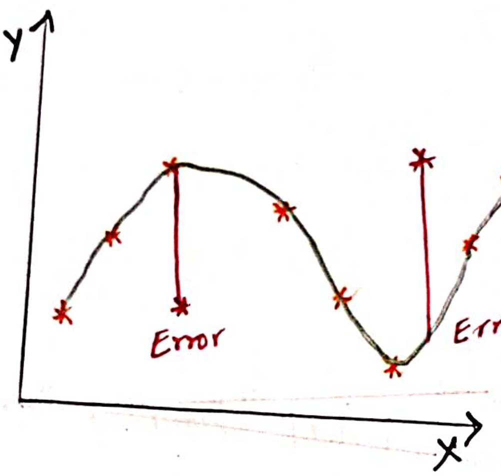
Dt:- 16/04/22
9:28 pm

* Regularization

Q: How To Reduce The overfitting problem in Regression?

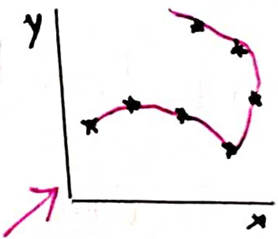
A :- Ex :- Polynominal Regression ⟶ Train Accuracy > Test Accuracy



* – Train Data
* – Test Data

Train_error = Low
Test-error = High

* Simple Linear Regression [86%]
* Polynominal Regression [98%]
  (Gives best solution for Training) ⇒ [OverFitting]
  But Less in Test Accuracy

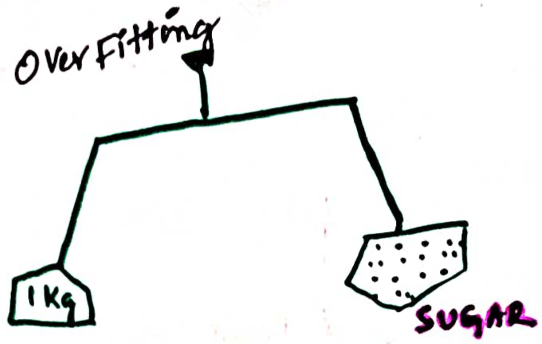


Ans :- "Regularization"
* Minimizing model Complexity
* Penalizing The loss function
* Reducing model overfitting [Add more bias to Reduce model varience].

*** Loss function : $\left[\sum [y-\hat{y}]^2\right]_{min}$

$[S.S.E]_{min}.$

Ex :-



Variance High

Bias low

↓

We are adding bias To Balance [Add "Penality Term"]

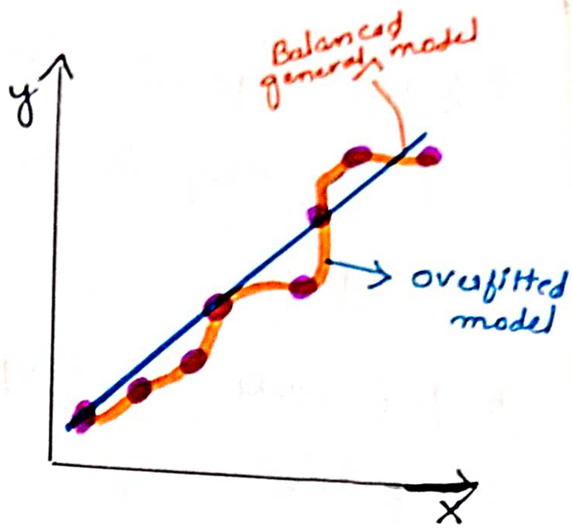
⇒ Three Main types of "Regularization":

1. L_1 Regularization [Lasso Regression]
2. L_2 Regularization [Ridge Regression]
3. Combining L_1 and L_2 [Elastic Net]

L_2 Regularization

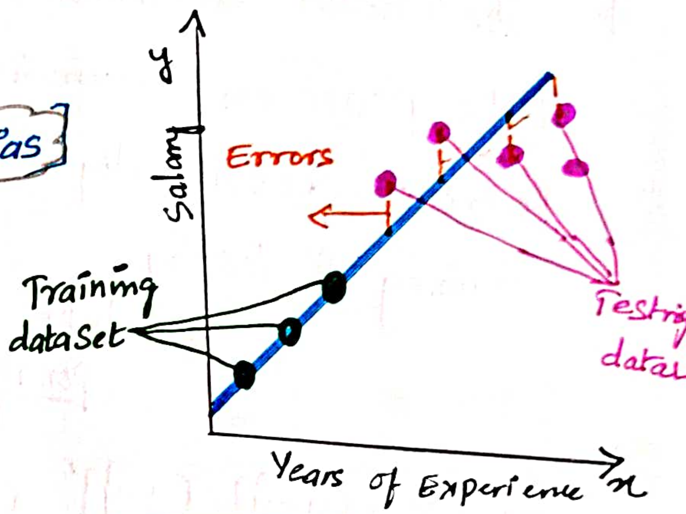
– Ridge Regression

Ridge regression Advantage is to avoid Overfitting Problem



Overfitting : performs good in Training data and poor in test Data.

Ridge regression works by applying a penalizing term [Reducing the weights and bias] to overcome Overfitting



Least Sum of Squares is applied to obtain best fit line

Since The line passes through the 3 Training data points The Sum of Squared Residuals = 0

But, For Test data, we have More errors. ie $\boxed{\text{high}}$

$\boxed{\text{variance}}$ ↳ it means that there is a difference in $\boxed{\text{fit}}$ (or) (variability) between the training dataset and Testing dataset.

Ridge regression works by attempting at increasing the bias to improve variance

This works by changing the slope of line

The model performance might be little poor on Training. But it perform consistently well on both Training & Testing

Slope has been reduced with ridge regression penalty and therefore the model becomes less sensitive to changes in independent variables.

Ex :-

x	y	$[y-\hat{y}]$	$[y-\hat{y}]^2$	$\alpha(\text{slope})^2 = \alpha(\beta_1)^2$

$\Sigma(y-\hat{y})^2$

$\therefore y = mx + c$

$y = \beta_0 + \beta_1 x$

$\boxed{\Sigma[y-\hat{y}]^2 + \alpha(\text{slope})^2}_{min}$

EX:

$$\begin{bmatrix} S_1 \text{ topper} + S_4 \text{ weak} \\ S_2 \text{ topper} + S_5 \text{ weak} \\ S_3 \text{ topper} + S_6 \text{ weak} \end{bmatrix}_{min}$$

* Loss function (For regression Line) $= \left[\Sigma[y-\hat{y}]^2\right]_{min}$

* Loss function (Ridge Regression) $= \left[\Sigma[y-\hat{y}] + \alpha(\text{slope})^2\right]$

Overall minimum

Least Squares Regression

min (Sum of Squared residuals)
[S.S.E]min

Ridge regression

min (Sum of Squared residuals + $\boxed{\alpha * slope^2}$)

not Fixed Value

Penality Term

$[S.S.E + \alpha (slope^2)]_{min}$

$y = \beta_0 + \beta_1 x$

Tuning ⇒ which is best one (or) value
Ex : Radio button.

L_2 regularization adds a penalty equal to the Square of the magnitude of coefficient

All coefficients are shrunk by the Same Factor

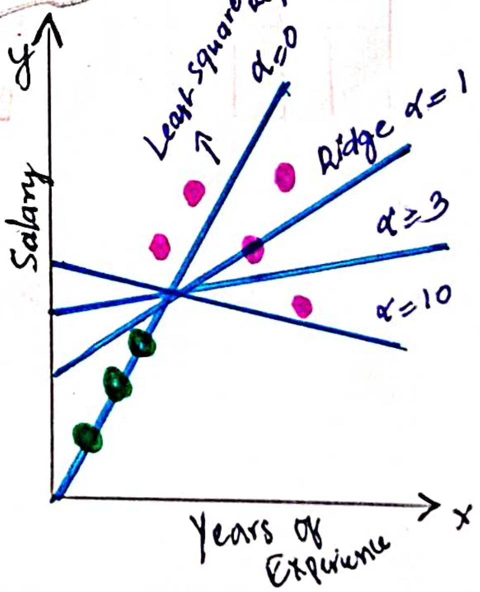
does not necessarily Eliminate coefficient

only one by all
$\alpha(\beta_1+\beta_2+\beta_3)$

Generally
" α " Value should be between "0 to 1"

As Alpha (α) increases, the slope of regression line is reduced and Become Horizontal.

As Alpha increases, the model becomes Less Sensitivity to variations of independent Variable (# Years of Exp)



L1 Regularization

— Lasso Regression.

Lasso regression is similar to Ridge regression.

it works by introducing a bias Term but instead of squaring slope The Absolute Value of Slope is added as penality term

"Apply modulo" To slope.

→ **Lasso Regression**

$$\text{Min}\left(\text{Sum of Squared residuals} + \underbrace{\alpha * |\text{slope}|}_{\text{Penality Term}}\right)$$

Ex :-

Tv	R	New	Sale

MLR

PR

Lasso Reg

Ridge reg

Elastic reg

dataScience	D.S	D.S
Lasso α	Ridge α	Elastic α
1st	1st	1st

↓

Exam

↓

1st

16/4/22

11:44pm

Dt: 17/4/22
6:30pm

* Least square Regression.

Min (Sum of The squared Residuals)

* Ridge Regression

Min [Sum of the squared residuals + {α * slope²}]

* Lasso Regression

Min [Sum of squared Residuals + {α * |slope|}]

Ridge Regression can reduce the slope close to zero (but not exactly zero) but Lasso Regression can reduce the slope to be exactly equal to zero.

HYPER TUNING :- [RIDGE, LASSO]

Step 1 Problem Understanding

Step 2 Data collection.

df = Pd. read_csv ("Adversting . csv")

df. head ( )

STep 2.2 DataSet Understanding

df.info()

STep 2.3 Data Understanding

df.shape

STep 3.1 Exploratory Data Analysis [EDA]

df.describe()
sns.pairplot(df)
df.corr()

STep 3.2 Data Cleaning

df.isnull.sum()

STep 3.3 Data Wrangling

STep 3.4 Train_test_split

X = df[["TV", "radio", "newspaper"]]

Always "X" be Two Dimensional Array.

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 = 29)

STEP : 4   MODELLING with default Parameters.

→ Ridge Regression

from sklearn.linear_model import Ridge

ridge_model = Ridge() → Help : Default Alpha = 1.0

ridge_model.fit (x_train, y_train)

$\boxed{\text{out}}$: Ridge ()

Prediction & Evaluation

test_predictions = ridge_model.Predict (x_test)

train_predictions = ridge_model.Predict (x_train)

from sklearn.metrics import mean_squared_error

test_rmse = np.sqrt (mean_squared_error (y_test, test_predi-ction

train_rmse = np.sqrt (mean_squared_error (y_train, train predictions)

print ("train RMSE :", train_rmse)

print ("test RMSE :", test_rmse)

$\boxed{\text{out}}$: train RMSE : 1.6408
Test RMSE : 1.780

R^2

```
# Print ("Train_R2":, ridge_model.score (x_train, y_train)
# print ( "Test_R2":, ridge_model. score (x_test, y_test)
```

Out:
Train_R2 : 0.888
Test_R^2 : 0.905

⇒ Apply "FOR Loop" for different Alpha values

```
for i in range (1,10):
    # ridge_model = Ridge (alpha = i)
    # ridge_model . fit (x_train, y_train)

    # train_predictions = ridge_model. predict (x_train)
    # test_predictions = ridge_model. predict (x_test)

    # Print (i)
    # print ("train_R²": ridge_model. Score (x_train, y_train)
    # print ("test_R²": ridge_model. score (x_test, y_test)
```

Out:

1. Train_R^2 : 0.888
   Test_R^2 : 0.905
2. Train_R^2 : 0.888
   test_R^2 : 0.905
3. Train_R^2 : 0.898
   Test_R^2 : 0.905
4. Train_R^2 : 0.888
   Test_R^2 : 0.905
5. Train_R^2: 0.888
   Test_R^2: 0.905
6. Train_R^2 : 0.898
   Test_R^2: 0.905
7. Train_R^2: 0.888
   Test_R^2: 0.905
8. Train R^2: 0.888
   Test_R^2: 0.905
9. Train_R^2 : 0.888
   Test_R^2 : 0.905

Dt:- 18/oct/22
11: 30pm

⇒ **Hyper Parameter tunning**

⇓

model Parameter

Ex:- Machine Learning Algorithm : Ridge ( ) → Model

Help:

Ridge (
alpha = 1.0, → default Parameters
*,
fit_intercept = True,
normalize = False,
⋮
)

∴ α (Alpha) = hyperparameter.

*** ⇒ Grid Search CV (instead of For loop & Best one & CV) → cross validation.

Sequential order, which is best (given data)

Identifying the best hyper parameter

estimator = Ridge ( )   (# MODEL, # model name)

param_grid = {"alpha": [0, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, 10]}   (# Parameter grid; Dictionary; list)

from Sklearn . model_selection import Grid Search CV   (capital)

model_hp = Grid Search CV (estimator, param_grid) → fold = 5

model_hp . fit (x_train, y_train)

Out: GridSearchCV (estimator = Ridge ( ),
Param_grid = {'alpha': [0, 0.1, 0.2, 0.3, 0.4, 1, 2, 10]}

model_hp.best_params_

Out: {"alpha" : 10}

model_hp.best_score_ → Accuracy

Out: 0.8831

→ Lasso Regression

MODEL.ING

```
from sklearn.linear_model import Lasso

# lasso_model = Lasso()
# lasso_model.fit(x_train, y_train)
```

Out: Lasso()

Prediction & Evaluation

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

from sklearn.metrics import mean_squared_error

# train_rmse = np.sqrt(mean_squared_error(y_test, test_predictio
# test_rmse = np.sqrt(mean_squared_error(y_train, train_predictions
```

Print ("train RMSE": train_rmse)
print (" test RMSE : test_rme)

Out : Train RMSE : 1.642
Test RMSE : 1.768

→ Elastic Net Regression

MODELING

from Sklearn.linear_model import ElasticNet

enr_model = ElasticNet()
enr_model. fit (x_train, y_train)

Out: Elastic Net ()

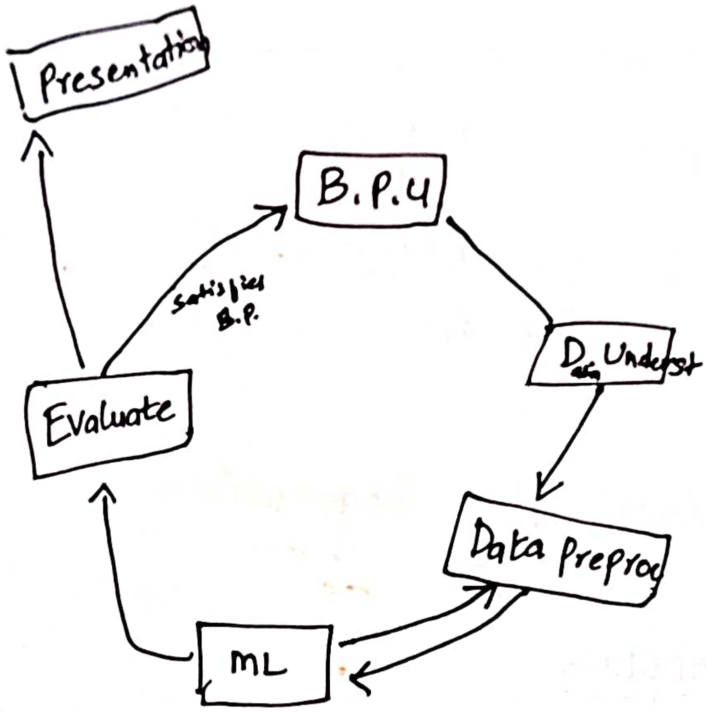
Predictions & Evaluating

train_Predictions = enr_model.Predict (x_train)
test_Predictions = enr_model. Predict (x_test)

from sklearn.metrics import mean_squared_error

train_rmse = np. sqrt (mean_squared_error (y_train, train_predictions
test_rmse = np. sqrt (mean_squared_error (y_test, test_Predictions)
Print ("train RMSE:", train_rmse)
Print ("test RMSE:", test_rmse)

Out : Train RMSE : 1.6414
Test RMSE : 1.7739



Dataset (B.P.4)

1. Load (import data)
2. Load data (read)
3. Understand data
4. data set — Continous / Disrete
5. EDA (past)
6. Data cleaning
7. Data Wrangling
8. Machine learning
9. Evaluation
10. Save model

	Default	Hyper Tuning Parameter
M.L.R (multiple lin	Test R^2 = Train R^2 = C.V =	Ex: Test R^2 Train R^2 = 86 C.V = 90
P.R (polynomial)	Test R^2 = Train R^2 = C.V =	degree = ? Train R^2 = Test R^2 = 88 C.V = 90
Lasso Regression	Test R^2 = Train R^2 = C.V =	α = ? 10 Train R^2 = Test R^2 = 90 ✓ C.V = 88
Ridge regression	Test R^2 = Train R^2 = C.V =	α = ? Train R^2 = Test R^2 = C.V =
Elastic Net regression	Test R^2 = Train R^2 = C.V =	α = ? Train R^2 = Test R^2 = C.V =

R^2 should be high. after (C.V) should be considered

Ex: final = (Lasso / α = 10) → model.

18/4/22 2:04pm