Global Power Plant Database

Problem Statement

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.

Import Libraries

In [1]:
```
#Importing Required libraries
import pandas as pd
import numpy as np
import seaborn  as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics  import accuracy_score
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Data Collection

In [2]:
```
#show all the columns and rows
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)

#Loding Dataset

df=pd.read_csv("Global_Power_Plant.csv")
df.head()
```

Out[2]:

	country	country_long	name	gppd_idnr	capacity_mw	latitude	longitude	primary_fuel	ot
0	IND	India	ACME Solar Tower	WRI1020239	2.5	28.1839	73.2407	Solar	
1	IND	India	ADITYA CEMENT WORKS	WRI1019881	98.0	24.7663	74.6090	Coal	
2	IND	India	AES Saurashtra Windfarms	WRI1026669	39.2	21.9038	69.3732	Wind	
3	IND	India	AGARTALA GT	IND0000001	135.0	23.8712	91.3602	Gas	
4	IND	India	AKALTARA TPP	IND0000002	1800.0	21.9603	82.4091	Coal	

In [3]:
```
#check the how many rows and how many columns
df.shape
```

Out[3]:
```
(907, 27)
```

In [4]:
```
#checking datatypes infromation
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 907 entries, 0 to 906
Data columns (total 27 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   country                  907 non-null    object 
 1   country_long             907 non-null    object 
 2   name                     907 non-null    object 
 3   gppd_idnr                907 non-null    object 
 4   capacity_mw              907 non-null    float64
 5   latitude                 861 non-null    float64
 6   longitude                861 non-null    float64
 7   primary_fuel             907 non-null    object 
 8   other_fuel1              198 non-null    object 
 9   other_fuel2              1 non-null      object 
 10  other_fuel3              0 non-null      float64
 11  commissioning_year       527 non-null    float64
 12  owner                    342 non-null    object 
 13  source                   907 non-null    object 
 14  url                      907 non-null    object 
 15  geolocation_source       888 non-null    object 
 16  wepp_id                  0 non-null      float64
 17  year_of_capacity_data    519 non-null    float64
 18  generation_gwh_2013      0 non-null      float64
 19  generation_gwh_2014      398 non-null    float64
 20  generation_gwh_2015      422 non-null    float64
 21  generation_gwh_2016      434 non-null    float64
 22  generation_gwh_2017      440 non-null    float64
 23  generation_gwh_2018      448 non-null    float64
 24  generation_gwh_2019      0 non-null      float64
 25  generation_data_source   449 non-null    object 
 26  estimated_generation_gwh  0 non-null      float64
dtypes: float64(15), object(12)
memory usage: 191.4+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
country                      0
country_long                 0
name                         0
gppd_idnr                    0
capacity_mw                  0
latitude                    46
longitude                   46
primary_fuel                 0
other_fuel1                709
other_fuel2                906
other_fuel3                907
commissioning_year         380
owner                      565
source                       0
url                          0
geolocation_source          19
wepp_id                    907
year_of_capacity_data      388
generation_gwh_2013        907
generation_gwh_2014        509
generation_gwh_2015        485
generation_gwh_2016        473
generation_gwh_2017        467
generation_gwh_2018        459
generation_gwh_2019        907
generation_data_source     458
estimated_generation_gwh   907
dtype: int64
```

In [6]:
```
df.isnull().sum().sum()
```

Out[6]:
```
10445
```

In [7]:
```
#Here delete all missing values columns
df1=df.dropna(axis=1)
```

In [8]:
```
df1.isnull().sum()
```

Out[8]:
```
country         0
country_long    0
name            0
gppd_idnr       0
capacity_mw     0
primary_fuel    0
source          0
url             0
dtype: int64
```

Here found no missing values

In [9]:
```
df1.head()
```

Out[9]:

	country	country_long	name	gppd_idnr	capacity_mw	primary_fuel	source	
0	IND	India	ACME Solar Tower	WRI1020239	2.5	Solar	National Renewable Energy Laboratory	http://www.
1	IND	India	ADITYA CEMENT WORKS	WRI1019881	98.0	Coal	Ultratech Cement ltd	
2	IND	India	AES Saurashtra Windfarms	WRI1026669	39.2	Wind	CDM	http
3	IND	India	AGARTALA GT	IND0000001	135.0	Gas	Central Electricity Authority	
4	IND	India	AKALTARA TPP	IND0000002	1800.0	Coal	Central Electricity Authority	

In [10]:
```
df1.shape
```

Out[10]:
```
(907, 8)
```

Encoding the Categorical Data

In [11]:
```
#load the label Ecoder fuction
from sklearn.preprocessing import LabelEncoder
label_encode=LabelEncoder()
df1['country']=label_encode.fit_transform(df['country'])

df1['name']=label_encode.fit_transform(df['name'])
df1['gppd_idnr']=label_encode.fit_transform(df['gppd_idnr'])
df1['primary_fuel']=label_encode.fit_transform(df['primary_fuel'])
df1['source']=label_encode.fit_transform(df['source'])
df1['url']=label_encode.fit_transform(df['url'])
```

In [12]:
```
df1.head()
```

Out[12]:

	country	country_long	name	gppd_idnr	capacity_mw	primary_fuel	source	url
0	0	India	0	657	2.5	6	109	128
1	0	India	1	519	98.0	1	174	173
2	0	India	2	853	39.2	7	21	205
3	0	India	3	0	135.0	2	22	58
4	0	India	4	1	1800.0	1	22	58

In [13]:
```
df1.drop(columns='country_long',axis=1,inplace=True)
df1.drop(columns='country',axis=1,inplace=True)
```

In [14]:
```
df1.head()
```

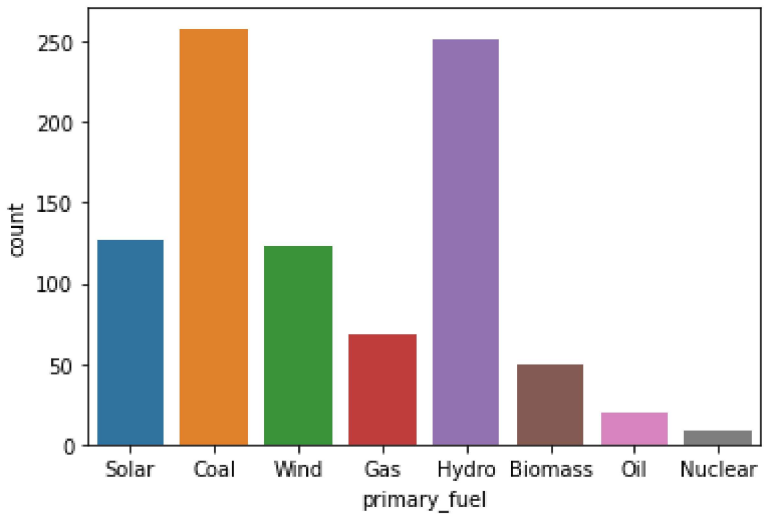
Out[14]:

	name	gppd_idnr	capacity_mw	primary_fuel	source	url
0	0	657	2.5	6	109	128
1	1	519	98.0	1	174	173
2	2	853	39.2	7	21	205
3	3	0	135.0	2	22	58
4	4	1	1800.0	1	22	58

Data Visualiztion

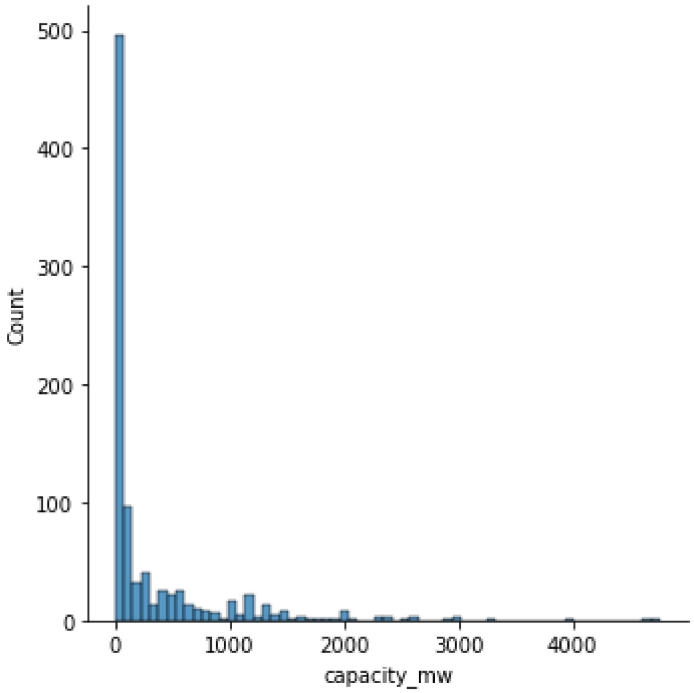
In [15]:
```
sns.countplot(x='primary_fuel',data=df)
print(df1["primary_fuel"].value_counts())
```

```
1    258
3    251
6    127
7    123
2     69
0     50
5     20
4      9
Name: primary_fuel, dtype: int64
```



In [16]:
```
sns.displot(x='capacity_mw',data=df)
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x1dfb37a0f70>
```



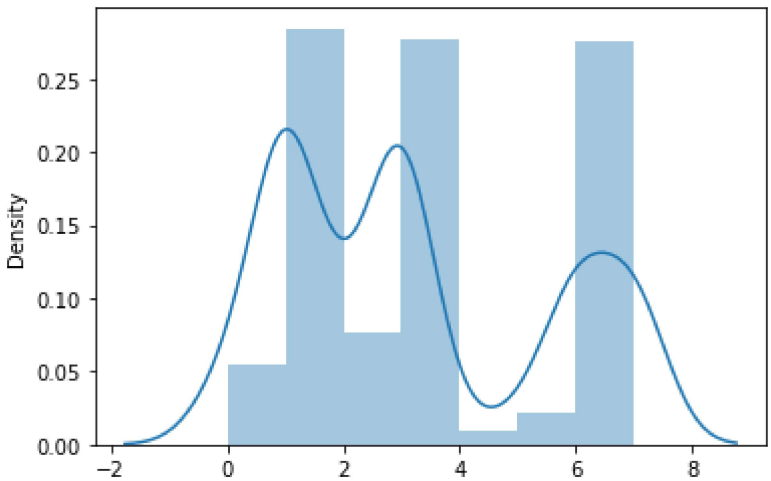
Distribution plot for all columns

In [17]:
```
#this is how we can get all the column names of the datafram
for column in df1:
    print(column)
```

```
name
gppd_idnr
capacity_mw
primary_fuel
source
url
```

In [18]:
```
 sns.distplot(x=df1.primary_fuel)
```

Out[18]:
```
<AxesSubplot:ylabel='Density'>
```



In [19]:
```
df1.describe()
```

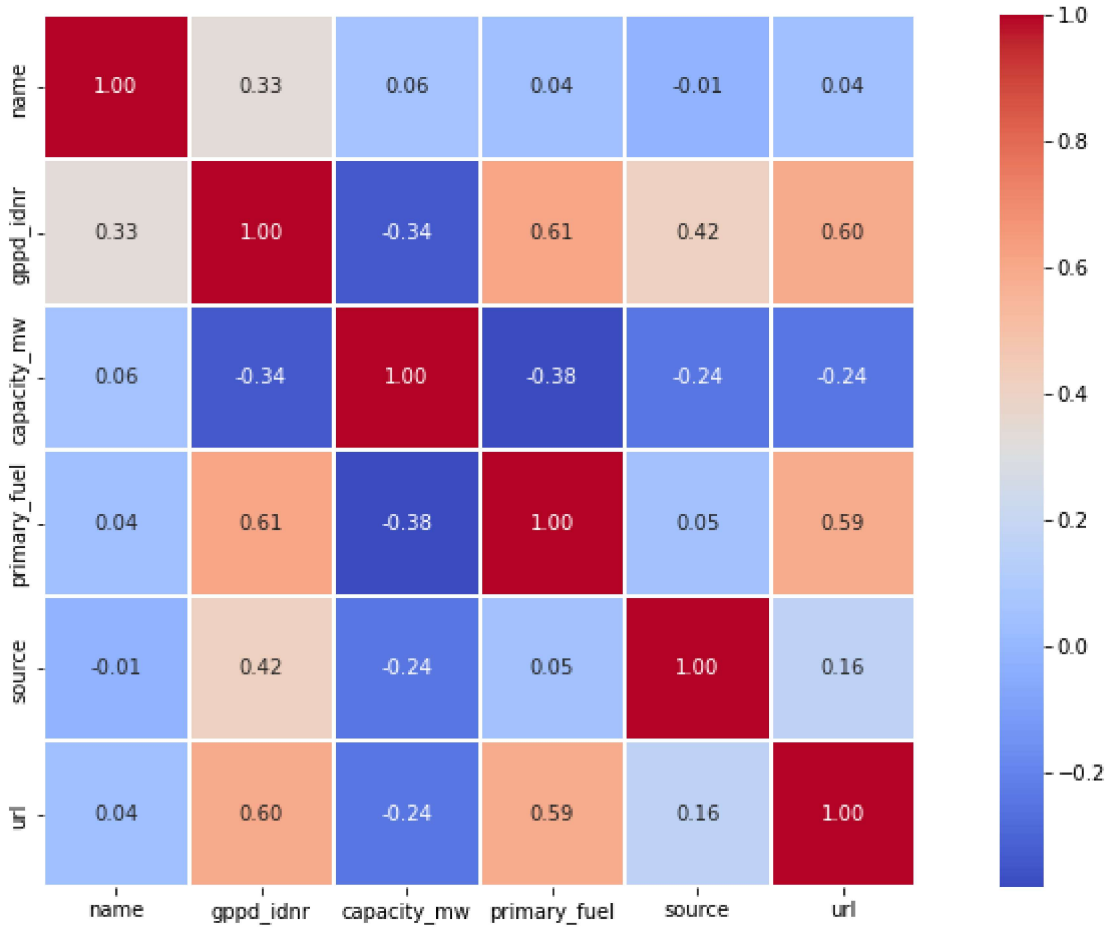
Out[19]:

	name	gppd_idnr	capacity_mw	primary_fuel	source	url
count	907.000000	907.000000	907.000000	907.000000	907.000000	907.000000
mean	453.000000	453.000000	326.223755	3.206174	43.847850	94.469680
std	261.972645	261.972645	590.085456	2.280652	44.642818	70.381222
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	226.500000	226.500000	16.725000	1.000000	22.000000	58.000000
50%	453.000000	453.000000	59.200000	3.000000	22.000000	58.000000
75%	679.500000	679.500000	385.250000	6.000000	29.500000	115.500000
max	906.000000	906.000000	4760.000000	7.000000	190.000000	303.000000

Correlation Matrix

In [20]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df1.corr(),cbar=True,square=True,fmt='.2f',linewidths=0.1,annot=True,annot
```

Out[20]:
```
<AxesSubplot:>
```



In [21]:
```
df1.head()
```

Out[21]:

	name	gppd_idnr	capacity_mw	primary_fuel	source	url
0	0	657	2.5	6	109	128
1	1	519	98.0	1	174	173
2	2	853	39.2	7	21	205
3	3	0	135.0	2	22	58
4	4	1	1800.0	1	22	58

Splitting features and Target

In [22]:
```
x = df1.drop(columns='primary_fuel', axis=1)
y = df1['primary_fuel']
```

In [23]:
```
print(x.shape)
print(y.shape)
```

```
(907, 5)
(907,)
```

Splitting the data into Training data & Testing Data

In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)

print('shape of x_train =', x_train.shape)
print('shape of y_train =', y_train.shape)
print('shape of x_test =', x_test.shape)
print('shape of y_test =', y_test.shape)
```

```
shape of x_train = (634, 5)
shape of y_train = (634,)
shape of x_test = (273, 5)
shape of y_test = (273,)
```

Model Training

In [25]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

Logistic Regression

In [26]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
predictions = lr.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[ 3  1  0  0  0  0  8  1]
 [ 2 55  0 16  0  0 15  0]
 [ 0  4  0 13  0  0  1  0]
 [ 0  9  0 59  0  0  1  0]
 [ 0  2  0  1  0  0  0  0]
 [ 1  0  0  3  0  0  1  1]
 [ 3  2  0  0  0  0 25  2]
 [ 0  0  0  0  0  0  0 44]]
              precision    recall  f1-score   support

           0       0.33      0.23      0.27        13
           1       0.75      0.62      0.68        88
           2       0.00      0.00      0.00        18
           3       0.64      0.86      0.73        69
           4       0.00      0.00      0.00         3
           5       0.00      0.00      0.00         6
           6       0.49      0.78      0.60        32
           7       0.92      1.00      0.96        44

    accuracy                           0.68       273
   macro avg       0.39      0.44      0.41       273
weighted avg       0.63      0.68      0.64       273

0.6813186813186813
```

Decision Tree

In [27]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
predictions = dt.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[ 8  2  1  0  0  2  0  0]
 [ 6 62  5  9  1  5  0  0]
 [ 0  9  4  4  1  0  0  0]
 [ 0  9  6 51  1  2  0  0]
 [ 0  3  0  0  0  0  0  0]
 [ 1  3  0  2  0  0  0  0]
 [ 0  0  0  0  3  0 28  1]
 [ 0  0  0  0  0  0  0 44]]
              precision    recall  f1-score   support

           0       0.53      0.62      0.57        13
           1       0.70      0.70      0.70        88
           2       0.25      0.22      0.24        18
           3       0.77      0.74      0.76        69
           4       0.00      0.00      0.00         3
           5       0.00      0.00      0.00         6
           6       1.00      0.88      0.93        32
           7       0.98      1.00      0.99        44

    accuracy                           0.72       273
   macro avg       0.53      0.52      0.52       273
weighted avg       0.74      0.72      0.73       273

0.7216117216117216
```

Random Forest Classifier

In [28]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
predictions = rf.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[11  2  0  0  0  0  0  0]
 [ 8 69  3  7  0  1  0  0]
 [ 0  8  4  6  0  0  0  0]
 [ 0  5  3 61  0  0  0  0]
 [ 0  1  0  2  0  0  0  0]
 [ 0  2  1  3  0  0  0  0]
 [ 0  0  0  0  0  0 31  1]
 [ 0  0  0  0  0  0  0 44]]
              precision    recall  f1-score   support

           0       0.58      0.85      0.69        13
           1       0.79      0.78      0.79        88
           2       0.36      0.22      0.28        18
           3       0.77      0.88      0.82        69
           4       0.00      0.00      0.00         3
           5       0.00      0.00      0.00         6
           6       1.00      0.97      0.98        32
           7       0.98      1.00      0.99        44

    accuracy                           0.81       273
   macro avg       0.56      0.59      0.57       273
weighted avg       0.78      0.81      0.79       273

0.8058608058608059
```

Support Vector Machine

In [29]:
```
from sklearn import svm
from sklearn.svm import SVC
svc=SVC(kernel='linear',random_state=0)
svc.fit(x_train,y_train)
predictions = svc.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[11  2  0  0  0  0  0  0]
 [ 9 64  0 15  0  0  0  0]
 [ 0  4  0 14  0  0  0  0]
 [ 0  9  0 60  0  0  0  0]
 [ 0  2  0  1  0  0  0  0]
 [ 1  2  0  3  0  0  0  0]
 [ 0  0  0  0  0  0 30  2]
 [ 0  0  0  0  0  0  0 44]]
              precision    recall  f1-score   support

           0       0.52      0.85      0.65        13
           1       0.77      0.73      0.75        88
           2       0.00      0.00      0.00        18
           3       0.65      0.87      0.74        69
           4       0.00      0.00      0.00         3
           5       0.00      0.00      0.00         6
           6       1.00      0.94      0.97        32
           7       0.96      1.00      0.98        44

    accuracy                           0.77       273
   macro avg       0.49      0.55      0.51       273
weighted avg       0.71      0.77      0.73       273

0.7655677655677655
```

In [ ]: