BIKE SHARING DEMAND PREDICTION

PROJECT 1

By:
Priyanka
Chandramohan

1. Introduction

Prediction of Bike Sharing demand can help bike sharing companies to allocate bikes better and ensure a more sufficient circulation of bikes for customers. This model proposes a real time method for predicting bike renting based on historical data,weather data and time data. This demand prediction model can provide a significant theortical basis for management strategies and vehicle scheduling in public bike rental system.

Business Problem we are trying to Solve:

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 the 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.

Main Goal:

- Create an analytical framework to understand **Key factors impacting Bike Counting**
- Develop a modeling framework

Prediction of Bike Sharing demand can help bike sharing companies to allocate bikes better and ensure a more sufficient circulation of bikes for customers.

Variable Description:

- Date : year-month-day
- Rented Bike count - Count of bikes rented at each hour
- Dew point temperature - Celsius
- Solar radiation - MJ/m2
- Rainfall - mm
- Snowfall - cm
- Seasons - Winter, Spring, Summer, Autumn
- Holiday - Holiday/No holiday
- Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours)
- Hour - Hour of he day
- Temperature-Temperature in Celsius
- Humidity - %
- Windspeed - m/s
- Visibility - 10m

This is what the dataset looks like,

Table1 - Dataset

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	Winter	No Holiday	Yes
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	Winter	No Holiday	Yes

The original dataset has 14 columns and 8760 rows.

```
1  #checking info about the data
2  bike_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #    Column                 Non-Null Count  Dtype
---   ------                 --------------  -----
 0    Date                   8760 non-null   object
 1    Rented_Bike_Count      8760 non-null   int64
 2    Hour                   8760 non-null   int64
 3    Temperature            8760 non-null   float64
 4    Humidity               8760 non-null   int64
 5    Wind_speed             8760 non-null   float64
 6    Visibility             8760 non-null   int64
 7    Dew_point_temperature  8760 non-null   float64
 8    Solar_Radiation        8760 non-null   float64
 9    Rainfall               8760 non-null   float64
 10   Snowfall               8760 non-null   float64
 11   Seasons                8760 non-null   object
 12   Holiday                8760 non-null   object
 13   Functioning_Day        8760 non-null   object
dtypes: float64(6), int64(4), object(4)
memory usage: 958.2+ KB
```

Table2 - Columns in

Data types of various columns object, Float and Integer. Dependent variable being Rented Bike Count and all other variables are our Independent variables.

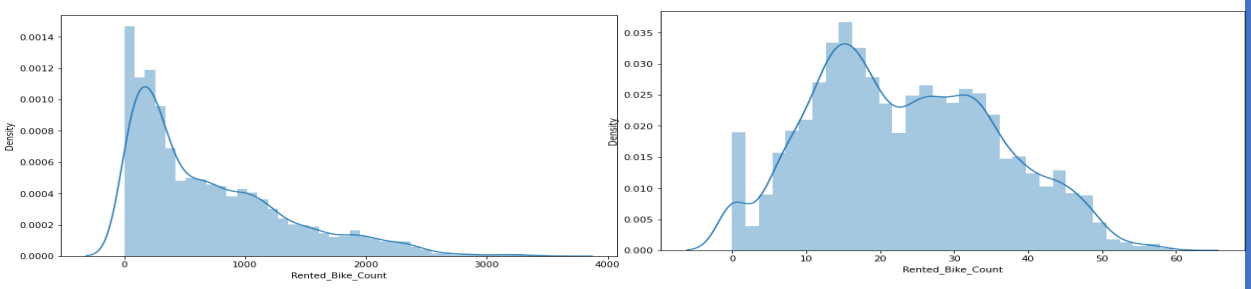
2. EDA & Business Implication:

EDA stands for exploratory data analysis where we explore our data and grab insights from it. EDA helps us in getting knowledge in form of various plots and diagrams where we can easily understand the data and its features.

	count	mean	std	min	25%	50%	75%	max
Rented_Bike_Count	8760.0	704.602055	644.997468	0.0	191.00	504.50	1065.25	3556.00
Hour	8760.0	11.500000	6.922582	0.0	5.75	11.50	17.25	23.00
Temperature	8760.0	12.882922	11.944825	-17.8	3.50	13.70	22.50	39.40
Humidity	8760.0	58.226256	20.362413	0.0	42.00	57.00	74.00	98.00
Wind_speed	8760.0	1.724909	1.036300	0.0	0.90	1.50	2.30	7.40
Visibility	8760.0	1436.825799	608.298712	27.0	940.00	1698.00	2000.00	2000.00
Dew_point_temperature	8760.0	4.073813	13.060369	-30.6	-4.70	5.10	14.80	27.20
Solar_Radiation	8760.0	0.569111	0.868746	0.0	0.00	0.01	0.93	3.52
Rainfall	8760.0	0.148687	1.128193	0.0	0.00	0.00	0.00	35.00
Snowfall	8760.0	0.075068	0.436746	0.0	0.00	0.00	0.00	8.80

Table-3: Analysis of Data

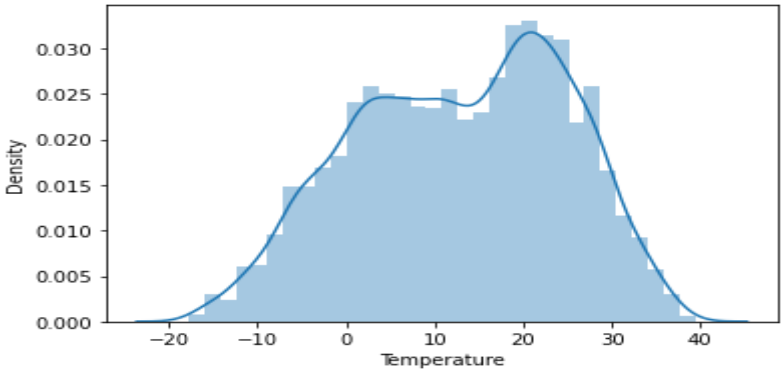
Distribution of Dependent Variable



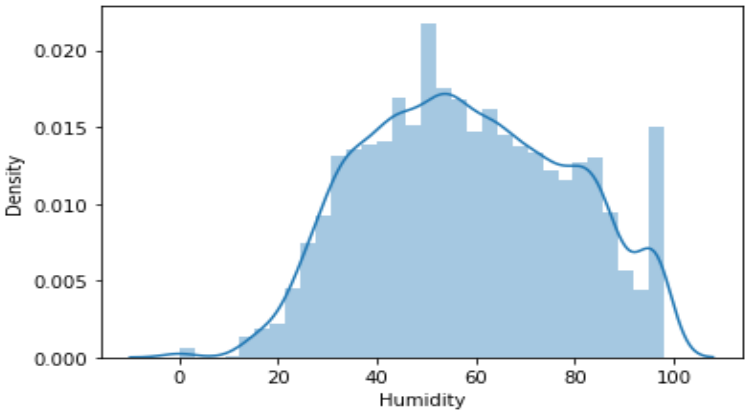
Plot1 – Analysis on Rented_bike_count

The above graph shows that Rented Bike Count has moderate right skew ness. And we know that the assumption of linear regression tells us that the distribution of dependent variable has to be normal, so we should perform some operation to make it normal.

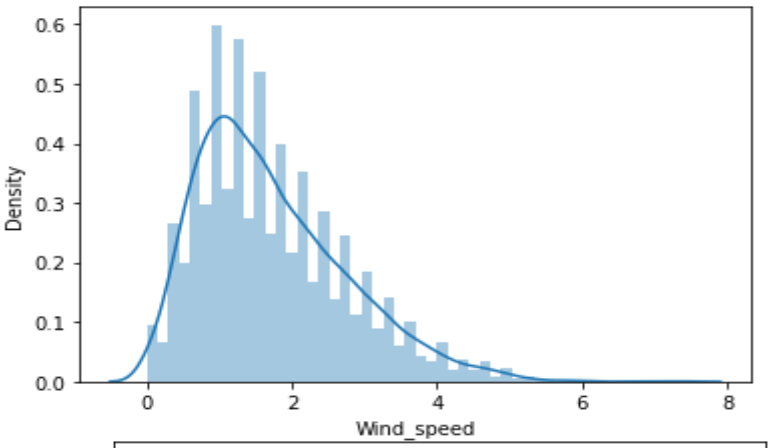
Visualizing Distributions



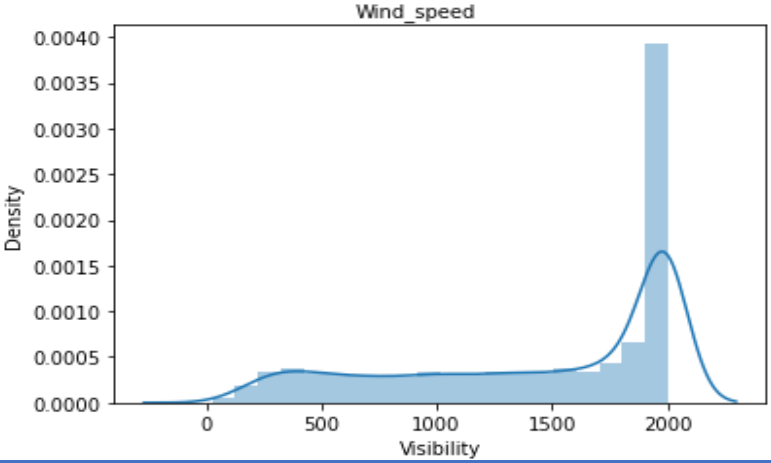
Plot-2: Analysis on Temperature



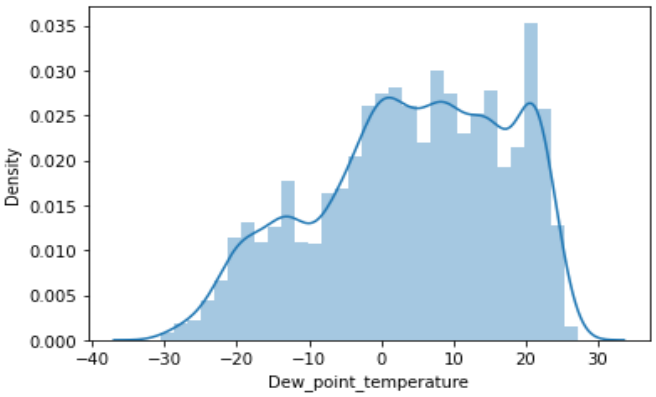
Plot-3: Analysis on Humidity



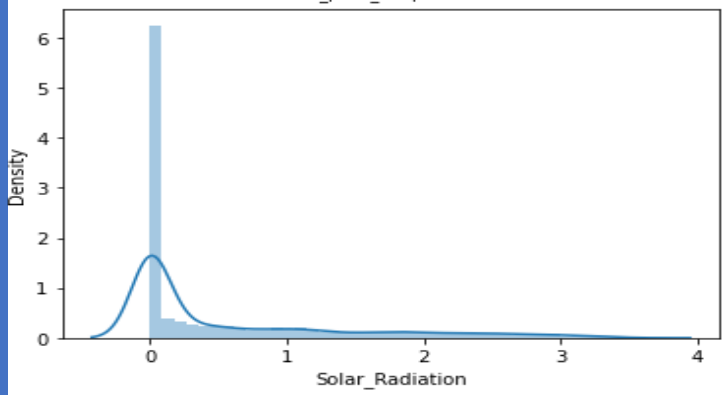
Plot-4: Analysis on Wind_speed



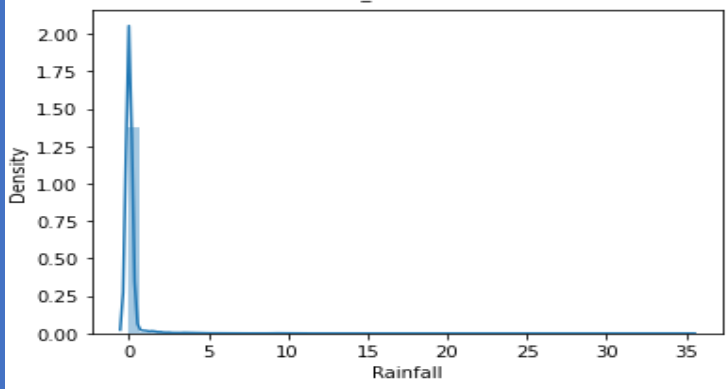
Plot-5: Analysis on Visibility



Plot-6: Analysis on Dew_point



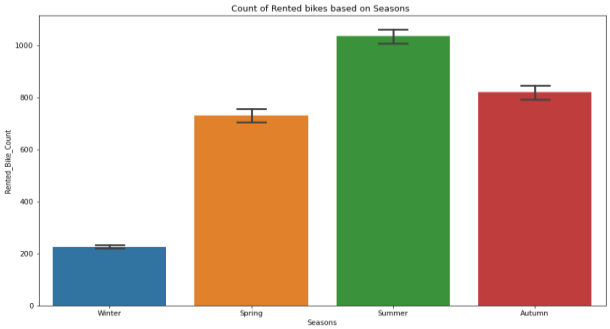
Plot-7: Analysis on Solar_radiation



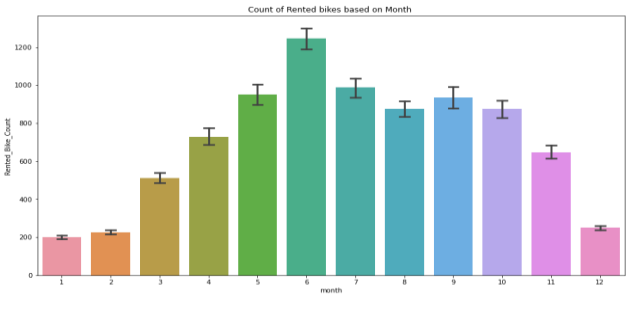
Plot-8: Analysis on Rainfall

- Temperature" and "Humidity" follows Uniform Distribution
- "Windspeed","Solar radition","Rainfall" and "Snowfall" are having right skewed distributions
- "Dew Point Temperature" and "Visibility" are having left skewed distributions.

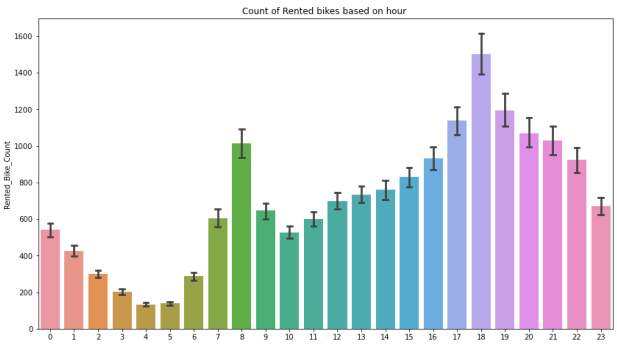
Bi- Variate Analysis



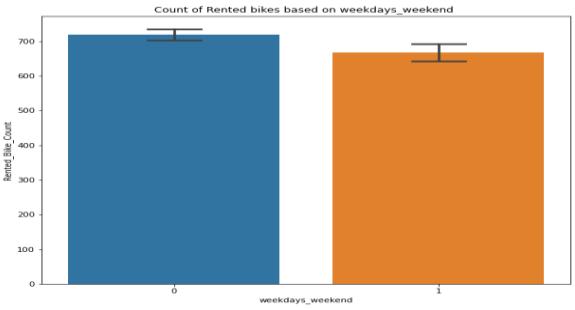
Plot-9: Analysis on Seasons and Rented_bike_count



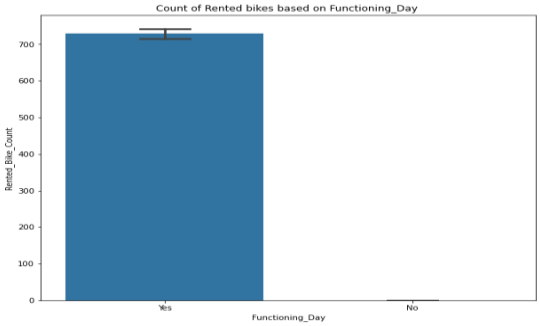
Plot-10: Analysis on month and Rented_bike_count



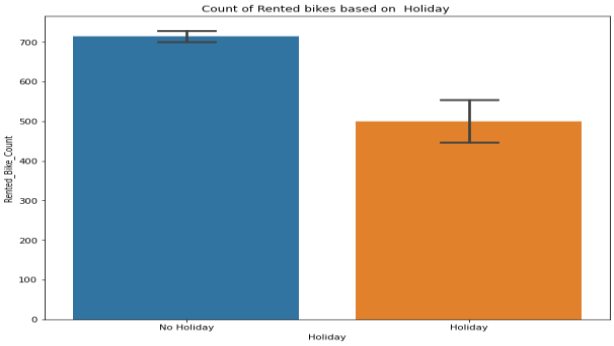
Plot-11: Analysis on Hour and Rented_bike_count



Plot-12: Analysis on weekdays_weekend and Rented_bike_count



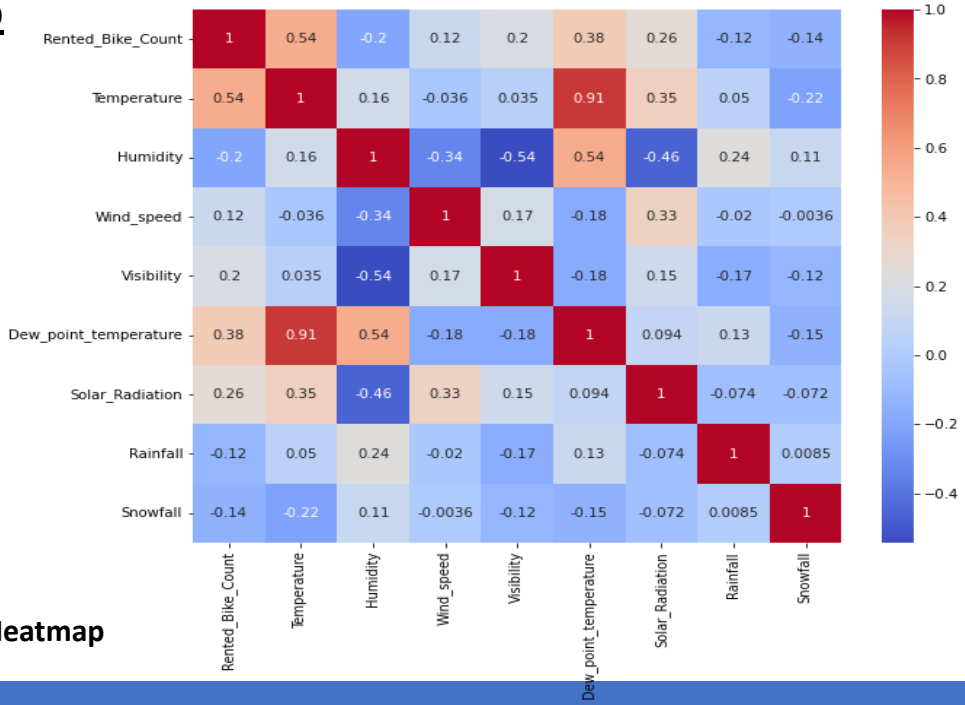
Plot-13: Analysis on Functioning_Day and Rented_bike_count



Plot-14: Analysis on Holiday and Rented_bike_count

- In summer season the use of rented bike is high whereas in winter season the use of rented bike is very low.
- The month 5 to 10 the demand of the rented bike is high as compare to other months.
- People generally use rented bikes during their working hour from 7am to 9am and 5pm to 8pm.
- In the week days which is in blue colur tells that the demand of the bike is higher because of the office as compare to the weekend.
- People use rented bike only in functioning day.
- Use of rented bike is more on no holiday as compare to holiday.

Heatmap



Plot15 – Heatmap

- there is a positive correlation between columns 'Temperature' and 'Dew point temperature' i.e 0.91 so even if we drop this column then it dont affect the outcome of our analysis.

3. Data Cleaning & Pre-processing

Data Cleaning is an important phase in any data science project, if our data is clean then only we can provide it to our machine learning model. Un-cleaned Data can further lead our model with low accuracy. And, if data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset.

```
#checking for null values in each column.
bike_df.isna().sum()
bike_df.isnull().sum()
```

```
Date                    0
Rented_Bike_Count       0
Hour                    0
Temperature             0
Humidity                0
Wind_speed              0
Visibility              0
Dew_point_temperature   0
Solar_Radiation         0
Rainfall                0
Snowfall                0
Seasons                 0
Holiday                 0
Functioning_Day         0
dtype: int64
```

Code Sample-1: Null Values

To identify any missing values in our data set we have used Pandas pre built function isnull() to detect any missing values in our datasets. As we can see that column body and acidity has a high number of missing values. So our next step is how to handle a large number of missing values. One approach is , that we will delete the column if we don't need that column for further analysis. And, what if we need that column for further analysis then we have use an approach will is a predefined function in Pandas called fillna().

As we can clearly see that we don’t have null values in any column so there is no need to perform any of the above mentioned steps in here.

<u>Need for Variable Transformation</u>:-

- Variable transformation is a way to make the data work better in your model. Here specifically we convert the "date" column into 3 different column i.e "year","month","day".
- The "year" column in our data set is basically contain the 2 unique number contains the details of from 2017 december to 2018 november .

```
# Changing the "Date" column into three "year","month","day" column

bike_df['Date'] = bike_df['Date'].apply(lambda x:
                                        dt.datetime.strptime(x,"%d/%m/%Y"))
```

```
bike_df['year'] = bike_df['Date'].dt.year
bike_df['month'] = bike_df['Date'].dt.month
bike_df['day'] = bike_df['Date'].dt.day_name()
bike_df['weekdays_weekend']=bike_df['day'].apply(lambda x : 1 if x=='Saturday' or x=='Sunday' else 0 )
bike_df=bike_df.drop(columns=['Date','day','year'],axis=1)
```

```
#Changing the int64 column into catagory column
cols=['Hour','month','weekdays_weekend']
for col in cols:
  bike_df[col]=bike_df[col].astype('category')
```

Code Sample2– Variable Transformation

Variables removed or added and why?

Variables are removed in such a scenario where we have a large number of null values in any columns particularly or else in our dataset, to make our data clean and ready for modelling. Whereas, variables are added in such a scenario where Consider an example where we have a column as start-date and another column as end-date so we can create a column named 'difference' where we subtract the start-date and end-date and have a number of days between them in a column named 'difference' and later drop start-date and end-date as if they are of no use.

- consider this is a one year then we don't need the "year" column so we drop it.
- The other column "day", it contains the details about the each day of the month, for our relevance we don't need each day of each month data but we need the data about, if a day is a weekday or a weekend so we convert it into this format and drop the "day" column.

4. Modeling Building

Model Selection and Why?

After cleaning and processing the data then comes the modelling part which includes building Machine Learning models, let's first understand in brief what Machine Learning is?

Machine Learning is a technique that analyses past data and tries to extract meaningful insights and patterns from them which can be further used to perform predictions in future. For example, classifying whether a tumor is benign or malignant, predicting stock prices, etc. One such application which we're using right here is predicting house prices. Before making predictions first we need to build a model and train it using past data.

First, we need to separate the dataset into two parts: features (property attributes) and labels (prices) which is the required format for any model to be trained on.

Then the data needs to be split into 3 sets

1. Training set - This will be the part of the dataset which the model will be using to train itself, the size should be at least 60-70% of the total data we've.

2. Validation set - This set is used for validating our model's performance for a different set of hyperparameters. After taking out the train set, the remaining set can be split into validation and test set.

3. Testing set - To evaluate how the model is performing on the unseen data on which the model will be doing future predictions on, test set is used. It helps to understand how much error is there between actual and predicted values.

Train Test split for regression

```
#Assigning the value in X and Y
X = new_bike_df.drop(columns=['Rented_Bike_Count'], axis=1)
y = np.sqrt(new_bike_df['Rented_Bike_Count'])
```

Code Sample 3: Splitting Data

We need to build different regression algorithms and using the testing set we can determine which model to keep for making final predictions.

Here is the list of all the algorithms we've to build and evaluated:

Following Models are used for predicting Sales of Big Mart Store wise:

- **Linear Regression**
- **Lasso Regression**
- **Ridge Regression**
- **Elastic net Regression**
- **Decision Tree**
- **Random Forest**
- **Gradient Boosting**

- Initially, we've tried Linear Regression and its variants Lasso (l1 norm) , Ridge (l2 norm) Regression and Elastic net Regression , but they are performing quite similarly on the 2 sets (train & test).
- They are not giving good result on train, neither test set , with lasso giving low r-2 score for both train test set which is not good.
- Let's look at some other algorithms, and how they are performing as compared to linear regression
- So, **Decision Tree Regression** is giving score of 0.70 on the train set and similarly on the valid and test set.
- If we look at the Random Forest Regressor is giving quite good score of 0.99 on the train set, and equally good on the valid and test set, looks like it is not overfitting on the training data.
- Finally, now we can try those boosting algorithms and see where they are getting us? Generally boosting algorithms give a very good performance, and they are giving on the training set but there isn't any significant improvement on the test set as compared to Tree-based models.
- Same Case with the Gradient boosting regressor , No Over fitting is seen in both of these as well.

Efforts to improve model performance

- Hyperparameter Tuning
- Hyperparameter tuning is the process of trying out a different set of model parameters, actually, they are algorithm's parameter for example theta1 and theta2 in the hypothesis function for Linear Regression is model parameter, but lambda which is a factor that decides the amount of regularization is algorithm's parameter called as a

hyperparameter. Tuning hyper parameters helps us to find out the optimal parameter values for which the model is giving less error and a better overall score. We've performed tuning for Gradient Boosting Regressor with both the methods RandomizedSearchCV as well as GridSearchCV. What random search does is from the given set of parameter values, it tries out n number of combinations whereas grid search builds a model with all the possible combinations from the given set of parameters and gives us the optimal values.

Gradient Boosting Regressor with GridSearchCV

```
# Number of trees
n_estimators = [50,80,100]

# Maximum depth of trees
max_depth = [4,6,8]

# Minimum number of samples required to split a node
min_samples_split = [50,100,150]

# Minimum number of samples required at each leaf node
min_samples_leaf = [40,50]

# HYperparameter Grid
parameter_dict = {'n_estimators' : n_estimators,
              'max_depth' : max_depth,
              'min_samples_split' : min_samples_split,
              'min_samples_leaf' : min_samples_leaf}
```

```
parameter_dict
```

```
{'n_estimators': [50, 80, 100],
 'max_depth': [4, 6, 8],
 'min_samples_split': [50, 100, 150],
 'min_samples_leaf': [40, 50]}
```

```
#importing package
from sklearn.model_selection import GridSearchCV
# Create an instance of the GradientBoostingRegressor
gb_model = GradientBoostingRegressor()

# Grid search
gb_grid = GridSearchCV(estimator=gb_model,
                       param_grid = parameter_dict,
                       cv = 5, verbose=2)

gb_grid.fit(X_train,y_train)
```

Code Sample4 – Hyper Parameter Tuning

After fitting GridSearchCV it returns those params with which it got the best score.

```
gb_grid.best_params_
```

```
{'max_depth': 8,
 'min_samples_leaf': 40,
 'min_samples_split': 50,
 'n_estimators': 100}
```

Code Sample5 – Best parameters

4. Performance Metrics

Just building is not enough, as we need to evaluate it using different metrics based on the problem we're solving. Model Validation helps us to understand how well the model is generalizing on the real-world data, the data which it has not seen during the training phase.

For regression problems the evaluation metrics we've used are:

- RMSE (Root Mean Squared Error)
- MSE (Mean Squared Error)
- MAE (Mean Absolute Error)
- MAPE (Mean Absolute Percentage Error)
- Adjusted R2

The R2 score is between 0 and 1 (it can also get -ve when the model is performing worse). The closer the value to 1 the better the performance of the model will be and a model which always predicts constant value irrespective of the input values gets an R2 score of 0.

MAE is on average how far those predicted values are from actual values, whereas MSE is the average of squared differences between actual and predicted values.

Let's look at these metrics for the best-performing algorithms on the test set, Here are the top 10 algorithms based on the test score, MAE, MSE, MAPE, and Adjusted R2. Depending on increasing or decreasing which metric helps solve the business problem, we can pick the appropriate model for final deployment.

Note: Other than Test scores all scores are sorted in ascending order

		Model	MAE	MSE	RMSE	R2_score	Adjusted R2
Training set	0	Linear regression	4.474	35.078	5.923	0.772	0.77
	1	Lasso regression	7.255	91.594	9.570	0.405	0.39
	2	Ridge regression	4.474	35.078	5.923	0.772	0.77
	3	Elastic net regression	5.792	57.574	7.588	0.626	0.62
	4	Dicision tree regression	5.077	46.748	6.837	0.696	0.69
	5	Random forest regression	0.803	1.588	1.260	0.990	0.99
	6	Gradient boosting regression	3.269	18.648	4.318	0.879	0.88
	7	Gradient Boosting gridsearchcv	1.849	7.455	2.730	0.952	0.95
Test set	0	Linear regression	4.410	33.275	5.768	0.789	0.78
	1	Lasso regression	7.456	96.775	9.837	0.387	0.37
	2	Ridge regression	4.410	33.277	5.769	0.789	0.78
	3	Elastic net regression Test	5.874	59.451	7.710	0.624	0.62
	4	Elastic net regression Test	5.874	59.451	7.710	0.624	0.62
	5	Dicision tree regression	5.304	53.148	7.290	0.664	0.66
	6	Random forest regression	2.212	12.729	3.568	0.919	0.92
	7	Gradient boosting regression	3.493	18.648	4.318	0.865	0.86
	8	Gradient Boosting gridsearchcv	2.401	12.393	3.520	0.922	0.92

Table 5 – performance metrics

5. Final Interpretation/Recommendation

In our analysis, we initially did EDA on all the features of our datset. We first analysed our dependent variable i.e, 'Rented Bike Count' and also transformed it. Next we analysed categorical variable and dropped the variable who had majority of one class. we also analysed numerical variable, check out the correlation, distribution and their relationship with the dependent variable. We also removed some numerical features who had mostly 0 values and hot encoded the categorical variables.

- No overfitting is seen.
- Random forest Regressor gives the highest R2 score of 99% for Train Set and Gradient Boosting gridsearchcv gives the highest R2 score of 92% for Test set.
- We can deploy this model.