WEEK-6
MODEL EVALUATION AND TESTING
Session-4

1. EVALUATION METRICS OF REGRESSION MODEL

Evaluation metrics are a measure of how good a model performs and how well it approximates the relationship.
A good regression model is one where the difference between the actual or observed values and predicted values for the selected model is small and unbiased for train, validation and test data sets. To measure the performance of regression model, some statistical metrics are used.

Some of the evaluation metrics are,

- Root Mean Squared Error (RMSE)
- R Square(R2)

ROOT MEAN SQUARED ERROR (RMSE)

As RMSE is clear by the name itself, that it is a simple square root of mean squared error(MSE). Mean squared error states that finding the squared difference between actual and predicted value. It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE. The lower the RMSE, the better a model fits a dataset.

$$MSE = \Sigma(\hat{y}_i - y_i)^2 / n$$

$$RMSE = \sqrt{}\Sigma(\hat{y}_i - y_i)^2 / n$$

where:

Σ is a symbol that means "sum"
$\hat{y}_i$ is the predicted value for the ith observation
y_i is the observed value for the ith observation
n is the sample size.

#python code of RMSE

```
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
```

R SQUARE(R2)

It is a statistical measure of how well the regression line approximates the actual data. It is therefore important when a statistical model is used either to predict future outcomes or in the testing of hypotheses.

#python code of R2

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test,y_pred)
print(r2)
```

Importing Evaluation Metrics

```
from sklearn.metrics import r2_score,mean_squared_error
import numpy as npdef run_experiment(model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print("R^2 : ", r2_score(y_test, y_pred))
    print("RMSE:",np.sqrt(mean_squared_error(y_test, y_pred)))
```

GRADIENT DESCENT

In linear regression, the model targets to get the best-fit regression line to predict the value of y based on the given input value (x). While training the model, the model calculates the cost function which measures the Root Mean Squared error between the predicted value (pred) and true value (y). The model targets to minimize the cost function.

The class Stochastic Gradient Descent Regressor (SGDRegressor) implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties to fit linear regression models. SGDRegressor is well suited for regression problems with a large number of training samples (> 10.000), for other problems we recommend Ridge, Lasso, or ElasticNet.

The concrete loss function can be set via the loss parameter. SGDRegressor supports the following loss functions:

- loss="squared_error": Ordinary least squares,
- loss="huber": Huber loss for robust regression,
- loss="epsilon_insensitive": linear Support Vector Regression.

The penalty parameter determines the regularization to be used.
The major advantage of SGD is its efficiency, which is basically linear in the number of training examples. If X is a matrix of size (n, p) training has a cost of $O(kn\bar{p})$, where k is the number of iterations (epochs) and $\bar{p}$ is the average number of non-zero attributes per sample. Recent theoretical results, however, show that the runtime to get some desired optimization accuracy does not increase as the training set size increases.

SGDRegressor provide two criteria to stop the algorithm when a given level of convergence is reached:

- With early_stopping=True, the input data is split into a training set and a validation set. The model is then fitted on the training set, and the stopping criterion is based on the prediction score (using the score method) computed on the validation set. The size of the validation set can be changed with the parameter validation_fraction.

- With early_stopping=False, the model is fitted on the entire input data and the stopping criterion is based on the objective function computed on the training data.

2. MULTIPLE LINEAR REGRESSION

Multiple Linear Regression (MLR) is a statistical method for estimating the relationship between a dependent variable and two or more independent (or predictor) variables or MLR is a method for studying the relationship between a dependent variable and two or more independent variables.

MLR Uses the ordinary least squares solution (as does simple linear or bi-variable regression) and describes a line for which the (sum of squared) differences between the predicted and the actual values of the dependent variable are at a minimum.
MLR represents the function that minimizes the sum of the squared errors.

$$Ypred = a + b1X1 + B2X2 \dots + BnXn$$

MLR produces a model that identifies the best weighted combination of independent variables to predict the dependent (or criterion) variable. It estimates the relative importance of several hypothesized predictors and assess the contribution of the combined variables to change the dependent variable.

ASSUMPTIONS OF MULTIPLE LINEAR REGRESSION

- **Independence**: The data of any particular subject are independent of the data of all other subjects.

- **Normality**: In the population, the data on the dependent variable are normally distributed for each of the possible combinations of the level of the X variables; each of the variables is normally distributed.

- **Homoscedasticity**: In the population, the variances of the dependent variable for each of the possible combinations of the levels of the X variables are equal.

- **Linearity**: In the population, the relation between the dependent variable and the independent variable is linear when all the other independent variables are held constant.

NORMAL EQUATION OF MULTIPLE LINEAR REGRESSION

Normal Equation is an analytical approach to Linear Regression with a Least Square Cost Function. We can directly find out the value of θ without using Gradient Descent. Following this approach is an effective and a time-saving option when are working with a dataset with small features.
Normal Equation is a follows:

$$\theta = \left(X^T X\right)^{-1} \cdot \left(X^T y\right)$$

In the above equation,
θ : hypothesis parameters that define it the best.
X : Input feature value of each instance.
Y : Output value of each instance

Consider a dataset,

area (x_1)	**rooms (x_2)**	**age (x_3)**	**price (y)**
23	3	8	6562
15	2	7	4569
24	4	9	6897
29	5	4	7562
31	7	6	8234
25	3	10	7485

Here area, rooms, age are features / independent variables and price is the target / dependent variable. As we know the hypothesis for multiple linear regression is given by:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_n x_n$$

Parameters: $\Theta = \theta_0, \theta_1 \ldots\ldots, \theta_n$

Features: $x = x_0, x_1, \ldots \ldots, x_n$

where, $x_0 = 1$

NOTE: Here our target is to find the optimum value for the parameters θ. To find the optimum value for θ we can use the normal equation. So after finding the values for θ, our linear hypothesis or linear model will be ready to predict the price for new features or inputs.

Normal Equation is:

$$\theta = \left(X^T X\right)^{-1} \cdot \left(X^T y\right)$$

Considering the above data set we can write,

X: an array of all independent features with size (*n x m*) where *m* is a total number of training samples and *n* is the total number of features including ($x_0 = 1$)

X^T: Transpose of array X

y: *y* is 1D array/column array/vector of target/dependent variable with size *m* where *m* is a total number of training samples.

So for the above example we can write:

X =[[1, 23, 3, 8], [1, 15, 2, 7], [1, 24, 4, 9], [1, 29, 5, 4], [1, 31, 7, 6], [1, 25, 3, 10]]

X^T=[[1, 1, 1, 1, 1, 1], [23, 15, 24, 29, 31, 25], [3, 2, 4, 5, 7, 3], [8, 7, 9, 4, 6, 10]]

y=[6562, 4569, 6897, 7562, 8234, 7485]

After Implementation of Linear Regression Model with Normal Equation by using Python the expected output is

Output :

```
Value of Intercept = 305.3333333334813
Coefficients are = [236.85714286  -4.76190476 102.9047619 ]
Actual value of Test Data = [8234, 7485]
Predicted value of Test Data = [8232, 7241.52380952]
```

APPLICATIONS OF MULTIPLE LINEAR REGRESSION

1. Businesses often use regression to understand the relationship between advertising spending and revenue.
2. Medical researchers often use regression to understand the relationship between drug dosage and blood pressure of patients.
3. Agricultural scientists often use regression to measure the effect of fertilizer and water on crop yields.
4. Data scientists for professional sports teams often use regression to measure the effect that different training regimens have on player performance.