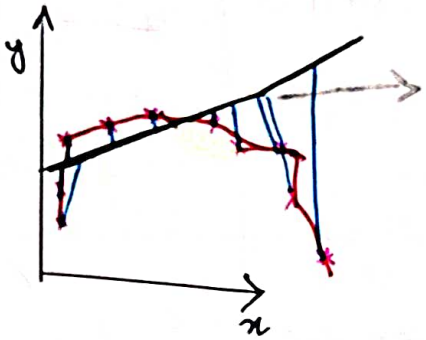
Dt: 16/04/23
1:00pm

Polynomial Regression

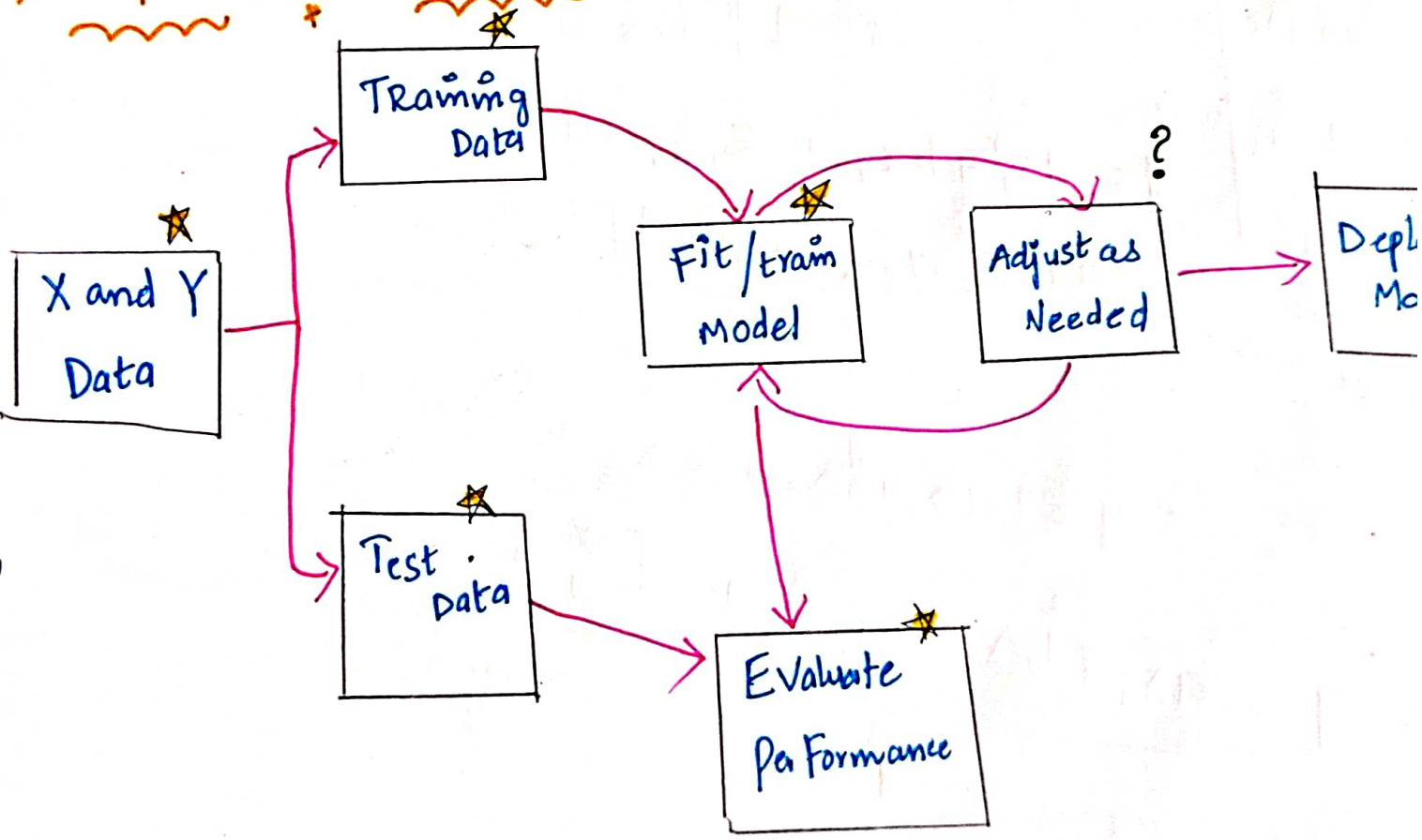
⇒ Any Regression problem. $[SSE]_{min}$ ⇒ $[\varepsilon[y-\hat{y}]^2]_{min}$



straight line (Linear line)

$[\varepsilon(y-\hat{y})^2]$

⇒ Supervised Machine learning process



S.L.R → 1 Variable

$y = ax + b$

Linear

M.L.R → 2 Variables

$y = ax + b + cx + d$

$y = ax_1 + bx_2 + c$

it is came from joining two equations

$= ax_1 + b + cx_2 + d$ (b, d: intercepts)

$y = ax_1 + cx_2 + e$ (always intercept have one value only)

3 Variable

$y = ax_1 + bx_2 + cx_3 + d$

it is came from joining three equations

$y = ax_1 + b$

$y = cx_2 + d$

$y = ex_3 + f$

Quadratic

1 Variable

$y = ax^2 + bx + c$

Here we write intercept value as "Z"

$y = ax_1^2 + bx_1 + dx_2^2 + ex_2 + z$

it is came from joining two equations

$y = ax_1^2 + bx_1 + z_1 c_1 + dx_2^2 + ex_2 + z_2 f_2$

* $y = ax_1 + bx_2 + cx_3 + z$

* $y = ax_1 + bx_2 + cx_3 + dx_1^2 + ex_2^2 + fx_3^3 + z$

(squaring of)

$+ gx_1x_2 + hx_2x_3 + ix_3x_1$

Combinations of variable [iteration terms]

original data | Squaring original Data | Combination (or) Iteration terms

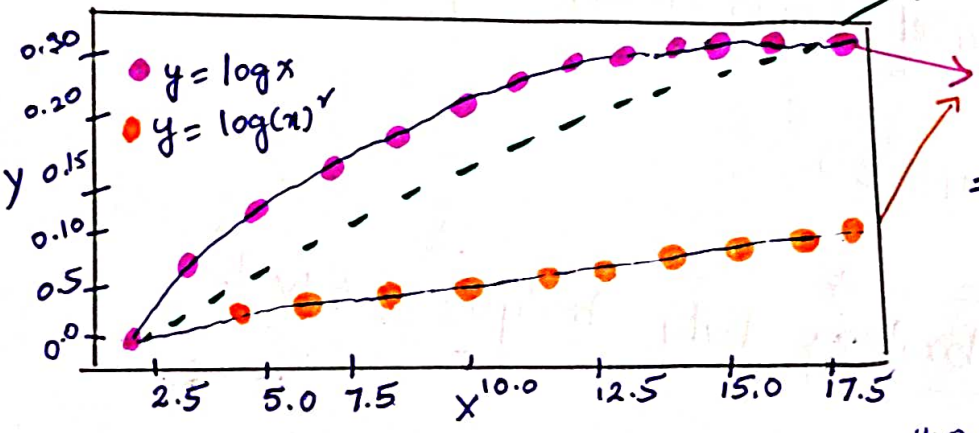
TV	Radio	newspaper	Sales	TV^2	$Radio^2$	$Newspaper^2$	TV*Radio	Radio*Newspaper	Newspaper*TV
—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—

* Quadratic equation : $x_1' * x_2' = x_1' + x_2'$?

multiply → degree 2 ; if we add two variables → degree 1

$a^m \times a^n = a^{m+n}$

⇒ Polynomial Regression



#quadratic – ax^2+bx+c

⇒ If we want to calculate Combinations with Squaring.

* Degree 2 = TV^2 (01) $x_1^2+x_2^2+x_3^2$ + $x_1x_2+x_2x_3+x_3x_1$ + $x_1+x_2+x_3$ (interaction terms)

* Degree 3 = $x_1^3+x_2^3+x_3^3$ + $x_1x_2^2+x_2x_3^2+x_3x_1^2$ + $x_1^2x_2+x_2^2x_3+x_3^2x_2$ + $x_1^2+x_2^2+x_3^2$ + $x_1+x_2+x_3$

cubic Equation : ax^3+bx^2+cx+d

Step 1 — Problem Understanding

Step 2 — Data Collection

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

Step 3 — Data Understanding

```
# df.shape
```

Step 4 — Dataset Understanding

```
# df.info()
```

Step 5 — Exploratory Data Analysis [EDA]

```
# df.describe()
# sns.pairplot(df)
```

Step 6 — Data cleaning

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

Step 7 — Data wrangling

```
# X = df.drop("sales", axis=1)
# y = df["sales"]
```

Adding new Features to original Data

Polynomial Regression with Sci-kit Learn

```
from sklearn.preprocessing import PolynomialFeatures
# polynomial_converter = PolynomialFeatures(degree=2, include_bias=False)
```

include_bias = False → don't include "z" value

$ax_1^2 + bx_2^2 + cx_3^2$
$dx_1x_2 + ex_2x_3 + fx_3x_1$
$gx_1 + hx_2 + ix_3$

```
# X_poly = polynomial_converter.fit_transform(X)
```

fit_transform: Calculate, Convert

```
# X_poly.shape
```

Out (200, 9) — changed 3 to 9

rows = 200, columns

STEP

```
from sklearn.model_selection import train_test_split
# X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.3, Random_state=29)
```

200 → 140 × 9 = train, 60 × 9 = test

MODEL Fitting On Polynomial Data.

```
from sklearn.linear_model import LinearRegression

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

Predictions

```
# train_pred = model.Predict(x_train)
# test_pred = model.predict(X_test)
```

Evaluation

```
# model.score(x_train, y_train)   [# Train R^2]
```

Out 0.986

```
# model.score(x_test, y_test)   [# Test R^2]
```

Out 0.980

Cross - Validation

```
from sklearn.model_selection import cross_val_score

# scores = cross_val_score (model, X_poly, y, cv=5)

# print (scores)

# scores.mean()
```

Out [0.987 , 0.989, 0.991, 0.958, 0.993]

mean : 0.984

RMSE

```
from sklearn.metrics import mean_squared_error

# test_RMSE = np.sqrt (mean_squared_error (y_test, test_pred))

# train_RMSE = np.sqrt (mean_squared_error (y_train, train_pred))

# print (train_RMSE, test RMSE)
```

Out 0.5950 , 0.7233

Earlier,

Multiple Linear Regression
* RMSE - 1.94

Polynomial Regression
* RMSE - 0.72

Applying loop for knowing better Accuracy From First to last.

- from sklearn.preprocessing import PolynomialFeatures
- from sklearn.model_selection import train_test_split
- from sklearn.linear_model import LinearRegression.

```
# train_rmse_errors = [ ]
# test_rmse_errors = [ ]

for d in range [1,10):
# polynomial_converter = PolynomialFeatures (degree = d, include_bias = False)
# X_poly = polynomial_converter.fit_transform(x)
# X_train, X_test, y_train, y_test = train_test_split (X_poly, y, test_size = 0.3, Random_state = 29)
# model = LinearRegression()
# model.fit (x_train, y_train)
# train_pred = model.predict (x_train)
# test_pred = model.predict (X_test)
# train_RMSE = np.sqrt (mean_squared_error (y_train, train_pred)
# train_rmse_errors.append (train_RMSE)
# test_RMSE = np.sqrt (mean_squared_error (y_test, test_Pred)
# test_rmse_errors.append (test_RMSE)
```

train_rmse_errors

Out : 0 [1.7345 d=1 ← degree
1 0.5879 d=2
2 0.4339 d=3
3 0.3517 d=4
4 0.2509 d=5
5 0.19704 d=6
6 5.4214 d=7
7 0.14180 d=8
8 0.16654 d=9

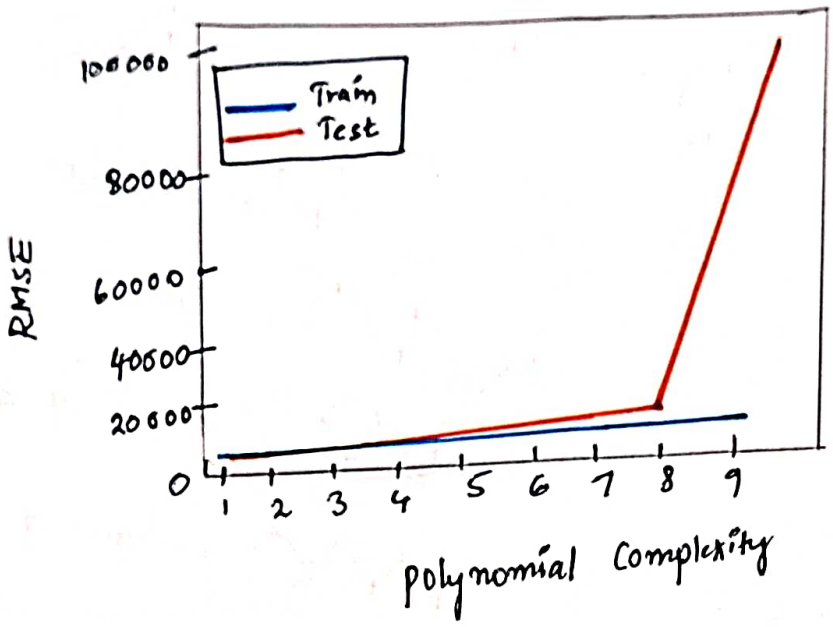
No. of bends
↓
∪ = Quadratic
∩∪ = Cubic
∪∩∪ = degree of 4
∪∩∪∩∪∩∪ = degree of 7

test_rmse_errors

Out : 0 1.5161 d=1 ↓
1 0.6646 d=2
2 0.5803 d=3
3 0.5077 d=4
4 2.5758 d=5 ↑
5 4.4926 d=6
6 1381.404 d=7
7 4449.599 d=8
8 9591.24 d=9

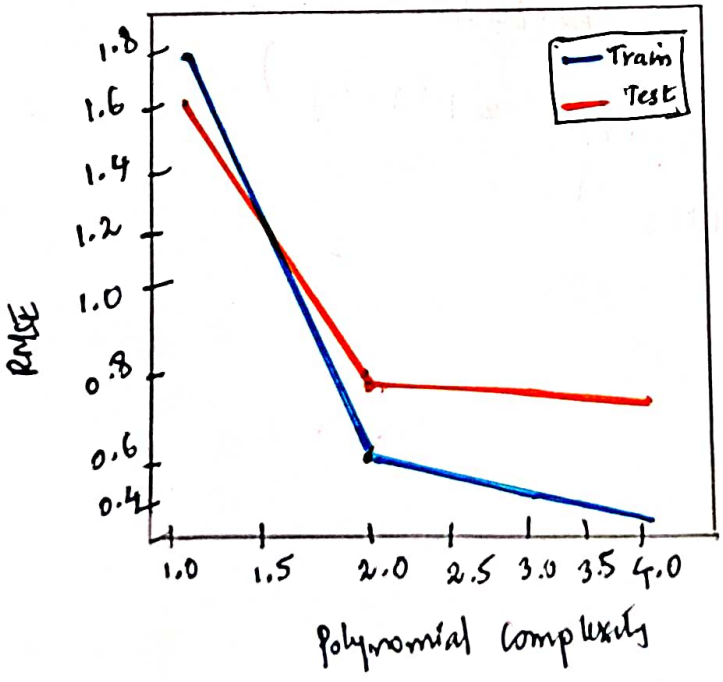
plt.plot (range (1,10), train_rmse_errors, Label = "TRAIN")
plt.plot (range (1,10), test_rmse_errors, Label = "TEST")
plt.xlabel ("Polynomial complexity")
plt.ylabel ("RMSE")
plt.legend()
plt.show()

Out :




```
# plt.plot(range(1,5), train_rmse_errors[:4], label = "TRAIN")
# plt.plot(range(1,5), test_rmse_errors[:4], label = "TEST")
# plt.xlabel("Polynomial Complexity")
# plt.ylabel("RMSE")
# plt.legend()
# plt.show()
```

Out :



S.No.5 Finalizing Model choice

```
# final_poly_converter = polynomial Features (degree = 2,
                                              include_bias = False)

# final_model = Linear Regression()
# final_model.fit (final_poly_converter.fit_transform(x), y)
```

Out: Linear Regression()

* Saving MODEL and CONVERTER *

```
from joblib import dump
# dump (final_model, "Sales_poly_model.joblib")
```

Out ["Sales_poly_model.joblib"]

```
# dump (final_poly_converter, "poly_converter.joblib")
```

Out: ["poly_converter.joblib"]

Deployment & predictions :

Client wants to spend 149k on "TV", 22k on "radio" 12k on "Newspaper" Ads. How many units could we expect to sell as a result ?

```
from joblib import load
```

```
# Loaded_poly = load ("poly_converter.joblib")
# loaded_model = load ("Sales_poly_model.joblib")

-
# Campaign_poly = loaded_poly.transform ([[149, 22, 12]])

-
# final_model.predict (campaign_poly)
```

Out: array ([14.51114])

16/04/22
9:25 pm