project2020

January 8, 2021

1 Higher Diploma in Science in Computing (Data Analytics)

1.1 ### Programme Module: Fundamentals of Data Analysis (COMP07084) - Project 2020

In this project we had to perform and explain simple linear regression using Python on the `powerproduction` dataset available on Moodle.

The goal was to accurately predict wind turbine `power` output from wind `speed` values using the data set as a basis. Our submission had to be in the form of a git repository containing, at a minimum, the following items:

1. Jupyter notebook that performs simple linear regression on the data set.
2. In that notebook, an explanation of your regression and an analysis of its accuracy.
3. Standard items in a git repository such as a README.

To enhance our submission, we also has to consider comparing simple linear regression to other types of regression on this data set. Rest assured, all the above concepts will be explored in lecture videos and other materials in the coming semester.

Table of Contents

1.2 ## 1.0 Wind Energy

1.2.1 1.1 Introduction

Energy demand across the world is increasing rapidly because of population and economic growth especially in emerging market economies. This has meant the search for new energy resources has intensified across the globe. The supply of energy is a key element in a countries social and economic development so much so that it can impact international relations. Conflicts in Iraq/Syria, South Sudan, the Crimea/Ukraine, and the South China Sea show the desire to control valuable oil and gas assets is fuelling long-standing historic tensions. Michael Klare argues that *"in a fossil-fuel world, control over oil and gas reserves is an essential component of national power".* (Klare, 2014)

These factors combined with the impact of climate change mean the world will increasingly use renewable energy instead of fossil fuels in order to meet energy demand. Wind energy is seen as a positive alternative to fossil fuels but as with all renewables it has advantages as well as disadvantges. Although the manufacturer and installation of wind turbines does involve the release of some pollution into the environment, in the long run it is a source of clean energy and does not emit any greenhouse gases. It is renewable, space efficient, low cost and promotes job creation. However, it does have disadvantages, including its intermittent and unpredictable nature, noise issues, wildlife habitat dislocation as well as impacting tourism with some considering it an eye sore. It also faces location limitation issues, *"because to be economically viable, they need to be installed in a place where they will produce enough electricity, which means coastal areas, the tops of hills, and open planes - essentially anywhere with strong, reliable wind. Most of these suitable places tend to be in remote areas far outside of cities and towns, in more rural areas or offshore. Because of this distance, new infrastructure, such as power lines, have to be built in order to connect a wind farm to the power grid".* (Lane, 2020)

1.2.2 1.2 Facts and figures

In 2019 global direct primary energy consumption stood at 158,839 TWh with renewable energy sources (RES) accounting for 19,219 TWh or 12.1% of this figure. Within (RES) wind energy contributed 6.6% or 1,270 TWh of all renewables. In 2019 Europe had 205 GW of wind energy capacity accounting for 15% of the EU-28 consumed electricity. An additional 15.4 GW of new wind power capacity was added in 2019 an increase of 27% on the previous year.

Within Europe ten member states are achieving wind power shares above 10%. *"The highest share, and a new record, was set by Denmark where 47% of the electricity demand in 2019 was met by wind energy, followed by Ireland at 32% and Portugal at 27%"* (Windeurope.org, 2020).

In Ireland wind energy is the largest contributing resource of renewable power. According to the Sustainable Energy Authority of Ireland (SEAI) in their Wind Energy Roadmap, *"wind energy has the potential to generate enough electricity to exceed domestic demand by 2030 and for its wind market to become export driven in the 2020-2030 timeframe"* (SEAI, 2011). This could generate €15 billion in economic value and lead to the creation of 20,000 jobs in the installation, operation and maintenance of wind farms. It would also present a large carbon abatement in the range of 400 to 450 metric tonnes of CO2 by 2050.

In order to achieve these ambitious goals significant investment is required and it is estimated that the wind industry would hit a peak annual investment of between €6 billion and €12 billion by 2040 (SEAI, 2011). Caution should be applied however when assessing investment in wind energy. One report has identified a payback period of 23 years for an investment in a wind energy project, which would be a disappointing timeframe for investors. However, this does not have to be the case if a significant feasibility assessment is carried out.

According to a study carried out by TUD, if due consideration is given to *"site selection, electricity market conditions, the quality of the control system and the competencies of the design/installation/commissioning company"* (Kealy, 2015) there should be a positive outcome for investors, consumers and the environment. Their study based on a wind farm in the North East of Ireland were an average capacity factor of 34% was returned from the wind farm's turbines a payback period of 6.7 years was set. Because of the intermittent nature of wind, turbines typically produce only 20% - 40% of their maximum possible output over the course of a year which is known as the capacity factor (Miller, 2014).

1.2.3 1.3 Location

The crucial factor in the location of wind farms is calculating the annual energy production and how the energy it produces compares to alternative sources of energy. Key to this is access to and modelling of long-term data. Data should be collected from a potential site over a two to three-year timeframe. From this data the long-term annual variability needs to be calculated and can the renewable energy production output be predicted/forecast (Nelson, 2019). Modelling solutions at their most basic can be done through physical modelling via computational fluid dynamics or mathematical modelling via regular linear algebra.

Simple mathematical modelling on its own won't provide the accuracy but will suffice for common use cases like estimating maximum power production. This is the case with wind turbine manufacturers using power curve modelling techniques. The wind turbine power curve shows the relationship between wind speed and power generated *"for different wind speeds. A typical wind turbine power curve has three main characteristic speeds: 1) cut-in (Vc); 2) rated (Vr); and 3) cut-out (Vs) speeds. The turbine starts generating power when the wind speed reaches the cut-in value. The rated speed is the wind speed at which the generator is producing the machine's rated power. When the wind speed reaches the cut-out speed, the power generation is shut down to prevent defects and damages"* (Wadhvani, 2017).

Power curves can oversimplify reality though and can err by plus or minus 20% the actual power output. In actuality additional variables must be factored in (Miller, 2014). For example, *"wind speed at heights below and above the hub, wind shear, and turbulence are also strong predictors of power production"*. Any model used in predicting power output from wind turbines must also quantify uncertainty or confidence level associated with them. *"Such confidence levels are particularly of value to electric grid operators, who need both the predictions of output and the associated levels of confidence to determine an optimal schedule for turning various sources of power on and off. Quantifying output uncertainty is also crucial for siting wind farms"*(Miller, 2014). Outliers also need to be considered in any model, wind ramps being one of the most important. Because power output is proportional to the cube of the wind speed a wind ramp can result in a dramatic change in power production. *"Consequently, accurate wind ramp prediction is extremely important, leading some experts to refer to it as "the Holy Grail of wind forecasting."(Miller, 2014)*

1.2.4 1.4 Notes

Referencing: For references I have used the following style in this document. (Author, Year(Reference number)). For example (Miller, W., 2014, (3)). 3 in this case refers to number 3 in the References section.

1.3 2.0 Simple linear regression

For this project we had to perform a simple linear regression on a dataset relating to wind turbines with the goal of accurately predicting wind turbine power output from the wind speed values. Regression is a statistical technique which is used to investigate the relationship between variables. When we only have one input variable, in this case wind speed, it is a simple linear regression, when there is more than one input variable then a multiple linear regression would be used.

The equation for a simple linear regression is $y = w0 + w1 * x1$. In our case we know the input variable $x1$ is wind speed and y is power output.

$w0$ and $w1$ are the two coefficients, where $w0$ is the intercept (of the y-axis), and $w1$ is the slope of the line. $w1$ shows the impact of the independent variable $x1$ on y.

For example, when $w1 = 0$, there's no impact of $x1$ on y since $(0 * x1 = 0)$. (Linear Regression in Machine Learning, 2020).

Therefore simply, linear regression is an algorithm that finds the best values of w0 and w1 to fit the training dataset. (Linear Regression in Machine Learning, 2020).

1.4 3.0 Importing the required libraries

Numerical Python or NumPy is the first library we require. NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

[1]:
```
import numpy as np
```

Next import the pandas library. pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,built on top of the Python programming language.

[2]:
```
import pandas as pd
```

Finally we need to import Matplotlib a plotting library available for the Python programming language as a component of NumPy. Matplotlib embeds plots in Python applications. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. The `%matplotlib inline` statement will cause our matplotlib visualizations to embed themselves directly in our Jupyter Notebook, which makes them easier to access and interpret.

[3]:
```
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

[4]:
```
import seaborn as sns
```

1.5 4.0 Create dataframe

Now let's load the data from the`powerproduction` csv file provided using the pandas.read_csv function.

[5]:
```
dataset = pd.read_csv('powerproduction.csv', delimiter=',')
```

1.6 5.0 Data exploration

Having obtained the data, the next step is to perform some exploratory data analysis, looking in more detail at the dataset attributes.

1.6.1 5.1 DataFrame.info

Now lets take a quick look at the data types and if there are any null records in the dataset using the pandas.DataFrame.info function

[6]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   speed   500 non-null    float64
 1   power   500 non-null    float64
dtypes: float64(2)
memory usage: 7.9 KB
```

The dataset has only two columns, speed and power and both are floats. There is 500 records in the dataset and no null records.

1.6.2 5.2 DataFrame.head

First I'm going to have a look at the first 10 records of the dataset using the pandas.DataFrame.head function.

[7]:
```
dataset.head(10)
```

[7]:
```
   speed  power
0  0.000  0.000
1  0.125  0.000
2  0.150  0.000
3  0.225  0.000
4  0.275  0.000
5  0.325  4.331
6  0.400  5.186
7  0.450  3.826
8  0.501  1.048
9  0.526  5.553
```

From the results it's clear the wind turbine only begins to produce output after a certain wind speed is reached. This makes sense as the blades on wind turbines only begin to rotate once the cut-in speed is reached. The cut-in speed for most turbines is around 3-5 metres per second (m/s), or 8-12 miles per hour (mph). Around cut-in, the generator may be used as a motor to help the wind overcome inertia and start the blades turning. (Enerpower, 2020). Now lets examine the last 10 records in the dataset using the pandas.DataFrame.tail function.

1.6.3 5.3 DataFrame.tail

```
[8]: dataset.tail(10)
```

```
[8]:      speed  power
     490  24.499    0.0
     491  24.525    0.0
     492  24.575    0.0
     493  24.650    0.0
     494  24.750    0.0
     495  24.775    0.0
     496  24.850    0.0
     497  24.875    0.0
     498  24.950    0.0
     499  25.000    0.0
```

From the data displayed it appears no power is produced after a set wind speed is reached, this is called the cut-out speed which is around 24 m/s or 55 mph or greater. This is to prevent wear and tear. Since winds of this strength occur only for a handful of hours per year, very little energy is lost in high wind periods.(Enerpower, 2020)

1.7 6.0 Summary statistics

1.7.1 6.1 DataFrame.describe

DataFrame.describe produces descriptive statistics that include summaries of the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.

```
[9]: dataset.describe()
```

```
[9]:             speed       power
     count  500.000000  500.000000
     mean    12.590398   48.014584
     std      7.224991   41.614572
     min      0.000000    0.000000
     25%      6.324750    5.288000
     50%     12.550500   41.645500
     75%     18.775250   93.537000
     max     25.000000  113.556000
```

```
[10]: dataset.mean()
```

```
[10]: speed    12.590398
      power    48.014584
```

```
dtype: float64
```

[11]:
```
dataset.median()
```

[11]:
```
speed     12.5505
power     41.6455
dtype: float64
```

[12]:
```
dataset.var()
```

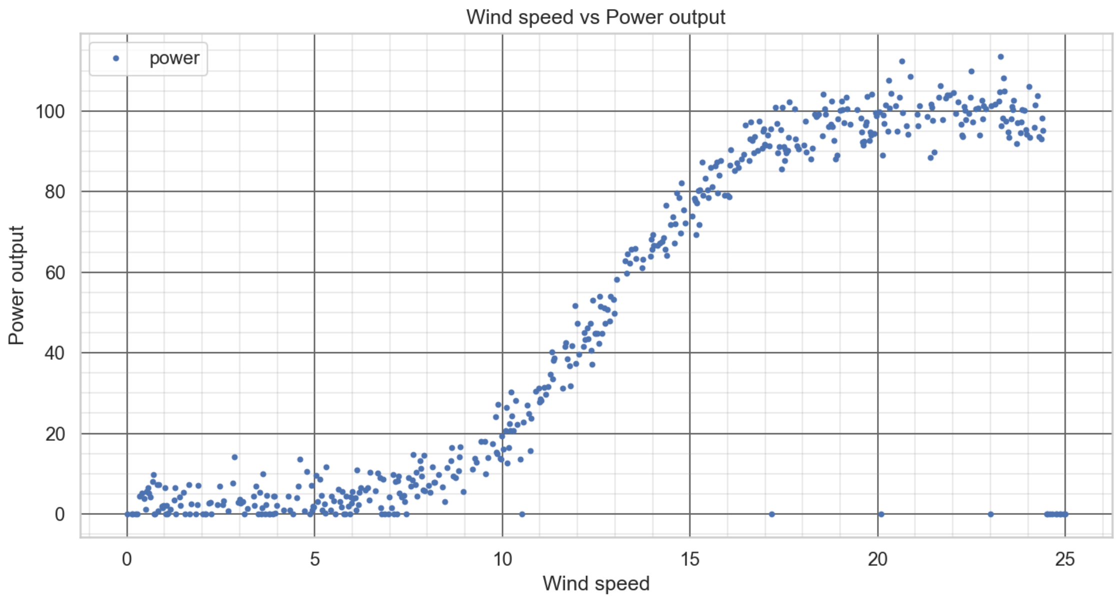
[12]:
```
speed       52.200499
power     1731.772627
dtype: float64
```

[13]:
```
dataset.std()
```

[13]:
```
speed      7.224991
power     41.614572
dtype: float64
```

Now lets plot our data to see what it looks like.

[14]:
```
sns.set(style='whitegrid', font_scale=1.1, rc={"figure.figsize": [12, 6]})
dataset.plot(x='speed', y='power', style=('o'), markersize=2.9)
plt.title('Wind speed vs Power output')
plt.xlabel('Wind speed')
plt.ylabel('Power output')
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2);
```



1.8 7.0 Simple linear regression algorithm

1.8.1 7.1 Preparing the data

In the previous sections we got to know the data a bit better, now it is time to split the data into "attributes" and "labels". (Robinson, S., 2020.,(15)).

Attributes and Labels: Attributes are the independent variables while labels are dependent variables whose values are to be predicted.(Robinson, S., 2020.,(15)).

As stated previously for this project, we were tasked with performing a simple linear regression on the wind turbines dataset with the goal of accurately predicting wind turbine power output from the wind speed values. Therefore our attribute set will consist of the "*speed*" column, and the label will be the "*output*" column. To extract the attributes and labels we need to execute the below script.

[15]:
```
# Taking the first column (speed) as attribute:
x = dataset.iloc[:, :-1].values

# Taking the second column (power) as label:
y = dataset.iloc[:, 1].values
```

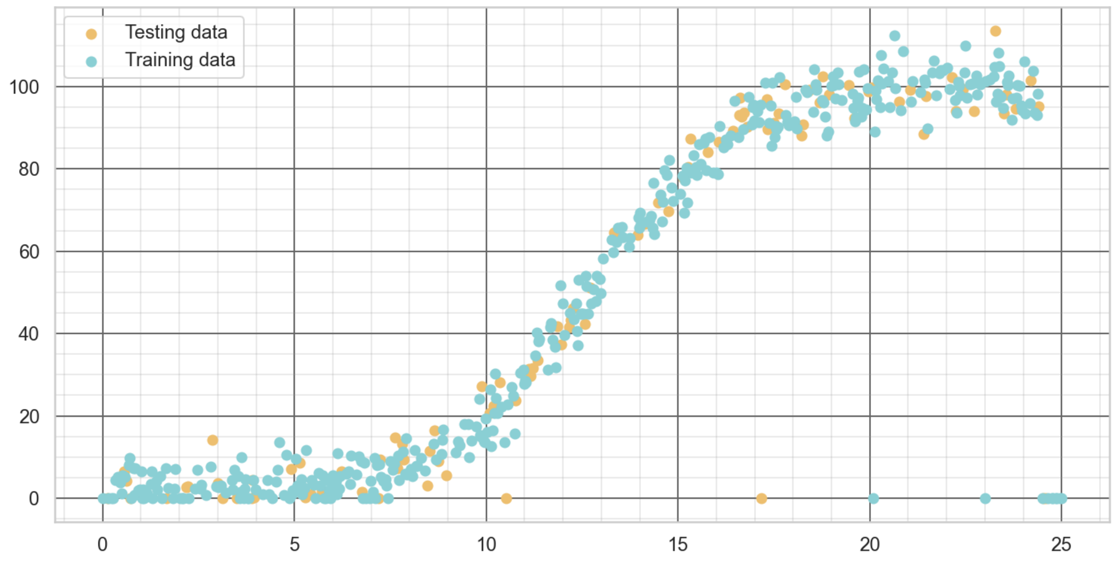
Now we will use Scikit-Learn's built in train_test_split() method to handle the "attributes" and "labels". The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

Train-test split procedure : Can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset. Train Dataset: Used to fit the machine learning model. Test Dataset: Used to evaluate the fit machine learning model.

[16]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=0)
```

The section of code above ascribes 80% of our data to the training set and 20% of the data to the test set. Lets visualise what this looks like.

[17]:
```
sns.set(style='whitegrid', font_scale=1.1, rc={"figure.figsize": [12, 6]})
plt.scatter(x_test, y_test, c='#edbf6f', label='Testing data')
plt.scatter(x_train, y_train, c='#8acfd4', label='Training data')
plt.legend(loc="upper left")
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2);
```



1.8.2 7.2 Training the algorithm

Linear regression is easy to implement using Scikit-Learn because all you need to do is import the `LinearRegression` class, instantiate it, and call the fit() method along with our training data. (Robinson, S., 2020, (15)).

[18]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

[18]:
```
LinearRegression()
```

A linear regression model basically finds the best value for the intercept and slope, which results in a line that best fits the data. To see the value of the intercept and slope calculated by the linear regression algorithm for our dataset, execute the following code.

To view the intercept we run the below code.

[19]:
```
print(regressor.intercept_)
```

```
-13.603433993820211
```

And to view the slope run the below.

[20]:
```
print(regressor.coef_)
```

```
[4.89542079]
```

The above figure tells us that for every one unit of change in wind speed the change in power output is approximately 4.89%.

1.8.3 7.3 Making predictions

The steps we have taken so far dealt with training our algorithm the goal now is to accurately predict wind turbine power output from the wind speed values. In order to this we will use the test data and see how accurately the algorithm predicts the power output %.

The section of code below creates a NumPy array containing all the predicted values for the input values in the `x_test` series.

[21]:
```
y_pred = regressor.predict(x_test)
```

Now we are going to compare the actual output values for `x_test` with the predicted values we will created a dataframe.

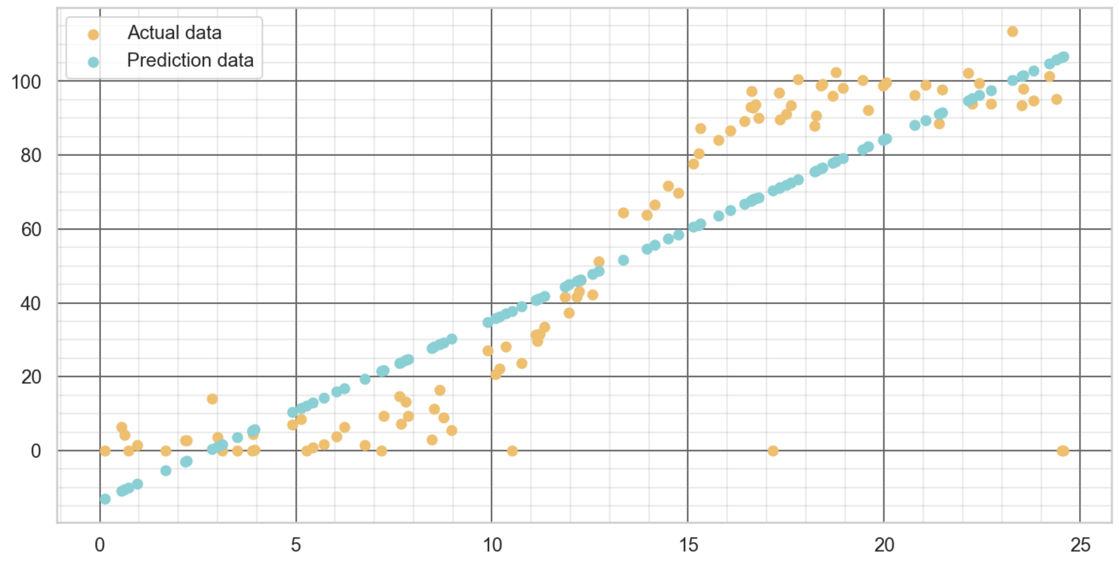
[22]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

[22]:
```
     Actual   Predicted
0     7.060   10.408605
1    51.149   48.632051
2    71.763   57.326318
3    99.357   96.161691
4   113.556  100.327694
..      …          …
95   96.058   77.911562
96    3.578    1.097515
97   93.931   95.304992
98    0.000    1.709442
99    0.000   37.852334

[100 rows x 2 columns]
```

Now lets visualize this model. Below is a scatter plot displaying the Actual(Test) data versus the Prediction data generated.

[23]:
```
sns.set(style='whitegrid', font_scale=1.1, rc={"figure.figsize": [12, 6]})
plt.scatter(x_test, y_test, c='#edbf6f', label='Actual data')
plt.scatter(x_test, y_pred, c='#8acfd4', label='Prediction data')
plt.legend(loc="upper left")
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2);
```



It’s clear from the above that there isn’t a linear relationship between speed and output. However let’s evaluate the results to see how accurate our predictions are.

1.8.4 7.4 Evaluating the algorithm

There are three main metrics for testing the performance of a regression machine learning model. They are 1. Mean Absolute Error, 2. Mean Squared Error and 3. Root Mean Squared Error. (McCullum, N., 2020.,(23)). I’ve outlined below what each one does before implementing them on our data.

Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. Its the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. (Medium. 2016.,(26))

Mean Squared Error (MSE): In statistics, the mean squared error (MSE)or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate.(En.wikipedia.org. 2021, (25))

Root Mean Squared Error (RMSE): RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It’s the square root of the average of squared differences between prediction and actual observation.(Medium. 2016.,(26))

So lets see how are algorithm did. Before doing this let’s get a quick understanding of what an algorithm is in machine learning.

Algorithm: A Machine Learning algorithm is the hypothesis set that is taken at the beginning before the training starts with real-world data. When we say Linear Regression algorithm, it means a set of functions that define similar characteristics as defined by Linear Regression and from those set of functions we will choose one function that fits the most by the training data.(Bhattacharjee, J., 2017., (32))

[24]:
```
from sklearn import metrics
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.
 ↪mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error (MAE): 15.371033053882327
Mean Squared Error (MSE): 496.3930965626669
Root Mean Squared Error (RMSE): 22.279880981788637
```

From the above you can see that the root mean squared error is 22.27 which is 46% of of the mean value of of the percentage of power output 48.01. This indicates that the algorithm did not do a very good job.

1.9 8.0 Polynomial regression algorithm

The polynomial regression process is similar to the linear regression carried out previously so I won't go into the same level of detail for each step as I've already explained it previously.

Polynomial Regression: is technically a type of Linear Regression. Although Polynomial Regression fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y|x) is linear in the unknown parameters that are estimated from the data. Therefore, Polynomial Regression is considered to be a special case of Multiple Linear Regression. (Haussmann, A., 2020., (27))

1.9.1 8.1 Preparing the data

As in the linear regression carried out earlier I'm going to read in the dataset using Pandas.

[25]:
```
dataset = pd.read_csv('powerproduction.csv', delimiter = ',')
```

[26]:
```
#Taking the first column (speed) as attribute:
x = dataset.iloc[:, :-1].values

#Taking the second column (power) as label:
y = dataset.iloc[:, 1].values
```

1.9.2 8.2 Training the algorithm

Now we will use Scikit-Learn's built in train_test_split() method to handle the "attributes" and "labels".

[27]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=0)
```

Next we need to import the PolynomialFeatures from Scikit-Learn.

Polynomial Features: are often created when we want to include the notion that there exists a nonlinear relationship between the features and the target.They are mostly used to add complexity to linear models with little features, or when we suspect the effect of one feature is dependent on another feature.(Van Dorpe, S., 2018., (31))

```
[28]: from sklearn.preprocessing import PolynomialFeatures
```

There are various methods for choosing the degree of the polynomial to use. But for this project I'm going use 5 as it seemed to give the best fit. In the end however it "boils down to ensuring you aren't under or over fitting the data.(Haussmann, A., 2020., (27))

```
[29]: #Creating a fifth order polynomial feature
      poly = PolynomialFeatures(degree=5)
```

```
[30]: #Converting our input linear dataset to the polynomial dataset
      x_poly = poly.fit_transform(x_train)
```

```
[31]: poly.fit(x_poly, y_train)
```

```
[31]: PolynomialFeatures(degree=5)
```

Next up is to train the model by creating an object `model`. Remember that Polynomial Regression is technically linear, so it falls under the same class and fit our transformed x values and y values to the model. (Haussmann, A., 2020., (27))

```
[32]: from sklearn.linear_model import LinearRegression
      model = LinearRegression()
      model.fit(x_poly, y_train)
```

```
[32]: LinearRegression()
```

1.9.3 8.3 Making predictions

```
[33]: y_pred = model.predict(poly.fit_transform(x_test))
```

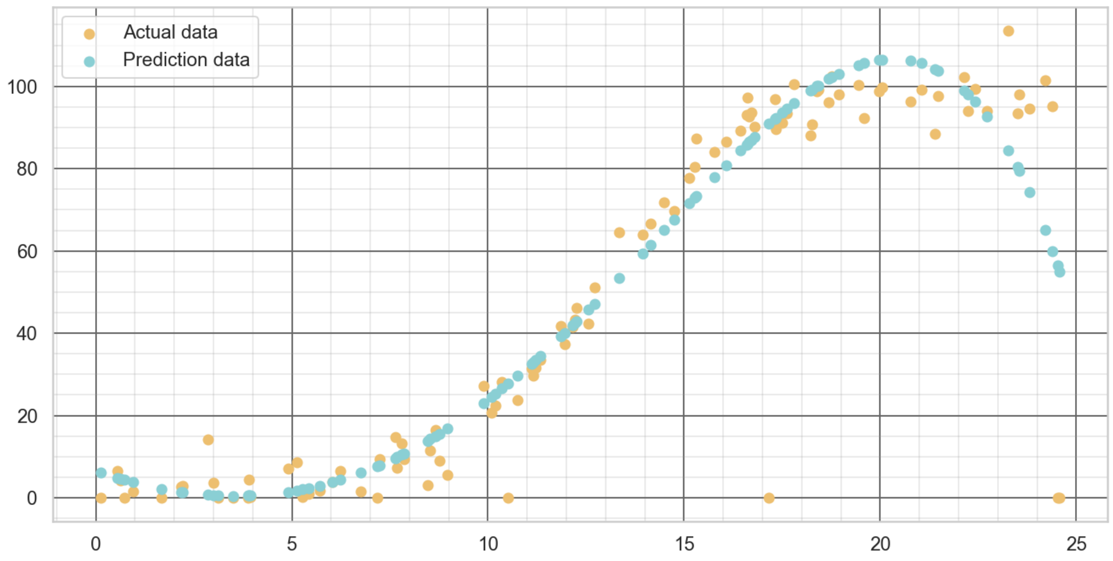
```
[34]: df2 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
      df2
```

```
[34]:      Actual   Predicted
      0     7.060    1.422121
      1    51.149   47.177977
      2    71.763   65.010088
      3    99.357   96.231341
      4   113.556   84.381448
      ..      …          …
      95   96.058  101.805353
      96    3.578    0.587268
```

```
97   93.931   98.050307
98    0.000    0.526946
99    0.000   27.706843

[100 rows x 2 columns]
```

[35]:
```
sns.set(style='whitegrid', font_scale=1.1, rc={"figure.figsize": [12, 6]})
plt.scatter(x_test, y_test, c='#edbf6f', label='Actual data')
plt.scatter(x_test, y_pred, c='#8acfd4', label='Prediction data')
plt.legend(loc="upper left")
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2);
```



1.9.4 8.4 Evaluating the algorithm

[36]:
```
from sklearn import metrics
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.
 ↪mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error (MAE): 7.849359425058364
Mean Squared Error (MSE): 224.94332416157653
Root Mean Squared Error (RMSE): 14.998110686402356
```

1.9.5 8.5 Comparison

Now lets see how the linear and polynomial regression compare.

Algorithm	Linear regression	Polynomial regression
Mean Absolute Error (MAE):	15.371033053882327	7.849359425058364
Mean Squared Error (MSE):	496.3930965626669	224.94332416157653
Root Mean Squared Error (RMSE):	22.279880981788637	14.998110686402356

As we can see in the table above we have significantly reduced the error. In the linear regression the root mean squared error was 22.27 which is 46% of of the mean value of of the percentage of power output 48.01 while in the polynomial regression the corresponding percentage is 31%. These are still big percentages which means another more accurate method would need to be used.

1.10 9.0 References

[1] Clifton, A., Kilcher, L., Lundquist, J. and Fleming, P., 2013. *Using Machine Learning To Predict Wind Turbine Power Output.* [online] ResearchGate. Available at: https://www.researchgate.net/publication/257748412_Using_machine_learning_to_predict_wind_turbine_power_output [Accessed 20 October 2020].

[2] Epaper.dk. 2020. *IEA Wind TCP - Annual Report 2019.* [online] Available at: https://www.epaper.dk/steppaper/iea/iea-wind-a-rsrapport-2019/ [Accessed 18 October 2020].

[3] Miller, W., 2014. *Predicting Wind Power With Greater Accuracy.* [online] Str.llnl.gov. Available at: https://str.llnl.gov/april-2014/miller [Accessed 18 October 2020].

[4] Windeurope.org. 2020. *Wind Energy In Europe In 2019.* [online] Available at: https://windeurope.org/wp-content/uploads/files/about-wind/statistics/WindEurope-Annual-Statistics-2019.pdf [Accessed 19 October 2020].

[5] SEAI. 2011. *Wind Energy Roadmap 2011-2050.* [online] Available at: https://www.seai.ie/publications/Wind_Energy_Roadmap_2011-2050.pdf [Accessed 18 October 2020].

[6] Sølverød, F., 2017. *Machine Learning For Wind Energy Prediction - Possible Improvements Over Traditional Methods.* [online] Duo.uio.no. Available at: https://www.duo.uio.no/bitstream/handle/10852/57735/Master_Thesis_Finn_Erik_20170525_FINAL.pdf?sequence=7&isAllowed=y [Accessed 19 October 2020].

[7] Wang, X., Guo, P. and Huang, X., 2011. *A Review Of Wind Power Forecasting Models.* [online] Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S1876610211019291 [Accessed 20 October 2020].

[8] Evans, R., 2019. *Simple Linear Regression - An Easy Introduction & Examples.* [online] Scribbr. Available at: https://www.scribbr.com/statistics/simple-linear-regression/ [Accessed 20 October 2020].

[9] Khamushkin, I., 2017. *Calculating Energy Production From Weather Forecast In Python.* [online] Medium. Available at: https://medium.com/planet-os/calculating-energy-production-from-weather-forecast-in-python-3c990047daa [Accessed 20 October 2020].

[10] Wadhvani, R., Shukla, S., Gyanchandani, M. and Rasool, A., 2017. *Analysis Of Statistical Techniques To Estimate Wind Turbine Power Generation.* [online] Paper.ijcsns.org. Available at: http://paper.ijcsns.org/07_book/201702/20170232.pdf [Accessed 21 October 2020].

[11] Kealy, T., Barrett, M. and Kearney, D., 2015. *How Profitable Are Wind Turbine Projects? An Empirical Analysis Of A 3.5 MW Wind Farm In Ireland Of A 3.5 MW Wind Farm In Ireland.* [online] Arrow.tudublin.ie. Available at: https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1101&context=engscheleart2 [Accessed 21 October 2020].

[12] Katabathun, N., Gundabathina, S. and Gummadi, D., 2020. *Prediciting Power Output Based On Weather Condition On Wind Turbines.* [online] Junikhyat.com. Available at: http://www.junikhyat.com/no_14_may_20/33.pdf?i=1 [Accessed 21 October 2020].

[13] Mester, T., 2018. *Pandas Tutorial 1: Pandas Basics (Read_Csv, Dataframe, Data Selection, Etc.).* [online] Data36. Available at: https://data36.com/pandas-tutorial-1-basics-reading-data-files-dataframes-data-selection/ [Accessed 21 October 2020].

[14] Klare, M., Dongre, S., James, M. and James, M., 2020. *Energy Wars: How Oil And Gas Are Fuelling Global Conflicts.* [online] Energy Post. Available at: https://energypost.eu/twenty-first-century-energy-wars-oil-gas-fuelling-global-conflicts/ [Accessed 24 October 2020].

[15] Robinson, S., 2020. *Linear Regression In Python With Scikit-Learn.* [online] Stack Abuse. Available at: https://stackabuse.com/linear-regression-in-python-with-scikit-learn/ [Accessed 21 October 2020].

[16] Nelson, V. and Starcher, K., 2019. *How To Select A Location For A Wind Farm.* [online] Routledge.com. Available at: <https://www.routledge.com/blog/article/how-to-select-a-location-for-a-wind farm? [Accessed 19 October 2020].

[17] Lane, C., 2020. *Wind Energy Pros And Cons.* [online] Solar Reviews. Available at: https://www.solarreviews.com/blog/wind-energy-pros-and-cons [Accessed 25 October 2020].

[18] Stojiljković, M., 2020. *Linear Regression In Python.* [online] Realpython.com. Available at: https://realpython.com/linear-regression-in-python/ [Accessed 21 October 2020].

[19] Chauhan, N., 2020. *A Beginner'S Guide To Linear Regression In Python With Scikit-Learn.* [online] KDnuggets. Available at: https://www.kdnuggets.com/2019/03/beginners-guide-linear-regression-python-scikit-learn.html [Accessed 20 October 2020].

[20] Just into Data. 2020. *Linear Regression In Machine Learning: Practical Python Tutorial.* [online] Available at: https://www.justintodata.com/linear-regression-machine-learning-python-tutorial/ [Accessed 26 October 2020].

[21] Statology. 2020. *How To Create A Scatterplot With A Regression Line In Python.* [online] Available at: https://www.statology.org/scatterplot-with-regression-line-python/ [Accessed 26 October 2020].

[22] Statology. 2020. *A Complete Guide To Linear Regression In Python.* [online] Available at: https://www.statology.org/linear-regression-python/ [Accessed 26 October 2020].

[23] McCullum, N., 2020. *Linear Regression In Python - A Step-By-Step Guide.* [online] Nickmccullum.com. Available at: https://nickmccullum.com/python-machine-learning/

linear-regression-python/ [Accessed 26 October 2020].

[24] Enerpower. 2020. *Wind Turbine F.A.Q.* [online] Available at: https://enerpower.ie/portfolio/wind-turbine-faq-ireland/ [Accessed 27 October 2020].

[25] En.wikipedia.org. 2021. *Mean Squared Error.* [online] Available at: https://en.wikipedia.org/wiki/Mean_squared_error [Accessed 4 January 2021].

[26] Medium. 2016. *MAE And RMSE — Which Metric Is Better?.* [online] Available at: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d [Accessed 4 January 2021].

[27] Haussmann, A., 2020. *Polynomial Regression: The Only Introduction You'Ll Need.* [online] Medium. Available at: https://towardsdatascience.com/polynomial-regression-the-only-introduction-youll-need-49a6fb2b86de [Accessed 1 January 2021].

[28] Jain, S., 2017. *Linear, Ridge And Lasso Regression Comprehensive Guide For Beginners.* [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/ [Accessed 5 January 2021].

[29] Tran, N., 2019. *Machine Learning: Polynomial Regression With Python.* [online] Medium. Available at: https://towardsdatascience.com/machine-learning-polynomial-regression-with-python-5328e4e8a386 [Accessed 4 January 2021].

[30] www.aionlinecourse.com. 2021. *Polynomial Regression In Two Minutes (With Python Code).* [online] Available at: https://www.aionlinecourse.com/tutorial/machine-learning/polynomial-regression [Accessed 4 January 2021].

[31] Van Dorpe, S., 2018. *Preprocessing With Sklearn: A Complete And Comprehensive Guide.* [online] Medium. Available at: https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide [Accessed 6 January 2021].

[32] Bhattacharjee, J., 2017. *Some Key Machine Learning Definitions.* [online] Medium. Available at: https://medium.com/technology-nineleaps/some-key-machine-learning-definitions-b524eb6cb48 [Accessed 8 January 2021].