Kmit

SONET

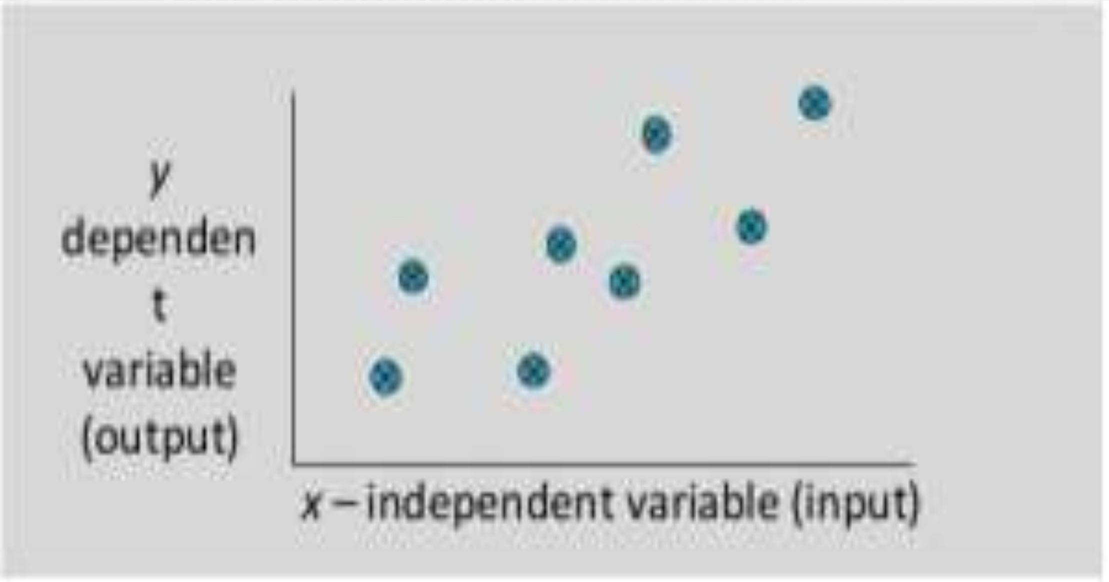
Data Science

(Level-I)

Regression Metrics

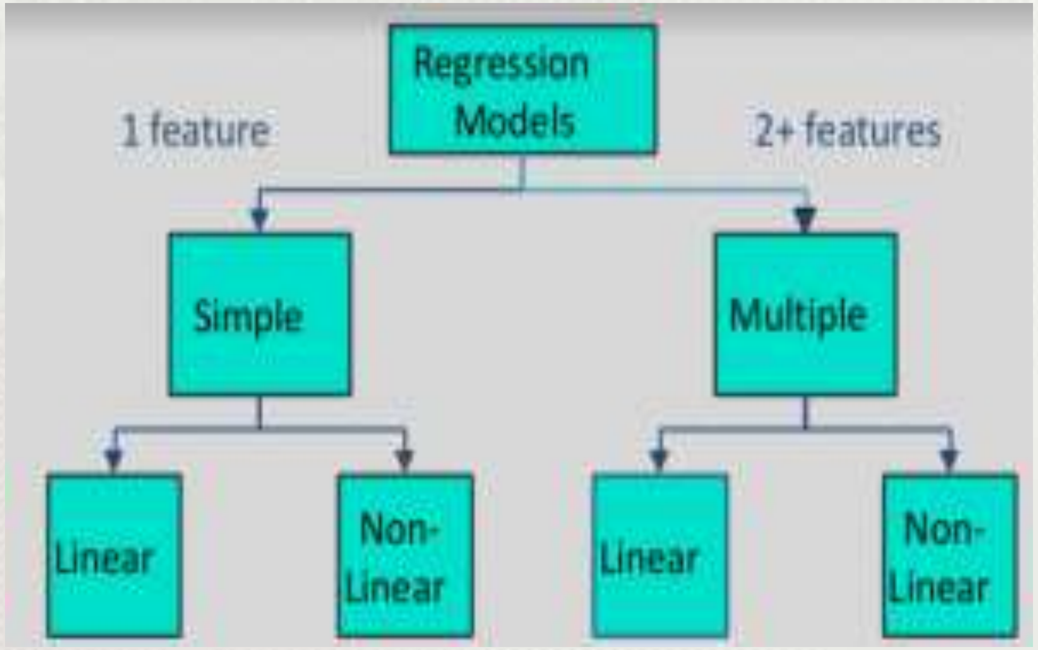
Introduction

- A regression problem is when the output variable is a real or continuous value, such as "salary" or "weight".
- Many different models can be used, the simplest is the linear regression.
- It tries to fit data with the best hyper-plane which goes through the points.

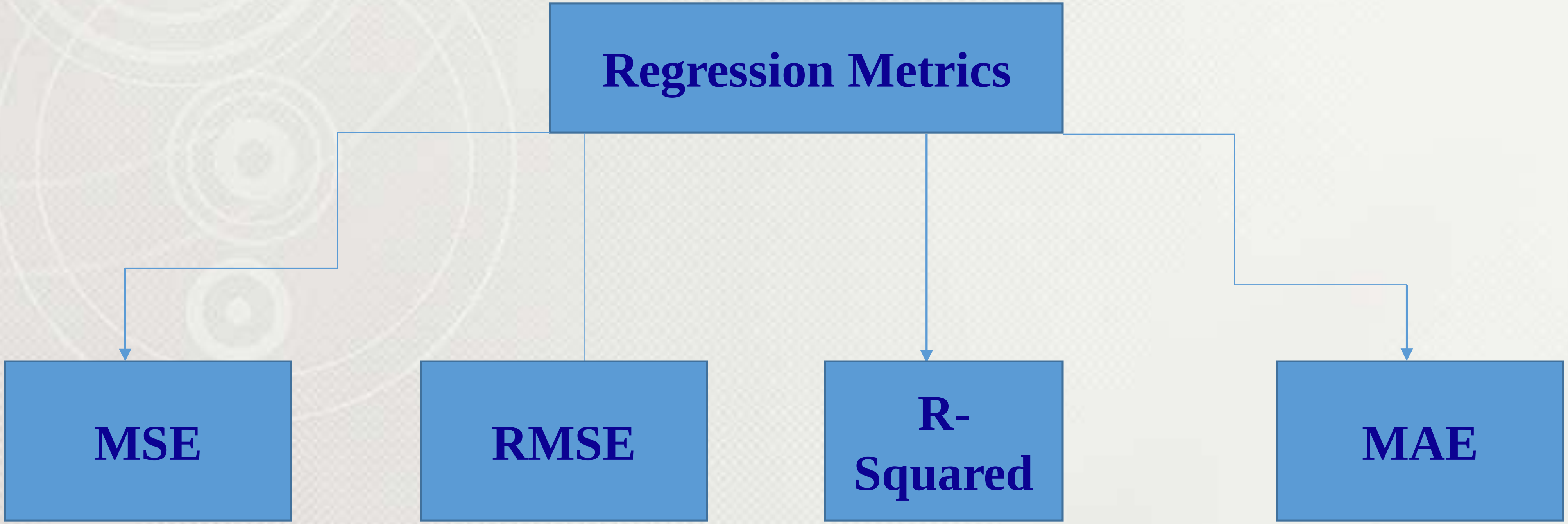


Introduction

Types Of Regression Models:



Introduction



MSE: Mean Square Error
RMSE: Root Mean Square Error
MAE: Mean Absolute Error

Mean Square Error

- The mean squared error tells how close a regression line is to a set of points.
- It does this by taking the distances from the points to the regression line and squaring them.
- The squaring is necessary to remove any negative signs. It also gives more weight to larger differences.
- ➢ The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better The fact that MSE is almost always strictly positive (and not zero).

Mean Square Error

Definition:

If $Y^{\wedge}$ is a vector predictions generated from a sample of n data points on all variables, and Y is the vector of observed values of the variable being predicted, then within-sample MSE of the predictor is computed as

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

n- Total Sample
y-independent variable(input sample)
$y^{\wedge}$-dependent(predicted)

Mean Square Error

Examples:

```
In [1]: import numpy as np
        from sklearn.metrics import mean_squared_error
        x=np.array([1,2,3,4,5,6,7])
        y=np.array([1,2,3,4,5,6,7])
        m2=mean_squared_error(x,y)
        print("mse ",m2)

        mse  0.0
```

```
In [2]: x=[1,2,3,4,5,6,7]
        y=[7,6,5,4,3,2,1]
        m2=mean_squared_error(x,y)
        print("mse ",m2)

        mse  16.0
```

Root Mean Square Error (RMSE):

- RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It’s the square root of the average of squared differences between prediction and actual observation.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

- n- Total Sample
- y- Independent variable (input sample)
- y^- dependent(predicted)
- We can say RMSE=Sqrt(MSE)

Kmit SONET

Machine Learning

Example:

In [3]:
```
import numpy as np
from sklearn.metrics import mean_squared_error
x=np.array([1,2,3,4,5,6,7])
y=np.array([1,2,3,4,5,6,7])
m2=mean_squared_error(x,y)
rmse=np.sqrt(m2)
print("rmse value ",rmse)
```

```
rmse value  0.0
```

In [4]:
```
x=np.array([1,2,3,4,5,6,7])
y=np.array([6,3,8,2,9,1,8])
m2=mean_squared_error(x,y)
rmse=np.sqrt(m2)
print("rmse value ",rmse)
```

```
rmse value  3.722518348798681
```

R-squared

- R-squared is a statistical measure of how close the data are to the fitted regression line.
- The definition of R-squared is fairly straight-forward- it is the percentage of the response variable variation that is explained by a linear model. Or:
    - R-squared = Explained variation / Total variation

➢ R-squared is always between 0 and 100%:

- 0% indicates that the model explains none of the variability of the response data around its mean.
- 100% indicates that the model explains all the variability of the response data around its mean.

R-squared

Formula for R-Squared is

$$R2 = 1 - \frac{SSE}{SST}$$

Where SSE=$\sum(y - y^{\wedge})^2$

SST=$\sum(y - y^{-})^2$

y---is actual value

$y^{\wedge}$---is the predicted value of y

y^{-}---is the mean of the y values

R-squared

Example:

```
In [6]: import numpy as np
        from sklearn.metrics import r2_score
        from sklearn.metrics import mean_squared_error
        x=np.array([1,2,3,4])
        y=np.array([1,2,3,4])
        r=r2_score(x,y)
        print("r squared ",r)

r squared  1.0

In [7]: x=np.array([1,2,3,4])
        y=np.array([4,2,1,3])
        r=r2_score(x,y)
        print("r squared ",r)

r squared  -1.7999999999999998
```

Mean Absolute Error (MAE)

- MAE measures the average magnitude of the errors in a set of predictions, without considering their direction.
- It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Example:

```
In [9]: import numpy as np
        from sklearn.metrics import mean_absolute_error
        x=np.array([1,2,3,4,5,6,7])
        y=np.array([6,4,3,2,7,1,5])
        mae=mean_absolute_error(x,y)
        print("mean_absolute_error ",mae)

mean_absolute_error  2.5714285714285716

In [10]: x=np.array([1,2,3,4,5,6,7])
         y=np.array([1,2,3,4,5,6,7])
         mae=mean_absolute_error(x,y)
         print("mean_absolute_error ",mae)

mean_absolute_error  0.0
```


THANK
YOU