9/4/22
9:30 pm

★ "Simple Linear Regression" ★

Cross-Industry Standard Process For Data mining.

→ Data Science project life cycle (crisp-dm) → Methodolgy

1. Bussiness problem Understanding
2. Data Understanding → Data collection [D...] → Data variables (research - domain Expert) → Dataset understanding
3. Data Preprocessing → EDA → Data cleaning → Data Wrangling → Train / test split
4. Modelling → Apply Different Algorithms
5. Evaluation → Accuracy of The Model checking (or) If not go back and start again
6. Presentation → Visulazation (Tableau, Power BI)

→ Basic Liberaries For all Machine Learning

```
# Import numpy as np
# Import Pandas as pd
# Import matplotlib.Pyplot as plt
%matplotlib inline
# Import Seaborn as sns
```

Project 1

Step 1 : Business Problem Understanding

- Is there a relationship between $\boxed{\text{total advertising}}$ Spend and Sales ?
- Our next ad Campaign will have a total spend of $200, how many units do we expect to $\boxed{\text{sell}}$ as a result of this ?

Step 2.1 Data collection

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

, comma Seperated values

Out

TV	Radio	newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	59.5	18.5
180.8	10.8	58.4	12.9

Step 2.2 Data Understanding

The sample data displays sales (in thousands of units) for a particular product as a function of advertising budgets for a particular product as a function of advertising budgets (in thousand of dollars) for Tv, radio and Newspaper media

Independent Variables

→ Tv : Advertising dollars spent on TV for a $\boxed{\text{Single}}$ product in a Given market (in Thousands of dollars) → Ex: 230.1×1000 already done scaling

→ Radio :- Advertising dollars Spent on Radio

→ Newspaper : Advertising dollars Spent on news paper

Target Variable

→ Sales : Sales of a $\boxed{\text{Single product}}$ in a given market (in thousands of widgets) → Ex:- 22.1 ×1000 = 22100 products sold

Step - 2.3 Dataset Understanding

df.info()

Out

	column	Non Null Count	Dtype
0	TV	200 non null	float 64
1	Radio	200 non null	float 64
2	Newspaper	200 non null	float 64
3	Sales	200 non null	float 64

(200, 4)

Que :- If Some One was to spend a total of $200, what would The expected sales be ?

A:- We have simplified This quite a bit by Combining all Features into "total spend"

We add new column, total of 3 column creating new one

df ["total_spend"] = df ["Tv"] + df ["radio"] + df ["newspaper"]

df head ()

Out

	TV	Radio	newspaper	Sales	total_spend
0	230.1	37.8	69.2	22.1	337.1
1	44.5	39.3	45.1	10.4	128.9
2	17.2	45.9	69.3	9.3	132.4
3	151.5	41.3	58.5	18.5	251.3
4	180.8	10.8	58.4	12.9	250.0

We drop 3 columns, Consider only two columns For Calculating

df. drop (columns = [" Tv", "radio", "newspaper"], inplace = True)

df. head ()

Out

Sales	Total_spend
22.1	337.1
10.4	128.9
9.3	132.4
18.5	251.3
12.9	250.3

Step-3.1 Exploratory Data Analysis [EDA]

On The basis of This data. How should you spend advertising money in future? These general questions might lead you to more specific questions :

1. Is there a relation ship between ads and Sales ?
2. How strong is that correlation ?
3. Given ad spending, can Sales be predicted ?

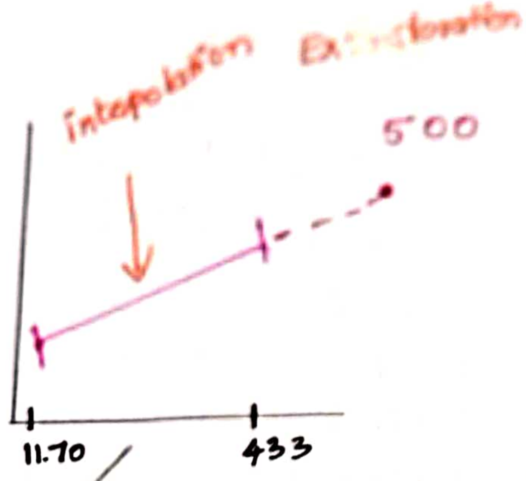
df. describe()

Out :-

	Sales	total_spend
count	200.000000	200.000000
Mean	$\boxed{14.022500}$	200.860500
std	5.217457	92.985181
min	1.600000	$\boxed{11.700000}$
25%	10.375000	123.550000
50%	$\boxed{12.900000}$	207.350000
75%	17.400000	281.125000
max	27.000000	$\boxed{433.600000}$

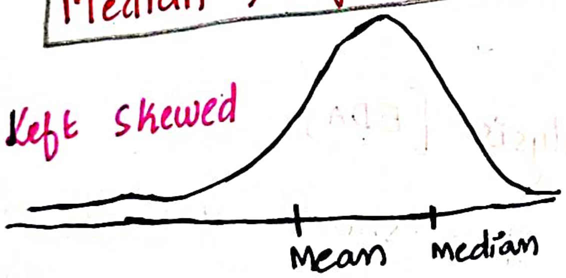
How close The values in dataset To The mean

Mean and median are close. They normal distribution

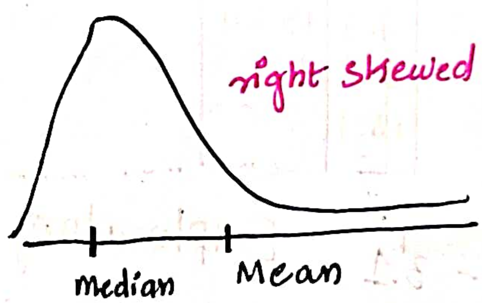


* **Interpolation** :- Estimation of Value within two Known Values in Sequence of Values.
* **Extrapolation** :- Prediction The Value, Outside of Known Values it is called Extrapolation.

*** ⇒ Median > (greater Than) Mean



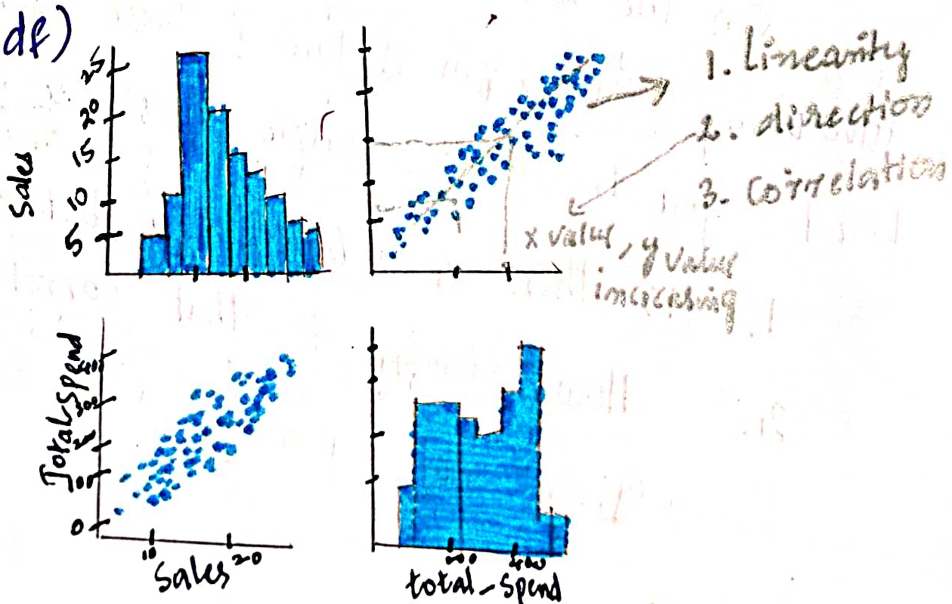
⇒ Median < [Less Than] Mean



Sns. pairplot(df)

plt. show()

Out :



df.corr()

Out

	Sales	total_spend
Sales	1.000000	0.867712
total_spend	0.867712	1.000000

Step 3.2 Data cleaning

df.isnull().sum()

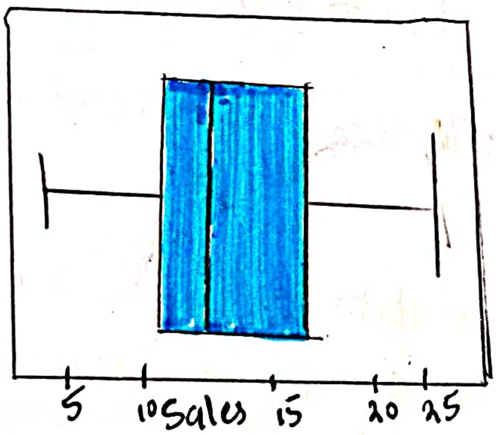
Out

sales 0
total_spend 0

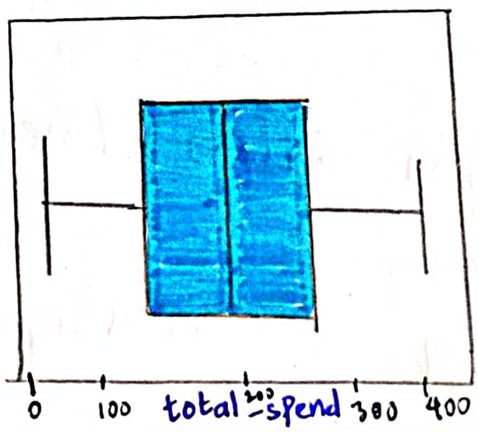
Step 3.3 Data Wrangling

* Outliers

sns.boxplot (df.sales)
plt.show ( )



sns.boxplot (df.total_spend)
plt.show()



* Feature transformation

df. sales. skew()

Out :- 0.407

df. total_spend. skew()

Out : 0.049

STep 3. Train Test split

x = df . drop (columns = "sale")

Y = df ["sales"]

from sklearn . model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split (x, y, test_size = 0.3, random_state=)

STep-4 Modelling :- If doing with data preprocessing. directly working on modelling is "Base line model (or) raw model.

from sklearn. linear_model import LinearRegression

storing

model = LinearRegression()

Fitting the data

```
# model.fit(x_train, y_train)
```

$\boxed{\text{Out}}$:- Linear Regression()

Attributes (help)

$\beta_0 + \beta x$ → y intercept, coefficient value

```
# Print("Model Intercept :", model.intercept_)
  Print("Model Coefficient :", model.coef_)
```

$\boxed{\text{Out}}$:- Model Intercept : 4.33176
Model Coefficient : 0.0480

$\hat{y} = 4.33176 + 0.0480(x)$

⇒ Predictions

```
# spend = 200
# predicted_sales = 4.33176 + 0.0480 * spend
# predicted_sales
```

$\boxed{\text{Out}}$:- 13.93860

If we have multiple Equations, we can write this

```
# model.predict([[200]])
```

→ input should given in 2dimensional

$\boxed{\text{Out}}$ array([13.93860])

Predicting on x train, x test

	x	y	$\hat{y}$	
x Train	1	10	14	→ Train accuracy
	2	15	12	
	3	20	13	
	4	30, 40	14, 33	
x Test	5	50	42	→ Test accuracy

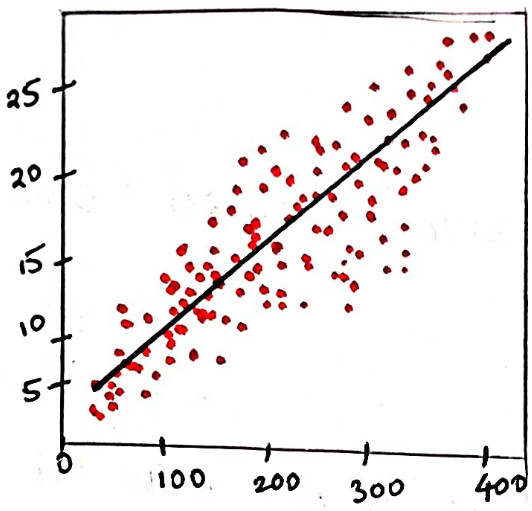
```
# train_predictions = model.predict(x_train)
# test_predictions = model.predict(x_test)
```

Only stores, No output

Regression line

⇒ Plotting The Least squares line

```
# plt.scatter (x_train, y_train, color="red")
# train_predictions = model.predict (x_train)
# plt.plot (x_train, train_predictions, color="Black")
# plt.show()
```



Step 5

⇒ Evaluation metrics

```
from sklearn.metrics import mean_absolute_error

# print ("MAE For Test data :", mean_absolute_error (y_test, test_predictions))

# print ("MAE For Train data :", mean_absolute_error (y_train, train_predictions))
```

Out: MAE For Train data : 1.97768
MAE For Test data : 1.90653

from sklearn.metrics import mean_squared_error

```
# print ("MSE for test data : "mean_squared_error (y_test,
                                    test_predictions))

# print ("MSE for train data": mean_squared_error(y_train
                                    train_predictions))
```

Out : MSE for Test data : 6.82220
MSE for Train data : 6.40720

RMSE (Just write np.sqrt).

```
# print ("RMSE for test data", np.sqrt(mean_squared_error
                                    (y_test, test_predictions)))

# print ("RMSE for train data", np.sqrt (mean_squared_error
                                    (y_train , train_predictions)))
```

Out : RMSE for Test data : 2.611935
RMSE for train data : 2.53124

from sklearn.metrics import r2_score

```
# print ("R2 for test data":, r2_score (y_test, test_
                                    Predictions))
# print ("R2 for train data":, r2_score ( y_train ,train_
                                    Predictions))
```

Out R2 for test data : 0.74001
R2 for train data : 0.76534

(or)

Another way for R^2

model. score (x_train, y_train)

$\boxed{Out}$: 0.76534

model. Score (x_test, y_test)

$\boxed{Out}$: 0.740017

cross_validation ⇒ K-Fold Cross Validation

from sklearn. model_selection import cross_val_score

scores = cross_val_score (model, x, y, cv=5)

Here K=5

print (scores)

scores. mean()

$\boxed{Out}$: [0.74964192, 0.79455226, 0.76417134, 0.74872042, 0.65980565]

⇒ 0.74337 → should be Equal to Test square Than it is good model.

Linear Regression assumptions

Errors

L → Linearity

I → Independent

N → Normality

E → Equal variance

10/4/22

3:30 AM

CHECK LIST

Assumptions of linear regression

1. check whether model has overfitting (or) underfitting problem
2. Is test Accuracy = Cross Validation Score
3. check assumptions (if it is linear Regression)
4. check model meets bussiness problem requirements
5. Finally, save the model and share to deployment Team.

Purely Assumption of The Researcher

1 Que :- Is model has overfitting (or) underfitting problem ?

Ans :- it's good model.

2 Que Is test accuracy = Cross Validation Score ?

Ans :- Applied K-Fold cross validation and it is Equal in test accuracy and CV score.

3 Que :- check Assumptions (if it is linear regression)

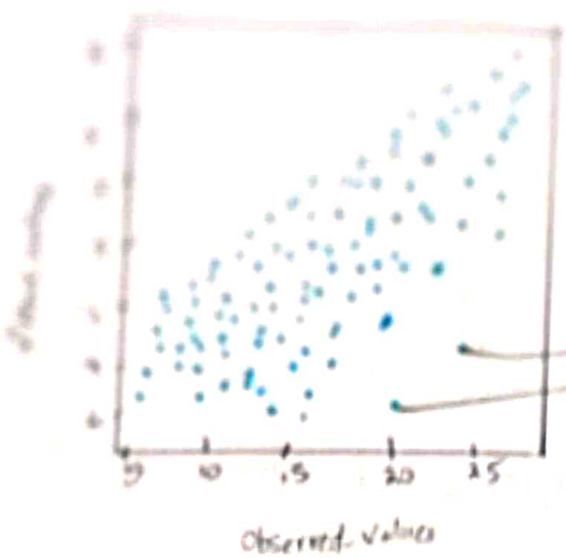
Ans :- Line Assumptions.

1. Linearity of Errors

```
# test_res = y_test - test_predictions.
```

Saved in This

```
# plt.scatter (y_test, test_res)
plt.xlabel ("observed_values")
plt.ylabel ("fitted_values")
plt.show()
```



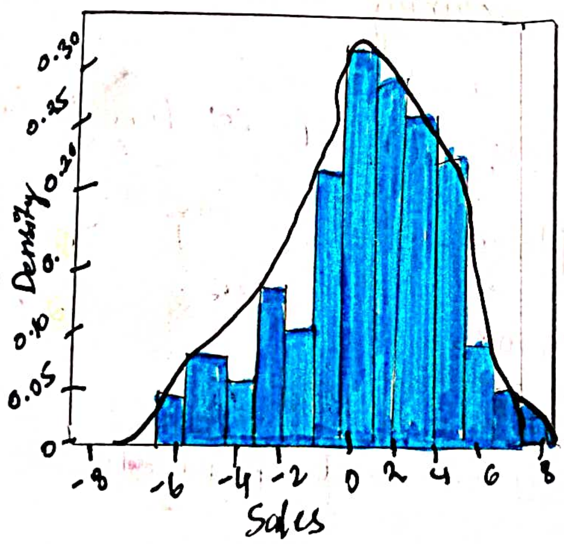
we can ignore The Edge points upto ± 5% error

2. Normality of Errors

sns. distplot (test_res, bins = 15, kde=True)

plt. show()

Out



Normal skewed

"Assumption meet"

% of positive Errors

% of negative Errors

3. Equal varience of Error (Homoscadesicity)

plt. scatter (test_predictions, test_res, c = "r")

color = red

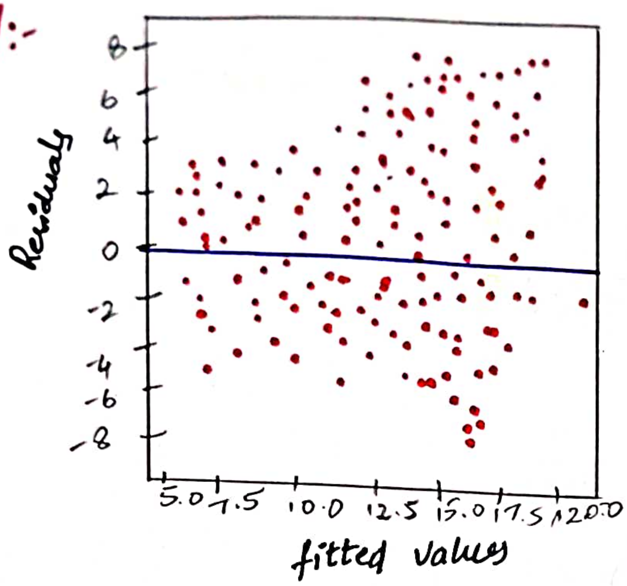
plt. axhline (y=0, color = "blue")

ax = axis

h = horizontal line

```
# plt.xlabel ("fitted values")
# plt.ylabel ("Residuals")
# plt.show()
```

Out :-



For model

Ho :- Avg line best fit line
H1 : regression line best line

Anova Test
P: ≤ α
P ≥ α

1 Sample Test

two Tail test

Variable

Ho :- $\beta_1 = 0$

H :- $\beta_1 \neq 0$

Significant

Consider

variable insignificant

don't

4. Variable significance

```
# import Statsmodels.formula.api as smf
# m = smf.ols ("y~x", data=df).fit()
# m.Summary()
```

"y" upon "x"

for multiple column y~x1+x2+x3+x4

ordinary least square (SSE) min

Out OLS Regression Results.

Dep. variable : Y

Model : OlS

method : Least squares

R-squared Train : 0.753

Adj. R-Squared : 0.752

F.static : 603.4

Prob.(F.static) : 5.06×10^{-62}

p = 5.06e-63
p < 0.05
Reject Ho

Confidence Interval

Std. Error = $\frac{\sigma}{\sqrt{n}}$

	coeffs	std.Err	t	P>|t|	[0.025	0.975]
intercept	4.2430	0.439	9.676	0.000	3.378	5.108
x	0.0487	0.002	24.564	0.000	0.045	0.053

One sample Test applied

$\bar{x} \pm Z_{1-\alpha} \frac{\sigma}{\sqrt{n}}$

Central limit Theorem

1 Sample T Test

Two Tail

95%

2.5

2.5

Error

Que: When Average Test is going to fail in Regression?

Ans: When Sum of Square (SSE) of Average line is Less than the Sum of Squares of regression line $(SSE)_{reg}$ it going To fail in this situation.

$\Rightarrow$ $\boxed{\overset{\text{Errors}}{(SST)_{avg}} < (SSE)_{reg}}$

Step 6 Final Inferences

```
# model.predict([[200]])
```

Out: array([13.9898])

⇒ Save a Model

```
from joblib import dump
# dump(model, "sales_model.joblib")
```

Out: ["sales_model.joblib"] → saves in working directory.

⇒ Load a Model → From client side

```
# from joblib import load
```

loaded_model = load ("sales_model.joblib")

loaded_model.predict ([[200]])

$\boxed{\text{Out}}$ array ([13.989])

loaded_model.predict ([[500]]) # check with multiple values.

$\boxed{\text{Out}}$: array ([28.3488])

with 74 % Accuracy.

19/4/22
5:00pm