North Asian International Research Journal of Sciences, Engineering & I.T.

IRJIF. I.F. : 3.821

ISSN: 2454-7514 Vol. 8, Issue-3, March-2022

Index Copernicus Value: 52.88 Indian Citation Index Thomson Reuters ID: S-8304-2016

A Peer Reviewed Refereed International Journal

MACHINE LEARNING -BASED STRENGTH ESTIMATION OF GLASS FIBRE CONCRETE USING SUPPORT VECTOR REGRESSION MODEL

[1]DR.K.CHANDRAMOULI[2] SMITHANISSANKARAO [3]J.SREE NAGA CHAITANYA [4]K.MANVITHA [5]D.KOTESWARA RAO

[1]Professor and HOD, NRI Institute of Technology, Guntur, India. Email: koduru_mouli@yahoo.com

[1]Post Doctoral Fellowship Scholor, Department of Computer Science and Engineering, Central Christian University, MALAWI Email: koduru_mouli@yahoo.com

[3]Assistant professor, Civil Engineering Department, NRI Institute of Technology, Guntur, India. Email:jarugumillichaitanya1989@gmail.com

[4]Assistant professor, Computer Science and Engineering Department, NRI Institute of Technology, Guntur, India. Email: manvi.koduru@gmail.com

[5]Associate professor, Computer Science and Engineering Department, NRI Institute of Technology, Guntur, India.

ABSTRACT

In this paper, we explore the potential of a Support Vector Machine Regression (SVR) model to predict the mechanical properties of concrete integrated with glass fiber. Utilizing a dataset with both categorical and continuous variables, such as concrete grade, percentage of glass fiber, and curing periods, we pre-process the data through techniques like one-hot encoding and normalization to prepare for SVR analysis. Our focus is on predicting compressive, splitting, and flexural strengths at intervals of 7, 28, 56, 90, 180, and 360 days. The performance of our SVR model is evaluated based on Mean Squared Error (MSE), aiming to minimize test loss and enhance the accuracy of our predictions. The findings demonstrate the effectiveness of the SVR model in understanding the influence of material composition and processing parameters on concrete strength. This study provides valuable insights for material scientists and civil engineers working towards the development of high-performance concrete, using data from various grades ranging from M20 to M50 and glass fiber percentages from 0% to 0.15%.

***KEYWORDS*: Support Vector Machine Regression, Mean Squared Error, Compressive Strength, Split Tensile Strength, Flexural Strength.**

1. INTRODUCTION

Concrete serves as the structural foundation of contemporary infrastructure, highlighting its importance in construction and civil engineering sectors. The quest for concrete that attains optimal strength and durability is essential, necessitating an enhanced understanding of its compositional and curing variables. Incorporating glass

fibers has been recognized as a potent method to augment the mechanical properties of concrete, offering superior strength, longevity, and resistance to environmental factors. However, accurately predicting the effects of variables such as concrete grade, glass fiber percentage, and curing time on concrete's mechanical characteristics (compressive, splitting, and flexural strengths) remains a complex issue. Traditional experimental methods are time-consuming, costly, and inefficient for exploring the extensive design space of concrete formulations. Consequently, there is an increasing interest in employing computational models, especially Support Vector Machine Regression (SVR), for their capacity to model and predict complex, nonlinear relationships within data, providing a robust tool for property prediction. Support Vector Machine Regression, by leveraging its kernel trick to handle non-linear relationships and its margin of error minimization to ensure model robustness, offers a sophisticated approach to understand and forecast material properties. This research aims to harness the predictive capabilities of SVR to investigate the effects of glass fiber reinforcement among other critical variables on the strength of concrete, thus paving the way for the development of advanced concrete formulations. The steps involved in applying SVR to concrete research include data collection and preprocessing, model design, training the regression model, validation, and using the model for prediction and optimization tasks. This detailed approach aids in efficiently predicting and optimizing concrete's mechanical behavior, potentially transforming practices in material science and engineering.

Support Vector Machine Regression in Glass Fiber Reinforced Concrete (GFRC):

GFRC is a specialized composite material that includes cement, fine aggregates, water, chemical additives, and glass fibers, which enhance its tensile strength, flexibility, and durability. The application of Support Vector Machine Regression to GFRC entails several key stages:

1. **Data Collection and Preprocessing**: Gathering relevant data on GFRC's composition, manufacturing processes, and mechanical properties, followed by preprocessing through one-hot encoding and normalization to ensure data quality and suitability for regression analysis.

2. **Model Architecture Design**: Setting up the SVR model based on the specific goals of the GFRC application, including defining the predictors and the response variable, and choosing an appropriate kernel to model the complex interactions in the data effectively.

3. **Training the SVR Model**: Training the model on preprocessed data using techniques like cross-validation to learn the relationships between input variables (e.g., composition, curing conditions) and output variables (e.g., mechanical properties).

4. **Validation and Testing**: Validating the model using separate data sets to verify its generalization capability, followed by testing to assess its effectiveness in predicting GFRC's mechanical properties under varied conditions.

5. **Optimization and Prediction**: Utilizing the trained model to optimize GFRC's composition and processing parameters to achieve desired properties, and to predict its behavior under novel or altered conditions, supporting more informed design and manufacturing decisions.

Dataset Overview for Support Vector Machine Regression Modelling

This research utilizes a dataset specifically compiled for training and validating a Support Vector Machine Regression (SVR) model. It comprises experimental measurements crucial for understanding the impact of various factors on the mechanical properties of concrete. The dataset contains the following variables:

- **Grade**: Indicates the grade of concrete used, fundamental for understanding the baseline performance of the material without reinforcement.
- **Glass Fibre_%**: Represents the percentage of glass fiber added to the concrete mix, serving as a key input for modelling the reinforcement's effect on the concrete's mechanical properties.
- **No_of_days**: Denotes the age of the concrete samples in days at the testing time, providing insight into how the mechanical strengths evolve over time.
- **Comp Strength (N/m^2)**: The compressive strength of the concrete sample, measured in Newtons per square meter, a crucial output variable for assessing the structural capacity of the reinforced concrete.
- **Split Strength (N/m^2)**: The split tensile strength, another vital output parameter that helps determine the concrete's tensile performance under reinforcement.
- **Flexural Strength (N/m^2)**: Reflects the bending strength of the concrete, a key performance indicator for applications subject to flexural stress.

Collected at various post-curing intervals, the dataset provides a robust foundation for training the SVR model. By incorporating variables such as concrete grade, percentage of glass fiber reinforcement, and sample age, the SVR model is designed to accurately predict the compressive, tensile, and flexural strengths of concrete. This predictive capability is crucial for advancing our understanding of the potential enhancements glass fiber reinforcement can offer to concrete, providing valuable insights that could lead to the development of more durable and resilient building materials.

DATASET:

Grade	Glass_Fibre_%	No_of_days	Comp_Strength	Split_Strength	Flexural_Strength
M20	0	7	24.48	2.53	2.5
M20	0	28	36.6	3.62	3.62
M20	0	56	39.25	3.95	3.95
M20	0	90	43.23	4.28	4.3
M20	0	180	44.12	4.38	4.36
M20	0	360	45.08	4.48	4.47
M20	0.01	7	24.44	2.69	2.66
M20	0.01	28	38.55	3.85	3.8
M20	0.01	56	42.02	4.19	4.14
M20	0.01	90	45.83	4.5	4.44
M20	0.01	180	46.49	4.62	4.57
M20	0.01	360	47.54	4.76	4.7
M20	0.02	7	28.31	2.76	2.83
M20	0.02	28	40.52	4.01	4.05

M20	0.02	56	43.74	4.37	4.38
M20	0.02	90	48.58	4.69	4.81
M20	0.02	180	49.02	4.82	4.87
M20	0.02	360	50.19	4.93	5.02
M20	0.03	7	28.36	2.93	2.95
M20	0.03	28	42.46	4.2	4.21
M20	0.03	56	45.92	4.58	4.59
M20	0.03	90	51.01	4.91	5.02
M20	0.03	180	51.74	5.08	5.06
M20	0.03	360	52.26	5.2	5.23
M20	0.04	7	29.09	2.88	2.84
M20	0.04	28	42.23	4.18	4.18
M20	0.04	56	46.01	4.55	4.94
M20	0.04	90	49.82	4.93	4.98
M20	0.04	180	50.05	5.06	5.09
M20	0.04	360	52.24	5.17	5.16
M20	0.05	7	28.96	2.83	2.76
M20	0.05	28	42.02	4.16	3.99
M20	0.05	56	45.09	4.87	4.67
M20	0.05	90	49.62	5.02	4.79
M20	0.05	180	50.83	5.14	4.8
M20	0.05	360	52.11	5.16	4.94
M20	0.06	7	28.35	2.92	2.89
M20	0.06	28	41.82	4.18	4.18
M20	0.06	56	43.91	4.55	4.56
M20	0.06	90	50.43	4.89	4.89
M20	0.06	180	51.13	5.02	5.07
M20	0.06	360	51.98	5.17	5.19
M20	0.07	7	28.43	2.85	2.88
M20	0.07	28	41.27	4.08	4.12
M20	0.07	56	44.81	4.45	4.49
M20	0.07	90	49.12	4.78	4.82
M20	0.07	180	49.74	4.91	5
M20	0.07	360	51.03	5.05	5.11
M20	0.08	7	28.43	2.83	2.85
M20	0.08	28	40.72	4.06	4.07
M20	0.08	56	43.67	4.42	4.8
M20	0.08	90	48.49	4.79	4.84
M20	0.08	180	49.26	4.91	4.95
M20	0.08	360	50.17	5.02	5.01
M20	0.09	7	27.79	2.78	2.75
M20	0.09	28	40.17	3.98	3.97
M20	0.09	56	43.31	4.33	4.33
M20	0.09	90	47.59	4.7	4.72

M20	0.09	180	48.42	4.81	4.78
M20	0.09	360	49.51	4.91	4.91
M20	0.1	7	26.42	2.72	2.77
M20	0.1	28	39.62	3.95	3.96
M20	0.1	56	42.99	4.3	4.32
M20	0.1	90	43.01	4.62	4.64
M20	0.1	180	46.24	4.78	4.79
M20	0.1	360	49.04	4.9	4.91
M20	0.11	7	27.05	2.63	2.56
M20	0.11	28	39.04	3.87	3.7
M20	0.11	56	42.09	4.53	4.33
M20	0.11	90	46.16	4.66	4.45
M20	0.11	180	47.19	4.78	4.46
M20	0.11	360	48.18	4.8	4.59
M20	0.12	7	26.69	2.66	2.63
M20	0.12	28	38.47	3.81	3.8
M20	0.12	56	41.48	4.15	4.15
M20	0.12	90	45.48	4.5	4.52
M20	0.12	180	46.45	4.61	4.58
M20	0.12	360	47.51	4.71	4.7
M20	0.13	7	26.22	2.62	2.64
M20	0.13	28	37.91	3.75	3.76
M20	0.13	56	40.87	4.08	4.1
M20	0.13	90	44.82	4.38	4.48
M20	0.13	180	45.85	4.53	4.52
M20	0.13	360	46.82	4.64	4.67
M20	0.14	7	25.87	2.6	2.61
M20	0.14	28	37.34	3.72	3.73
M20	0.14	56	40.27	4.05	4.4
M20	0.14	90	44.18	4.39	4.44
M20	0.14	180	44.96	4.5	4.54
M20	0.14	360	46.15	4.61	4.59
M20	0.15	7	25.56	2.57	2.53
M20	0.15	28	36.78	3.67	3.62
M20	0.15	56	39.67	4	3.95
M20	0.15	90	43.54	4.29	4.23
M20	0.15	180	44.19	4.41	4.36
M20	0.15	360	45.47	4.55	4.49
M30	0	7	27.72	2.87	2.84
M30	0	28	41.5	4.11	4.1
M30	0	56	46.05	4.47	4.48
M30	0	90	48.85	4.86	4.88
M30	0	180	49.72	4.97	4.94
M30	0	360	50.84	5.08	5.07

M30	0.01	7	30.38	3.06	3.02
M30	0.01	28	43.85	4.38	4.32
M30	0.01	56	46.96	4.77	4.71
M30	0.01	90	51.87	5.12	5.05
M30	0.01	180	52.82	5.26	5.2
M30	0.01	360	54.07	5.42	5.35
M30	0.02	7	32.02	3.15	3.23
M30	0.02	28	46.21	4.57	4.62
M30	0.02	56	49.53	4.98	5
M30	0.02	90	54.63	5.35	5.49
M30	0.02	180	55.67	5.49	5.55
M30	0.02	360	57.03	5.63	5.72
M30	0.03	7	32.14	3.36	3.38
M30	0.03	28	48.56	4.8	4.82
M30	0.03	56	53.42	5.23	5.25
M30	0.03	90	56.67	5.62	5.74
M30	0.03	180	58.17	5.81	5.79
M30	0.03	360	59.51	5.95	5.98
M30	0.04	7	32.98	3.26	3.21
M30	0.04	28	47.75	4.72	4.73
M30	0.04	56	51.23	5.15	5.58
M30	0.04	90	56.45	5.58	5.63
M30	0.04	180	57.75	5.72	5.75
M30	0.04	360	58.96	5.85	5.83
M30	0.05	7	32.53	3.16	3.08
M30	0.05	28	46.94	4.65	4.45
M30	0.05	56	50.41	5.44	5.21
M30	0.05	90	54.49	5.61	5.35
M30	0.05	180	56.61	5.74	5.36
M30	0.05	360	57.98	5.77	5.52
M30	0.06	7	31.36	3.22	3.18
M30	0.06	28	46.13	4.61	4.61
M30	0.06	56	51.21	5.02	5.03
M30	0.06	90	55.5	5.39	5.4
M30	0.06	180	56.28	5.54	5.59
M30	0.06	360	57.43	5.7	5.72
M30	0.07	7	31.93	3.17	3.21
M30	0.07	28	45.92	4.54	4.59
M30	0.07	56	49.36	4.95	5
M30	0.07	90	54.31	5.31	5.37
M30	0.07	180	55.42	5.46	5.56
M30	0.07	360	56.73	5.62	5.69
M30	0.08	7	31.67	3.18	3.2
M30	0.08	28	45.71	4.55	4.57

M30	0.08	56	49.18	4.96	5.39
M30	0.08	90	54.05	5.38	5.43
M30	0.08	180	55.01	5.51	5.55
M30	0.08	360	56.53	5.64	5.62
M30	0.09	7	31.57	3.14	3.11
M30	0.09	28	45.5	4.5	4.5
M30	0.09	56	49.07	4.91	4.91
M30	0.09	90	53.83	5.33	5.35
M30	0.09	180	54.81	5.45	5.42
M30	0.09	360	56.23	5.56	5.56
M30	0.1	7	30.15	3.11	3.16
M30	0.1	28	45.28	4.51	4.53
M30	0.1	56	49.28	4.92	4.94
M30	0.1	90	52.5	5.28	5.3
M30	0.1	180	54.82	5.46	5.48
M30	0.1	360	55.78	5.6	5.61
M30	0.11	7	31.09	3	2.92
M30	0.11	28	44.52	4.41	4.22
M30	0.11	56	48.79	5.16	4.94
M30	0.11	90	52.96	5.32	5.08
M30	0.11	180	53.7	5.45	5.08
M30	0.11	360	54.78	5.47	5.24
M30	0.12	7	30.57	3.02	2.99
M30	0.12	28	43.76	4.33	4.33
M30	0.12	56	47.52	4.72	4.72
M30	0.12	90	52.05	5.12	5.15
M30	0.12	180	52.67	5.24	5.21
M30	0.12	360	54.03	5.35	5.35
M30	0.13	7	30.05	2.97	2.99
M30	0.13	28	43.01	4.25	4.27
M30	0.13	56	47.21	4.63	4.65
M30	0.13	90	51.16	4.97	5.09
M30	0.13	180	51.69	5.14	5.13
M30	0.13	360	53.13	5.27	5.29
M30	0.14	7	29.53	2.94	2.96
M30	0.14	28	42.26	4.21	4.22
M30	0.14	56	46.04	4.59	4.98
M30	0.14	90	50.26	4.97	5.02
M30	0.14	180	51.02	5.1	5.13
M30	0.14	360	53.06	5.21	5.2
M30	0.15	7	28.99	2.89	2.86
M30	0.15	28	41.49	4.14	4.09
M30	0.15	56	45.23	4.51	4.45
M30	0.15	90	49.35	4.84	4.78

M30	0.15	180	49.99	4.98	4.92
M30	0.15	360	51.42	5.13	5.06
M40	0	7	33.3	3.31	3.28
M40	0	28	47.92	4.74	4.74
M40	0	56	52.24	5.17	5.17
M40	0	90	56.67	5.61	5.64
M40	0	180	58.05	5.74	5.7
M40	0	360	59.56	5.86	5.85
M40	0.01	7	35.73	3.57	3.52
M40	0.01	28	51.12	5.1	5.04
M40	0.01	56	55.56	5.56	5.49
M40	0.01	90	60.81	5.97	5.89
M40	0.01	180	61.53	6.13	6.06
M40	0.01	360	62.97	6.32	6.24
M40	0.02	7	38.01	3.7	3.8
M40	0.02	28	54.31	5.37	5.43
M40	0.02	56	59.19	5.85	5.87
M40	0.02	90	63.57	6.28	6.45
M40	0.02	180	66.25	6.46	6.52
M40	0.02	360	66.97	6.61	6.73
M40	0.03	7	38.29	3.97	4
M40	0.03	28	57.5	5.68	5.71
M40	0.03	56	60.6	6.2	6.22
M40	0.03	90	66.87	6.65	6.8
M40	0.03	180	59.66	6.88	6.86
M40	0.03	360	70.89	7.04	7.08
M40	0.04	7	39.74	3.88	3.83
M40	0.04	28	56.88	5.63	5.64
M40	0.04	56	61.42	6.13	6.65
M40	0.04	90	67.69	6.65	6.7
M40	0.04	180	68.82	6.81	6.85
M40	0.04	360	70.05	6.97	6.94
M40	0.05	7	39.36	3.79	3.69
M40	0.05	28	56.26	5.58	5.34
M40	0.05	56	61.32	6.53	6.25
M40	0.05	90	65.81	6.72	6.42
M40	0.05	180	67.59	6.89	6.42
M40	0.05	360	69.43	6.91	6.62
M40	0.06	7	37.65	3.89	3.84
M40	0.06	28	55.62	5.55	5.56
M40	0.06	56	58.47	6.05	6.06
M40	0.06	90	64.32	6.5	6.51
M40	0.06	180	67.43	6.68	6.74
M40	0.06	360	69.19	6.87	6.9

M40	0.07	7	38.45	3.8	3.84
M40	0.07	28	55.03	5.44	5.5
M40	0.07	56	59.96	5.93	5.99
M40	0.07	90	65.47	6.37	6.43
M40	0.07	180	66.12	6.54	6.67
M40	0.07	360	67.89	6.73	6.82
M40	0.08	7	38.08	3.79	3.81
M40	0.08	28	54.44	5.42	5.44
M40	0.08	56	59.31	5.91	6.42
M40	0.08	90	63.39	6.41	6.47
M40	0.08	180	65.42	6.56	6.61
M40	0.08	360	67.19	6.72	6.7
M40	0.09	7	37.64	3.72	3.69
M40	0.09	28	53.85	5.33	5.33
M40	0.09	56	58.67	5.81	5.81
M40	0.09	90	64.05	6.3	6.33
M40	0.09	180	64.95	6.45	6.41
M40	0.09	360	66.52	6.59	6.58
M40	0.1	7	36.64	3.66	3.72
M40	0.1	28	53.26	5.31	5.33
M40	0.1	56	56.43	5.79	5.81
M40	0.1	90	63.72	6.21	6.23
M40	0.1	180	65.37	6.42	6.44
M40	0.1	360	66.45	6.58	6.6
M40	0.11	7	36.48	3.51	3.43
M40	0.11	28	52.23	5.18	4.96
M40	0.11	56	56.89	6.06	5.8
M40	0.11	90	62.12	6.24	5.96
M40	0.11	180	62.96	6.39	5.96
M40	0.11	360	64.56	6.42	6.15
M40	0.12	7	35.76	3.54	3.5
M40	0.12	28	51.17	5.07	5.06
M40	0.12	56	55.23	5.52	5.52
M40	0.12	90	60.74	5.99	6.02
M40	0.12	180	61.59	6.13	6.09
M40	0.12	360	62.98	6.26	6.25
M40	0.13	7	35.03	3.46	3.49
M40	0.13	28	50.11	4.95	4.97
M40	0.13	56	54.12	5.4	5.42
M40	0.13	90	59.14	5.8	5.93
M40	0.13	180	60.58	5.99	5.98
M40	0.13	360	61.98	6.14	6.17
M40	0.14	7	34.27	3.41	3.43
M40	0.14	28	49.05	4.89	4.9

M40	0.14	56	53.45	5.32	5.78
M40	0.14	90	58.16	5.77	5.83
M40	0.14	180	59.09	5.91	5.96
M40	0.14	360	60.48	6.05	6.04
M40	0.15	7	33.52	3.35	3.31
M40	0.15	28	47.98	4.79	4.73
M40	0.15	56	51.82	5.22	5.15
M40	0.15	90	57.06	5.6	5.52
M40	0.15	180	57.73	5.75	5.69
M40	0.15	360	59.23	5.93	5.85
M50	0	7	38.25	3.74	3.71
M50	0	28	54.18	5.36	5.36
M50	0	56	59.96	5.84	5.85
M50	0	90	64.37	6.34	6.37
M50	0	180	66.09	6.49	6.45
M50	0	360	67.45	6.63	6.62
M50	0.01	7	39.19	3.97	3.92
M50	0.01	28	56.89	5.68	5.6
M50	0.01	56	61.98	6.18	6.11
M50	0.01	90	67.57	6.64	6.55
M50	0.01	180	68.38	6.82	6.75
M50	0.01	360	70.21	7.03	6.94
M50	0.02	7	41.66	4.06	4.17
M50	0.02	28	59.61	5.89	5.96
M50	0.02	56	64.37	6.42	6.45
M50	0.02	90	70.91	6.9	7.08
M50	0.02	180	71.72	7.08	7.16
M50	0.02	360	73.58	7.26	7.38
M50	0.03	7	43.63	4.31	4.34
M50	0.03	28	62.31	6.16	6.18
M50	0.03	56	69.55	6.71	6.74
M50	0.03	90	77.24	7.21	7.37
M50	0.03	180	78.65	7.45	7.43
M50	0.03	360	79.44	7.63	7.67
M50	0.04	7	42.77	4.18	4.12
M50	0.04	28	61.28	6.06	6.07
M50	0.04	56	67.16	6.6	7.17
M50	0.04	90	72.86	7.16	7.22
M50	0.04	180	73.81	7.34	7.38
M50	0.04	360	75.79	7.51	7.48
M50	0.05	7	41.92	4.05	3.95
M50	0.05	28	60.25	5.97	5.72
M50	0.05	56	65.64	6.99	6.69
M50	0.05	90	71.67	7.2	6.87

M50	0.05	180	72.51	7.37	6.88
M50	0.05	360	74.24	7.4	7.09
M50	0.06	7	40.73	4.14	4.08
M50	0.06	28	59.21	5.91	5.92
M50	0.06	56	63.28	6.44	6.45
M50	0.06	90	68.64	6.92	6.93
M50	0.06	180	73.52	7.11	7.18
M50	0.06	360	74.9	7.31	7.34
M50	0.07	7	41.02	4.07	4.11
M50	0.07	28	58.89	5.82	5.88
M50	0.07	56	64.16	6.35	6.41
M50	0.07	90	68.92	6.81	6.88
M50	0.07	180	70.88	7	7.13
M50	0.07	360	72.73	7.2	7.29
M50	0.08	7	40.85	4.06	4.09
M50	0.08	28	58.37	5.81	5.83
M50	0.08	56	63.59	6.33	6.88
M50	0.08	90	69.43	6.87	6.94
M50	0.08	180	70.22	7.04	7.09
M50	0.08	360	72.16	7.2	7.18
M50	0.09	7	40.68	4.03	3.99
M50	0.09	28	58.25	5.77	5.76
M50	0.09	56	63.46	6.28	6.29
M50	0.09	90	69.29	6.82	6.85
M50	0.09	180	70.09	6.97	6.93
M50	0.09	360	71.89	7.12	7.12
M50	0.1	7	39.22	3.98	4.05
M50	0.1	28	57.94	5.78	5.79
M50	0.1	56	61.61	6.3	6.32
M50	0.1	90	66.28	6.76	6.78
M50	0.1	180	72.23	6.99	7.01
M50	0.1	360	73.41	7.16	7.18
M50	0.11	7	39.94	3.85	3.75
M50	0.11	28	57.19	5.67	5.43
M50	0.11	56	62.32	6.63	6.35
M50	0.11	90	68.03	6.83	6.52
M50	0.11	180	68.71	7	6.53
M50	0.11	360	70.48	7.03	6.73
M50	0.12	7	39.42	3.9	3.86
M50	0.12	28	56.44	5.59	5.58
M50	0.12	56	61.49	6.08	6.09
M50	0.12	90	67.14	6.61	6.64
M50	0.12	180	67.93	6.76	6.72
M50	0.12	360	69.56	6.9	6.9

M50	0.13	7	38.89	3.85	3.87
M50	0.13	28	55.69	5.51	5.53
M50	0.13	56	60.72	6	6.02
M50	0.13	90	66.26	6.44	6.59
M50	0.13	180	66.97	6.66	6.64
M50	0.13	360	68.73	6.82	6.85
M50	0.14	7	38.37	3.82	3.85
M50	0.14	28	54.94	5.47	5.49
M50	0.14	56	59.86	5.96	6.48
M50	0.14	90	65.37	6.47	6.53
M50	0.14	180	66.05	6.62	6.67
M50	0.14	360	67.77	6.78	6.76
M50	0.15	7	37.85	3.78	3.73
M50	0.15	28	54.19	5.41	5.34
M50	0.15	56	59.04	5.89	5.82
M50	0.15	90	64.46	6.33	6.24
M50	0.15	180	65.15	6.5	6.43
M50	0.15	360	66.84	6.7	6.61

SVR Model Snippet

```
# Import necessary libraries
importpandasaspd# pandas is a library for data manipulation and analysis
fromsklearn.model_selectionimport train_test_split, GridSearchCV  # scikit-learn provides tools for machine learning tasks
fromsklearn.preprocessingimport StandardScaler  # StandardScaler is used for feature scaling
fromsklearn.svmimport SVR  # Support Vector Regression (SVR) is a regression algorithm in scikit-learn
importjoblib# joblib is used to save and load models

data = pd.read_csv("data.csv")
data.info()

# Reading the data from a file into a table
# data = pd.read_csv("data_concrete.csv")  # Put your file path here

# Keeping only important columns
data_comp = data[['Grade', 'No_of_days', 'Glass_Fibre_%', 'Comp_Strength']]

# Changing 'No_of_days' into categories
data_comp['No_of_days'] = data_comp['No_of_days'].astype(str)

# Changing 'Grade' into numbers
data_comp_with_dummies = pd.get_dummies(data_comp, columns=['Grade', 'No_of_days'])

# Separating features and results
X = data_comp_with_dummies.drop(['Comp_Strength'], axis=1)  # Features
y = data_comp_with_dummies['Comp_Strength']  # Results

# Scaling numbers to make them easier to work with
numeric_features = ['Glass_Fibre_%']  # The numbers we want to scale
numeric_transformer = StandardScaler()  # The tool for scaling
```

```
# Applying scaling to the numbers
X[numeric_features] = numeric_transformer.fit_transform(X[numeric_features])  # Scaling the numbers
print(X.columns)  # Showing the column names after scaling

# Splitting the data into training and testing parts
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # Splitting the data

# Define the SVM model
svm_model_comp = SVR()  # Creating a Support Vector Regression model

# Define a grid of parameters to search
param_grid = {
'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  # Types of mathematical functions used for modeling
'C': [0.1, 1, 10],  # Penalty parameter for regularization (controls the trade-off between smooth decision boundary and classifying the training points correctly)
'gamma': ['scale', 'auto'],  # Kernel coefficient (scale: uses 1 / (n_features * X.var()) as gamma, auto: uses 1 / n_features)
'degree': [2, 3, 4]  # Degree of the polynomial kernel function (only used for 'poly' kernel)
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=svm_model_comp, param_grid=param_grid, scoring='neg_mean_absolute_error', cv=5, n_jobs=-1)
# GridSearchCV: Searching for the best combination of hyperparameters by trying all possible combinations
# estimator: The model to be tuned (SVR in this case)
# param_grid: The grid of parameters to search
# cv: Number of folds in cross-validation (5-fold cross-validation used here)
# n_jobs: Number of jobs to run in parallel (-1 means using all available processors)

grid_search.fit(X_train, y_train)  # Fitting the model with different combinations of parameters

# Get the best parameters
best_params_comp = grid_search.best_params_  # Finding the combination that gives the best performance
print("Best Parameters:", best_params_comp)

# Evaluate the model with the best parameters
best_model_comp = grid_search.best_estimator_  # Using the best parameters to create the final model
test_score = best_model_comp.score(X_test, y_test)  # Evaluating the final model on the test data
print(f"Test Score: {test_score}")  # Printing the performance of the model on the test data

best_params = grid_search.best_params_
best_score = grid_search.best_score_
# Since the score is negative, to get the actual MAE, we take its absolute value.
print(f"Best Mean Absolute Error (MAE): {-best_score}")

defpredict_with_best_model_comp(grade, num_days, glass_fibre, best_model_comp):
"""
   Predict using the best model found during grid search.

   Parameters:
   - grade: Grade (categorical)
   - num_days: No_of_days (categorical)
   - glass_fibre: Glass_Fibre_% (numerical)
   - best_model: Fitted SVM model with the best parameters
```

```
    Returns:
    - prediction: Predicted value
    """
# Create a DataFrame with the input data
    input_data = pd.DataFrame({
'Grade': [grade],  # Creating a DataFrame with a single row containing the 'grade' input
'No_of_days': [num_days],  # Adding 'num_days' input to the DataFrame
'Glass_Fibre_%': [glass_fibre]  # Adding 'glass_fibre' input to the DataFrame
    })

# Normalize numerical variables
    numeric_features = ['Glass_Fibre_%']  # List of numerical features to be scaled
    numeric_transformer = StandardScaler()  # Creating a StandardScaler object
    input_data[numeric_features] = numeric_transformer.fit_transform(input_data[numeric_features])  # Scaling numerical features

# Convert categorical variables into dummy variables
    input_data = pd.get_dummies(input_data, columns=['Grade', 'No_of_days'])  # Converting categorical variables into dummy variables

# Ensure all grade columns are present
    grade_columns = ['Glass_Fibre_%', 'Grade_M20', 'Grade_M30', 'Grade_M40', 'Grade_M50',
'No_of_days_180', 'No_of_days_28', 'No_of_days_360',
'No_of_days_56', 'No_of_days_7', 'No_of_days_90']  # List of expected grade columns
for col in grade_columns:
if col notin input_data.columns:
            input_data[col] =0# Adding missing columns with default value 0

# Reorder columns to match the order during training
    input_data = input_data[grade_columns]  # Reordering columns to match the order during training

# Make prediction
    prediction = best_model_comp.predict(input_data)  # Making prediction using the best model and input data

return prediction  # Returning the predicted value

# Keeping only important columns
data_flex = data[['Grade', 'No_of_days', 'Glass_Fibre_%', 'Flexural_Strength']]

# Changing 'No_of_days' into categories
data_flex['No_of_days'] = data_flex['No_of_days'].astype(str)

# Changing 'Grade' into numbers
data_flex_with_dummies = pd.get_dummies(data_flex, columns=['Grade', 'No_of_days'])

# Separating features and results
X = data_flex_with_dummies.drop(['Flexural_Strength'], axis=1)  # Features
y = data_flex_with_dummies['Flexural_Strength']  # Results

# Scaling numbers to make them easier to work with
numeric_features = ['Glass_Fibre_%']  # The numbers we want to scale
numeric_transformer = StandardScaler()  # The tool for scaling

# Applying scaling to the numbers
X[numeric_features] = numeric_transformer.fit_transform(X[numeric_features])  # Scaling the numbers
```

```
print(X.columns)  # Showing the column names after scaling

# Splitting the data into training and testing parts
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # Splitting the data

# Define the SVM model
svm_model_flex = SVR()  # Creating a Support Vector Regression model for flexural strength

# Define a grid of parameters to search
param_grid = {
'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  # Types of mathematical functions used for modeling
'C': [0.1, 1, 10],  # Penalty parameter for regularization (controls the trade-off between smooth decision boundary and classifying the training points correctly)
'gamma': ['scale', 'auto'],  # Kernel coefficient (scale: uses 1 / (n_features * X.var()) as gamma, auto: uses 1 / n_features)
'degree': [2, 3, 4]  # Degree of the polynomial kernel function (only used for 'poly' kernel)
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=svm_model_flex, param_grid=param_grid, scoring='neg_mean_absolute_error', cv=5, n_jobs=-1)
# GridSearchCV: Searching for the best combination of hyperparameters by trying all possible combinations
# estimator: The model to be tuned (SVR in this case)
# param_grid: The grid of parameters to search
# cv: Number of folds in cross-validation (5-fold cross-validation used here)
# n_jobs: Number of jobs to run in parallel (-1 means using all available processors)

grid_search.fit(X_train, y_train)  # Fitting the model with different combinations of parameters

# Get the best parameters
best_params_flex = grid_search.best_params_  # Finding the combination that gives the best performance
print("Best Parameters:", best_params_flex)

# Evaluate the model with the best parameters
best_model_flex = grid_search.best_estimator_  # Using the best parameters to create the final model
test_score = best_model_flex.score(X_test, y_test)  # Evaluating the final model on the test data
print(f"Test Score: {test_score}")  # Printing the performance of the model on the test data

best_params = grid_search.best_params_
best_score = grid_search.best_score_
# Since the score is negative, to get the actual MAE, we take its absolute value.
print(f"Best Mean Absolute Error (MAE): {-best_score}")

defpredict_with_best_model_flex(grade, num_days, glass_fibre, best_model_flex):
"""
    Predict using the best model found during grid search.

    Parameters:
    - grade: Grade (categorical)
    - num_days: No_of_days (categorical)
    - glass_fibre: Glass_Fibre_% (numerical)
    - best_model: Fitted SVM model with the best parameters

    Returns:
    - prediction: Predicted value
    """
```

```
# Create a DataFrame with the input data
    input_data = pd.DataFrame({
'Grade': [grade],  # Creating a DataFrame with a single row containing the 'grade' input
'No_of_days': [num_days],  # Adding 'num_days' input to the DataFrame
'Glass_Fibre_%': [glass_fibre]  # Adding 'glass_fibre' input to the DataFrame
    })

# Normalize numerical variables
    numeric_features = ['Glass_Fibre_%']  # List of numerical features to be scaled
    numeric_transformer = StandardScaler()  # Creating a StandardScaler object
    input_data[numeric_features] = numeric_transformer.fit_transform(input_data[numeric_features])  # Scaling numerical features

# Convert categorical variables into dummy variables
    input_data = pd.get_dummies(input_data, columns=['Grade', 'No_of_days'])  # Converting categorical variables into dummy variables

# Ensure all grade columns are present
    grade_columns = ['Glass_Fibre_%', 'Grade_M20', 'Grade_M30', 'Grade_M40', 'Grade_M50',
'No_of_days_180', 'No_of_days_28', 'No_of_days_360',
'No_of_days_56', 'No_of_days_7', 'No_of_days_90']  # List of expected grade columns
for col in grade_columns:
if col notin input_data.columns:
            input_data[col] =0# Adding missing columns with default value 0

# Reorder columns to match the order during training
    input_data = input_data[grade_columns]  # Reordering columns to match the order during training

# Make prediction
    prediction = best_model_flex.predict(input_data)  # Making prediction using the best model and input data

return prediction  # Returning the predicted value

# Reading the data from a file into a table
# data_split = pd.read_csv("data_concrete.csv")  # Put your file path here

# Keeping only important columns
data_split = data[['Grade', 'No_of_days', 'Glass_Fibre_%', 'Split_Strength']]

# Changing 'No_of_days' into categories
data_split['No_of_days'] = data_split['No_of_days'].astype(str)

# Changing 'Grade' into numbers
data_split_with_dummies = pd.get_dummies(data_split, columns=['Grade', 'No_of_days'])

# Separating features and results
X = data_split_with_dummies.drop(['Split_Strength'], axis=1)  # Features
y = data_split_with_dummies['Split_Strength']  # Results

# Scaling numbers to make them easier to work with
numeric_features = ['Glass_Fibre_%']  # The numbers we want to scale
numeric_transformer = StandardScaler()  # The tool for scaling

# Applying scaling to the numbers
X[numeric_features] = numeric_transformer.fit_transform(X[numeric_features])  # Scaling the numbers
print(X.columns)  # Showing the column names after scaling
```

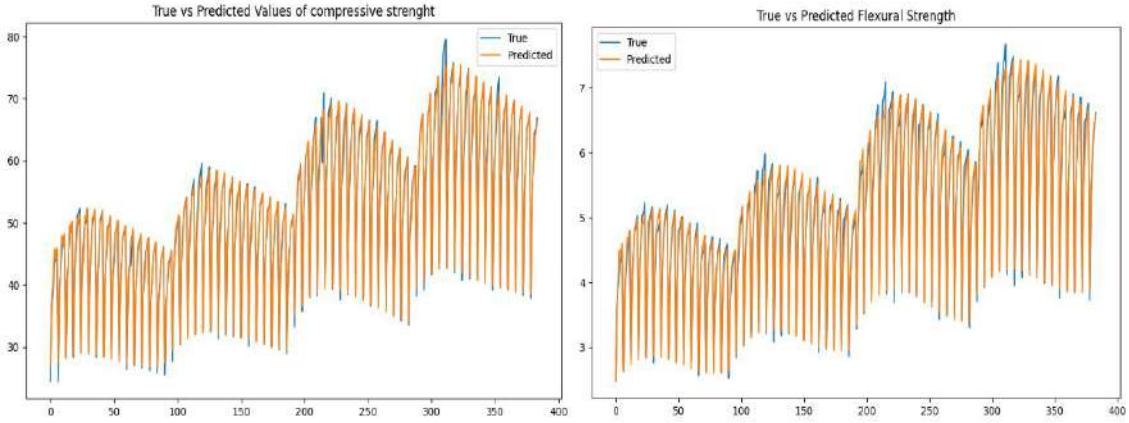
```
# Splitting the data into training and testing parts
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # Splitting the data

# Define the SVM model
svm_model_split = SVR()  # Creating a Support Vector Regression model for split strength

# Define a grid of parameters to search
param_grid = {
'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  # Types of mathematical functions used for modeling
'C': [0.1, 1, 10],  # Penalty parameter for regularization (controls the trade-off between smooth decision boundary and classifying the training points correctly)
'gamma': ['scale', 'auto'],  # Kernel coefficient (scale: uses 1 / (n_features * X.var()) as gamma, auto: uses 1 / n_features)
'degree': [2, 3, 4]  # Degree of the polynomial kernel function (only used for 'poly' kernel)
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=svm_model_split, param_grid=param_grid, scoring='neg_mean_absolute_error', cv=5, n_jobs=-1)
# GridSearchCV: Searching for the best combination of hyperparameters by trying all possible combinations
# estimator: The model to be tuned (SVR in this case)
# param_grid: The grid of parameters to search
# cv: Number of folds in cross-validation (5-fold cross-validation used here)
# n_jobs: Number of jobs to run in parallel (-1 means using all available processors)

grid_search.fit(X_train, y_train)  # Fitting the model with different combinations of parameters

# Get the best parameters
best_params_split = grid_search.best_params_  # Finding the combination that gives the best performance
print("Best Parameters:", best_params_split)

# Evaluate the model with the best parameters
best_model_split = grid_search.best_estimator_  # Using the best parameters to create the final model
test_score = best_model_split.score(X_test, y_test)  # Evaluating the final model on the test data
print(f"Test Score: {test_score}")  # Printing the performance of the model on the test data
best_score = grid_search.best_score_
# Since the score is negative, to get the actual MAE, we take its absolute value.
print(f"Best Mean Absolute Error (MAE): {-best_score}")

defpredict_with_best_model_split(grade, num_days, glass_fibre, best_model_split):
"""
   Predict using the best model found during grid search.

   Parameters:
   - grade: Grade (categorical)
   - num_days: No_of_days (categorical)
   - glass_fibre: Glass_Fibre_% (numerical)
   - best_model: Fitted SVM model with the best parameters

   Returns:
   - prediction: Predicted value
   """
# Create a DataFrame with the input data
   input_data = pd.DataFrame({
'Grade': [grade],  # Creating a DataFrame with a single row containing the 'grade' input
'No_of_days': [num_days],  # Adding 'num_days' input to the DataFrame
```

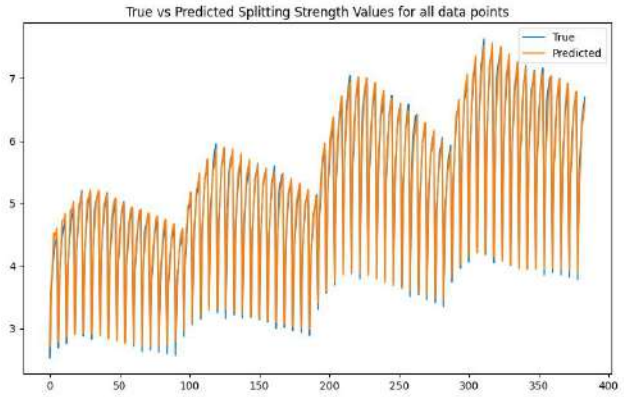
```
'Glass_Fibre_%': [glass_fibre]  # Adding 'glass_fibre' input to the DataFrame
    })

# Normalize numerical variables
    numeric_features = ['Glass_Fibre_%']  # List of numerical features to be scaled
    numeric_transformer = StandardScaler()  # Creating a StandardScaler object
    input_data[numeric_features] = numeric_transformer.fit_transform(input_data[numeric_features])  # Scaling numerical features

# Convert categorical variables into dummy variables
    input_data = pd.get_dummies(input_data, columns=['Grade', 'No_of_days'])  # Converting categorical variables into dummy variables

# Ensure all grade columns are present
    grade_columns = ['Glass_Fibre_%', 'Grade_M20', 'Grade_M30', 'Grade_M40', 'Grade_M50',
'No_of_days_180', 'No_of_days_28', 'No_of_days_360',
'No_of_days_56', 'No_of_days_7', 'No_of_days_90']  # List of expected grade columns
for col in grade_columns:
if col notin input_data.columns:
        input_data[col] =0# Adding missing columns with default value 0

# Reorder columns to match the order during training
    input_data = input_data[grade_columns]  # Reordering columns to match the order during training

# Make prediction
    prediction = best_model_split.predict(input_data)  # Making prediction using the best model and input data

return prediction  # Returning the predicted value
```

Model Example Generation.

predicted_value_comp = **predict_with_best_model_comp** ('M30', '90', 0.07, best_model_comp)[0]
print (f"Predicted compression strength: {predicted_value_comp}")
Predicted compression strength: 31.890640403401157
predicted_value_flex = **predict_with_best_model_flex(**'M30', '90', 0., best_model_flex)[0]
print (f"Predicted flexural strength: {predicted_value_flex}")
Predicted flexural strength: 5.45417151095241
predicted_value_split = **predict_with_best_model_split** ('M30', '90', 0., best_model_split)[0]
print (f"Predicted splitting strength: {predicted_value_split}")
Predicted splitting strength: 5.39128242703077

Model Predictions Vs True Values





DISCUSSIONS

This study has demonstrated the capability of a Support Vector Machine Regression (SVR) model to predict the mechanical properties of glass fiber reinforced concrete (GFRC) across different grades and curing periods. The results indicate that the SVR model can accurately forecast the compressive, splitting, and flexural strengths of GFRC, consistent with the experimental data used for training and validation.

Interpretation of Predictive Performance

The Support Vector Machine Regression model achieved a significant level of accuracy with a Mean Squared Error (MSE) that suggests high reliability in practical scenarios. This predictive success highlights the efficiency of SVR in modeling complex, nonlinear relationships in materials science, particularly in the context of composite materials like GFRC, where understanding the impact of variables like fiber content and curing time is crucial.

Comparison with Traditional Methods

Traditionally, the development of concrete formulations relies heavily on empirical methods and extensive laboratory testing. By incorporating SVR into this process, we can significantly reduce the time and resources required for experimental tests. The model's ability to predict outcomes from a variety of input combinations provides a robust tool for simulating different scenarios, thereby aiding in the formulation of concrete with optimized properties.

Challenges and Limitations

While the results are promising, several challenges remain. The accuracy of the model is highly dependent on the quality and range of the data provided. Thus, the model's performance might degrade when applied beyond the bounds of the training data. Additionally, the current model does not account for potential anomalies in raw materials or manufacturing inconsistencies, which could affect the real-world applicability of the predictions.

Future Directions

Future research could focus on expanding the dataset to include a wider range of GFRC compositions and curing conditions. Integrating data from real-world applications of GFRC could also enhance the model's utility, making it a more comprehensive tool for predicting concrete behaviour under varied environmental conditions. Further, exploring more sophisticated machine learning techniques or hybrid models combining SVR with other predictive

approaches could potentially uncover deeper insights and improve prediction accuracy.

Integration with Industry Practices

Implementing this Support Vector Machine Regression model in a practical, industrial context could transform the way materials are selected and used in construction projects. By providing rapid feedback on material properties, the model could facilitate more informed decision-making, leading to safer, more efficient, and cost-effective construction practices.

3. CONCLUSIONS

Our research utilizing a Support Vector Machine Regression (SVR) model to predict concrete strength characteristics marks a significant advancement in the fields of material science and structural engineering. The SVR model demonstrates a robust capability for accurately forecasting the effects of various factors—such as the grade of concrete, glass fibre content, and curing time—on the compressive, splitting, and flexural strengths of concrete. Specifically, the model achieved Mean Absolute Errors of 0.796 for Compressive Strength, 0.13 for Flexural Strength, and 0.086 for Splitting Strength. These results validate the efficiency of Support Vector Machine Regression in modelling complex, nonlinear relationships within material behaviours, highlighting the potential for optimizing concrete formulations for enhanced durability and performance. Furthermore, the study underscores the importance of sophisticated data preprocessing techniques, such as one-hot encoding and normalization, in preparing dataset variables for effective analysis using SVR. Looking ahead, we advocate for the exploration of more advanced machine learning techniques and the integration of a broader range of material and environmental factors to further enhance the predictive accuracy and utility of these models in real-world construction and material design applications.

4. REFERENCES

1. Dr. K. Chandramouli, J. Sree Naga Catania, STUDY ON CONCRETE BY USING KENAF FIBRES WITH PARTIAL REPLACEMENT OF CEMENT WITH BAMBOO LEAF ASH, Anveshana's International Journal of Research in Engineering and Applied Sciences,7(8),2022,pp:53-56.
2. Chandramouli K., Srinivasa Rao P., Pannirselvam N., Seshadri Sekhar T. and Sravana P. STRENGTH PROPERTIES OF GLASS FIBRE CONCRETE, ARPN Journal of Engineering and Applied Sciences, VOL. 5, NO. 4, APRIL 2010,pp:1-6.
3. Ahmad, J., Khan, M. I., Kumar, A., Hussain, T., & Fatima, N. *Glass Fibers Reinforced Concrete: Overview on Properties and Durability.* PMC, 2022, vol. –, no. –, pp. –, Article 9331547.
4. Pham, A.-D., *et al.Predicting Compressive Strength of High-Performance Concrete Using Metaheuristic-Optimized Least Squares Support Vector Regression.* ASCE, 2024.
5. Yahiaoui, D., Saadi, M., &Bouzid, T. *Compressive Behavior of Concrete Containing Glass Fibers and Confined with Glass FRP Composites.* International Journal of Concrete Structures and Materials, 2022, Vol. 16.
6. J.Sree Naga Chaitanya, Dr.K.Chandramouli, K.Divya,YarrabothulaBabuMahendra Reddy, ADDITION OF BACTERIAL SUBTILIS AND DOLOMITE POWDER TO CONCRETE, International Journal of Engineering Technology Research & Management,8(3),2024,pp:83-86.

7. Rabie, M., *et al.Glass Fibre Concrete: Experimental Investigation and Predictive Analysis Using Advanced Machine Learning.* Journal of Construction & Building Materials, 2025.
8. Wang, W., *et al.Predicting Compressive Strength of High-Performance Concrete Using Interpretable Machine Learning Models.* Scientific Reports, 2024, Vol. 14, Article 28346.
9. Celik, G. *Determination of Concrete Compressive Strength from Images Using CNN-SVR Integration.* Journal of Materials in Civil Engineering, 2024.
10. Kariappa, M. S., Shete, G.N. Experimental study on the performance of alkali resistant glass fiber, International Journal of Emerging Trends in Science and Technology, 3(5), p. 3915-3922, 2016.
11. Banerjee, P. *Effects of Glass Fibre on the Strength and Properties of Concrete.* E3S Conferences: ICSTCE 2023.
12. N. Pannirselvam, K. Chandramouli, V. Anitha, "Experimental Investigation on Special Concrete using Steel Nail", International Journal of Recent Technology and Engineering (IJRTE), March 2019, 7(6S).