Data Trained Blog

Advertising Sales Channel Prediction

ARGHA KUMAR MAITI

INSTUCTION

In this article, I will go through the whole process of creating a machine learning model on Advertisement Sales Channel Prediction. It provides the predict the total sales generated from all the sales channel.

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.

we'll build a regression model to predict Sales using an appropriate predictor variable.
The below link provided is for downloading the dataset.

Download Files:

https://github.com/argha-test/Projects-Intern-/tree/main/Advertisement%20sales%20channel%20datasets

Imported Libraries:

```
#import libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

Import CSV File For Analysis:

```
df= pd.read_csv("Advertising.csv")
df.head(10)
```

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
5	6	8.7	48.9	75.0	7.2
6	7	57.5	32.8	23.5	11.8
7	8	120.2	19.6	11.6	13.2
8	9	8.6	2.1	1.0	4.8
9	10	199.8	2.6	21.2	10.6

This will give you the overview of top 10 data.
Then I checked for list of all column as we are unable see all column in the above overview:

```
df.columns
```

```
Index(['Unnamed: 0', 'TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

Above is the list of all columns of our dataset. Let's check for the number of rows and columns i.e., shape of dataset.

```
df.shape
```

```
(200, 5)
```

We have 200 rows and 5 columns. Then I checked for datatypes of all the columns.

```
df.isnull().sum()
```

```
Unnamed: 0    0
TV            0
radio         0
newspaper     0
sales         0
dtype: int64
```

Our datasets, all columns has numeric values.
Now I will check for null values and work on it

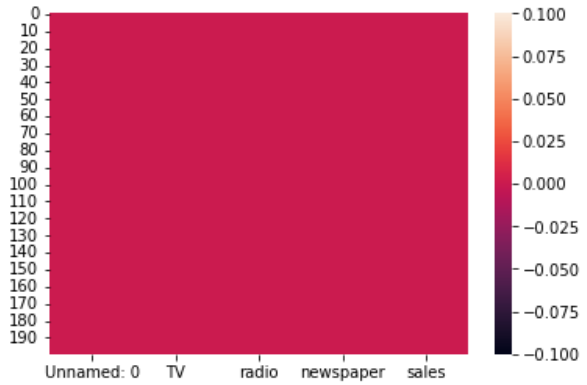
```
df.isnull().sum()
```

```
Unnamed: 0    0
TV            0
radio         0
newspaper     0
sales         0
dtype: int64
```

Datasets does not contain any null values present . Let's see it with visualization.

```
sns.heatmap(df.isnull())
```

```
<AxesSubplot:>
```



There is one column which name is "Unnamed: 0", This column is not required for further analysis. Let's drop this column for more better analysis.

```
df = df.drop(['Unnamed: 0'], axis=1) #dropped column "Unnamed: 0"
```

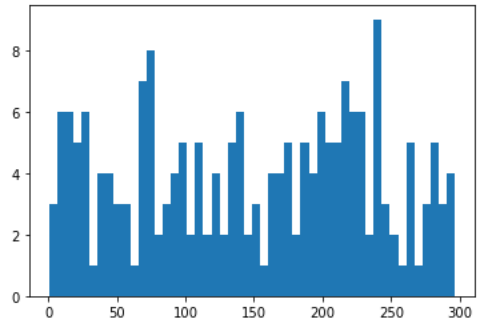
After dropping this column, now we have total 4 columns for further analysis.

Let's do some Visualization:

Now I will see all graphs through visualization technique with various type of plot.

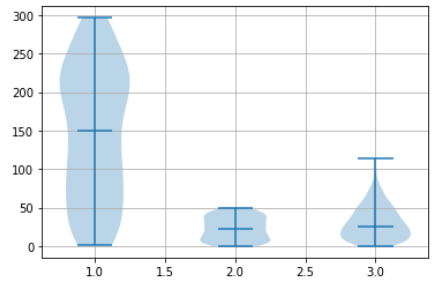
```
plt.hist(df["TV"],bins=50)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



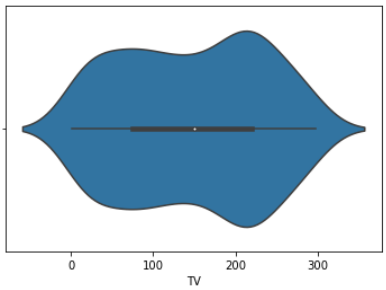

```
: list=(df["TV"],df["radio"],df["newspaper"])
  plt.violinplot(list,showmedians=True)
  plt.grid(True)
  plt.show
```

```
: <function matplotlib.pyplot.show(close=None, block=None)>
```



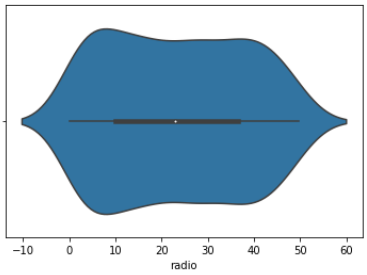
In the above plot data is highly distributed for TV then newspaper then last comes radio.

```
: sns.violinplot(x="TV",data=df)
  plt.show()
```



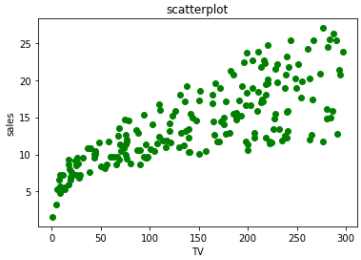
In above figure we can know that maximum data range of TV is 60 to 230(approx.).

```
sns.violinplot(x="radio",data=df)
plt.show()
```

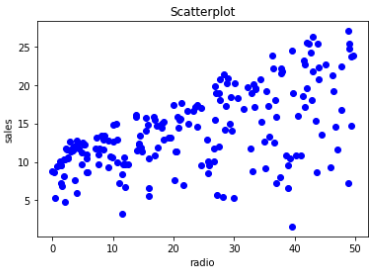


In above violin plot we can know that the maximum data range of radio is 7 to 45(approx.).

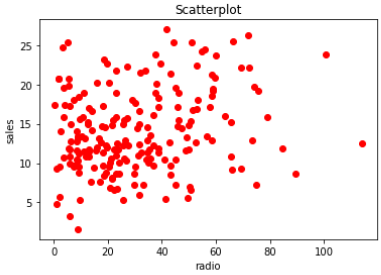
```
x=df["TV"]
y=df["sales"]
plt.figure()
plt.scatter(x,y,color="g")
plt.title("scatterplot")
plt.xlabel("TV")
plt.ylabel("sales")
plt.show()
```




```
x=df['radio']
y=df['sales']
plt.figure()
plt.scatter(x,y,color='b')
plt.title("Scatterplot")
plt.xlabel("radio")
plt.ylabel("sales")
plt.show()
```



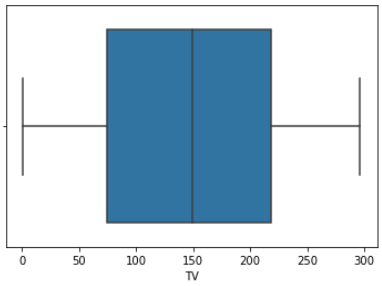

```
x=df['newspaper']
y=df['sales']
plt.figure()
plt.scatter(x,y,color='r')
plt.title("Scatterplot")
plt.xlabel("radio")
plt.ylabel("sales")
plt.show()
```

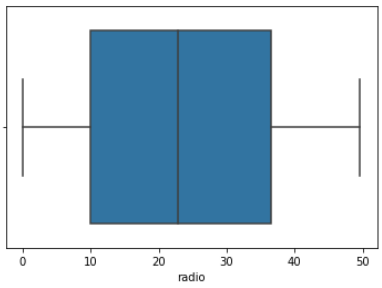


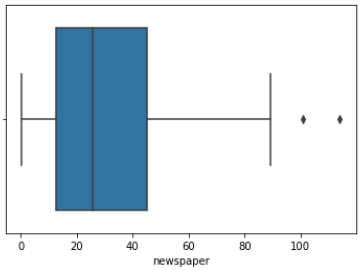
Above figure sales are highly getting increased with increase of advisement in TV, then sales is increase with Radio however with advertisement in newspaper its not always increasing, it varies.

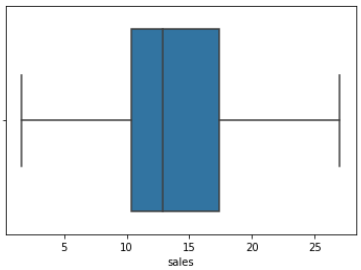
Now let's look at all columns with the help of distribution plot and boxplot.

```
: for i in df.columns:
    plt.figure()
    sns.boxplot(x=df[i])
    plt.show()
```



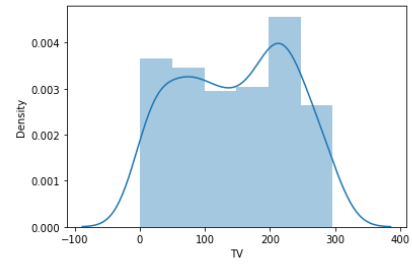


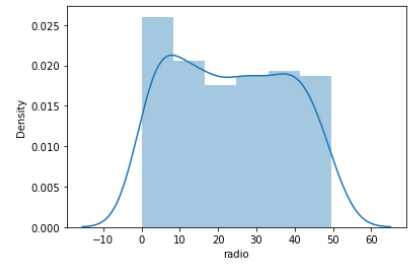


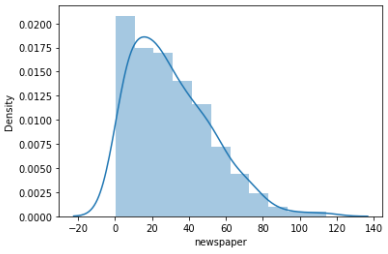


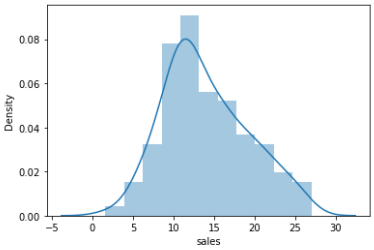
above boxplot shows that only newspaper column has few outliers.

```
for i in df.columns:
    plt.figure()
    sns.distplot(df[i])
    plt.show()
```









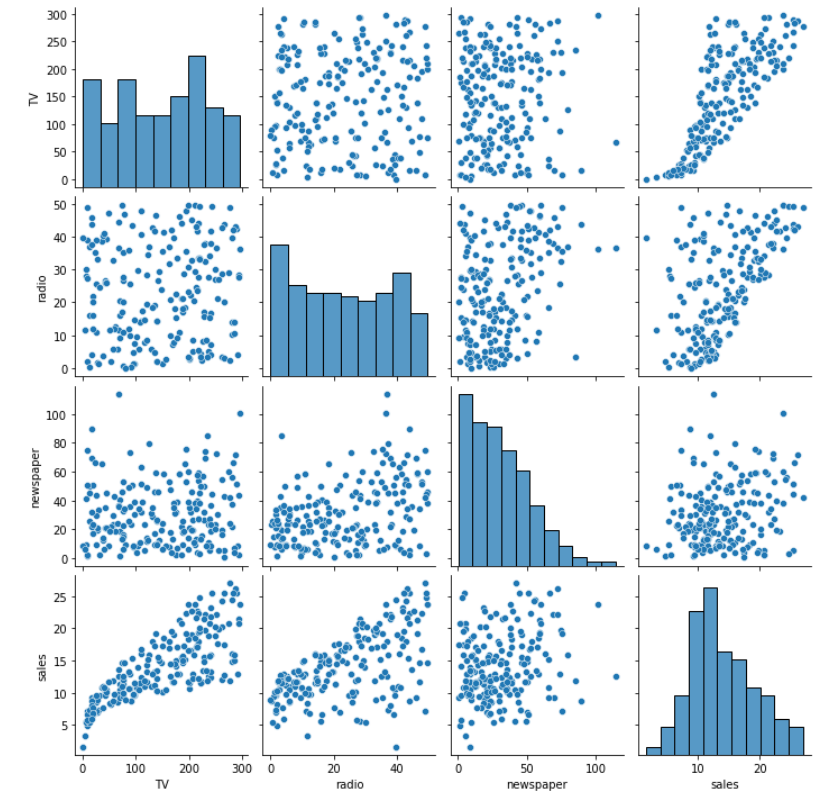
Above distribution plots shows that only newspaper has few skewness. Let's check how much skew data we have.

```
df.skew()
```

```
TV          -0.069853
radio        0.094175
newspaper    0.894720
sales        0.407571
dtype: float64
```

All the values of skewness are in range of -0.5 to +0.5 except of the Newspaper column. So need to remove these skew data for better result.

```
: sns.pairplot(df)
: <seaborn.axisgrid.PairGrid at 0x26f7f52ac10>
```



In above figure we can see that all columns lies between in your nature.

Describe the datasets:

```
: df.describe()
```

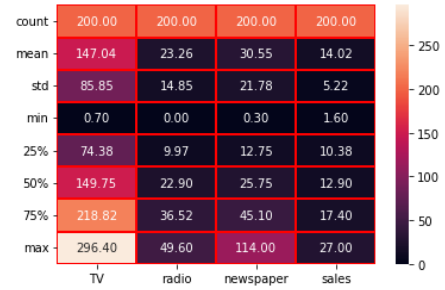
:

	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

- This function gives much information about data set such as count function tells that there are no missing values. Also it provides mean, median, max values from which we can derive many concl usions.
- Mean Values of TV is very higher than rest of all columns.
- Also Standard deviation is very high in TV column.
- There is a large difference between 75% and maximum for newspaper.so outliers are present in this column.

Lets see its with visualization with heatmap.

```
: plt.figure()
  sns.heatmap(df.describe(),annot=True,linewidths=0.1,linecolor="red",fmt="0.2f")
: <AxesSubplot:>
```



Correlation of the columns with target columns:

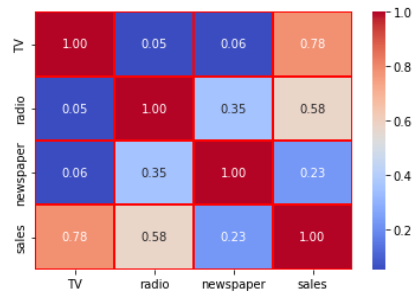
Correlation function gives information about in which manner features are correlated with each other.There may be strong positive corelation or negative corelation betweenvariables.From this we can understand which features arestrongly corelated to target variable.

```
df.corr()
```

	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

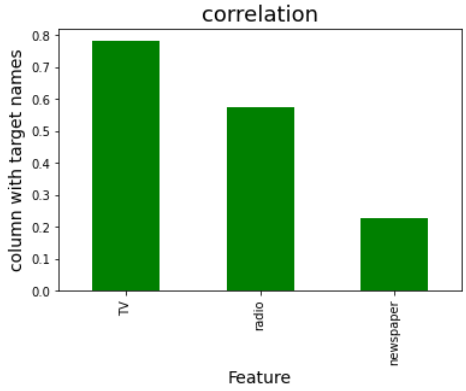
Now see it's with visualization through heatmap.

```
: plt.figure()
  sns.heatmap(df.corr(),annot=True,linewidths=0.1,linecolor="red",fmt="0.2f",cmap="coolwarm")
: <AxesSubplot:>
```



Checking the columns which are positively and negative correlated with the target columns:

```
plt.figure()
df.corr()['sales'].sort_values(ascending=False).drop(['sales']).plot(kind='bar',color='g')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```



Above figure we can see that TV is the strong positive correlation with the target column(“Sales”). And Newspaper is the low positive correlation with the target column(“Sales”).
There are no negative correlation with the target column.

Outliers and Skewness Removal:

Let’s Find out Z-score:
A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms **of standard deviations from the mean**. ... A Z-Score is a statistical measurement of a score's relationship to the mean in a group of scores.

I considered threshold value as 3, now removing outliers.

```
from scipy.stats import zscore
z=np.abs(zscore(df))
threshold=3
np.where(z>3)
```

```
(array([ 16, 101], dtype=int64), array([2, 2], dtype=int64))
```

```
df_new=df[(z<3).all(axis=1)]
df_new
```

	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
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

198 rows × 4 columns

Now we have no outliers in our datasets.
After performing outliers removal our new dataset is 198 rows and 4 columns present.

```
df_new.shape
```

```
(198, 4)
```

```
df.shape
```

```
(200, 4)
```

Our old datasets is 200 rows and 4 columns. Let's check how much data are lose from our given datasets.

```
data_loss=((200-198)/200)*100
data_loss
```

```
1.0
```

We can see that our 1 percentage data are loss after performing outliers removal.

Splitting data between dependent and independent variables i.e. x & y.

```
x=df_new.drop("sales",axis=1)
y=df_new["sales"]
print(x.shape)
print(y.shape)
```

```
(198, 3)
(198,)
```

Our dependent variable has 198 rows and 3 columns and target variable has 198 rows present.
Then I have to perform power-transform to removing skewdata. We only remove skewness on dependent variable.

```
from sklearn.preprocessing import power_transform
df_new=power_transform(x)

df_new=pd.DataFrame(df_new,columns=x.columns)
```

```
df_new.skew()
```

```
TV          -0.320682
radio       -0.236668
newspaper   -0.101288
dtype: float64
```

now skewness has been remove using power transform.

Then I have performed standardization. I have a minmax scaler as for maximum column data is within a particular range. I have not standardized our target variable.

Let's Perform Scaling:

Min-Max Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

```
from sklearn.preprocessing import MinMaxScaler
m=MinMaxScaler()
m.fit_transform(df_new)
```

```
array([[0.83184229, 0.83610008, 0.87878833],
       [0.23409837, 0.85792891, 0.70367754],
       [0.10772489, 0.95040548, 0.87943518],
       [0.60558624, 0.88654282, 0.80621464],
       [0.69283238, 0.3537725 , 0.80550206],
       [0.05922732, 0.99072969, 0.91545939],
       [0.28655711, 0.76080939, 0.49234092],
       [0.50738231, 0.53699982, 0.32300945],
       [0.05860268, 0.10061138, 0.04012979],
       [0.7474427 , 0.11984664, 0.46415525],
       [0.31957959, 0.22426791, 0.50061222],
       [0.78933131, 0.61659793, 0.1539589 ],
       [0.14124496, 0.79595184, 0.85713501],
       [0.43198413, 0.27420902, 0.23626232],
       [0.75961185, 0.76235653, 0.71106801],
       [0.73492021, 0.97471774, 0.76518395],
       [0.9684895 , 0.86225605, 0.78672515],
       [0.33120799, 0.55378751, 0.42608502],
       [0.59273723, 0.61485341, 0.43690074],
```

```
x=df_new
```

As we can see after standardization data is returned in the form of an array hence, I have transformed it into a data frame so that I can build a model. Before building a model, I have checked the best random state on which are getting best accuracy.

Model Training:

Let's First import Libraries for training model.

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
```

```
maxAcc=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    Ln=LinearRegression()
    Ln.fit(x_train,y_train)
    pred=Ln.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print("Best accuracy is ",maxAcc, " at Random State ",maxRS)
```

```
Best accuracy is  0.9473362332441402  at Random State  90
```

We found the best random state at 90. Let's check which regression method gives us the best output. Now lets divide data for training and testing.

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=maxRS)
```

Now data is ready to fit into different models. As this is a regression problem we have to use regression algorithms.
As shown above we have imported regression algorithm libraries, now we have to create instance of that. Now let's fit data into models and derive r2_scores.

I have use total 4 regression model.

- LinearRegression
- DecisionTreeRegressor
- RandomForestRegressor
- AdaBoostRegressor.

Output of above code will give me
r2_score, mean_absolute_error, mean_squared_error, Root_mean_squared_error of all models.

```
: model=[LinearRegression(),DecisionTreeRegressor(),RandomForestRegressor(),AdaBoostRegressor()]
for m in model:
    m.fit(x_train,y_train)
    #sc=m.score(x_train,y_train)
    predm=m.predict(x_test)
    acc=r2_score(y_test,predm)
    print('Accuracy Score of',m,'is:',acc)
    print('mean_absolute_error:',mean_absolute_error(y_test,predm))
    print('mean_squared_error:',mean_squared_error(y_test,predm))
    print('Root mean_squared_error:',np.sqrt(mean_squared_error(y_test,predm)))
    print("\n")
```

```
Accuracy Score of LinearRegression() is: 0.9473362332441402
mean_absolute_error: 1.027664676530612
mean_squared_error: 1.5876335108822055
Root mean_squared_error: 1.2600132978989569


Accuracy Score of DecisionTreeRegressor() is: 0.977990884544194
mean_absolute_error: 0.6299999999999998
mean_squared_error: 0.6634999999999995
Root mean_squared_error: 0.8145550932871266


Accuracy Score of RandomForestRegressor() is: 0.9839237484492447
mean_absolute_error: 0.6084250000000004
mean_squared_error: 0.4846443250000004
Root mean_squared_error: 0.696164007256911


Accuracy Score of AdaBoostRegressor() is: 0.9613752731196762
mean_absolute_error: 0.8563967685510487
mean_squared_error: 1.16440419137037
Root mean_squared_error: 1.0790756189305595
```

We can see the highest accuracy score is for **RandomForestRegressor** however before proceeding ahead let's check the cross-validation score.

K-Fold Cross Validation:

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

Cross Validation Model:

```
]: model=[LinearRegression(),DecisionTreeRegressor(),RandomForestRegressor(),AdaBoostRegressor()]
for m in model:
    score=cross_val_score(m,x,y,cv=5)
    print("Score for",m,"is: ",score.mean())
```

```
Score for LinearRegression() is:  0.8958270968555071
Score for DecisionTreeRegressor() is:  0.9512468154617058
Score for RandomForestRegressor() is:  0.9759871754320366
Score for AdaBoostRegressor() is:  0.9578544270737883
```

We can see the highest accuracy score is of RandomForestRegressor however With the **AdaBoostRegressor** model there is very little difference in accuracy and cross validation score, Hence the best model is **AdaBoostRegressor.**

Let's try to improve performance of the model by doing hyper parameter tuning with GridSearchCV. Let's import required libraries and create a dictionary for parameters.

Hyper Parameter Training:

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters.

We have to be tuned so that the model can optimally solve the machine learning problem. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss.

So now let's do hyper parameter tuning for AdaBoostRegressor..

```
: from sklearn.model_selection import GridSearchCV
```

```
: parameters={'n_estimators': [50, 100],
 'learning_rate' : [0.01,0.05,0.1,0.3,1],
 'loss' : ['linear', 'square', 'exponential'],
 'random_state' : [0]}
```

I have called the model and tried fitting out the train set i.e., x=train (Input variables), y_train (target variable). Let's see the best fitted parameters found then predict output for input variables(x_test) and then check for accuracy percentage.
And I have to used best 4 parameter in this model.

```
1]: GCV=GridSearchCV(AdaBoostRegressor(),parameters,cv=5)

2]: GCV.fit(x_train,y_train)

2]: GridSearchCV(cv=5, estimator=AdaBoostRegressor(),
                 param_grid={'learning_rate': [0.01, 0.05, 0.1, 0.3, 1],
                             'loss': ['linear', 'square', 'exponential'],
                             'n_estimators': [50, 100], 'random_state': [0]})

3]: GCV.best_params_

3]: {'learning_rate': 1,
     'loss': 'exponential',
     'n_estimators': 100,
     'random_state': 0}

4]: Final_mod= AdaBoostRegressor(learning_rate=1,loss="exponential",n_estimators=100,random_state=0)
    Final_mod.fit(x_train,y_train)
    pred=Final_mod.predict(x_test)
    r2=r2_score(y_test,pred)
    print(r2*100)

    96.62740999104112
```

Now the improved accuracy score is **96.62%** after performing GridSearchCV, then I have saved the model called and predict the target variable.

saving the best model:

Now I will import pickle library and save the best model.

```
]: import pickle
   filename= 'sales_prediction.pkl'
   pickle.dump(Final_mod, open(filename, 'wb'))
```

Now the best model is successfully save . with the file name is "**sales_prediction.pkl**".

Conclusion:

Again I have load the best model and predict the target variable.

```
: #load the model from the disk

  loaded_model = pickle.load(open('sales_prediction.pkl', 'rb'))
  result = loaded_model.score(x_test,y_test)
  print(result)

  0.9662740999104111

: conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],pred[:]],index=["Predicted","Orginal"])
  conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	30	31	32	33	
Predicted	17.24	6.666667	22.825	18.637037	16.083333	20.112963	21.706667	13.466667	22.825	8.856	...	13.566667	10.948148	16.276923	6.083333	10.7115
Orginal	17.24	6.666667	22.825	18.637037	16.083333	20.112963	21.706667	13.466667	22.825	8.856	...	13.566667	10.948148	16.276923	6.083333	10.7115

2 rows × 40 columns

Throughout this, we saw Data is important in Human Resource department. We used AdaBoostRegressor and learned how it can be very advantageous over other available machine learning algorithm. This is by no means a comprehensive analysis of the marketing data set but simply an example of how to perform and interpret a multiple regression.

THANK YOU