<u>How We Made Progress with Predicting Windmill Power</u>

<u>Article Outline</u>

1. The Challenge

It is the year 2021 and we are at the verge of a massive climatic change. With global warming at its peak and fossil fuels inching towards its extinction, it is the need of the hour to step up and take responsibility for our planet. Developing countries all over the world are making a shift towards a cleaner energy source and are looking at ways to expand their global energy source power.

Switching to renewable energy sources is a great way to reduce dependency on imported fuels and increase cost efficiency. It is time we move towards a low-carbon future by embracing solar, hydro, geothermal energy and so on, to protect mother nature.

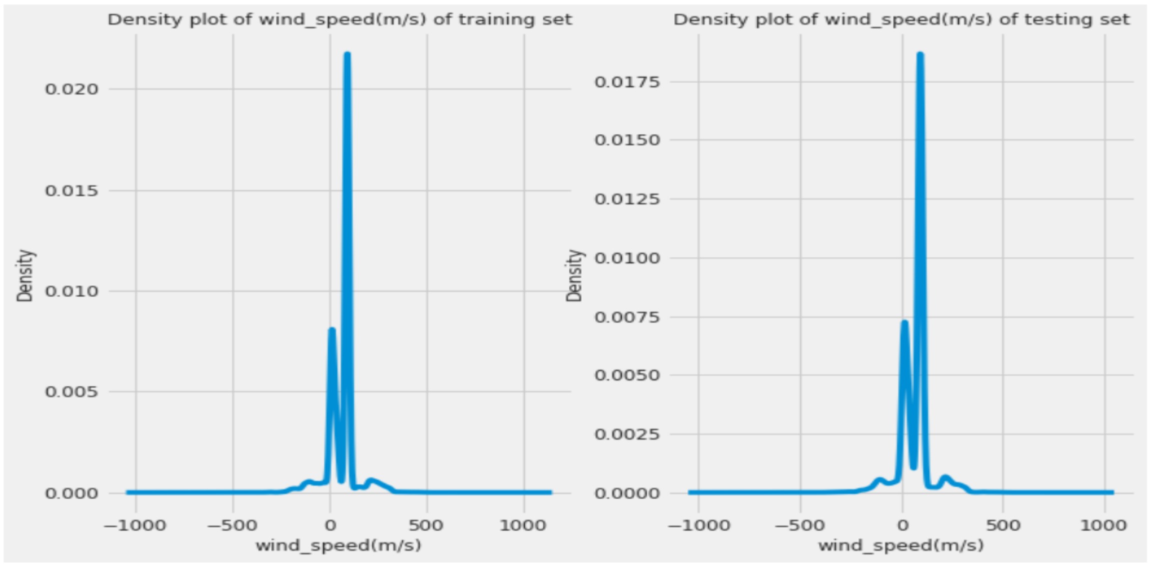
An efficient energy source that has been gaining popularity around the world is wind turbines. Wind turbines generate power by capturing the kinetic energy of the wind. Factors such as temperature, wind direction, turbine status, weather, blade length, and so on influence the amount of power generated.

Rules:

1. You are appointed by an environmentalist for their Non Government Organization as a climate warrior who comes to the rescue. Your task is to build a sophisticated Machine Learning model that predicts the power that is generated (in KW/h) based on the various features provided in the dataset.
2. Dataset: The dataset consists of parameters such as the temperature, wind direction, turbine status, weather, blade length, and the like.

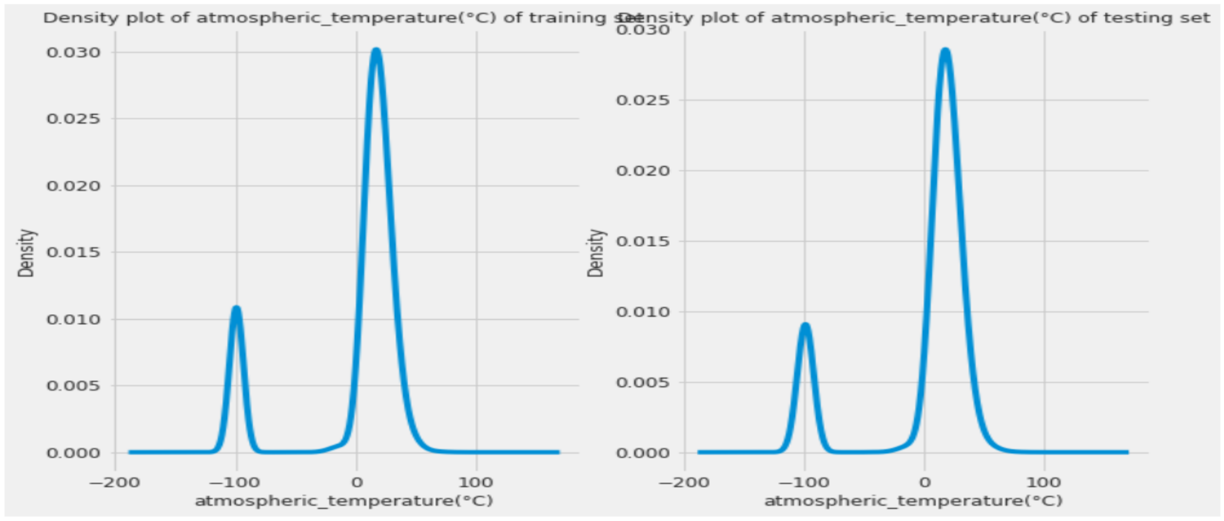
2. Data Exploration and Analysis:

a. *wind_speed(m/s)*



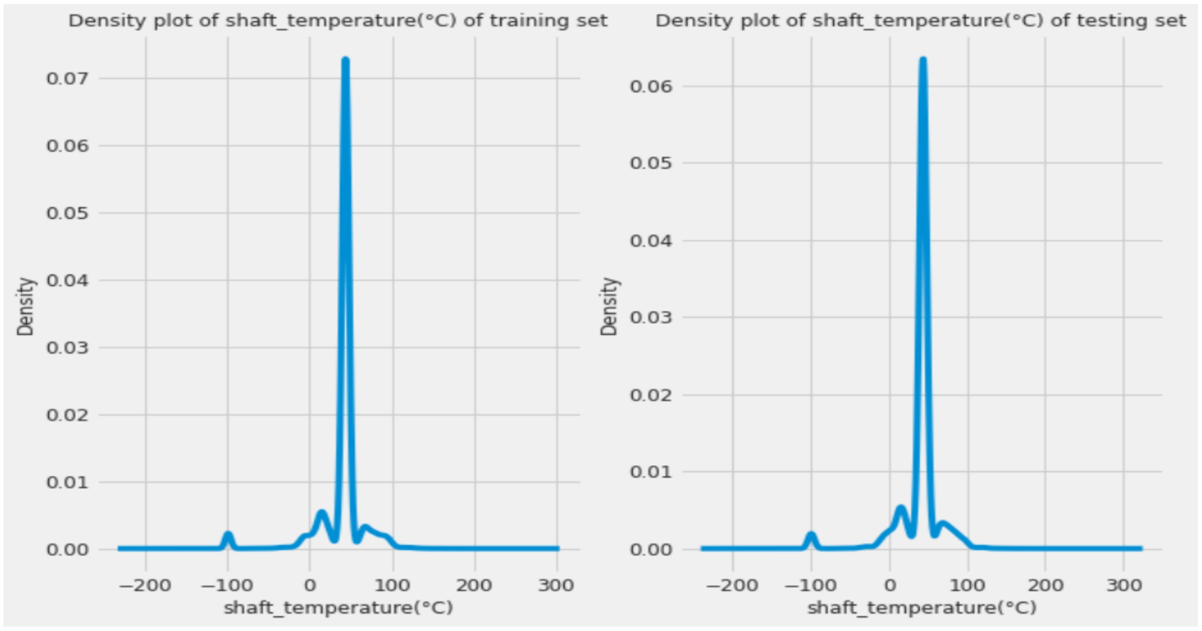
Distribution of feature wind_speed(m/s) of training and testing dataset are very similar.

b. *atmospheric_temperature(°C)*



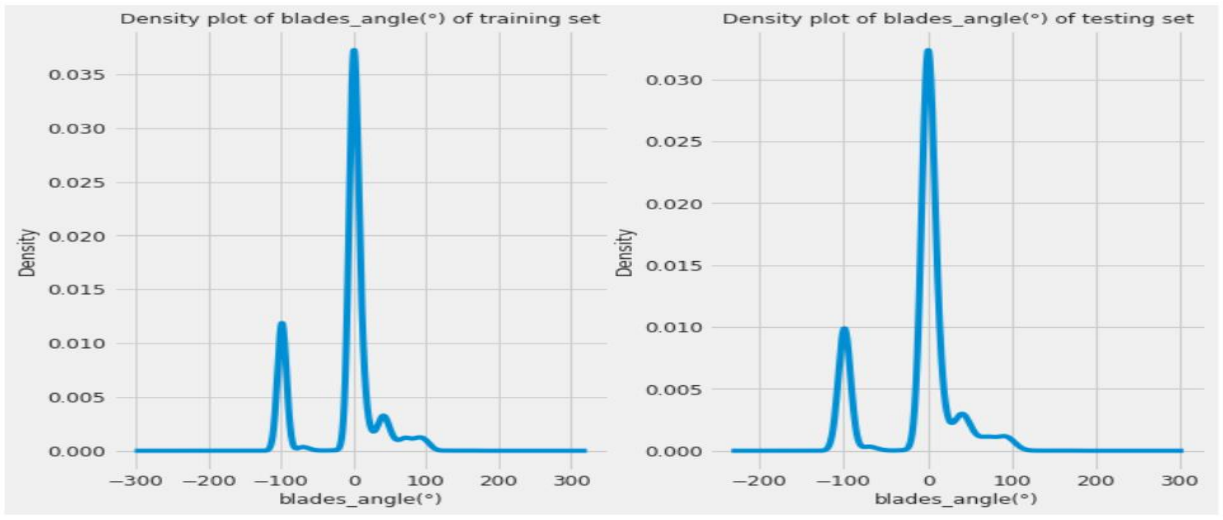
From the above density plot of feature "atmospheric_temperature(°C)", we found that the distribution of "atmospheric_temperature(°C)" is almost same in training and testing dataset, so we are not changing anything in it.

c. *shaft_temperature(°C)*



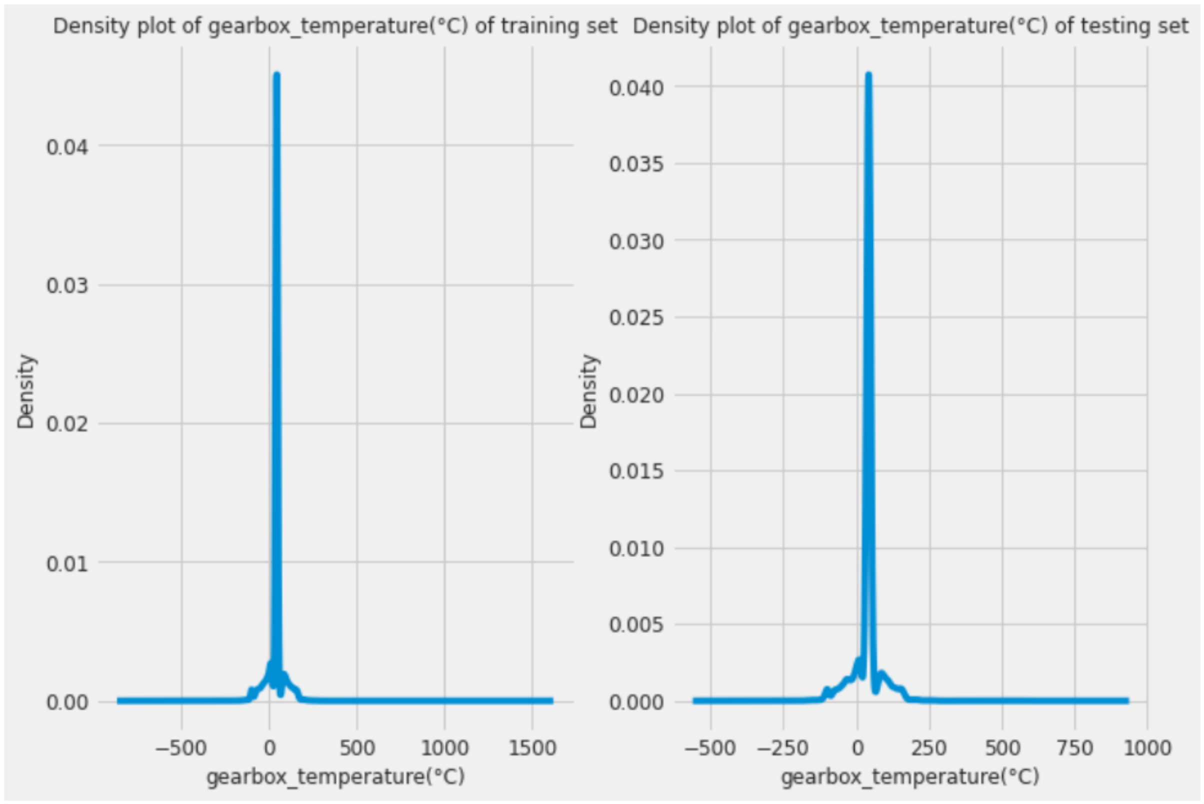
From the above density plot of feature "shaft_temperature(°C)", we found that the distribution of "shaft_temperature(°C)" is almost same in training and testing dataset, so we are not changing anything in it.

blades_angle(°):

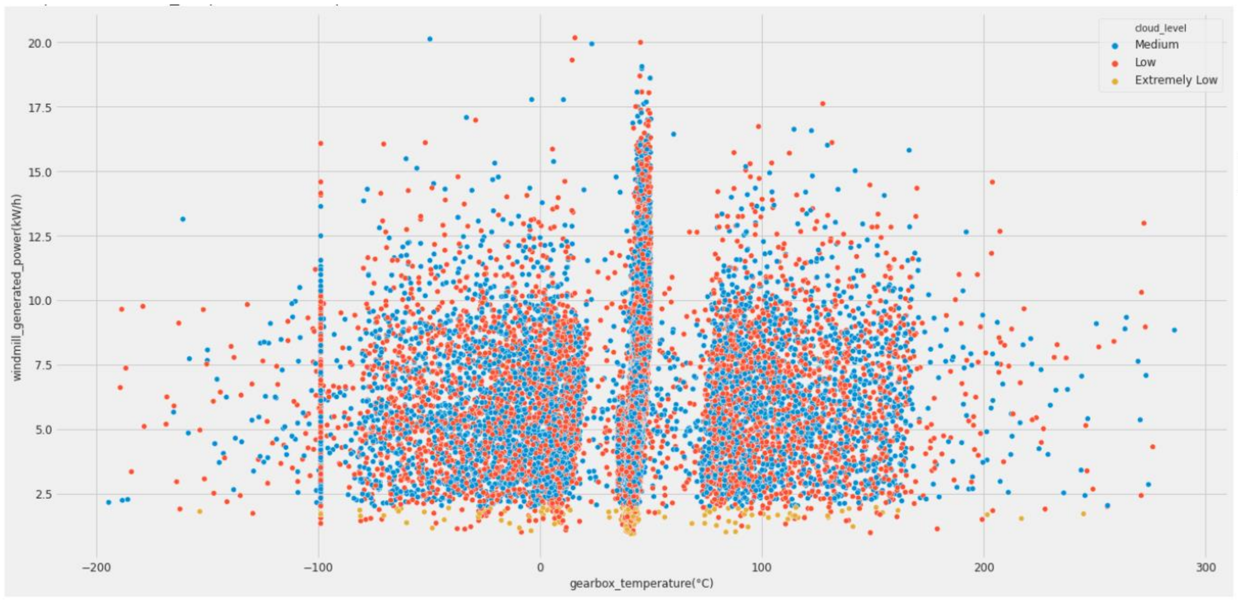


From the above density plot of feature "blades_angle(°)", we found that the distribution of "blades_angle(°)" is almost same in training and testing dataset, so we are not changing anything in it.

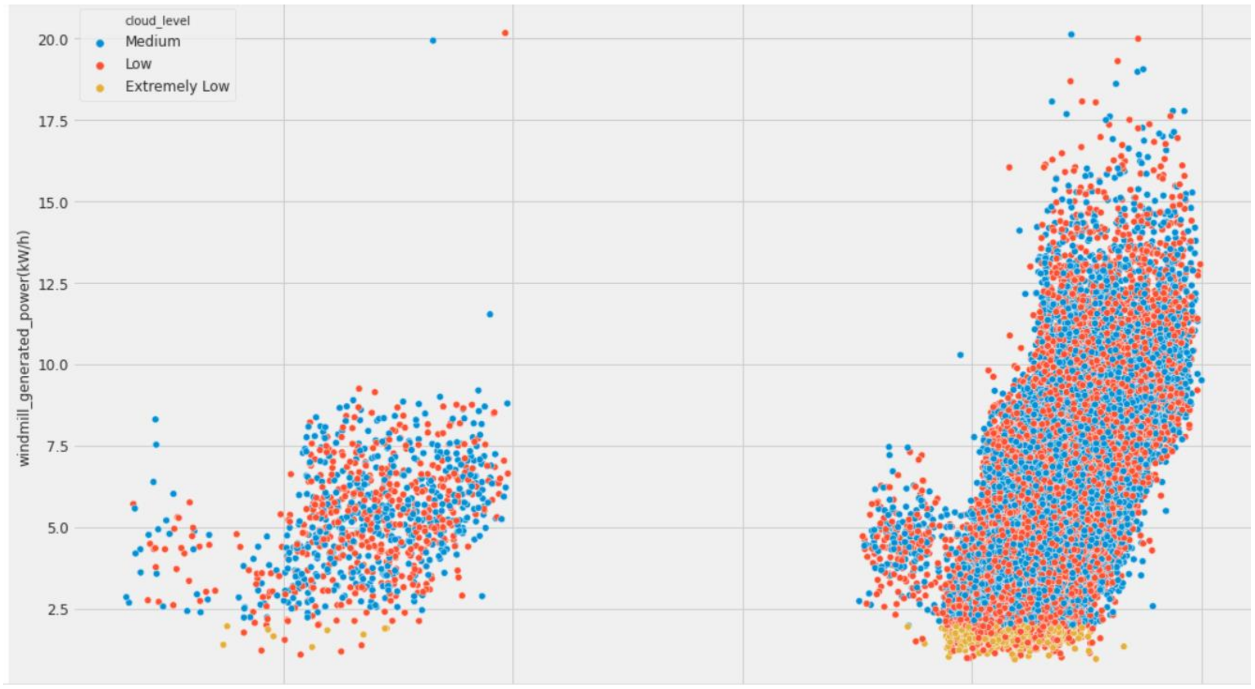
gearbox_temperature(°C)



From the above density plot, we found that the distribution is not same in training and testing dataset, so we are removing outliers that are above 300 and below -200.After removing outliers,

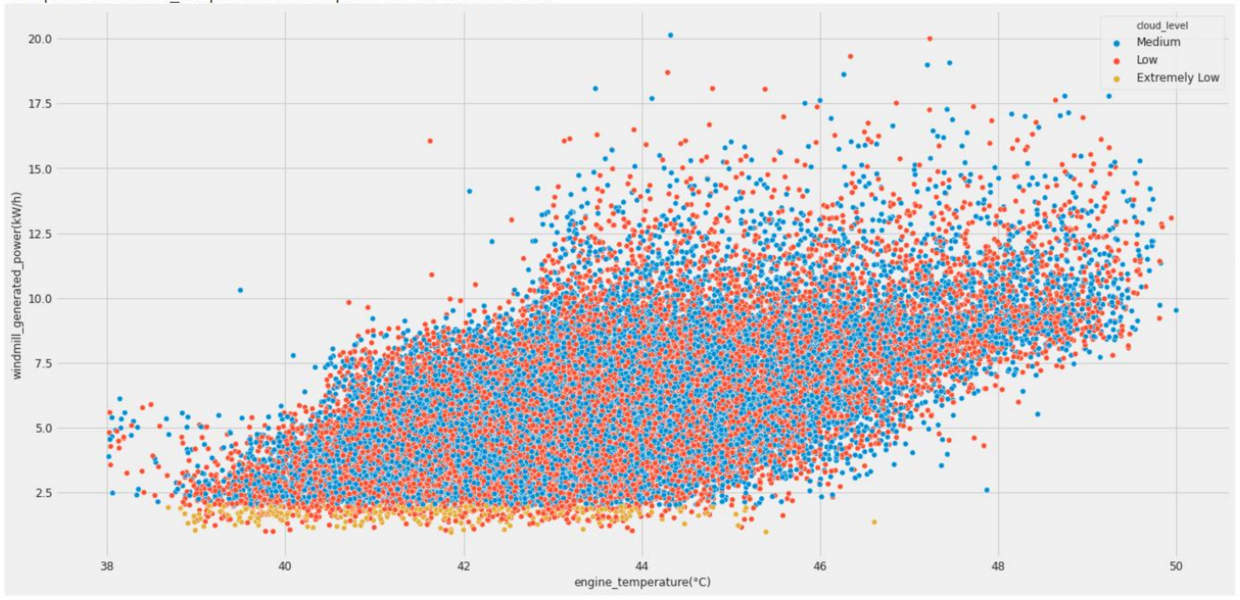


engine_temperature(°C)

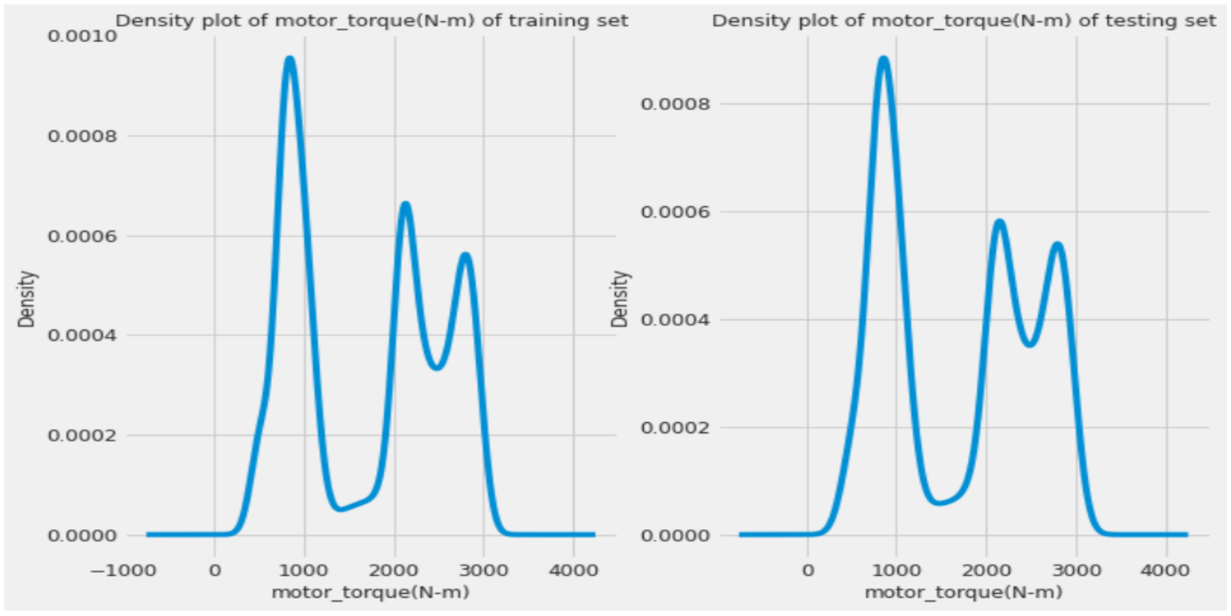


From the above density plot, we found that the distribution is not same in training and testing dataset, so we are removing outliers that are above 300 and below -200.After removing outliers,

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fb439d590>
```

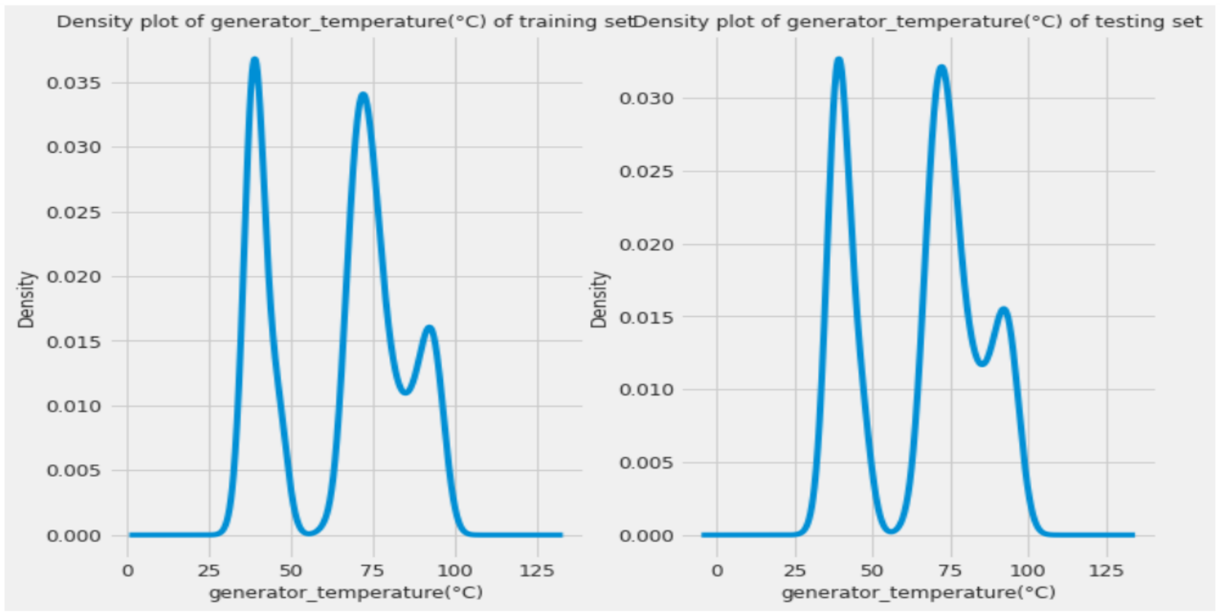


motor_torque(N-m)



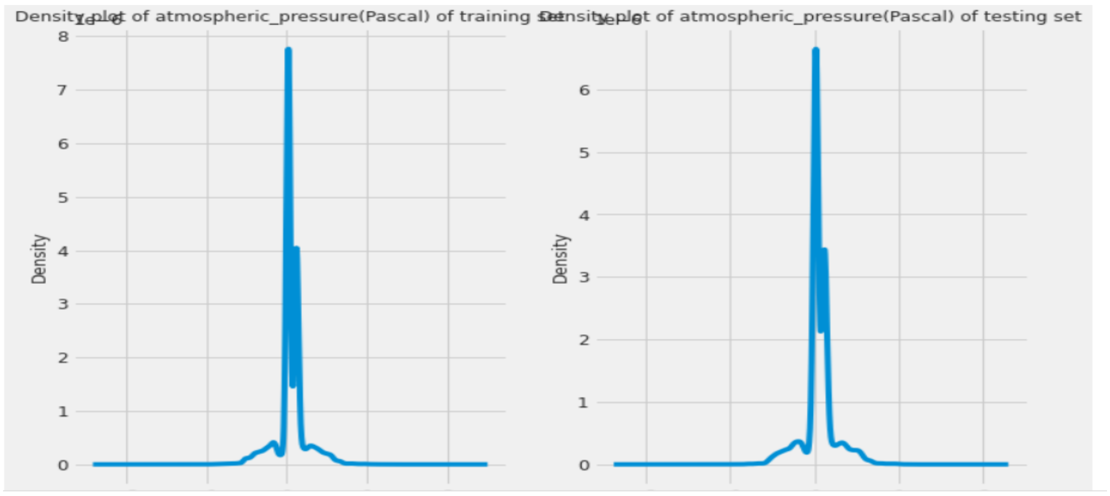
From the above density plot of feature "motor_torque(N-m)", we found that the distribution of "motor_torque(N-m)" is almost same in training and testing dataset, so we are not changing anything in it.

generator_temperature(°C)



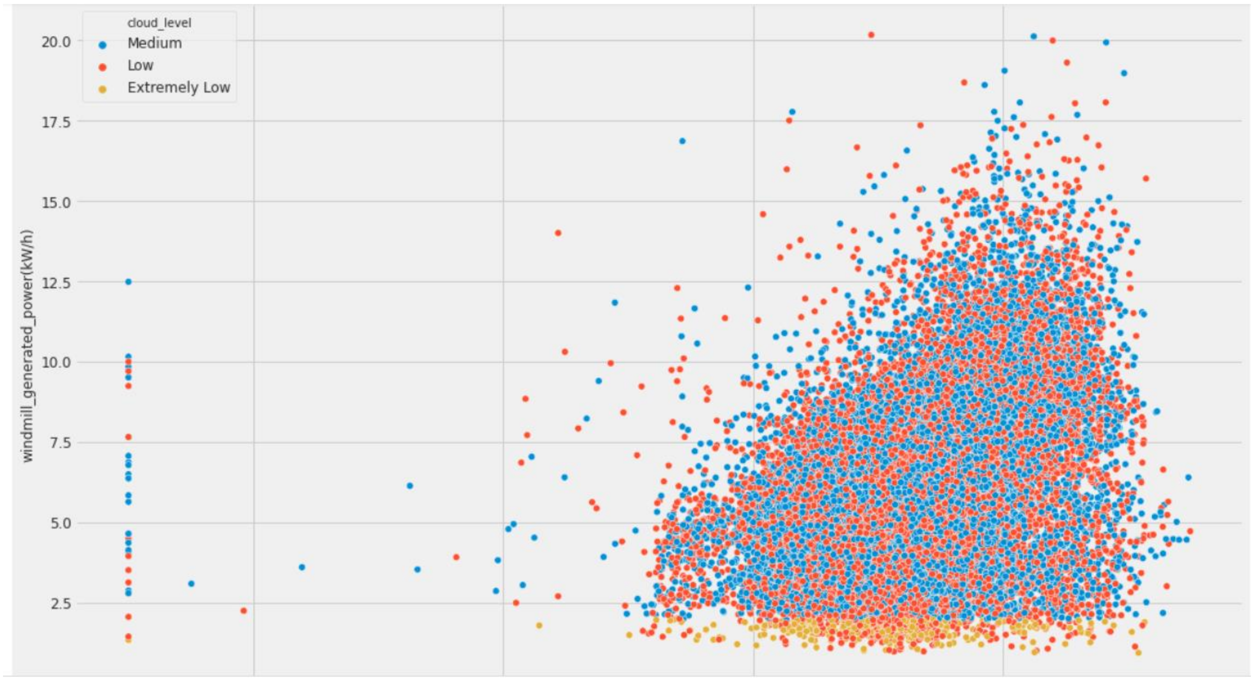
From the above density plot of feature "generator_temperature(°C)", we found that the distribution of "generator_temperature(°C)" is almost same in training and testing dataset, so we are not changing anything in it.

atmospheric_pressure(Pascal)

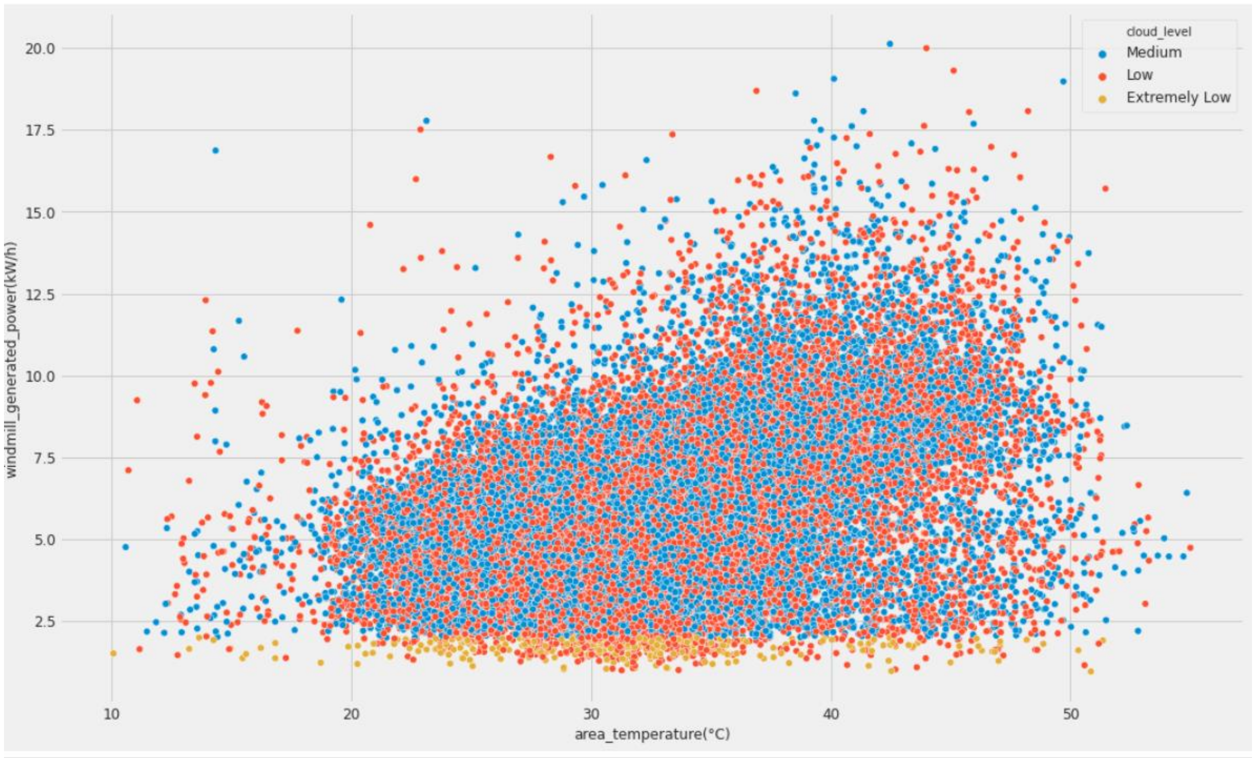


From the above density plot of feature "atmospheric_pressure(Pascal)", we found that the distribution of "atmospheric_pressure(Pascal)" is almost same in training and testing dataset, so we are not changing anything in it.

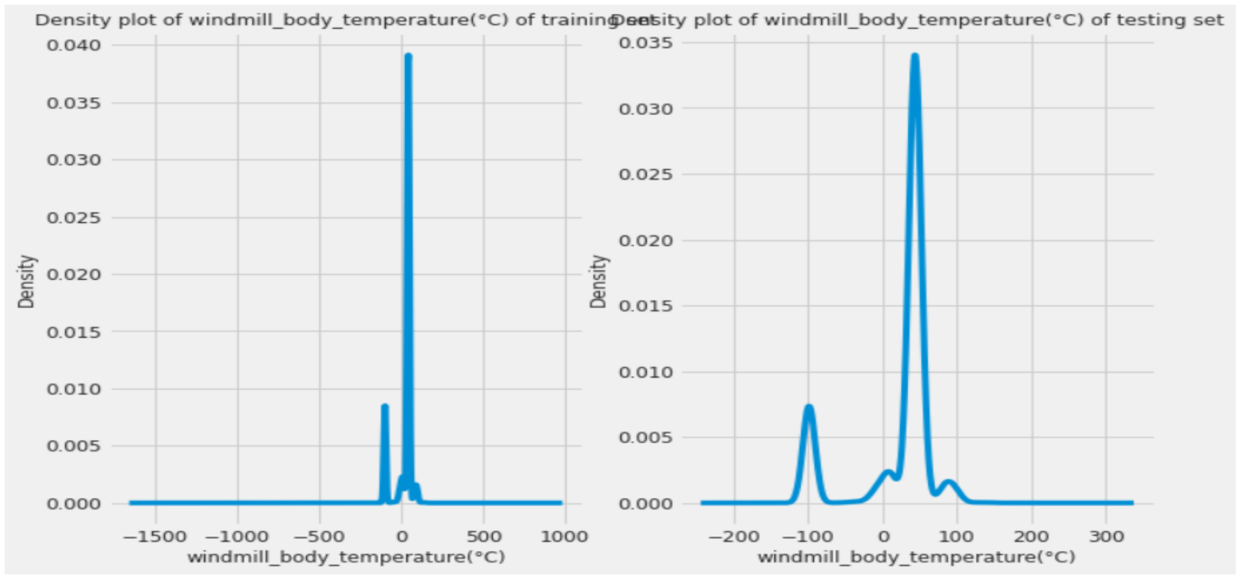
area_temperature(°C)



From the above density plot, we found that the distribution is not same in training and testing dataset, so we are removing outliers that are above 300 and below -200.After removing outliers,

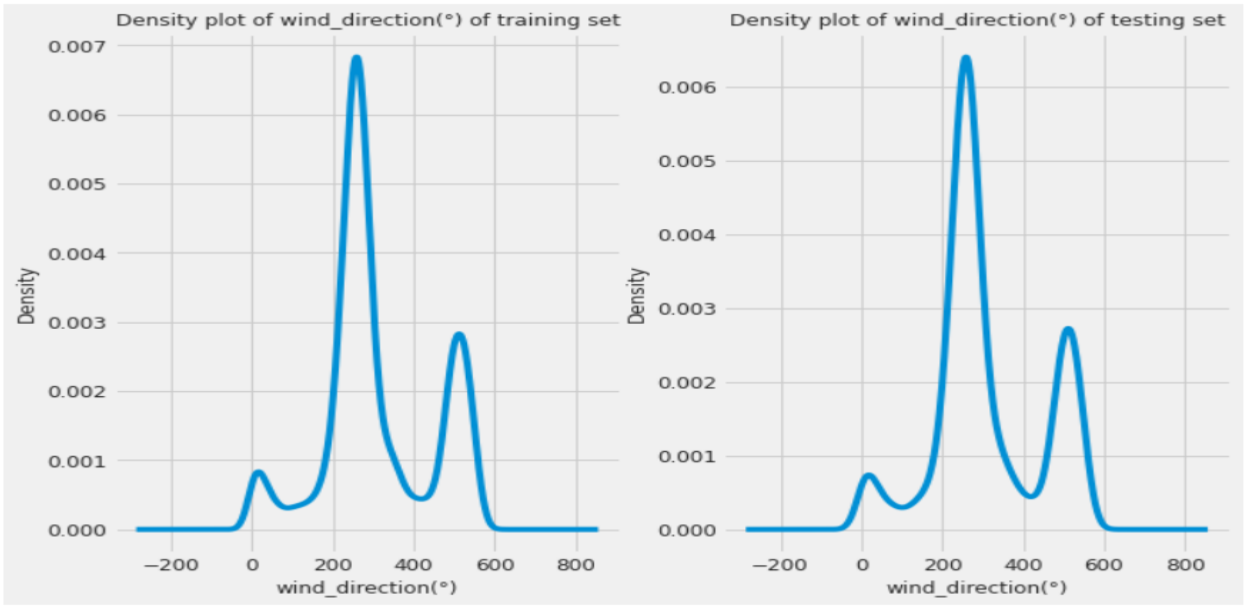


windmill_body_temperature(°C)



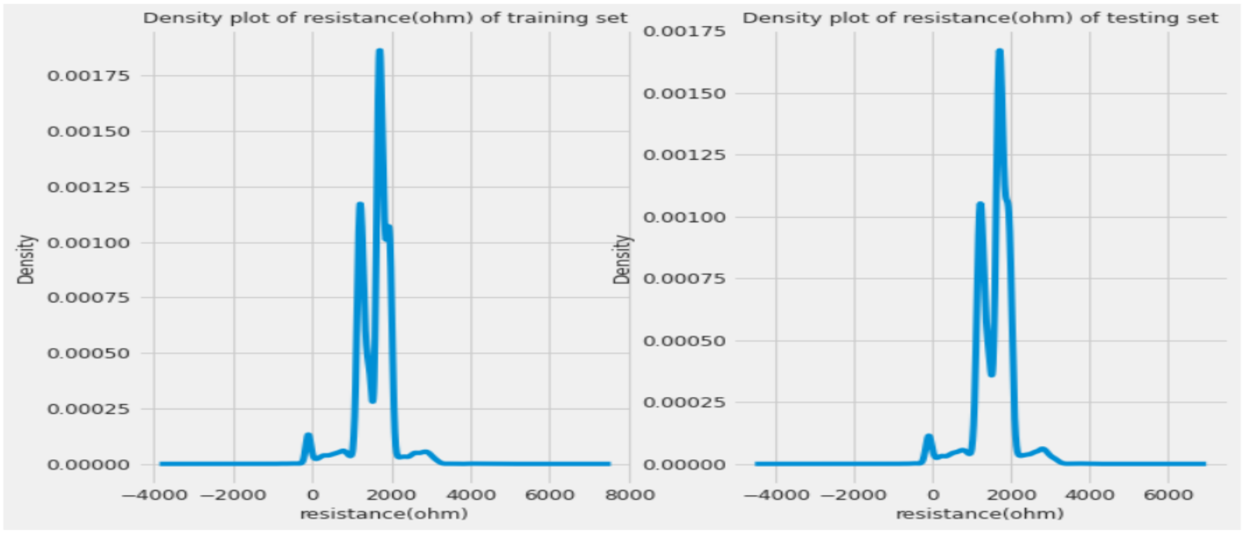
This feature "windmill_body_temperature(°C)" doesn't have same distribution in training and testing set and it is not much correlated with targetFeature as well. So, we will drop it in the end.

wind_direction(°)



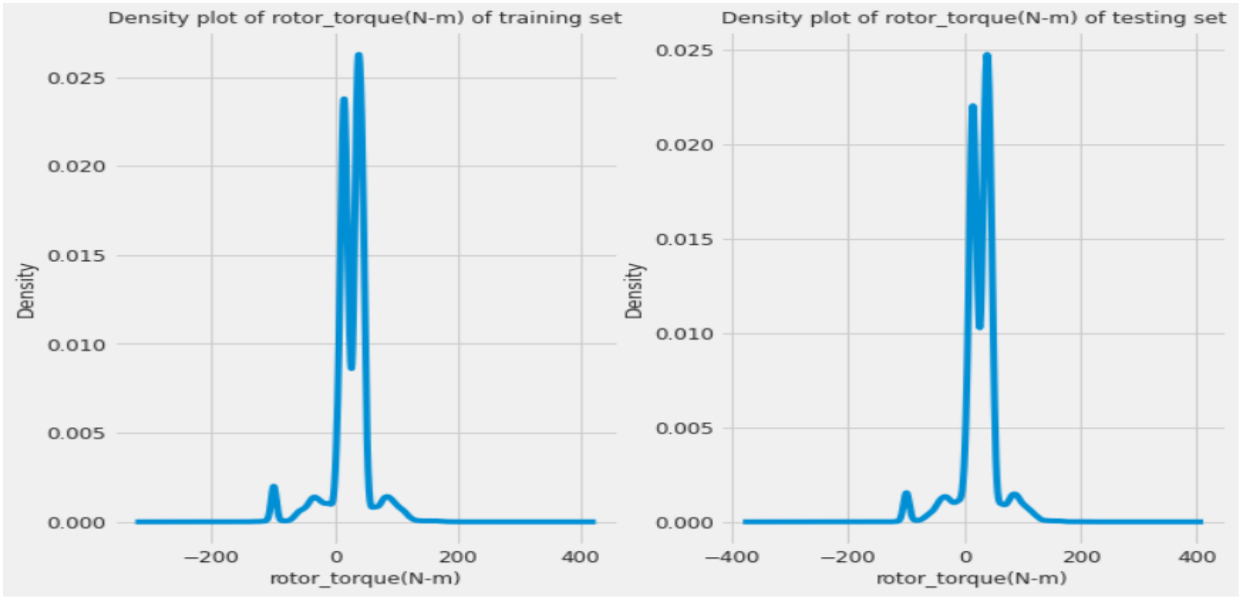
From the above density plot of feature "wind_direction(°)", we found that the distribution of "wind_direction(°)" is almost same in training and testing dataset, so we are not changing anything in it.

resistance(ohm)



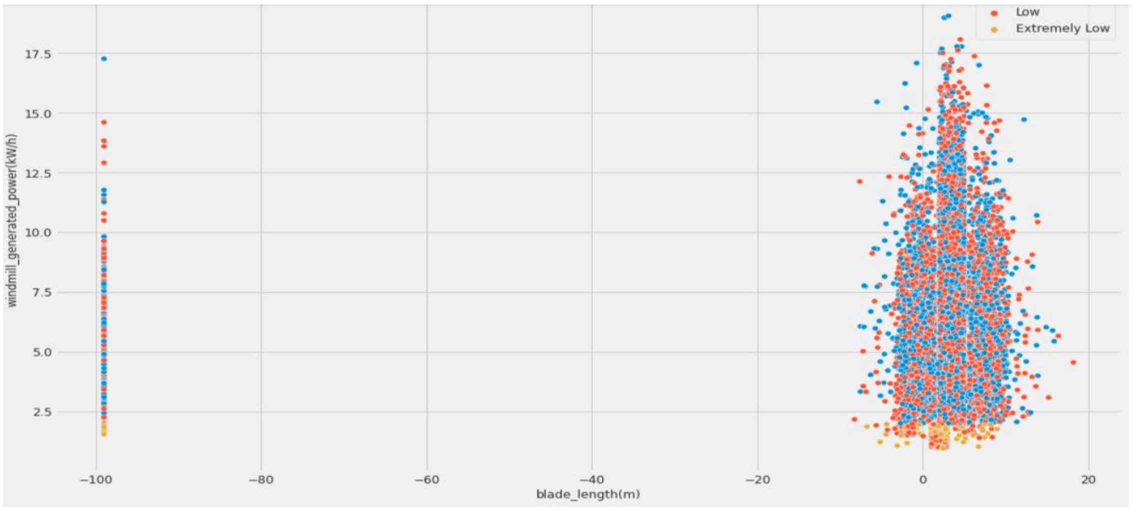
From the above density plot of feature "resistance(ohm)", we found that the distribution of "resistance(ohm)" is almost same in training and testing dataset, so we are not changing anything in it.

rotor_torque(N-m)

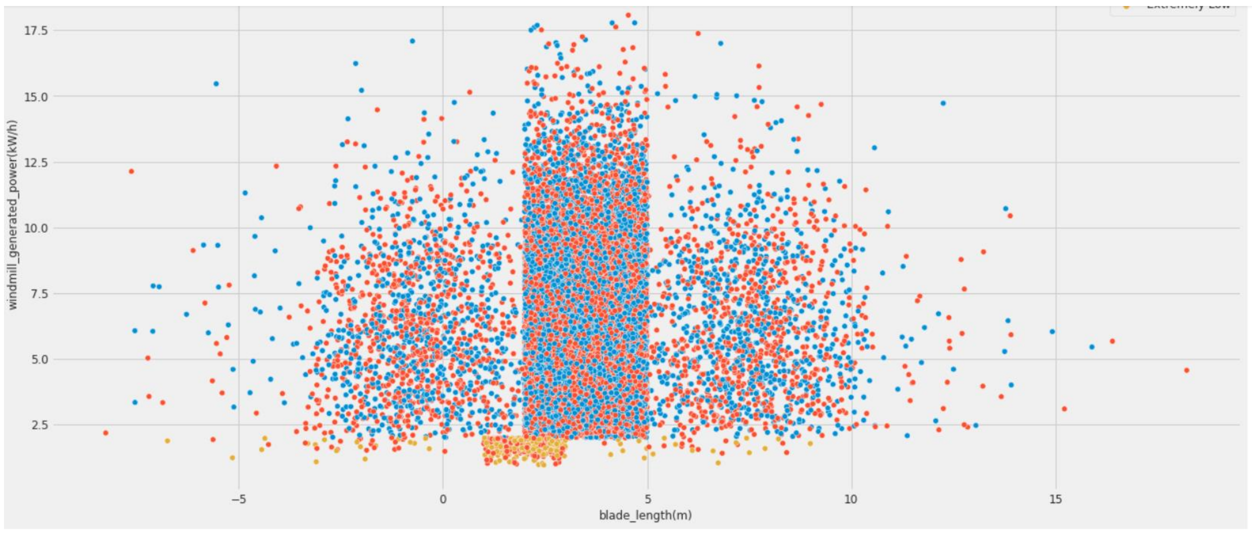


From the above density plot of feature "rotor_torque(N-m)", we found that the distribution of "rotor_torque(N-m)" is almost same in training and testing dataset, so we are not changing anything in it.

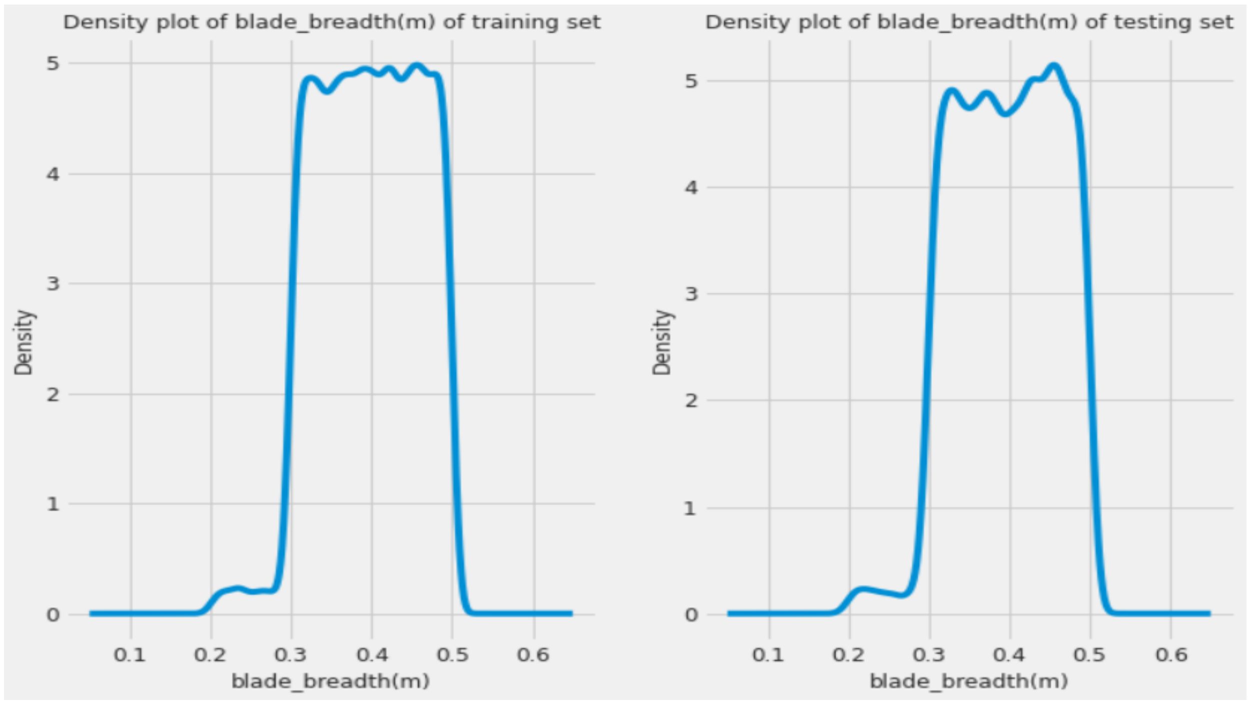
blade_length(m)



After removing outliers,

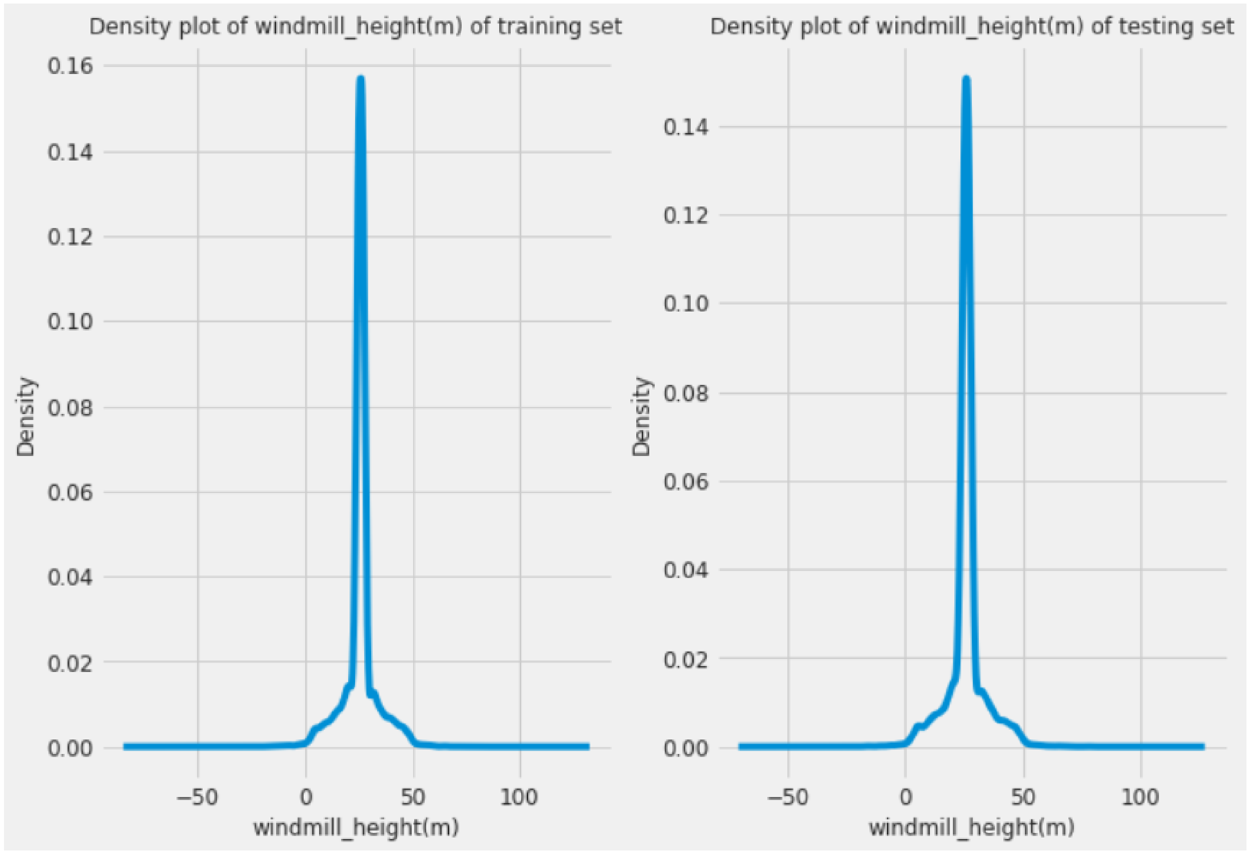


blade_breadth(m)



From the above density plot of feature "blade_breadth(m)", we found that the distribution of "blade_breadth(m)" is almost same in training and testing dataset, so we are not changing anything in it.

windmill_height(m)



From the above density plot of feature "windmill_height(m)", we found that the distribution of "windmill_height(m)" is almost same in training and testing dataset, so we are not changing anything in it.

3. Feature Engineering

- Imputed missing values for numerical columns with mean.
- Converted values of categorical columns into numerical using Label Encoder.
- Created new columns i.e., date, month, day of week from date_time column.

Used **Min-Max scaler** to bring all features into same scale as it is regression problem.

4. Tools Used

a. Google Colab Notebook
b. Numpy, Pandas, Seaborn, Matplotlib

5. Models used

a. Logistic Regression
b. Ridge and Lasso Regression
c. Polynomial Regression
d. SVR

e. AdaBoost
f. XGBoost
g. LightGBM
h. Random Forests
i. ANN network – with 'Adam' optimizer and 'mean absolute error' as metric to evaluate
j. CatBoostRegressor

6. Metrics Used to Evaluate Models:

We used

- r2 score
- MAE(Mea Absolute Error)
- MSE(Mean Squared Error)
- RMSE(Root Mean Squared Error)

to evaluate Regression and Boosting algorithms

7. Final Result:

We used ensemble model of RandomForestRegressor, Gradient Boosting, XGB Regressor for better score.