DATATRAINED

|  |
| --- |
| GLOBAL POWERPLANT  TUSHAR KUMAR PATEL  AUGUST 2021 |

# abstract background

|  |  |
| --- | --- |
| GLOBAL POWERPLANT |  |



Overview

# Objective

# Introduction

# Data Analysis

# Variables

# Pre-Processing

# Missing values

# Encoding

# Handling Missing Values

# Exploratory Data Analysis

# Bar Chart

# Heat Map

# Outliers Detection

# Outliers Removal

# Splitting feature and Label

# Skewness

# Model Building

# Train Test and Split

# Evaluation Metrics

# Hyper Parameter Tuning

# Conclusion

|  |
| --- |
| The power plant doesn’t have energy, it generates it. – BRENDON BURCHARD  From the topic itself you must have realized till now that we are going for the analysis of Global powerplant dataset using scikit-Learns’ Decision Tree Classifier and Random Forest Classifier. First we need to install a bunch of package that would come in handy in the construction and execution of our code. Writing the following commands in Jupyter Notebook. |
| ObjectiveThe objective of this article is to predict flight prices given the various parameters. Data used in this article is publicly available. This will be regression problem since the target or dependent variable is the fuel and Capacity (MW).IntroductionPower generation is a complex process and understanding and predicting power output is an important element in managing a plant and its connection to the power grid. The Global Power Plant Database is a comprehensive, open source database of power plants around the world. It centralizes power plant data to make it easier to navigate, compare and draw insights for one’s own analysis. The database covers approximately 35,000 power plants from 167 countries and includes thermal plants (e.g. coal, gas, oil, nuclear, biomass, waste, geothermal) and renewables (e.g. hydro, wind, solar). Each power plant is geolocated and entries contain information on plant capacity, generation, ownership, and fuel type. It will be continuously updated as data becomes available.Data Analysis # Importing required libraries.  import pandas as pd  import numpy as np  import seaborn as sns  import matplotlib.pyplot as plt  %matplotlib inline  import warnings  warnings.filterwarnings('ignore')  Pandas will be used to work with file formats like csv, xlxs etc, Numpy will be used for mathematical calculations, seaborn will be used for making statistical graphics, matplotlib.pyplot is a collection of functions that make matplotlib works like MATLAB. Each pyplot function makes some changes to a figure, creates aplotting area in a figures, plots some lines in a plotting area, decorates the plot with labels, etc, warnings will ignore the warning that are generated during the execution of command.  The procedure of extraction information from given raw data is called data analysis. Variables  * Dependent variable   T `Fuel\_Type' and `Capacity\_MW'.   * Independent variable   Except Fuel\_type and Capacity\_MW all are independent variable from the  census data.   Pre-Processing +\*In[7]:\*+  # converting the commissioning year which is float into object.  df['commissioning\_year']=df['commissioning\_year'].astype(object)  df['year\_of\_capacity\_data']=df['year\_of\_capacity\_data'].astype(object)  +\*In[8]:\*+  # converting string into float  df['year\_of\_capacity\_data']=df['year\_of\_capacity\_data'].astype(object)  +\*In[9]:\*+  df['commissioning\_year']  +\*Out[9]:\*+  0 2011.0  1 NaN  2 NaN  3 2004.0  4 2015.0  ...  903 2016.0  904 NaN  905 NaN  906 NaN  907 NaN  Name: commissioning\_year, Length: 908, dtype: object---- Missing values +\*In[12]:\*+  # To check the missing values present in the dataset.  df.isnull().sum()    +\*In[13]:\*+  # checking the missing values from the heatmap  sns.heatmap(df.isnull())  +\*Out[13]:\*+ Encoding +\*In[18]:\*+  #converting data from label encoder  from sklearn.preprocessing import LabelEncoder  lab\_enc=LabelEncoder()  lab\_enc=lab\_enc.fit\_transform(df['primary\_fuel'])  df['primary\_fuel']=lab\_enc  +\*In[19]:\*+  #converting data from label enoder  from sklearn.preprocessing import LabelEncoder  lab\_enc=LabelEncoder()  lab\_enc=lab\_enc.fit\_transform(df['commissioning\_year'])  df['commissioning\_year']=lab\_enc  +\*In[20]:\*+  df   Handling Missing Values    Exploratory Data Analysis +\*In[27]:\*+  df['primary\_fuel'].value\_counts().plot.bar()  +\*Out[27]:\*+  ----<AxesSubplot:>    +\*In[28]:\*+  # It can be inferred from the plot that coal is the most common fuel then hydro then solar.  # Nuclear is the least common fuel.  ----  +\*In[29]:\*+  sns.stripplot(x='primary\_fuel',y='Years\_old',data=df)  plt.show()  ----  +\*Out[29]:\*+    +\*In[30]:\*+  # hydro power is more than 80 years old  # coal power plant is more than 50 years old.  # solar is the newest one among all.  +\*In[31]:\*+  sns.stripplot(x='primary\_fuel',y='capacity\_mw',data=df)  plt.show()  +\*Out[31]:\*+    +\*In[32]:\*+  # from above plot we can see that capacity of coal is highest then hydro then gas then hydoro  # the capacity of nuclear is high but they are low in number.  +\*In[33]:\*+  #lets see how plant age affects capacity\_mw  plt.style.use('ggplot')  sns.scatterplot(x = "Years\_old", y = "capacity\_mw", data = df)  plt.show()  +\*Out[33]:\*+    +\*In[34]:\*+  # the power generating capacity increased year by year as the demand for power increases.  +\*In[35]:\*+  #Lets have a look on relation between source and capacity\_mw  plt.figure(figsize = (10,5))  sns.barplot(x = "geolocation\_source", y = "capacity\_mw", data = df,palette='turbo\_r')  plt.show()  +\*Out[35]:\*+    +\*In[36]:\*+  # WRI is the maximum in number according to geolocation.  +\*In[37]:\*+  #lets see how geolocation\_source related with primary fuel  plt.figure(figsize = (10,5))  sns.countplot(x = "primary\_fuel", hue = "geolocation\_source", data = df,palette='turbo\_r')  plt.legend()  plt.legend(loc = 'upper right')  plt.show()  +\*Out[37]:\*+    +\*In[38]:\*+  # coal wind gas hydro biomass oil nuclear are of WRI category  # solar is of national renewable category  +\*In[39]:\*+  # plot between generation\_gwh\_2013 vs capacity\_mw  sns.scatterplot(x = 'generation\_gwh\_2013', y = 'capacity\_mw', data = df)  plt.show()  +\*Out[39]:\*+    +\*In[40]:\*+  # power generation growth is more than 5000, capacity\_mw is also above 1000MW.  +\*In[41]:\*+  # plot between generation\_gwh\_2014 vs capacity\_mw  sns.scatterplot(x = 'generation\_gwh\_2014', y = 'capacity\_mw', data = df)  plt.show()  +\*Out[41]:\*+    +\*In[42]:\*+  # generation\_gwh\_2014 is linearly related to capacity\_mw and maximum generation growth is below 30000.  +\*In[43]:\*+  # plot between generation\_gwh\_2015 vs capacity\_mw  sns.scatterplot(x = 'generation\_gwh\_2015', y = 'capacity\_mw', data = df)  plt.show()  +\*Out[43]:\*+    +\*In[44]:\*+  # generation\_gwh\_2015 is linearly related to capacity\_mw, and maximum generation growth is slightly more than 30000  +\*In[45]:\*+  # plot between generation\_gwh\_2016 vs capacity\_mw  sns.scatterplot(x = 'generation\_gwh\_2016', y = 'capacity\_mw', data = df)  plt.show()  +\*Out[45]:\*+    +\*In[46]:\*+  # generation\_gwh\_2016 is linearly related to capacity\_mw, and maximum generation growth is close to 30000  +\*In[47]:\*+  # plot between generation\_gwh\_2017 vs capacity\_mw  sns.scatterplot(x = 'generation\_gwh\_2017', y = 'capacity\_mw', data = df)  plt.show()  +\*Out[47]:\*+    +\*In[48]:\*+  # maximum capacity is from 0 to 1000  +\*In[49]:\*+  # plot between generation\_gwh\_2017 vs capacity\_mw  sns.scatterplot(x = 'longitude', y = 'capacity\_mw', data = df)  plt.show()  +\*Out[49]:\*+    +\*In[50]:\*+  # The powerplants are located between 65 to 95 and have capacity between 0 to 1000MW  +\*In[51]:\*+  df['primary\_fuel'].value\_counts()  +\*Out[51]:\*+  1 259  3 250  6 127  7 123  2 69  0 50  5 21  4 9  Name: primary\_fuel, dtype: int64----  +\*In[52]:\*+  # bar plot for primary\_fuel based on capaciy\_mw  sns.stripplot(x = 'primary\_fuel', y = 'capacity\_mw', data = df)  plt.show()  +\*Out[52]:\*+   Bar Chart +\*In[53]:\*+  plt.rcParams["figure.figsize"] = 12,10  df.hist();  +\*Out[53]:\*+    +\*In[54]:\*+  # from above plot we can see that outliers and skewness are also present in the dataset.  +\*In[56]:\*+  #ploting heatmap  import matplotlib.pyplot as plt  plt.figure(figsize=(15,7))  sns.heatmap(df.corr(), annot=True, linewidth=0.5,linecolor="black",fmt=".2f")  +\*Out[56]:\*+   Outliers Detection == Outliers detection and removal  +\*In[61]:\*+  df.boxplot(figsize=[20,15])  plt.subplots\_adjust(bottom=0.25)  plt.show()  +\*Out[61]:\*+   Outliers Removal +\*In[64]:\*+  from scipy import stats  +\*In[65]:\*+  #removing outliers using z score  z=np.abs(stats.zscore(df1))  z  +\*Out[65]:\*+  array([[5.49310543e-01, 1.14839610e+00, 8.80420380e-01, ...,  1.58055100e-16, 1.55659091e-16, 2.87082313e-01],  [3.84627594e-01, 5.86730280e-01, 5.94079309e-01, ...,    +\*In[66]:\*+  print('Row Numbers:',np.where(z>3)[0])  +\*Out[66]:\*+  Row Numbers: [ 15 15 15 15 144 144 144 144 144 144 159 159 180 184 210 210 210 210  210 210 240 245 309 309 309 309 309 309 324 334 341 343 361 364 365 365  +\*In[67]:\*+  #removing outliers some rows  index=(np.where(z>3)[0])  df=df.drop(df.index[index]) Splitting feature and Label +\*In[71]:\*+  X=df.drop(columns=['latitude','longitude','capacity\_mw'],axis=1)  y=df['capacity\_mw'] Skewness +\*In[74]:\*+  df1.skew()    +\*In[75]:\*+  # any value greater than -0.5 to +0.5 will be consider under skewness & any value between -0.5 to +0.5 is considered as normal value.  # to remove the skewness from the dataset. we use log transformation, sqrt transformation, cbrt transformation or use boxcox transformation.  # if there is any negative value in the dataset and we use the boxcox transformation then it will show an error.  # if there is any negative value in the dataset then use any of the transforamtion other than boxcox transformation.  # there is another thing  # we can use the power\_transform(df) to remove the skewness in all the columns dataset.  # always use power\_transform function in all the column  # note:- if we use power\_transform for 1,2,3,4 or 5 columns then we have to write a very long code.  +\*In[76]:\*+  from sklearn.preprocessing import power\_transform  df1=power\_transform(X)  df1=pd.DataFrame(df1,columns=X.columns)  +\*In[77]:\*+  df1.skew()    +\*In[78]:\*+  # It can be seen that skewness have been removed.  +\*In[79]:\*+   Model Building Model building is a very important step for completing the analysis. These four steps, involved for model building which is:   * *Building a baseline:* This is a model that is straightforward but with a good chance of providing decent results, through quick modeling. * *Designing the model*: This includes selecting a target variable and prediction type. * *Training the model*: This is done on a subset of the data to evaluate how well it is able to map inputs to outputs and make accurate predictions. * *Selecting the algorithm and*[*hyperparameters*](https://blog.dataiku.com/narrowing-the-search-which-hyperparameters-really-matter): Decide which algorithm to use for your model based on your business goals and priorities.    Train Test and Split The train-test split is a technique for evaluating the performance of a machine learning algorithm.  It 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.   The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.  This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.  The train-test procedure is appropriate when there is a sufficiently large dataset available.  I have used various model for the analysis but chosen the best model for the analysis of fuel. ****Random Forest Regression**** is a supervised learning algorithm that uses ****ensemble learning**** method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. Testing score for the analysis is approx. 80% Evaluation Metrics  Hyper Parameter Tuning  Conclusion Annual generation from hydropower plants can be predicted less accurately, and depends significantly on water runoff. Natural gas plants were the most difficult to predict annual generation for, highlighting how system factors, for which we have limited information, are important in determining when and how they are dispatched. The analysis found that plant-level annual generation for wind and solar can be estimated fairly precisely given information on how much wind blows and sun shines at the plant location. The low penetration of intermittent renewables until recently means that system constraints have been limited in practice: When wind and solar resources are available, they are generally dispatched. |



