In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import plot
```

In [2]:

```python
df=pd.read_csv("C:\\Users\\Admin\\Downloads\\SolarPrediction.csv")
df
```

Out[2]:

	UNIXTime	Data	Time	Radiation	Temperature	Pressure	Humidity	WindDirection(Degrees)	Speed	TimeSunRise	TimeSunS
0	1475229326	9/29/2016 12:00:00 AM	23:55:26	1.21	48	30.46	59	177.39	5.62	06:13:00	18:13:0
1	1475229023	9/29/2016 12:00:00 AM	23:50:23	1.21	48	30.46	58	176.78	3.37	06:13:00	18:13:0
2	1475228726	9/29/2016 12:00:00 AM	23:45:26	1.23	48	30.46	57	158.75	3.37	06:13:00	18:13:0
3	1475228421	9/29/2016 12:00:00 AM	23:40:21	1.21	48	30.46	60	137.71	3.37	06:13:00	18:13:0
4	1475228124	9/29/2016 12:00:00 AM	23:35:24	1.17	48	30.46	62	104.95	5.62	06:13:00	18:13:0
...	...	...	...	...	...	...	...	...	...	...	
32681	1480587604	12/1/2016 12:00:00 AM	00:20:04	1.22	44	30.43	102	145.42	6.75	06:41:00	17:42:0
32682	1480587301	12/1/2016 12:00:00 AM	00:15:01	1.17	44	30.42	102	117.78	6.75	06:41:00	17:42:0
32683	1480587001	12/1/2016 12:00:00 AM	00:10:01	1.20	44	30.42	102	145.19	9.00	06:41:00	17:42:0
32684	1480586702	12/1/2016 12:00:00 AM	00:05:02	1.23	44	30.42	101	164.19	7.87	06:41:00	17:42:0
32685	1480586402	12/1/2016 12:00:00 AM	00:00:02	1.20	44	30.43	101	83.59	3.37	06:41:00	17:42:0

32686 rows × 11 columns

In [3]:

```python
df.describe(include='all')
```

Out[3]:

	UNIXTime	Data	Time	Radiation	Temperature	Pressure	Humidity	WindDirection(Degrees)	Speed	Ti
count	3.268600e+04	32686	32686	32686.000000	32686.000000	32686.000000	32686.000000	32686.000000	32686.000000	
unique	NaN	118	8299	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	12/1/2016 12:00:00 AM	16:20:18	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	288	24	NaN	NaN	NaN	NaN	NaN	NaN	
mean	1.478047e+09	NaN	NaN	207.124697	51.103255	30.422879	75.016307	143.489821	6.243869	
std	3.005037e+06	NaN	NaN	315.916387	6.201157	0.054673	25.990219	83.167500	3.490474	
min	1.472724e+09	NaN	NaN	1.110000	34.000000	30.190000	8.000000	0.090000	0.000000	
25%	1.475546e+09	NaN	NaN	1.230000	46.000000	30.400000	56.000000	82.227500	3.370000	
50%	1.478026e+09	NaN	NaN	2.660000	50.000000	30.430000	85.000000	147.700000	5.620000	
75%	1.480480e+09	NaN	NaN	354.235000	55.000000	30.460000	97.000000	179.310000	7.870000	
max	1.483265e+09	NaN	NaN	1601.260000	71.000000	30.560000	103.000000	359.950000	40.500000	

In [4]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32686 entries, 0 to 32685
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   UNIXTime                32686 non-null  int64  
 1   Data                    32686 non-null  object 
 2   Time                    32686 non-null  object 
 3   Radiation               32686 non-null  float64
 4   Temperature             32686 non-null  int64  
 5   Pressure                32686 non-null  float64
 6   Humidity                32686 non-null  int64  
 7   WindDirection(Degrees)  32686 non-null  float64
 8   Speed                   32686 non-null  float64
 9   TimeSunRise             32686 non-null  object 
 10  TimeSunSet              32686 non-null  object 
dtypes: float64(4), int64(3), object(4)
memory usage: 2.7+ MB
```

In [5]:

```
df.shape
```

Out[5]:

```
(32686, 11)
```

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
UNIXTime                  0
Data                      0
Time                      0
Radiation                 0
Temperature               0
Pressure                  0
Humidity                  0
WindDirection(Degrees)    0
Speed                     0
TimeSunRise               0
TimeSunSet                0
dtype: int64
```

In [7]:

```
df.drop(columns=['Data','Time'],inplace=True) #droping the columns which are not required
```

In [8]:

```
df
```

Out[8]:

	UNIXTime	Radiation	Temperature	Pressure	Humidity	WindDirection(Degrees)	Speed	TimeSunRise	TimeSunSet
0	1475229326	1.21	48	30.46	59	177.39	5.62	06:13:00	18:13:00
1	1475229023	1.21	48	30.46	58	176.78	3.37	06:13:00	18:13:00
2	1475228726	1.23	48	30.46	57	158.75	3.37	06:13:00	18:13:00
3	1475228421	1.21	48	30.46	60	137.71	3.37	06:13:00	18:13:00
4	1475228124	1.17	48	30.46	62	104.95	5.62	06:13:00	18:13:00
...	...	...	...	...	...	...	...	...	...
32681	1480587604	1.22	44	30.43	102	145.42	6.75	06:41:00	17:42:00
32682	1480587301	1.17	44	30.42	102	117.78	6.75	06:41:00	17:42:00
32683	1480587001	1.20	44	30.42	102	145.19	9.00	06:41:00	17:42:00
32684	1480586702	1.23	44	30.42	101	164.19	7.87	06:41:00	17:42:00
32685	1480586402	1.20	44	30.43	101	83.59	3.37	06:41:00	17:42:00

32686 rows × 9 columns

In [9]:

```
#changing the data in datetime format
df['TimeSunRise'] = pd.to_datetime(df['TimeSunRise'], format='%H:%M:%S')
df['TimeSunSet'] = pd.to_datetime(df['TimeSunSet'], format='%H:%M:%S')
```

In [10]:

```
df['TSRhour'] = df['TimeSunRise'].dt.hour.astype(int)
df['TSRmin'] = df['TimeSunRise'].dt.minute.astype(int)
df['TSShour'] = df['TimeSunSet'].dt.hour.astype(int)
df['TSSmin'] = df['TimeSunSet'].dt.minute.astype(int)
df.drop(columns=['TimeSunRise','TimeSunSet'],inplace=True)
df
```

Out[10]:

	UNIXTime	Radiation	Temperature	Pressure	Humidity	WindDirection(Degrees)	Speed	TSRhour	TSRmin	TSShour	TSSmin
0	1475229326	1.21	48	30.46	59	177.39	5.62	6	13	18	13
1	1475229023	1.21	48	30.46	58	176.78	3.37	6	13	18	13
2	1475228726	1.23	48	30.46	57	158.75	3.37	6	13	18	13
3	1475228421	1.21	48	30.46	60	137.71	3.37	6	13	18	13
4	1475228124	1.17	48	30.46	62	104.95	5.62	6	13	18	13
...	...	...	...	...	...	...	...	...	...	...	...
32681	1480587604	1.22	44	30.43	102	145.42	6.75	6	41	17	42
32682	1480587301	1.17	44	30.42	102	117.78	6.75	6	41	17	42
32683	1480587001	1.20	44	30.42	102	145.19	9.00	6	41	17	42
32684	1480586702	1.23	44	30.42	101	164.19	7.87	6	41	17	42
32685	1480586402	1.20	44	30.43	101	83.59	3.37	6	41	17	42

32686 rows × 11 columns

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32686 entries, 0 to 32685
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   UNIXTime                32686 non-null  int64  
 1   Radiation               32686 non-null  float64
 2   Temperature             32686 non-null  int64  
 3   Pressure                32686 non-null  float64
 4   Humidity                32686 non-null  int64  
 5   WindDirection(Degrees)  32686 non-null  float64
 6   Speed                   32686 non-null  float64
 7   TSRhour                 32686 non-null  int32  
 8   TSRmin                  32686 non-null  int32  
 9   TSShour                 32686 non-null  int32  
 10  TSSmin                  32686 non-null  int32  
dtypes: float64(4), int32(4), int64(3)
memory usage: 2.2 MB
```

In [12]:

```
Y = df[['Radiation']]
X = df.drop(columns=['Radiation'])
```

In [14]:

```
X
```

Out[14]:

	UNIXTime	Temperature	Pressure	Humidity	WindDirection(Degrees)	Speed	TSRhour	TSRmin	TSShour	TSSmin
0	1475229326	48	30.46	59	177.39	5.62	6	13	18	13
1	1475229023	48	30.46	58	176.78	3.37	6	13	18	13
2	1475228726	48	30.46	57	158.75	3.37	6	13	18	13
3	1475228421	48	30.46	60	137.71	3.37	6	13	18	13
4	1475228124	48	30.46	62	104.95	5.62	6	13	18	13
...	...	...	...	...	...	...	...	...	...	...
32681	1480587604	44	30.43	102	145.42	6.75	6	41	17	42
32682	1480587301	44	30.42	102	117.78	6.75	6	41	17	42
32683	1480587001	44	30.42	102	145.19	9.00	6	41	17	42
32684	1480586702	44	30.42	101	164.19	7.87	6	41	17	42
32685	1480586402	44	30.43	101	83.59	3.37	6	41	17	42

32686 rows × 10 columns

In [15]:

```
Y
```

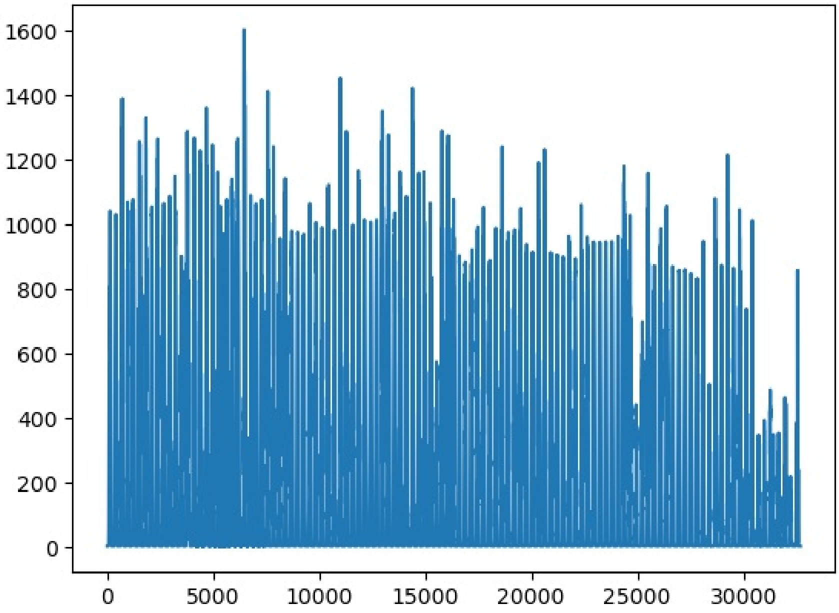
Out[15]:

	Radiation
0	1.21
1	1.21
2	1.23
3	1.21
4	1.17
...	...
32681	1.22
32682	1.17
32683	1.20
32684	1.23
32685	1.20

32686 rows × 1 columns

In [16]:
```
plt.plot(Y)
plt.show
```

Out[16]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [17]:
```
X.info()
Y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32686 entries, 0 to 32685
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   UNIXTime               32686 non-null  int64  
 1   Temperature            32686 non-null  int64  
 2   Pressure               32686 non-null  float64
 3   Humidity               32686 non-null  int64  
 4   WindDirection(Degrees)  32686 non-null  float64
 5   Speed                  32686 non-null  float64
 6   TSRhour                32686 non-null  int32  
 7   TSRmin                 32686 non-null  int32  
 8   TSShour                32686 non-null  int32  
 9   TSSmin                 32686 non-null  int32  
dtypes: float64(3), int32(4), int64(3)
memory usage: 2.0 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32686 entries, 0 to 32685
Data columns (total 1 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Radiation  32686 non-null  float64
dtypes: float64(1)
memory usage: 255.5 KB
```

AllFeatures

In [20]:
```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,random_state=42,shuffle=True)
```

In [21]:

```
from sklearn.ensemble import RandomForestRegressor
regr = RandomForestRegressor(max_depth=25, random_state=3)
regr.fit(x_train, y_train)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_25756\3187189820.py:3: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for ex
ample using ravel().
```

Out[21]:

```
RandomForestRegressor(max_depth=25, random_state=3)
```

In [22]:

```
Allfeatures=regr.score(x_train, y_train)
Allfeatures
```

Out[22]:

```
0.9868391602559925
```

In [23]:

```
regr.score(x_test, y_test)
```

Out[23]:

```
0.9066664115231688
```

In [24]:

```
#VarianceThreshold
```

In [25]:

```
x_train_1,x_test_1,y_train_1,y_test_1 = x_train.copy(),x_test.copy(),y_train.copy(),y_test.copy()
```

In [26]:

```
x_train_1.var(axis=0)
```

Out[26]:

```
UNIXTime                  9.059136e+12
Temperature               3.856347e+01
Pressure                  2.998423e-03
Humidity                  6.757779e+02
WindDirection(Degrees)    6.892881e+03
Speed                     1.198967e+01
TSRhour                   0.000000e+00
TSRmin                    2.411419e+02
TSShour                   2.269559e-01
TSSmin                    2.526056e+02
dtype: float64
```

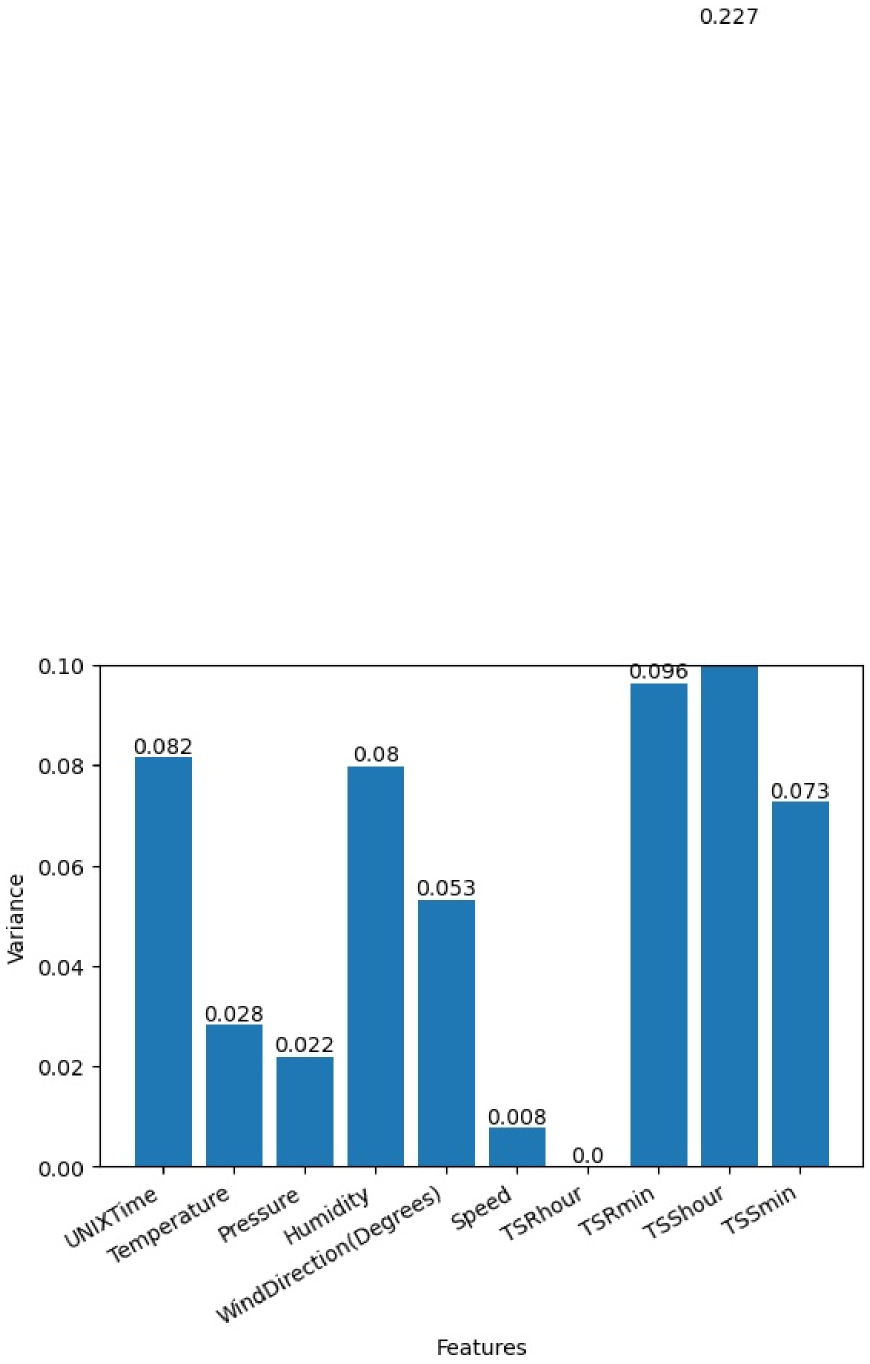
In [27]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_x_train_1= scaler.fit_transform(x_train_1)
```

In [28]:

```
fig,ax=plt.subplots()

x=X.columns
y=scaled_x_train_1.var(axis=0)

ax.bar(x,y,width=0.8)
ax.set_xlabel('Features')
ax.set_ylabel('Variance')
ax.set_ylim(0,0.1)

for index, value in enumerate(y):
    plt.text(x=index,y=value+0.001,s=str(round(value, 3)),ha='center')

fig.autofmt_xdate()
plt.tight_layout()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_25756\944144619.py:15: UserWarning:

Tight layout not applied. The bottom and top margins cannot be made large enough to accommodate all axes decora
tions.
```



In [29]:

```
from sklearn.metrics import f1_score
```

In [30]:

```
sel_x_train_1=x_train_1.drop(['Speed','TSRhour','Pressure'],axis=1)
sel_x_test_1=x_test_1.drop(['Speed','TSRhour','Pressure'],axis=1)
sel_y_train_1=x_train_1.drop(['Speed','TSRhour','Pressure'],axis=1)
sel_y_test_1=x_test_1.drop(['Speed','TSRhour','Pressure'],axis=1)

regr.fit(sel_x_train_1,sel_y_train_1)
```

Out[30]:

```
RandomForestRegressor(max_depth=25, random_state=3)
```

In [31]:

```
varianceScore=regr.score(sel_x_train_1,sel_y_train_1)
varianceScore
```

Out[31]:

```
0.9919279105929851
```

In [32]:

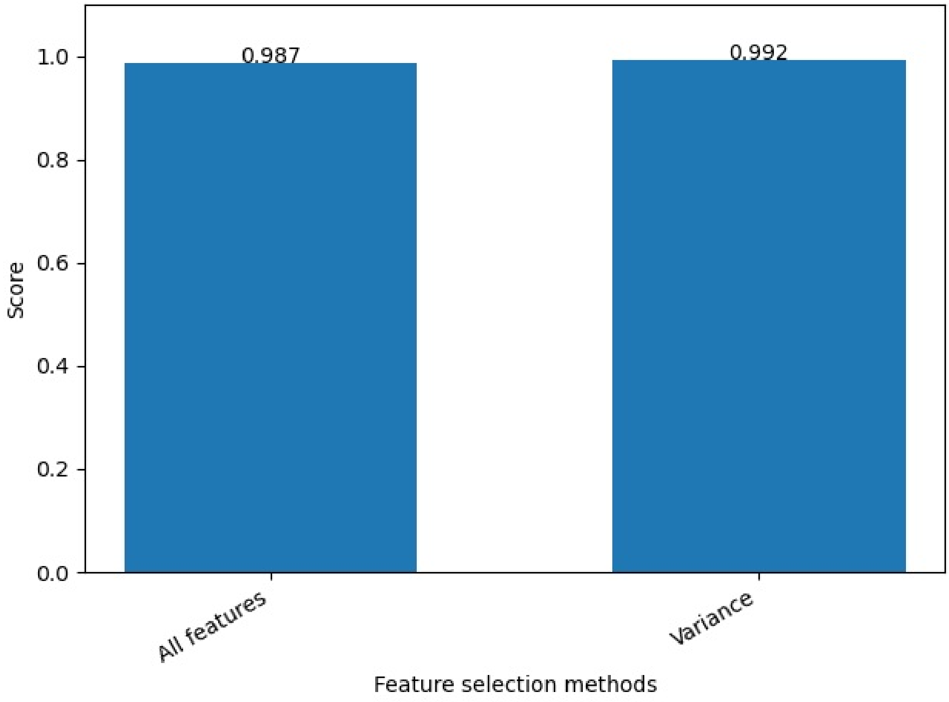
```
regr.score(sel_x_test_1,sel_y_test_1)
```

Out[32]:

```
0.9403113414068358
```

In [33]:

```
fig,ax=plt.subplots()

x=['All features','Variance']
y=[Allfeatures,varianceScore]

ax.bar(x,y,width=0.6)
ax.set_xlabel('Feature selection methods')
ax.set_ylabel('Score')
ax.set_ylim(0,1.1)

for index, value in enumerate(y):
    plt.text(x=index,y=value+0.001,s=str(round(value, 3)),ha='center')

fig.autofmt_xdate()
plt.tight_layout()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js