**PREDICTING SOLAR AND WIND GENERATION**

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING**

BY

**TANISHA JAIN**

**EN16CS301275**

Under the Guidance of

**MR. SACHIN SOLANKI(Internal guide)**

**&**

**MR. AJAY PORWAL(External guide)**



**Department of Computer Science & Engineering**

**Faculty of Engineering**

**MEDI-CAPS UNIVERSITY, INDORE- 453331**

**May 2020**

**Report Approval**

The project work **“Predicting solar and wind generation”** is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation

Affiliation

External Examiner

Name

Designation

Affiliation

**Declaration**

I/We hereby declare that the project entitled **“Predicting solar and wind generation”** submittedin partial fulfillment for the award of the degree of Bachelor of Technology/Master of Computer Applications in Computer Science completed under the supervision of Mr.Sachin Solanki**,** Professor, Department of Computer Science Engineering, Medi-Caps University Indore and Mr. Ajay Porwal, project guide, Free wings Power and Infra Limited, is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

**Signature and name of the student(s) with date**

**Certificate**

I/We, **Sachin Solanki and Ajay Porwal** certify that the project entitled **“Predicting solar and wind generation”** submittedin partial fulfillment for the award of the degree of Bachelor of Technology/Master of Computer Applications by **Tanisha Jain** istherecordcarried out by him/them under my/our guidance and that the work has not formed the basis of award of any other degree elsewhere.

**Mr.Sachin Solanki Mr. Ajay Porwal** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_** Computer Science Department

Medi-Caps University, Indore Free wings Power and Infra Limited

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr.Suresh Jain

Head of the Department

Computer Science & Engineering

Medi-Caps University, Indore

**Acknowledgements**

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Sunil K Somani,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Suresh Jain** for his continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my **External Guide, Mr.Ajay Porwal**, Project Lead, FREEWINGS POWER & INFRA LIMITED as well as to my Internal Guide, Mr.Sachin Solanki**,** Professor, Department of Computer Science Engineering, MU without whose continuous help and support, this project would never have reached to the completion.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

**TANISHA JAIN**

B.Tech. IV Year

Department of Computer Science & Engineering

Faculty of Engineering

**Medi-Caps University, Indore**

**ABSTRACT**

This report is about the project "Predicting solar and wind generation". One challenge with production, selling and usage of renewable energy is that it is always variable therefore it is a necessity to predict the future generation capability of a particular plant at a particular location on a particular day, date or time. This project builds a Desktop application which takes location and date as input and uses weather forecast data using a weather API for that location. These weather parameters are fed into the trained model which sends the predicted solar or wind generation (as set by the user) values.

This report contains a detailed explanation of two data science concepts Data Visualization and Exploratory Data Analysis. It also shows how these concepts are used in my project with all the steps, from filtering the most useful parameters from the given dataset to filling the null values.

It explains about the dataset containing weather data and actual energy production values. After understanding the given dataset we apply different steps to filter data and know about the most useful parameters on which the energy production depends or is affected by, which we then use to train our model and further to predict the future generation.

I have used Linear regression algorithm to train our model. To validate the efficiency, K-fold Cross Validation method is applied. For solar and wind generation this model provides an efficiency of 84 and 91 percent respectively which is pretty great.

The uses of the project are as follows:

1. Know about the efficiency and profitability of the plant.

2. To make changes in the design and use of the types of panels  
3. To predict the generation capability of a location which could be a possible new, project location.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | **Table of Contents** |  |  | | |
|  | |  | | | | |  |  |
|  | | Report Approval | | | | |  | i |
|  | | Declaration | | | | |  | ii |
|  | | Certificate | | | | |  | iii |
|  | | Acknowledgement | | | | |  | iv |
|  | | Abstract | | | | |  | v |
|  | | Table of Content | | | | |  | vi |
|  | | List of figures | | | | |  | vii |
|  | | List of tables | | | | |  | viii |
| Chapter 1 | | Introduction | | | | |  | 1 |
|  |  | |  |  |  | | | |
| Chapter 2 | | Project Handled | | | | |  | 4 |
|  | | 2.1 Tools and technologies used | | | | |  | 4 |
|  | | 2.1.1 Hardware Tools | | | | |  | 5 |
|  | | 2.1.2 Software Tools | | | | |  | 5 |
|  | | 2.2. Training Schedule | | | | |  | 6 |
| Chapter 3 | | DATA VISUALIZATION AND EXPLORATORY DATA ANALYSIS | | | | |  | 7 |
|  |  | |  |  |  |  | | |
|  |  | | 3.1 | DATA VISUALIZATION |  | 7 | | |
|  |  | |  | 3.1.1 Matplotlib |  | 8 | | |
|  |  | |  | 3.1.1(i) Scatter Plot |  | 9 | | |
|  |  | |  | 3.1.1(ii) Line Chart |  | 10 | | |
|  |  | |  | 3.1.1(iii) Histogram |  | 11 | | |
|  |  | |  |  |  |  | | |
|  |  | |  | 3.1.2 Seaborn |  | 13 | | |
|  |  | |  | 3.1.2(i) Line Chart |  | 14 | | |
|  |  | |  | 3.1.2(ii) Histogram |  | 15 | | |
|  |  | |  | 3.1.2(iii) Bar Chart |  | 16 | | |
|  |  | |  | 3.1.2(iv) Box Plot |  | 18 | | |
|  |  | |  | 3.1.2(v) Heat Map |  | 19 | | |
|  |  | |  | 3.1.2(vi) Faceting |  | 21 | | |
|  |  | |  | 3.1.2(vii)Pairplot |  | 24 | | |
|  |  | |  |  |  |  | | |
|  |  | |  |  |  |  | | |
|  |  | | 3.2 | EXPLORATORY DATA ANALYSIS |  | 26 | | |
|  |  | |  |  |  |  | | |
|  |  | |  |  |  |  | | |
| Chapter 4 |  | |  | MY WORK |  | 40 | | |
|  |  | |  |  |  |  | | |
|  |  | |  | 4.1 The Data |  | 41 | | |
|  |  | |  | 4.2 Weather Data |  | 42 | | |
|  |  | |  | 4.3 Predicting the wind and solar generation using linear regression |  | 44 | | |
|  |  | |  | 4.3.1 Wind Generation |  | 47 | | |
|  |  | |  | 4.3.2 Solar Generation |  | 52 | | |
|  |  | |  | 4.4 Front -end |  |  | | |
|  |  | |  | 4.4.1 Electron js |  |  | | |
|  |  | |  | 4.4.2 Python shell |  |  | | |
|  |  | |  | 4.4.3 Working of the application |  |  | | |
|  |  | |  |  |  |  | | |
|  |  | |  |  |  |  | | |
| Chapter 5 |  | |  | Learning after Training |  | 54 | | |
|  |  | |  |  |  |  | | |
| Chapter 6 |  | |  | Discussion |  | 55 | | |
|  |  | |  |  |  |  | | |
| Chapter 7 |  | |  | Conclusion |  | 56 | | |
|  |  | |  |  |  |  | | |

ii

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure** | **Figure Name** | **Page** |
| **No.** |  | **No.** |
| 3.1 | Iris dataset Scatter plot | 10 |
| 3.2 | Iris dataset Scatter plot (class division) | 11 |
| 3.3 | Iris dataset Line chart | 12 |
| 3.4 | Histogram | 12 |
| 3.5 | Bar chart | 13 |
| 3.6 | Seaborn Scatter plot | 14 |
| 3.7 | Seaborn Scatter plot (class division) | 15 |
| 3.8 | Seaborn Linechart | 16 |
| 3.9 | Histogram | 18 |
| 3.30 | Histogram gaussian kernel density | 19 |
| 3.11 | Bar chart | 21 |
| 3.12 | Box plot | 22 |
| 3.13 | Heatmap | 23 |
| 3.14 | Faceting | 24 |
| 3.15 | Pairplot | 26 |
| 3.16 | Iris Dataset pairplot | 27 |
| 3.17 | Histogram (car) | 28 |
| 3.18 | Heatmap | 29 |
| 3.19 | Scatterplot | 29 |
| 4.1 | Actual wind generation in MW | 30 |
| 4.2 | Actual solar generation in MW | 30 |
| 4.3 | Wind velocity 10 m above displacement height | 32 |
| 4.4 | Ground horizontal radiations | 34 |
| 4.5 | Wind generation as function of different weather qualities | 36 |
| 4.6 | Solar generation as function of different weather qualities | 37 |
| 4.7 | K fold cross validation | 40 |

**Chapter 1**

**1.1 Introduction**

One challenge with integrating renewable into the grid is that their power generation is intermittent and uncontrollable. Thus, predicting future renewable generation is important. In this project, “Predicting yearly wind and solar generation” I used linear regression and some other algorithms to predict renewable energy production from the given weather data and then compared the results. The aim to predict production on the basis of whether are as follows :

1. Know about the efficiency and profitability of the plant.

2. To make changes in the design and use of the types of panels

3. To predict the generation capability of a location which could be a possible new, project location.

These are the main goals to be accomplished by this project but there could be more uses to analyze the output.

Before this project I was give some Data Science related topics and concepts to understand and implement as well.

**1.2 Problem Statement**

The purpose of the problem statement and the specifications provided by the company were as follows:

“You are to design and implement a tool to predict the renewable energy production by wind and solar energy plants. You will be given a dataset containing energy production and weather information.”

**1.3 Purpose**

The main aim of developing Predicting yearly wind and solar generation is to provide way to predict renewable energy production from the given weather data which could then be used for various purposes. As integrating renewable into the grid is that their power generation is intermittent and uncontrollable. Thus, predicting future renewable generation is important.

The project was developed for the personal use of the company and the

proprietary rights of the project lie with the company itself.

**2**

**2.2 Training Schedule**

This internship followed a rigorous training schedule, weekly targets were assigned to me and were duly monitored by project mentor. To complete this module of the project I was given a time duration of 40 days. Before the project work I had to go through a training phase in which I learned different data science concepts and topics which then helped me in completing this project. Along with this project I will be given another project related to renewable energy. At the end of which we would receive the certificate.

**Chapter 2**

**Project Handled**

**2.1 Tools and Technology Used**

**2.1.1 Hardware Used**

Processor: Intel Core i5

Memory Size: 4 GB RAM

Input: Keyboard/Mouse

**2.1.2 Software used**

**Front End:**

Designing Language: HTML 5

CSS

Framework : Electron js

Semantic Ui

**2.1.3 Back End:**

Language : Python

Library used : numPy, Pandas, Seaborn, SK Learn

**CHAPTER - 3**

**DATA VISUALIZATION AND EXPLORATORY DATA ANALYSIS**

**3.1 DATA VISUALIZATION**

**Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.**

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

Here are a few popular plotting libraries:

* [**Matplotlib:**](https://matplotlib.org/) low level, provides lots of freedom
* [**Pandas Visualization:**](https://pandas.pydata.org/pandas-docs/stable/visualization.html) easy to use interface, built on Matplotlib
* [**Seaborn :**](https://seaborn.pydata.org/) high-level interface, great default styles
* [**ggplot :**](http://ggplot.yhathq.com/) based on R’s ggplot2, uses [Grammar of Graphics](https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448)
* [**Plotly :**](https://plot.ly/python/) can create interactive plots

I learned how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library.

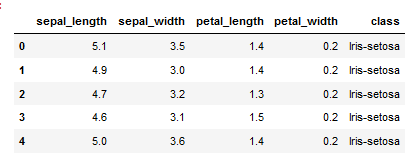
In the project I used matplotlib and seaborn library.

Before visualizing data using any library we import the required dataset which we can load in using pandas read\_csv method.

import pandas as pd

iris = pd.read\_csv('iris.csv', names=['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'class'])

print(iris.head())



wine\_reviews=pd.read\_csv('winemag-data-130k-v2.csv', index\_col=0)

wine\_reviews.head()

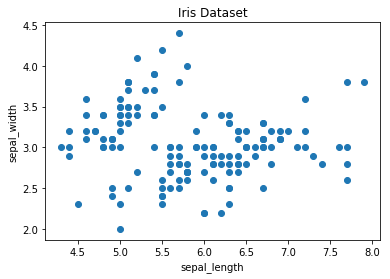
**3.1.1Matplotlib**

# Matplotlib is the most popular python plotting library. It is a low-level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

To install Matplotlib pip and conda can be used.

pip install matplotlib or conda install matplotlib

## **3.1.1(i) Scatter Plot**

To create a scatter plot in Matplotlib we can use the scatter method. We will also create a figure and an axis using plt.subplots so we can give our plot a title and labels.

|  |  |
| --- | --- |
|  | fig, ax = plt.subplots() |
|  |  |
|  | # scatter the sepal\_length against the sepal\_width |
|  | ax.scatter(iris['sepal\_length'], iris['sepal\_width']) |
|  | # set a title and labels |
|  | ax.set\_title('Iris Dataset') |
|  | ax.set\_xlabel('sepal\_length') |
|  | ax.set\_ylabel('sepal\_width') |

We can give the graph more meaning by coloring in each data-point by its class. This can be done by creating a dictionary which maps from class to color and then scattering each point on its own using a for-loop and passing the respective color.

# create color dictionary

colors = {'Iris-setosa':'r', 'Iris-versicolor':'g', 'Iris-virginica':'b'}

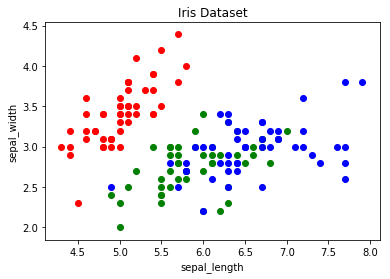
# create a figure and axis

fig, ax = plt.subplots()

# plot each data-point

for i in range(len(iris['sepal\_length'])):

ax.scatter(iris['sepal\_length'][i], iris['sepal\_width'][i],color=colors[iris['class'][i]])

# set a title and labels

ax.set\_title('Iris Dataset')

ax.set\_xlabel('sepal\_length')

ax.set\_ylabel('sepal\_width')

**fig 3.2 Iris dataset Scatter plot (class division)**

## **3.1.1(ii)Line Chart**

In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want and plotting each column on the same axis.

**# get columns to plot**

columns = iris.columns.drop(['class'])

# create x data

x\_data = range(0, iris.shape[0])

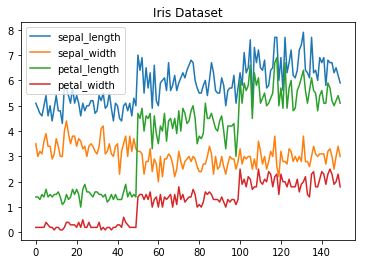
# create figure and axis

fig, ax = plt.subplots()

# plot each column

for column in columns:

ax.plot(x\_data, iris[column], label=column)

# set title and legend

ax.set\_title('Iris Dataset')

ax.legend()

fig 3.3 Iris dataset Line chart

## **3.1.1(iii) Histogram**

In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.

# create figure and axis

fig, ax = plt.subplots()

# plot histogram

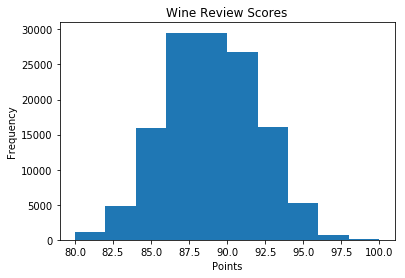
ax.hist(wine\_reviews['points'])

# set title and labels

ax.set\_title('Wine Review Scores')

ax.set\_xlabel('Points')

ax.set\_ylabel('Frequency')



## fig 3.4 Histogram

**3.1.1.(iv)Bar Chart**

A bar chart can be created using the bar method. The bar-chart isn’t automatically calculating the frequency of a category so we are going to use pandas value\_counts function to do this. The bar-chart is useful for categorical data that doesn’t have a lot of different categories (less than 30) because else it can get quite messy.

**# create a figure and axis**

fig, ax = plt.subplots()

# count the occurrence of each class

data = wine\_reviews['points'].value\_counts()

# get x and y data

points = data.index

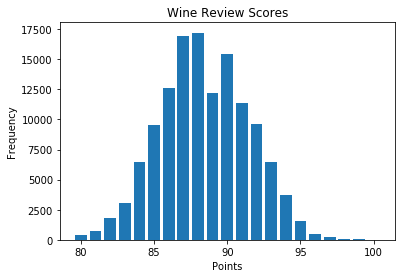
frequency = data.values

# create bar chart

ax.bar(points, frequency)

# set title and labels

ax.set\_title('Wine Review Scores')

ax.set\_xlabel('Points')

ax.set\_ylabel('Frequency')

**fig 3.5Bar chart**

**3.1.2 Seaborn**

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive graphs.

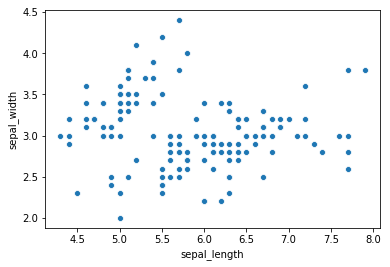
Seaborn has a lot to offer. You can create graphs in one line that would take you multiple tens of lines in Matplotlib. Its standard designs are awesome and it also has a nice interface for working with pandas dataframes.

It can be imported by typing:

import seaborn as sns

We can use the .scatterplot method for creating a scatterplot, and just as in Pandas we need to pass it the column names of the x and y data, but now we also need to pass the data as an additional argument

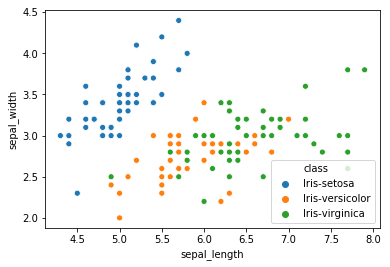
sns.scatterplot(x='sepal\_length', y='sepal\_width', data=iris)

 12

**fig 3.6Seaborn Scatter plot**

We can also highlight the points by class using the hue argument, which is a lot easier than in Matplotlib.

sns.scatterplot(x='sepal\_length', y='sepal\_width', hue='class', data=iris)

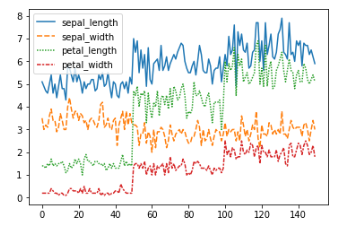


**fig 3.7Seaborn Scatter plot (class division)**

**3.1.2 (i)Line chart**

To create a line-chart the sns.lineplot method can be used. The only required argument is the data.We could also use the sns.kdeplot method which rounds of the edges of the curves and therefore is cleaner if you have a lot of outliers in your dataset.

sns.lineplot(data=iris.drop(['class'], axis=1))

****

**fig 3.8 Seaborn linechart**

**3.1.2.(ii)HISTOGRAM**

To create a histogram in Seaborn we use the sns.distplot method. We need to pass it the column we want to plot and it will calculate the

occurrences itself. We can also pass it the number of bins, and if we want to plot a gaussian kernel density estimate inside the graph.

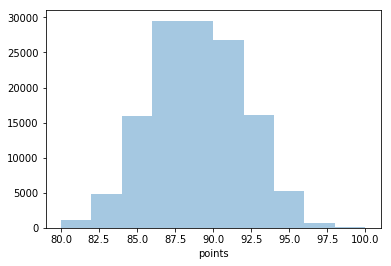
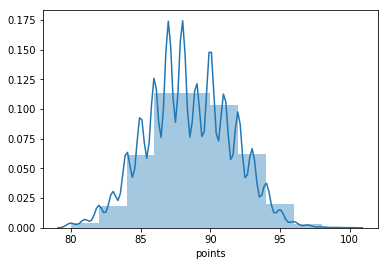
sns.distplot(wine\_reviews['points'], bins=10, kde=False)

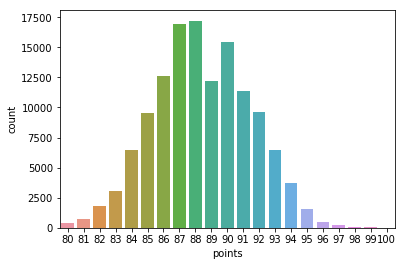
fig 3.9 Histogram



## **3.1.2(iii)Bar chart**

In Seaborn a bar-chart can be created using the sns.countplot method and passing it the data.

sns.countplot(wine\_reviews['points'])



**fig 3.11 Bar chart**

**3.1.2(iv)Other graphs**

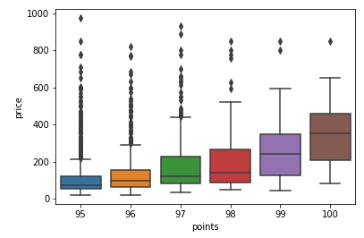
For most of these, Seaborn is the go-to library because of its high-level interface that allows for the creation of beautiful graphs in just a few lines of code.

## **3.1.2(v)Box plots**

A Box Plot is a graphical method of displaying the [five-number summary](https://en.wikipedia.org/wiki/Five-number_summary). We can create box plots using seaborns sns.boxplot method and passing it the data as well as the x and y column name.

df = wine\_reviews[(wine\_reviews['points']>=95) & (wine\_reviews['price']<1000)]

sns.boxplot('points', 'price', data=df)



**fig 3.10 Boxplot**

Box Plots, just like bar-charts are great for data with only a few categories but can get messy really quickly.

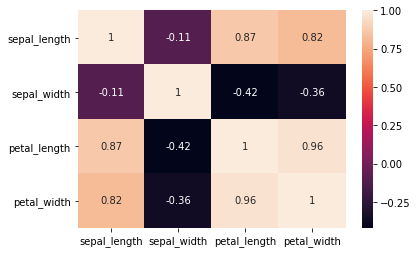
## **3.1.2(vi)Heatmap**

A Heatmap is a graphical representation of data where the individual values contained in a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) are represented as colors. Heatmaps are perfect for exploring the correlation of features in a dataset.

To get the correlation of the features inside a dataset we can call <dataset>.corr(), which is a Pandas dataframe method. This will give us the [correlation matrix](https://www.displayr.com/what-is-a-correlation-matrix/).

import numpy as np

# get correlation matrix **fig 3.13 Heatmap**

corr = iris.corr()

fig, ax = plt.subplots()

# create heatmap

im = ax.imshow(corr.values)

# set labels

ax.set\_xticks(np.arange(len(corr.columns)))

ax.set\_yticks(np.arange(len(corr.columns)))

ax.set\_xticklabels(corr.columns)

ax.set\_yticklabels(corr.columns)

# Rotate the tick labels and set their alignment.

plt.setp(ax.get\_xticklabels(), rotation=45,

ha="right", rotation\_mode="anchor")

To add annotations to the heatmap we need to add two for loops:

Seaborn makes it way easier to create a heatmap and add annotations: sns.heatmap(iris.corr(), annot=True)

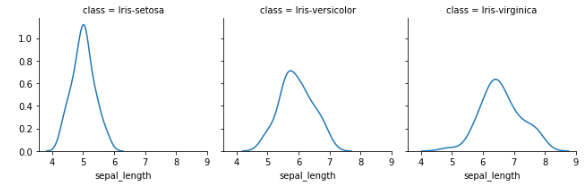
**3.1.2(vii)Faceting**

Faceting is the act of breaking data variables up across multiple subplots and combining those subplots into a single figure.

Faceting is really helpful if you want to quickly explore your dataset.To use one kind of faceting in Seaborn we can use the FacetGrid. First of all, we need to define the FacetGrid and pass it our data as well as a row or column, which will be used to split the data. Then we need to call the map function on our FacetGrid object and define the plot type we want to use, as well as the column we want to graph.

g = sns.FacetGrid(iris, col='class')

g = g.map(sns.kdeplot, 'sepal\_length')



**fig 3.14Faceting**

**3.1.2(viii)Pairplot**

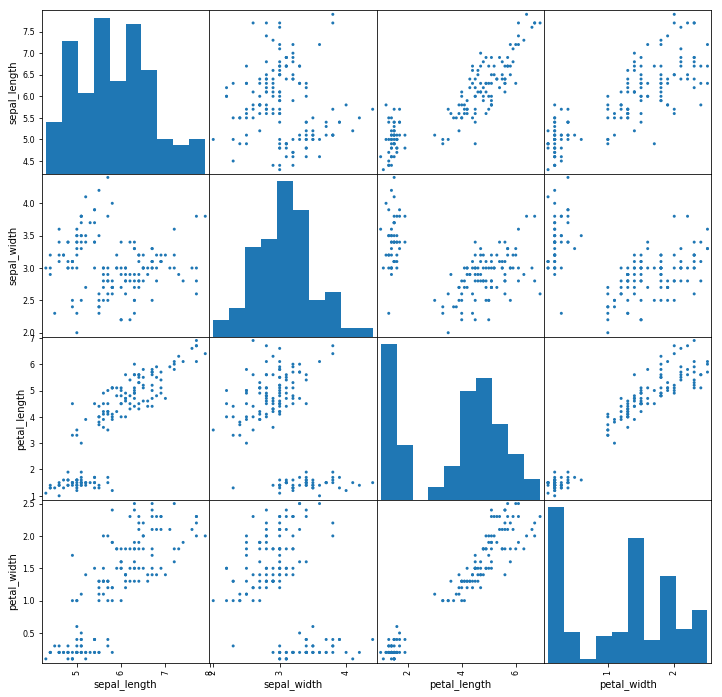
Lastly, Seaborns pairplot and Pandas scatter\_matrix , enable you to plot a grid of pairwise relationships in a dataset.

sns.pairplot(iris)

**3.1.2(ix)From pandas.plotting import scatter\_matrix**

fig, ax = plt.subplots(figsize=(12,12))

scatter\_matrix(iris, alpha=1, ax=ax)



**fig 3.16 Iris Dataset pairplot**

As we can see in the images above these techniques are always plotting two features with each other. The diagonal of the graph is filled with histograms and the other plots are scatter plots.

# 

# **3.2 Exploratory data analysis in Python.**

Exploratory Data Analysis or (EDA) is understanding the data sets by summarizing their main characteristics often plotting them visually. This step is very important especially when we arrive at modeling the data in order to apply Machine learning. Plotting in EDA consists of Histograms, Box plot, Scatter plot and many more. It often takes much time to explore the data. Through the process of EDA, we can ask to define the problem statement or definition on our data set which is very important.

I got a very beautiful data-set of cars from Kaggle. To give a piece of brief information about the data set this data contains more of 10, 000 rows and more than 10 columns which contains features of the car such as Engine Fuel Type, Engine Size, HP, Transmission Type, highway MPG, city MPG and many more.

**1. Importing the required libraries for EDA**

Below are the libraries that are used in order to perform EDA

# Importing required libraries.  
import pandas as pd  
import numpy as np  
import seaborn as sns #visualisation  
import matplotlib.pyplot as plt #visualisation  
%matplotlib inline   
sns.set(color\_codes=True)

**2. Loading the data into the data frame.**

Loading the data into the pandas data frame is certainly one of the most important steps in EDA, as we can see that the value from the data set is comma-separated. So all we have to do is to just read the CSV into a data frame and pandas data frame does the job for us.

To get or load the dataset into the notebook, all I did was one trivial step. In [**Google Colab**](https://colab.research.google.com/notebooks/welcome.ipynb) at the left-hand side of the notebook, you will find a “>” (greater than symbol). When you click that you will find a tab with three options, you just have to select Files. Then you can easily upload your file with the help of the Upload option

df = pd.read\_csv(“data.csv”)  
# To display the top 5 rows  
df.head(5)



#To display the bottom 5 rows  
df.tail(5)

**3. Checking the types of data**

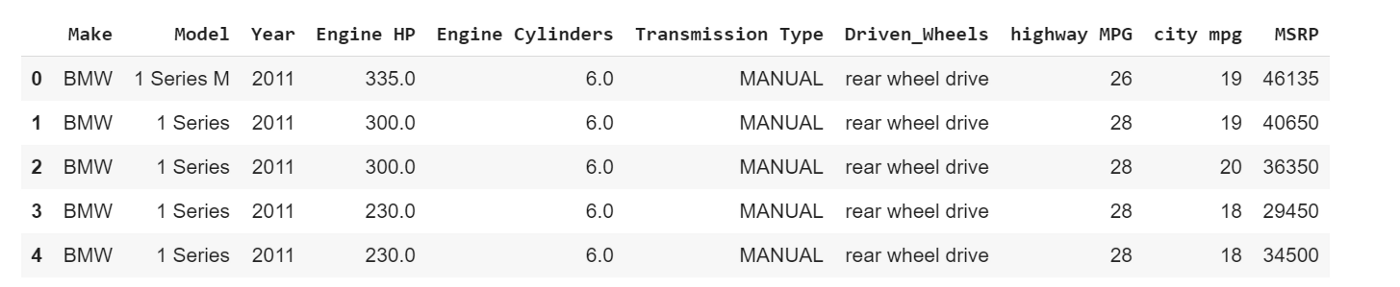
Here we check for the datatypes because sometimes the MSRP or the price of the car would be stored as a string or object, if in that case, we have to convert that string to the integer data only then we can plot the data via a graph. Here, in this case, the data is already in integer format so nothing to worry.

df.dtypes

**4. Dropping irrelevant columns**

This step is certainly needed in every EDA because sometimes there would be many columns that we never use in such cases dropping is the only solution. In this case, the columns such as Engine Fuel Type, Market Category, Vehicle style, Popularity, Number of doors, Vehicle Size doesn't make any sense to me so I just dropped for this instance.

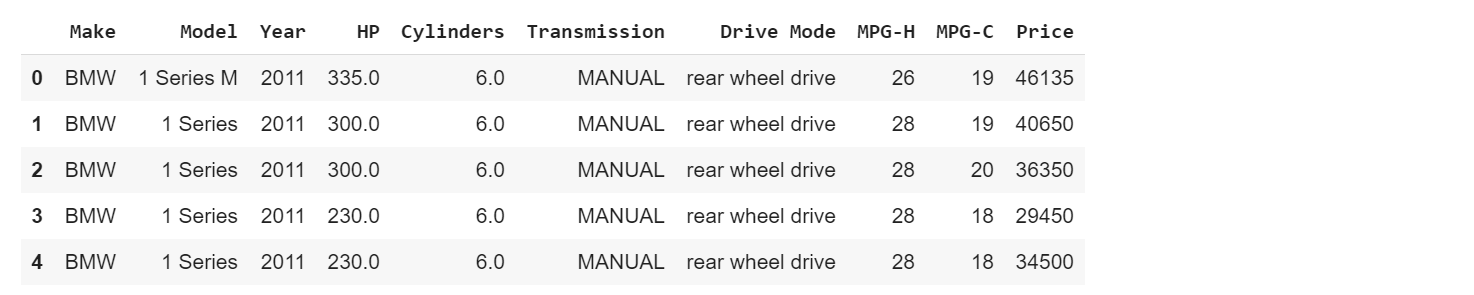
df = df.drop([‘Engine Fuel Type’, ‘Market Category’, ‘Vehicle Style’, ‘Popularity’, ‘Number of Doors’, ‘Vehicle Size’], axis=1)  
df.head(5)



**5. Renaming the columns**

In this instance, most of the column names are very confusing to read, so I just tweaked their column names. This is a good approach it improves the readability of the data set.

df = df.rename(columns={“Engine HP”: “HP”, “Engine Cylinders”: “Cylinders”, “Transmission Type”: “Transmission”, “Driven\_Wheels”: “Drive Mode”,”highway MPG”: “MPG-H”, “city mpg”: “MPG-C”, “MSRP”: “Price” })  
df.head(5)



**6. Dropping the duplicate rows**

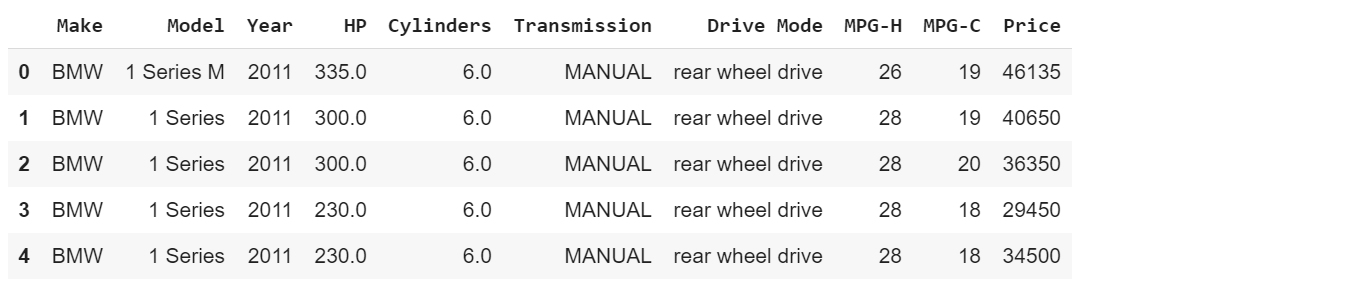
This is often a handy thing to do because a huge data set as in this case contains more than 10, 000 rows often have some duplicate data which might be disturbing, so here I remove all the duplicate value from the data-set. For example prior to removing I had 11914 rows of data but after removing the duplicates 10925 data meaning that I had 989 of duplicate data.

df.shape**(11914, 10)****# Rows containing duplicate data**

duplicate\_rows\_df = df[df.duplicated()]

print(“number of duplicate rows: “, duplicate\_rows\_df.shape)**number of duplicate rows: (989, 10)**

df = df.drop\_duplicates()  
df.head(5)



**7. Dropping the missing or null values.**

This is mostly similar to the previous step but in here all the missing values are detected and are dropped later. Now, this is not a good approach to do so, because many people just replace the missing values with the mean or the average of that column, but in this case, I just dropped that missing values. This is because there is nearly 100 missing value compared to 10, 000 values this is a small number and this is negligible so I just dropped those values.

**# Finding the null values.**

print(df.isnull().sum())

Make 0   
Model 0   
Year 0   
HP 69   
Cylinders 30   
Transmission 0   
Drive Mode 0   
MPG-H 0   
MPG-C 0   
Price 0   
dtype: int64

**# Dropping the missing values.**  
df = df.dropna()   
df.count()

Make 10827   
Model 10827   
Year 10827   
HP 10827   
Cylinders 10827   
Transmission 10827   
Drive Mode 10827   
MPG-H 10827   
MPG-C 10827   
Price 10827   
dtype: int64

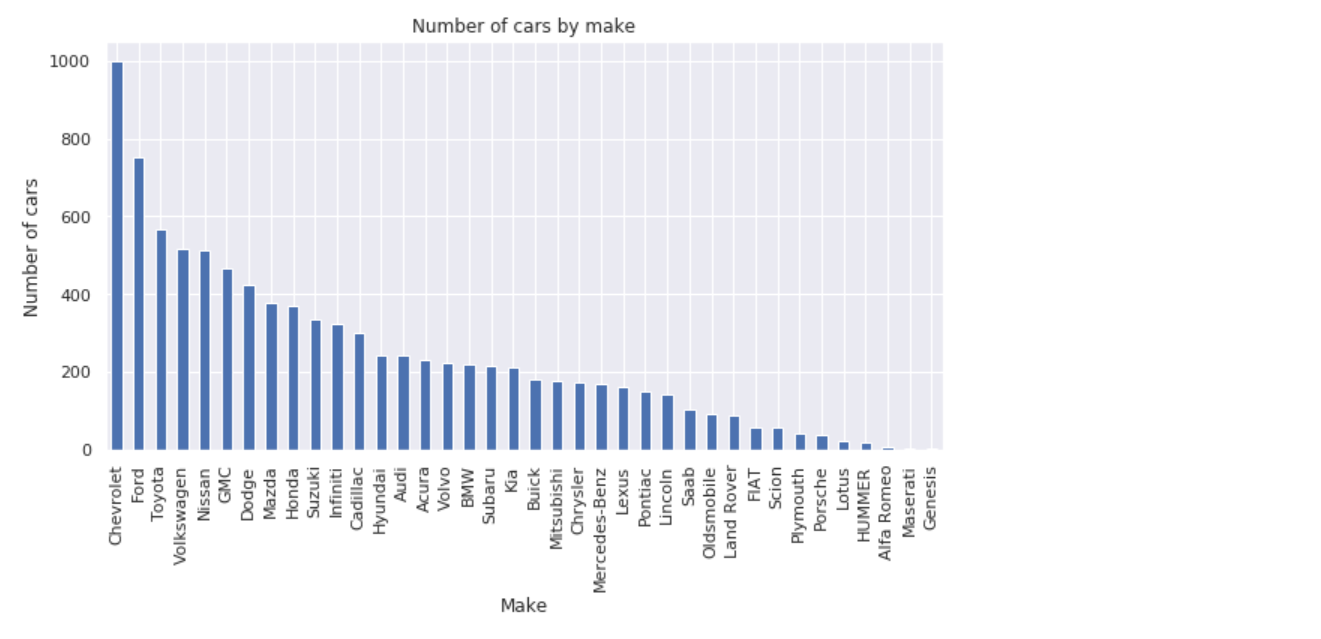
**8. Detecting Outliers**

An outlier is a point or set of points that are different from other points. Sometimes they can be very high or very low. It’s often a good idea to detect and remove the outliers. Because outliers are one of the primary reasons for resulting in a less accurate model. Hence it’s a good idea to remove them. The outlier detection and removing that I am going to perform is called IQR score technique. Often outliers can be seen with visualizations using a box plot. Shown below are the box plot of MSRP, Cylinders, Horsepower and EngineSize. Herein all the plots, you can find some points are outside the box they are none other than outliers.

**9. Plot different features against one another (scatter), against frequency (histogram**)

**9.1 Histogram**

df.Make.value\_counts().nlargest(40).plot(kind=’bar’, figsize=(10,5))  
plt.title(“Number of cars by make”)  
plt.ylabel(‘Number of cars’)  
plt.xlabel(‘Make’);  **fig 3.17 Histogram (car)**



**9.2Heat Maps**

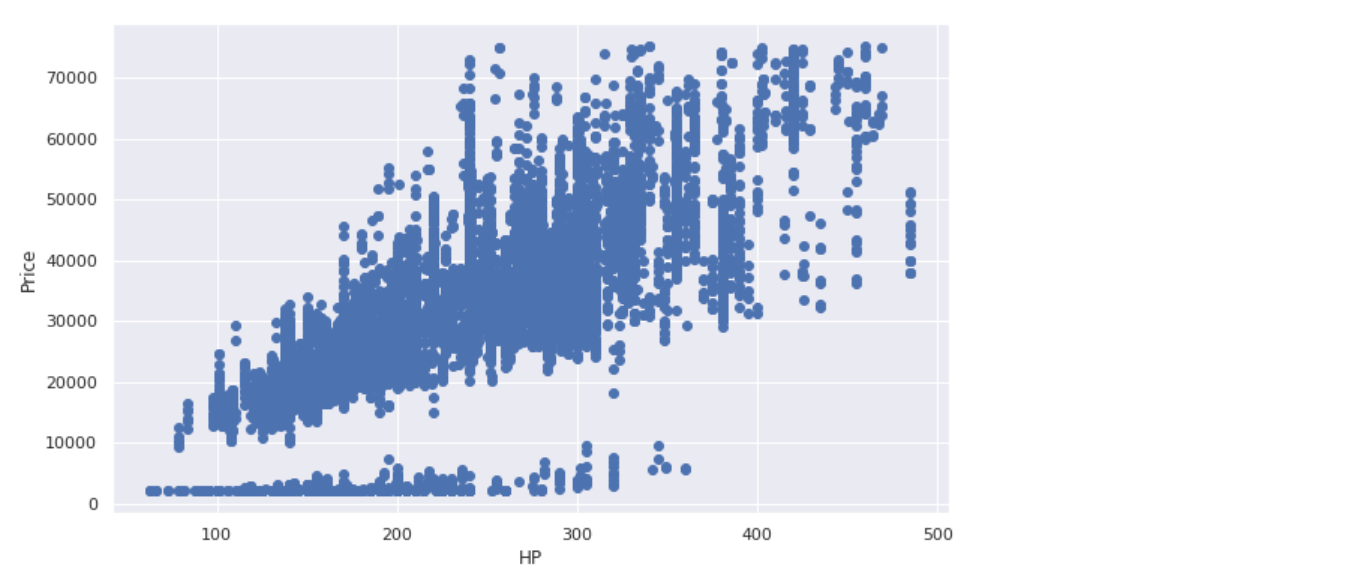
Heat Maps is a type of plot which is necessary when we need to find the dependent variables. One of the best way to find the relationship between the features can be done using heat maps. In the below heat map we know that the price feature depends mainly on the Engine Size, Horsepower, and Cylinders.

**Fig 3.18 Heatmap**

plt.figure(figsize=(20,10)) c= df.corr()sns.heatmap(c,cmap=”BrBG”,annot=True)

**9.3Scatterplot**

We generally use scatter plots to find the correlation between two variables. Here the scatter plots are plotted between Horsepower and Price and we can see the plot below.

fig, ax = plt.subplots(figsize=(10,6))ax.scatter(df[‘HP’], df[‘Price’])ax.set\_xlabel(‘HP’)ax.set\_ylabel(‘Price’)plt.show()

**fig 3.19 Scatterplot**

**Chapter 4**

**MY WORK**

# 4.1 The data:

The data I used in this analysis was provided to me by my company it contains data for various asian countries, but in this project I’m going to focus on data for India. In particular, I used these two datasets:

**(a)Time series**, the CSVfile with load, wind and solar, prices in hourly resolution.

**(b)Weather data** with wind speed, radiation, temperature and other measurements. Given the huge amount of data I was provided with a CSVfile to be used, mainly. And a script to download other data if required.

First, we start with the CSV file with the time series ,but read only the data of our use.

production = pd.read\_csv("data/time\_series\_60min\_singleindex.csv",

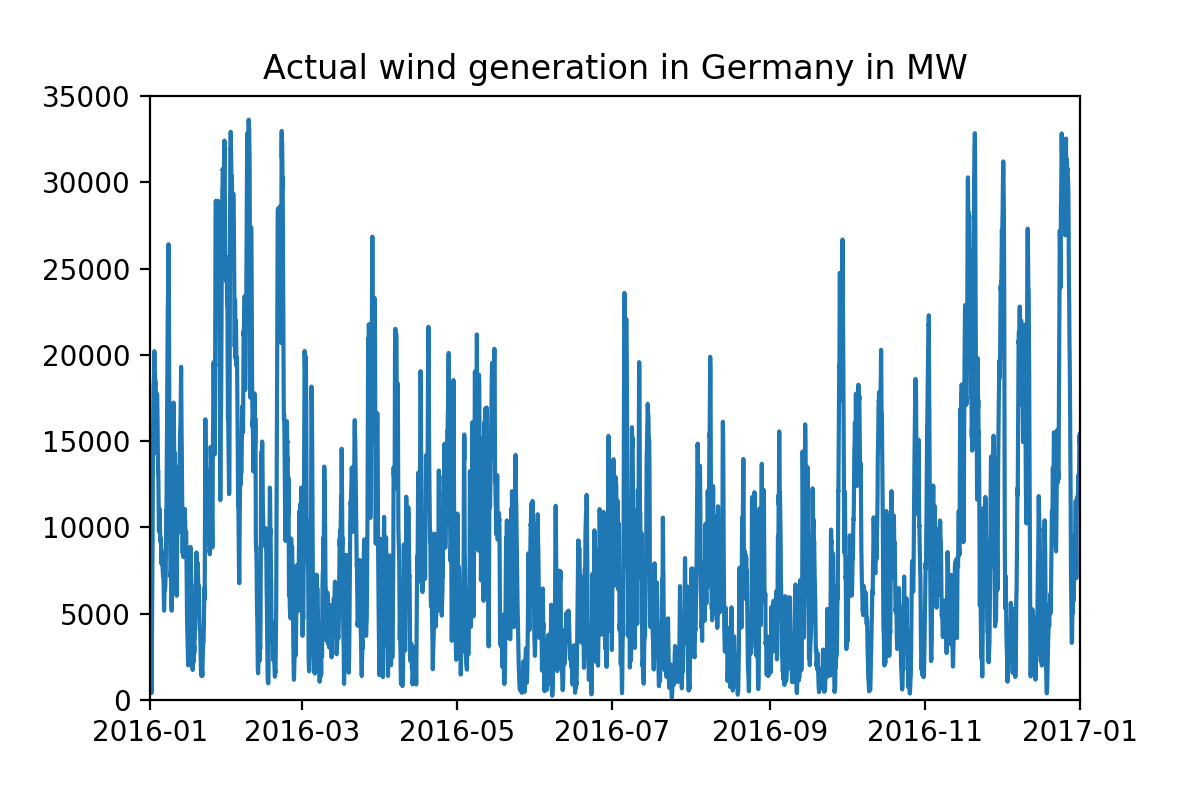
usecols=(lambda s: s.startswith('utc')s.startswith('DE')),parse\_dates=[0], index\_col=0)

Using the optionparse\_dates=[0], together with index\_col=0, guarantees that the column with the date and time of each measurement is stored as a DatetimeIndex.

After filtering for the rows for 2019, we end up with a DataFrame with 8784 entries and 48 columns, each relative to a different quantity such as solar capacity, wind capacity, etc. We were only interested in two of them:

* ‘DE\_solar\_generation\_actual’, with the actual solar generation in MW;
* ‘DE\_wind\_generation\_actual’, with the actual wind generation in MW.

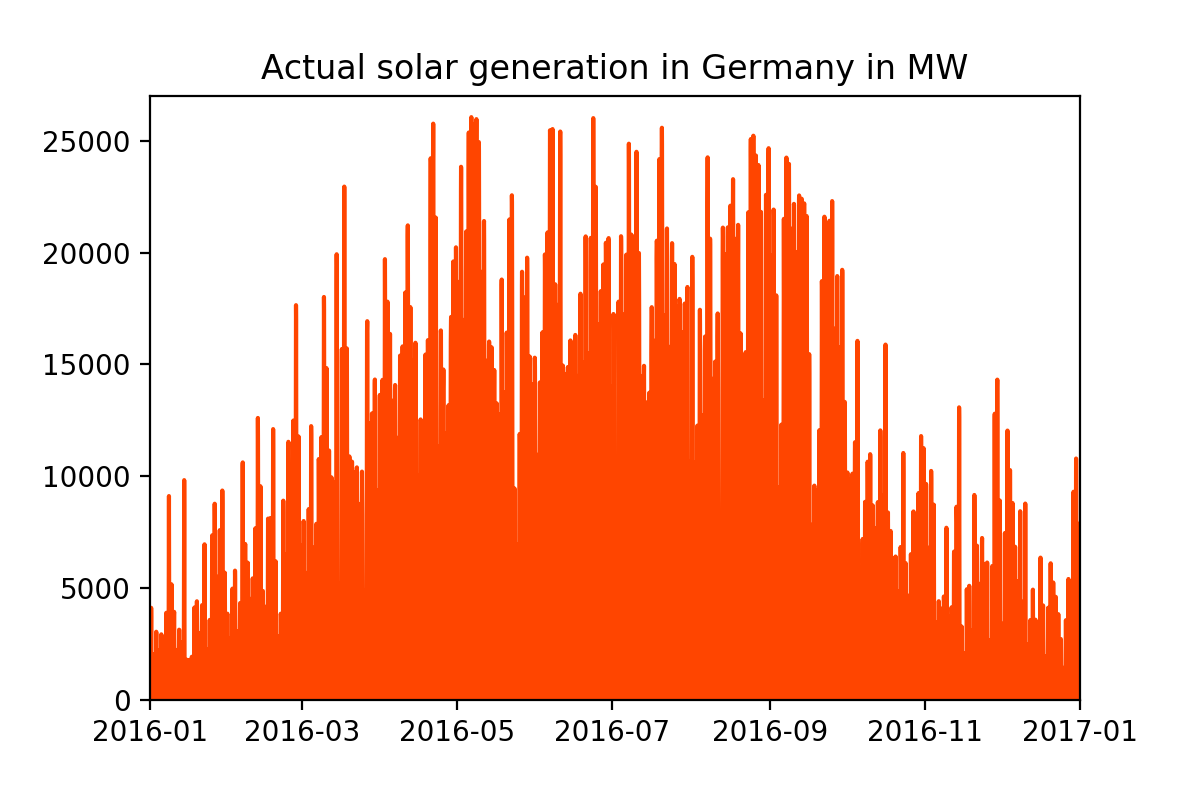
Fortunately, there are no missing values to worry about, so we made a couple of plots to have an idea about the data.



**fig 4.1**

As we would naively expect, there is no clear pattern for the wind generation across the year, even though we can see slightly larger production roughly in February, November and December.

As for the solar generation, as expected, it was significantly larger in the middle months of the year.

****

**Fig 4.2**

## **4.2 Weather Data**

Now, we read the CSV file containg the weather data

weather = pd.read\_csv("data/weather\_data\_GER\_2016.csv",parse\_dates=[0], index\_col=0)

If we check the info atribute of the weatherDataFrame, we obtain:

<class 'pandas.core.frame.DataFrame'>DatetimeIndex: 2248704 entries, 2016-01-01 00:00:00 to 2016-12-31 23:00:00Data columns (total 14 columns):cumulated hours int64lat float64lon float64v1 float64v2 float64v\_50m float64h1 int64h2 int64z0 float64SWTDN float64SWGDN float64T float64rho float64p float64dtypes: float64(11), int64(3)memory usage: 257.3 MB

That’s 2248704 entries! If you do the maths, it corresponds to 8784 entries for 256 geographical ‘chuncks’ of Germany, each characterized by its latitute lat and longitute lon. The other columns are as follows:

*Wind parameters:*

* v1: velocity [m/s] at height h1 (2 meters above displacement height);
* v2: velocity [m/s] at height h2 (10 meters above displacement height);
* v\_50m: velocity [m/s] at 50 meters above ground;
* h1: height above ground [m] (h1 = displacement height +2m);
* h2: height above ground [m] (h2 = displacement height +10m);
* z0: roughness length [m];

*Solar parameters:*

* SWTDN: total top-of-the-atmosphere horizontal radiation [W/m²];
* SWGDN: total ground horizontal radiation [W/m²];

*Temperature data:*

* T: Temperature [K] at 2 meters above displacement height (see h1);

*Air data:*

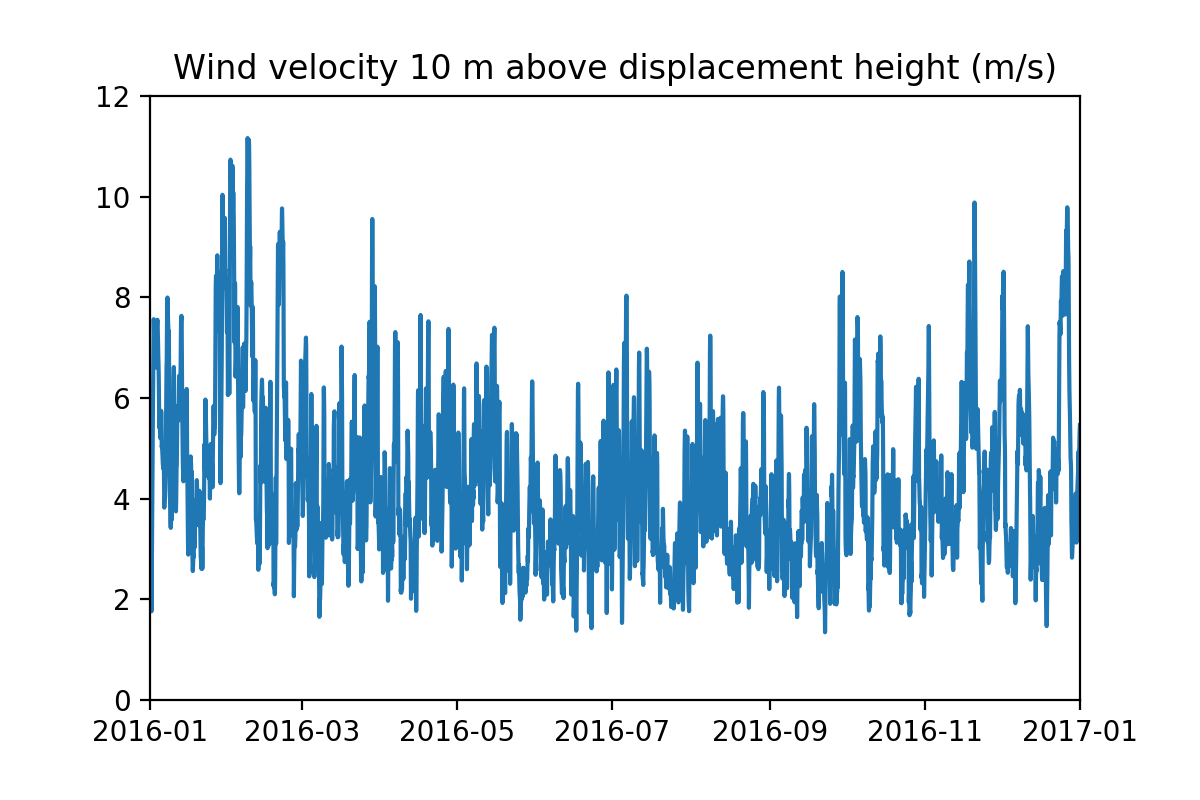
* Rho: air density [kg/m³] at surface;
* p: air pressure [Pa] at surface.

**A**t this point, we have information about the wind and solar generation at a *national*level and information about the weather at a more *local*level

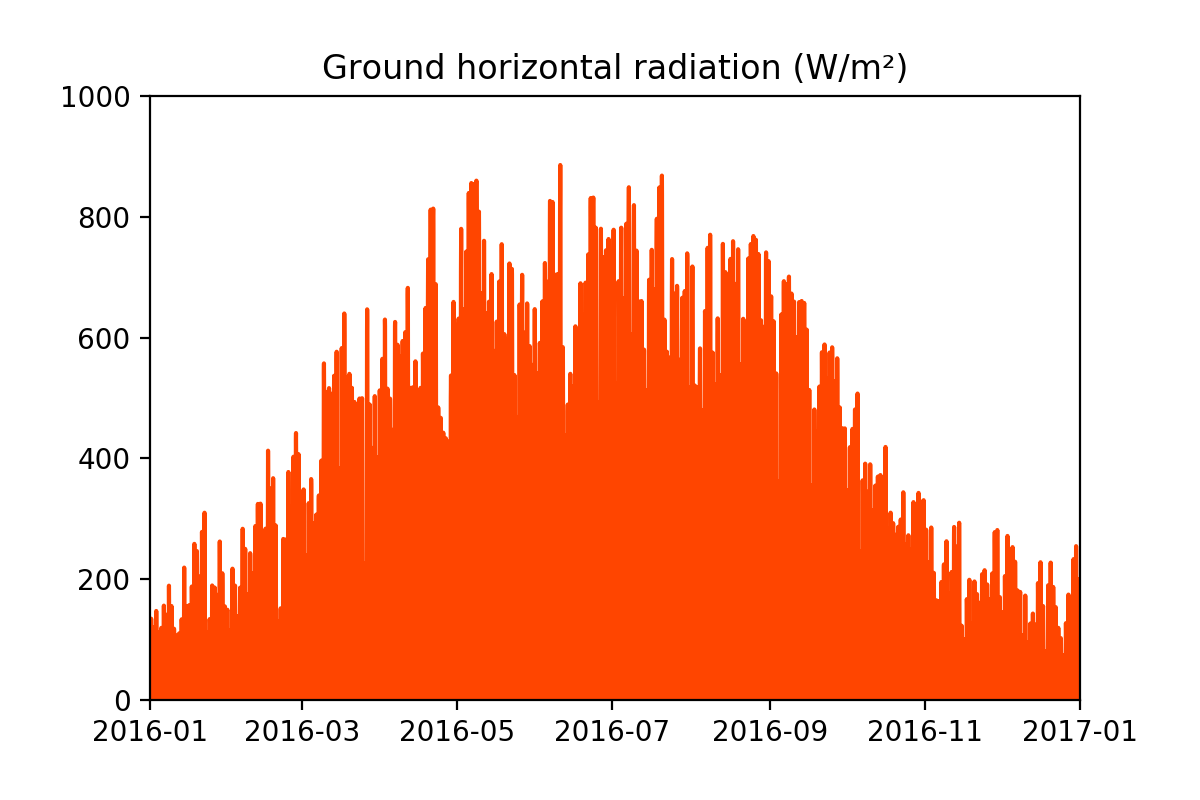
We have some limitations for a complete analysis, as, for instance, we don’t know the location within India of the wind turbines and solar panels. Given the purposes of this project, I’m going to simply group the weather data by each hour and take the average over the geographical ‘chucks’. In this way, we obtain a DataFrame with 8784 entries, which we can later merge with the first DataFrame.

Before doing that, let’s see how some of these averaged weather quantities behaved in 2019.

As with the wind generation, we see that the wind velocity does not follow a specific pattern, although it was larger in February, November and December.



**Fig 4.5**

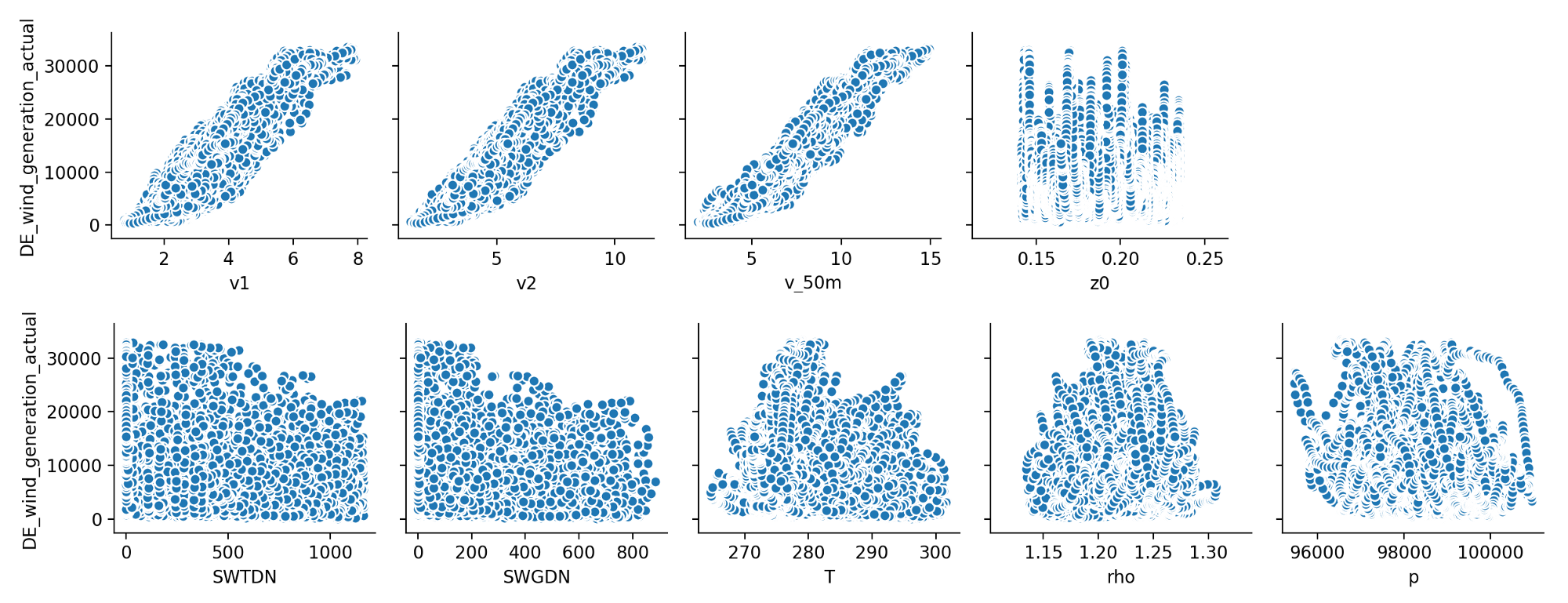


**Fig 4.6**

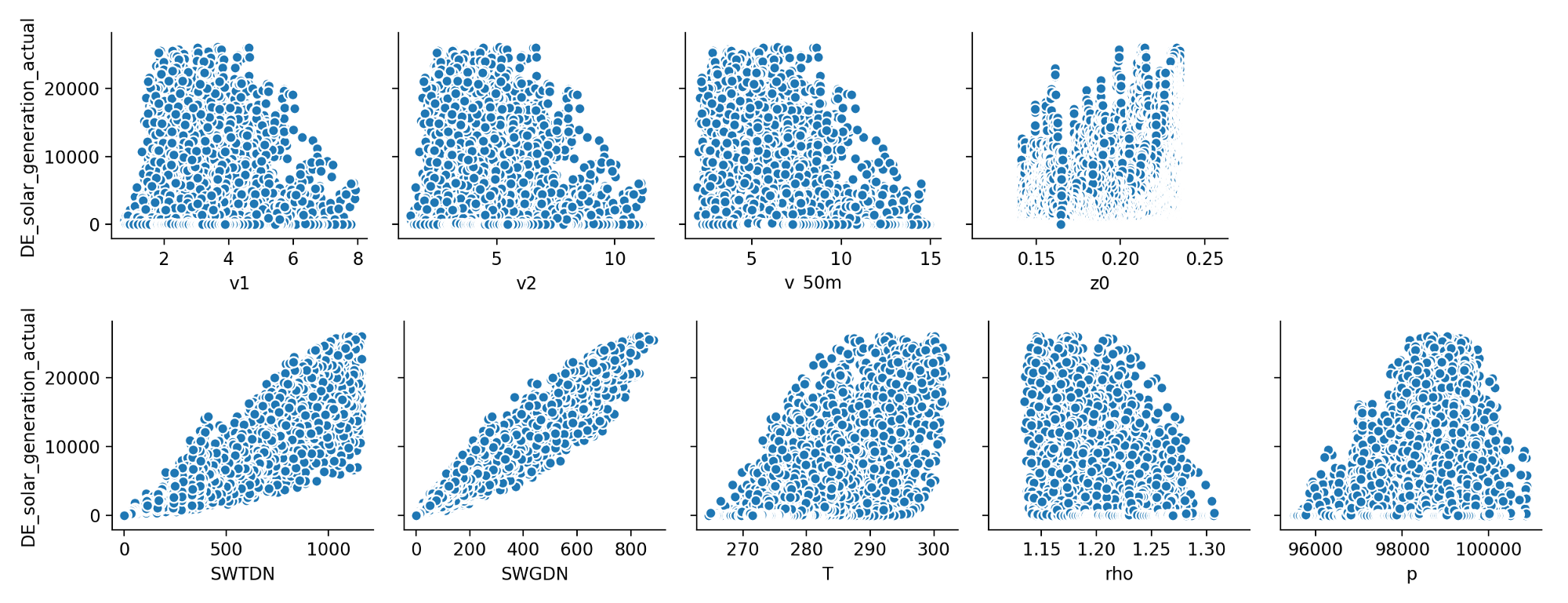
The horizontal radiation at the ground level was, as expected, larger during the Summer months, likewise with the temperature, as plotted below.

As suggested by the above plots from both datasets (and by common sense), there seems to be some correlation between the wind and solar generation and some of the measured weather quantities. Further evidence for this claim can be obtained from the following plots, in which the wind and solar generation is shown as a function of the several weather quantities.

**Fig 4.7**



# There seems to be a linear relation between the wind generation and the wind velocities v1, v2 and v\_50m, but not the other quantities.



**4.3 Predicting the wind and solar generation using linear regression**

The output of a linear regression algorithmis a linear function of the input:

where

is a vector of parameters.The objective is to find the parameters which minimize the mean squared error:

This can be achieved usingLinearRegression from the scikit-learn library.

## 

## **4.3.1 Wind generation**

To predict the wind generation, we construct the features matrix X\_wind with the features v1, v2 and v\_50m, and the target Y\_wind with actual wind generation. Then, we implement the algorithm:

**from** sklearn.linear\_model **import** LinearRegression

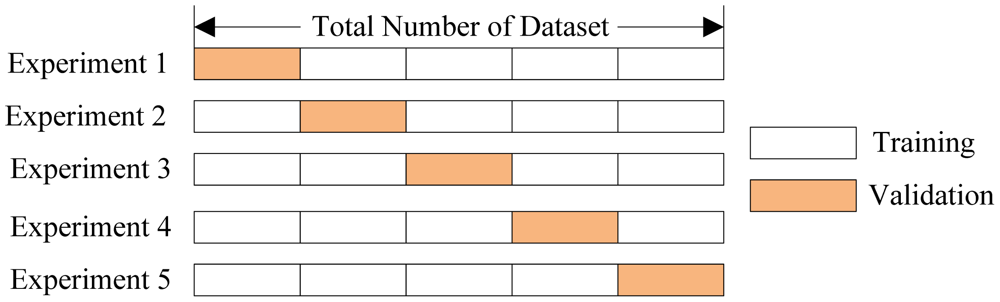
**from** sklearn.model\_selection **import** cross\_val\_scorelr = LinearRegression()scores\_wind = cross\_val\_score(lr, X\_wind, y\_wind,cv=5)

print(scores\_wind, "\naverage =", np.mean(scores\_wind))

We import LinearRegression from sklearn.linear\_model, which implements the ordinary least squares linear regression. (You can find more information in the [documentation](_blank).)

* In order to evaluate the performance of the algorithm, we divide the data using a procedure called **cross-validation** (CV for short). For the *k*-fold CV, the dataset is split into *k* smaller sets or ‘folds’, the model is trained in *k*-1 of those folds, and the resulting model is validated on the remaining part of the data. The performance measure provided by the CV is then the average of the performance measure computed in each experiment. In the code above, we use cross\_val\_score from sklearn.model\_selection, with number of folds cv=5 (more information in the [documentation](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html)).

The performance measure that LinearRegression gives by default is the **coefficient of determination** *R*² of the prediction. It measures how well the predictions approximate the true values. A value close to 1 means that the regression makes predictions which are close to the true values. It is formally computed using the formula:



The output of the code above for our case is:

[0.88261401 0.88886305 0.83623262 0.88974363 0.85338174]   
average = 0.870167010172279

The first line contains the five values of *R*² for each of the 5 folds in the cross validation procedure, whereas the second line is their average. We see that our linear model has an *R*² of approximately 0.87, which is quite good.

## **4.3.2 Solar generation**

To predict the solar generation, we follow a very similar procedure. We again construct the features matrix X\_solar, but now with the features SWTDN, SWGDN and T, and the target Y\_solar with actual solar generation. Then, we implement the algorithm:

scores\_solar = cross\_val\_score(lr, X\_solar, y\_solar, cv=5)

print(scores\_solar, "\naverage =", np.mean(scores\_solar))

The output is:

[0.8901974 0.95027431 0.95982151 0.95090201 0.8715077 ]

average = 0.9245405855731855

We get an even better value of *R*²! We can make good predictions about the solar generation given only the temperature and top-of-the-atmosphere and ground radiation.

Even with these good results, there is certainly much we can do to improve the analysis. As an example, it is very probable that some of the features used in the regression are *collinear*, that is, they are moderately or highly correlated. For a first analysis like this, in which we are only interested in the predictive power of the model, it is not a major concern. However, without further work, we cannot say much about the influence of each individual feature on the wind and solar generation.

It’s quite impressive how much we were able to accomplish using real-world data and a simple algorithm.

**4.4 Electron js**

Electron(formerly known as Atom Shell) is an open-source framework developed and maintained by GitHub Electron allows for the development of desktop GUI applications using web technologies: It combines the Chromium rendering engine and the runtime. Electron is the main [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) framework behind several notable open-source projects including

[Atom](https://en.wikipedia.org/wiki/Atom_(text_editor)),Github, [Light Table](https://en.wikipedia.org/wiki/Light_Table_(software)), [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code) and WordPress Desktop.

**4.4.2 Python Shell**

A simple way to run Python scripts from Node.js with basic but efficient inter-process communication and better error handling.

## **Features**

* Reliably spawn Python scripts in a child process
* Built-in text, JSON and binary modes
* Custom parsers and formatters
* Simple and efficient data transfers through stdin and stdout streams
* Extended stack traces when an error is thrown Installation

npm install python-shell

**4.4.3 Working Of The Application**

The basic working of the application consists of taking date and location as input from the user and providing the predicted values of the generation of solar or wind energy, as required, at that particular location and on that particular date as the output.

The input taken from users is send to the API using python script for weather forecast. The API sends back weather information which is converted using formulas and fed into the model to get the prediction. Finally, the predicted values are shown as output on the application. The parameter required by the model for prediction can be identified in the following two ways:

Before we see the two methods it is a must for us to remember the parameters our model requires for making the prediction. As this is already discussed in the above content, for wind and solar generation prediction model only requires wind velocity at two different heights and radiation at different levels respectively.

Now, the ways in which data can be found are:

1. **By using a weather API :** For prediction of wind energy the API will provide us with the wind velocity at a single height. Now, we have a formula to get wind velocity at different heights. It is called as the power law formulae which is as follows:

V(z)=Vr(Z/Zr)^2

where V(z) is speed of wind at required height Z, Vr is wind speed at reference height Zr and β is surface roughness length.

Similarly, we will get the solar radiation information from the api.

We have a formula to find solar radiations, one for the morning and another for evening. From this formula we get the required solar radiations to feed into the model.

2. **Using past few year’s data :** If we don’t have forecast data as we are using the free api, we can use past few year’s data of the same location and date. An average of all the required values will give us input values to use in our model.

**Chapter 5**

**Learning after training**

­Working on this project gave me deep insight about how things move in a professional environment. While working in an company is fun but it equally challenging too. While working on the development of this project I got plenty of opportunities to hone

my coding skills and use my knowledge to solve software engineering problems as well as learn new things that are very useful in any software development project.

This project gave me deep insights about working on Data Science concepts such as Exploratory data analysis and Data visualization.

Making a complete project from finding solution by training a model to creating an application to be used for quick results was a great learning experience for me.

On the technical front this project taught me a great deal about using different concepts of data science , using different algorithms and about creating a trained model. It also taught me about electron js and python shell which helped me built the application.

While the technical learning greatly enhanced my skills I also got to learn to deal with ever changing companies demands and to work under the pressure of a deadline. This proved to be a very new and different experience as the exposure to deal with people gave me a lot of confidence to deal with what would happen in a full time corporate job.

**Chapter 6**

**Discussion**

While doing this project a great many things caught my attention. First and foremost was how even small applications can cause a big difference in the society. Developing this application was not only a great learning experience it also gave me a sense of satisfaction as to how technology can affect and enhance a companies decisions and lives of so many people at once.

On a more personal front I have the satisfaction that this industrial training taught me so many aspects about corporate life and also on the technical aspects of it.

Learning to apply lessons learnt in theory to real life software development gave me a hands on experience about the challenges that lie ahead in the industry. I feel that my technical skills have enhanced and also the ability to use my skills to solve complex problems has also increased. I have also learnt how to deal with clients and their ever changing demands. Towards the end of this discussion I would like to express my extreme sense of satisfaction that I got at the end of this project.

**Chapter 7**

**Conclusion**

Predicting solar and wind generation system is a very dynamic project that employs the knowledge of Exploratory data analysis, Data Visualization, Electron js and Python Shell along with the building a model and using different algorithms such as linear regression algorithm.

This system provides the user with the predicted value of generation of renewable energy, solar or wind as required at a particular location , on a particular date.

I developed this project for Free wings Infra and Power Limited and the proprietary rights to this software lie with them.

Going through this industrial training taught me a lot of things about project development.

At the end of this training I feel much more mature and confident about my technical skills and also better equipped to handle the day to day nuances of the corporate world.