**INTRODUCTION**

Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time as it lessens waiting time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

**PROBLEM STATEMENT**

I have been given a dataset containing data on various factors that impact the usage of bike rental services. The main objective of this project is to understand the underlying trends and make prediction about the demand for bike rentals at different times. Our aim is also to identify the key factors that influence bike rental demand.

**DATA DESCRIPTION**

**Variables Description**

* **Date**: - Date
* **Rented Bike Count**: - Number of bikes rented per hour
* **Hour**: - Hour of the day (0-23)
* **Temperature**: - Temperature of the day (in degrees Celsius)
* **Humidity**: - Measure of humidity in percentage
* **Windspeed**: - Speed of the wind in m/s
* **Visibility**: - Measure of visibility in metres
* **Dew Point Temperature**: - Dew Point Temperature measure
* **Solar Radiation**: - Solar Radiation Measure
* **Rainfall**: - Rainfall in mm
* **Snowfall**: - Measure of snowfall in cm
* **Seasons**: - 1=Spring, 2=Summer, 3=Fall, 4=Winter
* **Holiday**: - Whether a holiday or not
* **Functional Day**: - Whether a functional day or not

**BREAKDOWN OF DATASET**

I have been given a .csv file which I load onto our Collaboratory notebook using !gdown method. !gdown method helps us to load a file available on Google Drive with a shareable link. I import necessary libraries –

1. **Numpy** and **Pandas** for mathematical calculations and dataframe manipulations
2. **Seaborn** and **Matplotlib** for data visualization
3. **Sklearn.linear\_model** – **LinearRegression**, **Lasso**, **Ridge**, **ElasticNet**
4. **Sklearn.ensemble** – **GradientBoostingRegressor**, **RandomForestRegressor**
5. **Sklearn.model\_selection** – **train\_test\_split**, **cross\_validate**, **GridSearchCV**
6. **Metrics** – **r2\_score**, **mean\_squared\_error**, **accuracy\_score**, **mean\_absolute\_error**
7. **Sklearn.preprocessing** – **MinMaxScaler**, **StandardScaler**
8. **Statsmodels.stats.outliers\_influence** import **variance\_inflation\_factor**
9. **Datetime**
10. **Missingno**
11. **Scipy.stats**

**DATA FIRST VIEW, CLEANING, WRANGLING AND TRANSFORMATION**

Data cleaning is the first and foremost step in any project. This includes removing duplicates from our dataset, finding any missing/null values, removing them if necessary or replacing them by meaningful values wherever applicable. For our dataset-

1. Dataset includes 8760 rows and 15 columns
2. There are no duplicates in our dataset
3. There are no missing/null values in our dataset
4. Renaming all column names to a simpler one which helps while writing code
5. Converted the “date” column from object to datetime and extracted “month” and “day of the Iek” by creating new columns “month” and “day” respectively.
6. Created a new column “week\_ends\_or\_days”, where 1 represents weekends and 0 represents weekdays. I used “day” column created earlier for this.

**DATA VIZUALIZATION**

Data Vizualization helps in simplifying complex data, reveals patterns and trends, aids in decision making, increases accessibility and helps us in identifying areas that need attention or improvement. I-

1. Plotted a bar chart for the number of bikes rented per month. I found out that the number of bikes rented are maximum during the summer months and minimum during the winter months.
2. Plotted a bar chart for comparing the number of bikes rented on Weekdays vs Weekends. I found out that demand on Weekdays and Weekends are almost similar, but considering that Weekend consists of two days and Weekday consists of 5 days, I can safely say that per day wise, demand is greater on Weekends as compared to Weekdays.
3. Plotted a pointplot to compare the number of bikes rented per hour on Weekdays vs Weekends. I can infer that during Weekdays, there are two major stakes at 8am and 6 pm indicating that a large number of people commute to their office/place of work and then to their place of accommodation at the two times respectively. While on Weekends, the demand is considerably higher during evening.
4. Plotted a bar chart to find the number of bikes rented during each season and I found that the demand is maximum during summers, followed by autumn, spring and winter.
5. Plotted bar chart to check the demand of bikes during holidays and functioning day and I discovered that on a per day, 715 bikes Ire being rented on non-holidays while 500 bikes Ire rented on holidays, 700 bikes Ire rented each day when it was a functioning day.
6. Plotted regression plots where target(dependent) variable is number of bikes rented and x(independent variables) are “hour”,”temperature”, “wind\_speed”, “visibility”, “dew\_point\_temperature”, “radiation”, “rainfall”, “snowfall”, “month”, “week\_ends\_or\_days”
7. Plotted a line plot for wind speed and average number of bikes rented. I concluded that the number of bikes is maximum when the weather is breezy/windy
8. I plotted a kdeplot for all the independent variables. I found that the values of dependent variable “bike\_count” is heavily right-skewed, “visibility” column is left-skeId indicating that the number of bikes rented increases as visibility increases and “solar\_radiation” column is highly right-skewed indicating that when the solar radiation/heat increases, the number of bikes decreases exponentially.
9. Plotted a boxplot for all numerical variables to find out how the values of these columns are statistically distributed.
10. Plotted a correlation heatmap, to check correlation between dependent and independent variables. I found out that
11. “bike\_count” is highly +vely correlated with “hour”, “temperature”, “dew\_point\_temperature” and “solar\_radiation” columns
12. “hour” column is highly +vely correlated with “wind\_speed” and highly -vely correlated with “humidity”
13. “humidity” is highly +vely correlated with “dew\_point\_temperature” and “rainfall” while highly -vely correlated with “bike\_count”, “hour”, “visibility” & “solar\_radiation”
14. Plotted a pairplot for all columns of dataset with “bike\_count”

**HYPOTHESIS TESTING**

Hypothesis testing is done to confirm our observation about the population using sample data, within the desired confidence level. Through Hypothesis testing, I can determine whether I have enough statistical evidence to conclude if the hypothesis about the population is true or not. H0 represents the Null Hypothesis while H1 represents the Alternate Hypothesis. For our evidence, I use the p-value obtained for each column using OLS(Ordinary Least Squares). I performed 4 such Hypothesis tests on the data given to us.

1. **H0**: Temperature has no effect on the number of bikes rented

**H1**: Temperature has a significant effect on the number of bikes rented.

I found enough evidence to reject the H0 and can conclude that temperature has a significant effect on the number of bikes rented.

1. **H0**: Snowfall has no effect on the number of bikes rented.

**H1**: Snowfall has a significant effect on the number of bikes rented.

I found enough evidence to reject H0 and can conclude that snowfall has a significant effect on the number of bikes rented.

1. **H0**: Dew Point Temperature has no effect on the number of bikes sold

**H1**: Dew Point Temperature has a significant effect on the number of bikes sold.

I couldn’t find enough evidence to reject H0 and hence can conclude that Dew Point Temperature has no effect on the number of bikes sold.

1. **H0**: The dependent variable “bike\_count” is normally distributed.

**H1**: The dependent variable “bike\_count” is not normally distributed.

Here, I performed the Shapiro-Wilk’s Test and found enough evidence to reject H0 and conclude that the dependent variable “bike\_count” is not normally distributed.

**FEATURE ENGINEERING**

The features/variables of dataset have a direct impact on predictive models and their results. The more carefully prepared and chosen the features are, the more accurate the results will be. Feature Engineering is a process that involves transforming raw data into features that more precisely represent the underlying problem for a predictive model. Basically, it helps prepare the data for modelling. Some of the major steps include Categorical Encoding, Feature Manipulation and Selection, Pruning the Outliers and dropping unnecessary columns.

1. Categorical Encoding-
2. For the sake of simplicity of learning for our model, I do one-hot encoding of the categorical columns of our dataset i.e. “holiday”, “functioning\_day” and used pd.get\_dummies() for “seasons” column.
3. After that, I drop the “seasons” and “date” column from our dataframe.
4. Feature Manipulation & Selection-
5. I first find out the skewness of the columns. Anything more than 6 and less than-6 is considered futile or the information obtained from the variable is negligible. To solve this, I use log transformation on the columns “snowfall” and “rainfall” as they have high skewness values; I replace the values 0 by 1 to avoid the infinity error(log 0 = undefined).
6. After creating the new log transformed columns, I safely drop “snowfall” and “rainfall” columns from our dataframe. Now all the skewness values are Ill within the healthy range.
7. For the next step, I defined a function which will return the VIF(Variance Inflation Factor) for each column of our dataframe. Here, if the VIFs are above the value of 6 and are highly correlated with other variables, we can drop them as we don’t want Multicollinearity in our dataframe(MulticollinearitNow y occurs when the dependent variables are highly correlated with each other).
8. First, we remove “dew\_point\_temperature” column because it has a high VIF and high correlation with “temperature” column
9. Next, we remove the columns “visibility”, “humidity”, “day”, “solar\_radiation” and “wind\_speed” because they have a high VIF and a correlation with “temperature” column.
10. Then, remove the columns “Spring”, “Summer” and “Winter” because they have a high correlation with “temperature” column as very high temperatures mostly indicate summers, very low temperatures indicate winters and in between temperatures indicate spring/autumn.

After removing these columns from our dataframe, the VIFs of the columns are well-within the range.

1. In the final step, we calculate the z-score for our columns and we remove the values from these columns where the z-score > 4 indicating outliers.

**Now our data is ready for modelling**

**DATA SPLITTING AND SCALING**

1. We first assign y(dependent variable/target variable) to “bike\_count” and x(independent variable(s)) to rest of the columns.
2. We split the data into training and test datasets using a 70/30 split and use a random\_state value of 8888 for reproducibility.
3. We use MinMaxScaler to scale the values of both training and test dataset so that the values are in between a certain range and contribute equally to the analysis.

**MACHINE LEARNING MODELS IMPLEMENTATION**

Before starting with our models’ implementation, we define four functions-

1. get\_time(name, t) which takes in the name of the model in use and the training time calculated. It also adds the log to a dataframe time\_df , which will contain the log of training time of each model we use and the fine-tuned version of it.
2. get\_metrics(x, y, z) which takes in the actual values(x), the predicted values(y) and the name of the model(z) and returns the metrics Mean Squared Error(MSE), Root-Mean Squared Error(RMSE), Mean Absolute Error(MAE) and R2-Score(r2) and prints them. It also logs the metrics for each model in a dataframe results\_df
3. plot\_scatter(x, y) which takes in the actual values(x) and the predicted values(y) and returns the corresponding scatter plot.
4. plot\_distplot(x, y) which returns the distplot for x = x – y.

We also define a train\_df and test\_df, which contains the metrics for each model during their training phase and prediction phase respectively in a separate dataframe.

**HYPERPARAMETER TUNING METHOD USED:**

**GridSearchCV():- GridSearchCV is a technique used for hyperparameter tuning, where the goal is to select the optimal set of hyperparameters for a machine learning model. It exhaustively searches through a specified parameter grid, evaluating the performance of the model with each combination of hyperparameters using cross-validation.**

**Objective function: maximize CV score**

* **CV score: The cross-validated performance metric used to evaluate each combination of hyperparameters. Common performance metrics include accuracy, precision, recall, F1-score, or mean squared error, depending on the task (classification or regression) and the specific problem.**

**Components of GridSearchCV:**

* **Parameter Grid: A dictionary or list of dictionaries specifying the hyperparameters and their corresponding values to be tuned. Each key in the dictionary represents a hyperparameter, and the corresponding value is a list of values to be tried.**
* **Cross-Validation: GridSearchCV uses cross-validation to evaluate the performance of each hyperparameter combination. The dataset is split into k folds, and the model is trained and evaluated k times, each time using a different fold for validation and the remaining folds for training.**
* **Scoring: The scoring parameter specifies the performance metric to optimize during grid search. It can be a string (e.g., 'accuracy' for classification tasks, 'neg\_mean\_squared\_error' for regression tasks) or a callable function.**
* **Grid Search: GridSearchCV performs an exhaustive search over all possible combinations of hyperparameters specified in the parameter grid. For each combination, it trains the model using cross-validation and computes the average performance metric across all folds.**
* **Best Hyperparameters: After grid search is complete, GridSearchCV selects the combination of hyperparameters that yielded the highest cross-validated score as the best set of hyperparameters.**
* **Best Estimator: GridSearchCV provides access to the best estimator (i.e., the model trained with the best hyperparameters), which can be used for making predictions on new data.**

**EVALUATION METRICS USED**

1. **Mean Squared Error (MSE):**

**Mean Squared Error (MSE) is a commonly used metric for evaluating the performance of regression models. It measures the average squared difference between the predicted values and the actual values.**

**Formula:**

**,**

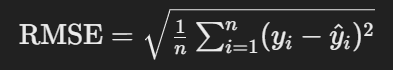
**Where:**

* + **𝑛 is the number of samples.**
  + **𝑦𝑖​ is the actual value of the target variable for the 𝑖𝑡ℎ sample.**
  + **𝑦^𝑖is the predicted value of the target variable for the 𝑖𝑡ℎ sample.**

1. **Root Mean Squared Error (RMSE):**

**Root Mean Squared Error (RMSE) is the square root of the MSE. It provides a measure of the average magnitude of the errors in the predictions, in the same units as the target variable.**

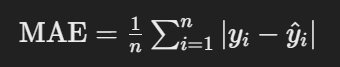
**Formula:**

****

1. **Mean Absolute Error (MAE):**

**Mean Absolute Error (MAE) is another metric for evaluating the performance of regression models. It measures the average absolute difference between the predicted values and the actual values.**

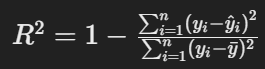
**Formula:**

****

1. **Coefficient of Determination (𝑅2 Score):**

**The coefficient of determination, often denoted as 𝑅2 (R-squared), is a measure of how well the regression model fits the observed data. It represents the proportion of variance in the dependent variable that is explained by the independent variables.**

**Formula:**

****

**Where:**

* + **𝑦ˉ​ is the mean of the observed values (𝑦𝑖).**

**Model 1: LINEAR REGRESSION**

**Description:**

**Linear regression is a fundamental statistical method used for modeling the relationship between a dependent variable (often denoted as "y") and one or more independent variables (often denoted as "x"). The relationship is assumed to be linear, hence the name "linear regression."**

**Here's a breakdown of its key components:**

1. **Dependent variable (y): This is the variable we want to predict or explain. It's the outcome or response variable.**
2. **Independent variable(s) (x): These are the variables that are used to predict the dependent variable. In simple linear regression, there's only one independent variable, but in multiple linear regression, there can be multiple independent variables.**
3. **Linear relationship: Linear regression assumes that there's a linear relationship between the independent variable(s) and the dependent variable. This means that a change in the independent variable(s) is associated with a proportional change in the dependent variable.**
4. **Regression equation: The relationship between the independent and dependent variables is expressed through a linear equation of the form:**

**𝑦=𝛽0+𝛽1𝑥1+𝛽2𝑥2+...+𝜀**

**Where:**

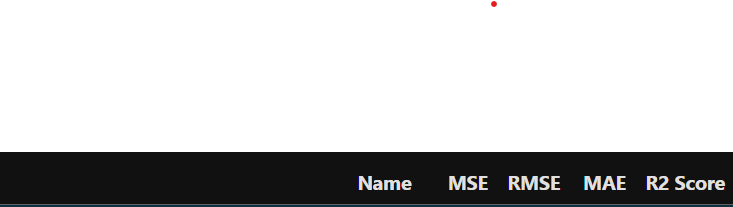
* + **𝑦*y* is the dependent variable.**
  + **𝑥1, 𝑥2, ... are the independent variables.**
  + **𝛽0, 𝛽1, 𝛽2, ... are the coefficients (slope parameters) that represent the effect of the independent variables on the dependent variable.**
  + **𝜀*ε* is the error term, representing the difference between the observed and predicted values of the dependent variable.**

1. **Ordinary Least Squares (OLS): The most common method for estimating the coefficients in the regression equation is Ordinary Least Squares. It finds the line that minimizes the sum of the squared differences between the observed and predicted values of the dependent variable.**

**Linear regression is widely used in various fields such as economics, finance, social sciences, and machine learning for tasks such as prediction, forecasting, and understanding the relationship between variables.**

**Steps:**

1. We fit the Linear Regression model and print the regression coefficients for each variable. But we found that even though I scaled the inputs, I am getting very high regression coefficients. This may be possible because of the high values in y variable. So, I decide to further standardize the values by taking square root of the values using np.sqrt(). I define the x and y variables again but this time with the square root values.
2. I again run the algorithm and this time the coefficient values are much smaller and in range. We got a regression score of around 0.55 indicating that my model could explain only around 55% of the variance of our dataset.
3. We use get\_metrics() function to log the metrics for training and test dataset respectively in the train\_df and test\_df. We use plot\_scatter() and plot\_distplot() to check the distribution of actual and predicted values.



**Model 2: LASSO REGRESSION**

**Description:**

1. **Lasso Regression (Least Absolute Shrinkage and Selection Operator):**

**Lasso regression is a linear regression technique that introduces a penalty term to the ordinary least squares objective. The objective function for Lasso regression is given by:**

**minimize:**

**,**

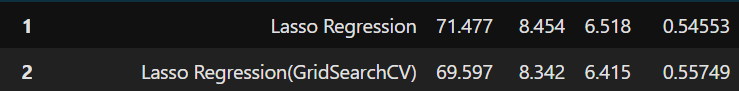
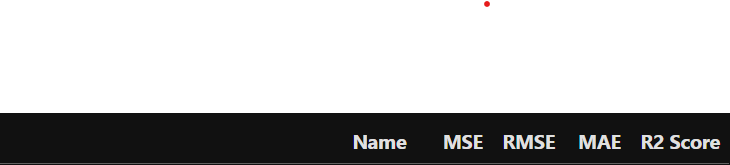
**Where:**

* + **𝑦𝑖is the observed value of the dependent variable for the 𝑖𝑡ℎ*ith* observation.**
  + **𝑦^𝑖​ is the predicted value of the dependent variable for the 𝑖𝑡ℎ*ith* observation.**
  + **𝛽𝑗​ is the coefficient of the 𝑗𝑡ℎ independent variable.**
  + **𝛼 is the regularization parameter that controls the strength of the penalty term.**

**Lasso regression encourages sparse solutions by shrinking the coefficients of less important features to zero, effectively performing feature selection.**

**Steps:**

1. We fit the Lasso Regression model and print the metrics using get\_metrics().
2. For the test dataset, R2 score is 0.5455 meaning the model is able to capture 54.55% variance of our dataframe.
3. We use plot\_scatter() and plot\_distplot() functions to have a look at the actual vs predicted values.
4. In the next step, we use hyperparameter tuning, hoping the model’s performance will increase. We use GridSearchCV to go through the different values of parameter alpha. Again fitting Lasso model with the best paramters obtained by GridSearch, and we got a R2 score of 0.55478 indicating model is able to capture 55.48% of variance which is an improvement over the non-tuned Lasso model but not enough.



**Model 3: RIDGE REGRESSION**

**Description:**

**Ridge regression is a linear regression technique that also adds a penalty term to the ordinary least squares objective. The objective function for Ridge regression is given by:**

**minimize**

**,**

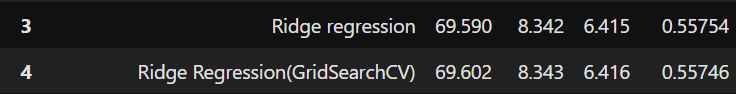
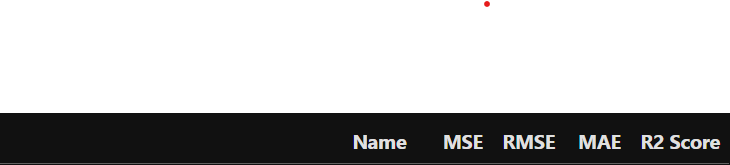
**Where:**

* ***n* is the number of observations.**
* **𝑦𝑖 is the observed value of the dependent variable for the 𝑖𝑡ℎ observation.**
* **𝑦^𝑖​ is the predicted value of the dependent variable for the 𝑖𝑡ℎ observation.**
* **p is the number of independent variables.**
* **𝛽𝑗 is the coefficient of the 𝑗𝑡ℎ independent variable.**
* **𝛼 is the regularization parameter that controls the strength of the penalty term.**

**Ridge regression adds a penalty term that is the sum of the squared values of the coefficients multiplied by a constant (alpha). Unlike Lasso regression, Ridge regression generally does not reduce coefficients to exactly zero, but it helps to mitigate multicollinearity.**

**Steps:**

1. We fit the Ridge Regression model and print the metrics using get\_metrics().
2. For the test dataset, we got a R2 score of 0.5575
3. We use plot\_scatter() and plot\_distplot() functions to have a look at the actual vs predicted values.
4. Now, time for tuning the hyperparameters. Again, we use GridSearchCV, to go through the different parameter values. We fit the Ridge Regression model with the best hyperparamter values obtained and got a R2 score of 0.55746 which is less than we got in our non-tuned model.



**Model 4: ELASTIC NET**

**Description:**

**Elastic Net regression combines the penalties of both Lasso and Ridge regression. The objective function for Elastic Net regression is given by:**

**minimize**

**,**

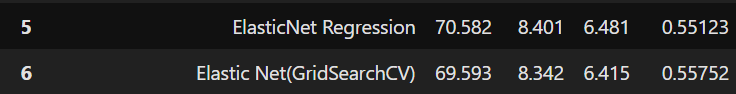
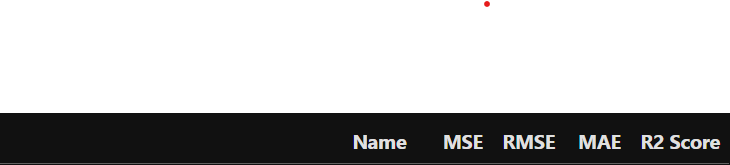
**Where:**

* ***n* is the number of observations.**
* **𝑦𝑖 is the observed value of the dependent variable for the 𝑖𝑡ℎ observation.**
* **𝑦^𝑖​ is the predicted value of the dependent variable for the 𝑖𝑡ℎ observation.**
* **α is the regularization parameter that controls the overall strength of regularization.**
* **𝜌 is a parameter that controls the balance between L1 and L2 penalties.**
* **𝛽𝑗 is the coefficient of the 𝑗𝑡ℎ independent variable.**

**Elastic Net regression adds two penalty terms: one that is a combination of the absolute values of the coefficients (L1 penalty) and another that is a combination of the squared values of the coefficients (L2 penalty).**

**Steps:**

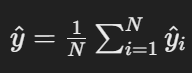
1. We fit the ElasticNet model with parameters alpha = 0.01, l1\_ratio = 0.5 and print the metrics using get\_metrics().
2. For the test dataset, we got an R2 score of 0.5512.
3. We use plot\_scatter() and plot\_distplot() functions to have a look at the actual vs predicted values.
4. For hyperparameter tuning, we use GridSearchCV with many values of alpha and fit the ElasticNet model with the best hyperparameters obtained. We got a R2 score of 0.5518 which is only a little better than our non-tuned model



**Model 5: RANDOM FOREST REGRESSOR**

**Description:**

**Random Forest is an ensemble learning method that constructs a multitude of decision trees during training. Each decision tree is trained on a random subset of the training data and a random subset of features, a process known as bagging (bootstrap aggregating). The predicted value in Random Forest regression is the average prediction of all the individual trees:**

**,**

**Where:**

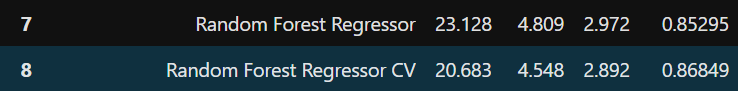
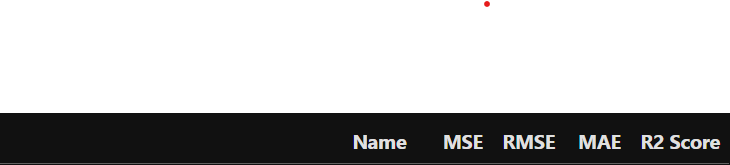
* **𝑁 is the number of decision trees in the Random Forest.**
* **𝑦^𝑖​ is the prediction of the 𝑖𝑡ℎ decision tree.**

**Components of Random Forest Regressor:**

* **Decision Trees: The base learners in the ensemble are decision trees, which are simple models that make predictions based on a series of binary splits.**
* **Bagging: Random Forest employs bagging to create diverse decision trees by training each tree on a random subset of the training data with replacement.**
* **Random Feature Selection: At each split in the decision tree, a random subset of features is considered, reducing the correlation between trees and improving generalization.**
* **Averaging: The final prediction in Random Forest regression is the average prediction of all the individual trees, which helps to smooth out predictions and reduce overfitting.**

**Steps:**

1. We fit the RandomForestRegressor model with random\_state = 101 (for reproducibility) and print the metrics using get\_metrics().
2. For the test dataset, R2 score is 0.8529 which means the model is able to capture around 85.29% of the variance, which is a great improvement over our previous models.
3. We use plot\_scatter() and plot\_distplot() functions to have a look at the actual vs predicted values.
4. For tuning the parameters, we use GridSearchCV to go through the parameter grid values of “n\_estimators”, “max\_features” and “max\_depth”. We use a large range of values for each parameter and after trial and testing we found the optimal value of each of these hyperparameter which we use to fit the RandomForestRegressor model again. Now, we got a R2 score of 0.8685, which is a little better than the non-tuned model
5. Using model\_name.feature\_importances\_ we print the relative feature importances of each variable and we found that “temperature” is the most important feature followed by “Hour”, “functioning\_day” and “month” respectively.



**Model 6: GRADIENT BOOSTING REGRESSOR**

**Description:**

**Gradient Boosting is an ensemble learning method that builds an ensemble of weak learners (typically decision trees) sequentially. The predicted value in Gradient Boosting regression is the sum of predictions from all the individual trees, weighted by a learning rate or shrinkage parameter:**

**,**

**Where:**

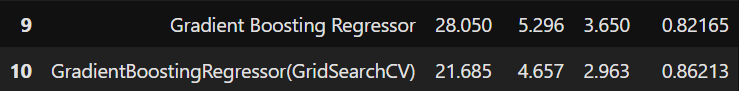
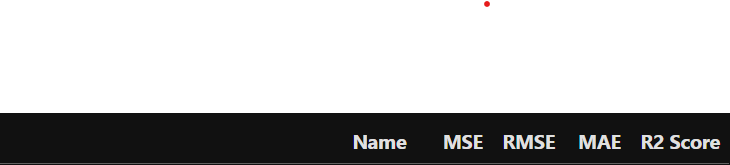
* **𝑁 is the number of weak learners in the ensemble.**
* **𝛾𝑖​ is the learning rate or shrinkage parameter for the 𝑖𝑡ℎ weak learner.**
* **𝑓𝑖(𝑥) is the prediction of the 𝑖𝑡ℎ weak learner.**

**Components of Gradient Boosting Regressor:**

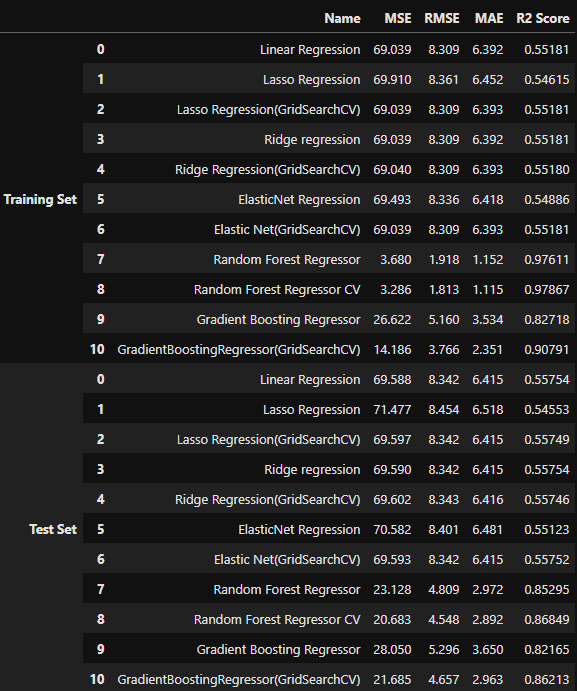
* **Weak Learners: The base learners in the ensemble are typically shallow decision trees, often referred to as weak learners, which are combined to create a strong predictor.**
* **Sequential Learning: Unlike Random Forest, where trees are built independently, in Gradient Boosting, each new tree in the ensemble corrects the errors made by the previous trees, improving the model iteratively.**
* **Gradient Descent: Gradient Boosting minimizes a loss function, such as mean squared error, by iteratively fitting new models to the negative gradient of the loss function.**
* **Regularization: Gradient Boosting models often include regularization techniques, such as shrinkage (learning rate) and tree depth constraints, to prevent overfitting and improve generalization performance.**

**Steps:**

1. We fit the GradientBoostingRegressor model and print the metrics using get\_metrics().
2. For the test dataset, we scored a R2 score of 0.8217 which is better than most of the models but less than random forest regressor model.
3. We use plot\_scatter() and plot\_distplot() functions to have a look at the actual vs predicted values.
4. For hyperparameter tuning, we again use GridSearchCV to go through the different values of hyperparameters “learning\_rate”, “n\_estimators” and “max\_depth”. We used hit and trial method for a large range of values and found the most optimal values for each hyperparameter which we use to fit the GradientBoostingRegressor model again. This time, we got a R2 score of 0.8621, which is better than the non-tuned model but less than the RandomForestRegressor.
5. We use model\_name.feature\_importances\_ to find out the relatively most important features which are “temperature”, “hour”, “functioning\_day” and “log\_rainfall” respectively.



**THE EVALUATION METRICS FOR BOTH TRAINING AND TESTING DATASETS TOGETHER:**



**Conclusion:**

After having a look at the R2 scores of all the models, I use the fine-tuned Random Forest Regressor model as it had the lowest Root Mean Squared Error, Mean Absolute Error and the highest R2 score among all the models.