Predicting the power generation in a solar plant

Yaswanth Kottana 110120056; Tom Joseph 110120112; Harshmeet Singh Saluja 110120040

The data set is used to predict the power generation of a solar plant according to the irradiating from the Sum, ambient temperature of the atmosphere around and the module temperature.

Github link for the code: https://github.com/TomJosephKavalam/CSOE18Proj

Visualising Data

The follwing code uses seaborn to visualise the dataset and how the features vary with each other.

In [29]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
```

In [30]:

```
df=pd.read_csv('Dataset/P1.csv')
df
```

Out[30]:

	Unnamed: 0	DC_POWER	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	0	0.0	23.128673	20.464305	0.0
1	1	0.0	23.032562	20.341429	0.0
2	2	0.0	22.967493	20.269493	0.0
3	3	0.0	22.810594	20.198918	0.0
4	4	0.0	22.611436	20.085866	0.0
...	...	...	...	...	...
3152	3152	0.0	23.670292	21.691071	0.0
3153	3153	0.0	23.795434	22.067778	0.0
3154	3154	0.0	23.727901	21.662972	0.0
3155	3155	0.0	23.497284	21.051402	0.0
3156	3156	0.0	23.244698	20.774560	0.0

3157 rows × 5 columns

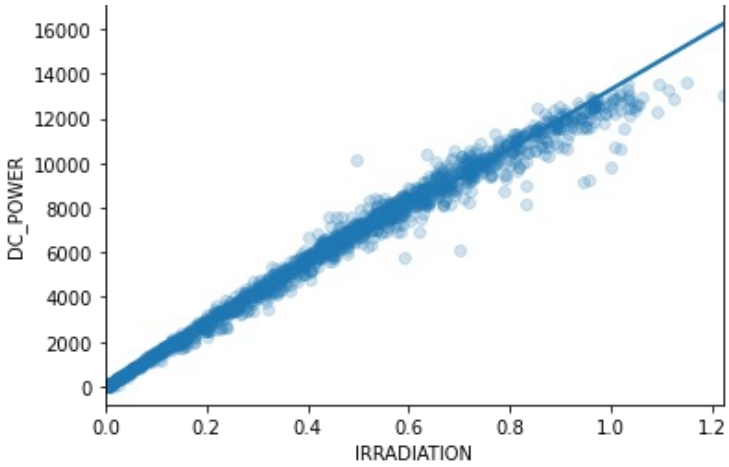
Visualising the trend between IRRADIATION and DC_POWER using scatter plot

In [31]:

```
sn.regplot(x = "IRRADIATION", y="DC_POWER", data=df, fit_reg = True, scatter_kws={"alpha
": 0.2})
```

Out[31]:

```
<AxesSubplot:xlabel='IRRADIATION', ylabel='DC_POWER'>
```



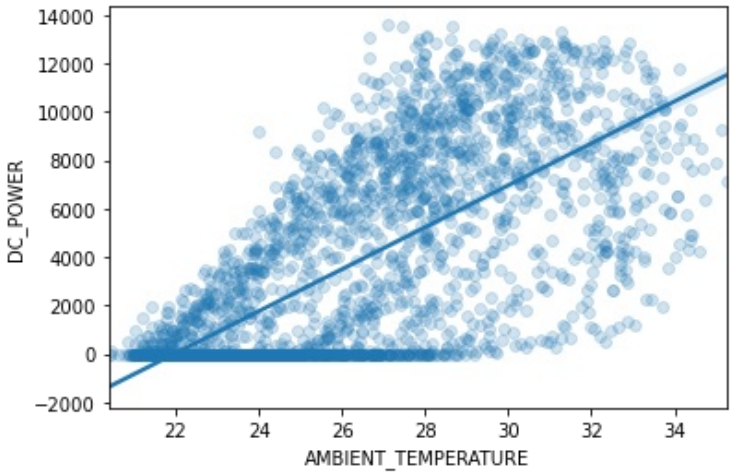
Visualising the trend between AMBIENT_TEMPERATURE and DC_POWER using scatter plot

In [32]:

```
sn.regplot(x = "AMBIENT_TEMPERATURE", y="DC_POWER", data=df, fit_reg = True, scatter_kws
={"alpha": 0.2})
```

Out[32]:

```
<AxesSubplot:xlabel='AMBIENT_TEMPERATURE', ylabel='DC_POWER'>
```



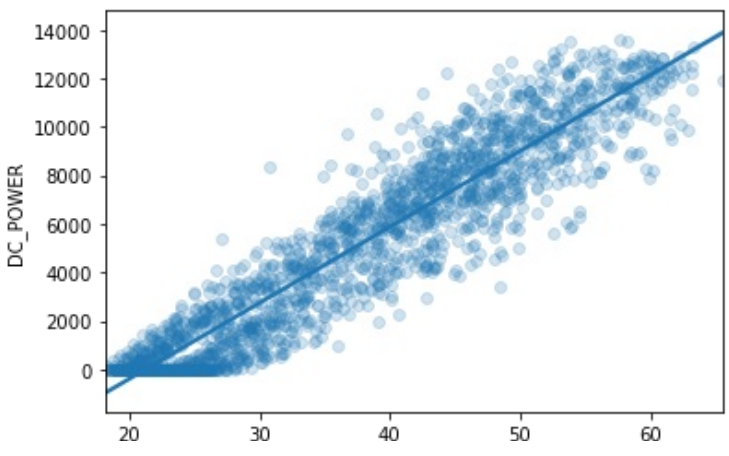
Visualising the trend between MODULE_TEMPERATURE and DC_POWER using scatter plot

In [33]:

```
sn.regplot(x = "MODULE_TEMPERATURE", y="DC_POWER", data=df, fit_reg = True, scatter_kws=
{"alpha": 0.2})
```

Out[33]:

```
<AxesSubplot:xlabel='MODULE_TEMPERATURE', ylabel='DC_POWER'>
```

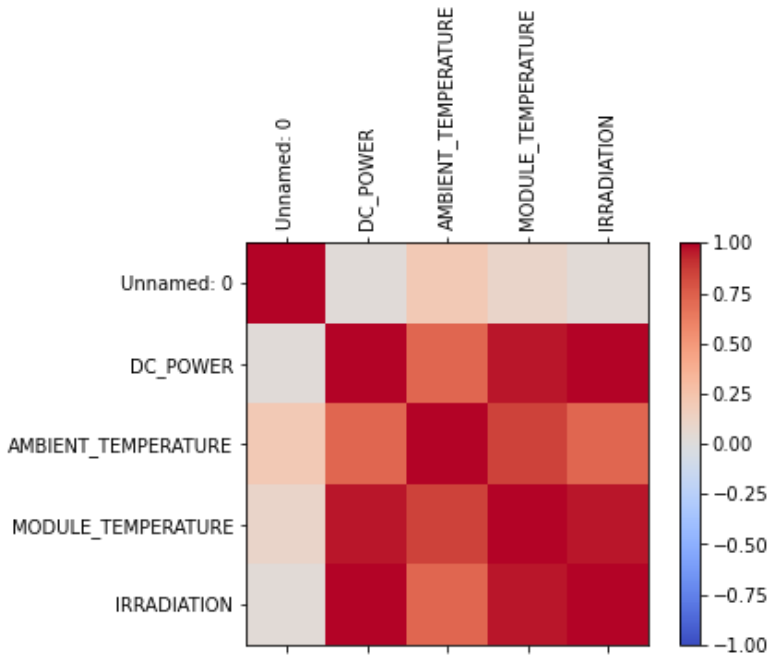


MODULE_TEMPERATURE

Visualising the correlation between the four variables (consider the column 'Unamed: 0' null as it is used for indexing the dataset).

In [34]:

```
columns=[]
corr = df.corr()
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(df.columns),1)
ax.set_xticks(ticks)
plt.xticks(rotation=90)
ax.set_yticks(ticks)
ax.set_xticklabels(df.columns)
ax.set_yticklabels(df.columns)
plt.show()
```



Trainig Code

The model being used is 'Linear Regression'. Basically, multivariate regression by training on multiple independent variables using linear regression. The DC_POWER was predicted using the features: AMBIENT_TEMPERATURE, MODULE_TEMPERATURE, IRRADIATION.
The data was split into one-third testing dataset and the rest training dataset. After predicting using the model, the R2 score showed up to be: 0.9921417157083059, and bias and variance as 122386.029 and 439.856 respectively.

In [83]:

```
import pandas as pd
import numpy as np
```

In [84]:

```
df=pd.read_csv('Dataset/P1.csv')
df=df.drop('Unnamed: 0',axis=1)
```

In [85]:

```
df
```

Out[85]:

	DC_POWER	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	0.0	23.128673	20.464305	0.0
1	0.0	23.032562	20.341429	0.0
2	0.0	22.967493	20.269493	0.0
3	0.0	22.810594	20.198918	0.0
4	0.0	22.611436	20.085866	0.0
...	...	...	...	...
3152	0.0	23.670292	21.691071	0.0
3153	0.0	23.795434	22.067778	0.0
3154	0.0	23.727901	21.662972	0.0
3155	0.0	23.497284	21.051402	0.0
3156	0.0	23.244698	20.774560	0.0

3157 rows × 4 columns

Setting up target variable and independent variable

In [86]:

```
X=df[['AMBIENT_TEMPERATURE','MODULE_TEMPERATURE', 'IRRADIATION']]
y=df['DC_POWER']
```

In [87]:

```
from sklearn.model_selection import train_test_split
```

Splitting data into training, testing data

In [88]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
3, random_state=42)
```

In [89]:

```
from sklearn.linear_model import LinearRegression
```

Fitting data using linear regression model

In [90]:

```
reg=LinearRegression()
reg.fit(X_train, y_train)
```

Out[90]:

```
LinearRegression()
```

Predicting using the model

In [91]:

```
y_pred=reg.predict(X_test)
y_pred
```

Out[91]:

```
array([  62.32466286,   45.22896531,   52.58443125, ...,   83.11129224,
        118.26822688, 4796.07909294])
```

In [97]:

```
from sklearn.metrics import r2_score, confusion_matrix, accuracy_score
from sklearn.metrics import mean_squared_error
from mlxtend.evaluate import bias_variance_decomp
```

Calculating mean square error, bias and variance

In [99]:

```
mse, bias, var = bias_variance_decomp(reg, X_train.values, y_train.valu
es, X_test.values, y_test.values, loss='mse', num_rounds=200, random_se
ed=1)
print('MSE: %.3f' % mse)
print('Bias: %.3f' % bias)
print('Variance: %.3f' % var)
```

```
MSE: 122825.884
Bias: 122386.029
Variance: 439.856
```

Calculating R2 score and root mean square error

In [93]:

In [93]:

```
score=r2_score(y_test,y_pred)
print('r2 socre is ',score)
print('root_mean_squared error of is==',np.sqrt(mean_squared_error(y_tes
t,y_pred)))
```

```
r2 socre is  0.9921417157083059
mean_sqrd_error is== 122378.64598680226
root_mean_squared error of is== 349.82659416745645
```

In [ ]:

```

```

Code to clean

The following data was used to make a perfect dataset out of the raw dataset by cleaning, merging, joining and dropping unnecessary columns.

In [57]:

```
import pandas as pd
import numpy as np
```

In [58]:

```
plant1_gen=pd.read_csv('Dataset/P1G.csv')
plant1_weat=pd.read_csv('Dataset/P1W.csv')
```

In [59]:

```
plant1_gen = plant1_gen.groupby('DATE_TIME').agg({'DC_POWER':'mean','AC_POWER':'mean', '
DAILY_YIELD':'mean','TOTAL_YIELD':'mean'})
plant1_weat=plant1_weat.set_index('DATE_TIME', drop=True)
```

In [60]:

```
plant1_gen
```

Out[60]:

DATE_TIME	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
01-06-2020 00:00	0.0	0.0	245.784091	6.978158e+06
01-06-2020 00:15	0.0	0.0	0.000000	6.978158e+06
01-06-2020 00:30	0.0	0.0	0.000000	6.978158e+06
01-06-2020 00:45	0.0	0.0	0.000000	6.978158e+06
01-06-2020 01:00	0.0	0.0	0.000000	6.978158e+06
...	...	...	...	...
31-05-2020 22:45	0.0	0.0	5695.045455	6.978158e+06
31-05-2020 23:00	0.0	0.0	5695.045455	6.978158e+06
31-05-2020 23:15	0.0	0.0	5695.045455	6.978158e+06
31-05-2020 23:30	0.0	0.0	5695.045455	6.978158e+06
31-05-2020 23:45	0.0	0.0	5169.870130	6.978158e+06

3158 rows × 4 columns

In [61]:

```
plant1_weat
```

Out[61]:

DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
15-05-2020 00:00	4135001	HmiyD2TTLFNqkNe	25.184316	22.857507	0.0
15-05-2020 00:15	4135001	HmiyD2TTLFNqkNe	25.084589	22.761668	0.0
15-05-2020 00:30	4135001	HmiyD2TTLFNqkNe	24.935753	22.592306	0.0
15-05-2020 00:45	4135001	HmiyD2TTLFNqkNe	24.846130	22.360852	0.0
15-05-2020 01:00	4135001	HmiyD2TTLFNqkNe	24.621525	22.165423	0.0
...	...	...	...	...	...
17-06-2020 22:45	4135001	HmiyD2TTLFNqkNe	22.150570	21.480377	0.0
17-06-2020 23:00	4135001	HmiyD2TTLFNqkNe	22.129816	21.389024	0.0
17-06-2020 23:15	4135001	HmiyD2TTLFNqkNe	22.008275	20.709211	0.0
17-06-2020 23:30	4135001	HmiyD2TTLFNqkNe	21.969495	20.734963	0.0
17-06-2020 23:45	4135001	HmiyD2TTLFNqkNe	21.909288	20.427972	0.0

3182 rows × 5 columns

In [62]:

```
plant1=pd.merge(plant1_gen, plant1_weat, how='inner', left_index=True, right_index=True)
df=plant1
df=df.reset_index(drop=False, inplace=False)
```

In [63]:

```
df=df.drop(labels=['SOURCE_KEY','PLANT_ID','TOTAL_YIELD', 'DAILY_YIELD', 'AC_POWER','DAT
E_TIME'], axis=1)
```

In [64]:

```
df.to_csv('Dataset/P1.csv')
df
```

Out[64]:

	DC_POWER	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	0.0	23.128673	20.464305	0.0
1	0.0	23.032562	20.341429	0.0
2	0.0	22.967493	20.269493	0.0
3	0.0	22.810594	20.198918	0.0
4	0.0	22.611436	20.085866	0.0
...	...	...	...	...
3152	0.0	23.670292	21.691071	0.0
3153	0.0	23.795434	22.067778	0.0
3154	0.0	23.727901	21.662972	0.0
3155	0.0	23.497284	21.051402	0.0
3156	0.0	23.244698	20.774560	0.0

3157 rows × 4 columns

In [65]:

```
df
```

Out[65]:

	DC_POWER	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	0.0	23.128673	20.464305	0.0
1	0.0	23.032562	20.341429	0.0
2	0.0	22.967493	20.269493	0.0
3	0.0	22.810594	20.198918	0.0
4	0.0	22.611436	20.085866	0.0
...	...	...	...	...
3152	0.0	23.670292	21.691071	0.0
3153	0.0	23.795434	22.067778	0.0
3154	0.0	23.727901	21.662972	0.0
3155	0.0	23.497284	21.051402	0.0
3156	0.0	23.244698	20.774560	0.0

3157 rows × 4 columns

In [ ]:

```

```