PHOTOVOLTAIC PLANT PERFORMANCE

MSA 6450 – Advanced Data Analysis Project

Nick Wawee

Bowling Green State University | 07/09/21 | nwawee@bgsu.edu

Motivation

Electrical power generated from solar panels could be the most accessible form of renewable energy. The performance of these photovoltaic systems is at the pinnacle of establishing a sustainable future in energy. Identifying factors that impact these systems are key to optimizing said future. The motivating questions behind this project are as follows:

1. Can we identify faulty or sub optimally performing equipment?
    - Reveal panels that connect to inverters that have performance issues as a segway to identify the root cause of the problem
2. Can we model the DC power output based on module temperature and irradiation?
    - Simulate panel performance given these variables and assess predictor impact on DC power
    - Identify low DC power was lower than normal indicating a need for maintenance
3. Can we predict the AC power generation for next couple of days?
    - Useful for grid management to optimize power draw from solar plant

Faulty equipment will be classified by comparing the performance of each inverter to spot any outliers. Identifying maintenance periods enables planning of time periods where the panels connected to the inverter needs attention. This planning will be scheduled such that it will minimize equipment downtime during peak irradiation times to maximize efficiency. After predicting the power generation for the next couple of days, the plant will be utilized as a source of power optimally by injecting power into the grid at peak times.

Data Description

The data includes measurements from inverters and weather sensors from a solar power plant in India that was connected to the electric grid [1]. The dataset from the first powerplant consists of 22 inverters and spanned from 05/15/20 – 06/17/20 at 15-minute intervals for 34 days. Weather data from a sensor at this plant was also collected at the same time intervals. Listed below are the variable descriptions:

- *DATE_TIME:* The date and time of the 15-minute interval that the observation was recorded
- *PLANT_ID:* Identification of which plant the inverter or weather sensor belongs to
- *SOURCE_KEY:* Inverter identifier
- *DC_POWER:* Amount of DC power coming in to the inverter in kW
- *AC_POWER:* Amount of AC power exiting the inverter in
- *DAILY_YIELD:* Cumulative sum of power generated up until that point of time in that day
- *TOTAL_YIELD:* Cumulative sum of power generated up until that point of time
- *AMBIENT_TEMPERATURE:* Temperature of surroundings in Celsius
- *MODULE_TEMPERATURE:* Sensor panel temperature in Celsius
- *IRRADIATION:* Amount of irradiation at the 15-minute interval, kW/m^2

Exploratory Data Analysis

Figure 1 displays all measurements of all inverters in the plant. As shown, the AC power, DC power, and daily yield all have several observations where it is nighttime, and the sun is not out. There appears to be a bimodal distribution in total yield measurements, the smaller mode is from the underperforming inverters which are shown as outliers in **Figure 2**. The two worse performing inverters are identified by the source keys 1BY6WEcLGh8j5v7 and bvBOhCH3iADSZry. There appears to be a right skew in all the ambient temperature, module temperature, and irradiation at the 15-minute measurement intervals.

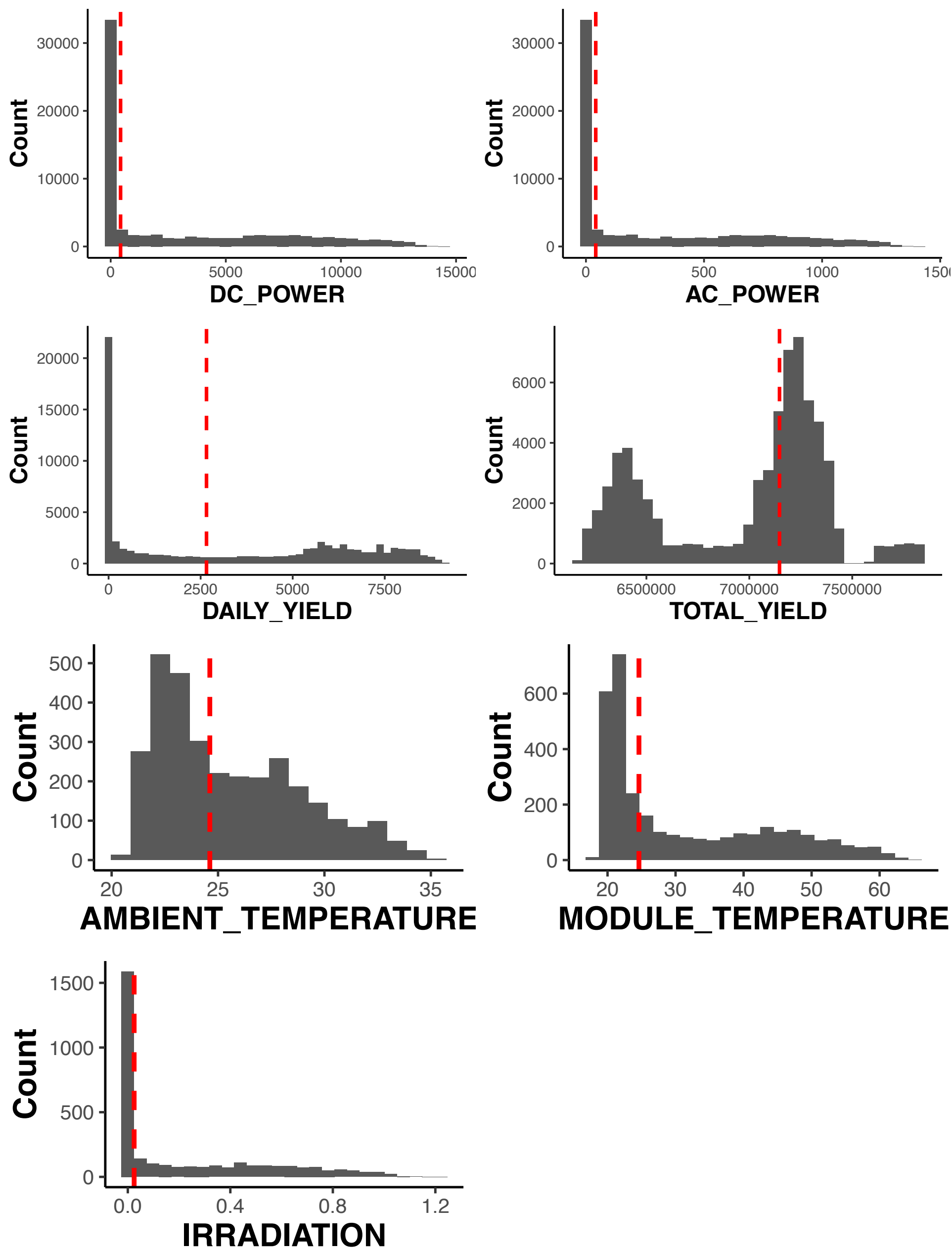


Figure 1: Distribution of Measurements at 15-minute Intervals

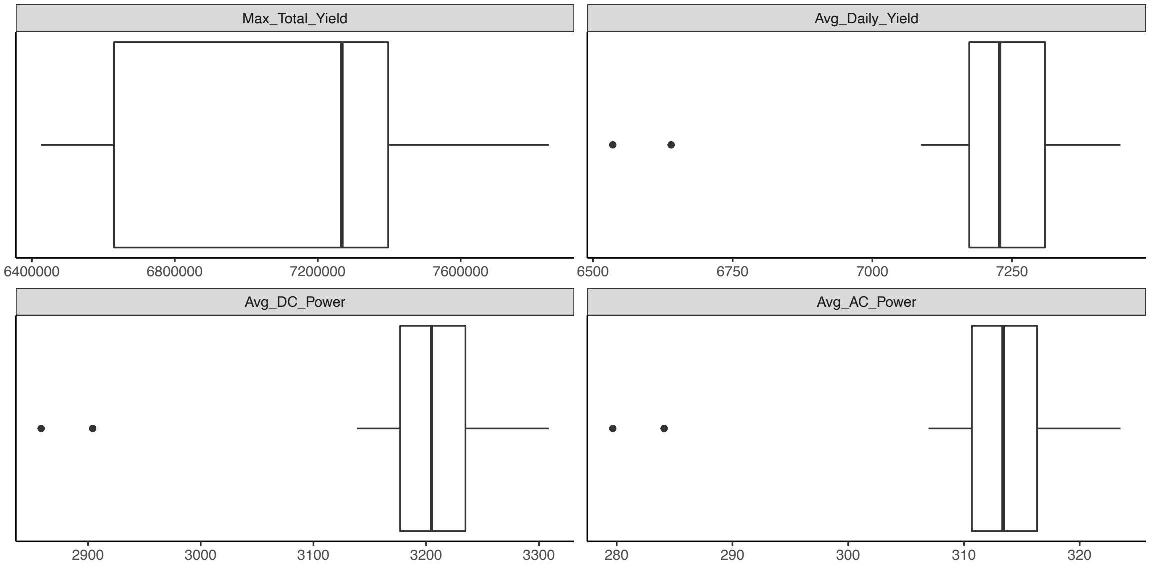


Figure 2: Inverter Distributions

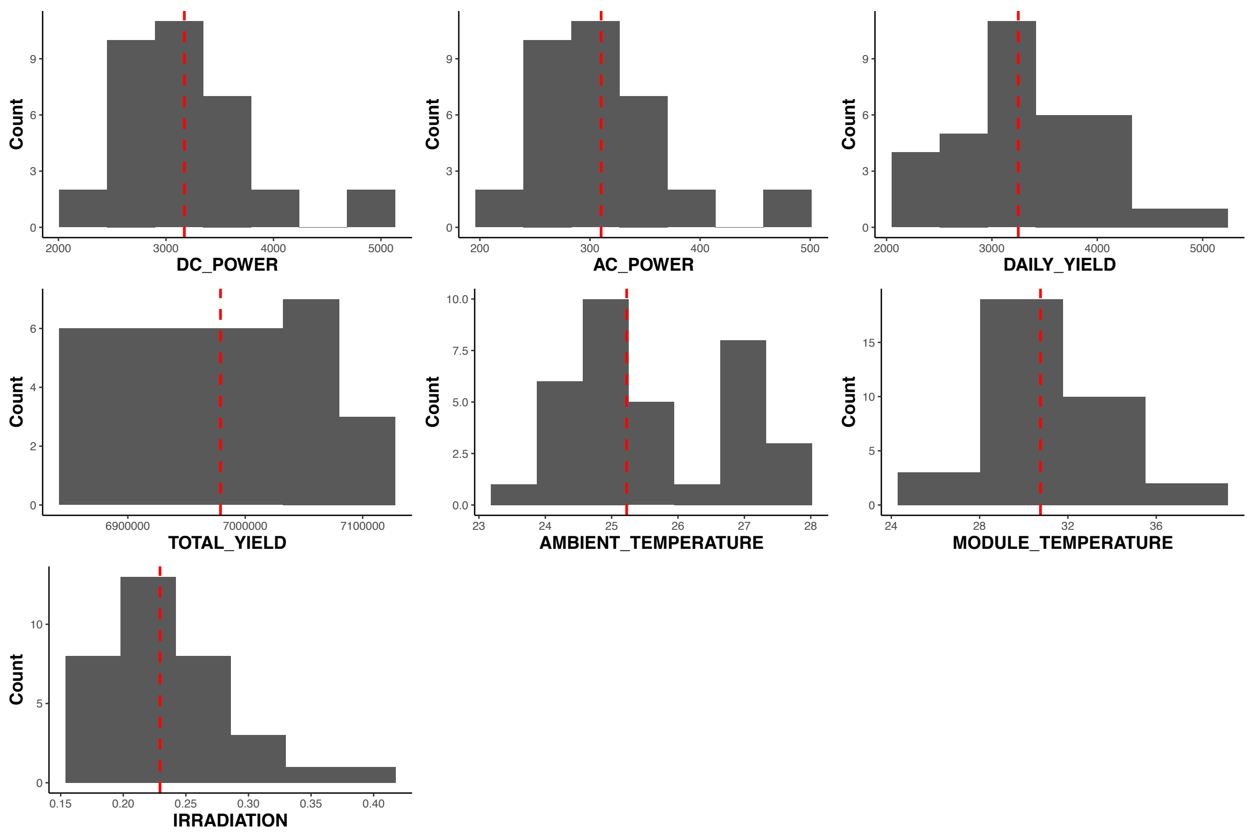


Figure 3: Daily Distributions of Equipment and Weather

When inspecting **Figure 3** to see how the daily distributions look, the irradiation and module temperature remain to have a right skew, but the ambient temperature appears to a have a second mode.

Advanced Data Analysis

Linear Regression

A multiple linear regression approach was used to model the DC power output based on module temperature and irradiation which yielded a RMSE of 585 kW. A simulation was conducted by estimating the population standard deviation, using the sample standard deviation, and randomly drawing from a χ^2 distribution 1000 times. The population standard deviations were used to simulate both population regression coefficients. The distribution of both coefficients is displayed below in **Figure 4**, which allows for physical interpretations of how changes in module temperature and irradiation impact DC power output as well as the expected range and frequency of both.

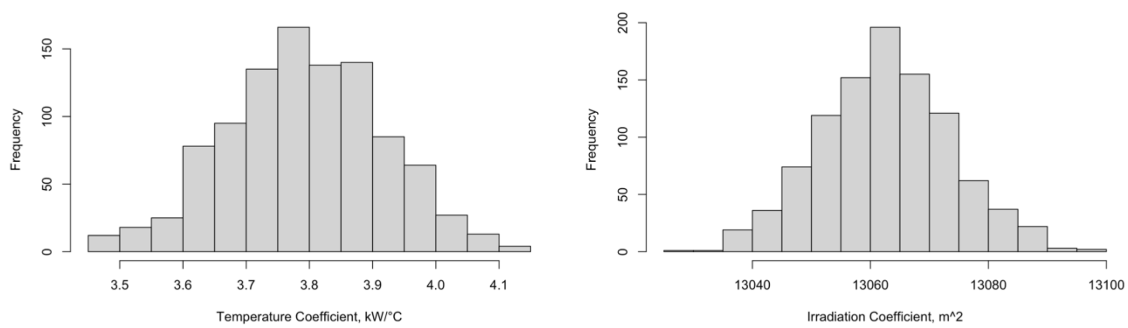


Figure 4: Distributions of Linear Regression Coefficients

The linear regression provided a simplistic way of demonstrating the relationship between DC power generated from the irradiance and module temperature. However, it has been shown that non-linear behavior in DC output occurs at higher temperatures [2] [3]. To account for this behavior, a non-linear model was fit to the data.

Non-Linear Regression

The non-linear behavior of the DC power output as a function of irradiance and module temperature was derived and empirically determined by [2] [3]. The four parameters were estimated via iterations to minimize the error of the model. The final equation is shown below.

$$P(t) = 11.1E(t)(1 + 8.30 * 10^{-10}\left(T(t) + \frac{E(t)}{800}(-2.91 * 10^{8} - 20) - 25\right) + 6.56 * 10^{-2}\, ln\big(E(t)\big))$$

Where E(t) [=] W/m^2 and T [=] °C. **Figure 5** shows the predictive values of DC power at different inverters and 15-minute intervals, as well as the actual values as a function of irradiance and module temperature. The predictions as a function of temperature have more variation than the predictions as a function of irradiance. The RMSE of this model is 549 kW, which is an improvement from the linear model.

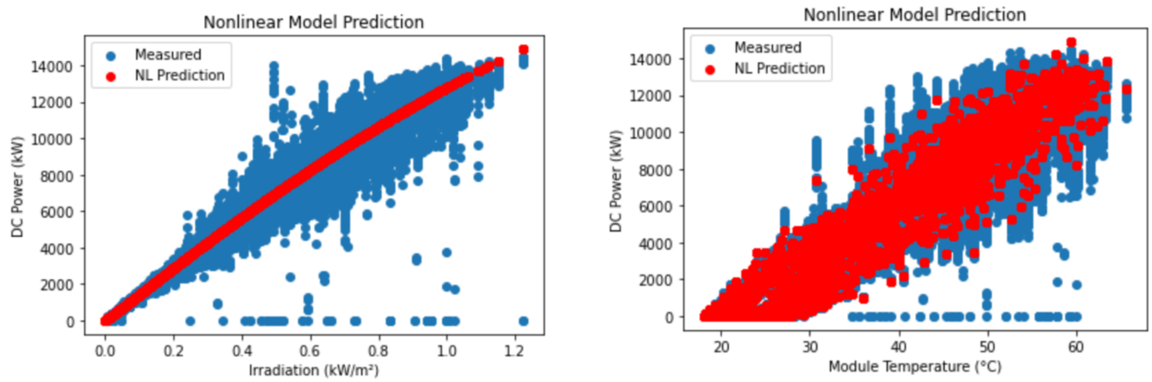


Figure 5: Non-linear Predictions of DC Power

By utilizing the model that fit the data a bit better, the sub-optimal performing equipment was identified by employing a residual threshold of 5000 kW. The residuals above this threshold were considered as underperforming. **Figure 6** depicts this classification of 84 faulty measurements as a function of irradiation.

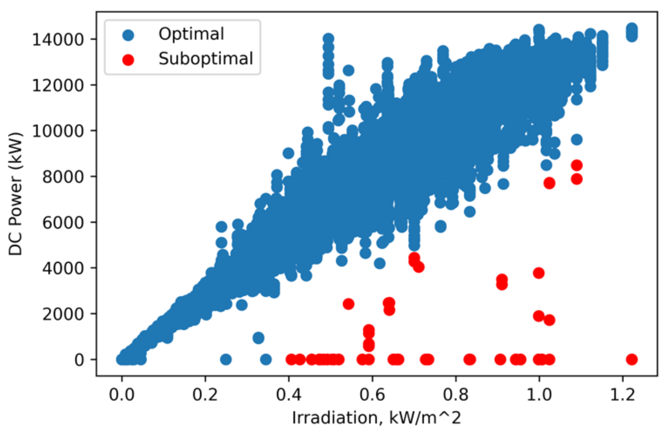


Figure 6: Faulty Equipment Classification

The two underperforming inverters discussed in the previous EDA section, 1BY6WEcLGh8j5v7 and bvBOhCH3iADSZry, were equally responsible for 46 of these measurements. The other measurements belonged to 11 of the inverters, which indicates that maintenance or cleaning may need to be performed on the panels connected to these inverters.

Time Series Predictions of Total AC Power

To provide an estimate of how much AC power is supplied to the electric grid, a seasonal time series model was fit to the total sum of AC power output of the inverters. This estimate of power is prior to the losses incurred when injecting power into the grid and it is assumed that the power injection losses are accounted for. A model that takes into consideration the periodic behavior solar panel power generation was warranted. The Facebook Prophet model was created to estimate time series that are periodic in nature [4]. Here it was used to approximate the total AC power from the plant. **Figure 7** displays the training data used to create the model, test data that the model generated forecasts on, and the original forecast that the model generated on the training data.

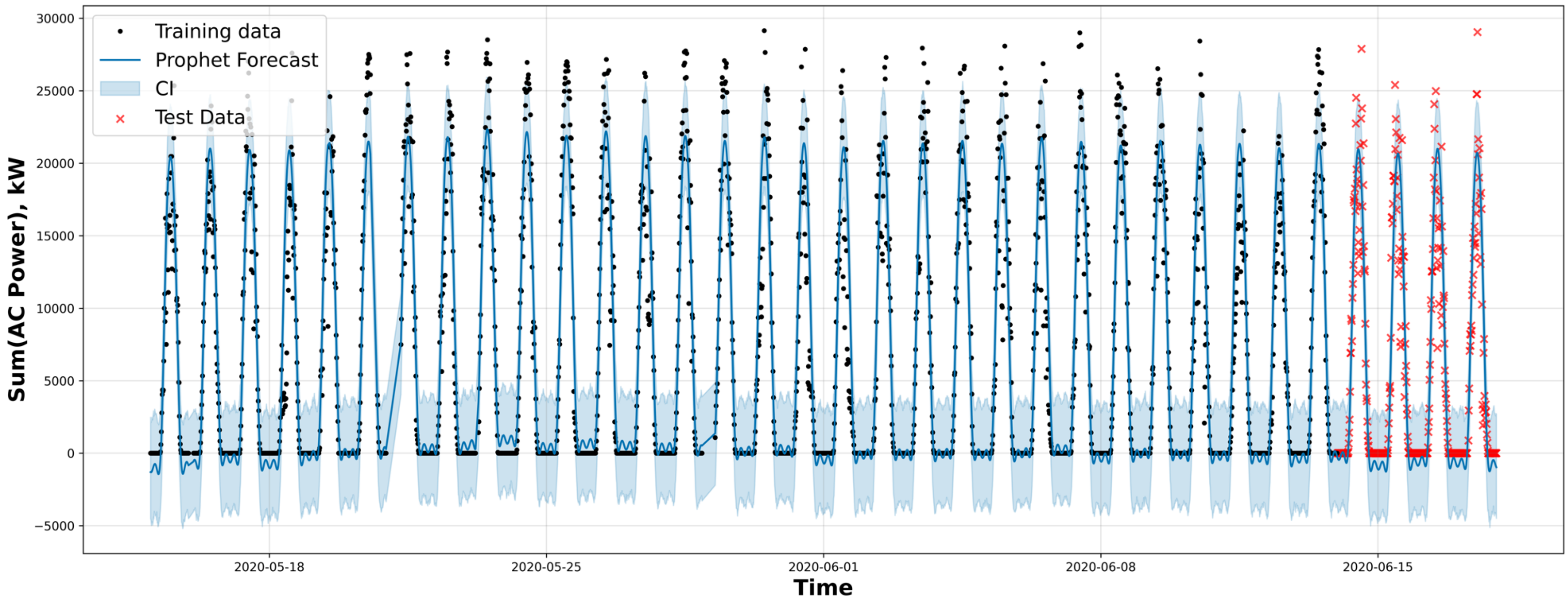


Figure 7: Prophet Forecast of AC Power Plant Output, RMSE = 2890 kW on the Test Data

Conclusion

Sub-optimal performing equipment was found by inspecting the distributions of power output at the inverter level. The two sub optimal inverters, 1BY6WEcLGh8j5v7 and bvBOhCH3iADSZry, were found as large residuals when modeling the DC power as a function of irradiation and module temperature. A linear and non-linear model was fit to the same data. The linear model provided interpretable regression coefficients while the non-linear model was a better estimate of DC power— with a RMSE of 549 kW. Residual analysis identified that solar panels connected to half of the inverters may need maintenance and are up for inspection. The Facebook Prophet model was used to predict the aggregate AC power of the plant for better grid management and yielded an RMSE of 2890 on unseen data.

References

[1] A. Kannal, "Solar Power Generation Data," 18 August 2020. [Online]. Available: https://www.kaggle.com/anikannal/solar-power-generation-data. [Accessed July 2021].

[2] W. E. B. J. A. K. D. L. King, "PHOTOVOLTAIC ARRAY PERFORMANCE MODEL," August 2004. [Online]. Available: https://energy.sandia.gov/wp-content/gallery/uploads/043535.pdf. [Accessed July 2021].

[3] A. P. A. P. K. K. S. V. A. M. I. P. N. Hooda, "PV Power Predictors for Condition Monitoring," in *IEEE*, Sydney, Australia.

[4] B. L. Sean J Taylor, "Forecasting at scale," *PeerJ Preprints,* vol. 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2, 2017.

Appendix A – EDA and Linear Regression

Nick Wawee

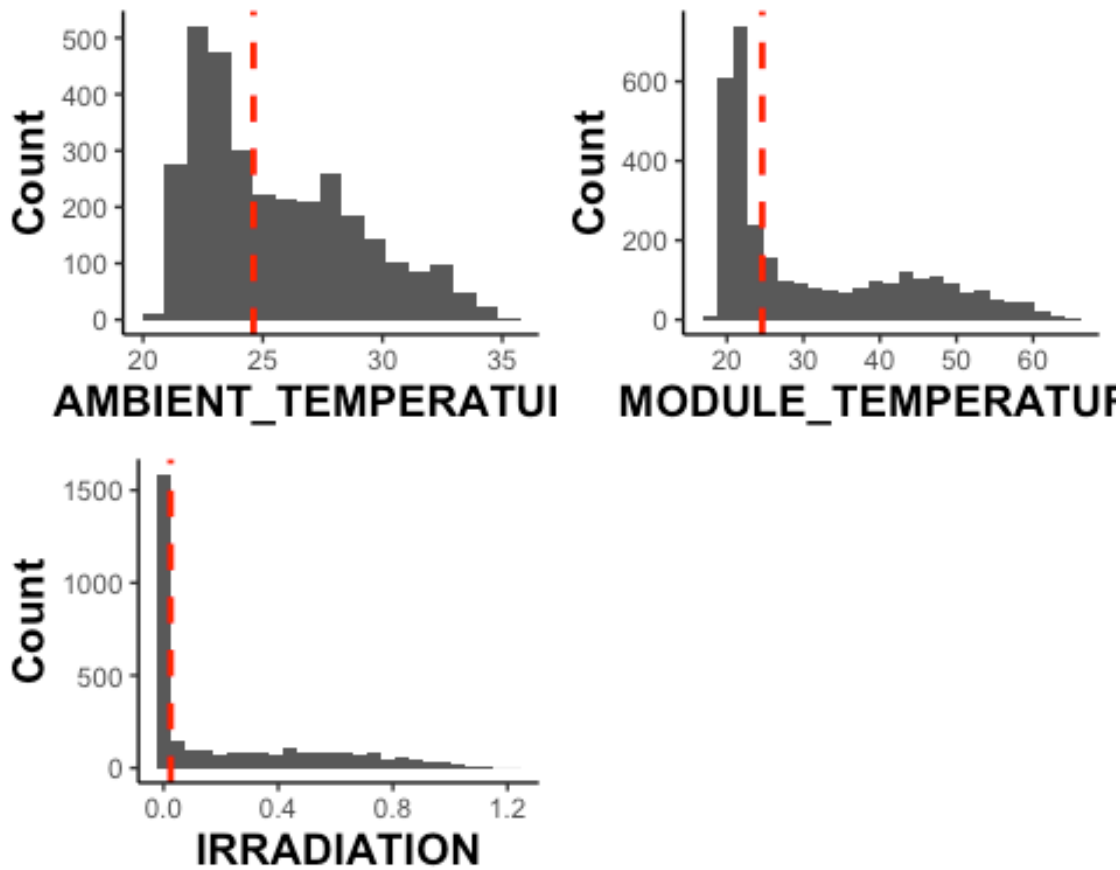
7/3/2021

Loading - Power Plant 1

```
df1 = read.csv('Data/Plant_1_Generation_Data.csv', stringsAsFactors = T)
df1$DATE_TIME = strptime(as.character(df1$DATE_TIME), format = "%d-%m-%Y %H:%
M")#Converting to timestamp

wdf1 = read.csv('Data/Plant_1_Weather_Sensor_Data.csv', stringsAsFactors = T)
wdf1$DATE_TIME = strptime(as.character(wdf1$DATE_TIME), format = "%Y-%m-%d %H
:%M:%S")#Conerting to timestamp

#Joining Datasets
wdf1 = wdf1[,c('DATE_TIME', "AMBIENT_TEMPERATURE", "MODULE_TEMPERATURE", "IRR
ADIATION" )]
df = merge(df1, wdf1)
```

Distribution of Weather Variables

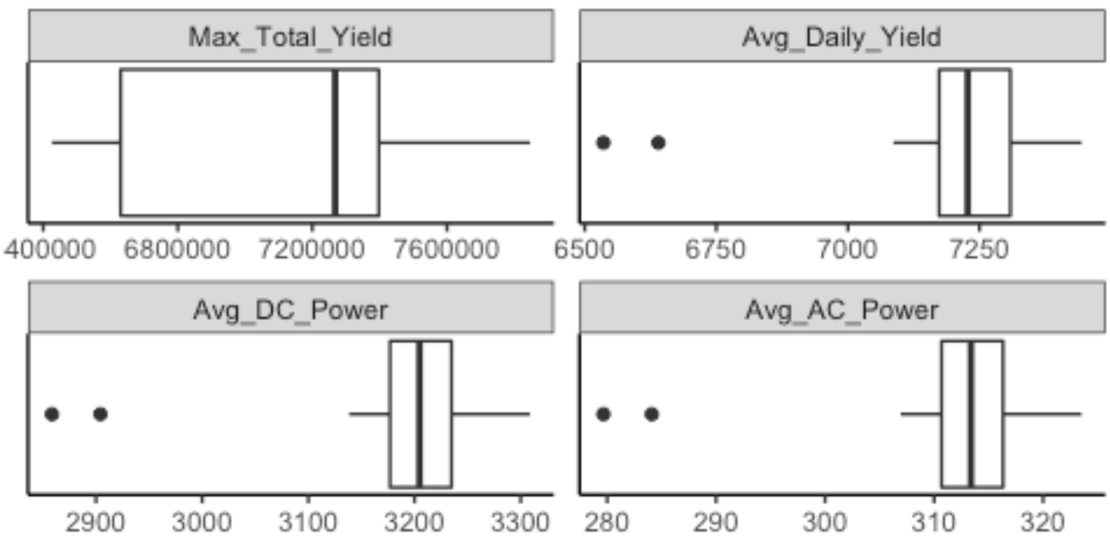
```
#plotdists(df1[,c(-1,-2,-3)],'Plots/p1_generation_', brtype = 'Scott')
plotdists(wdf1[,-1],'Plots/p1_weather_',)
```



Inverter EDA

```
#Max Yields
maxyields = data.frame('SOURCE_KEY' = as.character(), 'Max_Total_Yield' = as.
numeric())
for (key in levels(df$SOURCE_KEY)){
  yields = df$TOTAL_YIELD[df$SOURCE_KEY==key]
  dfa = data.frame('SOURCE_KEY' = key, 'Max_Total_Yield' = max(yields))
  maxyields = rbind(dfa, maxyields)
}

#Average Daily Yield
dailyyield = data.frame('SOURCE_KEY' = as.character(), 'Avg_Daily_Yield' = as
.numeric(),'Avg_DC_Power' = as.numeric(), 'Avg_AC_Power' = as.numeric())

for (key in levels(df$SOURCE_KEY)){
  keydf = df[df$SOURCE_KEY==key,]
  keydf$dates = factor(as.Date(keydf$DATE_TIME))
  maxvec = as.numeric()
  dcvec = as.numeric()
  acvec = as.numeric()
  for (d in levels(keydf$dates)){
    datedf = keydf[keydf$dates == d,]
    maxvec = c(max(datedf$DAILY_YIELD), maxvec)
```

```r
    dcvec = c(mean(datedf$DC_POWER), dcvec)
    acvec = c(mean(datedf$AC_POWER), acvec)
  }
  dfa = data.frame('SOURCE_KEY' = key, 'Avg_Daily_Yield' = mean(maxvec), 'Avg_DC_Power' = mean(dcvec), 'Avg_AC_Power' = mean(acvec) )
  dailyyield = rbind(dfa, dailyyield)
}

yielddf = merge(maxyields, dailyyield)

yielddf.m = melt(yielddf)

## Using SOURCE_KEY as id variables

ggplot(data = yielddf.m, aes(x = value))+geom_boxplot()+facet_wrap(~variable, scales = 'free')+plot_opts+theme(axis.text.y = element_blank(), axis.ticks.y = element_blank(), axis.title.x = element_blank())
```




```r
ggsave('Plots/invertereda.pdf', dpi = 600)

## Saving 5 x 2.5 in image
```

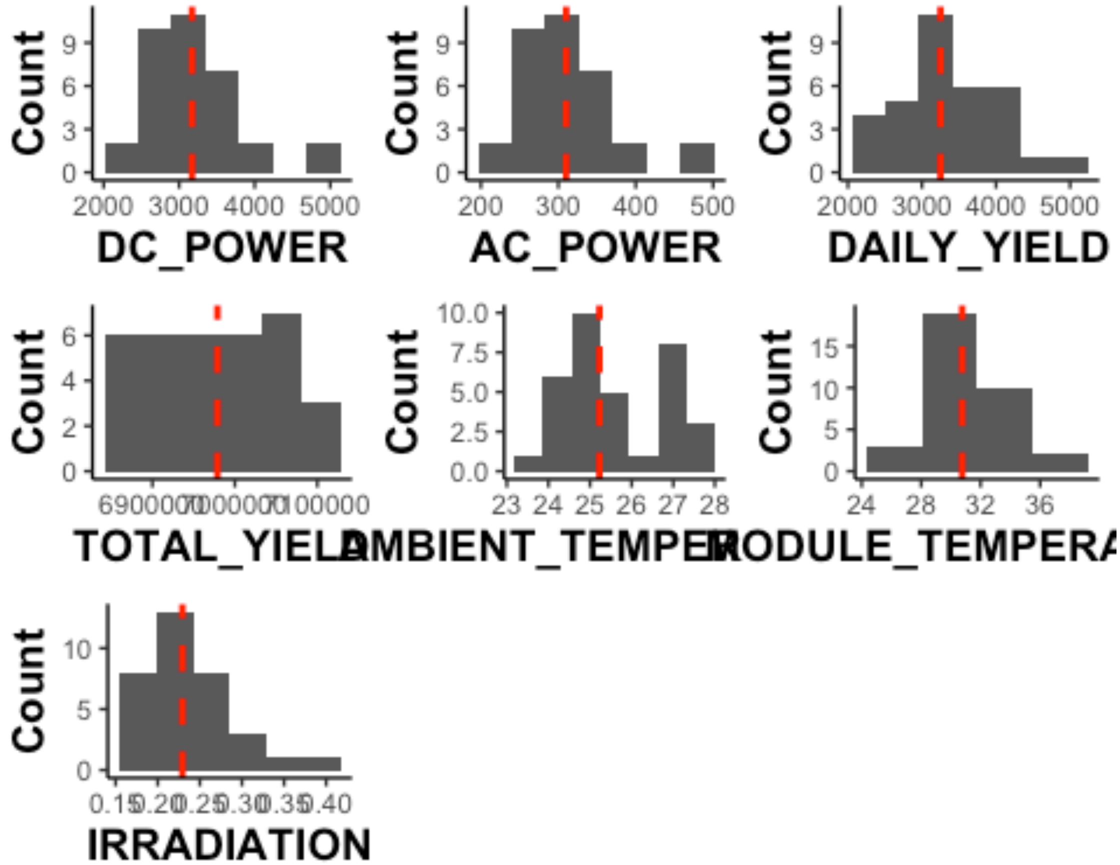
It looks like there are two outliers for each, the worse performing source key is 1BY6WEcLGh8j5v7 while the other is bvBOhCH3iADSZry .

Time EDA

Similarly, daily values of all variables will be analyzed.

```r
df$DATES = factor(as.Date(df$DATE_TIME))
dailydf = data.frame()

for (d in levels(df$DATES)){
  datedf = df[df$DATES==d,]
```

```
  datedf = datedf[,c(-1,-2,-3,-length(datedf))]
  dfnew = t(as.data.frame(colMeans(datedf)))
  row.names(dfnew) = d
  dailydf = rbind(dfnew, dailydf)
}

#dailydf$DATE = row.names(dailydf)

plotdists(dailydf, path = 'Plots/dailyeda')
```




```
write.csv(x = df, file = 'Data/Plant_1_Clean.csv')
```

ARIMA Model

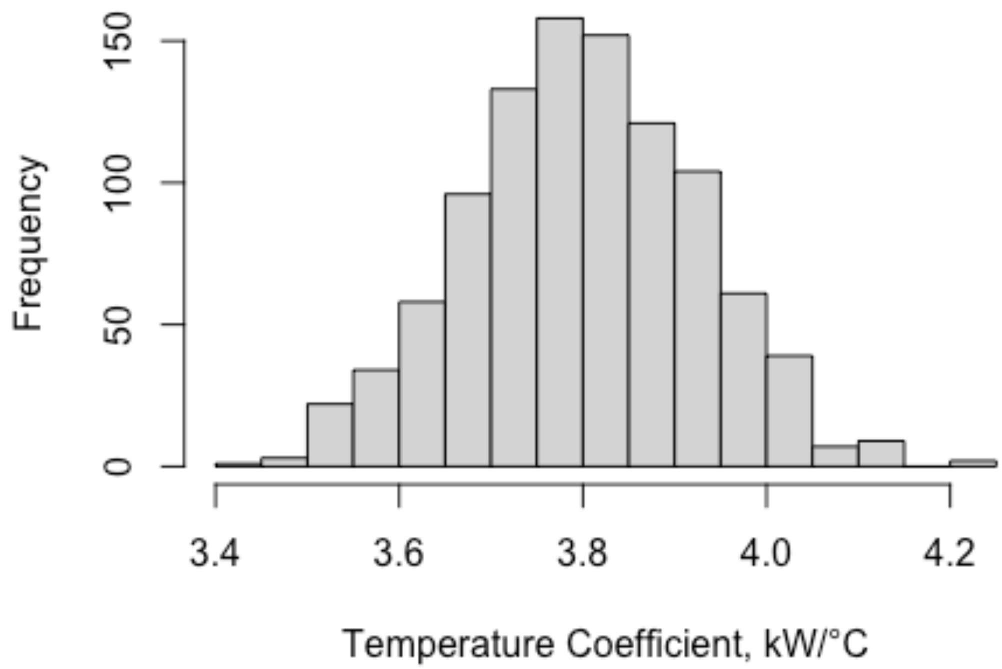
Below will aggregate all ac power measurements for the plant.

```
dt = factor(df$DATE_TIME)
dfnew = data.frame('DATE_TIME' = rep(NA,length(levels(dt))), 'Avg_AC' = rep(N
A,length(levels(dt))))

i = 1
for (d in levels(dt)){
  dtdf = df[df$DATE_TIME == d, ]
  dfnew$DATE_TIME[i] = d
```

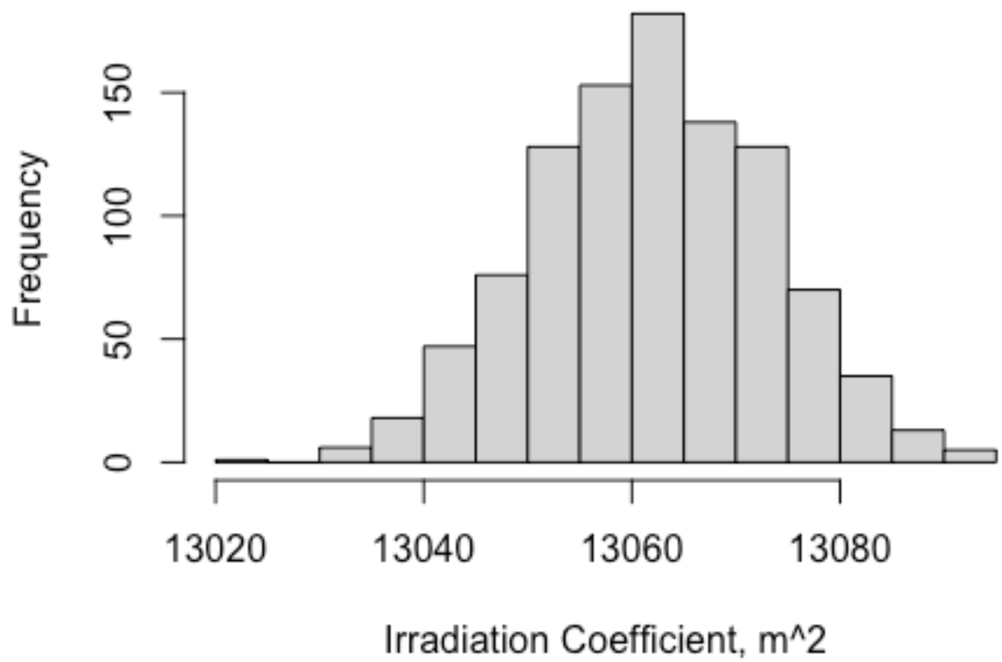
```r
  dfnew$Avg_AC[i] = sum(dtdf$DAILY_YIELD)
  i = i +1
}
dfnew$Sum_AC = dfnew$Avg_AC
dfnew = dfnew[,-2]

write.csv(x = dfnew, file = 'Data/aggregated_yield.csv')
```

Linear Regression

```r
regdf = df[,c('DC_POWER', 'IRRADIATION', 'MODULE_TEMPERATURE')]
lmodel = lm(DC_POWER~0 +MODULE_TEMPERATURE + IRRADIATION, data=regdf)
summary(lmodel)

##
## Call:
## lm(formula = DC_POWER ~ 0 + MODULE_TEMPERATURE + IRRADIATION,
##     data = regdf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16183.1    -85.7    -77.8     96.7   7349.6
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## MODULE_TEMPERATURE 3.796e+00  1.249e-01   30.39   <2e-16 ***
## IRRADIATION        1.306e+04  1.101e+01 1186.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 585.5 on 68772 degrees of freedom
## Multiple R-squared:  0.9869, Adjusted R-squared:  0.9869
## F-statistic: 2.594e+06 on 2 and 68772 DF,  p-value: < 2.2e-16

rmse = sqrt(mean(lmodel$residuals**2))
rmse

## [1] 585.4585

n.sims = 1000
sim.1 <- sim (lmodel, n.sims)

temp.coef <- sim.1@coef[,1]
irr.coef =sim.1@coef[,2]

hist(temp.coef, xlab = 'Temperature Coefficient, kW/°C', main = '')
```




```
hist(irr.coef, xlab = 'Irradiation Coefficient, m^2', main = '')
```



Frequency
0
50
100
150
13020
13040
13060
13080
Irradiation Coefficient, m^2

Appendix B - Non Linear Estimation and Faulty Equipment Classification

Loading and Training Data

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import numpy as np
```

In [2]:
```
df = pd.read_csv('Data/Plant_1_Clean.csv')
```

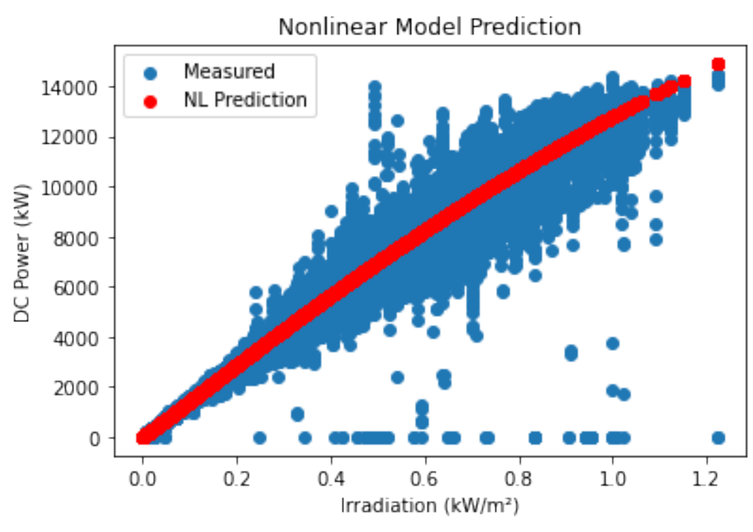
In [3]:
```
# choose training data
train_dates = ["2020-05-16", "2020-05-17","2020-05-18" ,"2020-05-19", "2020-05-20", "2020-05-21"]
df_train = df[df["DATES"].isin(train_dates)]
```

Defining Non-Linear Function

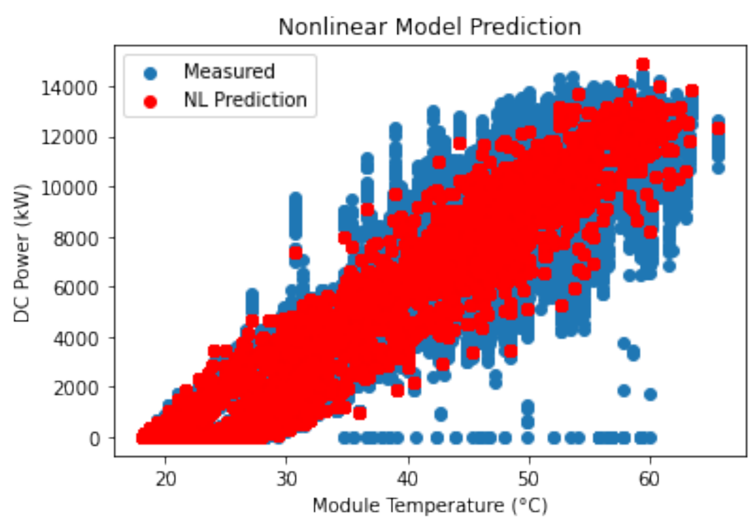
In [4]:
```
def func(X, a, b, c, d):
    '''Nonlinear function to predict DC power output from Irradiation and Temperature.'''
    x,y = X#E(t), T(t)
    x=x*1000
    y=y*1000
    return a*x*(1-b*(y+x/800*(c-20)-25)-d*np.log(x+1e-10))

# fit function
p0 = [1.,0.,-1.e4,-1.e-1] # starting values
popt, pcov = curve_fit(func, (df_train.IRRADIATION, df_train.MODULE_TEMPERATURE), df_train.DC_POWER, p0, maxfev=5000)
sigma_abcd = np.sqrt(np.diagonal(pcov))

# predict & save
df_train["Prediction_NL"] = func((df_train.IRRADIATION, df_train.MODULE_TEMPERATURE), *popt)
df_train["Residual_NL"] = df_train["Prediction_NL"] - df_train["DC_POWER"]

df["Prediction_NL"] = func((df.IRRADIATION, df.MODULE_TEMPERATURE), *popt)
df["Residual_NL"] = df["Prediction_NL"] - df["DC_POWER"]
```

```
<ipython-input-4-f4408d9065be>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_train["Prediction_NL"] = func((df_train.IRRADIATION, df_train.MODULE_TEMPERATURE), *popt)
<ipython-input-4-f4408d9065be>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_train["Residual_NL"] = df_train["Prediction_NL"] - df_train["DC_POWER"]
```

In [5]:
```
plt.scatter(df.IRRADIATION, df.DC_POWER, label="Measured")
plt.scatter(df.IRRADIATION, df.Prediction_NL, color="r", label="NL Prediction")
plt.legend()
plt.xlabel("Irradiation (kW/m²)")
plt.ylabel("DC Power (kW)")
plt.title("Nonlinear Model Prediction")
plt.show()
```



In [6]:
```
plt.scatter(df.MODULE_TEMPERATURE, df.DC_POWER, label="Measured")
plt.scatter(df.MODULE_TEMPERATURE, df.Prediction_NL, color="r", label="NL Prediction")
plt.legend()
plt.xlabel("Module Temperature (°C)")
plt.ylabel("DC Power (kW)")
plt.title("Nonlinear Model Prediction")
plt.show()
```



Parameter Estimates and Signal to Noise

In [7]:
```
popt
```

Out[7]:
```
array([ 1.11066843e+01, -8.30279162e-10, -2.90893800e+08, -6.56017151e-02])
```

In [8]:
```
(popt / sigma_abcd).astype(float)
```

Out[8]:
```
array([ 2.04447912e+01, -2.93911876e-03, -2.95693986e-03, -5.24375084e+00])
```

Calculating Residuals and Coming Up w/ Threshold for Classifying Faults

In [9]:
```
resids = df.Prediction_NL - df.DC_POWER
```

In [10]:
```
iqr = resids.quantile(0.75) - resids.quantile(0.25)

upperfence = resids.quantile(0.75) + 3*iqr
upperfence

faultdf = df[resids > 5000 ]
nonfaultdf = df[(resids <= 5000) ]
```

In [11]:
```
faultdf.to_csv('Data/faulty.csv')
```

In [12]:
```
faultdf.shape
```

Out[12]:
```
(82, 14)
```

In [13]:
```
faultdf.SOURCE_KEY.value_counts()
```

Out[13]:
```
1BY6WEcLGh8j5v7    23
bvBOhCH3iADSZry    23
z9Y9gH1T5YWrNuG     8
wCURE6d3bPkepu2     7
McdE0feGgRqW7Ca     7
sjndEbLyjtCKgGv     5
zBIq5rxdHJRwDNY     3
uHbuxQJl8lW7ozc     1
ih0vzX44oOqAx2f     1
VHMLBKoKgIrUVDU     1
rGa61gmuvPhdLxV     1
pkci93gMrogZuBj     1
7JYdWkrLSPkdwr4     1
Name: SOURCE_KEY, dtype: int64
```

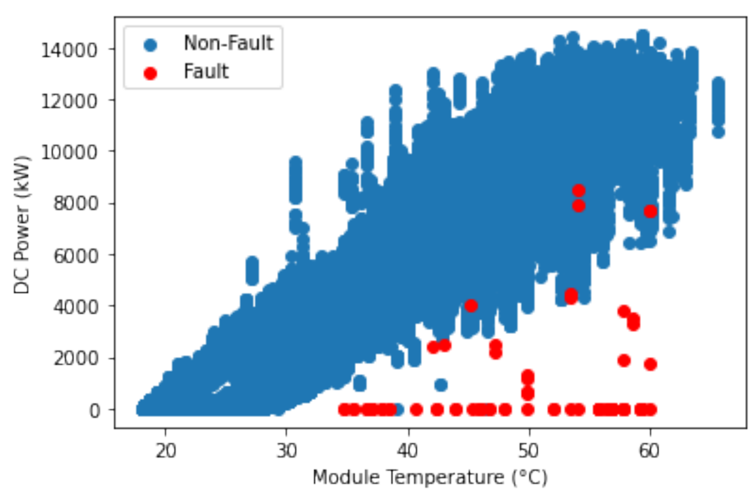
In [14]:
```
faultdf.describe()
```

Out[14]:

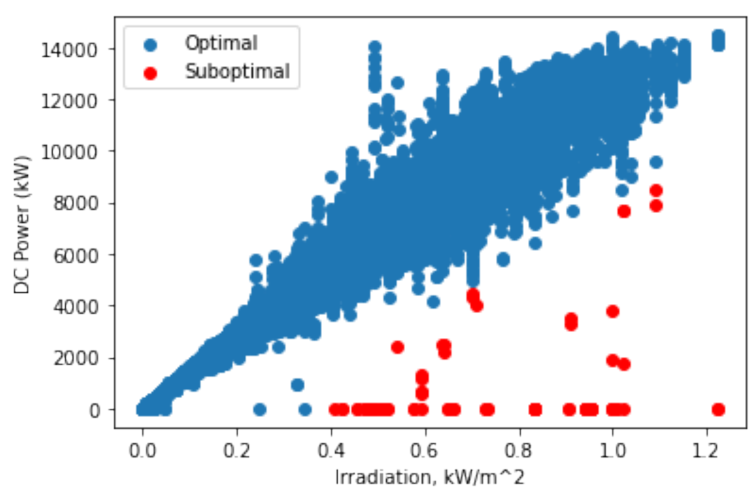
	Unnamed: 0	PLANT_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	Prediction_NL	Residual_NL
count	82.000000	82.0	82.000000	82.000000	82.000000	8.200000e+01	82.000000	82.000000	82.000000	82.000000	82.000000
mean	49298.768293	4135001.0	884.252831	86.397125	3226.540723	6.749501e+06	28.427553	50.547176	0.772513	10190.528142	9306.275311
std	12967.318944	0.0	1967.500697	192.150720	1011.282435	3.606795e+05	1.832779	7.796285	0.215521	2478.742175	2758.308371
min	11976.000000	4135001.0	0.000000	0.000000	1280.857143	6.258289e+06	24.932560	34.780682	0.406296	5737.528259	5035.275684
25%	46725.500000	4135001.0	0.000000	0.000000	2389.968750	6.466065e+06	27.048904	45.547812	0.591769	8150.855964	6866.400713
50%	46795.500000	4135001.0	0.000000	0.000000	3551.000000	6.520020e+06	28.884908	53.509308	0.783414	10440.270900	8983.796284
75%	61435.250000	4135001.0	429.281250	42.121875	4021.000000	7.181310e+06	29.951886	56.895085	0.954715	12320.832060	12202.754099
max	65737.000000	4135001.0	8479.428571	828.700000	6464.000000	7.645856e+06	33.761304	59.987771	1.221652	14891.731663	14891.731663

Plotting Faulty Classifications

In [15]:
```
plt.scatter(nonfaultdf.MODULE_TEMPERATURE, nonfaultdf.DC_POWER,  label="Non-Fault")
plt.scatter(faultdf.MODULE_TEMPERATURE, faultdf.DC_POWER, label="Fault",color="r")
plt.legend()
plt.xlabel("Module Temperature (°C)")
plt.ylabel("DC Power (kW)")
plt.savefig('Plots/suboptimal.png', dpi = 600)
plt.show()
```



In [16]:
```
plt.scatter(nonfaultdf.IRRADIATION, nonfaultdf.DC_POWER,  label="Optimal")
plt.scatter(faultdf.IRRADIATION, faultdf.DC_POWER, label="Suboptimal",color="r")
plt.legend()
plt.xlabel("Irradiation, kW/m^2")
plt.ylabel("DC Power (kW)")
plt.savefig('Plots/suboptimal.png', dpi = 600)
plt.show()
```



In [17]:
```
df.describe()
```

Out[17]:

	Unnamed: 0	PLANT_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	Prediction_NL	Residual_N
count	68774.000000	68774.0	68774.000000	68774.000000	68774.000000	6.877400e+04	68774.000000	68774.000000	68774.000000	68774.000000	68774.0000
mean	34387.500000	4135001.0	3147.177450	307.778375	3295.834644	6.978728e+06	25.558521	31.244997	0.232305	3163.865830	16.6883
std	19853.488044	0.0	4036.441826	394.394865	3145.220597	4.162707e+05	3.361300	12.308283	0.301948	4032.030214	548.5504
min	1.000000	4135001.0	0.000000	0.000000	0.000000	6.183645e+06	20.398505	18.140415	0.000000	-0.000000	-7111.3494
25%	17194.250000	4135001.0	0.000000	0.000000	0.000000	6.512007e+06	22.724491	21.123944	0.000000	-0.000000	-31.3461
50%	34387.500000	4135001.0	428.571429	41.450000	2658.473214	7.146685e+06	24.670178	24.818984	0.031620	427.415207	-0.0000
75%	51580.750000	4135001.0	6365.468750	623.561161	6274.000000	7.268751e+06	27.960429	41.693659	0.454880	6386.950755	3.9972
max	68774.000000	4135001.0	14471.125000	1410.950000	9163.000000	7.846821e+06	35.252486	65.545714	1.221652	14891.731663	14891.7316

Running Non Linear Fit on Each Day

In [18]:
```
p0 = [1.,0.,-1.e4,-1.e-1] # starting values
paramdf = pd.DataFrame()

for d in df.DATES.unique():
    ddf = df[df.DATES == d]
    popt, pcov = curve_fit(func, (ddf.IRRADIATION, ddf.MODULE_TEMPERATURE), ddf.DC_POWER, p0, maxfev=100000)

    dfa = pd.DataFrame({'a':[popt[0]], 'b': [popt[1]], 'c':[popt[2]], 'd':[popt[3]]})
    paramdf = paramdf.append(dfa)
```

In [19]:
```
paramdf.to_csv('Data/nonlinearparamsestimates.csv', index = False)
```

In [ ]:

Appendix C - Prophet Model Fitting

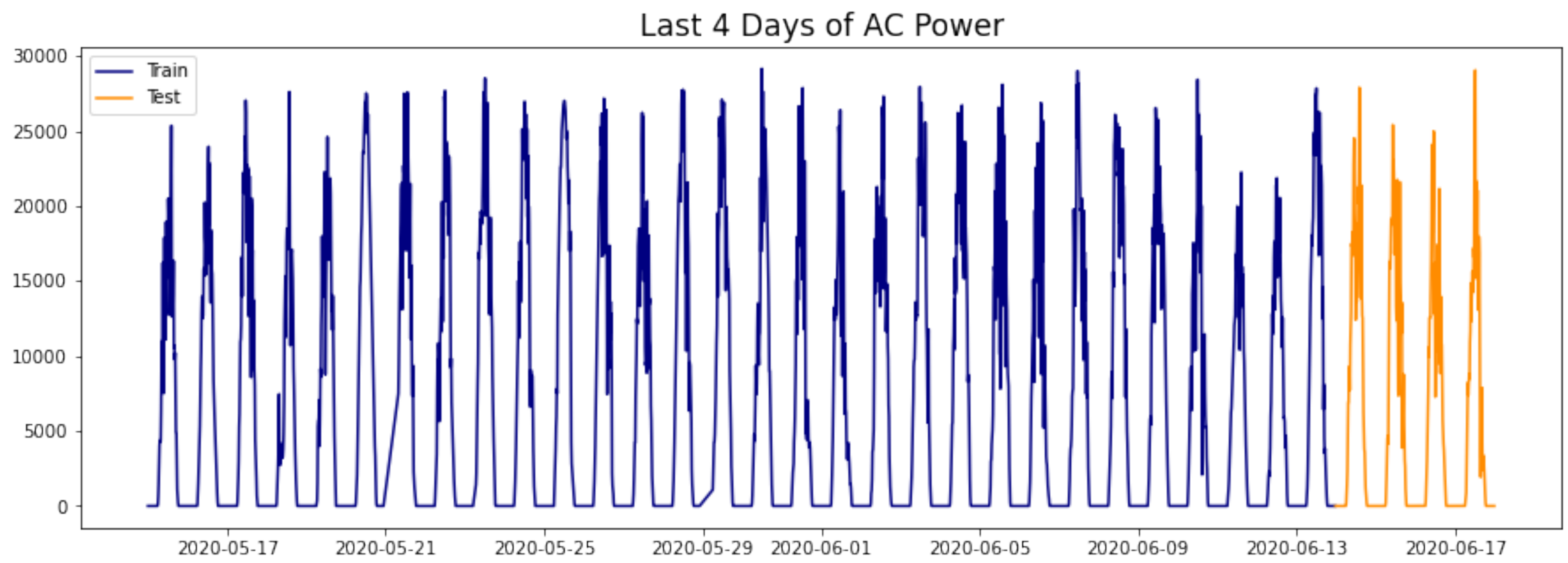
Loading

In [8]:
```
from statsmodels.tsa.stattools import adfuller
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from fbprophet import Prophet
```

In [9]:
```
pred_gen = pd.read_csv('/Users/nickwawee/Desktop/BGSU/MSA_6450/Project/Data/aggregated_ac.csv')
pred_gen.drop('Unnamed: 0', axis =1 , inplace = True )
```

In [10]:
```
pred_gen.DATE_TIME = pd.to_datetime(pred_gen.DATE_TIME)
```

In [11]:
```
train=pred_gen[:(pred_gen.shape[0] - 384)]
test=pred_gen[-384:]
plt.figure(figsize=(15,5))
plt.plot(train.DATE_TIME, train.Sum_AC, label='Train',color='navy')
plt.plot(test.DATE_TIME, test.Sum_AC, label='Test',color='darkorange')
plt.title('Last 4 Days of AC Power',fontsize=17)
plt.legend()
plt.show()
```



In [15]:
```
trainnew = pd.DataFrame({'ds': train.DATE_TIME, 'y': train.Sum_AC})
m = Prophet()
m.fit(trainnew)
```

```
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
```

Out[15]: `<fbprophet.forecaster.Prophet at 0x7f8b49daeeb0>`

In [16]:
```
fut = pd.DataFrame(columns = ['ds'])
fut['ds'] = pd.concat([test.DATE_TIME, train.DATE_TIME], axis = 0)
fut
```

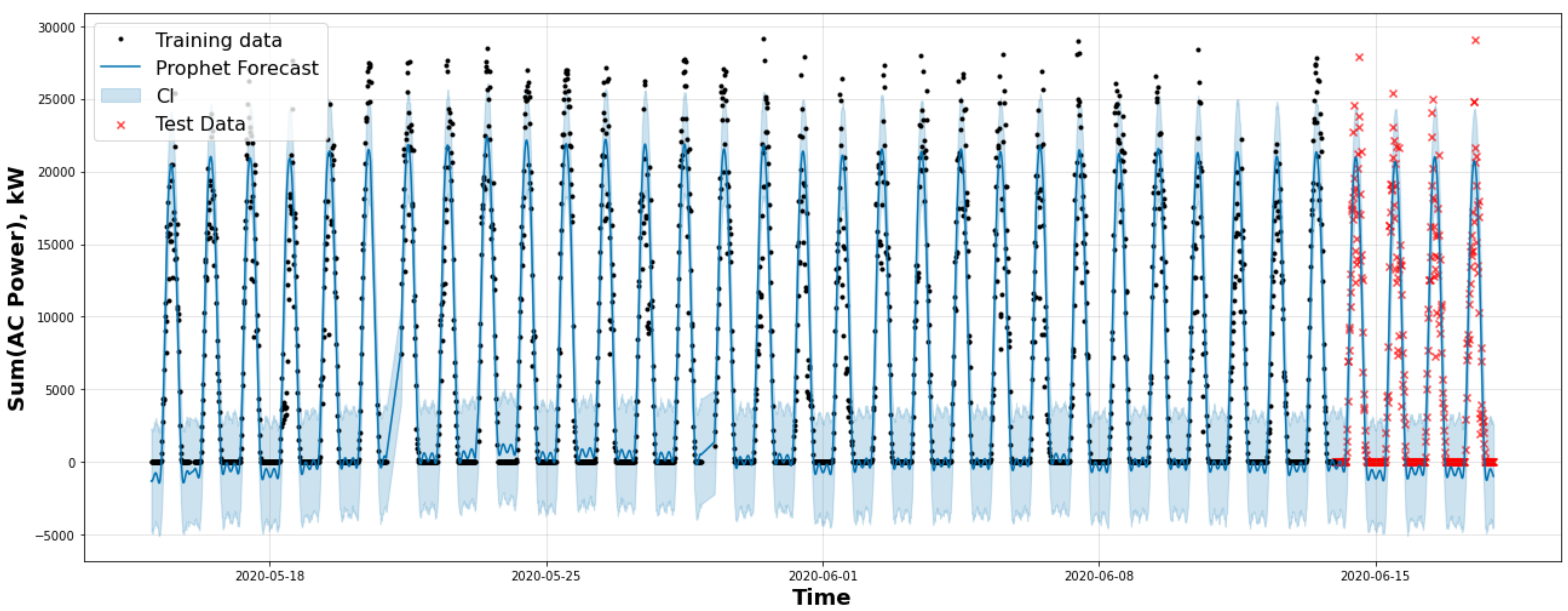
Out[16]:

	ds
2773	2020-06-13 23:30:00
2774	2020-06-13 23:45:00
2775	2020-06-14 00:00:00
2776	2020-06-14 00:15:00
2777	2020-06-14 00:30:00
...	...
2768	2020-06-13 22:15:00
2769	2020-06-13 22:30:00
2770	2020-06-13 22:45:00
2771	2020-06-13 23:00:00
2772	2020-06-13 23:15:00

3157 rows × 1 columns

In [17]:
```
forecast = m.predict(fut)
```

In [18]:
```
m.plot(forecast,figsize=(18,7))
plt.title('ok')
plt.scatter(x = test.DATE_TIME, y = test.Sum_AC, marker = 'x', color = 'red', alpha = 0.7)
plt.legend(labels=['Training data','Prophet Forecast', 'CI','Test Data'], prop={'size': 16})
plt.title('', fontweight = 'bold', fontsize = 20)
plt.xlabel('Time', fontweight = 'bold', fontsize = 18)
plt.ylabel('Sum(AC Power), kW',fontweight = 'bold', fontsize = 18 )
plt.savefig('/Users/nickwawee/Desktop/BGSU/MSA_6450/Project/Plots/Prophet.png', dpi = 600)
plt.show()
```



In [19]:
```
rmse = np.sqrt(np.mean((forecast.yhat - test.Sum_AC)**2))
```

In [20]:
```
rmse
```

Out[20]: 2890.653827175155

In [ ]: