**Bike Sharing Demand Prediction**

**(Supervised Machine Learning Regression)**

**By- N. SANTOSH KUMAR CHOUDHURY,**

**KURESH CHANDRA TRIPATHY**

**Data science trainees,**

**Alma Better, Bangalore**

# **Bike Sharing Demand Prediction (Supervised Machine Learning Regression)**

## **Objective:-**

We are provided with a Rental bike’s dataset name “**SeoulBikeData.csv**”. Our main objective is performing Supervised Machine Learning Regression on the given dataset and draw useful conclusions about the prediction of bike count required at each hour for the stable supply of rental bikes.

**Data Summary: -**

We are given a Rental bike’s dataset. This dataset contains.

It contains the following features.

1. Date - year-month-day

2. Rented Bike count - Count of bikes rented at each hour

3. Hour - Hour of the day

4. Temperature-Temperature in Celsius

5. Humidity - %

6. Wind Speed - m/s

7. Visibility - 10m

8. Dew point temperature - Celsius

9. Solar radiation - MJ/m2

10. Rainfall – mm

11. Seasons

12. Holiday

13. Functioning Day

* Total number of rows in data: 8760
* Total number of columns: 14

## **Data Cleaning and Feature Engineering**

### As we know clearing data will remove the conflict between any duplicate data/row.

1. Missing values finding and replacing with proper values.
2. Change required column data types from Object to appropriate datatypes.
3. Checking duplicate rows in dataset.
4. Adding some required new columns.

* Date column is converted from object to date type data.
* Date into Year, Month, Day column.
* Checking the Null value
* Checking Duplicate rows in Bike Data.

## **Exploratory Data Analysis**

* An EDA is a thorough examination meant to uncover the underlying structure of a data set and is important for a company because it exposes trends, patterns, and relationships that are not readily apparent.

Mainly performed using Matplotlib and Seaborn library and the following graph and plots had been used:

* Bar Plot.
* Histogram.
* Scatter Plot.
* Line Plot.
* Heatmap.
* Box Plot

**Analysis of Independent variable w.r.t Dependent variable**

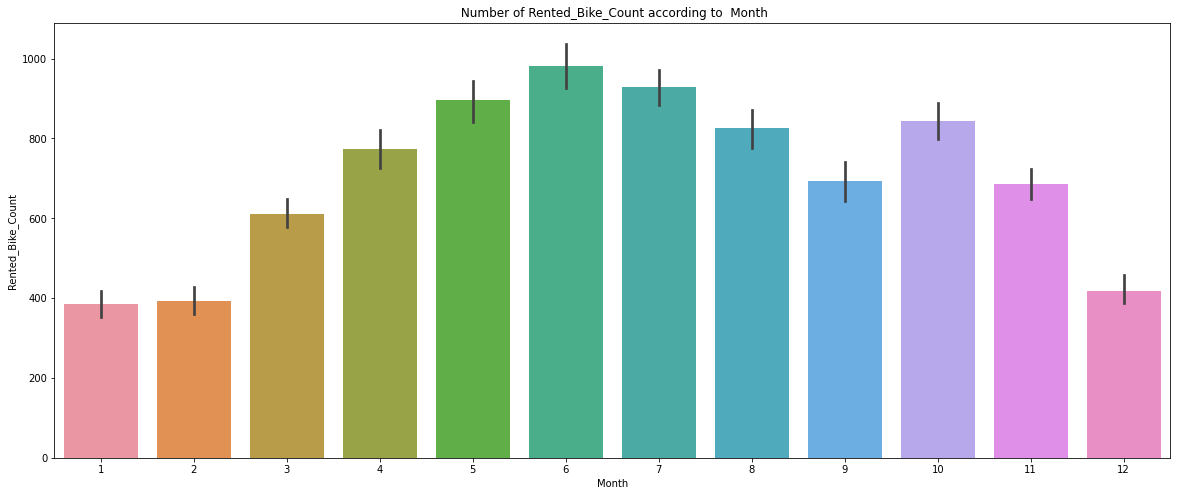
* 1.Categorical variables
* 2.Numerical variables

**Categorical variables**

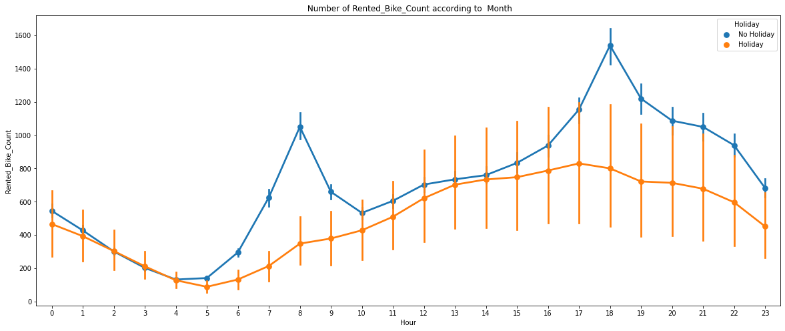
**What is a categorical variables in data analysis?**

* A categorical variable (sometimes called a nominal variable) is one that has two or more categories, but there is no intrinsic ordering to the categories.
* Our dependent variable is "Rented Bike Count" so we need to analysis this column with the other columns by using some visualization plot. First, we analyze the category data type then we proceed with the numerical data type.

## **1. Rented bike vs Month**

* From the above bar plot is clearly showing that from Jan to Jun demand for the rented bike is growing high and then slowly the demand start dropping till Sep and then again, a small demand increase for some a month and then it decreases.
* 

# **Rented bikes demand w.r.t hour on weekdays and weekends**

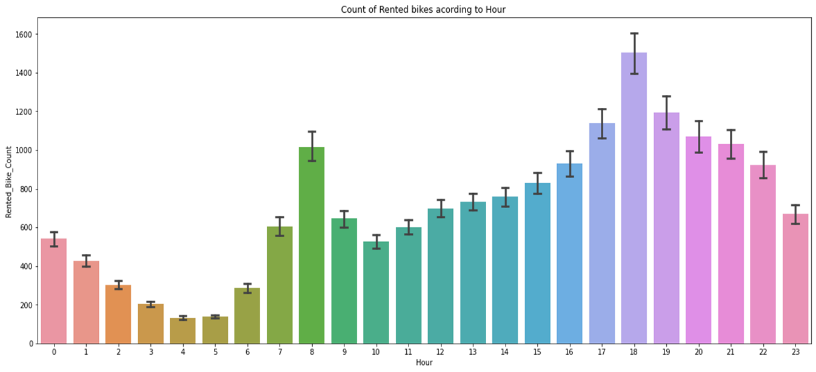


# From this point plot graph, we can see that on holiday ranted bike demand is less as compared to No-holiday. Between 7am-9am and 5pm-7pm demand of rented bike is high. Which Country have Most Babies During their Visit.

# **Rented** bikes demand w.r.t Season

* Form the bar chart and point plot we can clearly say that in summer demand is high as compared to other season and in winter the demand is lowest.
* Point chart also tell that in morning from 7-9am and 5-7pm demand is high in every season.

# **Rented** bikes demand w.r.t hour on weekdays and weekends



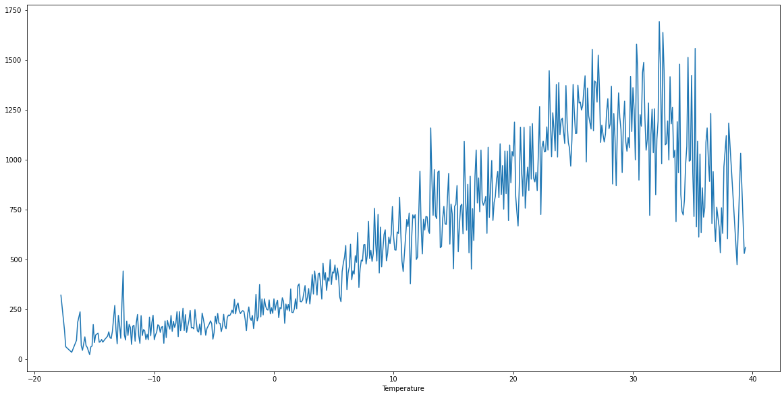
# From this point plot graph, we can see that on holiday ranted bike demand is less as compared to No-holiday. Between 7am-9am and 5pm-7pm demand of rented bike is high and in between 3-6 am it is low.

**Numerical variables**

**What is a categorical variables in data analysis?**

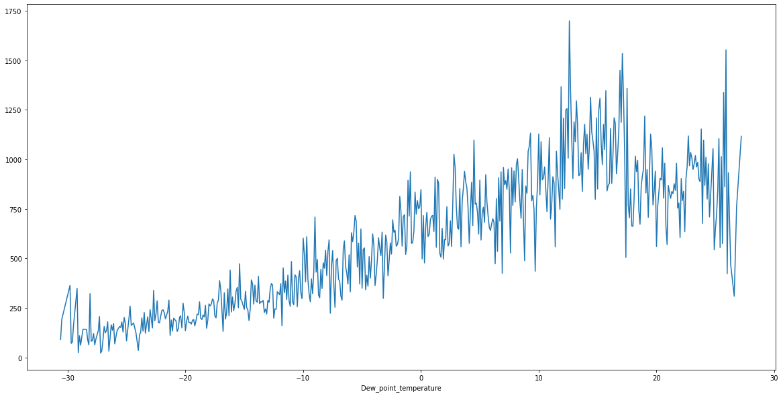
* A numeric variable (also called quantitative variable) is a quantifiable characteristic whose values are numbers.
* A numeric variable (also called quantitative variable) is a quantifiable characteristic whose values are numbers.

# **Relationship between “Rented\_Bike\_Count” and “Temperature”**



* In above plot the demand of rented bike is highest when temperature is around 20°C-30°C.

# Relationship between “Rented\_Bike\_Count” and “Dew\_point\_temperature”



# This plot the demand of rented bike is highest when dew point temperature is around 20°C-30°C.

# This graph also follows same trend as temperature.

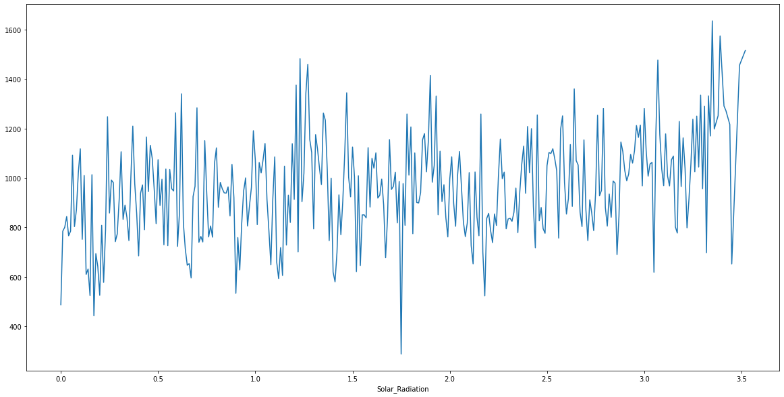
# Relationship between “Rented\_Bike\_Count” and “Wind\_speed”

Chart, line chart

Description automatically generated

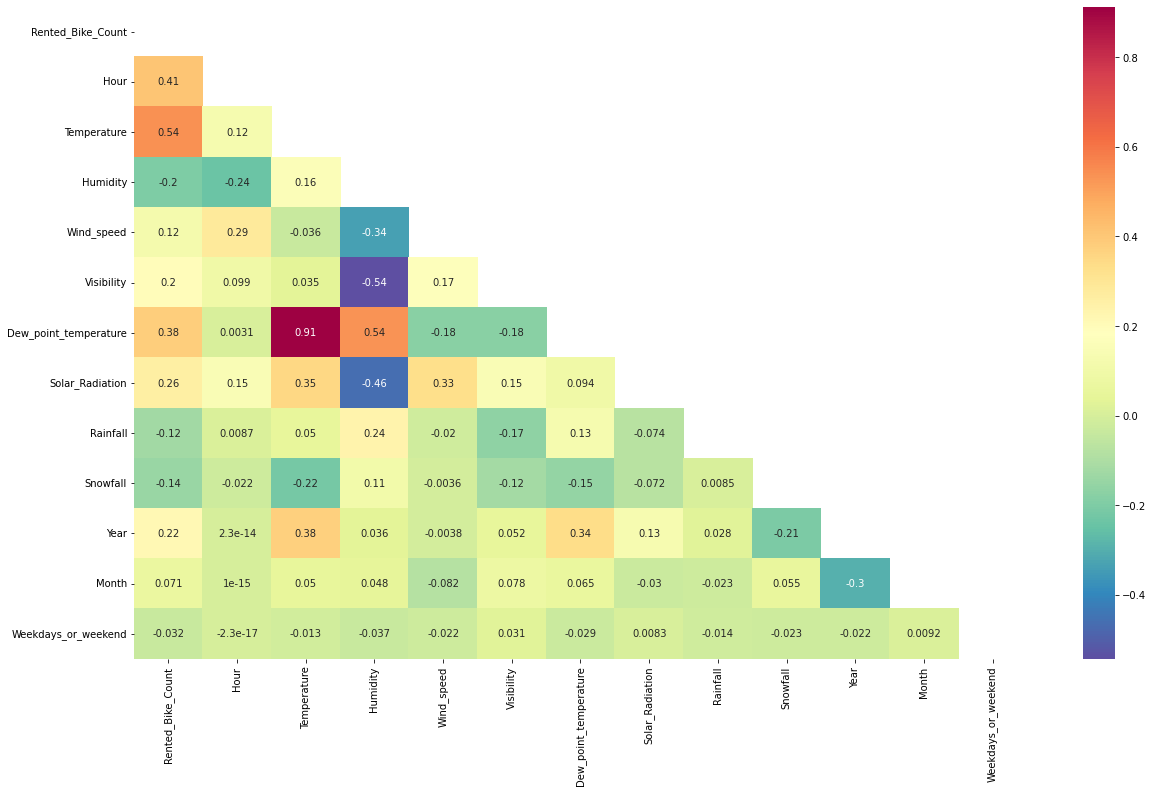
* In this plot the demand of rented bike is Same throughout the wind speed/ uniformly distributed but when the wind speed is (5-6) m/s demand rises
* When the wind speed is more than 7 m/s then demand for bike is also increase.
* Here we can say that the patten is very random.

# Relationship between “Rented\_Bike\_Count” and “Solar\_Radiation”



* In this plot the demand of rented bike is high if there is solar radiation. Mostly the number is around 1000(+/-200).

**Regression plot**

* 1. It is a plot give a visual guide that helps to emphasize patterns in a dataset between two parameters.
* 2. Regression lines can be used as a way of visually the liner relationship between,
* the Independent (x)----->Numerical variables and Dependent variables(y)----->Rented\_Bike\_Count variables here in the graphs.
* **Checking of Correlation between variables**
* ****
* We can observe on the heatmap that on the target variable line the most positively correlated variables to the rent are:
* Temperature
* Dew Point temperature
* Solar radiation
* Most negatively correlated variables are:
* Humidity
* Rainfall
* From the above graph we can see that relation between columns 'Temperature' and 'Dew point temperature' i.e 0.91 so we if we drop this column than it will not affect the outcome of our analysis.
* We can drop the column 'Dew point temperature'.

**Model Training & Testing**

* Assigning the dependent and independent variables.
* Splitting the model into train and test sets.
* Transforming data using min-maxscaler.
* Fitting linear regression on train set.
* Getting the predicted dependent variable values from the model.
* The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications. Before fitting any model, it a general rule that we have to split the dataset into a training and testing set. This dataset split in proportions of the data go into the training model and some proportion will go for evaluation how our model performs on any unseen data.
* Generally, the proportion may vary from 70:30,80:20,75:25 depending on the person. Mostly we prefer 80:20 ratio for the training and testing respectively.

**Code used to Analysis Booking**

* #Assign the value in X for independent variable and Y dependent variable.
* X = bike\_df\_copy.drop(columns=['Rented\_Bike\_ Count'], axis=1)

X.head()

* #Assign the value in Y fop dependent variable/target value

y = np.sqrt(bike\_df\_copy['Rented\_Bike\_Count'])

y.head()

bike\_df\_copy.describe().columns

# **LINEAR REGRESSION**

* Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.
* #import the packages and do LinearRegression from sklearn.linear\_model import LinearRegression

reg= LinearRegression().fit(X\_train, y\_train)

* #Checking the score

reg.score(X\_train, y\_train)

## **Training & Testing dataset summary**

***Training dataset summary****: -*

**MSE: 52.98715355670106**

**RMSE: 7.2792275384618295**

**MAE: 5.5688420275877135**

**R2: 0.6566702426180293**

## **Adjusted R2: 0.6533042646044807** ***Testing dataset summary****: -*

**MSE: 54.15503807932429**

**RMSE: 7.35901067259209**

**MAE: 5.652187638863334**

**R2: 0.6561284835266044**

# **Adjusted R2: 0.6527571941494141**

# **LASSO REGRESSION**

* Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage.
* Shrinkage is where data values are shrunk towards a central point as the mean.
* Lasso regression technique also called as 'L1 regularization technique'.
* #Import the packages and do Lasso Regression

from sklearn.linear\_model import Lasso

lasso = Lasso(alpha=1.0, max\_iter=3000)

* # Create the model score print(lasso.score(X\_test, y\_test), lasso.score(X\_train, y\_train))

## **Training & Testing dataset summary**

***Training dataset summary****: -*

**MSE: 52.98715355670106**

**RMSE: 7.2792275384618295**

**MAE: 5.5688420275877135**

**R2: 0.6566702426180293**

**Adjusted R2: 0.6533042646044807**

## ***Testing dataset summary****: -*

**MSE: 54.15503807932429**

**RMSE: 7.35901067259209**

**MAE: 5.652187638863334**

**R2: 0.6561284835266044**

# **Adjusted R2: 0.6527571941494141**

# **RIDGE REGRESSION**

* Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated.
* #import the packages and do LinearRegression from sklearn.linear\_model import LinearRegression

reg= LinearRegression().fit(X\_train, y\_train)

* #Checking the score

reg.score(X\_train, y\_train)

## **Training & Testing dataset summary**

***Training dataset summary****: -*

**MSE: 52.98716055296455**

**RMSE: 7.279228019025407**

**MAE: 5.568867766574528**

**R2: 0.6566701972858067**

**Adjusted R2: 0.6533042188278244**

## ***Testing dataset summary****: -*

**MSE: 54.15553815282732**

**RMSE: 7.359044649465535**

**MAE: 5.6522810783829645**

**R2: 0.6561253081796756**

**Adjusted R2: 0.6527539876716333**

# **ELASTIC NET REGRESSIN**

* Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models.
* #Import the packages

from sklearn.linear\_model import ElasticNet

elasticnet = ElasticNet(alpha=0.1, l1\_ratio=0.5)

#Fit The Model

elasticnet.fit(X\_train,y\_train)

* #Check the score

elasticnet.score(X\_train, y\_train)

## **Training & Testing dataset summary**

***Training dataset summary****: -*

**MSE: 64.29451208107879**

**RMSE: 8.018385877536625**

**MAE: 6.0926390605127025**

**R2: 0.5834043206308951**

## **Adjusted R2: 0.5793200492645314**

## ***Testing dataset summary****: -*

**MSE: 67.91452871048334**

**RMSE: 8.24102716355694**

**MAE: 6.2774700651873765**

**R2: 0.5687590147376134**

**Adjusted R2: 0.5645311619409233**

**DECISION TREE**

* A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.
* #import the packages

from sklearn.tree import DecisionTreeRegressor

decision\_regressor = DecisionTreeRegressor(criterion='mse', max\_depth=8, max\_features = 9,  max\_leaf\_nodes=100,)

#Fit The Model

decision\_regressor.fit(X\_train, y\_train)

## **Training & Testing dataset summary**

***Training dataset summary****: -*

**Model Score: 0.8318783076277656**

**MSE: 25.94674576963378**

**RMSE: 5.093794829950827**

**MAE: 3.5545568887680794**

**R2: 0.8318783076277656**

**Adjusted R2: 0.8302300557417633**

***Testing dataset summary****: -*

**MSE: 32.35470890253302**

**RMSE: 5.688119979618311**

**MAE: 3.972700956412215**

**R2: 0.7945553505276359**

**Adjusted R2: 0.7925411872975**

**RANDOM FOREST**

* The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.
* #import the packages

from sklearn.ensemble import RandomForestRegressor

* # Create an instance of the RandomForestRegressor

rf\_model = RandomForestRegressor()

rf\_model.fit(X\_train,y\_train)

* #Check the score

print("Model Score:",rf\_model.score(X\_train,y\_train))

## **Training & Testing dataset summary**

***Training dataset summary****: -*

**Model Score: 0.9887201710487845**

**MSE: 1.7408512250408463**

**RMSE: 1.3194132123943758**

**MAE: 0.8515986122974984**

**R2: 0.9887201710487845**

**Adjusted R2: 0.9886095844904392**

***Testing dataset summary****: -*

**MSE: 12.586329973345695**

**RMSE: 3.5477218004440108**

**MAE: 2.335333977552613**

**R2: 0.920079820303527**

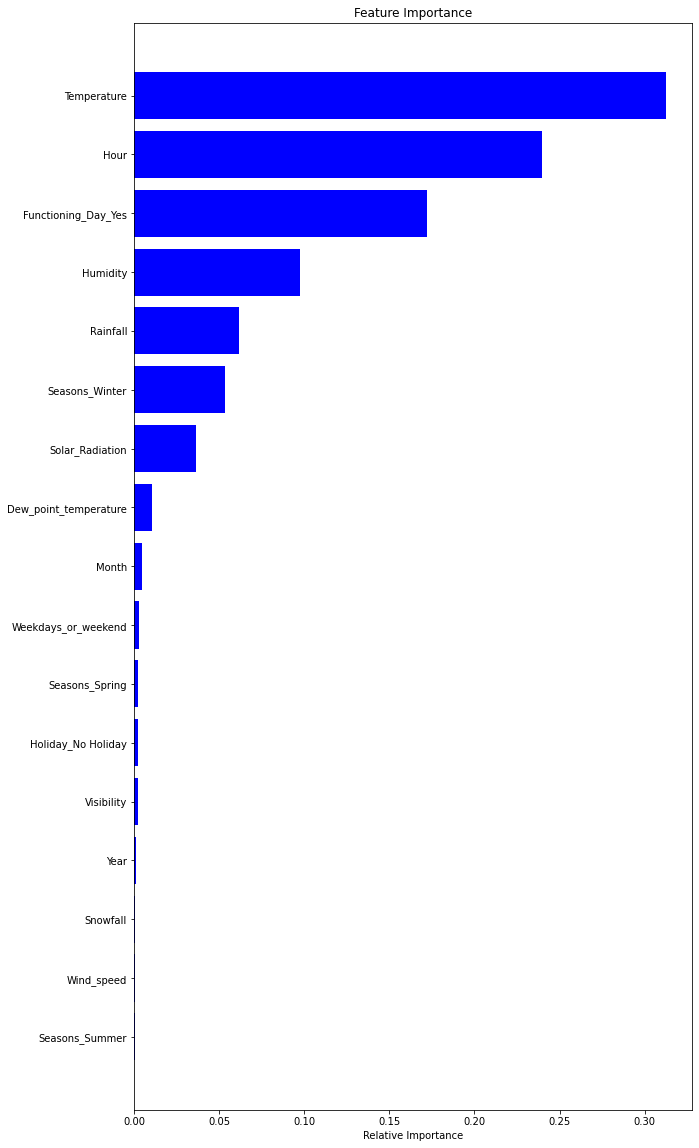
**Adjusted R2: 0.9192962891300321**

**UNDERSTANDING THE IMPORTANCE OF THE FEATURES**

# importance of the features

rf\_model.feature\_importances\_

**GRADIENT BOOSTING**

* Gradient boosting is a method standing out for its prediction speed and accuracy, particularly with large and complex. datasets.
* 
* #import the packages

from sklearn.ensemble import GradientBoostingRegressor

# Create an instance of the GradientBoostingRegressor

gb\_model = GradientBoostingRegressor()

gb\_model.fit(X\_train,y\_train)

* #Check the score

print("Model Score:",gb\_model.score(X\_train,y\_train))

## **Training & Testing dataset summary**

***Training dataset summary****: -*

**Model Score: 0.8906247080946529**

**MSE : 16.88023034091356**

**RMSE : 4.108555748789781**

**MAE: 2.9516149687601465**

**R2: 0.8906247080946529**

## **Adjusted R2: 0.8895524013112671**

## ***Testing dataset summary****: -*

**MSE: 19.34969054848829**

**RMSE: 4.398828315414036**

**MAE: 3.1317719226265672**

**R2: 0.8771341011254881**

**Adjusted R2: 0.8759295334894635**

**HYPERPARAMETER TUNING**

## Hyperparameter tuning (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance.

* It works by running multiple trials in a single training process. Each trial is a complete execution of your training application with values for your chosen hyperparameters, set within the limits you specify.
* This process once finished will give you the set of hyperparameter values that are best suited for the model to give optimal results.
* There are two type way to tuning hyperparameter

## GridSearchCV

## RandomizedSearchCV

## **Gradient Boosting Regressor with GridSearchCV**

***Training dataset summary****: -*

**Model Score: 0.9560877024082052**

**MSE: 6.777122010240805**

**RMSE: 2.603290611944968**

**MAE: 1.7487400972558536**

**R2: 0.9560877024082052**

## **Adjusted R2: 0.955657189686717**

## ***Testing dataset summary****: -*

**MSE: 12.26168802923979**

**RMSE: 3.5016693203727547**

**MAE: 2.3409379430228046**

**R2: 0.9221412188656893**

**Adjusted R2: 0.9213778974820196**

* #Plot the figure

plt.figure(figsize=(10,20))

plt.title('Feature Importance')

plt.barh(range(len(indices)), importances[indices], color='blue', align='center')

plt.yticks(range(len(indices)), [features[i] for i in indices])

plt.xlabel('Relative Importance')

plt.show()

* #Showing top 5 features

importance\_df.head()

Feature Feature Importance

0 Temperature 0.33

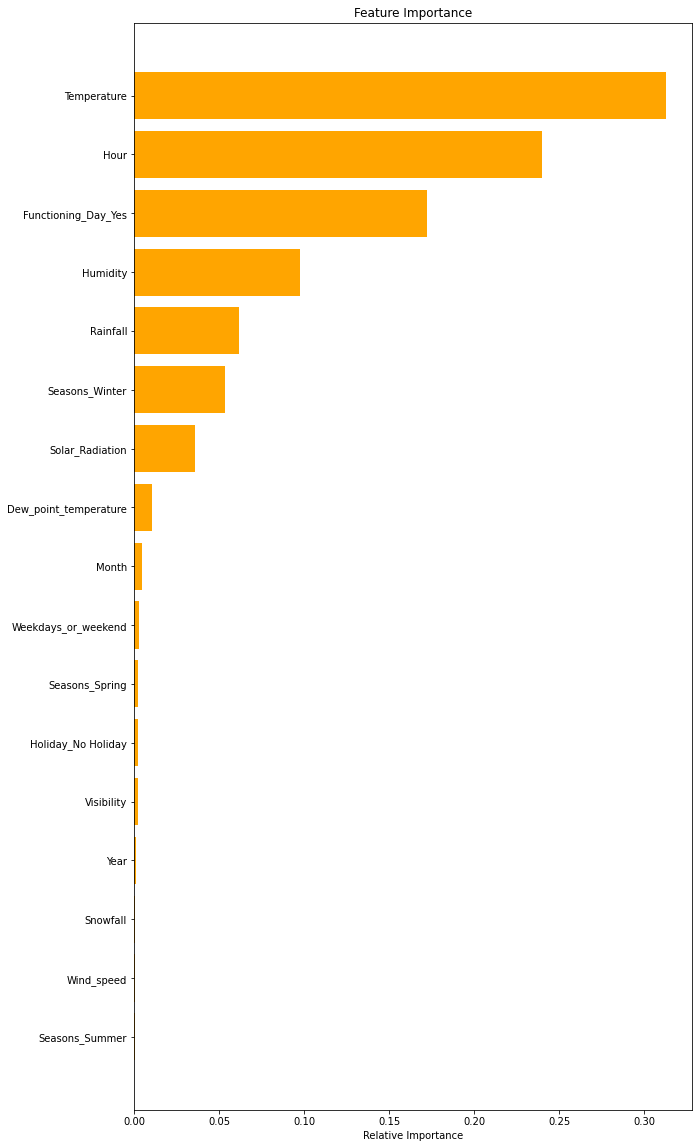
1 Humidity 0.15

2 Wind\_speed 0.01

3 Visibility 0.01

4 Solar\_Radiation 0.03

**(Showing all features with importance in graph)**

****

# **Conclusion: -**

* Random forest Regressor and Gradient Boosting gridsearchcv gives the highest R2 score of 99% and 90% respectively for Train Set and 92% for Test set.
* Feature Importance value for Random Forest and Gradient Boost are different.
* We can deploy this model.

**Challenges: -**

1. Data was present in wrong datatype format.
2. Choosing appropriate visualization techniques to use was difficult.
3. Approach for regression process was time consuming and little bit hard to understand the process at initial stage of work.
4. Selecting the appropriate models to maximize the accuracy of our predictions was one of the challenges faced.

.