



ataTrained

Keep Skilling, Keep Growing

Advertising Sales Channel Prediction

Submitted by:
Sarthak Gupta

Problem Statement:

When a company enters a market, the distribution strategy and channel it uses are keys to its success in the market, as well as market know-how and customer knowledge and understanding. Because an effective distribution strategy under efficient supply-chain management opens doors for attaining competitive advantage and strong brand equity in the market, it is a component of the marketing mix that cannot be ignored.

The distribution strategy and the channel design have to be right the first time. The case study of Sales channel includes the detailed study of TV, radio and newspaper channel. The predict the total sales generated from all the sales channel.

Hardware and Software Requirements and Tools Used –

PYTHON Jupyter Notebook:

Key Features:

An open-source solution that has simple coding processes and syntax so it's fairly easy to learn Integration with other languages such as C/C++, Java, PHP, C#, etc. Advanced analysis processes through machine learning and text mining.

Python is extremely accessible to code in comparison to other popular languages such as Java, and its syntax is relatively easy to learn making this tool popular among users that look for an open-source solution and simple coding processes. In data analysis, Python is used for data crawling, cleaning, modelling, and constructing analysis algorithms based on business scenarios. One of the best features is actually its user-friendliness: programmers don't need to remember the architecture of the system nor handle the memory – Python is considered a high-level language that is not subject to the computer's local processor.

Libraries and Packages used:

Matplotlib:

Matplotlib is a Python library that uses Python Script to write 2-dimensional graphs and plots. Often mathematical or scientific applications require more than single axes in a representation. This library helps us to build multiple plots at a time. You can, however, use Matplotlib to manipulate different characteristics of figures as well.

The task carried out is visualization of dataset i.e., nominal data, ordinal data, continuous data, heat map display distribution for correlation matrix and null values, boxplot distribution for checking outliers, scatter plot distribution for modelling approach, subplot distribution for analysis and comparison, feature importance and common importance features, line plot for prediction values vs actual values.

Numpy:

Numpy is a popular array – processing package of Python. It provides good support for different dimensional array objects as well as for matrices. Numpy is not only confined to providing arrays only, but it also provides a variety of tools to manage these arrays. It is fast, efficient, and really good for managing matrices and arrays.

The Numpy is used to managing matrices i.e., MAE, MSE and RMSE and arrays i.e., described the values of train test dataset.

Pandas:

Pandas is a python software package. It is a must to learn for data-science and dedicatedly written for Python language. It is a fast, demonstrative, and adjustable platform that offers intuitive data-structures. You can easily manipulate any type of data such as – structured or time-series data with this amazing package.

The Pandas is used to execute a Data frame i.e., test set.csv, train set.csv, skewness, co-efficient, predicted values of model approach, conclusion.

Scikit Learn:

Scikit learn is a simple and useful python machine learning library. It is written in python, C, and C++. However, most of it is written in the Python programming language. It is a free machine learning library. It is a flexible python package that can work in complete harmony with other python libraries and packages such as Numpy and Scipy.

Scikit learn library is used to import a pre-processing function i.e., power transform, ordinal encoder, minimax scaler, linear, random forest, decision tree, xgboost, k-nearest neighbours, r2 score, mean absolute error, mean squared error, train test split, grid search cv and ensemble technique.

DATA ANALYSIS:

It is a process of cleaning, analysing, transforming and modelling the raw data into meaningful insights which will further leads to the support into decision making.

There is different type of models through which one can find the relationship between the data, but for this we are going to use the Regression Model.

Regression Model:

The regression models are used to examine relationships between variables. Regression models are often used to determine which independent variables hold the most influence over dependent variables information that can be leveraged to make essential decision. The most traditional regression model is linear regression, decision tree regression, random forest regression, xgboost regression and knn-neighbours.

There are 4 main components of an analytics model, namely:

1. Data Component
2. Algorithm Component
3. Real World Component
4. Ethical Component

Data Preparation:

In this study, we will use a raw data of a company sales revenue. The company uses data analytics to find the profit margins in which type of ADVERTISEMENT business. The data is provided in the csv file Advertising.csv.

Data Description:

The dataset contains 200 rows and 4 columns.

Total Numbers of Rows and Column

```
In [6]: 1 Advertising.shape
```

```
Out[6]: (200, 4)
```

Having the column names,

Checking All Column Names

```
In [7]: 1 Advertising.columns
```

```
Out[7]: Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

Then we move to see statistical information about the non-numerical columns in our dataset:

```
In [14]: 1 Advertising.describe()
```

```
Out[14]:
```

	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

We can get a lot of information about our dataset variables from the table like the mean value, count value, min & max value of particular column and percentile of the column.

Data Cleaning

Dealing with Missing Values:

We should deal with the problem of missing values because some machine learning models don't accept data with missing values. Firstly, let's see the number of missing values in our dataset.

Checking for Null Values

```
In [9]: 1 Advertising.isnull().sum()
```

```
Out[9]: TV          0
radio         0
newspaper     0
sales         0
dtype: int64
```

In the above table we got, count represents the number of non-null values in each column. As there are no NULL values found in the data, we will further move to describing the dataset through heatmap.

DATASET's HEATMAP:



Also Check the unique values in all the columns:

```
In [16]: 1 Advertising.nunique()
```

```
Out[16]: TV          190  
radio        167  
newspaper    172  
sales        121  
dtype: int64
```

As we can see from above image, there are a total of 190 unique values in TV, 172 unique values in newspaper, 167 unique values in radio, 121 unique values in sales.

Data Visualization:

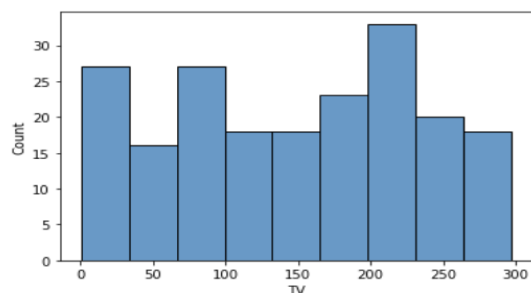
Data Visualization helps in analyse the data with the help of visualization by plotting graphs in different formats.

It also eases for one to understand the data whether there is any kind of non-valuable data is present that will further lead to affect the model training.

I. Using Histogram Plot to show the graph in between the count & TV ads.

```
In [17]: 1 sns.histplot(x=Advertising['TV'])
```

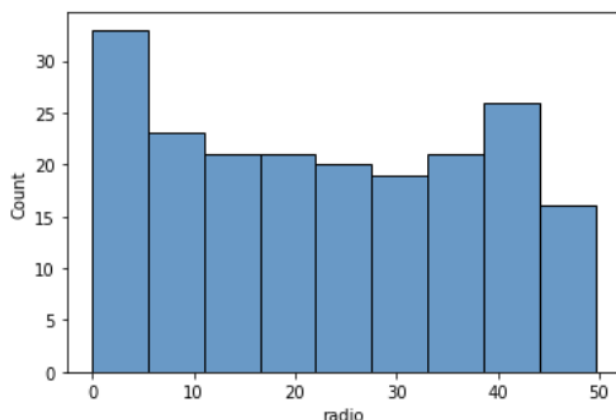
```
Out[17]: <AxesSubplot:xlabel='TV', ylabel='Count'>
```



between the count & radio ads.

I

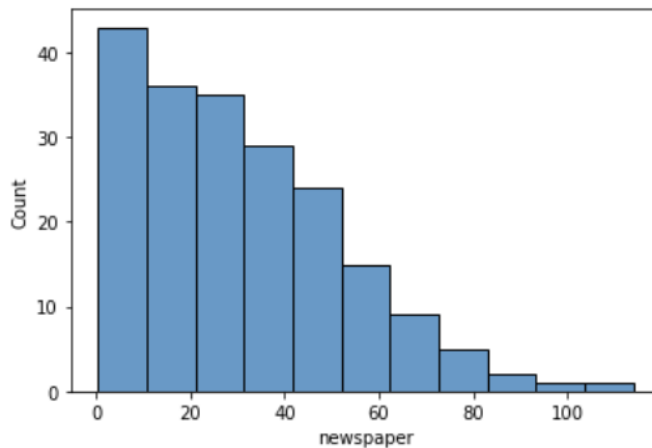
```
Out[18]: <AxesSubplot:xlabel='radio', ylabel='Count'>
```



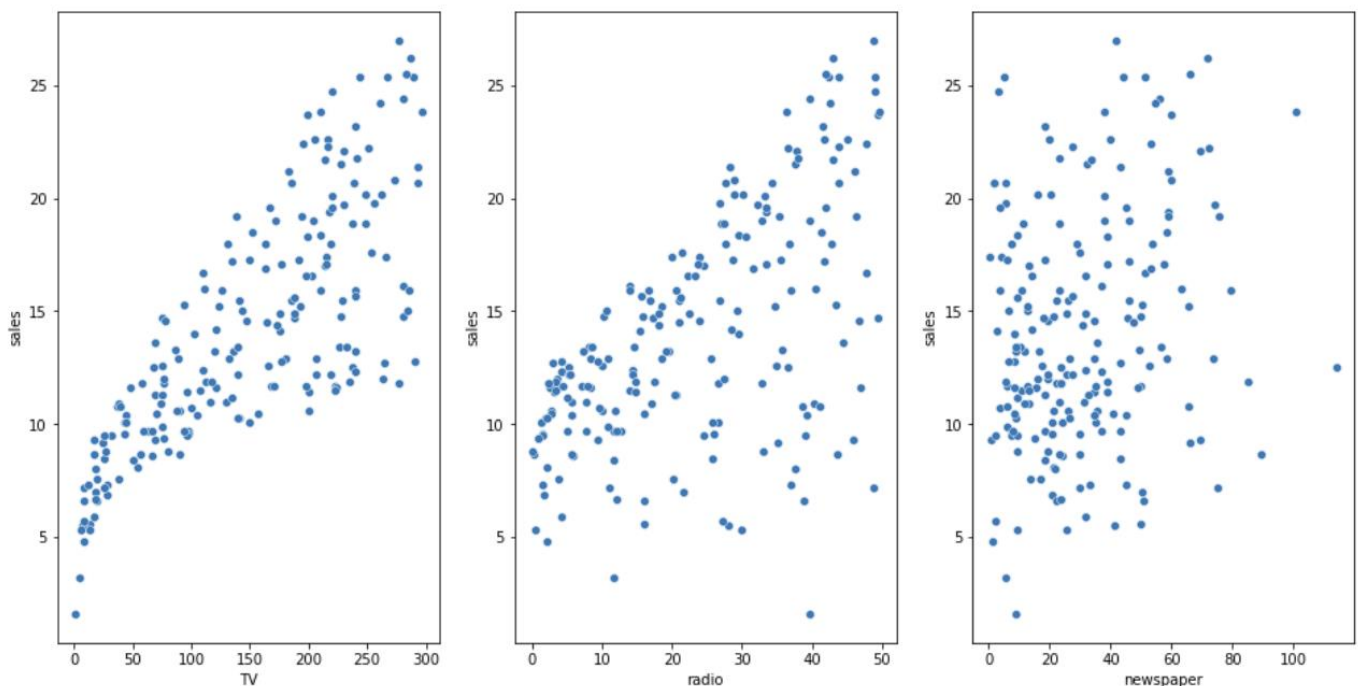
III. Using Histogram Plot to show the graph in between the count & newspaper ads.

```
In [19]: 1 sns.histplot(x=Advertising['newspaper'])
```

```
Out[19]: <AxesSubplot:xlabel='newspaper', ylabel='Count'>
```



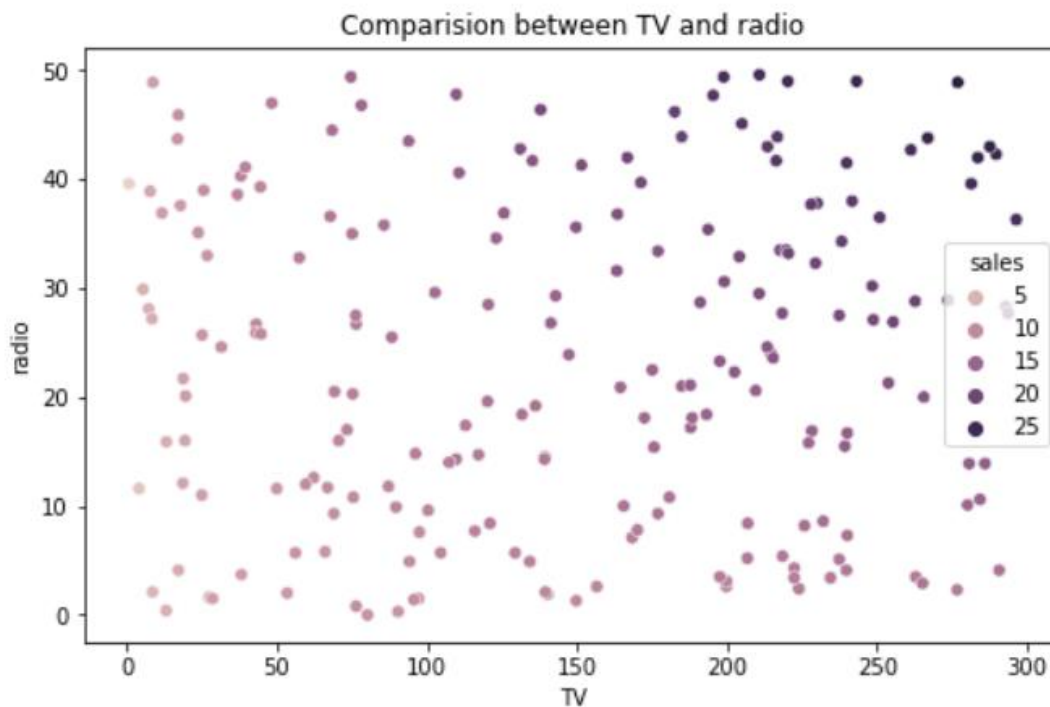
Also showing the data for using the Scatter Plotting:



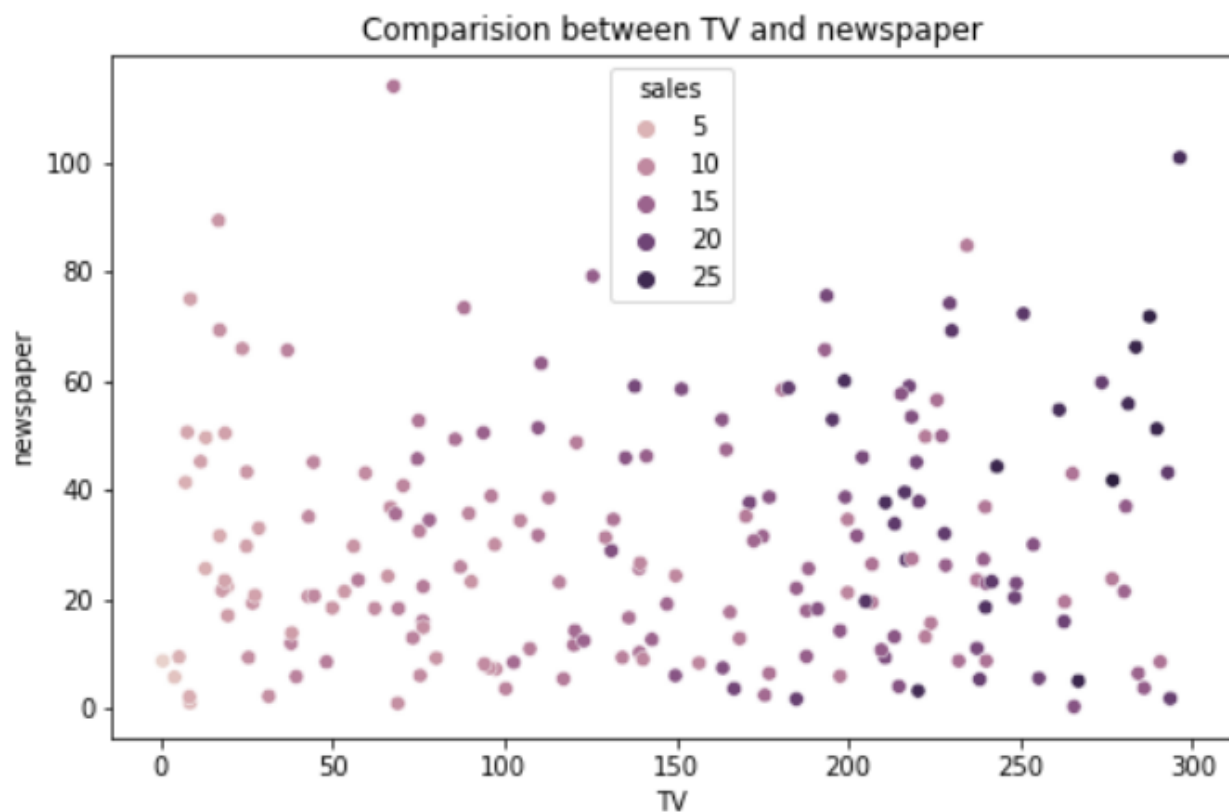
As we can observe that there is a linear relationship between the features TV, radio with the sales but on the other hand there is no any type of relation between newspaper & sales which will further leads to the loss as there is much of the investment in newspaper type of advertisement. So we will further work with TV and radio methods.

Now we will further check the relation between two types of methods in terms of sales:

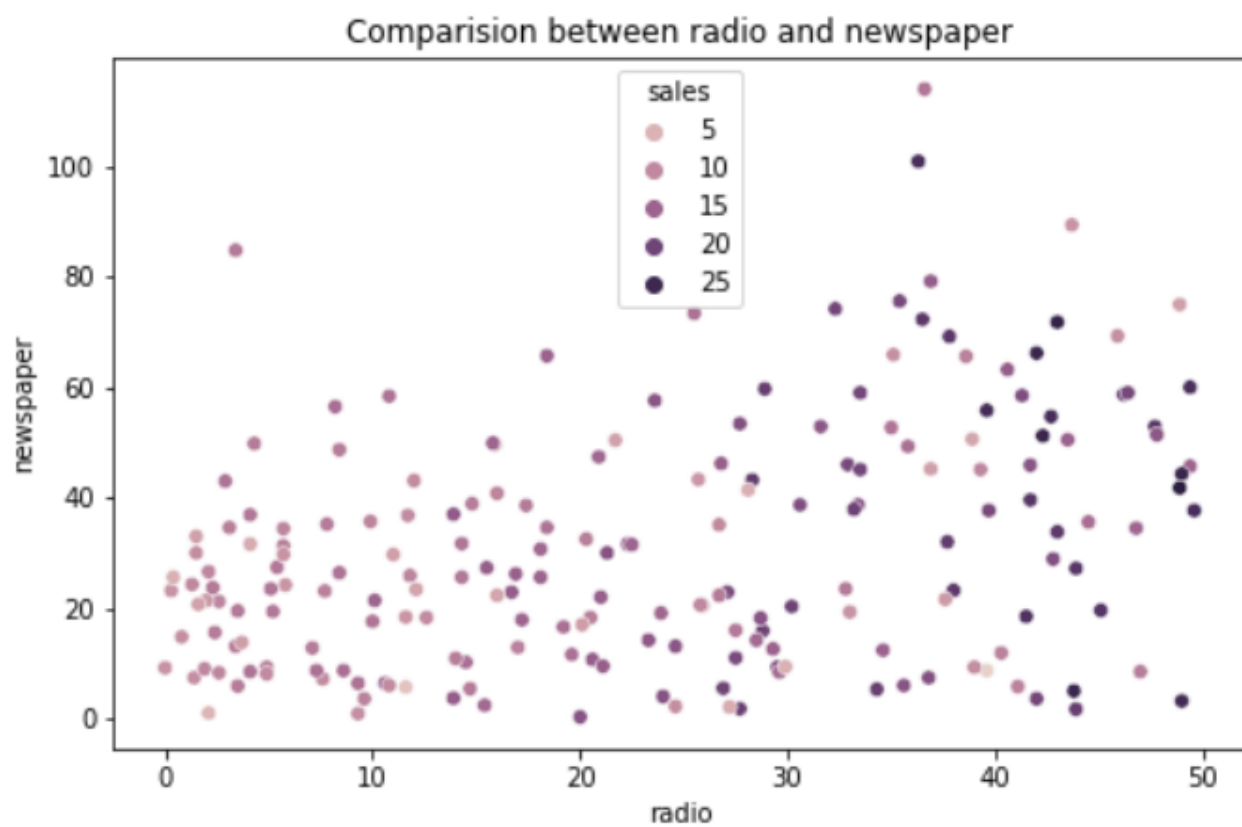
1) TV & radio with sales -



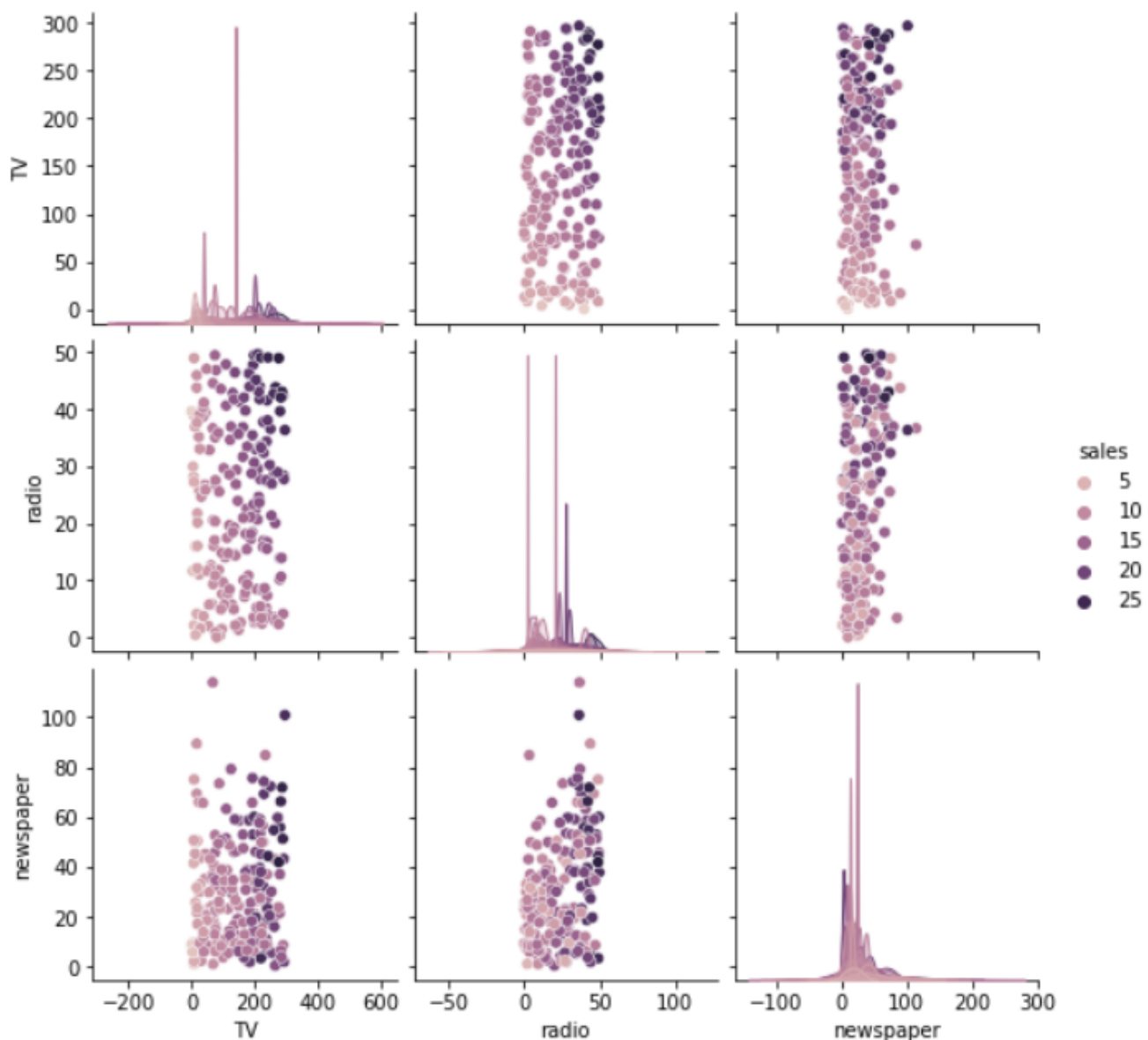
2) TV & newspaper with sales -



3) Radio & newspaper with sales -

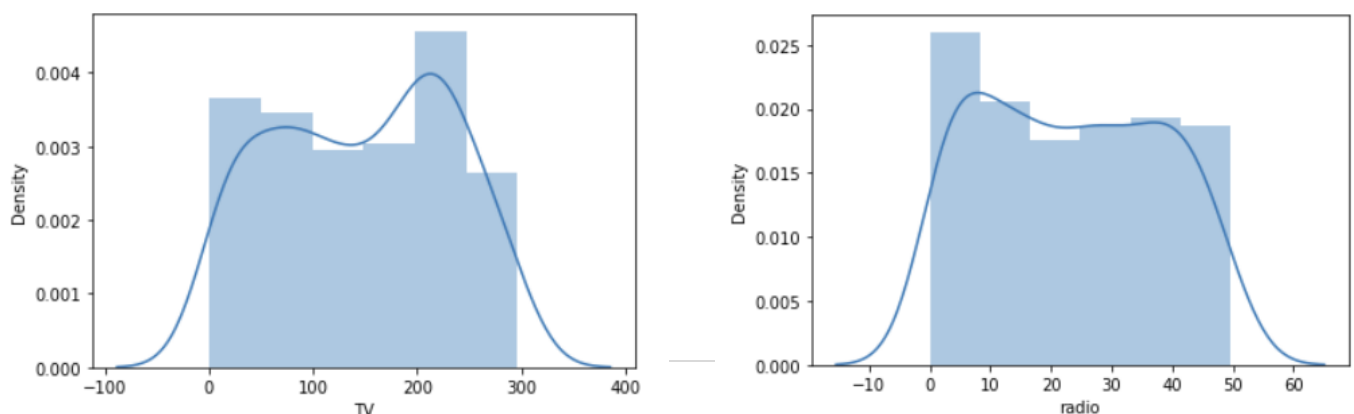


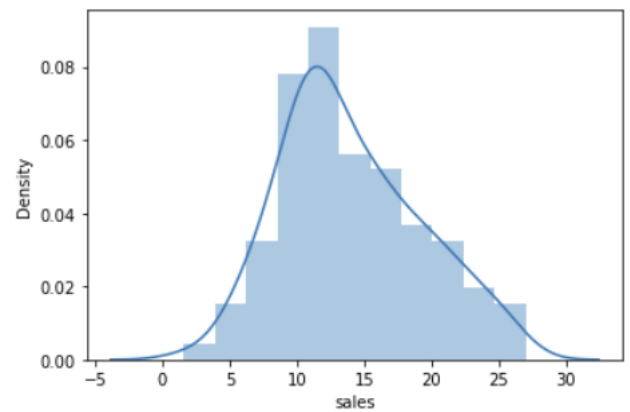
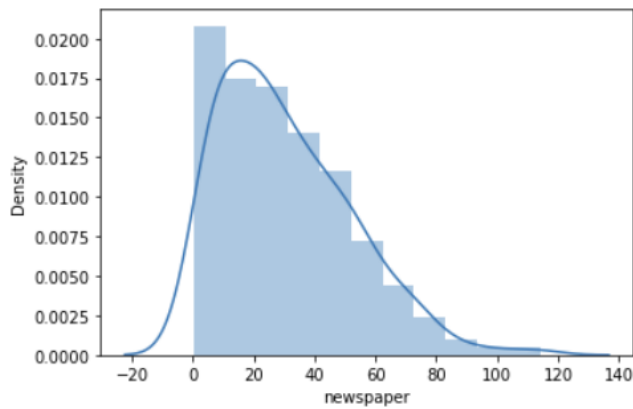
Pair plotting:



Through the pair plot one can check & understand the plotting between columns of the dataset & it could also help to analyse that which of the methods will lead to improve profits in advertisement business.

Distribution Plot:





Through the distribution plotting we can see that newspaper data is right skewed due to which we will be removing in further below steps.

Checking Co-relations:

In [29]: `1 Advertising.corr()`

Out[29]:

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

This shows the correlation between the independent values & dependent value.

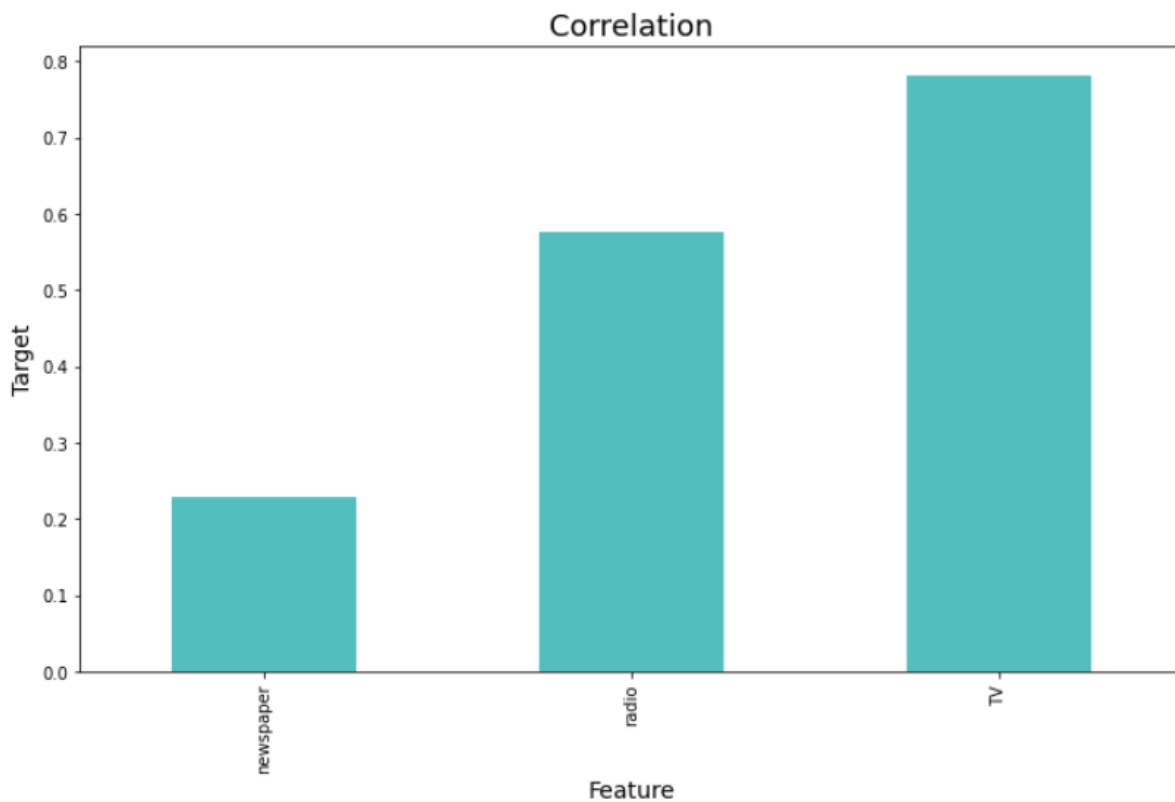
1. TV has 78% correlation with the target column which can be considered as strong bond.
2. Radio has 58% correlation with the target column which can be considered as good bond
3. Newspaper has 23% correlation with the target column which can be considered as weak bond

Maximum & minimum corelation are TV & newspaper respectively.

```

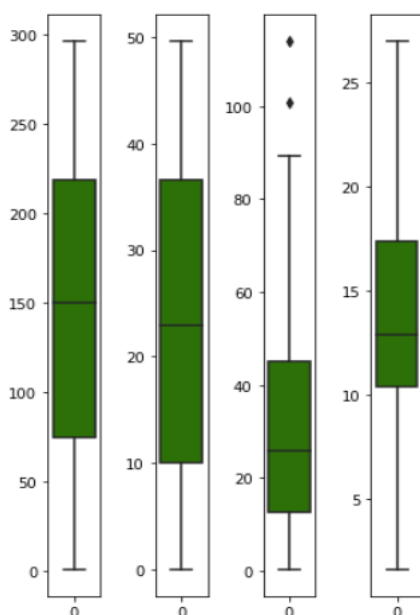
3 plt.figure(figsize=(12,7))
4 Advertising.corr()['sales'].sort_values(ascending=True).drop(['sales']).plot(kind='bar',color='c')
5 plt.xlabel('Feature',fontsize=14)
6 plt.ylabel('Target',fontsize=14)
7 plt.title('Correlation',fontsize=18)
8 plt.show()

```



Target variable is positively correlated with all the features.

🔍 Checking Outliers:



While checking for outliers we have only found that there are outliers present in only newspaper feature, others have no outliers so we are going to remove the outliers from the newspaper columns using Z-score method.

```
In [35]: 1 variables = Advertising[['newspaper']]
          2
          3 z=np.abs(zscore(variables))
          4
          5 # Creating new dataframe
          6 Advertising_Sales = Advertising[(z<3).all(axis=1)]
          7 Advertising_Sales.head()
```

Out[35]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

In this first we have the Z-score of the dataset & then made a dataset having the Z-score value more than 3 standard deviation and then we will be dropping the data so that while training and testing our model should be with the filtered & meaningful data.

Further we will check the percentage loss of the data –

```
In [37]: 1 print("Old DataFrame data in Rows and Column:",Advertising.shape)
          2 print("New DataFrame data in Rows and Column:",Advertising_Sales.shape)
          3 print("Total Dropped rows:",Advertising.shape[0]-Advertising_Sales.shape[0])
```

```
Old DataFrame data in Rows and Column: (200, 4)
New DataFrame data in Rows and Column: (198, 4)
Total Dropped rows: 2
```

Percentage Data Loss using Zscore

```
In [38]: 1 loss_percent=(200-198)/200*100
          2 print(loss_percent,"%")
```

```
1.0 %
```

So, we can further say that only 1.0% of the data is loss which is negligible so we are good for further steps of EDA.

Skewness:

Skewness is a measure of symmetry in a distribution. Actually, it's more correct to describe it as a measure of lack of symmetry. A standard normal distribution is perfectly symmetrical and has zero skew. Therefore, we need a way to calculate how much the distribution is skewed.

Checking Skewness:

As earlier we have seen while distribution plotting that there is the right skewness is present in the newspaper feature so we will further move to remove the skewness.

```
In [42]: 1 Advertising_Sales.skew()
```

```
Out[42]: TV          -0.082332
         radio       0.114842
         newspaper   0.650112
         sales       0.407130
         dtype: float64
```

Skewness threshold taken is +/-0.65. "newspaper" Column have only skewness. So, let's remove the skewness.

We will be removing skewness with the help of **yeo-johnson method** -

Import libraries PowerTransformer from sklearn

```
In [44]: 1 from sklearn.preprocessing import PowerTransformer
```

```
In [45]: 1 variable=["newspaper"]
         2 scaler = PowerTransformer(method='yeo-johnson')
         3 Advertising_Sales[variable] = scaler.fit_transform(Advertising_Sales[variable].values)
         4 Advertising_Sales[variable].head()
```

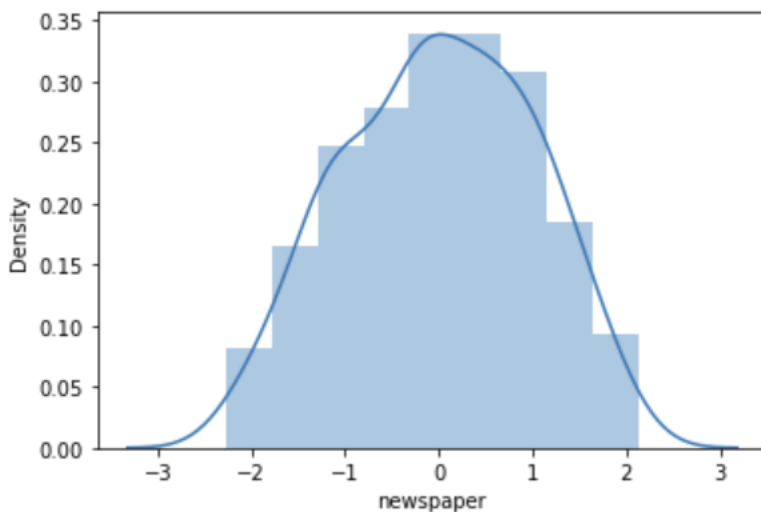
```
Out[45]: newspaper
0      1.604196
1      0.831044
2      1.607052
3      1.283768
4      1.280622
```

Checking again skewness -

```
In [46]: 1 #checking skewness after removal
         2 Advertising_Sales.skew()
```

```
Out[46]: TV          -0.082332
         radio       0.114842
         newspaper   -0.101288
         sales       0.407130
         dtype: float64
```

As we can see the skewness of newspaper has been reduced from **0.65** to **-0.10** (approx.), and we will again plot to final check & confirm the removal.



We can see that the data is now normally distributed.

Data Pre-processing:

Splitting Data –

We will now split the dependent value & assign it into 'y' & independent data into 'x'.

```
In [48]: 1 x = Advertising_Sales.drop("sales", axis=1)
         2 y = Advertising_Sales["sales"]
```

```
In [49]: 1 x
```

Out[49]:

	TV	radio	newspaper
0	230.1	37.8	1.604196
1	44.5	39.3	0.831044
2	17.2	45.9	1.607052
3	151.5	41.3	1.283768
4	180.8	10.8	1.280622
...
195	38.2	3.7	-0.687435
196	94.2	4.9	-1.146090
197	177.0	9.3	-1.314437
198	283.6	42.0	1.517395
199	232.1	8.6	-1.091084

198 rows × 3 columns

```
In [50]: 1 y
```

Out[50]:

```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
...
195    7.6
196    9.7
197   12.8
198   25.5
199   13.4
```

Name: sales, Length: 198, dtype: float64

Above one can see the length of both dependent & independent value is same which further will conclude to move ahead.

Scaling data with Standard Scaler –

In order to make all algorithms work properly with our data, a way to normalize the input features/variables is the Min-Max scaler. By doing so, all features will be transformed into the range [0,1] meaning that the minimum and maximum value of a feature/variable is going to be 0 and 1, respectively.

```
In [51]: 1 from sklearn.preprocessing import StandardScaler
```

```
In [52]: 1 scaler = StandardScaler()  
2 x = pd.DataFrame(scaler.fit_transform(x), columns=x.columns)  
3 x
```

Out[52]:

	TV	radio	newspaper
0	0.978697	0.989521	1.604196
1	-1.199012	1.090705	0.831044
2	-1.519332	1.535913	1.607052
3	0.056456	1.225616	1.283768
4	0.400243	-0.831784	1.280622
...
193	-1.272932	-1.310720	-0.687435
194	-0.615864	-1.229773	-1.146090
195	0.355657	-0.932968	-1.314437
196	1.606431	1.272836	1.517395
197	1.002164	-0.980187	-1.091084

198 rows × 3 columns

The data is scaled.

Creating Model:

Import the libraries which will be used in train & test of the model –

1. from sklearn.model_selection import train_test_split, cross_val_score
2. from sklearn.linear_model import LinearRegression
3. from sklearn.metrics import roc_curve, auc, roc_auc_score, r2_score, classification_report, mean_absolute_error, mean_squared_error
4. from sklearn.ensemble import RandomForestRegressor
5. from sklearn.tree import DecisionTreeRegressor
6. from sklearn.neighbors import KNeighborsRegressor as KNN
7. from sklearn.svm import SVR
8. from sklearn.linear_model import SGDRegressor
9. from sklearn.metrics import classification_report


```
In [53]: 1 from sklearn.model_selection import train_test_split, cross_val_score
2 from sklearn.linear_model import LinearRegression
3 from sklearn.metrics import roc_curve, auc, roc_auc_score, r2_score, classification_report, mean_absolute_error, mean_squared_error
4 from sklearn.ensemble import RandomForestRegressor
5 from sklearn.tree import DecisionTreeRegressor
6 from sklearn.neighbors import KNeighborsRegressor as KNN
7 from sklearn.svm import SVR
8 from sklearn.linear_model import SGDRegressor
9 from sklearn.metrics import classification_report
```

```
In [54]: 1 maxAccu=0
2 maxRS=0
3 for i in range(1,200):
4     x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state=i)
5     mod = LinearRegression()
6     mod.fit(x_train, y_train)
7     pred = mod.predict(x_test)
8     acc = r2_score(y_test, pred)
9     if acc>maxAccu:
10         maxAccu = acc
11         maxRS = i
12 print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

Best accuracy is 0.9358346827439895 on Random_state 90

Creating train-test-split –

```
In [55]: 1 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state = maxRS)
```

```
In [56]: 1 x_train.shape,y_train.shape, x_test.shape,y_test.shape
```

```
Out[56]: ((138, 3), (138,), (60, 3), (60,))
```

```
In [57]: 1 x.shape, y.shape
```

```
Out[57]: ((198, 3), (198,))
```

We have split the data into the training & testing data.

This will further use to train the model & test the trained model through which one could be able to check whether the prepared model is working with accuracy or not.

Regression Algorithms:

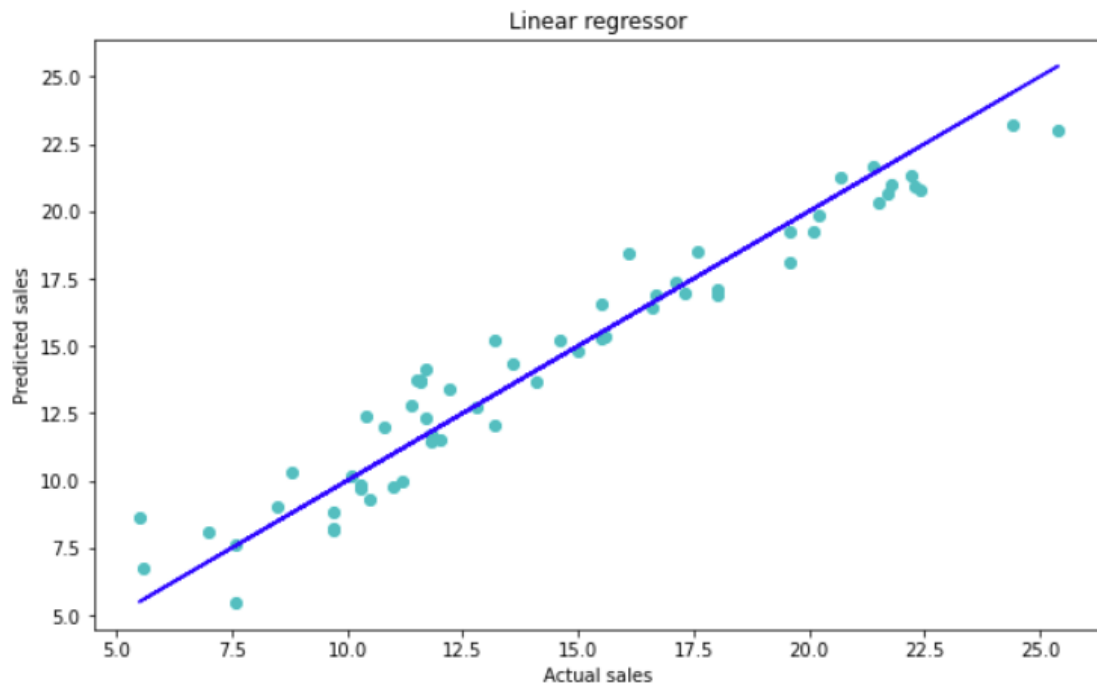
Linear Regression –

This technique models the relationship between the target variable and the independent variables (predictors). It fits a linear model with coefficients to the data in order to minimize the residual sum of squares between the target variable in the dataset, and the predicted values by the linear approximation.

```
In [58]: 1 LR = LinearRegression()
2 LR.fit(x_train,y_train)
3
4 # prediction
5 pred_LR=LR.predict(x_test)
6 print('R2_score:',r2_score(y_test,pred_LR))
7 print('Mean abs error:',mean_absolute_error(y_test, pred_LR))
8 print('Mean squared error:',mean_squared_error(y_test, pred_LR))
```

R2_score: 0.9358346827439895
Mean abs error: 1.0293867863083985
Mean squared error: 1.579629622636131

```
In [59]: 1 plt.figure(figsize=(10,6))
2 plt.scatter(x=y_test,y=pred_LR,color='c')
3 plt.plot(y_test,y_test,color='b')
4 plt.xlabel("Actual sales")
5 plt.ylabel("Predicted sales")
6 plt.title("Linear regressor")
7 plt.show()
```



There is a good accuracy

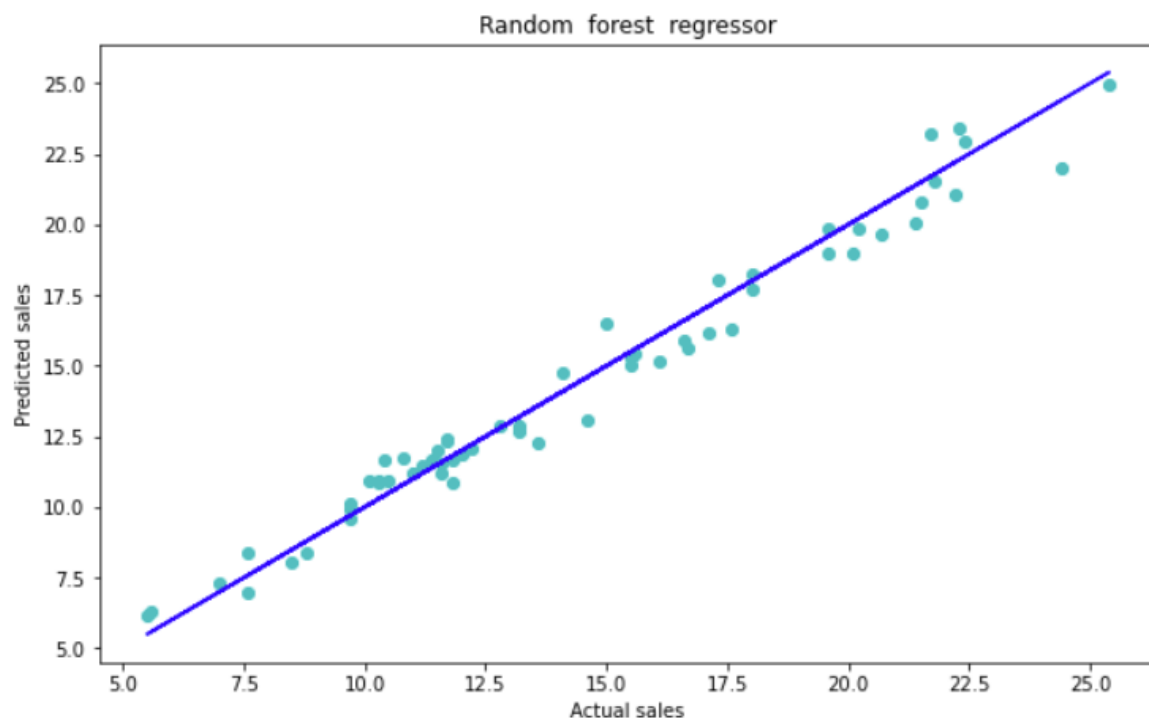
Random Forest Regressor –

Bagging is an ensemble method where many base models are used with a randomized subset of data to reduce the variance of the base model.

```
In [60]: 1 RFR=RandomForestRegressor(n_estimators=200, random_state=90)
2 RFR.fit(x_train,y_train)
3
4 # prediction
5 pred_RFR=RFR.predict(x_test)
6 print('R2_Score:',r2_score(y_test,pred_RFR))
7 print('Mean abs error:',mean_absolute_error(y_test, pred_RFR))
8 print('Mean squared error:',mean_squared_error(y_test, pred_RFR))
```

```
R2_Score: 0.9739854190521797
Mean abs error: 0.6575999999999994
Mean squared error: 0.6404301333333344
```

```
In [61]: 1 plt.figure(figsize=(10,6))
2 plt.scatter(x=y_test,y=pred_RFR,color='c')
3 plt.plot(y_test,y_test,color='b')
4 plt.xlabel("Actual sales")
5 plt.ylabel("Predicted sales")
6 plt.title("Random forest regressor")
7 plt.show()
```



There is a good accuracy

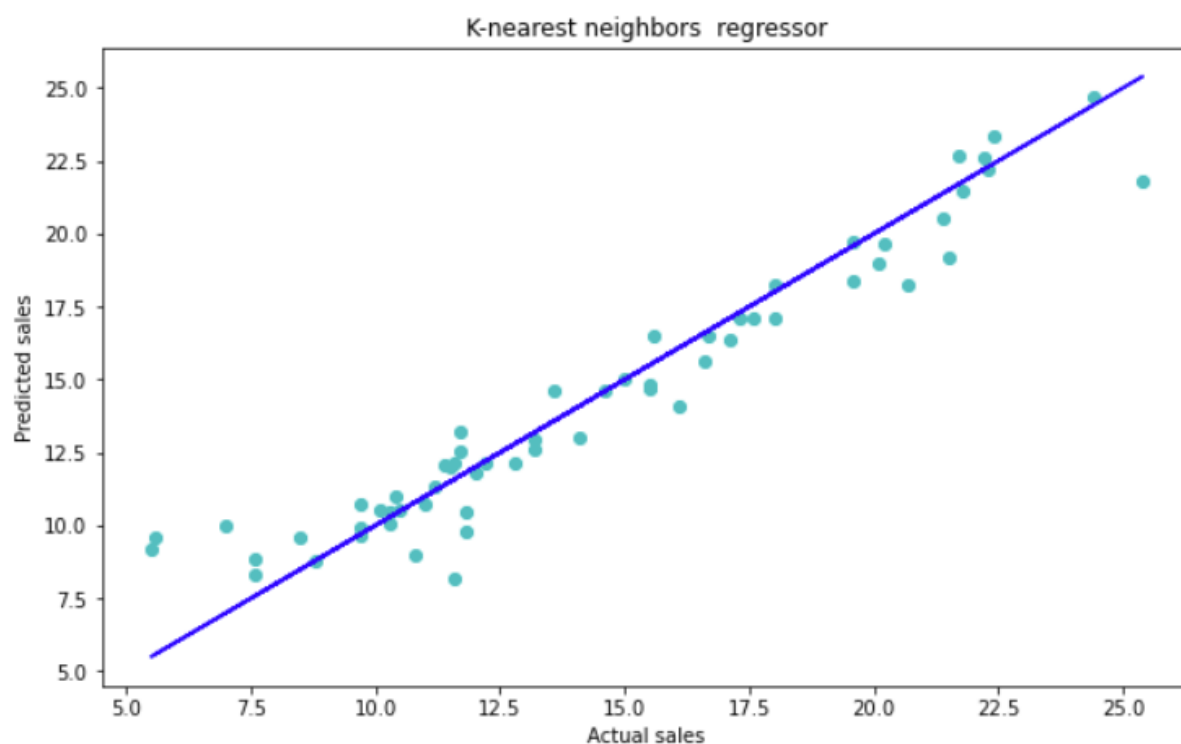
KNN Regressor –

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems.

```
In [62]: 1 knn=KNN()
2 knn.fit(x_train,y_train)
3
4 #prediction
5 pred_knn=knn.predict(x_test)
6 print('R2_Score:',r2_score(y_test,pred_knn))
7 print('Mean abs error:',mean_absolute_error(y_test, pred_knn))
8 print('Mean squared error:',mean_squared_error(y_test, pred_knn))
```

```
R2_Score: 0.925394255631917
Mean abs error: 0.9446666666666669
Mean squared error: 1.8366533333333334
```

```
In [63]: 1 plt.figure(figsize=(10,6))
2 plt.scatter(x=y_test,y=pred_knn,color='c')
3 plt.plot(y_test,y_test,color='b')
4 plt.xlabel("Actual sales")
5 plt.ylabel("Predicted sales")
6 plt.title("K-nearest neighbors regressor")
7 plt.show()
```



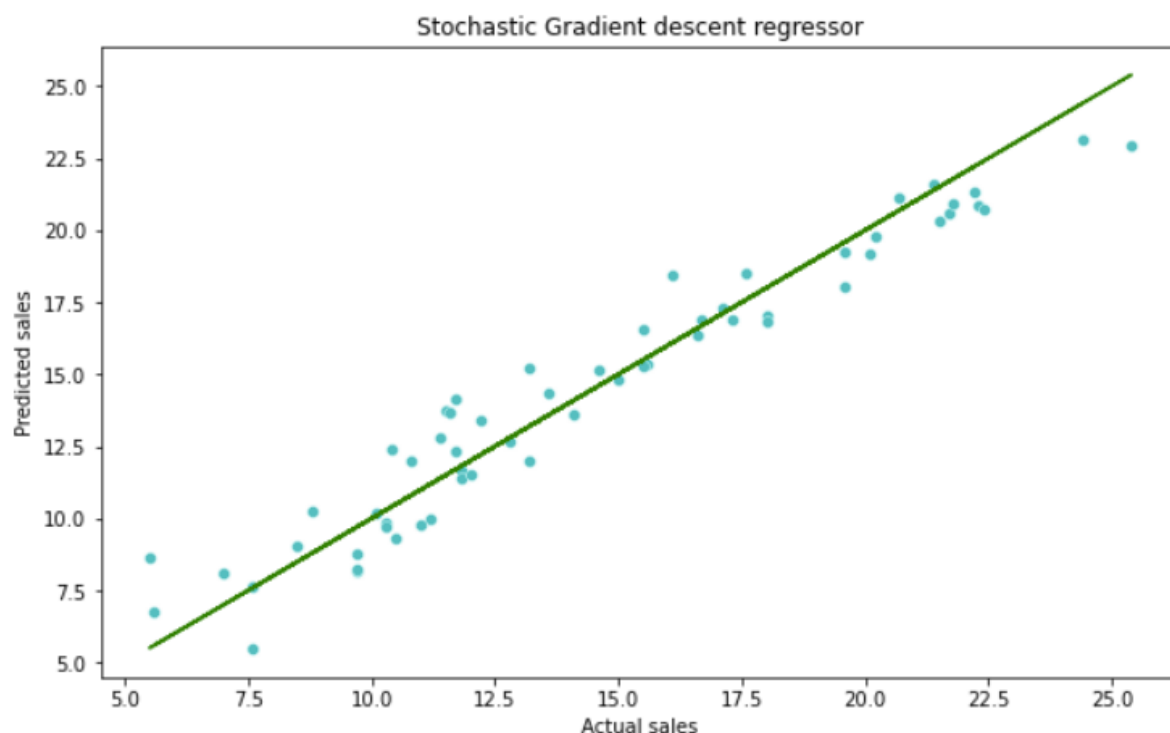
There is a good accuracy

Stochastic Gradient Descent Regressor –

```
In [64]: 1 SGD=SGDRegressor(epsilon=1, max_iter=500, alpha=0.001,fit_intercept=True)
2 SGD.fit(x_train,y_train)
3
4 #prediction
5 pred_SGD=SGD.predict(x_test)
6 print('R2_Score:',r2_score(y_test,pred_SGD))
7 print('Mean abs error:',mean_absolute_error(y_test, pred_SGD))
8 print('Mean squared error:',mean_squared_error(y_test, pred_SGD))
```

```
R2_Score: 0.9350460429234514
Mean abs error: 1.0400377410303432
Mean squared error: 1.599044454127449
```

```
In [65]: 1 plt.figure(figsize=(10,6))
2 sns.scatterplot(x=y_test,y=pred_SGD,color='c')
3 plt.plot(y_test,y_test,color='g')
4 plt.xlabel("Actual sales")
5 plt.ylabel("Predicted sales")
6 plt.title("Stochastic Gradient descent regressor")
7 plt.show()
```



Checking Cross Validation Score –

Checking CV score for Linear Regression

```
In [66]: 1 print('The CV score for Linear_regression is :',cross_val_score(LR,x,y,cv=9).mean())
```

The CV score for Linear_regression is : 0.8812869214578534

Checking cv score for Random Forest Regression

```
In [67]: 1 print("The CV score for the Random forest regressor is:",cross_val_score(RFR,x,y,cv=9).mean())
```

The CV score for the Random forest regressor is: 0.9781945175791701

Checking cv score for KNN Regression

```
In [68]: 1 print('The CV score for the KNN regressor is :',cross_val_score(knn,x,y,cv=9).mean())
```

The CV score for the KNN regressor is : 0.9319585778098676

Checking cv score for SGD Regression

```
In [69]: 1 print('The CV score for the SGD regressor is :',cross_val_score(SGD,x,y,cv=9).mean())
```

The CV score for the SGD regressor is : 0.8816646138550672

```
In [70]: 1 score= pd.DataFrame({'CV_RFR':0.9772000164869434, 'R2_score_RFR':0.9736204897912449}, index=['0'])
2 score
```

Out[70]:

	CV_RFR	R2_score_RFR
0	0.9772	0.97362

Hyper Parameter Tuning for Best Model using GridsearchCV:

Hyper parameters are crucial as they control the overall behaviour of a machine learning model. The ultimate goal is to find an optimal combination of hyper parameters that minimizes a predefined loss function to give better results.

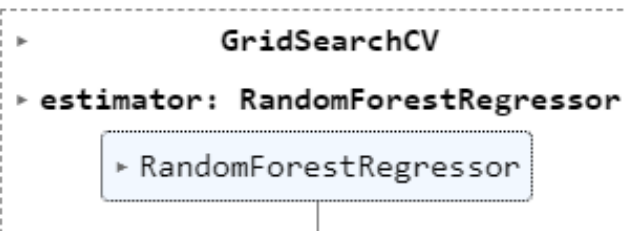
```
In [71]: 1 from sklearn.model_selection import GridSearchCV
```

```
In [72]: 1 parameters = {'criterion':['mse', 'mae'],  
2                  'max_features':['auto', 'sqrt', 'log2'],  
3                  'n_estimators':[0,200],  
4                  'max_depth':[2,4,6]}
```

```
In [73]: 1 GCV = GridSearchCV(RandomForestRegressor(),parameters,cv=6)
```

```
In [74]: 1 GCV.fit(x_train,y_train)
```

```
Out[74]:
```



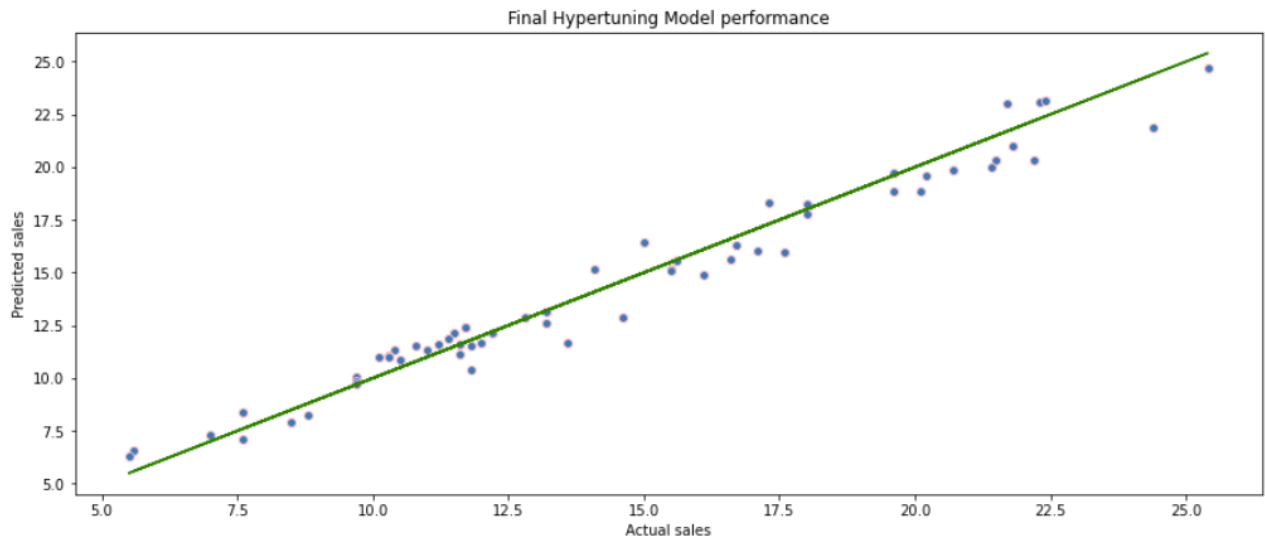
```
  ▸ GridSearchCV  
  ▸ estimator: RandomForestRegressor  
    ▸ RandomForestRegressor
```

```
In [75]: 1 GCV.best_params_
```

```
Out[75]: {'criterion': 'mse',  
          'max_depth': 6,  
          'max_features': 'auto',  
          'n_estimators': 200}
```

Final Model Graph:

```
In [77]: 1 plt.figure(figsize=(15,6))
2 plt.scatter(x=y_test,y=pred_GCV, edgecolors='pink', linewidth=1, cmap='Set3')
3 plt.plot(y_test,y_test,color='g')
4 plt.xlabel("Actual sales")
5 plt.ylabel("Predicted sales")
6 plt.title(" Final Hypertuning Model performance ")
7 plt.show()
```



We are getting R2 score as 97% which is good.

Saving the Model:

```
In [78]: 1 import pickle
2 filename='Titanic_Train_Project.pickle'
3 pickle.dump(GCV,open(filename,'wb'))
4 loaded_model = pickle.load(open(filename, 'rb'))
5 loaded_model.predict(x_test)

Out[78]: array([17.53548339,  6.44919815, 23.37722983, 19.78498554, 15.62593866,
 20.90388543, 20.92603583, 12.74988172, 23.14440741,  8.47911993,
 23.2711215 , 25.03796416, 11.1348593 , 19.87474014, 22.23445878,
  7.2255172 , 11.944737 , 11.42267088, 15.52682302, 10.73674247,
  7.00282752, 14.92715873, 11.02177822, 11.1330617 , 20.089959 ,
 12.02224645, 11.56961136, 10.04405239, 11.09577986, 21.6074782 ,
 13.04447612, 10.94056163, 15.95175441,  6.27626321,  9.87470493,
 11.60394822,  9.72509126, 12.52018216, 15.85963095, 14.90911676,
 18.91871318, 12.12808902, 13.03829553, 16.56505384, 11.73250438,
 18.16621266, 16.06921834, 15.16463978, 18.95752091, 19.9088346 ,
 11.72029387, 12.95639107, 11.64741699,  8.2774614 , 18.15031285,
 15.07229276,  8.04823411, 11.71000833, 12.32556404, 11.31945225])
```

Compare Predicted & Original Values:

```
In [79]: 1 a =np.array(y_test)
2 predicted=np.array(GCV.predict(x_test))
3 Sales_Adv=pd.DataFrame({'Original':a, 'Predicted':predicted}, index=range(len(a)))
4 Sales_Adv
```

Out[79]:

	Original	Predicted
0	18.0	17.535483
1	5.6	6.449198
2	22.3	23.377230
3	19.6	19.784986
4	16.6	15.625939
5	22.2	20.903885
6	21.5	20.926036
7	13.2	12.749882
8	22.4	23.144407
9	8.8	8.479120
10	21.7	23.271122
11	25.4	25.037964
12	10.1	11.134859
13	20.7	19.874740
14	24.4	22.234459
15	7.0	7.225517
16	13.6	11.944737
17	10.4	11.422671
18	15.6	15.526823

As we can see that the original values & Predicted values are approximately same which concludes that the model is working on 100% accuracy.

Conclusion

In this paper, we built several regression models, we evaluated and compared each model to determine the one with highest performance. We also looked at how some models rank the features according to their importance. In this paper, we followed the data science process starting with getting the data, then cleaning and pre-processing the data, followed by exploring the data and building models, then evaluating the results and communicating them with visualizations.

Above model can be used to predict the sales of advertisement using different methods like newspaper, TV and radio.

Learning Outcomes of the Study in respect of Data Science:

- Obtain, clean/process, and transform data.
- Analyse and interpret data using an ethically responsible approach.
- Use appropriate models of analysis, assess the quality of input, derive insight from results, and investigate potential issues.
- Apply computing theory, languages, and algorithms, as well as mathematical and statistical models, and the principles of optimization to appropriately formulate and use data analyses
- Formulate and use appropriate models of data analysis to solve hidden solutions to business-related challenges