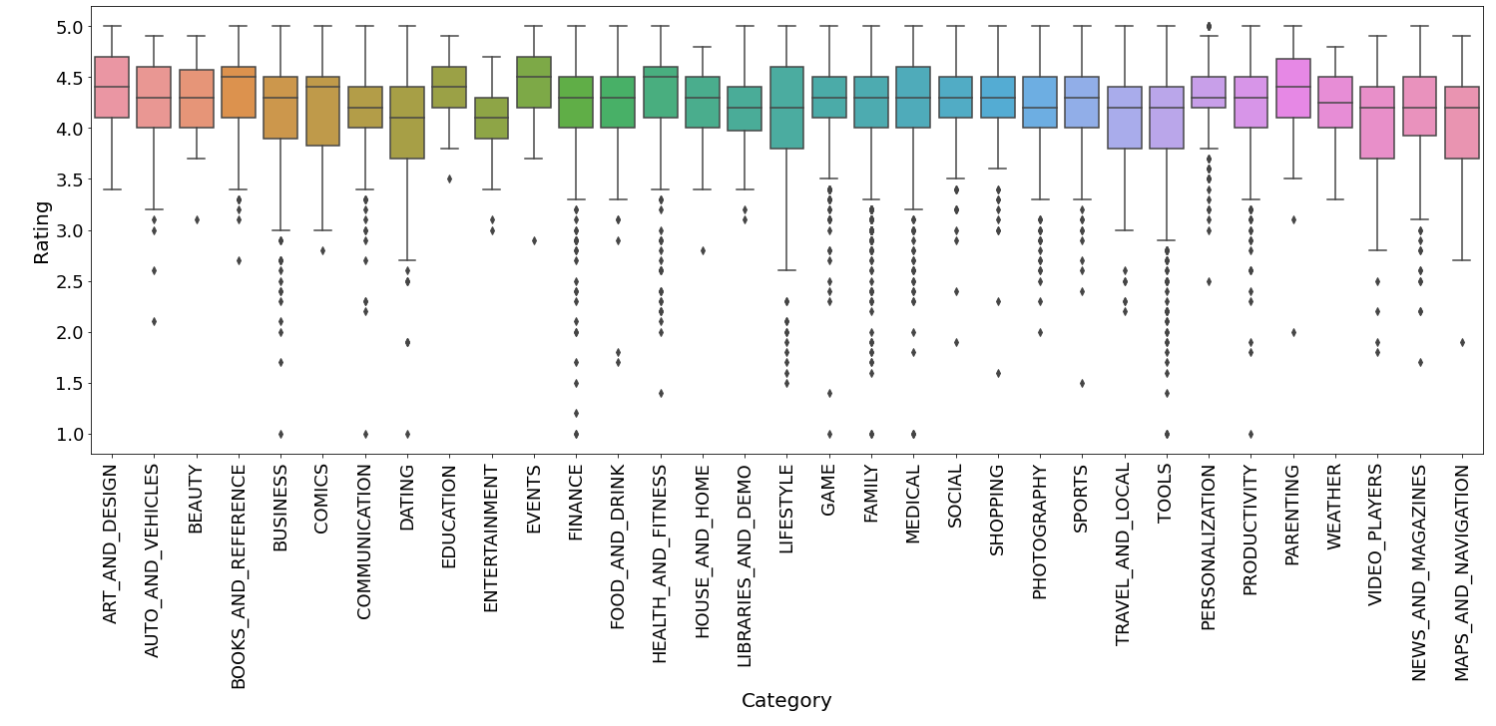
sns.boxplot(my\_data.Category,my\_data.Rating)

plt.xticks(fontsize=18,rotation='vertical')

plt.yticks(fontsize=18)

plt.xlabel("Category",fontsize=20)

plt.ylabel("Rating",fontsize=20)



Which genre has the best ratings?

*Events*

For each of the plots above, note down your observation.

8. Data preprocessing

For the steps below, create a copy of the dataframe to make all the edits. Name it inp1.

inp1 = my\_data.copy(deep = True)

1. Reviews and Install have some values that are still relatively very high. Before building a linear regression model, you need to reduce the skew. Apply log transformation (np.log1p) to Reviews and Installs.

my\_data.Reviews=my\_data.Reviews.apply(func=np.log1p)

my\_data.New\_installs=my\_data.New\_installs.apply(func=np.log1p)

1. Drop columns App, Last Updated, Current Ver, and Android Ver. These variables are not useful for our task.

inp1.drop(['App','Last Updated','Current Ver','Android Ver','Installs','Price','Size'], axis=1, inplace = True)

1. Get dummy columns for Category, Genres, and Content Rating. This needs to be done as the models do not understand categorical data, and all data should be numeric. Dummy encoding is one way to convert character fields to numeric. Name of dataframe should be **inp2**.

inp1.drop(['App','Last Updated','Current Ver','Android Ver','Installs','Price','Size'], axis=1, inplace = True)

9. Train test split  and apply 70-30 split. Name the new dataframes df\_train and df\_test.

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error as ms

X=inp2.iloc[:,1:].values

y=inp2.iloc[:,:1].values

10. Separate the dataframes into X\_train, y\_train, X\_test, and y\_test.

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.30, random\_state=1)

X\_train.shape,X\_test.shape

((5947, 156), (2549, 156))

11 . Model building

* Use linear regression as the technique
* Report the R2 on the train set

from sklearn.linear\_model import LinearRegression

lr=LinearRegression()

model=lr.fit(X\_train,y\_train)

model.score(X\_train, y\_train)

0.16739076320835045

12. Make predictions on test set and report R2.

y\_pred= model.predict(X\_test)

r2\_score(y\_test, y\_pred)

0.12859034252868462

*In our case R2 is very low 0.167 . The R2 should be large enough to explain the strength of the model. If R2 is low it means that the dependent variable is not explained any changes by the independent variables.*