ClassifierTestBase

July 28, 2021

1 Solar Power Generation Data: KMM, Naive and Decision Tree.

1.1 Solar power generation and sensor data for two power plants.

Description This data has been gathered at two solar power plants in India over a 34 day period. It has two pairs of files - each pair has one power generation dataset and one sensor readings dataset. The power generation datasets are gathered at the inverter level - each inverter has multiple lines of solar panels attached to it. The sensor data is gathered at a plant level - single array of sensors optimally placed at the plant.

There are a few areas of concern at the solar power plant -

- Can we predict the power generation for next couple of days? - this allows for better grid management
- Can we identify generation profiles?
- Can we identify the need for panel cleaning/maintenance?
- Can we identify faulty or suboptimally performing equipment?

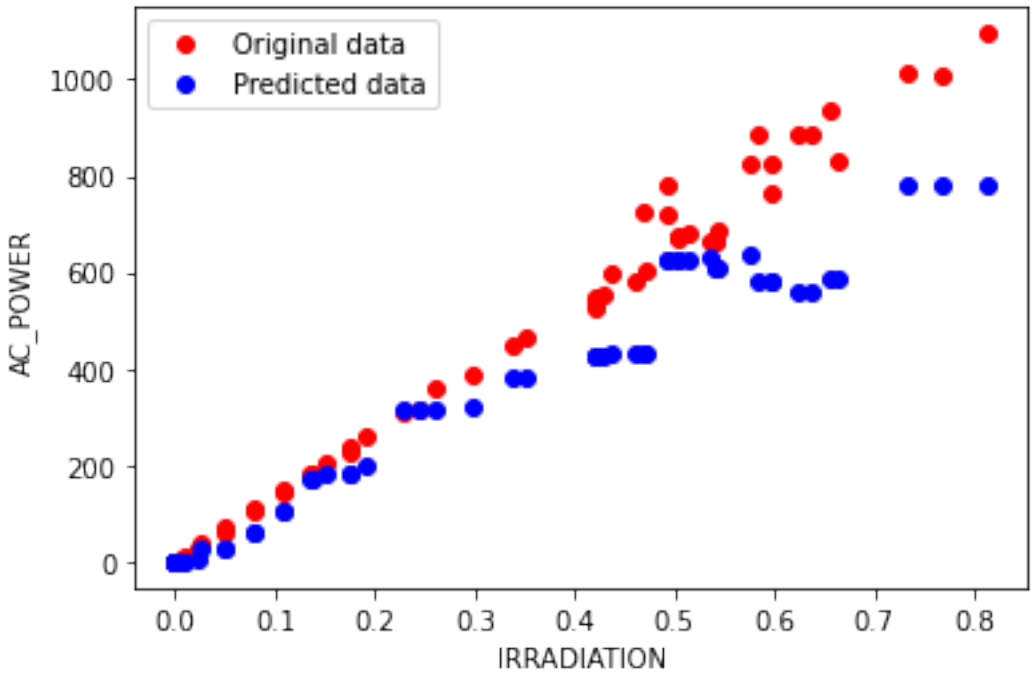
Link to source ## Aluno: Zaú Júlio A. Galvão

[1]:
```
import sys

sys.path.append('../')
```

[2]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math

from read import (
    sourcesGen01Cleaned,
    groupsWeather01Cleaned,
)
```

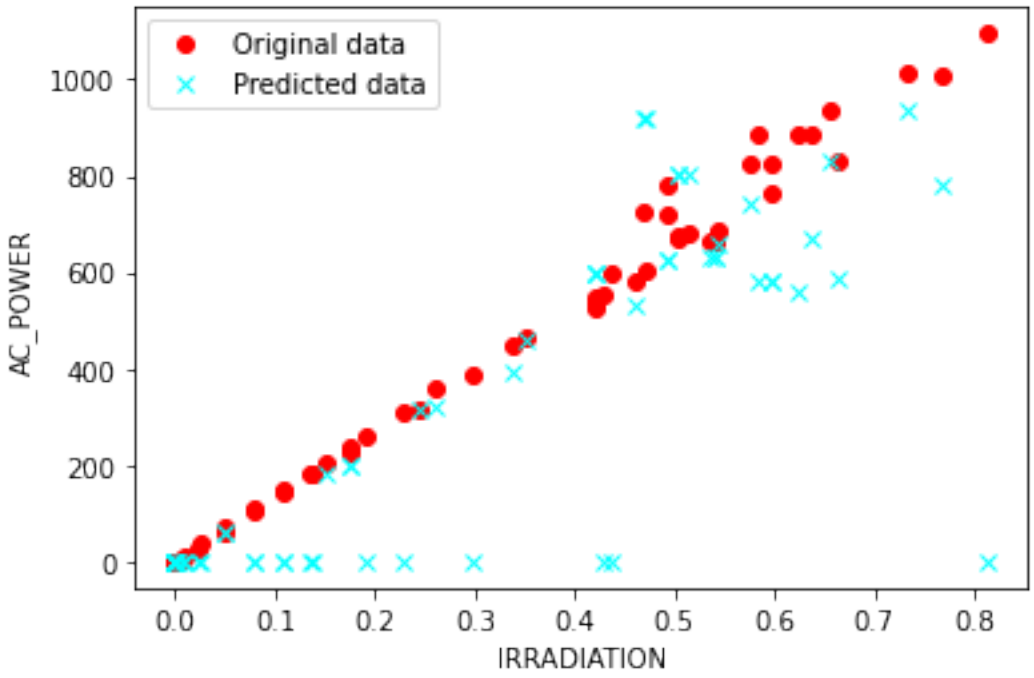
[3]:
```
X = groupsWeather01Cleaned["IRRADIATION"].T
y = sourcesGen01Cleaned["AC_POWER"][list(sourcesGen01Cleaned["AC_POWER"].
 ↪keys())[0]].T
```

```
X0 = X.iloc[0].values
y0 = y.iloc[0].values

X1 = X.iloc[1].values
y1 = y.iloc[1].values

plt.plot(X1, y1, 'o', color='red', label='Original data')
plt.xlabel("IRRADIATION")
plt.ylabel("AC_POWER")

neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X0.reshape(-1, 1), y0.astype(int))

pred = neigh.predict(X1.reshape(-1, 1))
knnScore = neigh.score(X0.reshape(-1, 1), y0.astype(int))

plt.plot(X1, pred, 'o', color='blue', label='Predicted data')
plt.legend()
plt.show()
```



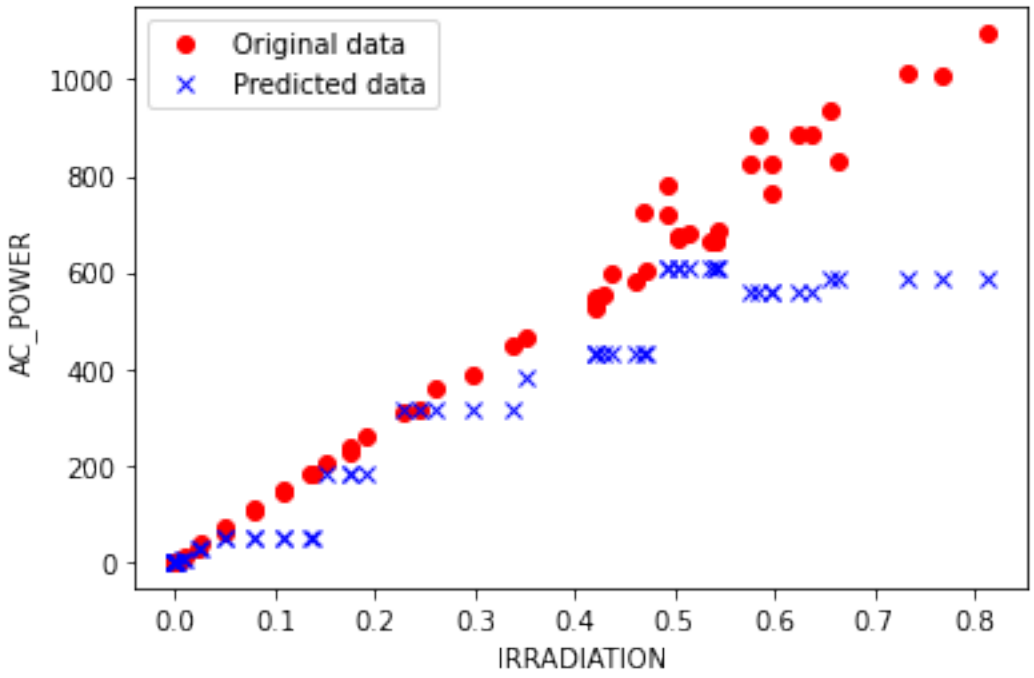
[4]:
```
X = groupsWeather01Cleaned["IRRADIATION"].T
y = sourcesGen01Cleaned["AC_POWER"][list(sourcesGen01Cleaned["AC_POWER"].
 ↪keys())[0]].T
```

```
X0 = X.iloc[0].values
y0 = y.iloc[0].values

X1 = X.iloc[1].values
y1 = y.iloc[1].values

plt.plot(X1, y1, 'o', color='red', label='Original data')
plt.xlabel("IRRADIATION")
plt.ylabel("AC_POWER")

clf = GaussianNB()
clf.fit(X0.reshape(-1, 1), y0.astype(int))

pred = clf.predict(X1.reshape(-1, 1))
naiveBayesScore = clf.score(X0.reshape(-1, 1), y0.astype(int))

plt.plot(X1, pred, 'x', color='cyan', label='Predicted data')
plt.legend()
plt.show()
```



[5]:
```
X = groupsWeather01Cleaned["IRRADIATION"].T
y = sourcesGen01Cleaned["AC_POWER"][list(sourcesGen01Cleaned["AC_POWER"].
 ↪keys())[0]].T
```

```
X0 = X.iloc[0].values
y0 = y.iloc[0].values

X1 = X.iloc[1].values
y1 = y.iloc[1].values

plt.plot(X1, y1, 'o', color='red', label='Original data')
plt.xlabel("IRRADIATION")
plt.ylabel("AC_POWER")

clf = DecisionTreeClassifier(random_state=0, criterion="entropy", max_depth=4)
clf.fit(X0.reshape(-1, 1), y0.astype(int))

pred = clf.predict(X1.reshape(-1, 1))
treeScore = clf.score(X0.reshape(-1, 1), y0.astype(int))

plt.plot(X1, pred, 'x', color='blue', label='Predicted data')
plt.legend()
plt.show()
```



[6]:
```
height = [knnScore, naiveBayesScore, treeScore]
bars = ('KNeighbors', 'Naive Bayes', 'Tree Decision')
x_pos = np.arange(len(bars))
```

```
plt.bar(x_pos, height)
plt.xticks(x_pos, bars)
plt.show()
```