Climate Connect

Solar Power Generation from the plant.

Objectives: To Predict the Power Generation from the Forecast Data.

Introduction: Design a profitable solar power plant by analyzing the radiation and metrological data. You will benefit in the operation by reduced maintenance costs and high efficiency.

The sun radiation on earth surface combines **Direct normal irradiation (DNI)** and **Diffuse Horizontal Irradiation (DHI)** .Both are linked in the formula for **Global Horizontal Irradiation (GHI).**

GHI +DHI+DNI-$\cos(\boldsymbol{\theta})$

GTI is total amount of direct and diffuse irradiation received from above by a tilted surface.

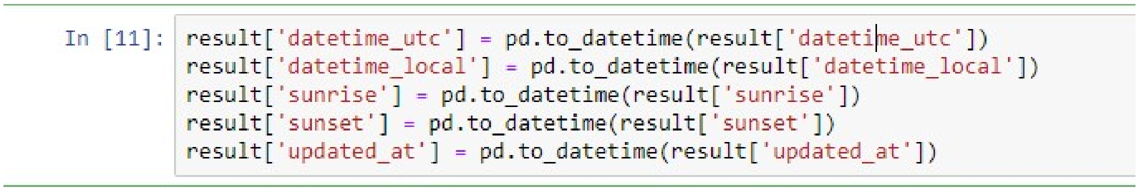
From the Given Data **GHI** in known to you and how much Power is generated with help of GHI.

From the Research the Parameter that influence the Solar Energy Production **Air-temperature, wind-speed, wind-direction, humidity etc.** details which contains in Weather Actuals file.

Approach:

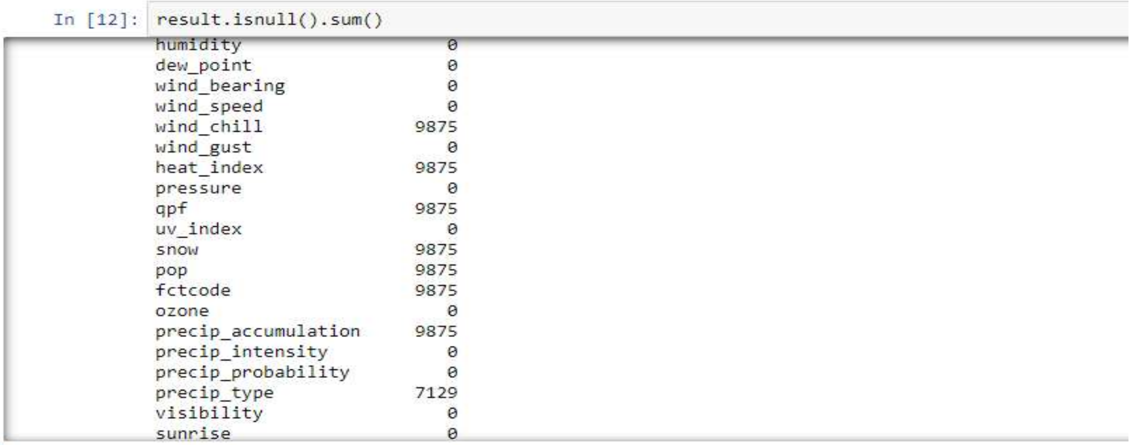
Feature Engineering

1. Converting variables into proper **date time** format.

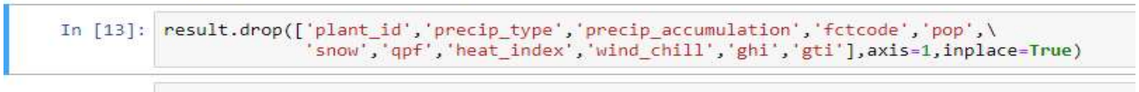


```
In [11]: result['datetime_utc'] = pd.to_datetime(result['datetime_utc'])
         result['datetime_local'] = pd.to_datetime(result['datetime_local'])
         result['sunrise'] = pd.to_datetime(result['sunrise'])
         result['sunset'] = pd.to_datetime(result['sunset'])
         result['updated_at'] = pd.to_datetime(result['updated_at'])
```

2. Identifying **Null values** and **Drop** the variables.

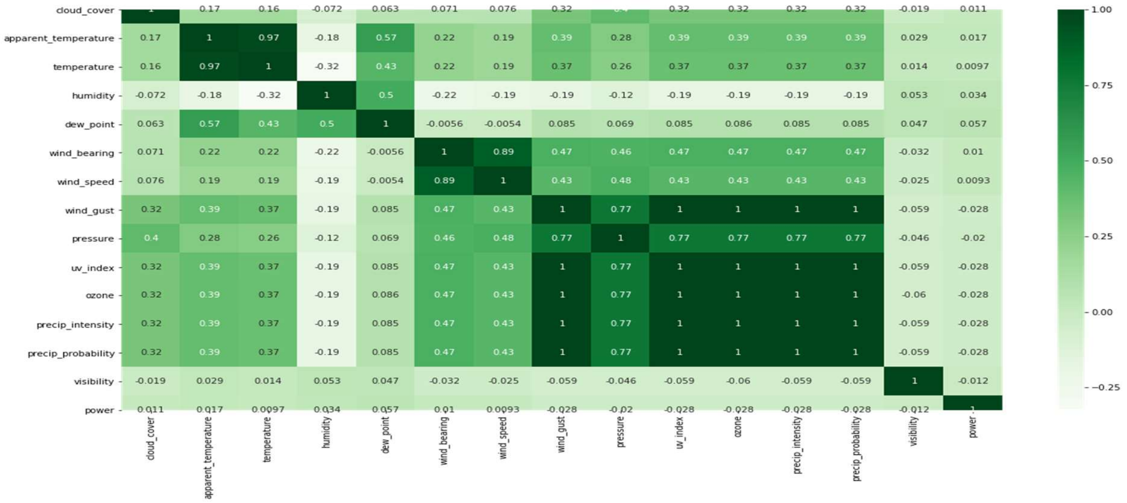


```
In [12]: result.isnull().sum()
         humidity                 0
         dew_point                0
         wind_bearing             0
         wind_speed               0
         wind_chill            9875
         wind_gust                0
         heat_index            9875
         pressure                 0
         qpf                   9875
         uv_index                 0
         snow                  9875
         pop                   9875
         fctcode               9875
         ozone                    0
         precip_accumulation   9875
         precip_intensity         0
         precip_probability       0
         precip_type           7129
         visibility               0
         sunrise                  0
```



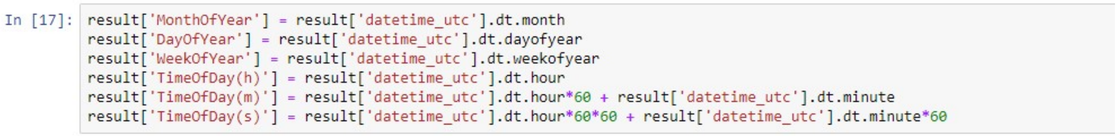
```
In [13]: result.drop(['plant_id','precip_type','precip_accumulation','fctcode','pop',\
                      'snow','qpf','heat_index','wind_chill','ghi','gti'],axis=1,inplace=True)
```

3. **Heat map** describing the correlation between the variables and drop the variables highly correlated with each other.

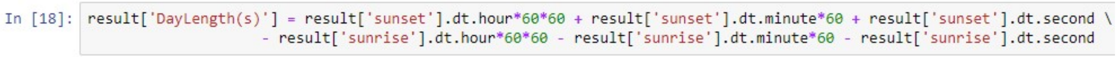


	cloud_cover	apparent_temperature	temperature	humidity	dew_point	wind_bearing	wind_speed	wind_gust	pressure	uv_index	ozone	precip_intensity	precip_probability	visibility	power
cloud_cover	1	0.17	0.16	-0.072	0.063	0.071	0.076	0.32	0.4	0.32	0.32	0.32	0.32	-0.019	0.011
apparent_temperature	0.17	1	0.97	-0.18	0.57	0.22	0.19	0.39	0.28	0.39	0.39	0.39	0.39	0.029	0.017
temperature	0.16	0.97	1	-0.32	0.43	0.22	0.19	0.37	0.26	0.37	0.37	0.37	0.37	0.014	0.0097
humidity	-0.072	-0.18	-0.32	1	0.5	-0.22	-0.19	-0.19	-0.12	-0.19	-0.19	-0.19	-0.19	0.053	0.034
dew_point	0.063	0.57	0.43	0.5	1	-0.0056	-0.0054	0.085	0.069	0.085	0.086	0.085	0.085	0.047	0.057
wind_bearing	0.071	0.22	0.22	-0.22	-0.0056	1	0.89	0.47	0.46	0.47	0.47	0.47	0.47	-0.032	0.01
wind_speed	0.076	0.19	0.19	-0.19	-0.0054	0.89	1	0.43	0.48	0.43	0.43	0.43	0.43	-0.025	0.0093
wind_gust	0.32	0.39	0.37	-0.19	0.085	0.47	0.43	1	0.77	1	1	1	1	-0.059	-0.028
pressure	0.4	0.28	0.26	-0.12	0.069	0.46	0.48	0.77	1	0.77	0.77	0.77	0.77	-0.046	-0.02
uv_index	0.32	0.39	0.37	-0.19	0.085	0.47	0.43	1	0.77	1	1	1	1	-0.059	-0.028
ozone	0.32	0.39	0.37	-0.19	0.086	0.47	0.43	1	0.77	1	1	1	1	-0.06	-0.028
precip_intensity	0.32	0.39	0.37	-0.19	0.085	0.47	0.43	1	0.77	1	1	1	1	-0.059	-0.028
precip_probability	0.32	0.39	0.37	-0.19	0.085	0.47	0.43	1	0.77	1	1	1	1	-0.059	-0.028
visibility	-0.019	0.029	0.014	0.053	0.047	-0.032	-0.025	-0.059	-0.046	-0.059	-0.06	-0.059	-0.059	1	-0.012
power	0.011	0.017	0.0097	0.034	0.057	0.01	0.0093	-0.028	-0.02	-0.028	-0.028	-0.028	-0.028	-0.012	1

4. Calculating the time duration between **Sunrise** and **Sunset**.

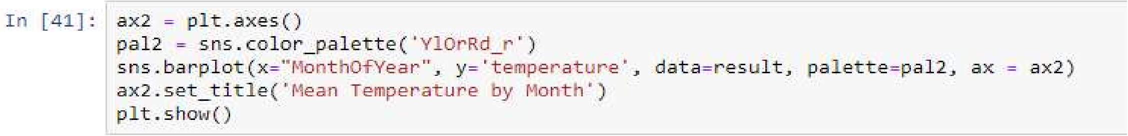


```
In [17]: result['MonthOfYear'] = result['datetime_utc'].dt.month
         result['DayOfYear'] = result['datetime_utc'].dt.dayofyear
         result['WeekOfYear'] = result['datetime_utc'].dt.weekofyear
         result['TimeOfDay(h)'] = result['datetime_utc'].dt.hour
         result['TimeOfDay(m)'] = result['datetime_utc'].dt.hour*60 + result['datetime_utc'].dt.minute
         result['TimeOfDay(s)'] = result['datetime_utc'].dt.hour*60*60 + result['datetime_utc'].dt.minute*60
```

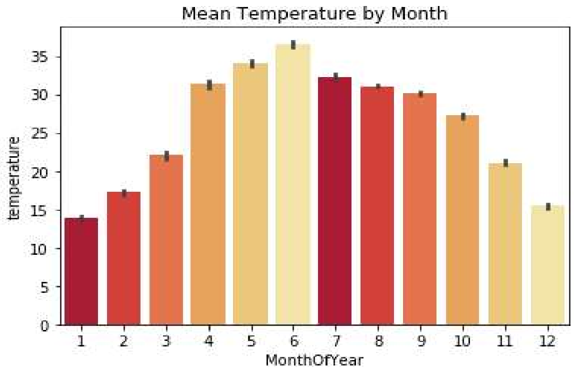


```
In [18]: result['DayLength(s)'] = result['sunset'].dt.hour*60*60 + result['sunset'].dt.minute*60 + result['sunset'].dt.second \
                                - result['sunrise'].dt.hour*60*60 - result['sunrise'].dt.minute*60 - result['sunrise'].dt.second
```

5. visualizing the **temperature** vary in month. It describes in **month of 4, 5, 6(summer)** keeps on increasing and decreases **after summer over**.

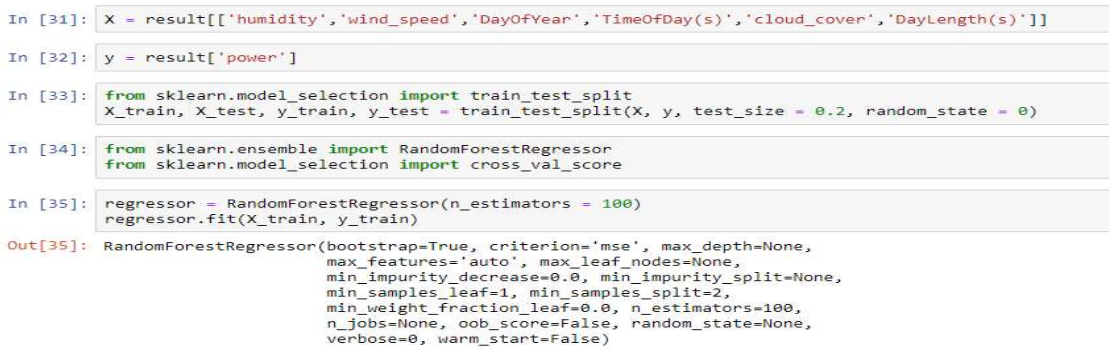


```
In [41]: ax2 = plt.axes()
         pal2 = sns.color_palette('YlOrRd_r')
         sns.barplot(x="MonthOfYear", y='temperature', data=result, palette=pal2, ax = ax2)
         ax2.set_title('Mean Temperature by Month')
         plt.show()
```



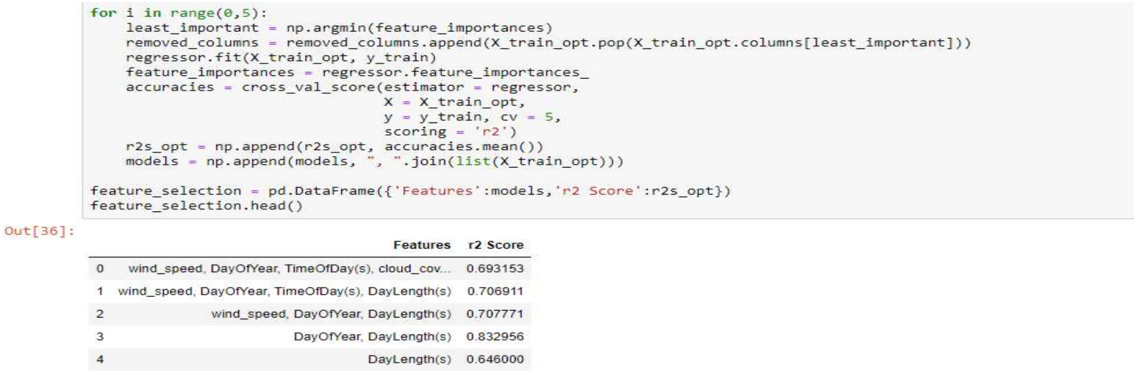
Model Selection:

1]. Select the suitable variables for prediction and split the data and applying **RandomForestRegressor (RFR)** Algorithm.



```
In [31]: X = result[['humidity','wind_speed','DayOfYear','TimeOfDay(s)','cloud_cover','DayLength(s)']]

In [32]: y = result['power']

In [33]: 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)

In [34]: from sklearn.ensemble import RandomForestRegressor
         from sklearn.model_selection import cross_val_score

In [35]: regressor = RandomForestRegressor(n_estimators = 100)
         regressor.fit(X_train, y_train)

Out[35]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                               max_features='auto', max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=100,
                               n_jobs=None, oob_score=False, random_state=None,
                               verbose=0, warm_start=False)
```

2. Applying feature Importance we get r2 score. From this we can Say **DayOfYear and DayLength(s)** are highly correlated with target variable with **r2 score** of **0.832956.**

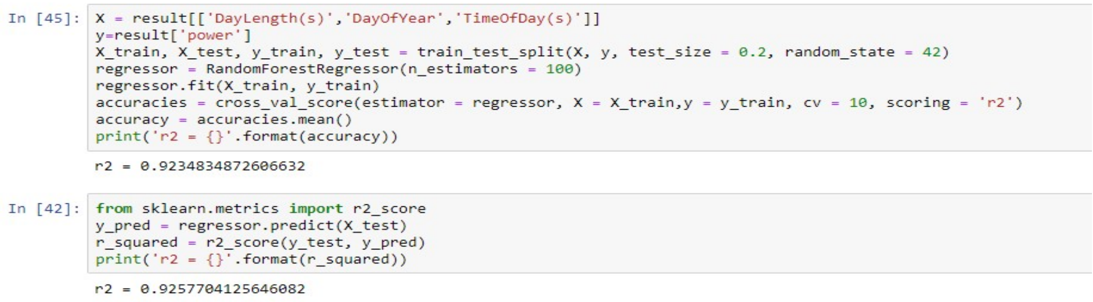


```
for i in range(0,5):
    least_important = np.argmin(feature_importances)
    removed_columns = removed_columns.append(X_train_opt.pop(X_train_opt.columns[least_important]))
    regressor.fit(X_train_opt, y_train)
    feature_importances = regressor.feature_importances_
    accuracies = cross_val_score(estimator = regressor,
                                 X = X_train_opt,
                                 y = y_train, cv = 5,
                                 scoring = 'r2')
    r2s_opt = np.append(r2s_opt, accuracies.mean())
    models = np.append(models, ", ".join(list(X_train_opt)))

feature_selection = pd.DataFrame({'Features':models,'r2 Score':r2s_opt})
feature_selection.head()
```

Out[36]:

	Features	r2 Score
0	wind_speed, DayOfYear, TimeOfDay(s), cloud_cov...	0.693153
1	wind_speed, DayOfYear, TimeOfDay(s), DayLength(s)	0.706911
2	wind_speed, DayOfYear, DayLength(s)	0.707771
3	DayOfYear, DayLength(s)	0.832956
4	DayLength(s)	0.646000

3. Finally the Best variables which predict the power supply are **DayOfLength(s), DayOfYear, and TimeOfDay** with **r2 score** of **0.9234** even it performs better on test data with r2 score more than train data with r2 score of **0.92577** on test data as well.



```
In [45]: X = result[['DayLength(s)','DayOfYear','TimeOfDay(s)']]
         y=result['power']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
         regressor = RandomForestRegressor(n_estimators = 100)
         regressor.fit(X_train, y_train)
         accuracies = cross_val_score(estimator = regressor, X = X_train,y = y_train, cv = 10, scoring = 'r2')
         accuracy = accuracies.mean()
         print('r2 = {}'.format(accuracy))

         r2 = 0.9234834872606632

In [42]: from sklearn.metrics import r2_score
         y_pred = regressor.predict(X_test)
         r_squared = r2_score(y_test, y_pred)
         print('r2 = {}'.format(r_squared))

         r2 = 0.9257704125646082
```

Conclusion:

From the Given Data DayLength in seconds and DayOfYear and TimeOfDay in seconds gives best predictive model for prediction of Solar Power Generation.