Weibull vs Nonparametric Estimation for Wind Resource Assessment

Using a Weibull distribution is a common approach for modeling wind speeds in the context of calculating the Annual Energy Production (AEP) of a wind farm. However, it's not an absolute requirement, and you can use other methods as well.

The Weibull distribution is often used because it can provide a reasonable approximation for the distribution of wind speeds at a particular site. Fitting a Weibull distribution to the observed wind speed data can help you model the wind speed probabilities, which in turn is important for estimating the AEP.

Weibull distribution consists of two parameters which are shape (k) and scale (a), that's why it is called a parametric distribution. In a parametric approach, you assume that your data follows a specific mathematical form or distribution, which is characterized by a set of parameters. By estimating these parameters from your data, you can describe the entire distribution using a simple mathematical formula.

But most of the time especially for the complex site, weibull distribution wouldn't be enough the represent the characteristics of wind speed data.

In these cases, the goodness of fit statistical performance indicators like Kolmogorov-Smirnov test should be checked. The goodness of fit of a distribution involves comparing the fitted distribution (e.g., Weibull) to the observed data to assess how well the distribution represents the data.

Besides, it is wiser to assess the alternative statistical distributions that represent your empirical distribution well. In this study, we will introduce the Kernel Density Estimate (KDE) which is a non-parametric approach. In a non-parametric approach, you avoid making specific assumptions about the functional form of the distribution. Instead, you seek to estimate the distribution directly from the data without assuming a particular distribution shape. KDE is a classic example of a non-parametric approach. It involves placing a kernel (a smooth function) at each data point and summing them up to create a smoothed estimate of the underlying probability density function (PDF). KDE doesn't assume any specific distribution; it adapts to the data's shape.

Consequently, both Weibull and KDE methods is going to be compared with the empirical distribution of the data. First, KS-test will be performed and then annual energfy production will be calculated with both of them. Power curve model created with the LOESS function will be used to calculate the power generation for each wind speed and it will be served as grand truth in the study by neglecting the lossess and uncertainties.

Let's start!

```
# First load the libraries
library(tidyverse)
conflicted::conflict_prefer(name = "select", winner = "dplyr")
conflicted::conflict_prefer(name = "filter", winner = "dplyr")

# For color configuration
library(wesanderson)

# Avoid the scientific notation
options(scipen = 999)
```

Now let's download the power curve of the sample NREL onshore wind turbine which has 3.3 MWs of rated power and 148 meter rotor diameter.

```
nrel_pc_url = "https://raw.githubusercontent.com/NREL/turbine-models/master/Onshore/2023NREL_Bespoke_3.3
nrel_pc = read.csv(nrel_pc_url)

head(nrel_pc, n = 15)
```

```
##    Wind.Speed..m.s. Power..kW.    Cp....
## 1              0.00          0 0.0000000
## 2              0.25          0 0.0000000
## 3              0.50          0 0.0000000
## 4              0.75          0 0.0000000
## 5              1.00          0 0.0000000
## 6              1.25          0 0.0000000
## 7              1.50          0 0.0000000
## 8              1.75          0 0.0000000
## 9              2.00          0 0.0000000
## 10             2.25          0 0.0000000
## 11             2.50          0 0.0000000
## 12             2.75          0 0.0000000
## 13             3.00          0 0.0000000
## 14             3.25        138 0.3803031
## 15             3.50        173 0.3836384
```

Select only wind speed and power. Rename them and subset only power values other than 0.

```
nrel_pc = nrel_pc %>%
  select(1,2) %>%
  rename(WindSpeed = 1, Power = 2) %>%
  filter(Power != 0)

head(nrel_pc, n = 15)
```

```
##    WindSpeed Power
## 1       3.25   138
## 2       3.50   173
## 3       3.75   223
## 4       4.00   286
## 5       4.25   358
## 6       4.50   440
## 7       4.75   529
## 8       5.00   623
## 9       5.25   722
## 10      5.50   827
## 11      5.75   941
## 12      6.00  1069
## 13      6.25  1211
## 14      6.50  1367
## 15      6.75  1535
```

Since we will discuss the statistical distributions and their limitations in the wind energy applications, we should have a wind speed data. For this purpose, we've decided to use Riso met mast located at the DTU

Riso Campus, Denmark. This met mast has measurements period includes 12 years which starts from 1995 and extends up to 2007. Wind speed has been measured at 44, 77, 125 meter heights above from the ground. We will use the top anemometers wind speed measurements in this study. Here you can reach to the data by clicking **this.**. Although it has CC 4.0 license, I can not add the data to the repository because of the file size limits of the GitHub. So that, you should've download the data manually.

```
library(ncdf4)
dtu_riso_metmast = nc_open("risoe_m_all.nc")

#Let's inspect the variables and the dimensions found in the netcdf file.
names(dtu_riso_metmast[["var"]])
```

```
##  [1] "ws44"     "ws44_qc"  "ws77"     "ws77_qc"  "ws125"    "ws125_qc"
##  [7] "wd77"     "wd77_qc"  "wd125"    "wd125_qc" "t003"     "t003_qc"
## [13] "t044"     "t044_qc"  "t118"     "t118_qc"  "td01"     "td01_qc"
## [19] "rain"     "rain_qc"  "press"    "press_qc" "rhum"     "rhum_qc"
## [25] "grad"     "grad_qc"
```

```
names(dtu_riso_metmast[["dim"]])
```

```
## [1] "time"
```

```
# Let's extract the wind speed at 125 meter height.
ws125 = ncvar_get(dtu_riso_metmast, "ws125")
```

Let's extract the time and covert it into human readable format. Metadata about the time variable states that it is stored as "minutes since 1995-11-20 16:25:00".

```
time = as.POSIXct(ncvar_get(dtu_riso_metmast, "time")*60,
                  origin = "1995-11-20 16:25:00", tz = "Europe/Copenhagen")
```

Although it's a fact that one year data is not enough for the wind resource studies, this study aims to show the limitations of the parametric statistical distributions hence, we have chosen the complete year 1997 and taken the subset of the data.

```
ws125_dtu = data.frame(time, ws125) %>%
  filter(year(time) == 1997) %>%
  rename(DateTime = 1,
         WindSpeed = 2)

summary(ws125_dtu)
```

```
##     DateTime                       WindSpeed
##  Min.   :1997-01-01 00:05:00.000   Min.   : 0.15
##  1st Qu.:1997-04-02 06:55:00.000   1st Qu.: 4.95
##  Median :1997-07-02 13:05:00.000   Median : 7.43
##  Mean   :1997-07-02 13:00:36.856   Mean   : 7.78
##  3rd Qu.:1997-10-01 19:05:00.000   3rd Qu.:10.25
##  Max.   :1997-12-31 23:55:00.000   Max.   :26.01
##                                    NA's   :22
```

We have 22 NA values and we have to omit the NA values before proceeding. Our recovery rate is:

```r
1-(sum(is.na(ws125_dtu$WindSpeed))/nrow(ws125_dtu))
```

```
## [1] 0.9995814
```

which is very close to the 1 which is 0.9995814. So it's okay to omit these values and proceed to the calculations.

```r
ws125_dtu = na.omit(ws125_dtu)
```

Since one of the main goals of this document is investigating the comparison between non-parametric and parametric estimation in the annual energy production calculation of a site, we should have a power curve function as well. For this purpose, loess function of R has been used. Span and degree parameters of the loess model function have also been searched over a grid.

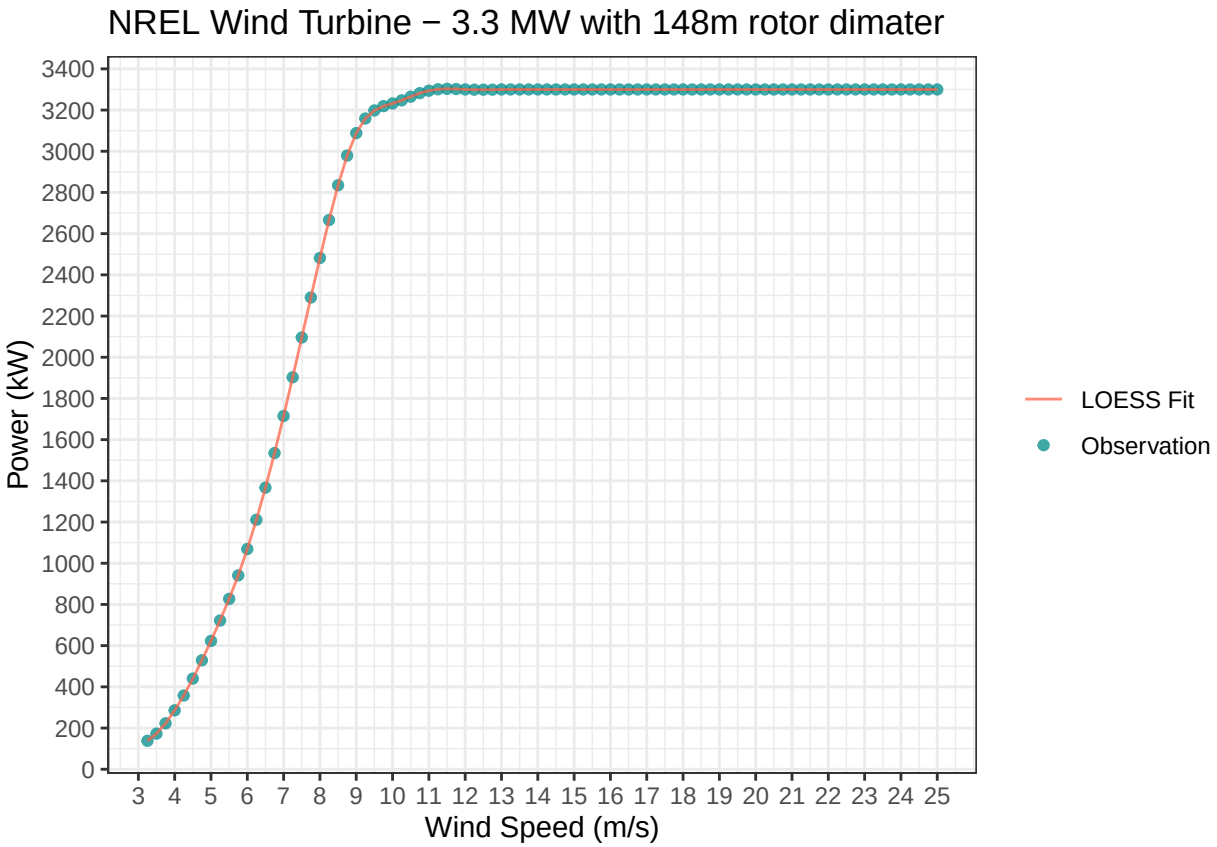
```r
#Let's create the hyperparameter grid.
grid_search = expand.grid(degree = 0:2,
                          span = c(seq(0.05, 0.1, 0.01),
                                   seq(0.15, 0.5, 0.05))) %>%
  mutate(RMSE = NA_real_)

for (i in 1:nrow(grid_search)) {

  nrel_model = loess(Power ~ WindSpeed,
                     data = nrel_pc,
                     degree = grid_search[i,1],
                     span = grid_search[i,2])

  rmse = nrel_pc %>%
    mutate(Power_P = predict(nrel_model, newdata = nrel_pc)) %>%
    summarise(RMSE = sqrt(mean((Power_P - Power)^2))) %>%
    pull()

  grid_search[i, 3] = rmse

}

#Best parameters are:
(loess_best_parameters = grid_search[which.min(grid_search$RMSE),])
```

```
##   degree span                RMSE
## 3      2 0.05 0.000000000001732742
```

```r
nrel_model = loess(Power ~ WindSpeed,
                   data = nrel_pc,
                   degree = loess_best_parameters$degree,
                   span = loess_best_parameters$span)
```

Let's plot the best result:

```r
nrel_pc %>%
  mutate(Power_P = predict(nrel_model, newdata = nrel_pc)) %>%
  ggplot(aes(x = WindSpeed)) +
  geom_point(aes(y = Power, color = "Observation"), alpha = 0.75) +
  geom_line(aes(y = Power_P, color = "LOESS Fit"), alpha = 0.75) +
  scale_x_continuous(name = "Wind Speed (m/s)", breaks = seq(0, 25, 1)) +
  scale_y_continuous(name = "Power (kW)", breaks = seq(0, 3400, 200)) +
  scale_color_manual(values = c("Observation" = "cyan4",
                                "LOESS Fit" = "tomato")) +
  guides(color = guide_legend(override.aes = list(shape = c(NA, 19), linetype = c(1,NA)))) +
  ggtitle(label = "NREL Wind Turbine - 3.3 MW with 148m rotor dimater") +
  theme_bw() +
  theme(legend.title = element_blank())
```



While the cyan colored points represents the real data taken from NREL wind power curve, tomato line indicates the curve fitted to the predictions. Since we have a power curve model that overfits the real data (taken from NREL), it's time to calculate the annual energy production of the virtual wind turbine if it would be erected directly on the location and the height (125 m) of the met mast found at the DTU Riso Campus.

```r
ws125_dtu = ws125_dtu %>%
  mutate(Power_P = predict(nrel_model, newdata = WindSpeed)) %>%
  # Since the nrel_model will create NA values for the wind speed values outside
  # the input range (WindSpeed<3.25 & WindSpeed >25), let's replace them with 0.
  mutate(Power_P = replace_na(Power_P, 0))
```

```r
# Let's calculate the annual energy production (AEP) in MWh.
# It should'nt be forgotten that the data has 10 minutes time resolution
# That's why we're dividing the result to 6.
(aep_pc = sum(ws125_dtu$Power_P)/1000/6)
```

```
## [1] 16604.14
```

```r
# Let's calculate the capacity factor of this virtually erected 3.3 MW WTG
(cf_pc = aep_pc/nrow(ws125_dtu)/3.3*6)
```

```
## [1] 0.574652
```

We've calculated the AEP by applying the power curve model we've created to the each wind speed in each row in time series. We'll call it power curve approach since then. It could be assumed as the grand truth by neglecting the losses and uncertainties. Now let's calculate the AEP with the most popular statistical distribution in wind energy sector as well. We are going to fit Weibull distribution with two different fitting methods. While the first one is maximum likelihood estimation, second one is moment matching estimation.

#Weibull Distribution - (The Famous One)

The Weibull distribution is commonly used in the wind energy sector due to its simplicity and historical precedence. It provides a convenient and relatively straightforward way to represent wind speed distributions for resource assessment and energy production calculations. Many commercial software packages also incorporate the Weibull distribution because it allows for quick and efficient calculations, making it easier to perform wind energy assessments on a large scale.

Weibull can be described with the two parameters named as shape (k) and scale (a). Since it is a parametric distribution and quite good at representing the nature of the wind speed over a time for a location, it provides a common framework for wind resource assessment and energy production calculations. This standardization allows for easier comparison of results across different projects, and locations. For instance, anyone can calculate the AEP of a point by just only knowing the parameters of the weibull and the power curve function of the desired wind turbine for that location.

For the statistical distribution fitting we'll use fitdistrplus library.

```r
library(fitdistrplus)

# First method for fitting - Maximum likelihood estimation
weibull_mle = fitdist(data = ws125_dtu$WindSpeed,
                      distr = "weibull",
                      method = "mle")

# Second method for fitting - Maximum goodness of fit estimation
weibull_mge = fitdist(data = ws125_dtu$WindSpeed,
                      distr = "weibull",
                      method = "mge",
                      gof = "KS")
```

Since we fit the weibull distributions with different methods, and calculated the shape and scale parameters of the weibull distributions with different methods it's time to evaluate those fits by comparing them with the empirical distribution.

Empirical distribution is derived from the observed data, and it represents the frequencies or proportions of different wind speeds in the dataset. So, it provides and actual representation of the data collected. It can be directly calculated with the ecdf() function found in base R, but we will do it step by step to better understand what it actually is.

```
empirical_df = ws125_dtu %>%
  mutate(Bin = cut(WindSpeed, breaks = seq(0, max(WindSpeed) + 0.5, by = 0.5))) %>%
  group_by(Bin) %>%
  summarise(Freq = n()) %>%
  ungroup() %>%
  mutate(
    #Probabilities can be calculated by dividing each freq with the total freq
    Prob = Freq / sum(Freq),
    #Cumulative probabilities can be calculated by cumulative sum
    CDF = cumsum(Prob),
    Bin = as.character(Bin),
    #Lower bin values as numeric with the help of some gibberish REGEX
    BinLower = as.numeric(str_extract(Bin, "\\d+\\.?\\d*(?=,)")),
    #Upper bin values as numeric with the help of some gibberish REGEX
    BinUpper = as.numeric(str_extract(Bin, "(?<=,)\\s*\\d+\\.?\\d*")),
    #Middle points of the bins
    WindSpeed = (BinLower+BinUpper)/2)
```

Now we have empirical distributions' PDF and CDF values for the bins that have 0.5 m/s width.

Let's evaluate if we've calculated it correct. Let's compare the calculation above with the R's base ecdf() function. To be able to calculate the CDF after applying the ecdf function to the data, WindSpeed values that separates the bins should be used, not the WindSpeed values at the middle points of the bins.

```
ws125_dtu_ecdf = ecdf(ws125_dtu$WindSpeed)

empirical_df = empirical_df %>%
  mutate(CDF_R = ws125_dtu_ecdf(BinUpper))

#Check if we've calculated the CDF true?
nrow(empirical_df) == sum(round(empirical_df$CDF, 3) == round(empirical_df$CDF_R, 3))
```

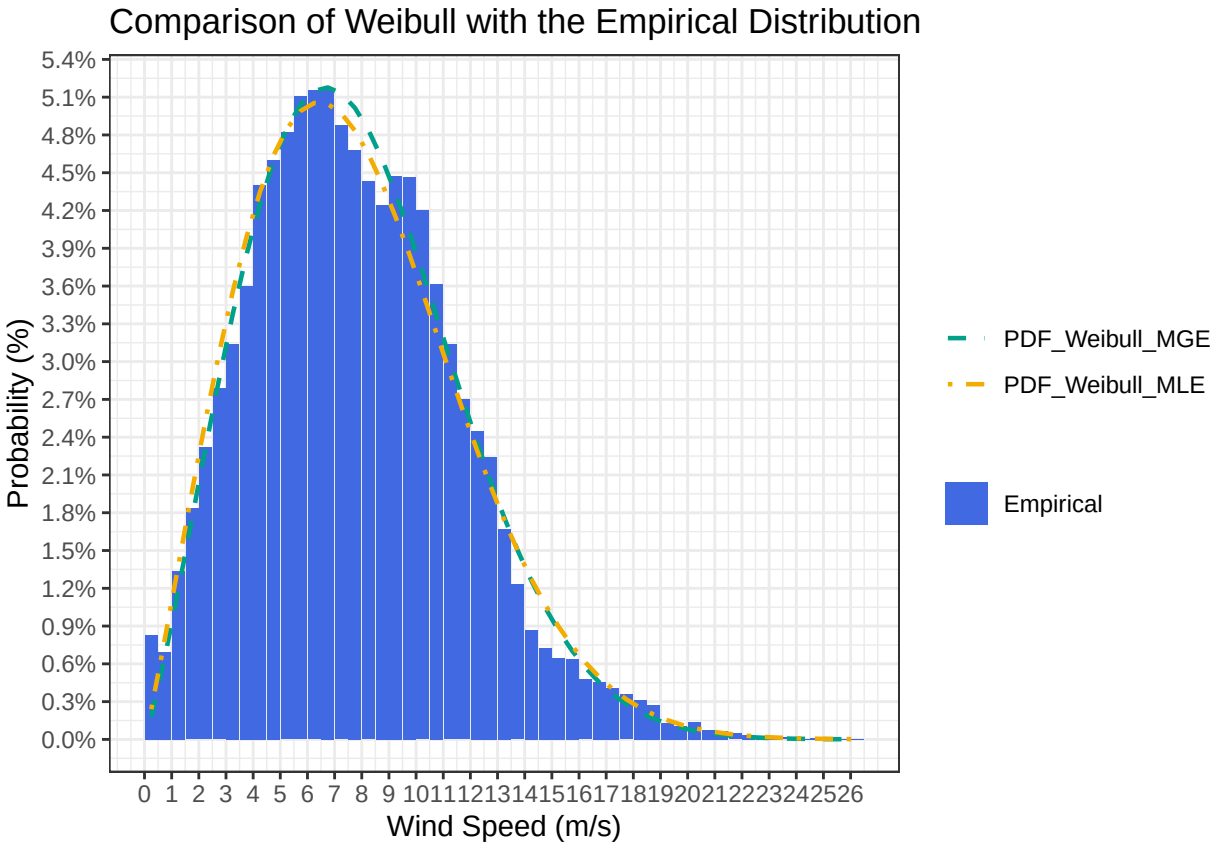
```
## [1] TRUE
```

All the CDF and CDF_R values are equal to each other! So that we've calculated the cdf for empirical distribution correct.

Now let's plot the probability density functions calculated with the weibull distributions and compare them with the empirical distribution. First, we need to calculate the densities/probabilities for the desired wind speeds for each weibull distribution we've calculated above. In empirical distribution, we have used a dataset from 0.5 to 18.5 wind speed with 0.5 m/s step. Now with the dweibull function of base R, we can calculate the density for the desired wind speeds and we would like to calculate the probabilities for each bin for visualization. To do this, we have to use the wind speed that represents each bin. Therefore, the middle points of each bin can be used for a better visualization.

```
empirical_df_weibull = empirical_df %>%
  mutate(Density_Weibull_MLE = dweibull(empirical_df$WindSpeed,
                                        shape = weibull_mle$estimate["shape"],
                                        scale = weibull_mle$estimate["scale"]),
         PDF_Weibull_MLE = Density_Weibull_MLE/sum(Density_Weibull_MLE),
         Density_Weibull_MGE = dweibull(empirical_df$WindSpeed,
                                        shape = weibull_mge$estimate["shape"],
                                        scale = weibull_mge$estimate["scale"]),
```

```
        PDF_Weibull_MGE = Density_Weibull_MGE/sum(Density_Weibull_MGE))

empirical_df_weibull %>%
    select(WindSpeed, starts_with("PDF"), Prob) %>%
    pivot_longer(names_to = "PDF_Type", values_to = "Value", starts_with("PDF")) %>%
  ggplot(.) +
  geom_col(aes(x = WindSpeed, y = Prob, fill = "Empirical"), position = position_dodge(width =0.8)) +
  geom_line(aes(x = WindSpeed, y = Value, color = PDF_Type, linetype = PDF_Type), lwd = 0.8) +
  scale_color_manual(values = wes_palette("Darjeeling1", 3)[c(2,3)]) +
  scale_fill_manual(values = "royalblue") +
  scale_linetype_manual(values = c(2,4)) +
  scale_y_continuous(name = "Probability (%)",
                     n.breaks = 20,
                     labels = scales::percent_format(accuracy = 0.1)) +
  scale_x_continuous(name = "Wind Speed (m/s)", breaks = seq(0, 26, 1)) +
  ggtitle(label = "Comparison of Weibull with the Empirical Distribution") +
  theme_bw() +
  theme(legend.title = element_blank())
```



Weibull and It's Limitations

As it can be clearly seen, all the weibull distributions fitted with different methods have similar curves. But, the empirical wind speed distribution has multimodality (multiple peaks) between 6.5 - 7 m/s and 9 - 10 m/s wind speeds and it seems like the weibull distributions doesn't represent these peaks. So do you think

that they represent empirical distribution (observed data) well? It's obvious to state that the distributions don't fit well by just observing the plot. So we can state that weibull distribution has some limitations and these limitations can be expressed as:

1. Simplicity: The Weibull distribution assumes a single-peaked, symmetric distribution, which might not always match real-world wind data. **2. Lack of flexibility:** The Weibull distribution has limited flexibility in modeling different shapes of wind speed distributions. It cannot capture characteristics like multimodality (multiple peaks), skewness, or heavy tails that might be present in actual wind data. **3. Data Fit Issues:** Fitting a Weibull distribution to observed wind data can be challenging. Real-world wind data often deviates from the Weibull assumption, leading to poor fits and inaccurate representations of the wind speed distribution. This can result in significant errors when estimating energy production. **4. Overestimation of low wind speeds:** The Weibull distribution tends to overestimate the frequency of low wind speeds. This can lead to overly optimistic energy production estimates, which can be problematic for project planning and financing. **5. Underestimation of high wind speeds:** Conversely, the Weibull distribution may underestimate the frequency of high wind speeds. This can lead to conservative energy production estimates, potentially causing project developers to oversize equipment or miss out on revenue.

On the other hand, for all of the limitations mentioned above about the weibull distribution which is a parametric distribution, let's focus on the non-parametric estimation. For this purpose, kernel density estimation (KDE) algorithm has been chosen.

Kernel Density Estimation (KDE)

KDE calculates the PDF by placing a kernel (a smooth, continuous function, usually a Gaussian) on each data point and summing these kernels to create a smooth, continuous density estimate. The bandwidth parameter determines the width of the kernels, affecting the smoothness of the estimate. Now let's talk about the advantages of KDE over a parametric distribution:

1. Flexibility: It can handle wind speed data with complex and non-standard distributions. **2. No Distribution Assumptions:** The Weibull distribution makes specific assumptions about the shape of the wind speed distribution, which may not always hold true. **3. Accuracy in energy production:** By providing a more accurate representation of the wind speed distribution, KDE can lead to more precise estimates of annual energy production for wind turbines.

For KDE, we'll use density() function of base R. We'll use n, width, from and to arguments of this function. These arguments is configured based on the WindSpeed variable of the empirical_df dataframe created before on this study. It should be highlighted that the WindSpeed variable consists of the middle points of the each bin created before. So it will begin with the 0.25 and will continue through the 18.25. This function directly calculates the frequencies (densities) for the wind speeds we've provided.

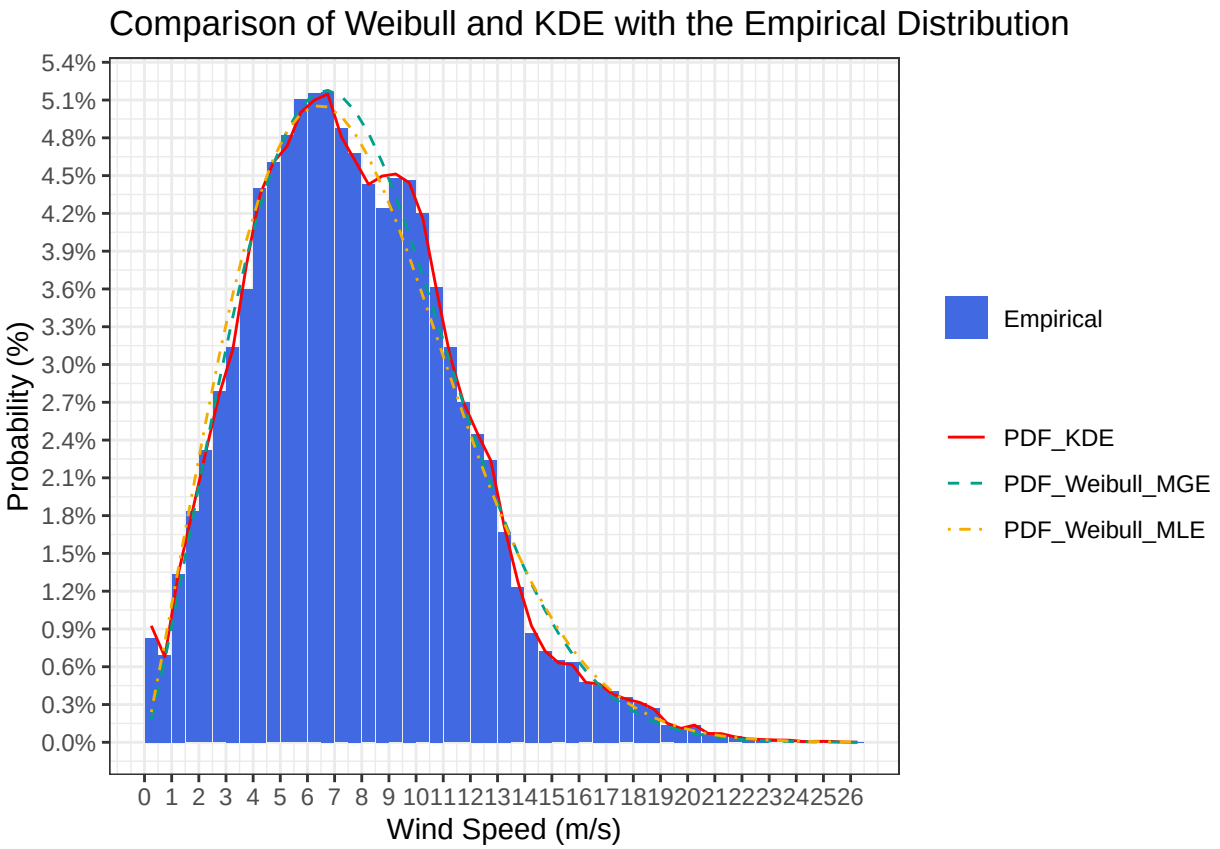
```
kde_method = density(x = ws125_dtu$WindSpeed,
                     n = nrow(empirical_df_weibull),
                     width = 0.5,
                     # From the middlepoint of the first bin
                     from = empirical_df$WindSpeed[1],
                     # To the middlepoint of the last bin
                     to = empirical_df$WindSpeed[nrow(empirical_df)])
```

Once again, after calculating the densities, now it's time to calculate the probabilities.

```
empirical_df_kde = empirical_df_weibull %>%
  mutate(Density_KDE = kde_method$y,
         PDF_KDE = Density_KDE/sum(Density_KDE))
```

Let's plot the previous PDF graph with KDE as well.

```
empirical_df_kde %>%
  select(WindSpeed, starts_with("PDF"), Prob) %>%
  pivot_longer(names_to = "PDF_Type",
               values_to = "Value",
               starts_with("PDF")) %>%
  ggplot(.) +
  geom_col(aes(x = WindSpeed,
               y = Prob,
               fill = "Empirical"),
           position = position_dodge(width =0.8)) +
  geom_line(aes(x = WindSpeed,
                y = Value,
                color = PDF_Type,
                linetype = PDF_Type)) +
  scale_color_manual(values = c(
    #For KDE color (Red)
    wes_palette("Darjeeling1", 2)[1],
    #For Weibull colors (cyan and orange)
    wes_palette("Darjeeling1", 3)[c(2,3)])) +
  scale_fill_manual(values = "royalblue") +
  scale_linetype_manual(values = c(1,2,4)) +
  scale_y_continuous(name = "Probability (%)",
                     n.breaks = 20,
                     labels = scales::percent_format(accuracy = 0.1)) +
  scale_x_continuous(name = "Wind Speed (m/s)", breaks = seq(0, 26, 1)) +
  ggtitle(label = "Comparison of Weibull and KDE with the Empirical Distribution") +
  theme_bw() +
  theme(legend.title = element_blank())
```



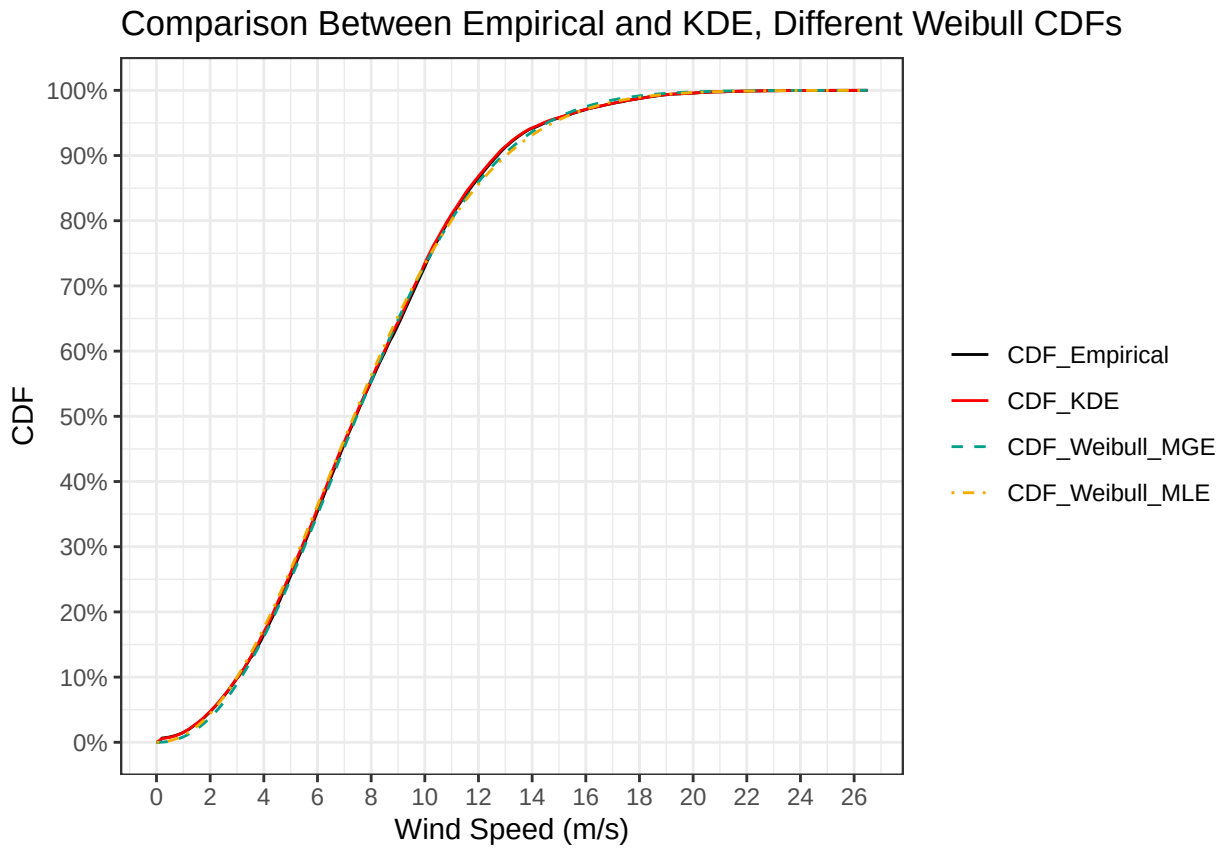
As it can be seen clearly seen in the plot, KDE non-parametric distribution represents empirical distribution much more better compared to other weibull distribution fits. Let's prove that with a statistical performance metric. Kolmogorov-Smirnov (KV) test assesses the goodness of fit between the observed data and the expected/empirical distribution (usually a theoretical or hypothesized distribution). Now, we can use KS test for evaluating whether weibull or KDE distributions represent the empirical distribution or not. For this, ks.test() function of base R will be used. This test is calculated by using the Cumulative Distribution Functions so that, CDF's have to be calculated. For weibull distributions, pweibull() functions calculates the CDF values. The KS test is a non-parametric test that compares the empirical distribution to the expected distribution. The more data points you have, the more sensitive the test can be to detecting differences between the distributions. So, CDF values will be kept in another dataset.

```
windspeed_to_calc = seq(0,26.5,0.1)

cdf_to_ks = data.frame(WindSpeed = windspeed_to_calc,
                       CDF_Weibull_MLE = pweibull(windspeed_to_calc,
                                                  shape = weibull_mle$estimate["shape"],
                                                  scale = weibull_mle$estimate["scale"]),
                       CDF_Weibull_MGE = pweibull(windspeed_to_calc,
                                                  shape = weibull_mge$estimate["shape"],
                                                  scale = weibull_mge$estimate["scale"]))

#Let's make a quick function that scales data between 0 and 1.
scaling = function(x) {
  x/sum(x)
}
```

```r
cdf_to_ks = cdf_to_ks %>%
  mutate(CDF_KDE = cumsum(scaling(density(x = ws125_dtu$WindSpeed,
                                  width = 0.1,
                                  from = 0,
                                  to = 26.5,
                                  n = nrow(cdf_to_ks))$y)),
         CDF_Empirical = ws125_dtu_ecdf(seq(0,26.5,0.1)))
```

Let's plot the CDF's of Empirical Distribution, Weibull Distribution and, KDE.

```r
cdf_to_ks %>%
  select(WindSpeed, starts_with("CDF")) %>%
  pivot_longer(names_to = "Cesit", values_to = "Deger", -WindSpeed) %>%
  ggplot(., aes(x = WindSpeed, y = Deger, color = Cesit, linetype = Cesit)) +
  geom_line() +
  scale_linetype_manual(values = c(1, 1, 2, 4)) +
  scale_color_manual(values = c(
    #For Empirical CDF color
    "black",
    #For KDE CDF color (Red)
    wes_palette("Darjeeling1", 2)[1],
    #For Weibull CDF colors (cyan and orange)
    wes_palette("Darjeeling1", 3)[c(2,3)])) +
  scale_y_continuous(name = "CDF", labels = scales::percent, breaks = seq(0,1,0.1)) +
  scale_x_continuous(name = "Wind Speed (m/s)", breaks = seq(0,26,2)) +
  ggtitle("Comparison Between Empirical and KDE, Different Weibull CDFs") +
  theme_bw() +
  theme(legend.title = element_blank())
```



It's obvious to state that all three CDF's are not quite different from each other. But, the differences can be seen especially in the ranges from 0 to 4 m/s and especially from 10 to 15 m/s which has crucial effect on the AEP calculation. It should also be noted that, Empirical CDF (black line) is followed closely by the CDF of the KDE (red line).

```
# Here KS test results can be seen;
(ks_weibull_mle = ks.test(x = cdf_to_ks$CDF_Weibull_MLE,y = cdf_to_ks$CDF_Empirical))
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  cdf_to_ks$CDF_Weibull_MLE and cdf_to_ks$CDF_Empirical
## D = 0.033835, p-value = 0.9981
## alternative hypothesis: two-sided
```

```
(ks_weibull_mge = ks.test(x = cdf_to_ks$CDF_Weibull_MGE,y = cdf_to_ks$CDF_Empirical))
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  cdf_to_ks$CDF_Weibull_MGE and cdf_to_ks$CDF_Empirical
## D = 0.052632, p-value = 0.8549
## alternative hypothesis: two-sided
```

```r
(ks_kde = ks.test(x = cdf_to_ks$CDF_KDE,y = cdf_to_ks$CDF_Empirical))
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  cdf_to_ks$CDF_KDE and cdf_to_ks$CDF_Empirical
## D = 0.015038, p-value = 1
## alternative hypothesis: two-sided
```

KS-Test Results

All three p-values are relatively large, with the smallest being 0.8549. Large p-values suggest that your data is consistent with the specified distributions. The D statistic for each test is quite small, indicating that the EDF and CDF are close in shape. Besides, D statistic that represents the maximum absolute difference between the EDF of the observed data and the CDF of the distribution we are testing and it has the smallest value in the KDE distribution. Since a smaller D value indicates a better fit because it means the EDF and CDF are closer in shape, we prove that KDE represents our data better.

AEP Calculation By Using the PDF and Power Curve Function

Now we can calculate the AEP based on these distributions and compare the results with the power curve approach which will be served as grand truth in our study.

To be able to calculate the AEP from these distributions, we have to integrate the area resulted by multiplying the wind speed PDF and power curve function. So we have to first calculate the energy equivalent of each wind speed in the PDF's by using the loess model of the power curve we've introduced above.

```r
aep_dists = empirical_df_kde %>%
  mutate(Power = predict(nrel_model, newdata = data.frame(WindSpeed = WindSpeed)),
         Power = replace_na(Power, 0))
```

Now we've calculated the power equivalent of each wind speed bin in the PDFs. It's time to calculate areas covered by each bin and sum all of them (integration). After this integration we will have calculated the hourly mean power generation of the virtual wind turbine for the location of the met mast. If we would have full complete year data and this study wouldn't be about the direct comparison of the distributions with the power curve approach introduced above, we would multiply this with the 365.25 and 24 to be able to calculate the annual energy production. But we want to compare the results and evaluate which distribution does have more realistic results. So, NA values in the data and the row number have crucial impacts on the result as well. Let's consider all of them, and instead of multiplying the integrated area with the **8760** (total hours in a year), multiply this area with a realistic hour value taken from our data. Since our data has 10 minutes of time resolution and has nrow(ws125_dtu), 52535 row after na.omit, we can directly calculate how many hours does it have by dividing the total row number to 6. It is **8755.833** hours.

```r
(data_hours = nrow(ws125_dtu)/6)
```

```
## [1] 8755.833
```

```r
(aep_weibull_mle = sum(aep_dists$Power*aep_dists$PDF_Weibull_MLE)/1000*data_hours)
```

```
## [1] 16351.17
```

```r
(aep_weibull_mge = sum(aep_dists$Power*aep_dists$PDF_Weibull_MGE)/1000*data_hours)
```

```
## [1] 16670.6
```

```r
(aep_kde = sum(aep_dists$Power*aep_dists$PDF_KDE)/1000*data_hours)
```

```
## [1] 16620.68
```

```r
(aep_powercurve = sum(ws125_dtu$Power_P)/1000/6)
```

```
## [1] 16604.14
```

```r
data.frame(PowerCurve = aep_powercurve,
           Diff_Weibull_MLE = aep_weibull_mle-aep_powercurve,
           Diff_Weibull_MGE = aep_weibull_mge-aep_powercurve,
           Diff_KDE = aep_kde-aep_powercurve,
           Ratio_Weibull_MLE = aep_weibull_mle/aep_powercurve,
           Ratio_Weibull_MGE = aep_weibull_mge/aep_powercurve,
           Ratio_KDE = aep_kde/aep_powercurve)
```

```
##   PowerCurve Diff_Weibull_MLE Diff_Weibull_MGE Diff_KDE Ratio_Weibull_MLE
## 1   16604.14        -252.9701         66.45937 16.53537         0.9847646
##   Ratio_Weibull_MGE Ratio_KDE
## 1          1.004003  1.000996
```

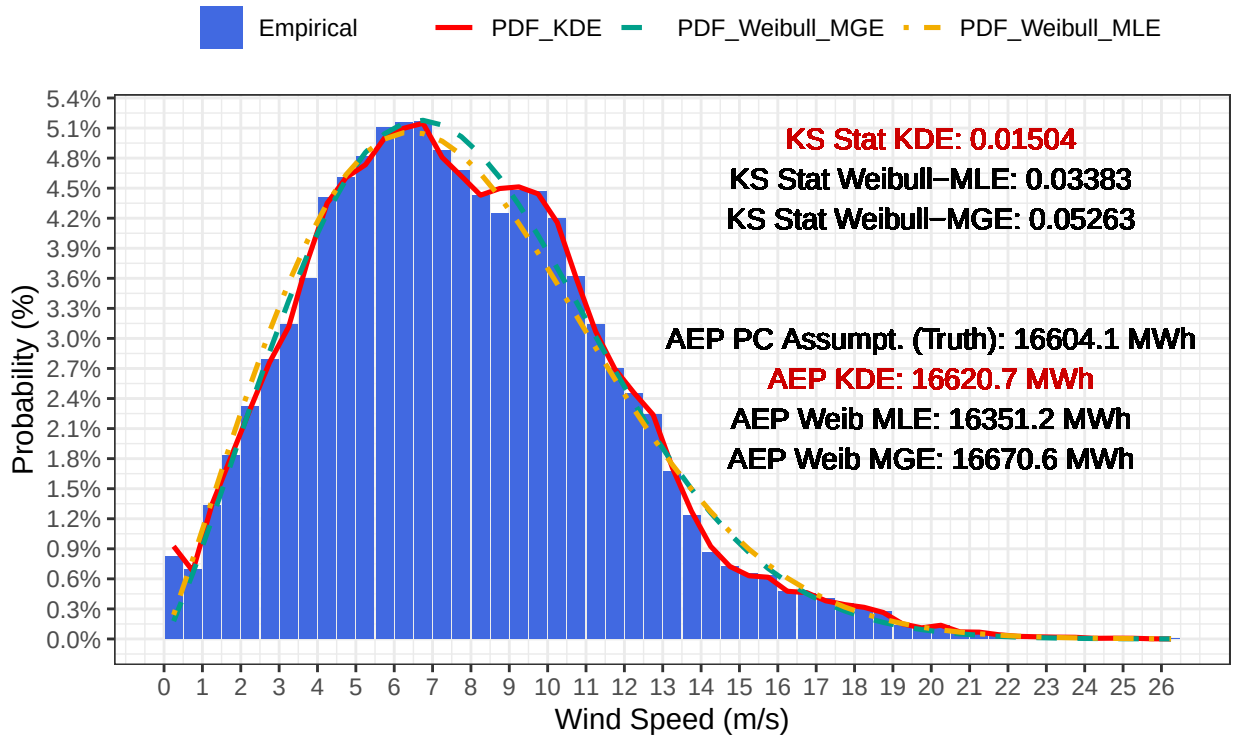
Conclusion and Discussion

As it can be clearly seen, best method is calculated as the KDE non-parametric approach with an about 16.55 MWh error for AEP calculation although the differences are quite small. Since the surrounding of the met mast is can be classified as a flat(flat landscape) coastal(water and land) site according to the DTU's website, it's clear to state that in more complex terrains there will be very different wind speed distributions and it will be very hard to fit weibull distributions to these observations. In my experience through working as wind resource asessment engineer in different companies, I have faced lots of complex sites located in Turkey and the weibull fit unfortunately doesn't represent the empirical distribution. That's why it's very important to implement other fitting methodologies or non-parametric distributions to the worldwide known softwares like WAsP, windPRO or Meteodyn.

Now let's make the final plot!

```r
# Final plot!
empirical_df_kde %>%
  select(WindSpeed, starts_with("PDF"), Prob) %>%
  pivot_longer(names_to = "PDF_Type",
               values_to = "Value",
```

```
             starts_with("PDF")) %>%
ggplot(.) +
geom_col(aes(x = WindSpeed,
             y = Prob,
             fill = "Empirical"),
         position = position_dodge(width =0.8)) +
geom_line(aes(x = WindSpeed,
              y = Value,
              color = PDF_Type,
              linetype = PDF_Type), lwd = 0.9) +
scale_color_manual(values = c(
  #For KDE color (Red)
  wes_palette("Darjeeling1", 2)[1],
  #For Weibull colors (cyan and orange)
  wes_palette("Darjeeling1", 3)[c(2,3)])) +
scale_fill_manual(values = "royalblue") +
scale_linetype_manual(values = c(1,2,4)) +
scale_y_continuous(name = "Probability (%)",
                   n.breaks = 20,
                   labels = scales::percent_format(accuracy = 0.1)) +
geom_text(aes(x = 20, y = 0.05), label = paste0("KS Stat KDE: ", round(ks_kde$statistic,5)), color =
geom_text(aes(x = 20, y = 0.046), label = paste0("KS Stat Weibull-MLE: ", round(ks_weibull_mle$statis
geom_text(aes(x = 20, y = 0.042), label = paste0("KS Stat Weibull-MGE: ", round(ks_weibull_mge$statis
geom_text(aes(x = 20, y = 0.03), label = paste0("AEP PC Assumpt. (Truth): ", round(aep_pc, 1), " MWh"
geom_text(aes(x = 20, y = 0.026), label = paste0("AEP KDE: ", round(aep_kde,1), " MWh"), color = "red3
geom_text(aes(x = 20, y = 0.022), label = paste0("AEP Weib MLE: ", round(aep_weibull_mle,1), " MWh"))
geom_text(aes(x = 20, y = 0.018), label = paste0("AEP Weib MGE: ", round(aep_weibull_mge,1), " MWh"))
scale_x_continuous(name = "Wind Speed (m/s)", breaks = seq(0, 26, 1)) +
ggtitle(label = "PDF's of Weibull and KDE with the Empirical Distribution",
        subtitle = "Data: Met Mast Located at DTU Riso Campus @125m height (1997)") +
theme_bw() +
theme(legend.title = element_blank(),
      legend.position = "top")
```




```
ggsave("final_plot.jpeg", dpi = 300, height = 9/1.25, width = 16/1.25)
```