CHAPTER 2 LITERATURE REVIEW

2.1 GENERAL

In the present chapter, the previously attempted research studies focusing on developing the prediction model for the California bearing ratio value of various soil types are discussed. Additionally, this chapter also covers the literature of several artificial intelligence techniques adopted for developing the prediction equations for the CBR value of different soil types.

2.2 LITERATURE REVIEW ON CBR OF SOIL

California bearing ratio (CBR) test is labor intensive and time-consuming as discussed previously in section 1.3 of CHAPTER 1. Therefore, numerous researchers attempted to develop the prediction models for the CBR value from the laboratory test results and archival data of index and engineering properties of the soil. The correlation between CBR and standard parameters was proposed previously by many researchers for different soil types i.e. clayey, fine-grained, coarse-grained and granular soils, covering a definite range of index and engineering properties using graphical, statistical and artificial intelligence techniques. Prior to making a discussion on the existing literature of the prediction model for CBR value, a detailed investigation of the factors influencing and correlating with the CBR value of soil needs to be discussed.

2.2.1 Factors influencing the shear strength of soil

Shear strength of soil, in soaked or un-soaked condition, is influenced by many factors. However, the relationship obtained for the un-soaked strength may not be similar to the soaked strength of the soil as after soaking swelling of soil take place which disturbs the initial values of index and engineering properties of soil. Numerous researches are

available on the factors influencing the soil's un-soaked strength and a few investigators have discussed the soaked strength of the soil. The factors affecting the shear strength of soil particles depends upon the types of soil i.e. cohesive and cohesionless soil. Shear strength of cohesive soil is primarily controlled by moisture content, dry density, gradation, thixotropy, and soil structures, whereas size, shape and gradation of the individual particles, mineralogical composition and dry density are essentially controlling parameters for cohesionless soil (Langfelder and Nivargikar, 1967).

Chang (1990) found that beyond a certain range of transition fines content, soil strength increases with further increase in fines content. Below 25% of fine contents, shear strength of sand-clay mixtures is controlled by the sand (Vallejo and Mawby, 2000). Salgado et al. (2000) observed that when it is more than 20%, shear strength is affected by fine content. Kim et al. (2018) studied the influence of clay content on the shear strength of clay-sand mixtures in terms of direct shear and angle of repose test. Shear strength, estimated in terms of unconfined compressive strength, increases as the bentonite content increases (Ghazi, 2015). Nagaraj and Suresh (2018) found that clay minerals present in soil greatly influence the CBR value of soil samples. Among the three gradations of sand (fine, medium and coarse), Nagaraj (2016) observed that irrespective of clay minerals present in the mixtures medium sand is much significant to enhance the undrained strength of clay-sand mixtures. Research done by Cabalar and Mustafa (2017) on sand-clay mixtures shows that CBR values of sand-clay mixtures increase as the sand content increases. Increased level of coarse fractions beyond 10% decreases the CBR value (Sreenivasulu et al., 2014). Researchers in the past had investigated the influence of fines content (comprises of clay and silt size particles) and sand particles on the shear strength of several types of soil mixtures and based on that some significant findings were obtained.

The compacted structure of soil is initially controlled by the optimum moisture content and types of compaction used. Seed et al. (1961) show that irrespective of compaction types, soil structures get flocculated on the dry side of OMC while on the wet side of OMC compaction type produces dispersive soil structures. Because of more rigid nature of the soil skeleton on the dry side of OMC, soil exhibits higher shear strength than the dispersive nature obtained on the wet side of OMC (Langfelder and Nivargikar, 1967). Data presented by Seed and Monismith (1954) and Seed et al. (1961) state that for a given moisture content, the shear strength of soil increases as the dry density increases. The maximum enhancement in shear strength with the increase in dry density was observed at the lowest moisture content.

In addition to the above parameters, some other factors i.e. soaking period and surcharge load, also influence the soaked strength of the soil. Turnbull and Foster (1956) indicate that compacted soil doesn't retain its high shear strength after soaking, a considerable reduction in CBR value is always observed. Research conducted in the past (Chauhan, 2010; Razouki and Kuttah, 2004) shows that CBR of soil decreases as the soaking period increases. Nini (2018) found that the CBR of soil increases as the loading rings increase. This is because increasing the surcharge weight may result in minimum enhancement in the volume of the soil samples consequently CBR get increases.

2.2.2 Literature's correlation and estimation model for the CBR value

The significance for predicting the CBR value of soil was discussed briefly in section 1.3. Several researchers attempted to predict the CBR value for the various types of engineering soil (fine-grained, coarse-grained and granular soil for the base and sub-base material) using graphical, statistical and computation approaches. To the author's knowledge, the very first fame in the domain of predicting the CBR value was earned by Kleyn (1955). Earlier, he attempted to address the discrepancy in the CBR test, later

prepared a chart based on a nest of straight lines that relate CBR to PI and grading module for over 1000 soaked CBR tests obtained from road and airport work throughout central and southern Africa. The chart shown in Figure 2.1 was marked for base course material. The work done by Black (1962), which deals only with remolded inorganic cohesive and cohesionless fine-grained soils, showed that the CBR of soil is closely related to its bearing capacity and CBR of soil changes with a change in moisture content and degree of saturation. He suggested that the relationship between CBR and ultimate bearing capacity depends on the type of soil and method of compaction i.e. static or dynamic.

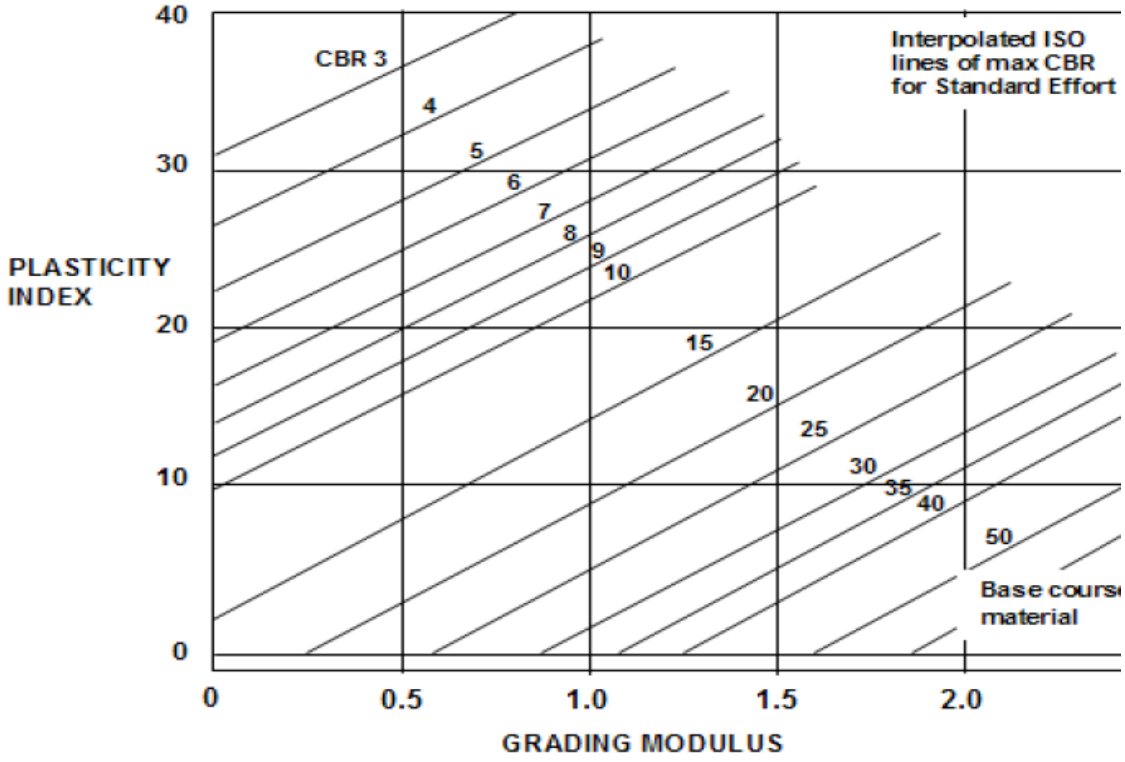


Figure 2.1 The relation of CBR against plasticity index and grading modulus **(Kleyn, 1955)**.

The graphically obtained correlation between the CBR and plasticity index for various consistency index (CI) values (can be calculated through equation (2.3) is shown in Figure 2.2, which is referred to the saturated soils. For the soils satisfying equation

(2.1), a multiplying factor (k) can be obtained directly from Figure 2.3 for CBR of unsaturated soils.

$$PI = 0.838\,LL - 14.2 \tag{2.1}$$

$$wPI = P_{200} \times PI \tag{2.2}$$

$$CI = \frac{LL - In\ situ\ moisture\ content}{PI} \tag{2.3}$$

Gawith and Perrin (1962) expressed the correlation for CBR using PI, linear shrinkage, grading module and percentages passing 2 mm, 0.425 mm and 0.075 mm sieves. The method was then republished in 1964 and 1980 by Victoria Country Roads Board (Victoria Country Roads Board, 1964, 1980). Sood et al. (1978) developed the equation from the sieve analysis results for moorums soil. Hight and Stevens (1982) tried to correlate CBR of a saturated clay with strength and stiffness, found that CBR doesn't correlate consistently with either strength or stiffness of clays. The relative impact of each varies from soil to soil and also with the structure in the same soil. Greenstein and Livneh (1975) correlated the CBR with the uniformity coefficient for dune sand.

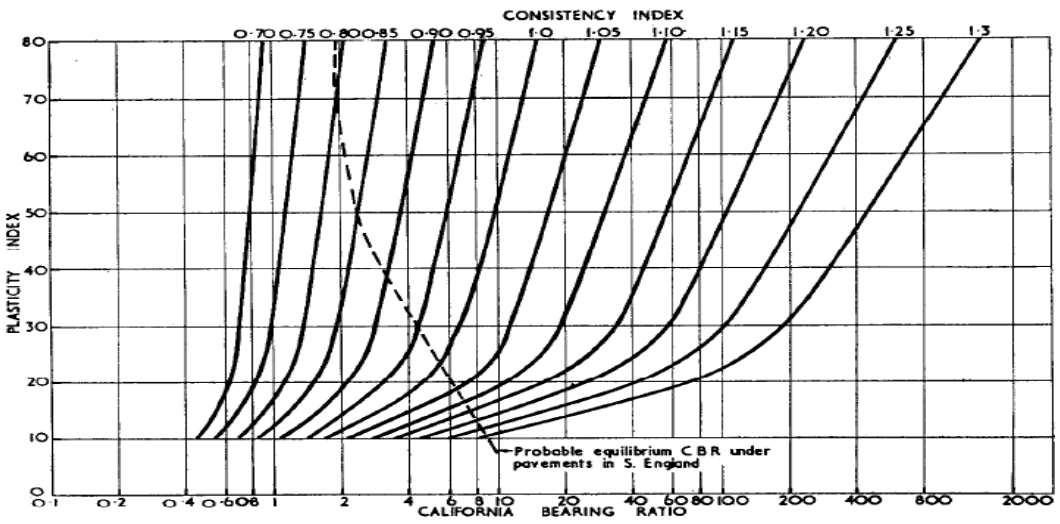


Figure 2.2 The relationship between CBR and plasticity index at various consistency indices **(Black, 1962)**.

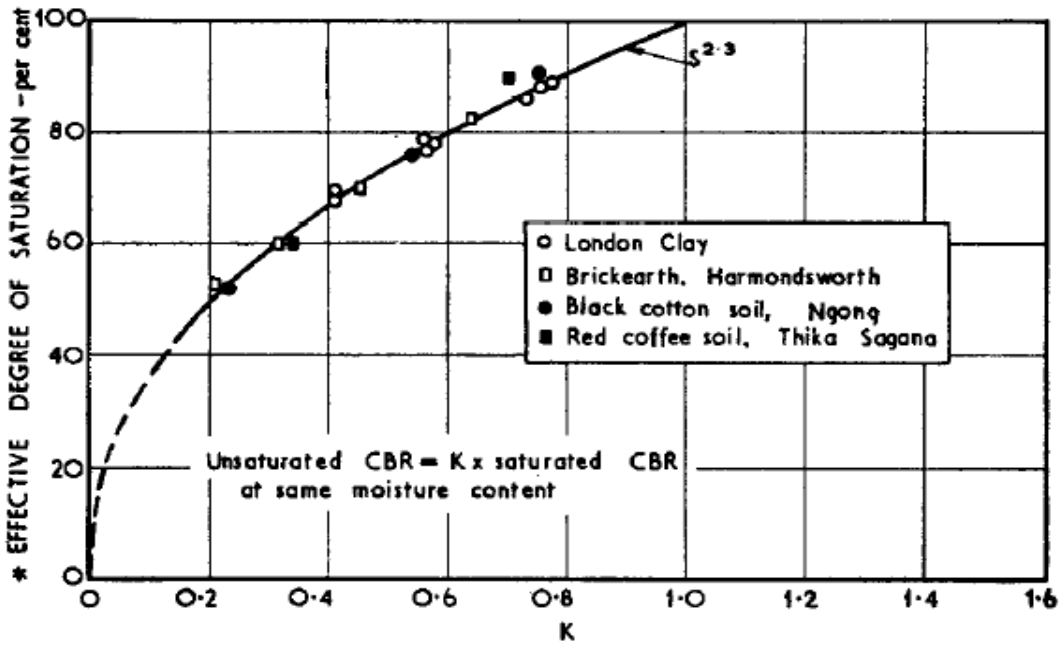


Figure 2.3 The relationship between saturated and un-saturated CBR of soil at same moisture content **(Black, 1962)**.

Agarwal and Ghanekar (1970) tried to generate a correlation equation (2.5) to (2.7) as per Table 2.1, through statistical analysis, between CBR and either LL, PL or PI for 48 soils collected from different parts of India. However, they could not find any significant correlation between these parameters. But when LL and OMC were added, they found an improved correlation with adequate accuracy for the preliminary identification of materials. Based on more than 300 laboratory test results, Doshi et al. (1983) studied various parameters of grain size distribution, includes grading constant, mean grain size, size factor and granulometric modulus, and compaction parameters (comprises of MDD and OMC) on the CBR of soils. Among the aforementioned grain size distribution parameters, grading constant was found to define the CBR more effectively. It was also found that CBR is much dependent on MDD and least on OMC. Moreover, the combined correlation of CBR with all three parameters i.e. grading constant, MDD and OMC, gives the best estimation. Stephens (1990) investigated

archival data collected from Natal Roads Department. The data were available in the standard summary sheet, which comprises of Atterberg's limits, particle size distribution, linear shrinkage, maximum swell, OMC and MDD and CBR values for each sample. The relationship between CBR and other parameters was described in both simple and multivariate forms; the models were unsatisfactory. In this study, a good correlation was found with the maximum swell, but it forms a part of the actual CBR test; therefore, it could not be used directly in any prediction model. The influence of clay proportions on CBR was reported and minimum CBR was proposed for shrinking and non-shrinking soils. Offbeat, Pandian et al. (1999) used the Proctor mould itself with a proportionately smaller plunger of 33.3 mm in diameter with a correction factor to simplify the CBR test, especially for fine-grained soils.

National Cooperative Highway Research Program (2001) tried to develop general correlations between soil index properties and CBR through the "Guide for Mechanistic-Empirical Design of New and Rehabilitated Pavement Structures". Two correlation equations were developed; one was for the coarse-grained material of non-plastic nature (wPI = 0) and another for materials containing more than 12% fines with some plasticity (wPI > 0). The equation (2.8) was for the coarse-grained material of non-plastic in nature (wPI = 0), which is limited to D_{60} (Diameter at 60% material passing from grain size distribution in mm) values lie between 0.01 mm and 30 mm. For material with D_{60} values less than 0.01 mm, the recommended CBR value is 5%, whereas a CBR value of 95% is recommended for material with D_{60} values greater than 30 mm. For the second group (fine-grained soils with plasticity, wPI > 0) the index properties chosen to correlate CBR were percentage passing No. 200 US sieve (P_{200}) and PI. These properties were combined into a parameter termed as wPI, defined in equation (2.2). The final correlation for CBR is represented in equation (2.9). Nomograph for computing the soaked CBR value from

sieve analysis dataset contained in the Operations Manual of the Pradhan Mantri Gram Sadak Yojana (PMGSY) is shown in Figure 2.4 (National Rural Road Development Agency, 2005).

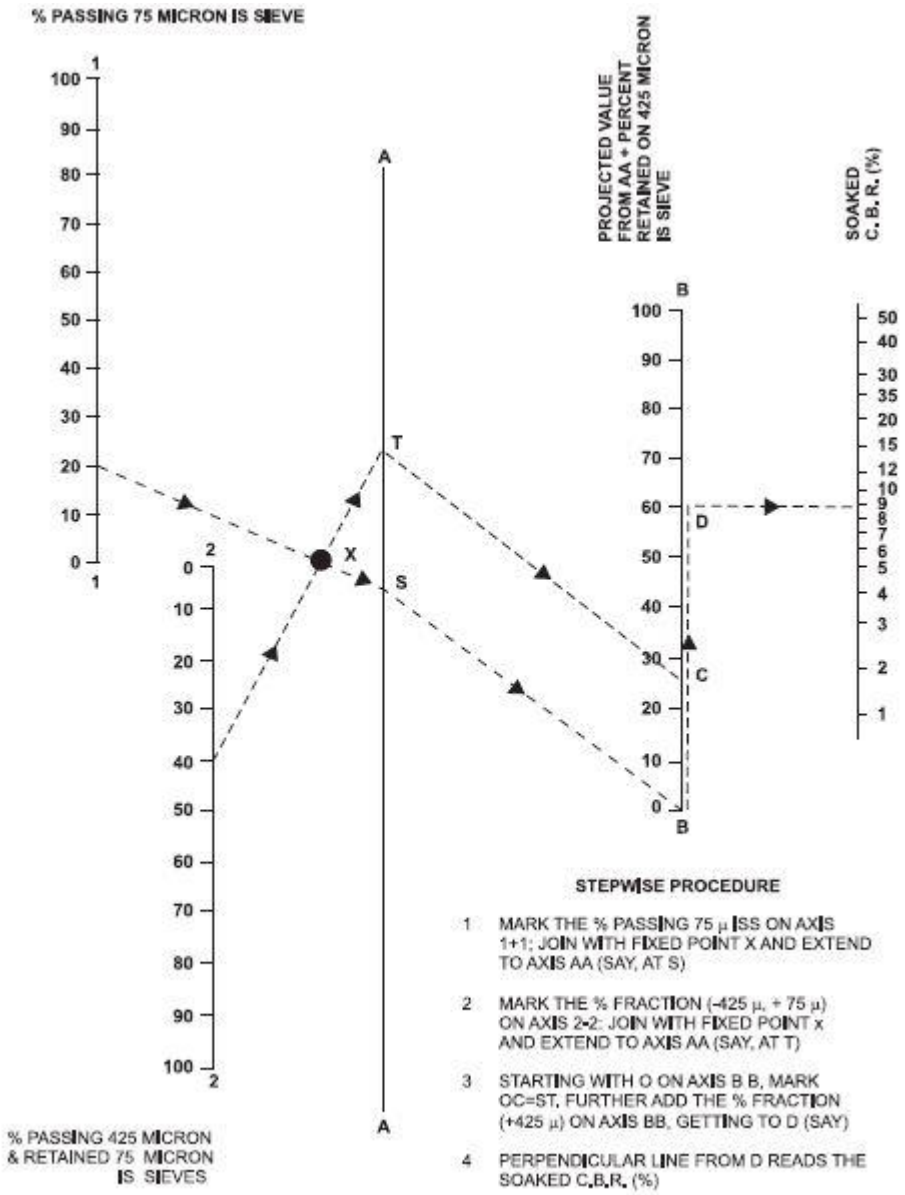


Figure 2.4 PMGSY developed Nomograph for computing the soaked CBR value.

Kin (2006) developed the equation (2.10) for 57 data samples of fine-grained soil collected from Malaysia. Breytenbach (2009) developed the correlation equation between the CBR and index testing for soils in South Africa. Patel and Desai (2010) correlated CBR value with MDD, OMC and PI of cohesive soils from the various zone of Surat city

(India). They found that CBR is influenced by Atterberg's limits, however doesn't vary with plasticity index. Alawi and Rajab (2013) studied the CBR prediction for sub-base layer materials using MLR models. The fitted regression model uses Los Angeles, OMC and MDD as input parameters as shown in equation (2.15). Yared (2013) developed the correlation equations (2.16) and (2.17) for CBR as a function of LL, PI and MDD. A moderate determination coefficient (R^2), R^2 0.458 and 0.629 for single regression and multiple regression analysis, respectively, was obtained. Deepak et al. (2014) developed the correlation equation (2.18) for CBR from the results of LL, PL, compaction parameters and percent fines of 81 datasets. Rakaraddi and Gomarsi (2015) established a relationship between CBR and different soil properties for fine-grained soils in India. They found that LL is the significant parameter for predicting the soaked CBR values followed by OMC, MDD and PI. A. u. Rehman et al. (2017) developed the correlation equation (2.22) for CBR of granular soil from the coefficient of uniformity and average particle size. A case study was carried by Bassey et al. (2017) for the correlation between index properties and CBR values of soils. Based on the 33 soil samples collected from the ongoing roadway construction project, Katte et al. (2019) observed that MLR analysis gives an improved correlation as compared to the SLR. The developed relationship, equation (2.30), for CBR versus PL, PI, OMC, MDD, % fine, % sand and % gravel is presented in Table 2.1.

As seen from the previous discussion, many investigators focused on the utilization of graphical and statistical techniques (comprises of simple linear and multiple linear regression analysis) for predicting the CBR value of various engineering soils. Among them some investigators developed significant models for predicting CBR from the index and engineering properties of soil. Variation in index and engineering properties of soil make the CBR more complex, hence, over a period of time, for different boundary

conditions the linearity of models developed through statistical techniques (as of having few limitations) might become non-generalized. Keeping that in mind, numerous researchers attempted to develop the models through some well-known soft computing techniques viz. artificial neural network (ANN), group method of data handling (GMDH), multivariate adaptive regression splines (MARS), support vector machine (SVM), genetic algorithm (GA), genetic programming (GP), gene expression programming (GEP) etc. which are discussed below, used for solving the various geotechnical and highway engineering problems. These techniques are the choice of the users because of their data-driven process ability as well as self-adaptive learning techniques (Dibike et al., 2001). Taskiran (2010) developed the correlation for 151 CBR test data, among 354 total tests data, of fine-grained soils which were belong to A-4, A-5, A-6, A-7 (AASHTO M 145) soil groups by Artificial Intelligence (AI) methods. Obtained data contains the results for LL, PL, PI, No: 200 sieve passing percentage (clay + silt), sand percent (S), and gravel percent (G), OMC and MDD. A total seven number of models were trained with different input parameters; the best results were obtained for both the techniques (ANN and GEP) with seven input parameters as shown in equation (2.11). The performance of models was evaluated in term of the coefficient of determination (R^2), mean square error (MSE), mean (Xmean), standard deviation (Sx), skewness (Csx), coefficient of variation (Cv), minimum (Xmin) and maximum (Xmax). Additionally, a sensitivity analysis was performed and it was found that dry unit weight is the most effective parameter and then plasticity index, OMC, sand content, clay + silt content, LL and gravel content respectively. Yildirim and Gunaydin (2011) had also studied the estimation of CBR by soft computing systems. SRA, MLR and ANN methods were applied on 124 compaction and soil classification data of different soil types (CH, CI, CL, GC, GM, GP-GC, MH, MI, ML, SC), collected from the public highway of

Turkey's different regions. A strong correlation (R^2 = 0.80-0.95) was found between Atterberg limits, sieve analysis, MDD and OMC by regression analysis and ANNs. Percentage of fine-grained and gravel was found as the dominant independent parameters in SRA, equation (2.12) and (2.13); percentage of sand, percentage of gravel, MDD and OMC in MLR analysis as shown in equation (2.14). The constructed ANN model shows higher performance than the statistical model (SRA and MLR) for CBR estimation. A comparative study of regression analysis and ANN conducted by Bhatt et al. (2014) on 124 soil samples exhibit that ANN can predict CBR more accurately in comparison to regression analysis. Erzin and Turkoz (2016) were the only researchers who considered the mineralogical parameters into the CBR estimation. They studied for nine different Aegean sands with varying different soil properties, using ANN and MRA. The soil properties, as well as the mineralogical properties, were used as input parameters simultaneously, as presented in equation (2.20). Farias et al. (2018) developed the prediction model (2.27) through (2.29) for CBR using the index properties of large number of soil samples using parametric and non-parametric techniques, local polynomial regression (LPR) and redial basic network (RBN). The obtained result shows that CBR is predominantly affected by PI of the soil as well as LPR model exhibits best accuracy among the other models. A comparative study conducted by Tenpe and Patel (2018) for 389 datasets collected from City and Industrial Development Corporation, Maharashtra state in India reveals that not much difference in the results of ANN and GEP for predicting the CBR value. Thereafter, in another study Tenpe and Patel (2020) found that SVM has better ability to predict the CBR value as compared to GEP. Additionally, the sensitivity analysis shows that CBR value is greatly influenced by G and S content. Using the regression analysis and ANN, Taha et al. (2019) developed the correlation equation for CBR of granular soil from the results of MDD and D_{60} of soil.

They observed that correlation obtained through ANN is of excellent accuracy and lower bias as compared to the regression analysis. A research conducted by Kurnaz and Kaya (2019) on 158 soil samples describe that GMDH models performs more successfully in predicting the CBR value as compared to ANN and MLR models. Based on 20 soil samples, Alam et al. (2020) reveals that both un-soaked and soaked CBR of soil can be predicted successfully through GEP, ANN and kriging methods. Recently, Bardhan, Samui, et al. (2021) attempted to predict the soaked CBR value of 312 soil datasets through particle swarm optimization (PSO) algorithm with adaptive and time-varying acceleration coefficients. The comparative analysis of various extreme learning machine (ELM) based adaptive neuro swarm intelligence (ANSI) such as ELM coupled-modified PSO (ELM-MPSO), ELM coupled-time varying acceleration coefficients PSO (ELM-TPSO) and ELM coupled-improved PSO (ELM-IPSO) reveals that the modified and improved version of PSO has high accuracy at early iterations than the standard PSO. In the another investigation, Bardhan, Gokceoglu, et al. (2021) observed that multivariate adaptive regression splines with piecewise linear (MARS-L) demonstrate the higher accuracy in predicting the soaked CBR as compared to MARS with piecewise-cubic (MARS-C), Gaussian process regression and genetic programming. Hassan et al. (2021) attempted to predict the CBR value of fine-grained plastic soil from their index properties and compaction parameters through multi linear regression analysis (MLRA). The study was conducted for the standard Proctor compactive energy level whereas the engineers always prefers the modified Proctor compactive energy level for constructing the highways and expressways.

Table 2.1 Summary of literature model for various type of soil.

S. no.	Agency/Authors (year)	Purposed equations/models	Prediction technique used	Types of soil (no. of samples tested)	Test Condition (Soaked/Un-soaked)	Description of used independent properties with their covered range/Limitation of the models	R/R^2	Eqn. no.
1.	Kleyn (1955)	$\log_{10} CBR = 0.29\,GM - 0.024\,PI + 1.23$	Statistically	Base course material (1134)	Soaked	NA	NA	(2.4)
2.	Agarwal and Ghanekar (1970)	$CBR = 21.2786 - 16.2921 \log OMC + 0.0696\,LL$	Statistically (Partial Regression)	Sand, Inorganic clay and silt, (48)	Soaked	LL = 18.5 - 69.9 PL = 11.6 - 33.7 PI = 1.3 - 44.7 OMC = 8.9 – 25	R= 0.58	(2.5)
		$CBR = 18.735 - 12.86 \log OMC + 0.052\,PL$					R= 0.51	(2.6)
		$CBR = 20.2809 - 14.3128 \log OMC + 0.0745\,PI$					R= 0.57	(2.7)
3.	National Cooperative Highway Research Program (2001)	$CBR = 28.09(D_{60})^{0.358}$	Statistically	Non-Plastic Coarse grained soils (7)	NA	NA	0.84	(2.8)
		$CBR = \frac{75}{1 + 0.728(wPI)}$		Plastic Fine grained soils (11)			0.67	(2.9)
4.	Kin (2006)	$CBR_{Top} = OMC(\frac{MDD}{19.2})^{20}$	Statistically	Fine-grained soil (57)	Soaked	NA	NA	(2.10)

5.	Taskiran (2010)	$$CBR = \left(\frac{\left({}^{MDD}/_{PI}\right)}{\left(\left((SC+FC)-(LL-31.99)\right)-\left((LL+11.99)+31.99\right)\right)}\right) + \left(\sqrt{\left(\left((LL-OMC)+G\right)-PI\right)*\left(\left({}^{PI}/_{(-24.55)}\right)+MDD\right)}\right) + \left(\left(MDD-\left(\frac{\left(MDD*(OMC-MDD)\right)}{\left((FC-PI)-11.30\right)}\right)\right)*MDD\right) + \left(\frac{\left(\left(\left((44.14-G)-FC\right)-(G+PI)\right)-\left({}^{G}/_{(18.91-PI)}\right)\right)}{18.91}\right) + \left(\sqrt{\left(\frac{FC}{\left(\left((FC+1.04)-(SC+PI)\right)*(-22.56+SC)\right)}\right)}+SC\right)$$	Computationally (ANN/GEP)	Fine-grained soil (151)	Soaked	NA	0.92	(2.11)
6.	Yildirim and Gunaydin (2011)	$$CBR = 0.2353\,G + 3.0798$$	Statistically (SRA, MRA) and Computationally (ANN)	Granular soil (124)	NA	G = 0 - 78 SC =0.90 - 49 FC = 10 - 99.10 OMC = 7.2 - 40.20 MDD = 1.21 - 2.18	0.86	(2.12)
		$$CBR = -0.1805\,FC + 18.508$$					0.80	(2.13)
		$$CBR = 0.22\,G + 0.045\,SC + 4.739\,MDD + 0.122\,OMC$$					0.88	(2.14)
7.	Alawi and Rajab (2013)	$$CBR = -112.4335 - 0.2856\,LA - 4.7280\,OMC + 98.4613\,MDD$$	Statistically (MLR)	Sub-base layer material (19)	NA	LA = 13.4 - 32 OMC = 6.2 - 8.1 MDD = 1.99 - 2.28	0.95	(2.15)
8.	Yared (2013)	$$CBR = 16.270 - 0.179\,LL$$	Statistically (SLR, MLR)	Sub-grade soil samples (42)	Soaked	LL = 42 - 72 PI = 12 - 52 MDD = 1.48 - 1.65	0.458	(2.16)
		$$CBR = -21.522 - 0.141\,LL + 0.137\,PI + 20.244\,MDD$$					0.629	(2.17)
9.	Deepak et al. (2014)	$$CBR = -3.06 + \frac{188.64}{LL} - \frac{24.15}{PL} + \frac{38.06}{OMC} + 0.225MDD + \frac{0.018}{FC}$$	Statistically	Fine-grained soil (81)	Soaked	LL = 25 - 73 PL = 17 - 46 OMC = 8 - 25 MDD = 1.54 - 1.98 FC = 76 - 97.5	0.87	(2.18)
10.	Bhatt et al. (2014)	$$CBR = -0.3776\,G - 0.4528\,SC - 0.4094\,FC + 0.3487\,OMC + 24.7518\,MDD$$	Statistically and ANN	Fine-grained and coarse-	Soaked	G = 2.75 - 31.14	0.88	(2.19)

				grained soil (124)		SC = 12.61 - 51.5 FC = 28.18 - 74.73 OMC = 10 - 21 MDD = 1.42 - 2.03		
11.	Erzin and Turkoz (2016)	$CBR = -140.132 - 0.160\ Q - 0.305\ Fel - 0.195\ Ca - 0.436\ C - 0.450\ A + 102.192\ MDD - 6.890\ G_s + 49.869\ C_c - 13.195\ C_u + 0.844\ OMC$	Statistically and ANN	Sand (61)	Un-soaked	Q = 0 - 100 Fel = 0 - 75 Ca = 0 - 10 C = 0 - 100 A = 0 - 100 G_s = 2.48 - 3.57 OMC = 2.57 - 14.90 MDD = 1.41 - 2.50 C_u = 1.58 - 5.91 C_c = 0.88 - 1.78	0.81	(2.20)
12.	Araujo and Ruiz (2016)	$CBR = 0.510\ LL + 0\ PL - 0.820\ PI - 2.917\ OMC - 17.991\ MDD + 0.681\ G + 0.205\ SC + 0.032\ FC + 64.890$	Statistically	Granular and fine-grained soil (75)	Soaked	G = 0 - 68 SC = 0 - 83 FC = 7 - 98 LL = 15 - 75 PL = 13 - 26 PI = 1 - 49 MDD = 1.62 - 2.29 OMC = 14.5 - 19.2	0.776	(2.21)
13.	A. u. Rehman et al. (2017)	$CBR = 6.508 D_{50} + 1.48 C_u + 3.970$	Statistically	Granular soil (70)	Soaked	D_{50} = 0.2 - 2.3 C_u = 1.7 - 9.7	0.85	(2.22)
14.	Z. Rehman et al. (2017)	$CBR = -0.10\ LL - 0.425\ PI + 15.73$	Statistically	Fine-grained soil (43)	Soaked	LL = 0 - 41 PI = 0 - 19 C_u = NA MDD (lb/ft^3) = 103 – 130	0.9	(2.23)
		$CBR = 0.7\ C_u + 8.5$		Coarse-grained soil (41)			0.8	(2.24)
		$CBR = 0.7\ C_u + 0.045\ MDD + 3.4$					0.8	(2.25)

15.	Tenpe and Patel (2018) and Tenpe and Patel (2020)	$CBR = \frac{G + SC}{OMC + PI} + G^{\frac{1}{3}} - MDD - 3.738$	GEP	Mixed soil samples (389)	Soaked	G = 0 - 63.68 SC = 0.14 – 97.23 PI = 0 - 24 MDD = 1.6 – 2.5 OMC = 7 – 25.3	R = 0.82	(2.26)
16.	Farias et al. (2018)	$CBR = 1.19 - 1.12\,FC - 7.79\,PI^2 - 6.82\,OMC$	Statistically and ANN	Granular and fine soil (96)	Soaked	FC = 0 - 98 LL = 15 - 51 PL = 12 - 35 PI = 7.3 - 12.3 OMC = 4.5 – 17	0.73	(2.27)
		$CBR = 0.23 - 0.20\,FC - 0.29\,LL + 0.40\,PL$					0.53	(2.28)
		$CBR = 1.20 - 1.12\,FC - 0.96\,LL + 1.22\,PL - 7.33\,OMC$					0.66	(2.29)
17.	Katte et al. (2019)	$CBR = -20.139 - 0.091\,PL - 0.055\,PI - 2.895\,OMC + 47.130\,MDD + 0.000\,FC - 0.668\,SC + 0.049\,G$	Statistically	NA (33)	Soaked	PL = 26.5 - 62.1 PI = 13.6 - 44.3 OMC = 9.6 - 16.5 MDD = 1.91 - 2.25 FC = 10.5 - 38.2 SC = 2.9 - 13.7 G = 47.2 - 86.5	0.84	(2.30)
18.	Bardhan, Gokceoglu, et al. (2021) and Bardhan, Samui, et al. (2021)	$CBR = 5(0.41189 - 1.9269BF_1 - 6.7653BF_2 + 1.7537BF_3 - 0.5018BF_4 - 0.41784BF_5 + 1.5985BF_6 - 0.66849BF_7 + 1.1551BF_8 + 0.21844BF_9 + 0.16481BF_{10} - 1.0543BF_{11} - 0.39357BF_{12} + 0.57399BF_{13} - 3.837BF_{14} + 1.066BF_{15} + 3.0431BF_{16} + 5.6426BF_{17} - 1.5195BF_{18}) + 5.2$	MARS-L	Mixed soil samples (312)	Soaked	G = 0 – 30 CS = 0 - 15 FS = 7 - 82 FC = 6 - 91 PI = 0 - 26 MDD = 1.77 – 2.05 OMC = 7.1 – 15.5	0.90	(2.31)
		$CBR = 5(0.41769 - 1.871BF_1 - 7.7205BF_2 + 1.7228BF_3 - 0.51647BF_4 - 0.45338BF_5 + 1.8154BF_6 - 0.84649BF_7 + 1.5757BF_8 + 0.22837BF_9 + 0.17982BF_{10} - 1.1155BF_{11} - 0.34312BF_{12} + 0.66219BF_{13} - 2.2008BF_{14} + 0.87241BF_{15} + 1.6047BF_{16} + 6.2335BF_{17} - 1.6493BF_{18}) + 5.2$	MARS-C				0.90	(2.32)
		$CBR = 5(0.1507FS + 0.07025\exp(\exp(\sin(MDD))) + 8.645\tan^{-1}(\sin(\cos(CS))) - 4.093\cos(CS) - 0.6447\cos(PI) - 0.7673PI \times OMC - 0.947) + 5.2$	GP				0.88	(2.33)

GM: grading module; C_u: coefficient of uniformity; C_c: coefficient of curvature; D_{50}: 50% particle finer; D_{60}: 60% particle finer; LL: liquid limit; PL: plastic limit; PI: plasticity index; wPI: weighted plasticity index; MDD: maximum dry density; OMC: optimum moisture content; G: gravel; SC: sand content; FC: fine content; FS: fine sand; CS: coarse sand; G_s: specific gravity; LA: los angeles; BF: basis functions (see Table 2.2); Q: quartz; Fel: feldspar; Ca: calcite; C: corund; A: amorphous.

Table 2.2 Basis function values for MARS-L and MARS-C model **(Bardhan, Gokceoglu, et al., 2021)**.

BFs	Expressions for MARS-L model	Expressions for MARS-C model
1	Max (0, MDD - 0.71429)	C (MDD| +1, 0.66071, 0.71429, 0.83929)
2	Max (0, 0.42529 – OMC)	C (OMC| -1, 0.36782, 0.42529, 0.44828)
3	Max (0, MDD – 0.60714)	C (MDD| +1, 0.44643, 0.60714, 0.66071)
4	Max (0, 0.60714 – MDD)	C (MDD| -1, 0.44643, 0.60714, 0.66071)
5	Max (0, 0.41176 – FC)	C (FC| -1, 0.24118, 0.41176, 0.48235)
6	Max (0, OMC – 0.47126)	C (OMC| +1, 0.44828, 0.47126, 0.48276)
7	Max (0, OMC – 0.65517)	C (OMC| +1, 0.5977, 0.65517, 0.82759)
8	Max (0, 0.31034 – OMC)	C (OMC| -1, 0.15517, 0.31034, 0.36782)
9	Max (0, CS – 0.13333)	C (CS| +1, 0.066667, 0.13333, 0.56667)
10	Max (0, FS – 0.37333)	C (FS| +1, 0.18667, 0.37333, 0.68667)
11	Max (0, 0.14286 – MDD)	C (MDD| -1, 0.071429, 0.14286, 0.21429)
12	Max (0, PI – 0.69231)	C (PI| +1, 0.34615, 0.69231, 0.84615)
13	Max (0, 0.28571 – MDD)	C (MDD| -1, 0.21429, 0.28571, 0.44643)
14	Max (0, 0.63529 – FC)	C (FC| -1, 0.59412, 0.63529, 0.64706)
15	Max (0, 0.55294 – FC)	C (FC| -1, 0.48235, 0.55294, 0.59412)
16	Max (0, 0.65882 – FC)	C (FC| -1, 0.64706, 0.65882, 0.82941)
17	Max (0, 0.49425 – OMC)	C (OMC| -1, 0.48276, 0.49425, 0.51724)
18	Max (0, OMC – 0.54023)	C (OMC| +1, 0.51724, 0.54023, 0.5977)

2.3 LITERATURE REVIEW ON MACHINE LEARNING (ML) ALGORITHMS

2.3.1 ML Algorithms

The term 'machine learning' is a subfield/type of Artificial Intelligence (AI) and is referred to as predictive analytics or predictive modelling. ML is the development of computer systems that can learn and adapt with or without following explicit instructions by using algorithms and statistical models to analyze and draw inferences from patterns in data. There are four types of ML algorithms which are listed below and the further classification of these methods is depicted in Figure 2.5.

1. Supervised learning: In supervised learning, the machine is trained through an example. The operator provides the dataset that includes desired inputs and outputs, and the corresponding ML algorithm must find a method to determine how to reach those inputs and outputs. The operator knows the correct answers to that particular problem, the algorithm identifies patterns in data, learns from observations and makes predictions. The algorithm makes predictions and is corrected by the operator – and this process continues until the algorithm achieves a high level of accuracy/performance.
    1.1 Classification: In classification tasks, the ML program must draw a conclusion from observed values and determine what category new observations belong.
    1.2 Regression: In regression tasks, the machine learning program must estimate – and understand – the relationships among variables. Regression analysis focuses on one dependent variable and a series of other independent variables – making it particularly useful for prediction and forecasting.

2. Semi-supervised learning: This is similar to supervised learning, but instead uses both labelled and unlabeled data. Labelled data is essential information that has meaningful tags so that the algorithm can understand the data, whilst unlabeled data lacks that information. By using this combination, machine learning algorithms can learn to label unlabeled data.
3. Unsupervised learning: Here, the machine learning algorithm studies data to identify patterns. There is no answer key or human operator to provide instruction. Instead, the machine determines the correlations and relationships by analyzing available data. In an unsupervised learning process, the machine learning algorithm is left to interpret large data sets and address that data accordingly. The algorithm tries to organize that data in some way to describe its structure. This might mean grouping the data into clusters or arranging it to look more organized.
    3.1 Clustering: Clustering involves grouping sets of similar data (based on pre-defined criteria). It's useful for segmenting data into several groups and analyzing each data set to find patterns.
    3.2 Dimensionality reduction: Dimension reduction reduces the number of variables being considered to find the exact information required.
4. Reinforcement learning: Reinforcement learning focuses on regimented learning processes, where a machine learning algorithm is provided with a set of actions, parameters and end values. By defining the rules, the machine learning algorithm then explores different options and possibilities, monitoring and evaluating each result to determine which one is optimal. Reinforcement learning teaches the machine by trial and error. It learns from past experiences and begins to adapt its approach in response to the situation to achieve the best possible result.

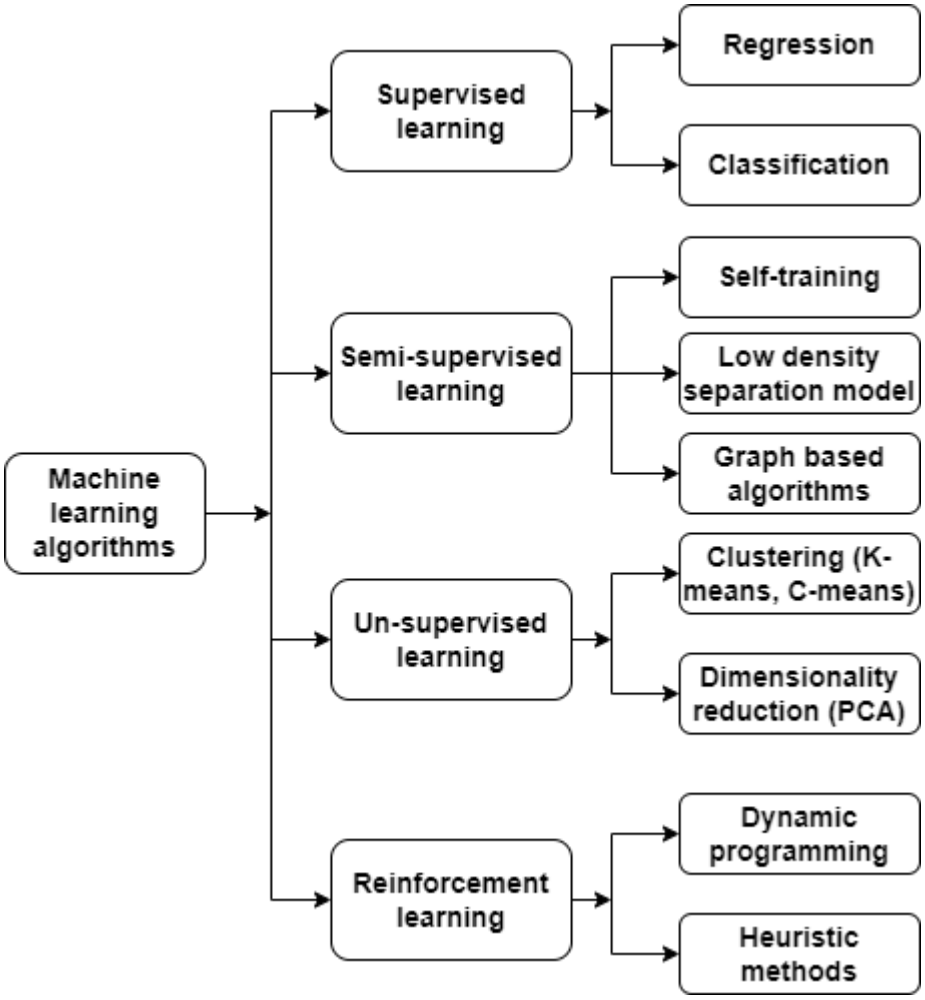


Figure 2.5 Classifications of machine learning algorithms

2.3.2 Application of ML Algorithms in Geotechnical Engineering

Over the last few decades, ML has become more popular for solving the most complex problems of geotechnical as well as highway engineering due to its superior predictive ability as compared to conventional statistical methods. Numerous techniques; like Artificial neural networks (ANNs), Particle swarm optimization (PSO), Ant colony optimization (ACO), Genetic programming (GP), Support vector machine learning (SVM), Artificial bee colony (ABC) algorithm, decision-tree-based algorithms, etc. are available which comes under the ML umbrella. The present study focused especially on multi-expression programming (MEP), a variant of genetic programming (GP) and

extreme gradient boosting (XGBoost), which belongs to a boosting class of ensemble learning algorithm techniques as they have also been proved to be the most successful in the field of geotechnical engineering by several researchers. Genetic programming (Alam et al., 2020; Alavi et al., 2013; Alavi et al., 2010; Ardakani and Kordnaeij, 2017; Armaghani et al., 2018; Bardhan, Gokceoglu, et al., 2021; Bardhan, Samui, et al., 2021; Baykasoğlu et al., 2009; F.E. Jalal et al., 2021; Moayed et al., 2017; Pattanaik et al., 2020; Sivrikaya et al., 2013; Taskiran, 2010; Tenpe and Patel, 2018, 2020; Tsai and Lin, 2011; Wang and Yin, 2020; Yang et al., 2012; Yildirim and Gunaydin, 2011) is most commonly used while XGBoost (Ali et al., 2021; Cao et al., 2021; Dong et al., 2020; Duan et al., 2021; X. Zhang et al., 2020) has recently entered in solving the complex civil engineering problems therefore more attention is required.

2.3.2.1 Genetic programming

Genetic Programming (GP) is an extension of genetic algorithms (GA), which belongs to a class of evolutionary algorithms used for breeding a population of computer programs (Koza, 1992) and is inspired by Darwin's evolution theory of survival of the fittest. GA was first introduced by Holland (1975) and further developed by Goldenberg (1989), while GP was invented by Cramer (1985) and later acquired by Koza (1992). The main difference between GA and GP is that GA is represented as a list of actions and values, often a fixed-length binary string. In contrast, GP generates a tree-structured representation for a set of input variables and corresponding outputs. Intuitively GP is a subset of GA where the solutions are computer program rather than fixed-length binary strings. It is quite interesting to consider that GP is more advantageous over the GA as of having an ability to evolve any program by using various number of functional sets (+, _, *, /, log, pow, trigonometry, etc.). GP optimizes and manipulates a population of computer models (or programs) that have been proposed to solve a particular problem,

so that the model that best fits the problem is obtained. Figure 2.6 depicts the general flow chart for the GP algorithm.

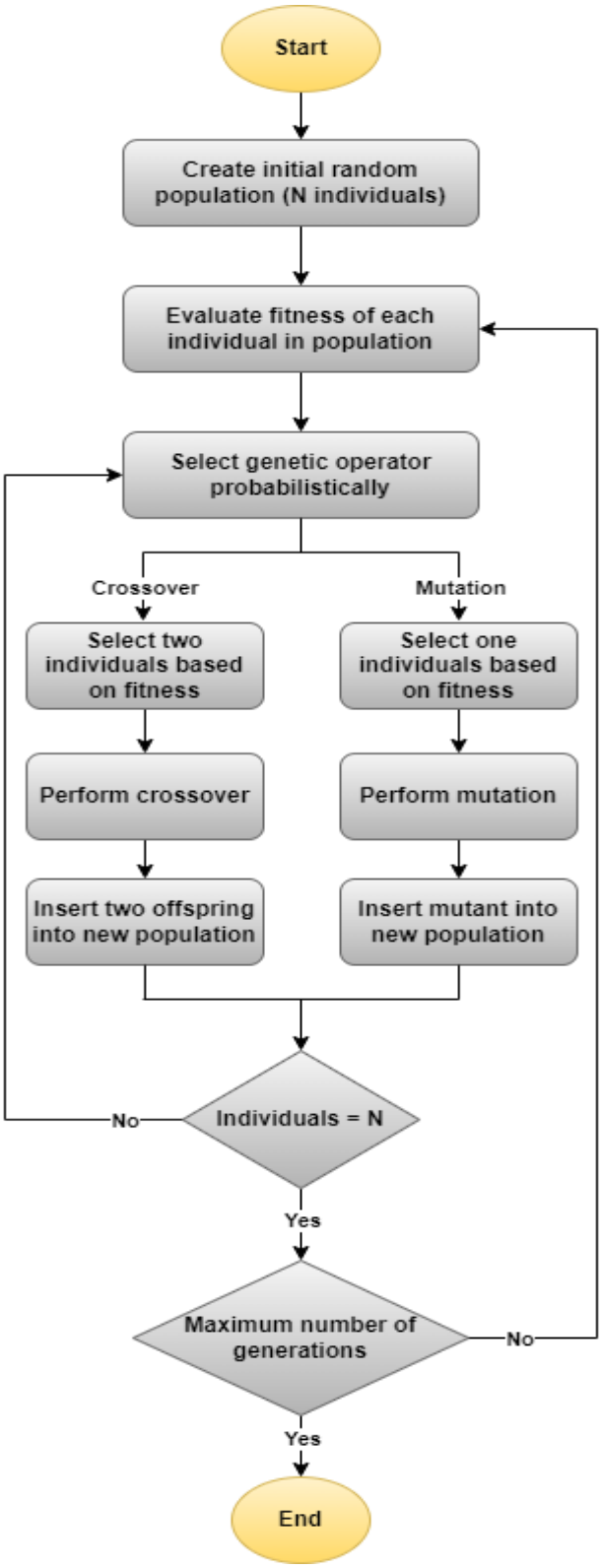


Figure 2.6 Genetic programming algorithm's flowchart

2.3.2.1.1 Model development

Like genetic algorithms (GA), here we also had an initial population which is produced by several individuals or chromosomes and then using the pre-defined fitness function, an error value is obtained for all the chromosomes. In GP, chromosomes are computer programs that are composed of two sets (i.e., functional sets and terminal sets) which are defined by the users for the suitability of a certain problem. The functional set consists of mathematical operators (×, /, −, +), trigonometric functions (sin, cos, tan, etc.), Boolean logic functions (AND, OR and NOT) as well as any other user defined functions. The terminal set may contain logical constants, numerical constants and variables. The performance of genetic programming is most significantly influenced by the functional and terminal set however, there are no systematic criteria for selecting the functional and terminal sets. The selection of functional and terminal sets is performed randomly on a trial and error basis and arranged in a tree-like structure to form a computer model which contains root nodes, and branches of functional and terminal nodes (as shown in Figure 2.7).

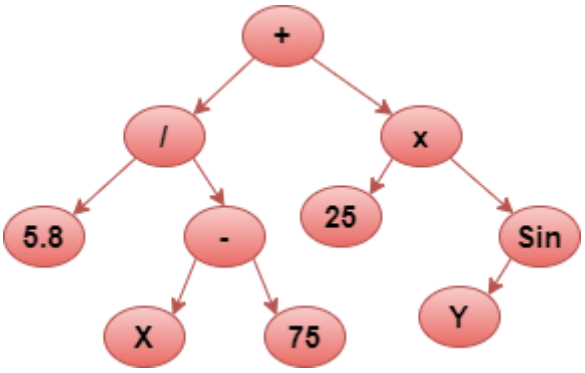


Figure 2.7 Genetic programming tree representation of function $\left(\frac{5.8}{(X-75)} + 25 \sin Y\right)$

Formerly, the population of the computer model is generated, each model is processed for execution using the existing data for a given problem and the fitness of the model is estimated in terms of its ability to solve the problems. For most of the problems,

the error between the estimated and actual values is used to examine model fitness. Thereafter, the existing population is substituted by breeding the new population for a computer model and this is performed by applying three core operations: reproduction, crossover and mutation (Das, 2005; Rakesh et al., 2006; M Amaranatha Reddy et al., 2004). These three operations are applied to the existing population at certain proportions and the selection of the model is done as per their fitness value. The detailed information about these three operations is discussed below:

2.3.2.1.1.1 Reproduction

Reproduction is the process of copying a computer model from an existing population into a new population without alteration. It is the stage of the genetic algorithm in which individual genomes are chosen from the string of chromosomes. The solutions are evaluated based on the fitness of the function and less than the average population is replaced by the above-average population to keep the population size constant. Therefore, the strings with high fitness enter the mating pool and the remaining ones die off. There are three different commonly used techniques for selecting chromosomes: Roulette wheel, Ranking selection, and Steady-State selection.

Roulette wheel

In this method, the parents are selected according to their fitness. Better chromosomes, are having more chances to be selected as parents. The methodology is described by selecting an example of the Roulette wheel (as shown in Figure 2.8) where each of the chromosomes is assigned a section in the Roulette wheel and the size of section is proportional to the fitness value of the respective chromosomes. This means that higher the fitness value of the chromosome, the larger the size of the section. It is the most common method for implementing fitness proportionate selection. The process of the Roulette wheel algorithm is described below:

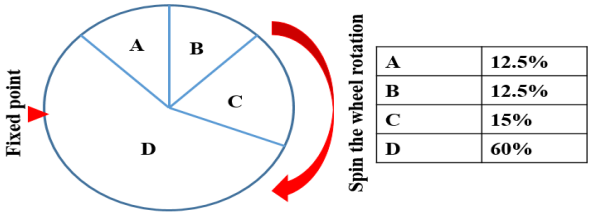
Step 1: Calculate the sum of all chromosomes fitness in the population; sum = *S*.

Step 2: Generate the random number *r* from the given population interval (*0*, *S*).

Step 3: Go through the entire population and sum the fitness from *0* to sum S_i. When this sum (S_i) is more than a fitness criteria value (*r*), stop and return to the i^{th} chromosome.

Step 4: Repeat steps 2 and 3.

Obviously, step 1 is performed only once for each population.



A	12.5%
B	12.5%
C	15%
D	60%

Figure 2.8 Roulette wheel showing the fitness section for four different solutions

Ranking selection

The applicability of Roulette wheel methods becomes unsatisfactory in the genetic algorithm when there is a huge difference between the fitness values of chromosomes. For instance, if the best chromosomes have the fitness of 85% of the sum of all fitness, then the selection of other chromosomes is negligible. On the other hand, the Rank selection ranks the population first depending upon their respective fitness then every chromosome receives the revised fitness value determined by this ranking. The worst will have fitness 1, and the second-worst will have fitness 2. The best one will have fitness *N* (number of chromosomes in the population).

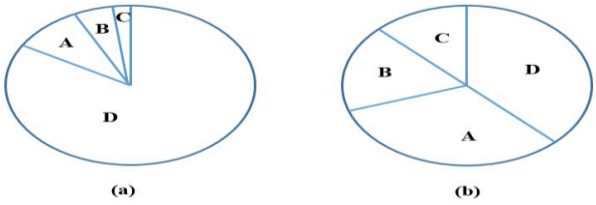


Figure 2.9 Roulette wheel showing the fitness section for before ranking (a) and after ranking (b)

Figure 2.9 presents the ranking selection procedure in which the initial fitness of solutions are 80, 12, 6 and 2, respectively. Therefore, the rank assigned to the chromosomes are 4, 3, 2, and 1, respectively. Hence, the average ranking value is 2.5 and the revised fitness of chromosomes are 1.6, 1.2, 0.8 and 0.4 (which is obtained by dividing the rank by average value) corresponding to 80, 12, 6 and 2, respectively. Now all the chromosomes have a chance to be selected. However, this method is a slower convergence technique as the best chromosomes don't differ too much from others.

Steady-state selection

This method replaces a few individuals in each generation and is not a particular method for selecting the parents. Only a small number of newly created offspring are put in place of the least fit individual. The main idea of steady-state selection is that a bigger part of the chromosome should retain a successive population.

2.3.2.1.1.2 Crossover

The formation of a new population is performed through crossover and mutation. The crossover is like creating lots of search directions and reproduction is to find out the best direction. Crossover is the swapping of randomly chosen parts of two selected computer models. Generally, two parent chromosomes (solutions) are needed to perform a crossover operation. In this process, genes are selected from the parent chromosomes and new offspring (children) are created. Therefore, parents might be considered as the current search direction, whereas offspring is the restructured search direction. Crossover can be performed with binary encoding, permutation encoding, value encoding and tree encoding.

Binary encoding crossover

In the process of binary encoding, the chromosomes may crossover at a single point, two points, uniformly or arithmetically. In a single point crossover operation, a single crossover point is chosen randomly and the data before this point is exactly copied from the first parent and the data after this point are exactly copied from the second parent to create new offspring. Hence, two offspring are obtained from two parents. Similarly, in two-point crossover, two crossover points are selected randomly and the data in between these two points are exchanged to obtain new offspring. In the uniform crossover, typically, data from both parents are randomly copied with equal probability. Other mixing ratios are also sometimes used; resulting offspring gets more genetic information from one parent than the other. In the arithmetic crossover, a crossover of chromosomes are performed by using OR and AND operators to create new offspring. Figure 2.10 shows the various crossover techniques used in genetic algorithms.

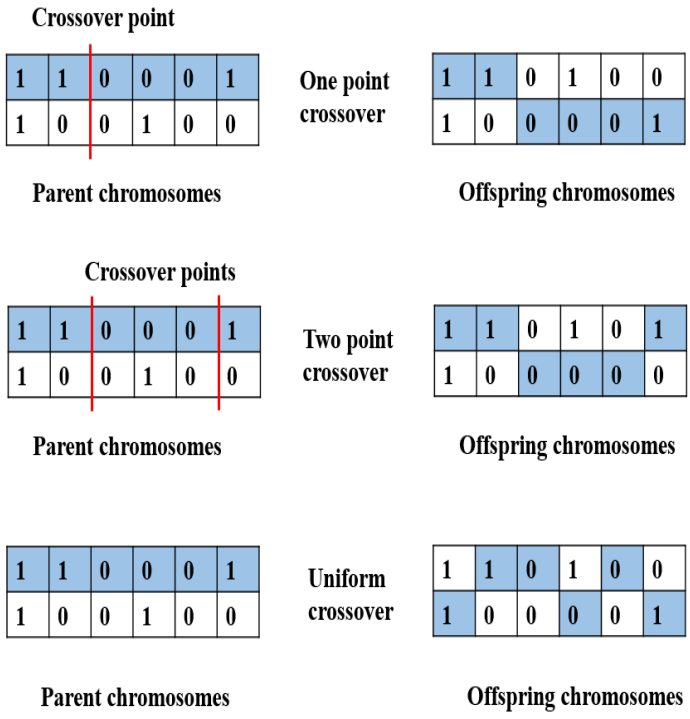


Figure 2.10 Crossover techniques in genetic algorithm

Permutation, value and tree encoding crossover

In this encoding system one crossover point is selected. The permutation is copied from first parent chromosome till the point of crossover and the other parent chromosome is exactly copied to ensure that no number is left to be put in the offspring. Further, if any number is not present in the offspring then it is added to the offspring chromosome.

Value encoding crossover

Like binary encoding system it can be performed at single point, two point, uniform and arithmetic representation.

Tree encoding crossover

In this type of crossover, one point of crossover is selected in both parent tree chromosomes, which are divided at a point. The parts of tree below crossover point are exactly exchanged to produce new offspring, as illustrated in Figure 2.11.

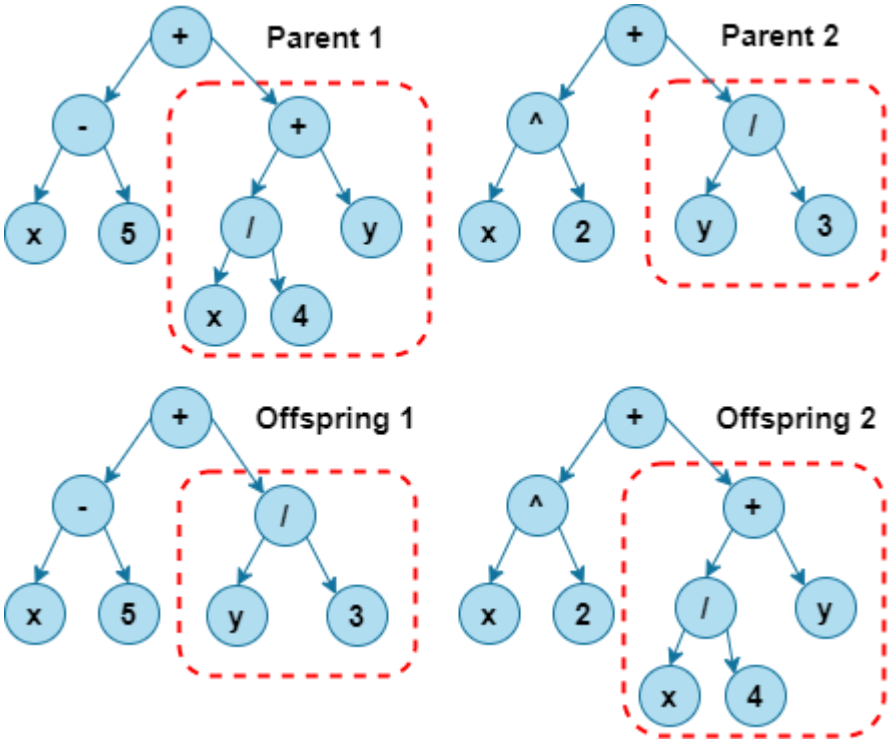


Figure 2.11 Tree encoding crossover

2.3.2.1.1.3 Mutation

Mutation is the processes of replacing a randomly selected functional or terminal node with another node from the same function or terminal set, a functional node replaces a functional node and a terminal node replaces a terminal node. In the processes of mutation, a new individual is created by doing some modification to a selected individual. Crossover operation can't generate different offspring from their parents because the acquired information is used to crossover the chromosome. There is no much difference in between the crossover and mutation operation. The difference is that mutation can operate on the same parents. Like crossover, mutation can also be performed for all types of encoding techniques (binary, permutation, value and tree).

The evolutionary processes for evaluating the fitness of existing population using these three operations (reproduction, crossover, mutation) and producing new populations is continued until a termination criterion is met, which can either be a certain number of generations or error term or any other user defined criteria. The main advantage of genetic programming over the traditional methods and other soft computing techniques is the ability of generating the prediction equations without prior form of existing relationship. The developed equations can easily be manipulated in the practical circumstances. Genetic programming can be classified into three categories: tree-based GP, linear-based GP and graph-based GP. In comparison to the other two types of GP, the linear-based is more efficient as it requires slow interpreters.

In general, GP can be classified into three types i.e., linear-based GP, graph-based GP and tree-based GP (Alavi and Gandomi, 2011). Several linear variants of GP have been proposed, some of them are, grammatical evolution (GE) (Ryan et al., 1998; Ryan and O'Neill, 1998), linear genetic programming (LGP) (Brameier and Banzhaf, 2001, 2002), Cartesian genetic programming (CGP) (Miller J, 2002), gene expression

programming (GEP) (Ferreira, 2001), genetic algorithm for deriving software (GADS) (Patterson, 2002) and multi expression programming (MEP) (M Oltean and D Dumitrescu, 2002; Mihai Oltean and Groşan, 2003). All these GP variants make a clear distinction between the genotype and phenotype of the individuals (Banzhaf, 1994). Thus, the individuals are represented as linear entities (strings) that are decoded and expressed like nonlinear entities (trees) (Mihai Oltean and Grosan, 2003). A comparative analysis of GE, GEP, LGP and MEP variants alongwith their strength and weakness shows that MEP overall represents the best performance (Mihai Oltean and Grosan, 2003), however, this may vary from problem to problem. In consideration of the accuracy and efficiency, the present study adopted multi expression programming (MEP) to solve the particular problem.

2.3.2.1.2 Multi-expression programming (MEP)

2.3.2.1.2.1 MEP algorithm

As discussed earlier that MEP is subset of GP which was introduced by M Oltean and D Dumitrescu (2002). MEP uses linear chromosomes for encoding the solution and having an ability to encode multiple solutions (computer programs) of a problem through a single chromosome. As per the fitness value of individuals, the best encoded solution is selected to represents the chromosomes. The evolutionary steady-state MEP algorithm starts by the formation of a random population of individuals. So as to evolve the best expression from the available data file of inputs and outputs along a specified number of generations, MEP uses the following steps until a termination condition is reached (Mihai Oltean and Groşan, 2003):

1. Selects the two parents using binary tournament procedure (Koza, 1992) and recombined them with a fixed crossover probability.
2. Two offspring are obtained by recombination of two parents.

The offspring are mutated and the best of them is selected and that replaces the worst individual presents in the existing population (if the offspring is better than the worst individual in the existing population).

2.3.2.1.2.2 MEP representation

MEP is represented similar to the way in which C and Pascal compilers translate mathematical expressions into machine code (Alavi et al., 2013; Alavi et al., 2010). The MEP genes are represented by substrings of variable length. The number of genes in a chromosome is constant and it represents the chromosome length. Each gene encodes a terminal (an element in the terminal set *T*) or a function symbol (an element in the function set *F*). A gene that encodes a function includes pointers towards the function arguments. Function parameters always have indices of lower values than the position of that function itself in the chromosome. As per the proposed representation scheme, the first symbol in a chromosome must be a terminal symbol which is randomly chosen from the terminal set. In this way only syntactically correct programs are obtained (Mihai Oltean and Groşan, 2003). An example of MEP chromosome is given below:

0: a

1: b

2: pow 0, 1

3: c

4: + 2, 3

5: d

6: / 4, 5

The terminal set for the above example is a, b, c and d whereas “pow”, “+” and “/” are the function set. The translation of MEP individuals into the computer programs can be achieved by reading the chromosome in a top-down fashion starting with the first

position. A terminal symbol specifies a simple expression. A function symbol specifies a complex expression (formed by linking the operands specified by the argument positions with the current function symbol). In the above example, genes 0, 1, 3 and 5 encode simple expressions formed by single terminal symbol. These expressions are

$E_0 = a$

$E_1 = b$

$E_3 = c$

$E_5 = d$

Gene 2 indicates the operation *pow* on the operands located at position 0 and 1 of the chromosome. Hence, gene 2 encodes the expression:

$E_2 = a^b$.

Gene 4 indicates the operation + on the operands located at position 2 and 3 of the chromosome. Hence, gene 4 encodes the expression:

$E_4 = a^b + c$.

Gene 6 indicates the operation / on the operands located at position 4 and 5 of the chromosome. Hence, gene 6 encodes the expression:

$E_6 = (a^b + c)/d$.

Each of the above expression can be considered as a possible solution as well as represented as a forest of genes trees (see Figure 2.12). We have to choose one of these expressions (E_0, ..., E_6) to represent the chromosome. There is neither practical nor theoretical evidence that one of them is best than the others. Thus we choose to encode multiple solutions in a single chromosome. Each MEP chromosome encodes a number of expressions equal to the chromosome length (the number of genes). The expression associated to each chromosome position is obtained by reading the chromosome bottom-up from the current position, by following the links provided by the functions pointers.

The fitness of each expression encoded in a MEP chromosome is computed in a conventional manner (the fitness depends on the problem being solved). The best expression encoded in a MEP chromosome is chosen to represent the chromosome (the fitness of a MEP individual equals the fitness of the best expression encoded in that chromosome).

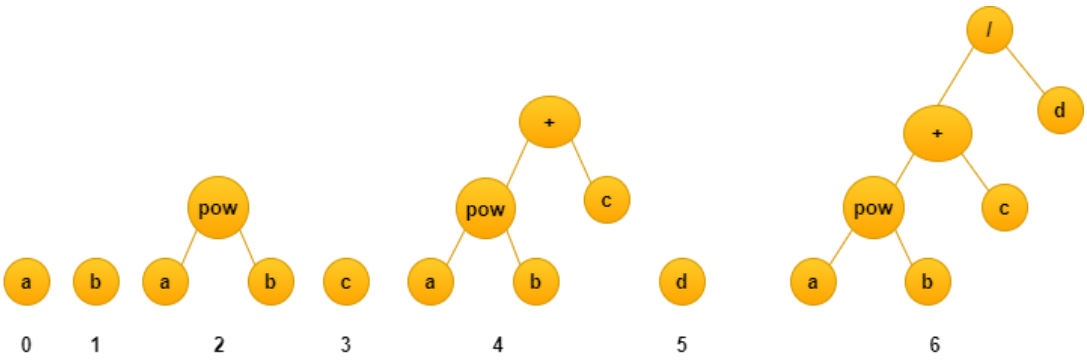


Figure 2.12 Encoded expression of MEP chromosome represented as a tree structure

2.3.2.2 Ensemble ML algorithm

Ensemble methods is a machine learning technique that combines various base models and transform them into a strong optimal predictive model. A single decision tree (discussed in detail in section 2.3.2.2.1) or model will rarely generalize well to data it wasn't trained for. Therefore, we can combine the predictions of a large number of decision trees to make our predictive model more accurate. Mathematically, a decision tree has low bias and high variance. Averaging the result of many decision trees reduces the variance while maintaining that low bias. Combining all the trees is known as an 'ensemble method'. There are different type of ensemble methods which are depicted in Figure 2.13.

1. Bagging: Bagging is the combination of the short form of **b**ootstrapping and **agg**regation. Initially multiple bootstrapped subsamples are obtained from complete dataset. A decision tree is formed for each of the bootstrapped

subsamples. Once decision tree is made for each of the subsample, an algorithm is used to aggregate over the decision trees to form the most efficient predictor. Random Forest (RF) models can be thought of as bagging, with a slight tweak. RF models decide where to split according to the level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over and producing a more accurate predictor.

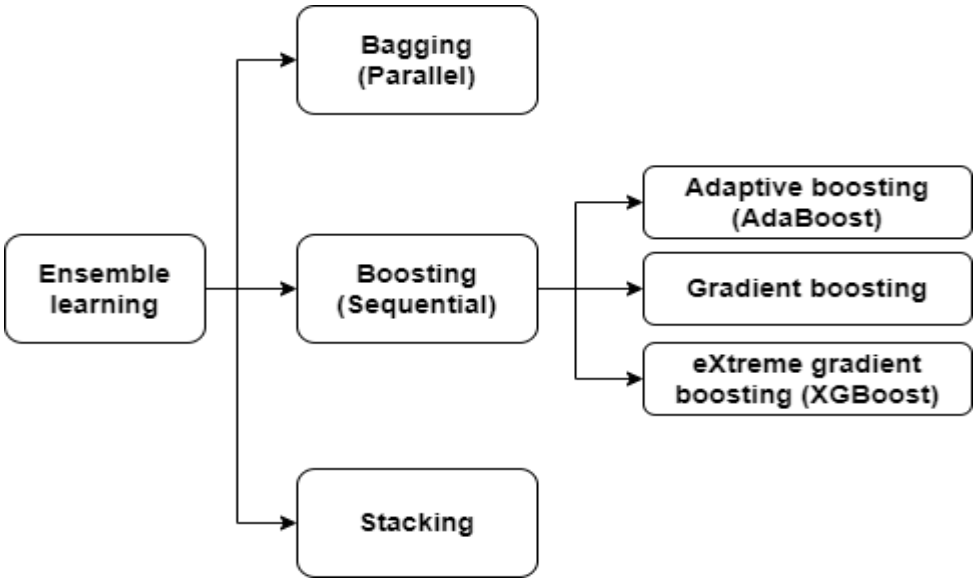


Figure 2.13 Different types of ensemble methods

2. Boosting: Boosting is an ensemble technique that learns from previous predictor mistakes to make better predictions in the future. The technique combines several weak base learners to form one strong learner, thus significantly improving the predictability of models. Boosting works by arranging weak learners in a sequence, such that weak learners learn from the next learner in the sequence to create better predictive models. Boosting further categorized into three different algorithms which are shown in Figure 2.13.

3. Stacking: Stacking, another ensemble method, is often referred to as stacked generalization. This technique works by allowing a training algorithm to ensemble several other similar learning algorithm predictions. Unlike bagging, in stacking, the models are typically different (e.g. not all decision trees) and fit on the same dataset (e.g. instead of samples of the training dataset). Unlike boosting, in stacking, a single model is used to learn how to best combine the predictions from the contributing models (e.g. instead of a sequence of models that correct the predictions of prior models).

2.3.2.2.1 Decision trees in ML

Decision trees are very specific type of probability trees that enables you to make a decision about some kind of process. Decision trees belongs to the family of supervised learning algorithms (having a predefined target variable) that can be used for solving both regression and classification problems. The name itself suggests that we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input parameters. Initially, we consider the whole training data as root node and compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node and finally ends with a decision made by leaves. Based on the type of target variable we have two types of decision tree:

1. Classification tree: Decision tree which has a categorical or class type target variable then it called as classification decision tree.
2. Regression tree: Decision tree which has a real number type target variable then it is called as classification decision tree.

The term classification and regression tree (CART) analysis is an umbrella term used to refer to either of the above procedures which was first introduced by Breiman et al.

(2017). Trees used for regression and trees used for classification have some similarities as well as some dissimilarities. The difference is how the impurity is calculated and how the output is predicted.

Some significant terminology related to decision-tree

1. Root node: It represents the entire population or sample and this further gets divided into two or more homogeneous sets (see Figure 2.14).
2. Splitting: It is a process of dividing a node into two or more sub-nodes.
3. Decision node: When a sub-node splits into further sub-nodes, then it is called the decision node.
4. Terminal/ Leaf node: Nodes which is not further processed for splitting is called terminal or leaf node.
5. Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. Sub-tree/ Branch: A subsection of the entire tree is called branch or sub-tree.
7. Parent and child node: A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node. In Figure 2.14, A is parent node and A_1 and A_2 are child node.

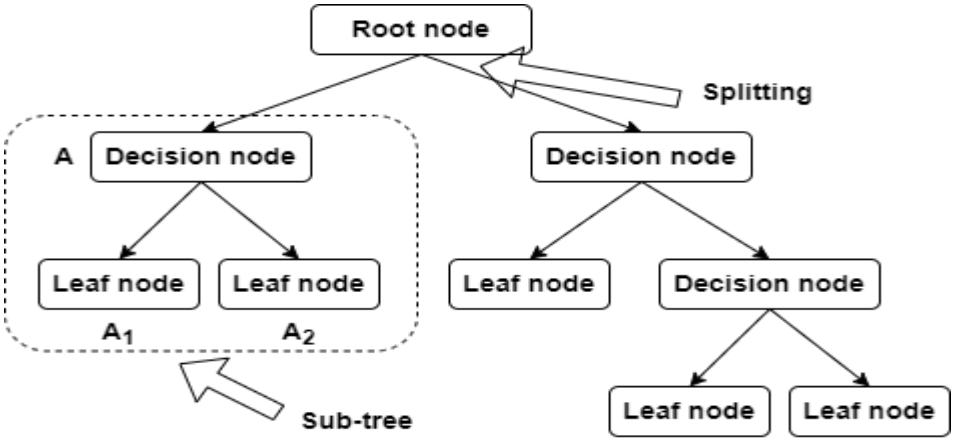


Figure 2.14 Decision-tree structure

2.3.2.2.2 eXtreme gradient boosting (XGBoost) algorithm

Boosting builds models from the individual weak learners in an iterative way. XGBoost is a decision tree-based ensemble ML algorithm developed by Chen et al. (2015). It makes use of gradient boosting. The algorithm can deal with both classification and regression problems. Being an effective tree-based ensemble learning algorithm, it is considered a powerful tool among data science researchers. Initially, XGBoost is based on gradient boosting architecture (Friedman, 2001), which uses various complement functions to estimate the results using eq. (2.34).

$$\overline{y_i} = y_i^0 + \eta \sum_{j=1}^{n} f_j(S_i) \tag{2.34}$$

Where, $\overline{y_i}$ denotes the predicted output for the i[th] data with the parameter vector S_i; n denotes the number of estimators corresponding to independent tree structures for each f_j; y_i^0 is the primary hypothesis i.e. mean of the original parameters in the training dataset; η is the learning rate.

According to eq. (2.34) in the j[th] stage, the j[th] estimator is connected to the model and the prediction of the j[th] y_i^{-j} is calculated from the estimated output $y_i^{-(j-1)}$ in the next step, and the established f_j of the j[th] complementary estimator is shown in (2.35).

$$y_i^{-j} = y_i^{-(j-1)} + \eta f_j \tag{2.35}$$

where, f_k represents the leaves weight that is established by reducing the objective function of the j[th] tree and is given by eq. (2.36).

$$f_{obj.} = \gamma Z + \sum_{k=1}^{m} \left[g_k \omega_k + \frac{1}{2}(h_k + \lambda)\omega_k^2 \right] \tag{2.36}$$

where, Z denotes the leaf nodes quantity, λ denotes constant coefficient, γ indicates the complexity parameter, ω_k^2 indicates the leaf weight from 1 to Z, g_k and h_k are the summation parameters for the entire dataset associated with k leaf of the initial and previous loss function gradient, respectively.

In order to build the j^{th} tree, a leaf is distributed into several leaves. Such a system is implied by using the gain parameters which is expressed through eq. (2.37).

$$G = \frac{1}{2}\left[\frac{X_L^2}{Y_L + \lambda} + \frac{X_R^2}{Y_R + \lambda} + \frac{(X_L + X_R)^2}{Y_L + Y_R + \lambda}\right] \tag{2.37}$$

In ML, a model is represented by its parameters which are known as hyper-parameters. The hyper-parameters are used to control the learning process of that particular algorithm. The prefix 'hyper_' suggests that they are 'top-level' parameters that control the learning process and the model parameters that result from it. The process of training a model involves choosing the optimal hyper-parameters that the learning algorithm will use to learn the optimal parameters that correctly map the input features (independent variables) to the labels or targets (dependent variable) such that you achieve some form of intelligence. XGBoost algorithm also have different hyper-parameters which are categorized into three different class as listed below:

1. General parameters
2. Booster parameters
3. Learning task parameters

2.3.2.2.2.1 General parameters

Booster type: Usually this is either a tree or a linear function. In the case of trees, the model will consist of an ensemble of trees. For the linear booster, it will be a weighted sum of linear functions.

N jobs: It allows us to specify the number of parallel jobs to run. If 1 is given, no CPU are used in parallelism, which is useful for debugging. If set to -1, all CPUs are used.

Validate parameters: When set to True, XGBoost will perform validation of input parameters to check whether a parameter is used or not. The feature is still experimental. It's expected to have some false positives.

2.3.2.2.2.2 Tree booster parameters

Colsample by level: Denotes the subsample ratio of columns for each level. Subsampling occurs once for every new depth level reached in a tree. Columns are subsampled from the set of columns chosen for the current tree.

Colsample by node: It is the subsample ratio of columns for each node (split). Subsampling occurs once every time a new split is evaluated. Columns are subsampled from the set of columns chosen for the current level.

Colsample by tree: It is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.

Gamma: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.

Learning rate: Step size shrinkage used in an update to prevent the overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative. The range is 0 to 1. Low learning rate value means the model is more robust to overfitting.

Maximum delta step: In maximum delta step we allow each tree's weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed,

but it might help in logistic regression when class is extremely imbalanced. Set it to value of 1-10 might help control the update. It ranges from 0 to ∞.

Maximum depth: Maximum depth of a tree. It ranges from 0 to ∞. Increasing this value will make the model more complex and more likely to over-fit.

Minimum child weight: It is the minimum sum of weights of all the observations required in a child. This parameter is generally used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.

N estimators: The maximum number of terminal nodes or leaves in a tree. It can be defined in place of max depth. Since binary trees are created, a depth of 'n' would produce a maximum of 2^n leaves.

Alpha: L1 regularization term on weight (analogous to Lasso regression). It can be used in case of very high dimensionality so that the algorithm runs faster when implemented. Increasing this value will make model more conservative.

Lambda: L2 regularization term on weights (analogous to Ridge regression). It is used to handle the regularization part of XGBoost. It should be explored to reduce overfitting.

Scale position weight: Control the balance of positive and negative weights, useful for unbalanced classes. A value greater than 0 should be used in case of high class imbalance as it helps in faster convergence.

Subsample: Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees and this will prevent overfitting. Subsampling will occur once in every boosting iteration. It ranges from 0 to 1.

2.3.2.2.2.3 Learning task parameters

Base score: The initial prediction score of all instances, global bias.

Objectives: This defines the loss function to be minimized. Mostly used values are: binary logistic, multi softmax, multi soft probability.

Evaluation metric: Evaluation metrics for validation data, a default metric will be assigned according to objective (RMSE for regression, and log-loss for classification, mean average precision for ranking).

Seed: The random number seed. It can be used for generating reproducible results and also for parameter tuning.

2.4 LITERATURE ON DATA ANALYSIS AND DATA DIVISIONAL APPROACHES

2.4.1 Data Analysis

Data analysis is the process of collecting and organizing the dataset with a goal to conclude some significant information and pattern from the dataset to develop an efficient predictive model. Dataset is generally categorized into quantitative and qualitative analysis. The quantitative analysis is defined as the value of dataset in the form of mathematical numbers where each observations has unique numerical values associated with it. While, qualitative analysis, also known as categorical analysis, is non-numerical in nature where the dataset is arranged categorically based on their properties and attributes. In general, the analysis of datasets can be performed through four different ways:

1. Descriptive analysis: Descriptive data analysis looks at past data and tells what happened. This is often used when tracking Key Performance Indicators (KPIs).
2. Diagnostic analysis: Diagnostic analysis takes the insights found from descriptive analytics and drills down to find the causes of those outcomes. Once your descriptive analysis shows that something negative or positive happened, diagnostic analysis can be done to figure out the reason.
3. Predictive analysis: Predictive data analysis predicts what is likely to happen in the future. In this type of research, trends are derived from past data which are then used to form predictions about the future.
4. Prescriptive analysis: Prescriptive data analysis combines the information found from the previous 3 types of data analysis and forms a plan of action for the organization to face the issue or decision. This is where the data-driven choices are made.

2.4.1.1 Statistical analysis of the dataset

The statistical analysis is one of the most significant and probably be the very first stage for identifying the behavior and pattern of dataset. The statistical analysis of dataset including the construction of tables, graphical displays, and basic statistical computations that facilitates the ways to summarize and collecting information into a set of descriptive measures and visual devices which assist in understanding the complex dataset. The descriptive and graphical statistical analysis of dataset can be performed through either of the software packages environment like: Microsoft Excel, Statistical Analysis System (SAS), MINITAB, STATISTICA and Statistical Package for Social Sciences (SPSS) etc.

2.4.1.1.1 Frequency distribution/ Histogram plot

Frequency distribution is the measurement of number of occurrence of an observation in the dataset. In statistic, the frequency distribution for the dataset can be presented graphically (known as histogram plot) and in the tabular form. The number obtained corresponding to an interval in the table and vertical column in the histogram plot represents that how frequently an observation is repeated in the complete dataset. The histogram plots are look like bar graph but not it as there are some basic difference between them. The histogram plot is used for the analysis of quantitative data or continuous variable having some ranges of data which are grouped by an interval or bins while bar graph is used for the categorical data. Moreover, in the histogram plot, the bars are adjacent with each other.

2.4.1.1.2 Pearson's cross-correlation (R) Analysis

The correlation coefficient, denoted by R, was invented by Karl Pearson in 1896. It is the measure of the strength of relationship between two parameters. The coefficient value varies from +1 to -1. The three extreme situation for correlation coefficient is

depicted graphically in Figure 2.15. Here are few interpretations for the correlation coefficient:

1. +1 indicates that a perfect positive correlation between the two parameters which means that as the value of one parameter increases the value of other also get increases in an exact linear proportions.
2. A value of 0 indicates that there is no linear relationship between the parameters.
3. -1 indicates that a perfect negative relationship between the parameters which imply that with the increase in the value of one parameter the value of another parameter get decreases in an exact linear proportions.
4. The value of R ranges from 0 to 0.19, 0.20 to 0.39, 0.40 to 0.59, 0.60 to 0.79 and 0.80 to 1.0 (or 0 to -0.19, -0.20 to -0.39, -0.40 to -0.59, -0.60 to -0.79 and -0.80 to -1.0) suggest a very weal, weak, moderate, strong and very strong, respectively, positive (or negative) linear relationship between the parameters.

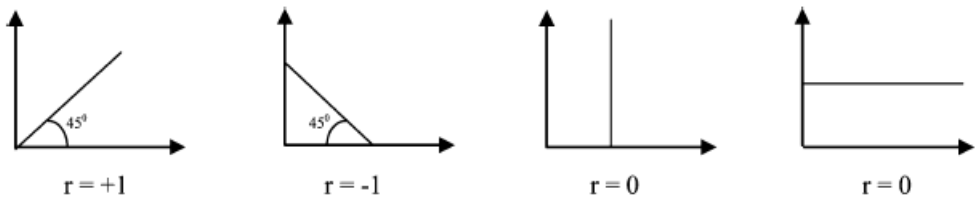


Figure 2.15 Graphical representation of correlation coefficient extreme cases **(J. Verma, 2012)**

The correlation coefficient (R) is calculated as:

$$R = \frac{N\sum XY - (\sum X)(\sum Y)}{\sqrt{[N\sum X^2 - (\sum X)^2][N\sum Y^2 - (\sum Y)^2]}} \quad (2.38)$$

2.4.1.2 Model Performance Measurement Parameters

The precision of each of the developed model was assessed through several statistical performance measurement indicators. The widely used performance measurement indicators are coefficient of determination (R^2), coefficient of correlation (R), mean absolute error (MAE), root mean square error (RMSE), variance accounted for (VAF), Willmott's index of agreement (IOA), index of scattering (IOS) and a20-index (Alzabeebee, 2020; Alzabeebee et al., 2021; Alzabeebee and Chapman, 2020; Bardhan, Gokceoglu, et al., 2021; Bardhan, Samui, et al., 2021; Bharati et al., 2021; Hanandeh et al., 2020; F.E. Jalal et al., 2021; Fazal E Jalal et al., 2021; Kardani, Bardhan, Kim, et al., 2021; Kardani, Bardhan, Samui, et al., 2021; Tenpe and Patel, 2020). The performance measurement parameters MAE, RMSE and IOS belongs to the class of error, whereas, R^2, R, VAF, IOA and a20-index belong to the trend analysis. In order to select an efficient prediction model, espousing many more statistical indicators might be useful for assessing the performance of predictive models in terms of error and trend point of view (Bardhan, Gokceoglu, et al., 2021). Using the least amount of parameters might be challenging, especially when comparing the performance of two or more models at an instant, many a time, the models shows not much difference in their statistical indices value. In that particular situation considering many more statistical performance indices might be helpful in selecting the best-fitted model.

2.4.1.2.1 Coefficient of Determination (R^2)

The coefficient of determination (R^2) usually measures the amount of variability in the response variable that can be explained by the number of regressor variables present in the model. R^2 ranges in between 0 and 1, where 0 indicates that the adopted regressor variables are incapable in explaining the variability in the response variable and a value of 1 represent that the model is perfectly fitted (all the observed data can be estimated

exactly from the predictive model with zero residual sum of square ($SS_{res.}$)). Therefore, a model having R^2 value closer to 1 is selected as the best-fitted model.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_a - y_p)^2}{\sum_{i=1}^{N}(y_a - \overline{y_a})^2} = 1 - \frac{SS_{res}}{SS_{tot}} \qquad 0 < R^2 < 1 \qquad (2.39)$$

2.4.1.2.2 Mean absolute error

The mean absolute error (MAE) is the average of the total amount of error obtained in all the observations. The MAE is estimated through equation (2.40). For any model, the MAE value close to zero means that model is well efficient in predicting the values nearby to the actual value.

$$MAE = \left[\frac{1}{N}\sum_{i=1}^{N}|y_a - y_p|\right] \qquad 0 < MAE < \infty \qquad (2.40)$$

2.4.1.2.3 Root mean square error

Root Mean Square Error (RMSE) is the standard deviation of the errors. RMSE is a measure of how spread out the errors are. In other words, it tells us how the data is concentrated around the line of best fit. The RMSE can be measured using equation (2.41). The best-fitted model shall have RMSE value close to zero.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_a - y_p)^2} \qquad 0 < RMSE < \infty \qquad (2.41)$$

2.4.1.2.4 Variance accounted for

The variance account for (VAF) calculate the variance account for between two signals. It is often used to verify the correctness of a model, by comparing the actual output with the predicted output of the model.

$$VAF\ (\%) = \left[1 - \frac{Var(y_a - y_p)}{Var(y_a)}\right] \times 100 \qquad 0 < \text{VAF} < 100 \qquad (2.42)$$

2.4.1.2.5 Willmott's index of agreement (IOA)

Index of agreement (IOA) is a standardized measure of the degree of model prediction error which varies between 0 and 1. The index of agreement represents the ratio of the mean square error and the potential error as shown through equation (2.43). The agreement value of 1 indicates a perfect match, and 0 indicates no agreement at all.

$$IOA = 1 - \frac{\sum_{i=1}^{N}(y_a - y_p)^2}{\sum_{i=1}^{N}(|y_p - \overline{y_a}| + |y_a - \overline{y_a}|)^2} \qquad 0 < \text{IOA} < 1 \qquad (2.43)$$

2.4.1.2.6 Index of scattering (IOS)

The index of scattering (IOS) is calculated by dividing RMSE with mean of the predicted dataset as represented in equation (2.44). A model with IOS value close to 0 indicates that model is best-fitted for the estimation purposes.

$$IOS = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_a - y_p)^2}}{\overline{y_p}} \qquad 0 < \text{IOS} < \infty \qquad (2.44)$$

2.4.1.2.7 a20-index

It is the measurement of percentage of observations predicted within ±20% error. It ranges from 0 to 100% in which 100% indicates that all the observations can be predicted within the pre-defined index value. Like, a20-index, we can also measure a5, a10 and a15-index for our prediction model.

$$a20 - index = \frac{n20}{N} \times 100 \qquad 0 < \text{a20-index} < 100 \qquad (2.45)$$

In the above equations (from (2.39) to (2.45)), y_a = actual value (laboratory obtained value); y_p = predicted value (value obtained through the developed model); $\overline{y_a}$= mean of actual value; $\overline{y_p}$= mean of predicted value; $n20$ = number of observations exist within error range of ±20% and N= number of observations; SS_{res} and SS_{tot} are sum-of-square of residual and total. The ideal value for these performance measurement parameters are tabulated in Table 2.3.

Table 2.3 Ideal values of different statistical performance measurement parameters

Parameters	R^2	R	MAE	RMSE	VAF	IOA	IOS	a20-index
Ideal value	1	1	0	0	100	1	0	1

In order to select an efficient prediction model, espousing many more statistical indicators might be useful for assessing the performance of predictive models in terms of error and trend point of view. Using the least amount of parameters might be challenging, especially when comparing the performance of two or more models at an instant, many a time, the models show not much difference in their statistical indices value. In that particular situation considering many more statistical performance indices might be helpful in selecting the best-fitted model.

2.4.2 Data Divisional Approaches

Data division is the process of separating the complete dataset into the training and testing subset. The training data is used to train the model and for testing that model, we use test data which also offers us the performance evaluation of the model. The basic problem with machine learning modelling is that we are unknown to the fact that how well a model performs or will perform until it is tested on unseen dataset. The unseen dataset is a dataset which is not used to train the machine learning model or we can say the test dataset. One can build a perfect model on the training data with 100% accuracy or 0 error, but it may fail to generalize for unseen data. It is not a good model as it over-fits the training data. Machine Learning is all about generalization meaning that model's performance can only be measured with data points that have never been used during the training process. To overcome this problem, three data divisional approaches were adopted which are discussed below in details.

2.4.2.1 Statistical approach

In the statistical approach the dataset is divided in such a manner that the values for each of the statistical parameters of both training and testing dataset are as close as possible. The statistical parameters used for the analysis are minimum, maximum, mean, median and standard deviation. Initially, a simple code for the random splitting of dataset was written in python programming language. Using that python code, trial and error method was used to achieve the closest value of all the statistical parameters for both training and testing dataset. The t-test and F-test were conducted to examine the difference between the mean and standard deviation, respectively, of two datasets. The F-test investigate the null hypothesis of no difference in the standard deviation of two dataset and t-test examine the null hypothesis of no difference in the mean of two datasets.

The level of significance (α) used for the investigation was 5% which means that the training and testing dataset are 95% statistically consistent.

2.4.2.2 K-fold cross validation approach

The k-fold cross validation is one of the most prevalent and demanding technique which is widely used in machine learning for splitting the dataset into training and testing subset. The following steps are followed in k-fold approach for splitting the dataset:

1. Initially shuffle the dataset randomly and split it into K number of folds (as seen from Figure 2.16).
2. Once the dataset is separated, the first fold is used as testing dataset and the remaining k-1 folds are used for the training purposes.
3. The model with specific hyper-parameters is trained with training data (K-1 folds) and testing data as 1 fold. The performance of the model is recorded.
4. The above steps are repeated until each of the k-fold got used for testing purpose. This is why it is called k-fold cross validation.
5. Finally, the mean and standard deviation of the model performance is estimated by considering all the scores calculated in step 3 for each of the k-fold model.
6. Repeat the process from step 3 to step 5 for the different values of hyper-parameters. Finally, the hyper-parameter which exhibits the most optimal values for mean and standard deviation is selected.
7. The model is then trained for the training dataset and its performance is measured on the testing dataset.

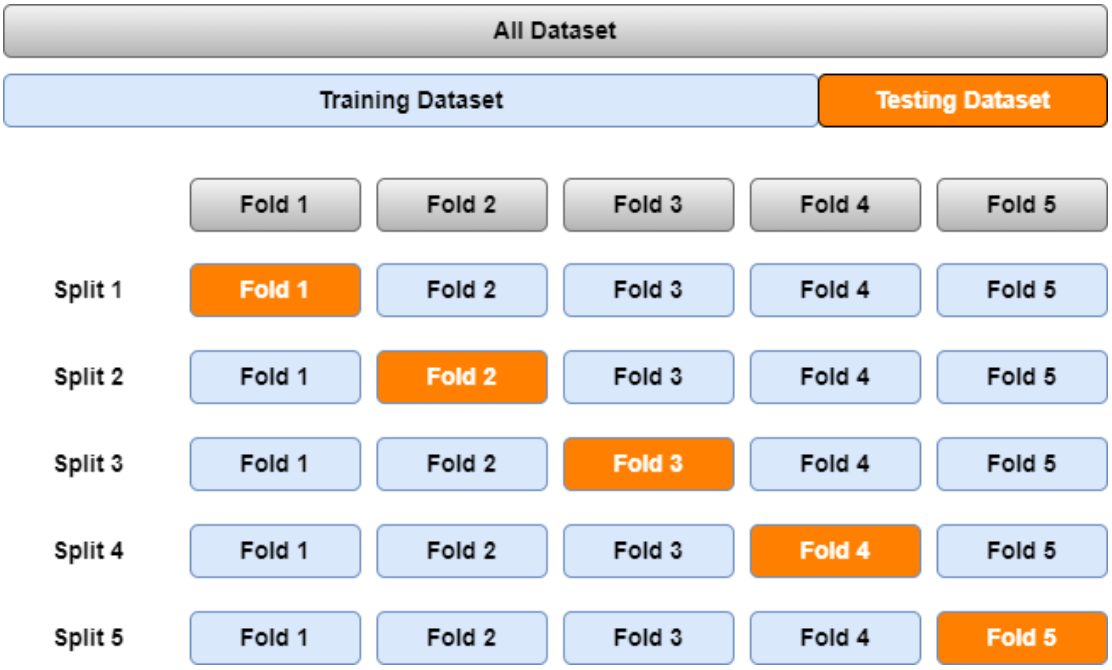


Figure 2.16 Data splitting process in K-fold cross validation

2.4.2.3 Fuzzy C-means (FCM) clustering approach

The most significant fuzzy clustering algorithm is fuzzification K-means and fuzzy C-means (also abbreviated as FCM). Among them, FCM is predominantly used for solving the many problems in the field of geotechnical engineering.

Clustering is the process of separating the dataset into a number of groups, where each group represents the observations that are homogeneous to each other and the objects which are dissimilar to each other are clustered into different groups. In fuzzy clustering, each observation can belong to more than one cluster based on the membership value obtained for them. The total membership value for any observation distributed over the total cluster is 1.0.

Mathematically a fuzzy clustering algorithm tries to minimize an objective function (equation (2.46))) that represents the distance from a give observation to the center of the cluster weighted by the observation's membership value.

$$C = \sum_{n=1}^{N} \frac{\sum_{i,j=1}^{k} u_{in}^2 u_{jn}^2 d_{ij}}{2\sum_{j=1}^{k} u_{jn}^2} \tag{2.46}$$

Where, N is the number of clusters, u_{in} is the unknown membership of point i to the cluster n and k is the number of observations. d_{ij} is the given distance between observation i and j. the constraints for this minimization problem are

$$u_{in} \geq 0 \; for \; i = 1,2, \ldots \ldots \ldots, k; n = 1,2, \ldots \ldots \ldots, N \tag{2.47}$$

$$\sum_{n} u_{in} = 1 \; for \; i = 1,2, \ldots \ldots \ldots, k \tag{2.48}$$

The preceding constraints imply that memberships cannot be negative and each of the observation has a constant total membership value normalized to 1.

Initially, an analytical procedure is used to determine the optimum number of clusters. The silhouette value $s\,(i)$, ranges from -1 to +1, was estimated using equation (2.49) given by Kaufman and Rousseeuw (2009). It is a measure of how well an individual observation lies within the cluster they have been assigned to at the end of the clustering process.

$$s\,(i) = \frac{b\,(i) - a\,(i)}{\max[a\,(i), b\,(i)]} \tag{2.49}$$

Where $a\,(i)$ is the average dissimilarity of the observation i to all other observations in a particular cluster, say A,; and $b\,(i)$ is the smallest average dissimilarity of observation i to all other observations in any cluster, say B, different from cluster A.

For a known observation (i) in cluster A, if $s\,(i)$ is close to 1, this means that the average dissimilarity $a\,(i)$ is smaller than smallest average dissimilarity $b\,(i)$; hence, the observation (i) is considered to have strong membership to cluster A. In this manner, the

average silhouette width $\bar{s}(k)$ for the entire dataset for different number of cluster is estimated. The number of clusters representing the maximum value of $\bar{s}(k)$ is considered as the optimum amount of cluster.

2.5 LITERATURE GAP

1. Previously attempted studies were based on limited dataset (maximum to 389 observations). Using a large amount of dataset is always considered to be much worthwhile from generalization point of view (Bardhan, Gokceoglu, et al., 2021; Bardhan, Samui, et al., 2021; Karimpour-Fard et al., 2019; Kurnaz and Kaya, 2019; G. Verma and Kumar, 2022; Wang and Yin, 2020; Zou et al., 2021).

2. Past researchers investigated for various type of soils such as granular, Non-plastic and plastic coarse-grained and plastic fine-grained soils (as summarized in Table 2.1). The range of input parameters used for the analysis of fine-grained soils were shorter. Some researchers tried to extend the range of input parameters, however, the studies were belongs to the granular soils or sub-base layer materials which are considered to be insufficient for the pavement subgrade layer.

3. Most of the developed models were not validated for independent dataset, except those of Bardhan, Samui, et al. (2021) and Bardhan, Gokceoglu, et al. (2021). However, they have also not cross-validated their on model on literature dataset as well as literature models on their own dataset which might be significant in knowing the influence of soil origin on the predictive ability of any developed model.

4. Past researchers used random data division approach for dividing the dataset into training and testing set, except the study conducted by (Bardhan, Samui, et al., 2021). A randomly divided dataset may be responsible for vanishing some of the significant features of the training dataset, which may lead to under-fitting and over-fitting of the model and subsequently the accuracy of any predictive models.

5. Formerly developed equations were lengthier and complicated making difficult for field engineers.

The main contribution of this study is to develop an efficient model for predicting one of the challenging real-world problems of highway engineers i.e. estimation of soil California bearing ratio value. Aprt from that many sub-objectives (as discussed in section 1.4) have also been fulfilled.