Final phase

In this project analysis article we assess the power plant database to predict what was asked int the project.

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. This database covers The database covers approximately 14,000 power plants from 3 countries(USA, AUS, INDIA) and includes thermal plants (e.g. coal, gas, oil, nuclear, biomass, waste, geothermal) and renewables (e.g. hydro, wind, solar).

Our goals in this project is to make predictions about Primary fuels used in the power plants and the Capacity of power plants in megawatts. For this we use various methods. For fuel type aggregation, we define the fuel type attribute of our database, based on common fuel catagories. By analyzing data on the power plant attributes such as fuel type, capacity, and generation, machine learning models are trained to predict the primary fuel type for the power plants. This predictive capability can aid in assessing the evolving energy mix and informing investment decisions in the energy sector.

For the Prediction of Capacity, Machine learning algorithms are applied to predict the capacity of power plants. This predictive insight can assist in planning energy infrastructure, optimizing resource allocation, and mitigating risks associated with under or overcapacity.

Firstly we import relevant libraries to aid us in the working of the project, we imported numpy, pandas for creation and workings, seaborn and matplotlib for visual graphical analysis. Then we import warnings to deal with useless warning that our notebook platform gives and finally for the scientific processes, i.e. machine learning and importing models we import sklearn.

In this project we import three data files that were given to us, they were databases about the power plants of Australia, India and US. We used pandas to create dataframe.

In the Exploratory data analysis, we first calculated its shape where we found that there were 13298 rows and 27 columns from combining three data sets . We found out that there were two columns with the name of country and country\_long, They both were the same with country names and its short form. So we dropped both columns as they were of no use. We checked all the columns, the data types and found out that there were 14 object or string data types and 13 numerical int data types. We scanned for unique values off each column, using .nunique() function and found that there were thousands of unique values except 2 columns that were weep\_id and estimated\_generation. We found out there were no data so we dropped the columns. We then checked for duplicates and found out there were none. We then explored the data of our target variable which were primary\_fuel and capacity\_mw using graphical visuals. We dropped four more object data type columns which weren’t going to be of use in calculating target variable. After final cleaning and dropping useless columns we had 13298 row and 17 columns. After that we treated the null values using the traditional methods, we used .fillna() function and filled the nulls with mean, median and mode wherever it was reasonable for the normal data. For the missing values in input data we use KNNImputer from scikit-learn, which imputes missing values from the nearby 3 neighbours. After these workings, there were no missing values. We checked the heat map again and saw no null vales. We found out from the data that the mean was more than median in all columns because of the huge data implying that they are skewed to the right. There was huge difference between mean and standard deviation. We also found out that there was a huge difference between maximum and 75% percentile from which we understood that there were huge outliers present in most columns. We used appropriate methods to work on them.  
  
After that we separated the categorical columns i.e. object columns from the numerical columns. We did visualization of the data, we first used the univariate analysis plotting, and found that the Solar and gas fuels were the highly used primary fuels followed by hydro and wind and then waste and biomass. The lesser used fuels were Petcoke. We then searched for sources and found that the source of fuel from the “ U.S. Energy Information Administration” was the highest. We then checked the geolocation source and again found the U.S. Energy Information Administration to be the at top list.

Then we checked for bivariant plotting and confirmed that coal, hydro and gas fuels were used in the highest number. After that we plotted a scatterplot for the year 2013 and found from visual inspection that there was a linear relationship between the feature and the label, the electricity generation reported for the year 2013 was above 1000 MW and as the generation growth increased, the capacity of plants increased moderately. For the year 2014, the scatterplot again showed a linear relationship between the feature and the label, the electricity generation also was above 1000 MW for the year. For the year 2015 we noticed the linear relationship again between the feature and the label, the electricity generation reported was above 1000 MW and as the generation growth increases the capacities of the plants are also increasing moderately. This was seen the same for the years 2014, 2015 , as the generation increases so does the power plant capacities. For the year 2017, the capacity of the plant is increasing moderately with respect to the generation. Same was seen for the years 2018 and 2019. We then created Bar Plots to check the relation between primary fuel and capacity , we found that nuclear power plants had the most capacity followed by coal and petcoke. After that we also check for the other fuels and capacity. We then checked relation between geolocation sources and capacity, finding that the geolocation source of U.S Energy Information Administration was the maximum with a capacity more than 1000 MW.  
We then created Line plots and found that the power generation is increasing every year and so is the capacity of the plant simultaneously for the year 2015,2016,2017,2018. There is a positive linear relationship between the capacity and the electricity generation.

We then created Distribution plot for all the numeric columns to check their behaviour.  
then created Pair plots by using the function sns.pairplot(). We checked all the pairplots and found linear relationship between few variables.

After that we checked for outliers, and found that all of them had outliers present.   
So we worked on removing the outliers using Zscore method. Zscore removed all the outliers, and in the process we also calculated the data loss percentage by using the formula

((original\_rows **-** new\_rows) **/** original\_rows) **\*** 100

We found that there was a data loss of 9.96%.

Then we got to the encoding process,

We use label encoder from scikit-learn to encode all the categorical columns, this will convert categorical data into numerical labels. After that we check for skewness, and found high skewness in all the columns.

We checked for correlation, using heatmap and found that the target variable “capacity” is highly and positively correlated with the features columns. The target variable is negatively correlated with the features columns. The feature columns have no relation with the target variable, so we drop them.

After that we use standard scaler from scikit-learn as a pre processing technique used to transform features so that they have a mean of 0 and a standard deviation of 1. After that we use variance inflation factor(vif) from statsmodels to check for multicollinearity , with vif we quantify the severity of multicollinearity in an ordinary last squares regression analysis.

We observed that all the columns had VIF less than 10 except some columns, which we can ignore.

After all the pre-processing, we send the data for training and testing using train\_test\_split() function. We first use the RandomForestRegressor classifier and get the R-squared value of 0.71, mean absolute error was 33.74, mean squared error was 10990.46, Root mean squared error was calculated from 104.83, mean squared logarithmic error 0.67, explained variance score was 0.71, median absolute error was 2.02 and max error was found to be 1982.23, we also plot the graph of actual vs predicted too.

We then run it through the LinearRegression model and get the r-sqaured value of 0.47 , mean absolute error was 70.92, mean squared error was 21263.91, Root mean squared error was calculated to be 145.82 , mean absolute percentage error 6.07, explained variance score was 0.45, median absolute error was 44.44 and max error was found to be 2953.36

,we also plotted the graph.

We then run it through KNeigboursRegressor model, and get the r-sqaured value of 0.99, mean absolute error was 33.93, mean squared error was 12854.79, Root mean squared error was calculated to be 113.37 , mean absolute percentage error 5642995673179024, mean squared logarithmic error was 0.66 explained variance score was 0.67, median absolute error was 2.08 and max error was found to be 1950.30, we also created a graph with actual vs predicted values.

We then run it through DecisionTreeRegressor model, and get the r-sqaured value of 0.99, mean absolute error was 44.65, mean squared error was 21224.04, Root mean squared error was calculated to be 145.68, mean absolute percentage error 6349793411985258, mean squared logarithmic error was 0.73 explained variance score was 0.45, median absolute error was 1.05 and max error was found to be 2553, we also created a graph with actual vs predicted values.

We then run it through Ridge model and get the r-sqaured value of 0.47, mean absolute error was 70.92, mean squared error was 21263.93, Root mean squared error was calculated to be 145.82, mean absolute percentage error 6.07, explained variance score was 0.45, median absolute error was 44.48 and max error was found to be 2953.40.

We used Lasso after that and got the r2-score value of 0.45, mean absolute error was 70.53, mean squared error was 21222.145, Root mean squared error was calculated to be 145.67, R-squared was 0.47, mean absolute error 70.53,mean squared error was 21222.14, root mean squared error 145.67,mean absolute percentage error was 6.05, explained variance score was 0.45 , median absolute error was 45.06 and max error was found to be 2957.96.

And lastly we use AdaBoostRegressor which gave us r-squared value of 0.35, mean absolute error was 99.68, mean squared error was 28427.40, root mean squared error was 168.60, mean absolute percentage error was 1.75, mean squared logarithmic error was 5.16, explained variance score was 0.38, median absolute error was 25.56 and max error was determined to be 1952.69. We also created a scatter plot.

After carefully verifying we found that the DecisionTreeRegressor worked best, So we saved the model using joblib.dump() function.

After we found about the first problem of capacity and creating a model nows time to go for the second that was primary fuel.

We checked for correlation and then made a heatmap, after that we made a correlation bar plot to check the correlation between the label and feature using bar plot. We dropped certain columns like source and longitude because they did not contribute in building the model. Then we separated features and labels, We didn’t do standard scalarization in this case as the model might predict wrong. Then we checked for the accuracy for RandomForestClassifier, the test accuracy was 0.80. Then we used the Decision tree Classifier which gave us an accuracy score of 0.75. we plotted a confusion matrix of Decisiontreeclassifier. Extratreesclassifier gave us a test accuracy of 0.79, after that we plotted an ExtraTreesclassifier confusion matrix. Then we used the Gradient Boosting Classifier and got an accuracy score of 0.78. created a confusion matrix for it and at last used AdaBoostClassifier which gave us a test accuracy score of 0.61. We created a confusion matrix and at last found out that the GradientBoostingClassifier worked best. We ran a cross validation score which confirms the result. We saved the model as an object file using the joblib.dump function. We then called the model and gave it some random arrays and tested the model and confirmed that the model gave correct predictions.