Unit-5

Supervised Learning: Regression Analysis

- Regression is the most popular algorithm in statistics and machine learning.
- In the machine learning and data science field, regression analysis is a member of the supervised machine learning domain that helps us to predict continuous variables such as stock prices, house prices, sales, rainfall, and temperature.

- Regression analysis identifies how the dependent variable depends upon independent variables.

- For example, say as an education officer you want to identify the impact of sports activities, smart classes, teacher-student ratio, extra classes, and teachers' training on students' results.

Linear regression

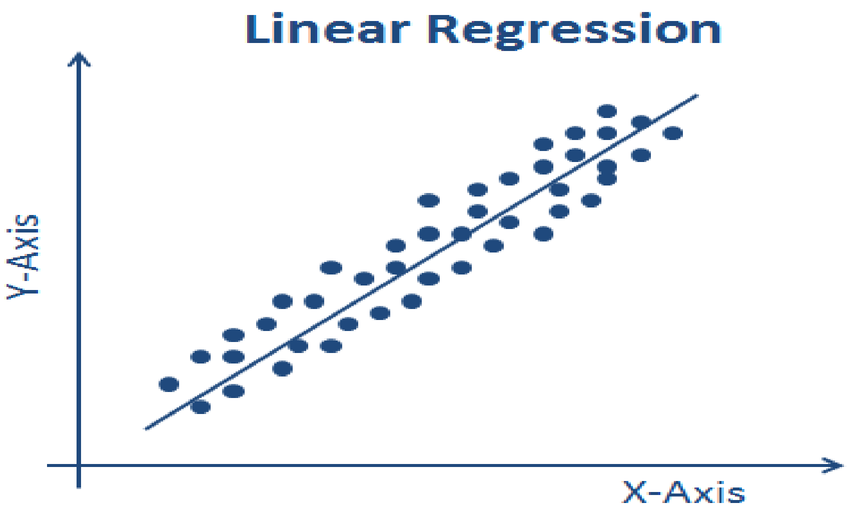
- Linear regression is a kind of curve-fitting and prediction algorithm.

- It is used to discover the linear association between a dependent (or target) column and one or more independent columns (or predictor variables).

- This relationship is deterministic, which means it predicts the dependent variable with some amount of error.

- In regression analysis, the dependent variable is continuous and independent variables of any type are continuous or discrete.

- Linear regression has been applied to various kinds of business and scientific problems, for example, stock price, crude oil price, sales, property price, and GDP growth rate predictions.



- The main objective is to find the best-fit line to understand the relationship between variables with minimum error.
- Error in regression is the difference between the forecasted and actual values.
- Coefficients of regression are estimated using the **Ordinary Least Square(**OLS) method. OLS tries to minimize the sum of squares residuals.
- Let's see the equation for the regression model.

$$y = \beta_0 + \beta_1 x + \varepsilon$$

- Here, x is the independent variable and y is a dependent variable β_0 intercepts are the coefficient of x, and (the Greek letter pronounced as epsilon) is an error term that will act as a random variable.

Multiple linear regression

- MLR is a generalized form of simple linear regression. It is a statistical method used to predict the continuous target variable based on multiple features or explanatory variables.
- The main objective of MLR is to estimate the linear relationship between the multiple features and the target variable.
- MLR has a wide variety of applications in real-life scenarios. The MLR model can be represented as a mathematical equation:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon$$

- Here, x_1, x_2,…x_p are the independent variables and y is a dependent variable.
- β_0 intercepts are coefficients of *x* and (the Greek letter pronounced as epsilon) is an error term that will act as a random variable.

Understanding multicollinearity

- Multicollinearity represents the very high intercorrelations or inter-association among the independent (or predictor) variables.
- Multicollinearity takes place when independent variables of multiple regression analysis are highly associated with each other. This association is caused by a high correlation among independent variables.
- This high correlation will trigger a problem in the linear regression model prediction results.
- It's the basic assumption of linear regression analysis to avoid multicollinearity for better results:

1. It occurs due to the inappropriate use of dummy variables.
2. It also occurs due to the repetition of similar variables.
3. It is also caused due to synthesized variables from other variables in the data.
4. It can occur due to high correlation among variables.

- Multicollinearity causes the following problems:

1. It causes difficulty in estimating the regression coefficients precisely and
2. coefficients become more susceptible to minor variations in the model.
3. It can also cause a change in the signs and magnitudes of the coefficient.
4. It causes difficulty in assessing the relative importance of independent variables.

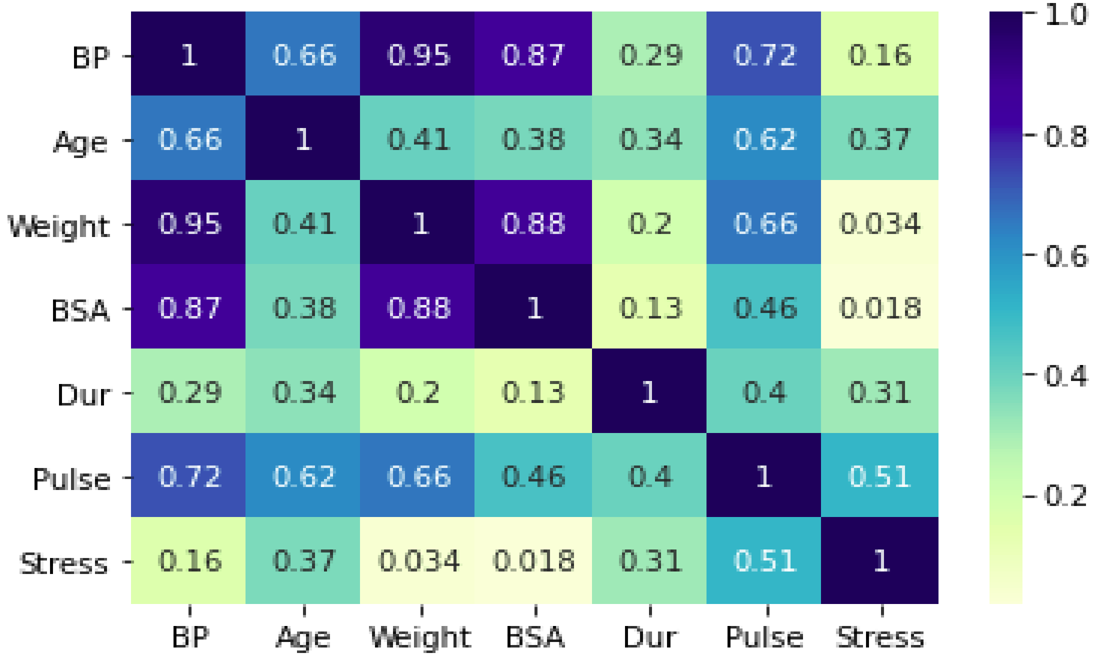
removing multicollinearity

Multicollinearity can be detected using the following:

- The correlation coefficient (or correlation matrix) between independent variables
- **Variance Inflation Factor** (**VIF**)
- Eigenvalues

- Correlation coefficients or correlation matrices will help us to identify a high correlation between independent variables.
- Using the correlation coefficient, we can easily detect the multicollinearity by checking the correlation coefficient magnitude

```
# Import seaborn and matplotlib
import seaborn as sns
import matplotlib.pyplot as plt
# Correlation matrix
corr=data.corr()
# Plot Heatmap on correlation matrix
sns.heatmap(corr, annot=True, cmap='YlGnBu')
# display the plot
plt.show()
```



Dummy variables

- Dummy variables are categorical independent variables used in regression analysis. It is also known as a Boolean, indicator, qualitative, categorical, and binary variable.
- Dummy variables convert a categorical variable with N distinct values into N–1 dummy variables.
- It only takes the 1 and 0 binary values, which are equivalent to existence and nonexistence.

- pandas offers the get_dummies() function to generate the dummy values.

```
# Dummy encoding
encoded_data = pd.get_dummies(data['Gender'])
# Check the top-5 records of the dataframe
encoded_data.head()
```

Output:

	F	M
0	1	0
1	0	1
2	0	1
3	1	0
4	0	1

- We can remove one column to avoid collinearity using the drop_first=True argument and drop first the N–1 dummies out of N categorical levels by removing the first level:

```
# Dummy encoding
encoded_data = pd.get_dummies(data['Gender'], drop_first=True)
# Check the top-5 records of the dataframe
encoded_data.head()
```

Output:

	M
0	0
1	1
2	1
3	0
4	1

Developing a linear regression model

- Build the regression model using the scientific toolkit for machine learning (scikitlearn):

1. We will first load the dataset using the read_csv() function:

```
# Import pandas
import pandas as pd
# Read the dataset using read_csv method
df = pd.read_csv("Advertising.csv")
# See the top-5 records in the data
df.head()
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

2. In this step, we will split the data two times:

- Split into two parts: dependent or target variable and independent variables or features.
- Split data into training and test sets. This can be done using the following code:

```
# Independent variables or Features
X = df[['TV', 'Radio', 'Newspaper']]
# Dependent or Target variable
y = df.Sales
```

- We will split the data into train and test sets in a 75:25 ratio using train_test_split(). The ratio can be specified using the test_size parameter and random_state is used as a seed value for reproducing the same data split each time.

```
# Lets import the train_test_split method
from sklearn.model_selection import train_test_split
# Distribute the features(X) and labels(y) into two parts training
and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=0)
```

3. Let's import the LinearRegression model, create its object, and fit it to the training dataset (X_train, y_train)

```
# Import linear regression model
from sklearn.linear_model import LinearRegression
# Create linear regression model
lin_reg = LinearRegression()
# Fit the linear regression model
lin_reg.fit(X_train, y_train)
# Predict the values given test set
predictions = lin_reg.predict(X_test)
# Print the intercept and coefficients
print("Intercept:",lin_reg.intercept_)
print("Coefficients:",lin_reg.coef_).
```

output:

- Intercept: 2.8925700511511483
- Coefficients: [0.04416235 0.19900368 0.00116268]

Evaluating regression model performance

- Model evaluation is one of the key aspects of any machine learning model building process.
- It helps us to assess how our model will perform when we put it into production.

We will use the following metrics for model evaluation:

- **R-squared**
- **MSE**
- **MAE**
- **RMSE**

R-squared

- R-squared (or coefficient of determination) is a statistical model evaluation measure that assesses the goodness of a regression model.
- It helps data analysts to explain model performance compared to the base model.
- Its value lies between 0 and 1.
- A value near 0 represents a poor model while a value near 1 represents a perfect fit. Sometimes, R-squared results in a negative value. This means your model is worse than the average base model.

$$R - Square = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

- **Sum of Squares Regression** (**SSR**): This estimates the difference between the forecasted value and the mean of the data.
- **Sum of Squared Errors** (**SSE**): This estimates the change between the original or genuine value and the forecasted value.
- **Total Sum of Squares** (**SST**): This is the change between the original or genuine value and the mean of the data.

MSE

- MSE is an abbreviation of mean squared error.
- It is explained as the square of change between the original and forecasted values and the average between them for all the values:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y - \hat{y})^2$$

- Here, y is the original value and $\bar{y}$ is the forecasted value.

MAE

- MAE is an abbreviation of mean absolute error.
- It is explained as the absolute change between the original and forecasted values and the average between them for all the values:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y - \hat{y}|$$

- Here, y is the original value and $\bar{y}$ is the forecasted value.

RMSE

- RMSE is an abbreviation of root mean squared error. It is explained as the square root of MSE:

$$RMSE = \sqrt{MSE}$$

```
# Import the required libraries
import numpy as np
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
# Evaluate mean absolute error
print('Mean Absolute Error(MAE):', mean_absolute_error(y_test,predictions))
# Evaluate mean squared error
print("Mean Squared Error(MSE):", mean_squared_error(y_test, predictions))
# Evaluate root mean squared error
print("Root Mean Squared Error(RMSE):", np.sqrt(mean_squared_error(y_test,
predictions)))
# Evaluate R-square
print("R-Square:",r2_score(y_test, predictions))
```

Output:

```
Mean Absolute Error(MAE): 1.3000320919235455
Mean Squared Error(MSE): 4.0124975229171
Root Mean Squared Error(RMSE): 2.003121944095541
R-Square: 0.8576396745320893
```

fitting polynomial regression

- Polynomial regression is a type of regression analysis that is used to adapt the nonlinear relationships between dependent and independent variables.
- In this type of regression, variables are modeled as the *n*th polynomial degree.
- It is used to understand the growth rate of various phenomena, such as epidemic outbreaks and growth in sales.
- Let's understand the equation of polynomial regression:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon.$$

- Here, x is the independent variable and y is a dependent variable. The β_0 intercepts, β_1, β_2 ..., are a coefficient of *x* and (the Greek letter pronounced as epsilon) is an error term that will act as a random variable.