# Data Analytics

Seunghoi Kim, Byoung Hun Min, Kiet Tran,
Wei Jiang, Sandiarta Hadiprawira

# Task I: Part A

**Exploring the dataset**

- **How many cities are there in the dataset?**
  The dataset was **imported** and parsed into a **Pandas DataFrame** object using the *read_pickle* function. By printing the DataFrame object and its column values, we found there were **5 different cities**: Aalborg, Aarhus, Esbjerg, Odense and Roskilde.

- **How many observations and features are there in this dataset?**
  By printing the *shape* of the DataFrame, we found the **dimensions** of the dataset:
  **333110 rows** x **20 columns**.
  The number of **rows** in the dataset is **equivalent** to the number of **observations**. Thus, we can deduce that there are **333110 observations** in this dataset. The number of columns in the dataset is 20. However, we know there are 5 different cities from our earlier observations. Hence, we can compute the number of features which is 20 / 5 = **4 features**.

- **What are the names of the different features?**
  The names of the different features were found by printing the columns of the DataFrame table.
  The feature names were as follows: Temp, Pressure, WindSpeed, and WindDir.

```
df = pd.read_pickle("weather-denmark-resampled.pkl")
```

```
print(df.columns)
#print(df)
```

```
MultiIndex([( 'Aalborg',      'Temp'),
            ( 'Aalborg',  'Pressure'),
            ( 'Aalborg', 'WindSpeed'),
            ( 'Aalborg',   'WindDir'),
            ( 'Aarhus',      'Temp'),
            ( 'Aarhus',  'Pressure'),
            ( 'Aarhus', 'WindSpeed'),
            ( 'Aarhus',   'WindDir'),
            ( 'Esbjerg',      'Temp'),
            ( 'Esbjerg',  'Pressure'),
            ( 'Esbjerg', 'WindSpeed'),
            ( 'Esbjerg',   'WindDir'),
            ( 'Odense',      'Temp'),
            ( 'Odense',  'Pressure'),
            ( 'Odense', 'WindSpeed'),
            ( 'Odense',   'WindDir'),
            ('Roskilde',      'Temp'),
            ('Roskilde',  'Pressure'),
            ('Roskilde', 'WindSpeed'),
            ('Roskilde',   'WindDir')],
           )
```

```
print(df.shape)
```

```
(333110, 20)
```

# Task I: Part B

**Evaluating if the dataset contains any missing values and removing them using the pandas built-in function.**

The Pandas built-in function: `isna` is used to identify the missing values in the dataset.

The Pandas dataframe `dropna` function is used to remove all the missing values from the dataset.

By observing the shape of the dataframe before and after the `dropna` function is applied we can see that 333110 - 332070 = **1040 observations** contained missing values and were therefore removed.

```
##########################
# Removing missing values
new_df = df.dropna()

# Comparing changes in the number of observations
print(df.shape[0])
print(new_df.shape[0])
print(df.shape[0] - new_df.shape[0])

##########################
```

```
333110
332070
1040
```

# Task I: Part C

**Extracting general statistics like minimum, maximum, mean, median and standard deviation.**

We parse through the previously created Pandas Dataframe table with the missing values removed and calculate the min, max, median, mean and standard deviation values for each city and feature.

```
cities = ['Aalborg', 'Aarhus', 'Esbjerg', 'Odense', 'Roskilde']
features = ['Temp', 'Pressure', 'WindSpeed', 'WindDir']

for c in cities:
    print(c + " statistics")
    print("**************************")
    for f in features:
        print(f + " - max: " + "{:.1f}".format(new_df[(c, f)].max()) + ", min: "
              + "{:.1f}".format(new_df[(c, f)].min()) + ", median: "
              + "{:.1f}".format(new_df[(c, f)].median()) + ", mean: "
              + "{:.1f}".format(new_df[(c, f)].mean()) + ", std: "
              + "{:.1f}".format(new_df[(c, f)].std()))
    print("**************************")
```

## Results:

Aalborg statistics
**********
Temp - max: 30.8, min: -25.0, median: 8.1, mean: 8.3, std: 7.0
Pressure - max: 1050.8, min: 951.9, median: 1013.4, mean: 1012.7, std: 11.7
WindSpeed - max: 32.9, min: 0.0, median: 4.6, mean: 4.9, std: 2.8
WindDir - max: 360.0, min: 10.0, median: 210.0, mean: 192.4, std: 88.0
**********

Aarhus statistics
**********
Temp - max: 30.9, min: -24.3, median: 8.0, mean: 8.3, std: 7.0
Pressure - max: 1050.0, min: 955.5, median: 1014.0, mean: 1013.3, std: 11.3
WindSpeed - max: 33.4, min: 0.0, median: 3.6, mean: 4.0, std: 2.5
WindDir - max: 360.0, min: 10.0, median: 213.3, mean: 201.4, std: 82.1
**********

Esbjerg statistics
**********
Temp - max: 54.0, min: -27.0, median: 8.3, mean: 8.6, std: 6.7
Pressure - max: 1049.3, min: 959.3, median: 1014.1, mean: 1013.1, std: 10.9
WindSpeed - max: 39.1, min: 0.0, median: 4.5, mean: 4.9, std: 2.7
WindDir - max: 360.0, min: 10.0, median: 216.7, mean: 201.8, std: 87.8
**********

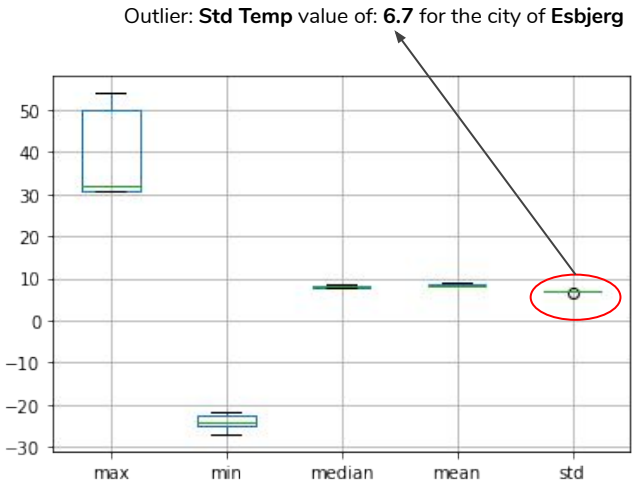
Odense statistics
**********
Temp - max: 49.9, min: -22.5, median: 8.7, mean: 8.8, std: 6.9
Pressure - max: 1048.9, min: 959.7, median: 1014.4, mean: 1013.8, std: 11.0
WindSpeed - max: 62.5, min: 0.0, median: 4.6, mean: 4.8, std: 2.8
WindDir - max: 360.0, min: 10.0, median: 210.0, mean: 195.9, std: 83.7
**********

Roskilde statistics
**********
Temp - max: 32.0, min: -21.8, median: 8.0, mean: 8.3, std: 7.1
Pressure - max: 1048.1, min: 959.8, median: 1013.8, mean: 1012.8, std: 11.7
WindSpeed - max: 25.0, min: 0.0, median: 4.4, mean: 4.8, std: 2.8
WindDir - max: 360.0, min: 10.0, median: 220.0, mean: 202.8, std: 86.1
**********

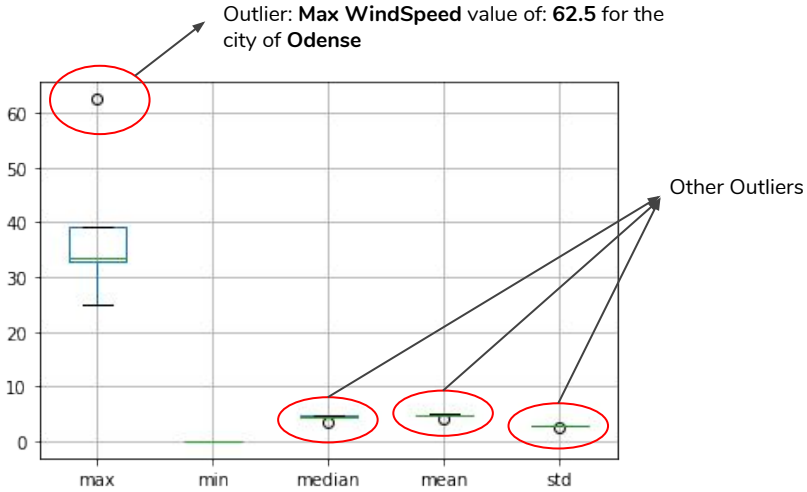
# Task I: Part C

**Spotting any anomalies in the properties**

We generated Boxplots for each property and the boxplots for Temperature and Wind Speed data displayed several outliers in the properties which indicate **potential anomalies.**
Further analysis is required to determine whether these outliers can be classified as anomalies.




**Boxplot A: Temperature Statistics for each city**



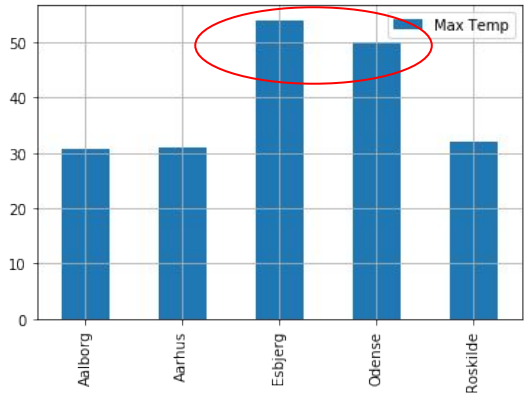

**Boxplot B: Wind Speed Statistics for each city**

# Task I: Part C

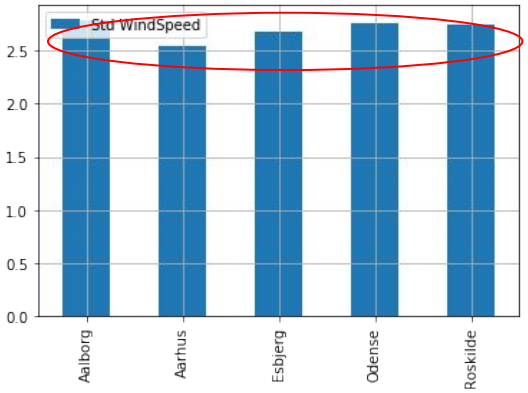
**Spotting any anomalies in the properties**

The outliers identified in the previous slide were analysed in more depth using bar charts for each feature property.

The following bar charts show that two of the outliers have insufficient evidence to be classified as anomalies.




Max Temp of Esbjerg and Odense have noticeably higher values than the other three cities. However, there is insufficient evidence to classify these as anomalies since there are only five cities in total.

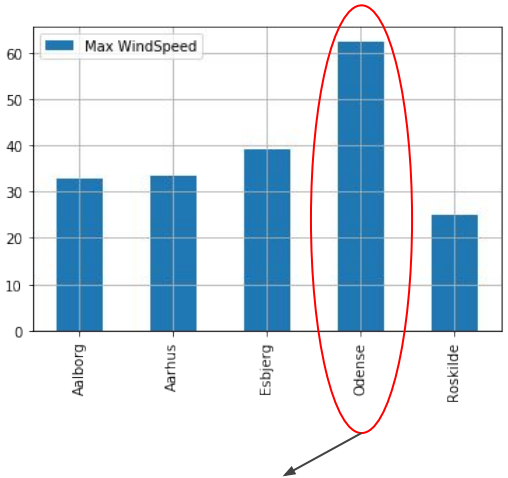



The standard deviation (std) of the Wind Speed of Aarhus has a lower value than the rest of the cities. However, it is only approximately 11% lower than Aalborg which has the highest value of the five cities. Therefore, this is insufficient evidence to classify the std wind speed of Aarhus as an anomaly.

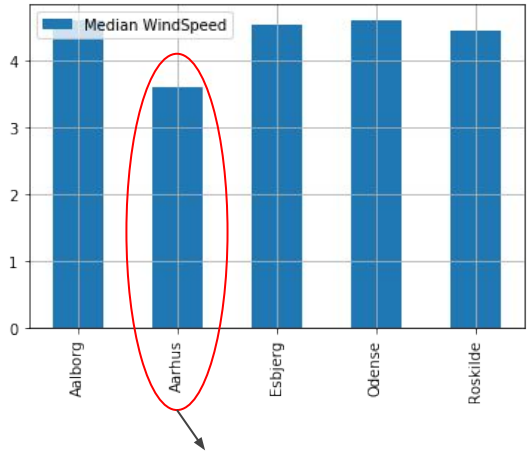
# Task I: Part C

**Spotting any anomalies in the properties**

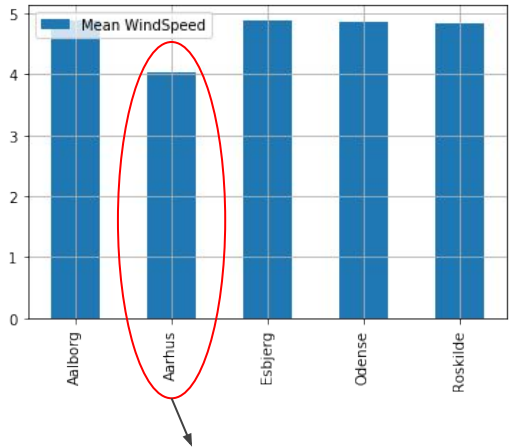
The following bar charts show that three of the outliers can be classified as anomalies.




The maximum Wind Speed property of Odense can be classified as an anomaly. Its value is 62.5, while the same property for all the other cities range between 25.0 and 39.1.




The median Wind Speed property of Aarhus can also be classified as an anomaly. Its value is 3.6, while the same property for all the other cities lie strictly between 4.4 and 4.6 which is a difference of over 20%.




The mean Wind Speed property of Aarhus can also be classified as an anomaly for similar reasons. Its value is 4.0 compared to the other cities which have mean Wind Speed values of 4.8 and 4.9.

# Task I: Motivation

- **Pandas library**

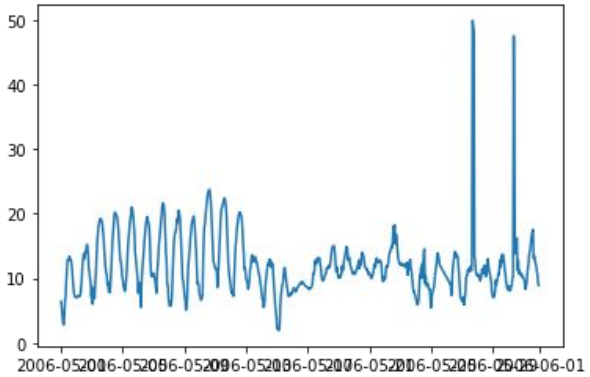
  The Pandas library contains various **useful built-in functions** and **data structures** including:
  - The **Pandas DataFrame** object which is useful for analysing and manipulating tabular data.
  - **read_pickle()** function which parses the pickle file into a DataFrame object
  - **dropna()** function which removes observations containing missing values from the dataset
  - various statistical functions including: min(), max(), median(), mean(), and std().
  - **boxplot()** function to produce a box plot from the dataset
  - **plot.bar()** function to produce a bar chart from the dataset

  The primary motivation for using the Pandas library is that Task I was entirely solvable using just the Pandas built-in functions mentioned above.

# Task II: Outliers

**Storing the temperature measurements in May 2006 for the city of Odense and producing a simple plot of the temperature versus time.**

The temperature measurements in May 2006 for the city of Odense were selected by filtering the DataFrame table. First, we referenced the multicolumn index: (‘Odense’, ‘Temp’) tuple to select just the temperature values for the city of Odense. Then, we filtered the values between the dates 2006-05-01 and 2006-06-01 resulting in the measurements taken in May 2006. The filtered data was plotted with Time (X-axis) against Temperature (Y-axis).

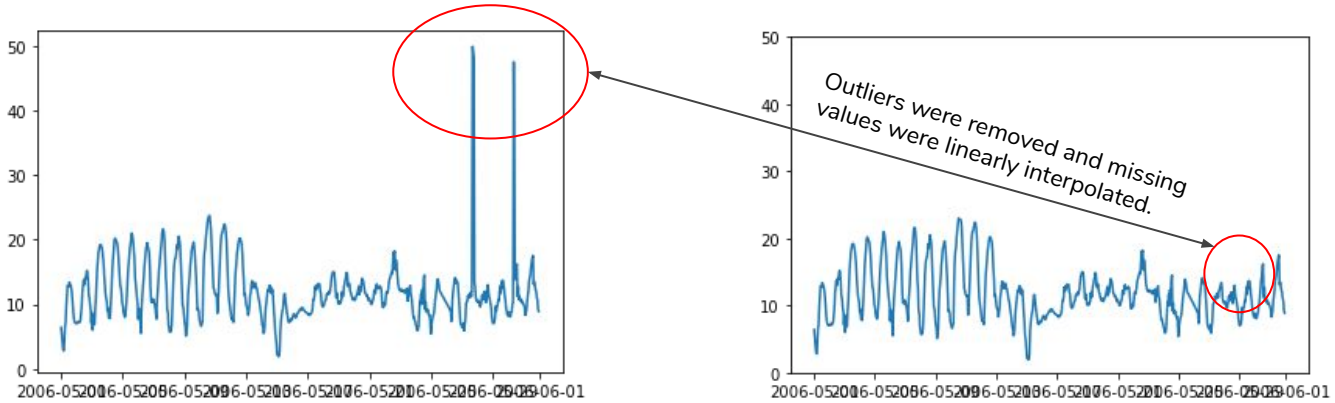



**Temperature/Time Plot of Odense on May 2006**

# Task II: Outliers

**Finding the outliers in this set of measurements (if any) and replacing them using linear interpolation.**

The outliers were found by first calculating the **mean** and **standard deviation** of the set of measurements. Then, calculating the **lower** and **upper bounds** defined by **3 standard deviations** from the mean. Any observations that weren't between the lower and upper bounds were **classified** as **outliers** and **removed** by setting as null.
The Pandas DataFrame built-in function ***interpolate()*** was used to linearly interpolate the missing values in the dataset.




**Temperature/Time Plot of Odense on May 2006**

**Temperature/Time Plot of Odense on May 2006 (Outliers replaced by Linear Interpolation)**

# Task II: Motivation

- **Pandas library**
  As with Task I, we used the Pandas library for Task II to pre-process the dataset and extract temperature measurements in May 2006 for the city of Odense.

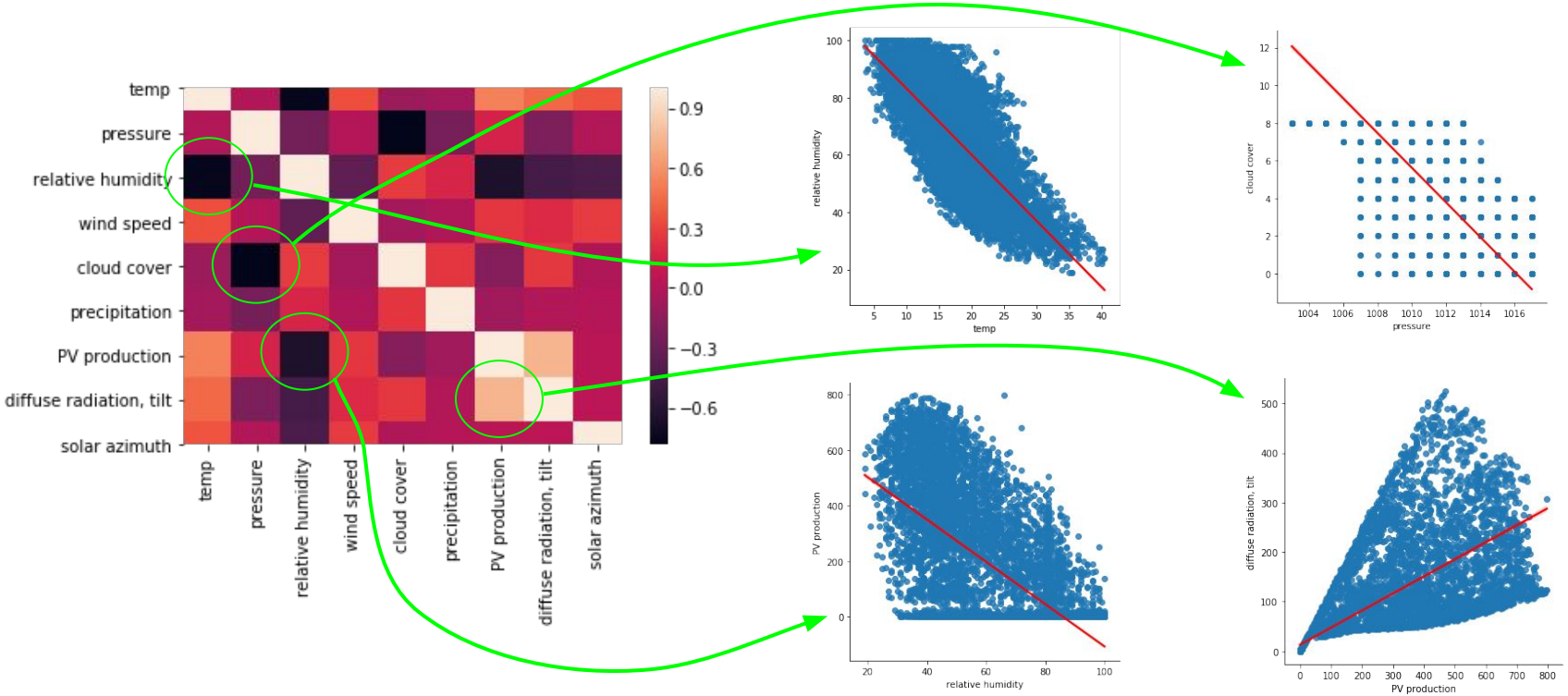
  We also used the **interpolate()** built-in function from the Pandas library to linearly interpolate the missing values in the dataset.

- **Matplot library**
  Using the Matplotlib **plot()** function, we were able to produce a simple temperature versus time plot to visualise the temperature measurement data.

  This library is also useful as it is compatible with the Pandas DataFrame object and can plot it directly without any further processing.

# Task III: Correlation

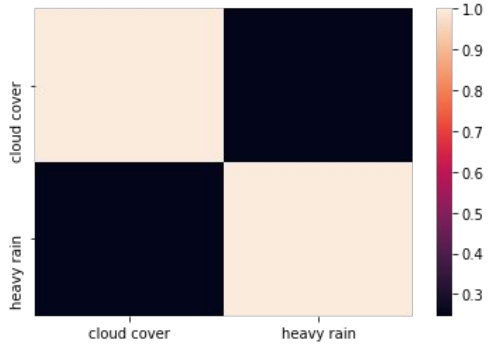
**Finding significant correlations between features in the df_perth.pkl dataset.**



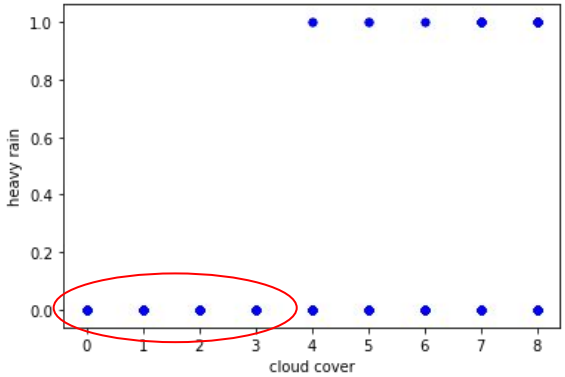
# Task III: Correlation

**Finding the correlation between precipitation and cloud cover and attempting to infer the probability of having moderate to heavy rain (> 1 mm/h) as a function of the cloud cover index**

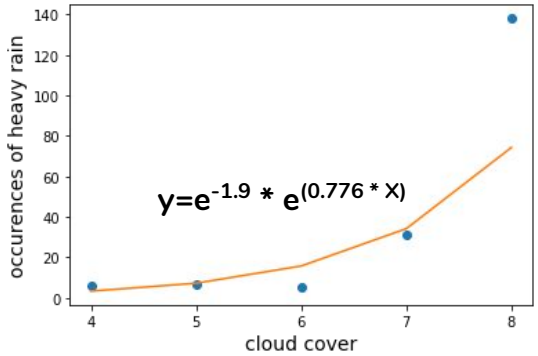
| | cloud cover | heavy rain |
|---|---|---|
| **cloud cover** | 1.000000 | 0.246993 |
| **heavy rain** | 0.246993 | 1.000000 |




**Heatmap** showing **correlation** between **cloud cover index** and **heavy rain** (> 1 mm/h)



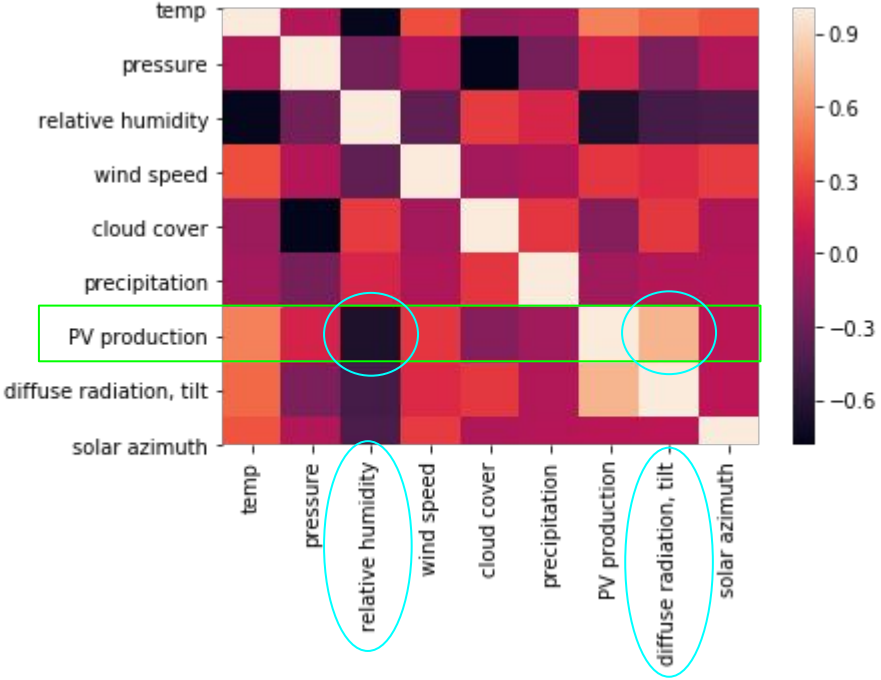

**Scatter** plot between **cloud cover** and **heavy rain** (> 1 mm/h)




**Exponential** plot between **cloud cover** and **number of occurrences** of **heavy rain** for cloud cover indexes: **4 ~ 8**

# Task III: Inference

**Finding the features that are statistically significant in modelling the target variable (PV production) in order to predict PV production using multiple linear regression.**




- We devised a heat map which represents the correlation between each feature.
- As shown in the image on the left, different colours have different correlation values.
- Hence, we chose features by the highest positively and negatively correlated features.
- We can observe that ‘diffuse radiation, tilt’ and ‘relative humidity’ have correlated values over +/-0.6 in relation to PV production.

# Task III: Inference

**Creating a multivariate model using the predictors chosen in the previous question.**

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | PV production | **R-squared:** | **0.552** |
| Model: | OLS | Adj. R-squared: | 0.552 |
| Method: | Least Squares | F-statistic: | 1.077e+04 |
| Date: | Tue, 31 Mar 2020 | Prob (F-statistic): | 0.00 |
| Time: | 10:01:50 | Log-Likelihood: | -56269. |
| No. Observations: | 8760 | AIC: | 1.125e+05 |
| Df Residuals: | 8758 | BIC: | 1.126e+05 |
| Df Model: | 1 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 53.2457 | 1.925 | 27.653 | 0.000 | 49.471 | 57.020 |
| **diffuse radiation, tilt** | **1.6023** | 0.015 | 103.794 | 0.000 | 1.572 | 1.633 |

| | | | |
|---|---|---|---|
| Omnibus: | 1872.113 | Durbin-Watson: | 0.161 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 3769.181 |
| Skew: | 1.277 | Prob(JB): | 0.00 |
| Kurtosis: | 4.951 | Cond. No. | 151. |

# Task III: Inference

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | PV production | **R-squared:** | **0.670** |
| Model: | OLS | Adj. R-squared: | 0.669 |
| Method: | Least Squares | F-statistic: | 8870. |
| Date: | Wed, 01 Apr 2020 | Prob (F-statistic): | 0.00 |
| Time: | 04:27:26 | Log-Likelihood: | -54932. |
| No. Observations: | 8760 | AIC: | 1.099e+05 |
| Df Residuals: | 8757 | BIC: | 1.099e+05 |
| Df Model: | 2 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 373.0959 | 5.956 | 62.641 | 0.000 | 361.420 | 384.771 |
| **diffuse radiation, tilt** | **1.2191** | 0.015 | 81.698 | 0.000 | 1.190 | 1.248 |
| **relative humidity** | **-4.5618** | 0.082 | -55.897 | 0.000 | -4.722 | -4.402 |

| | | | |
|---|---|---|---|
| Omnibus: | 995.412 | Durbin-Watson: | 0.174 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 1506.026 |
| Skew: | 0.837 | Prob(JB): | 0.00 |
| Kurtosis: | 4.150 | Cond. No. | 562. |

# Task III: Inference

OLS Regression Results

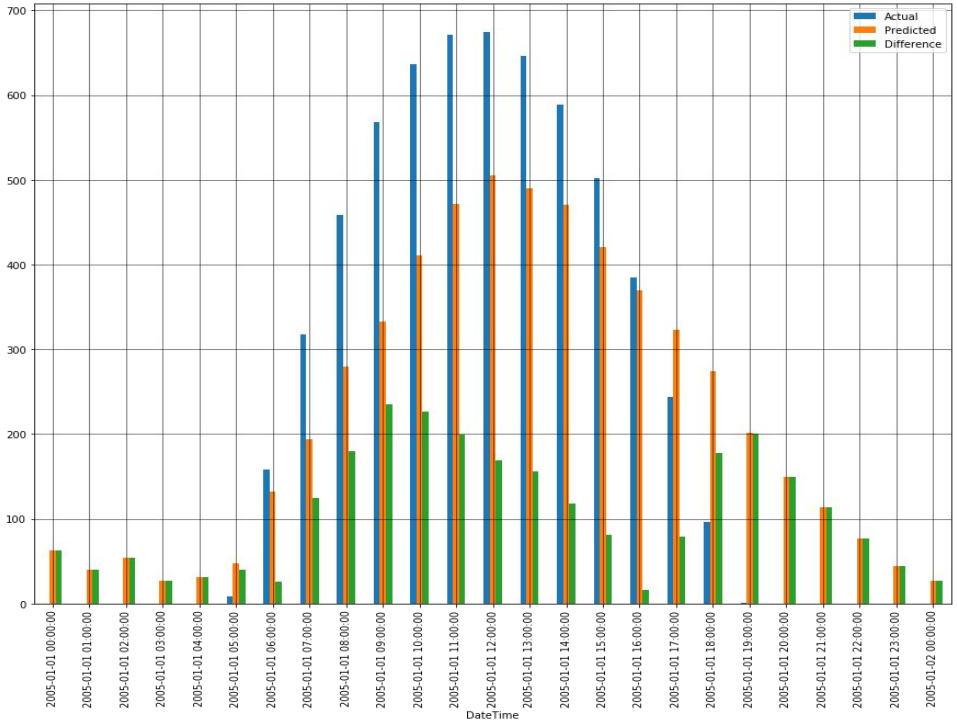
| | | | |
|---|---|---|---|
| Dep. Variable: | PV production | **R-squared:** | **0.670** |
| Model: | OLS | Adj. R-squared: | 0.670 |
| Method: | Least Squares | F-statistic: | 4446. |
| Date: | Wed, 01 Apr 2020 | Prob (F-statistic): | 0.00 |
| Time: | 04:33:19 | Log-Likelihood: | -54924. |
| No. Observations: | 8760 | AIC: | 1.099e+05 |
| Df Residuals: | 8755 | BIC: | 1.099e+05 |
| Df Model: | 4 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 407.9130 | 13.071 | 31.207 | 0.000 | 382.291 | 433.535 |
| **diffuse radiation, tilt** | **1.2268** | 0.015 | 81.238 | 0.000 | 1.197 | 1.256 |
| **relative humidity** | **-4.8223** | 0.115 | -41.852 | 0.000 | -5.048 | -4.596 |
| **temp** | **-1.3015** | 0.346 | -3.758 | 0.000 | -1.980 | -0.623 |
| **wind speed** | **1.0089** | 0.576 | 1.752 | 0.080 | -0.120 | 2.138 |

| | | | |
|---|---|---|---|
| Omnibus: | 978.524 | Durbin-Watson: | 0.177 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 1468.377 |
| Skew: | 0.830 | Prob(JB): | 0.00 |
| Kurtosis: | 4.127 | Cond. No. | 1.24e+03 |

# Task III: Inference

**Bar chart of real and predicted values of PV production using the optimal regression model (using 'diffuse radiation, tilt' and 'relative humidity' features)**



# Task III: Motivation

- **Scikit-learn library**

  The Scikit-learn library contains specialised machine learning tools which includes simple steps for fitting a linear regression model. We use this produce our multivariate model to predict PV production.

- **Heatmap** (**Pandas library)**

  The **heatmap** function in Pandas library provides a graphically visualised diagram of the correlation between each column which makes it easy to find patterns and analyse the data quickly.

- **Scatter-plot (Matplot library)**

  We used scatter plot function provided by ‘Matplotlib’ to show the line of best fit for each correlation.

- **Statsmodel library**

  This library provides functions for estimating many statistical models and statistical data exploration. It also includes a simple regression function and a summary report function which is nicely laid out, hence we use it for comparing the results of each feature.