ch1

November 23, 2019

[18]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('../../data/training_dataset_500.csv')
```

[19]:
```
from sklearn.preprocessing import PolynomialFeatures, LabelEncoder, MinMaxScaler
df = df.drop(columns=['ID','Label'])
#df = df[df.Month.isin([4,5,6,7,8])]
df.corr()
```

[19]:

	House	Year	Month	Temperature	Daylight	\
House	1.000000e+00	0.000000	-1.816873e-18	0.000881	0.001583	
Year	0.000000e+00	1.000000	-6.340757e-01	-0.356800	0.524603	
Month	-1.816873e-18	-0.634076	1.000000e+00	0.353837	-0.276307	
Temperature	8.810764e-04	-0.356800	3.538369e-01	1.000000	-0.053363	
Daylight	1.582656e-03	0.524603	-2.763068e-01	-0.053363	1.000000	
EnergyProduction	-8.302696e-03	0.267481	-2.327484e-01	0.272789	0.531577	

	EnergyProduction
House	-0.008303
Year	0.267481
Month	-0.232748
Temperature	0.272789
Daylight	0.531577
EnergyProduction	1.000000

[20]:
```
def preprocessing(df):
    #from keras.utils import np_utils
    X = df[['House','Month','Temperature','Daylight']]
    #X = X[X.Month.isin([4,5,6,7,8])]
    y = df[['EnergyProduction']]
    #enc = LabelEncoder()
    #house = X.House.values.reshape(-1,1)
    #X['House'] = enc.fit(house).transform(house).toarray()
    #sc = MinMaxScaler()
    #X = sc.fit_transform(X)
    return np.array(X), np.array(y).reshape(len(y),)
```

[21]:
```
def MAPE(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true))*100
```

[22]:
```
from sklearn import linear_model
from sklearn import svm
reg = linear_model.LinearRegression()
#reg = sum.SVR(kernel='rbf', C=8, gamma=5e-5)
X, y = preprocessing(df)
reg.fit(X, y)
```

[22]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[23]:
```
dft = pd.read_csv('../../data/test_dataset_500.csv')
Xt, yt = preprocessing(dft)
```

[24]:
```
MAPE(yt, reg.predict(Xt))
```

[24]:
```
14.450698807918974
```

[25]:
```
def individual(houseId, df=df):
    X = df[['House','Month','Temperature','Daylight']]
    X = X[X.House==houseId].drop(columns=['House'])
    y = df[['House','EnergyProduction']]
    y = y[y.House==houseId].drop(columns=['House'])
    return np.array(X), np.array(y)
```

[52]:
```
# individual prognoses
ireg = linear_model.LinearRegression()
y_pred, y_true = [],[]
with open('predicted_energy_production.csv','w') as f:
    f.write('House, EnergyProduction\n')
    for i in set(dft.House.values):
        x,y = individual(i)
        ireg.fit(x,y)
        p = ireg.predict(individual(i,dft)[0])[0][0]
        y_pred.append(p)
        y_true.append((individual(i,dft)[1])[0][0])
        f.write('{}, {}\n'.format(i,p))
with open('mape.txt','w') as f:
    f.write(str(MAPE(y_true, y_pred)))
```

[27]:
```
# neural network
```

[29]:
```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
```

[40]:
```
model = Sequential()
model.add(Dense(units=40, activation='relu', input_dim=4))
model.add(Dense(units=20, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam',
```

```
              loss='mse',
              metrics=['mape'])

X, y = preprocessing(df)
model.fit(X, y, epochs=12, verbose=3)

Xt, yt = preprocessing(dft)
model.evaluate(Xt, yt)
```

```
Epoch 1/12
Epoch 2/12
Epoch 3/12
Epoch 4/12
Epoch 5/12
Epoch 6/12
Epoch 7/12
Epoch 8/12
Epoch 9/12
Epoch 10/12
Epoch 11/12
Epoch 12/12
500/500 [==============================] - 0s 138us/step
```

[40]:
```
[21690.6206796875, 17.908103942871094]
```

[787]:
```
#LTSE
```