week6_Dowell

January 30, 2021

[ ]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
df = pd.read_csv("/content/drive/MyDrive/dowell/week 6/Week 6 Data - Sheet1.
 ↪csv", header =1)
```

[ ]:
```
print(df.shape)
df.head()
```

```
(1000, 31)
```

[ ]:
```
   Days  M  T  W  TH  F  M.1  T.1  …  W.4  TH.4  F.4  M.5  T.5  W.5  TH.5  F.5
0     1  4  5  5   6  7    1    2  …    7     7    7  4.0  5.0  6.0   7.0  8.0
1     2  1  2  3   4  5    5    5  …    5     6    7  NaN  5.0  6.0   7.0  8.0
2     3  3  4  5   6  7    2    3  …    5     6    7  5.0  NaN  NaN   7.0  8.0
3     4  1  2  3   4  5    1    2  …    6     7    7  2.0  3.0  NaN   5.0  6.0
4     5  2  3  4   5  6    5    5  …    5     6    7  4.0  NaN  6.0   7.0  8.0

[5 rows x 31 columns]
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['Days', 'M', 'T', 'W', 'TH', 'F', 'M.1', 'T.1', 'W.1', 'TH.1', 'F.1',
       'M.2', 'T.2', 'W.2', 'TH.2', 'F.2', 'M.3', 'T.3', 'W.3', 'TH.3', 'F.3',
       'M.4', 'T.4', 'W.4', 'TH.4', 'F.4', 'M.5', 'T.5', 'W.5', 'TH.5', 'F.5'],
      dtype='object')
```

[ ]:
```
data = df.rename(columns={'Days':'Student',
 'M':"M1", 'T': "T1", 'W':"W1", 'TH':"TH1", 'F':"F1",
 'M.1':"M2", 'T.1':"T2", 'W.1':"W2", 'TH.1':"TH2",'F.1':"F2",
 'M.2':'M3', 'T.2':"T3", 'W.2':"W3", 'TH.2':"TH3", 'F.2':"F3",
 "M.3":"M4", "T.3":"T4","W.3":"W4","TH.3":"TH4","F.3":"F4",
 "M.4":"M5","T.4":"T5","W.4":"W5","TH.4":"TH5","F.4":"F5",'M.5':"M6", 'T.5':
 ↪"T6", 'W.5':"W6", 'TH.5':"TH6", 'F.5':"F6"})
```

[ ]:
```
data.columns
```

```
[ ]: Index(['Student', 'M1', 'T1', 'W1', 'TH1', 'F1', 'M2', 'T2', 'W2', 'TH2', 'F2',
            'M3', 'T3', 'W3', 'TH3', 'F3', 'M4', 'T4', 'W4', 'TH4', 'F4', 'M5',
            'T5', 'W5', 'TH5', 'F5', 'M6', 'T6', 'W6', 'TH6', 'F6'],
           dtype='object')
```

```
[ ]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 31 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Student  1000 non-null   int64
 1   M1       1000 non-null   int64
 2   T1       1000 non-null   int64
 3   W1       1000 non-null   int64
 4   TH1      1000 non-null   int64
 5   F1       1000 non-null   int64
 6   M2       1000 non-null   int64
 7   T2       1000 non-null   int64
 8   W2       1000 non-null   int64
 9   TH2      1000 non-null   int64
 10  F2       1000 non-null   int64
 11  M3       1000 non-null   int64
 12  T3       1000 non-null   int64
 13  W3       1000 non-null   int64
 14  TH3      1000 non-null   int64
 15  F3       1000 non-null   int64
 16  M4       1000 non-null   int64
 17  T4       1000 non-null   int64
 18  W4       1000 non-null   int64
 19  TH4      1000 non-null   int64
 20  F4       1000 non-null   int64
 21  M5       1000 non-null   int64
 22  T5       1000 non-null   int64
 23  W5       1000 non-null   int64
 24  TH5      1000 non-null   int64
 25  F5       1000 non-null   int64
 26  M6       819 non-null    float64
 27  T6       816 non-null    float64
 28  W6       817 non-null    float64
 29  TH6      784 non-null    float64
 30  F6       821 non-null    float64
dtypes: float64(5), int64(26)
memory usage: 242.3 KB
```

```
[ ]: data.skew()
[ ]: Student    0.000000
     M1        -0.065773
     T1        -0.433057
     W1        -1.097004
     TH1       -1.235211
     F1        -1.266516
     M2         0.072407
     T2        -0.299664
     W2        -0.343609
     TH2       -0.980083
     F2        -1.074264
     M3        -0.087948
     T3        -0.108119
     W3        -0.547100
     TH3       -0.639398
     F3        -1.451353
     M4        -0.022931
     T4        -0.332342
     W4        -0.392791
     TH4       -0.846319
     F4        -1.587826
     M5         0.001339
     T5        -0.010587
     W5        -0.303412
     TH5       -0.777279
     F5        -1.496726
     M6        -0.039754
     T6        -0.352240
     W6        -0.838248
     TH6       -0.917327
     F6        -0.904050
     dtype: float64
```

```
[ ]: data.kurtosis()
```

```
[ ]: Student   -1.200000
     M1        -1.305880
     T1        -1.170964
     W1         0.405842
     TH1        1.207742
     F1         1.366346
     M2        -1.321019
     T2        -1.269884
     W2        -0.951474
     TH2        0.445306
     F2         0.738511
     M3        -1.259816
```

```
T3        -1.087696
W3        -0.657309
TH3       -0.251166
F3         2.138656
M4        -1.269557
T4        -1.251389
W4        -1.002642
TH4       -0.194935
F4         2.365219
M5        -1.325567
T5        -1.217117
W5        -1.098261
TH5       -0.294989
F5         1.995675
M6        -1.275539
T6        -1.185389
W6        -0.331043
TH6        0.137185
F6         0.256055
dtype: float64
```

[107]:

```
week_1_4 = data[data.columns[0:21]]
```

[ ]:

```
week_5_6 = data[data.columns[21:31]]
week_5_6.head()
```

[ ]:

```
   M5  T5  W5  TH5  F5   M6   T6   W6  TH6   F6
0   5   6   7    7   7  4.0  5.0  6.0  7.0  8.0
1   3   4   5    6   7  NaN  5.0  6.0  7.0  8.0
2   3   4   5    6   7  5.0  NaN  NaN  7.0  8.0
3   4   5   6    7   7  2.0  3.0  NaN  5.0  6.0
4   3   4   5    6   7  4.0  NaN  6.0  7.0  8.0
```

[ ]:

```
week_5_6.describe()
```

[ ]:

```
                M5           T5  …         TH6          F6
count  1000.000000  1000.000000  …  784.000000  821.000000
mean      3.450000     4.323000  …    5.720663    6.573691
std       1.741417     1.769124  …    1.377536    1.488489
min       1.000000     1.000000  …    1.000000    2.000000
25%       2.000000     3.000000  …    5.000000    6.000000
50%       3.000000     4.000000  …    6.000000    7.000000
75%       5.000000     6.000000  …    7.000000    8.000000
max       6.000000     7.000000  …    7.000000    8.000000

[8 rows x 10 columns]
```

```
[ ]: week_5_6.skew()
```

```
[ ]: M5      0.001339
     T5     -0.010587
     W5     -0.303412
     TH5    -0.777279
     F5     -1.496726
     M6     -0.039754
     T6     -0.352240
     W6     -0.838248
     TH6    -0.917327
     F6     -0.904050
     dtype: float64
```

```
[ ]: week_5_6.kurtosis()
```

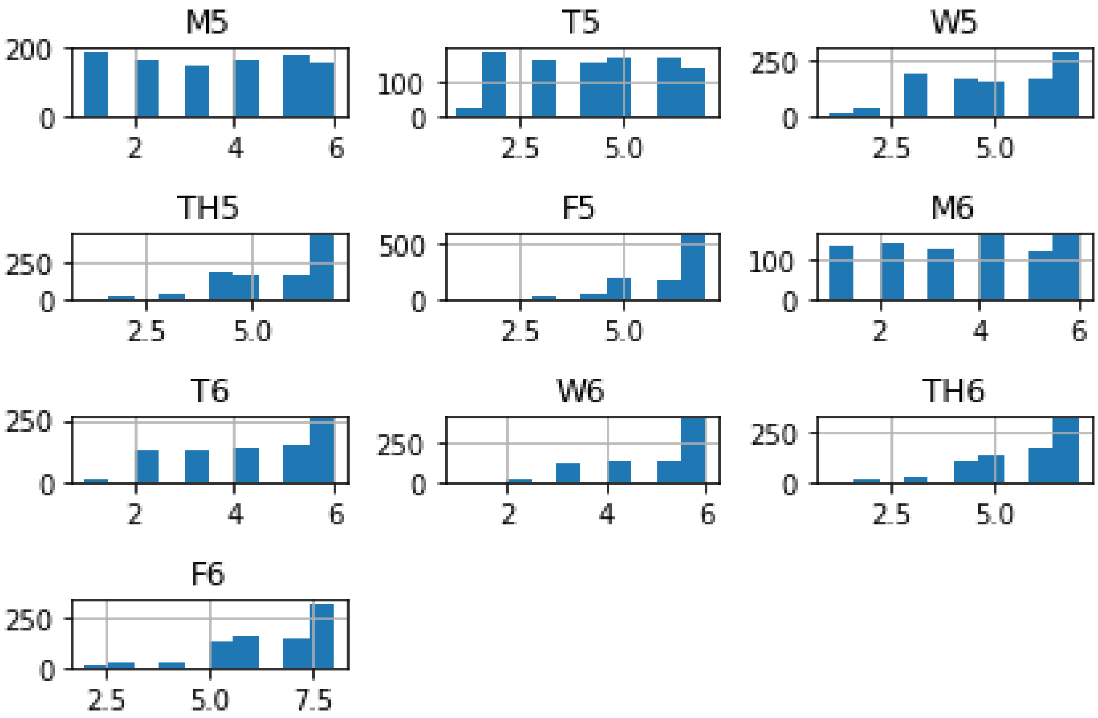
```
[ ]: M5     -1.325567
     T5     -1.217117
     W5     -1.098261
     TH5    -0.294989
     F5      1.995675
     M6     -1.275539
     T6     -1.185389
     W6     -0.331043
     TH6     0.137185
     F6      0.256055
     dtype: float64
```

0.1 check the variance

```
[ ]: week_5_6.var(axis=0,ddof=1)
```

```
[ ]: M5      3.032533
     T5      3.129801
     W5      2.695206
     TH5     2.048364
     F5      1.463864
     M6      2.986202
     T6      2.361168
     W6      1.686578
     TH6     1.897606
     F6      2.215600
     dtype: float64
```

```
[139]: week_5_6.hist()

       plt.tight_layout()
```



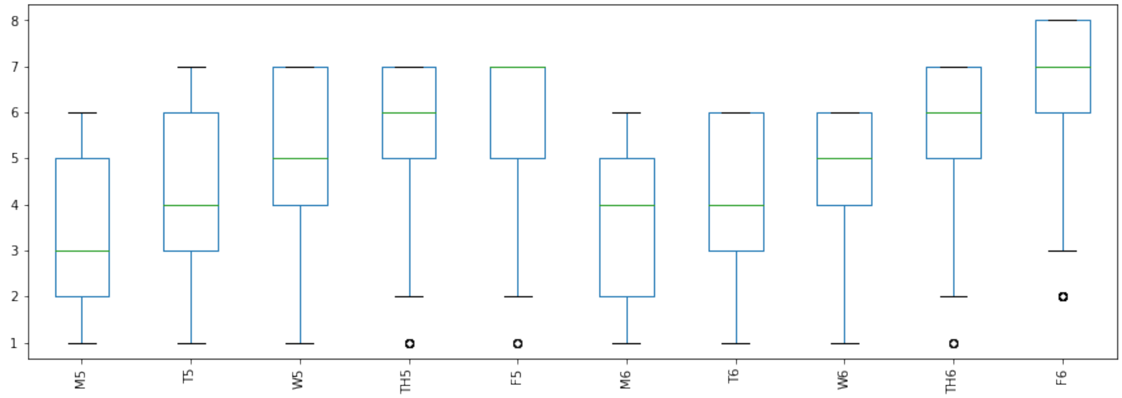
[135]:

```
week_5_6.plot.box(figsize=(15,5))

plt.xticks(rotation='vertical')
```

```
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

[135]:

```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10]),
 <a list of 10 Text major ticklabel objects>)
```



0.1.1 check the sigma for upto +- 3

[ ]:

```
print("value for sigma -3 and 3")
print("for -3 ,           for 3")
for i in week_5_6.columns:
  sigmam_3 = (week_5_6[i].mean() - 3 * week_5_6[i].std())
  sigmap_3 =  (week_5_6[i].mean() + 3 * week_5_6[i].std())
  print("For day ",i)
  print(sigmam_3,sigmap_3)
```

```
value for sigma -3 and 3
for -3 ,           for 3
For day  M5
-1.7742504527245648 8.674250452724564
For day  T5
-0.9843729101323904 9.63037291013239
For day  W5
0.14187504159990993 9.99212495840009
For day  TH5
1.3843675891758984 9.971632410824101
For day  F5
2.5102927425514263 9.769707257448573
For day  M6
-1.6139814806879325 8.754396621103073
For day  T6
-0.34634712563324577 8.873307909946972
For day  W6
0.9742080692550896 8.766306006632304
For day  TH6
1.5880549838367246 9.85327154677552
For day  F6
2.108223467749596 11.039157774637737
```

[ ]:

```
print("value for sigma -2 and 2")
print("for -2 ,           for 2")
for i in week_5_6.columns:
  sigmam_3 = (week_5_6[i].mean() - 2 * week_5_6[i].std())
  sigmap_3 =  (week_5_6[i].mean() + 2 * week_5_6[i].std())
  print("For day ",i)
  print(sigmam_3,sigmap_3)
```

```
value for sigma -2 and 2
for -2 ,           for 2
For day  M5
-0.03283363514970983 6.93283363514971
For day  T5
0.7847513932450734 7.861248606754927
For day  W5
1.783583361066607 8.350416638933392
For day  TH5
2.8155783927839324 8.540421607216068
For day  F5
3.7201951617009508 8.559804838299048
For day  M6
0.11408153627723516 7.026333604137905
For day  T6
1.1902620469634573 7.336698737350268
For day  W6
2.2728910588179585 7.467623017069435
For day  TH6
2.965591077659857 8.475735452952387
For day  F6
3.5967125188976197 9.550668723489713
```

[ ]:

```
print("value for sigma -1 and 1")
print("for -1 ,           for 1")
for i in week_5_6.columns:
  sigmam_3 = (week_5_6[i].mean() - 1 * week_5_6[i].std())
  sigmap_3 =  (week_5_6[i].mean() + 1 * week_5_6[i].std())
  print("For day ",i)
  print(sigmam_3,sigmap_3)
```

```
value for sigma -1 and 1
for -1 ,           for 1
For day  M5
1.7085831824251452 5.191416817574855
For day  T5
2.553875696622537 6.092124303377464
For day  W5
3.4252916805333036 6.708708319466696
For day  TH5
```

```
4.246789196391966 7.1092108036080335
For day  F5
4.930097580850475 7.349902419149524
For day  M6
1.8421445532424028 5.298270587172738
For day  T6
2.7268712195601603 5.8000895647535655
For day  W6
3.5715740483808274 6.168940027506566
For day  TH6
4.3431271714829895 7.0981993591292545
For day  F6
5.085201570045643 8.06217967234169
```

0.1.2 imputing the missing values

[ ]:

```
#!pip install missingpy
from missingpy import KNNImputer
imputer = KNNImputer(n_neighbors = 2, weights ="uniform")
x = imputer.fit_transform(week_5_6)
```

```
/usr/local/lib/python3.6/dist-packages/missingpy/pairwise_external.py:135:
FutureWarning: 'warn_on_dtype' is deprecated in version 0.21 and will be removed
in 0.23. Don't set `warn_on_dtype` to remove this warning.
  warn_on_dtype=warn_on_dtype, estimator=estimator)
/usr/local/lib/python3.6/dist-packages/missingpy/pairwise_external.py:138:
FutureWarning: 'warn_on_dtype' is deprecated in version 0.21 and will be removed
in 0.23. Don't set `warn_on_dtype` to remove this warning.
  warn_on_dtype=warn_on_dtype, estimator=estimator)
```

[ ]:

```
week_5_6.columns
```

[ ]:

```
Index(['M5', 'T5', 'W5', 'TH5', 'F5', 'M6', 'T6', 'W6', 'TH6', 'F6'],
dtype='object')
```

[ ]:

```
imputed = pd.DataFrame(data = x , columns = ['M5', 'T5', 'W5', 'TH5', 'F5',␣
 ↪'M6', 'T6', 'W6', 'TH6', 'F6'] )
```

[100]:

```
for i in imputed.columns:
  imputed[i] = imputed[i].astype(int)
```

[101]:

```
imputed
```

[101]:

```
     M5  T5  W5  TH5  F5  M6  T6  W6  TH6  F6
0     5   6   7    7   7   4   5   6    7   8
1     3   4   5    6   7   4   5   6    7   8
2     3   4   5    6   7   5   5   6    7   8
```

```
3     4   5   6    7   7   2   3   4    5   6
4     3   4   5    6   7   4   5   6    7   8
..   ..  ..  ..  …   ..  ..  ..  ..  …  ..
995   1   1   2    3   4   6   6   6    7   7
996   6   7   7    7   7   4   4   4    4   5
997   5   5   5    6   6   6   6   6    6   7
998   4   4   4    5   6   3   3   3    3   3
999   1   2   2    3   4   4   5   4    6   6

[1000 rows x 10 columns]
```

[103]:

```
from sklearn import metrics
print(metrics.mean_absolute_error(df,imputed))
```

```
0.46320000000000006
```

[104]:

```
print(metrics.mean_squared_error(df,imputed))
```

```
2.5957999999999997
```

[105]:

```
print(np.sqrt(metrics.mean_squared_error(df,imputed)))
```

```
1.6111486585663037
```

[106]:

```
from sklearn.metrics import r2_score
r2_score(df, imputed)
```

[106]:

```
0.5767121174597585
```

[109]:

```
data_imputed = week_1_4.join(imputed)
data_imputed
```

[109]:

```
     Student  M1  T1  W1  TH1  F1  M2  T2  …  W5  TH5  F5  M6  T6  W6  TH6  F6
0          1   4   5   5    6   7   1   2  …   7    7   7   4   5   6    7   8
1          2   1   2   3    4   5   5   5  …   5    6   7   4   5   6    7   8
2          3   3   4   5    6   7   2   3  …   5    6   7   5   5   6    7   8
3          4   1   2   3    4   5   1   2  …   6    7   7   2   3   4    5   6
4          5   2   3   4    5   6   5   5  …   5    6   7   4   5   6    7   8
..       …   ..  ..  ..  …   ..  ..  ..  …  .. …   ..  ..  ..  ..  …  ..
995      996   2   3   4    5   5   1   2  …   2    3   4   6   6   6    7   7
996      997   5   5   5    5   7   2   2  …   7    7   7   4   4   4    4   5
997      998   1   1   1    2   3   4   4  …   5    6   6   6   6   6    6   7
998      999   1   1   2    2   3   1   1  …   4    5   6   3   3   3    3   3
999     1000   2   2   2    2   3   2   2  …   2    3   4   4   5   4    6   6

[1000 rows x 31 columns]
```

[112]:

```
week_6_imputed = imputed[imputed.columns[5:10]]
week_6 = week_5_6[week_5_6.columns[5:10]]
```

0.1.3 comparing the predicted values

[113]:

```
def accuracy(y_test, y_preds):


    total_correct = 0
    for i in range(len(y_test)):
        if int(y_test[i]) == int(y_preds[i]):
            total_correct += 1
    acc = total_correct/len(y_test)
    return acc
```

[115]:

```
data_p = data_imputed.values
days = ["Mon","Tues","Wed","Thus","Fri"]
week_5_w4 = []
print("Model accuracy of week 4 data with previous weeks 5 data : ")
for i in range(5):
  X =data_p[:,i+15]
  y =  data_p[:,i+20]
  acc = accuracy(y, X)
  week_5_w4.append(acc)
  print(days[i], acc*100)
```

```
Model accuracy of week 4 data with previous weeks 5 data :
Mon 39.1
Tues 18.5
Wed 18.2
Thus 23.200000000000003
Fri 30.3
```

[117]:

```
data_p = data_imputed.values
days = ["Mon","Tues","Wed","Thus","Fri"]
week_5_w6 = []
print("Model accuracy of Imputed missing data with previous weeks data : ")
for i in range(5):
  X =data_p[:,i+20]
  y =  data_p[:,i+25]
  acc = accuracy(y, X)
  week_5_w6.append(acc)
  print(days[i], acc*100)
```

```
Model accuracy of Imputed missing data with previous weeks data :
Mon 42.8
Tues 17.0
```

```
Wed 17.9
Thus 16.2
Fri 29.099999999999998
```

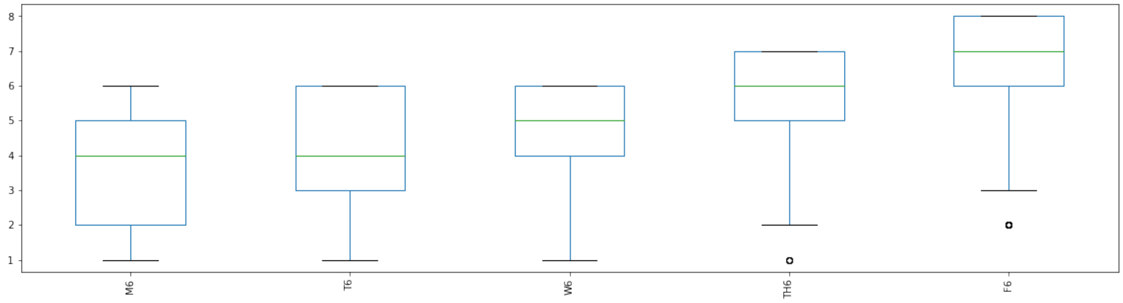
[122]:
```
from statistics import mean

per = (mean(week_5_w4)/mean(week_5_w6))*100

print("The accuracy percenntage of our imputed value is :",per)
```

```
The accuracy percenntage of our imputed value is : 61.951219512195124
```

0.1.4 Plotting the missing data and imputed data

[124]:
```
from matplotlib import pyplot as plt
week_6_imputed.plot.box(figsize=(20,5))

plt.xticks(rotation='vertical')
```

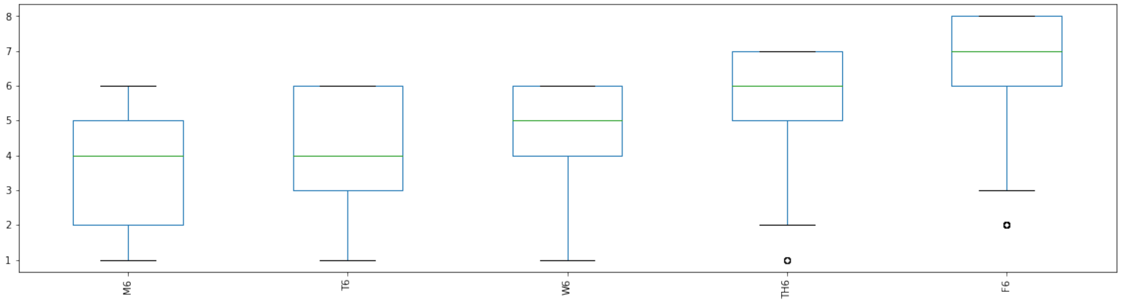
[124]:
```
(array([1, 2, 3, 4, 5]), <a list of 5 Text major ticklabel objects>)
```



[125]:
```
week_6.plot.box(figsize=(20,5))

plt.xticks(rotation='vertical')
```

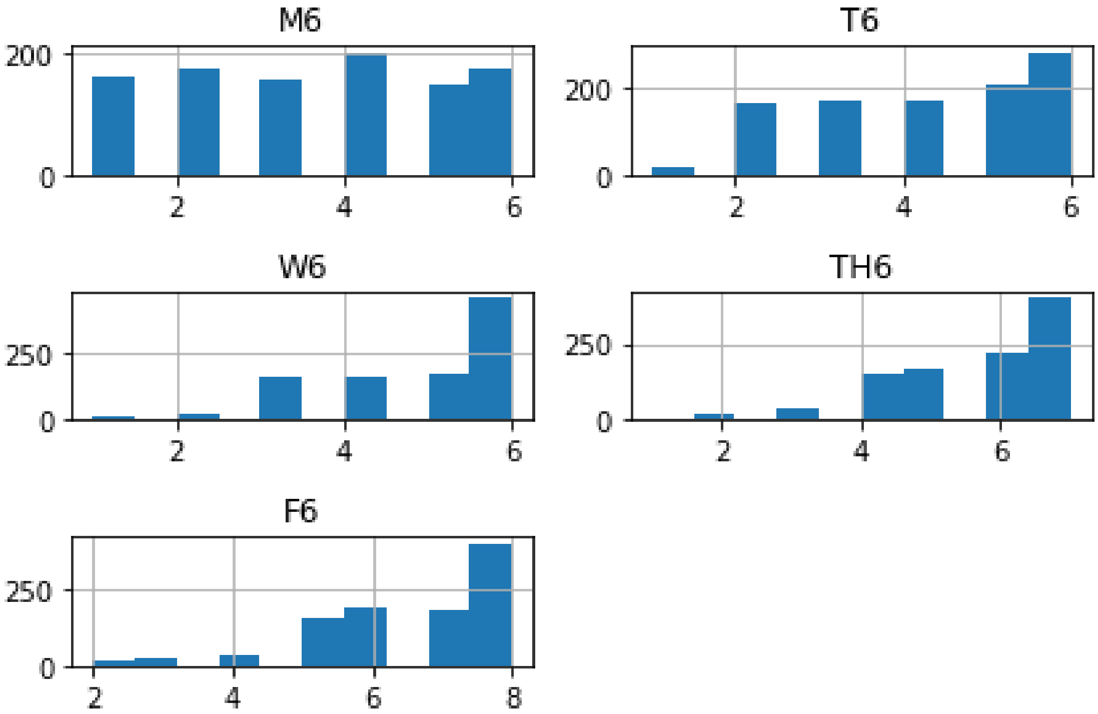
```
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

[125]:
```
(array([1, 2, 3, 4, 5]), <a list of 5 Text major ticklabel objects>)
```



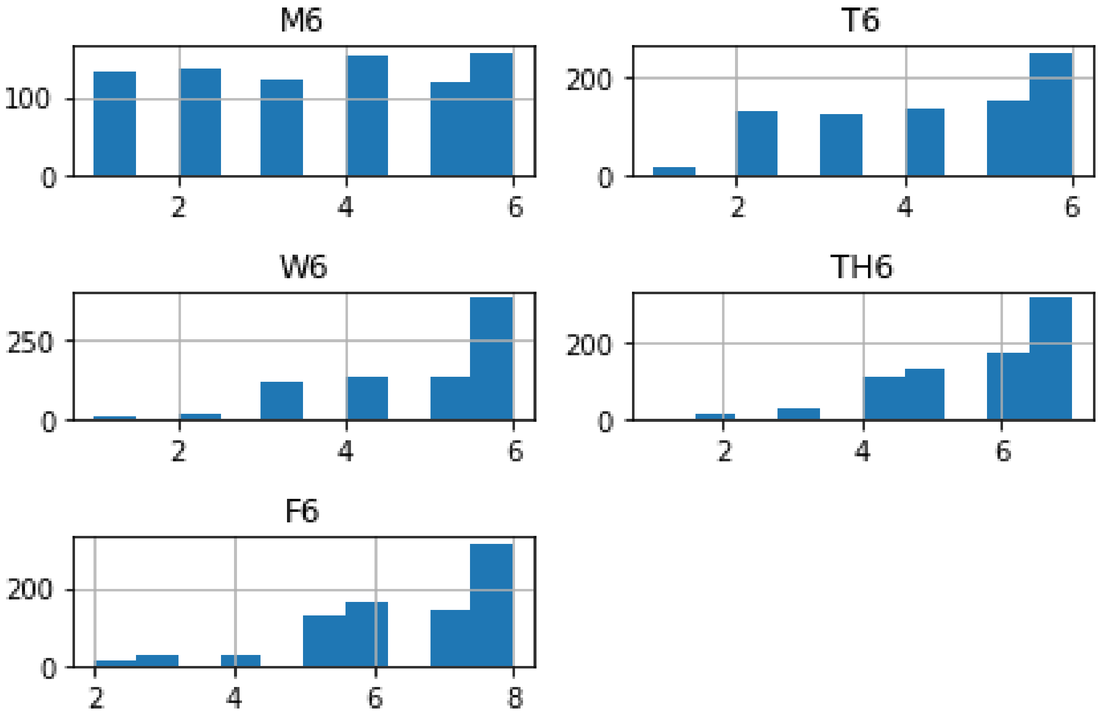
[126]:

```
week_6_imputed.hist()

plt.tight_layout()
```



[127]:

```
week_6.hist()

plt.tight_layout()
```



```
[143]: print(week_6.skew(),"\n",week_6_imputed.skew())
```

```
M6     -0.039754
T6     -0.352240
W6     -0.838248
TH6    -0.917327
F6     -0.904050
dtype: float64
 M6     -0.002250
T6     -0.298525
W6     -0.775548
TH6    -0.889151
F6     -0.949342
dtype: float64
```

```
[141]: print(week_6.kurtosis(),"\n", week_6_imputed.kurtosis())
```

```
M6     -1.275539
T6     -1.185389
W6     -0.331043
TH6     0.137185
F6      0.256055
dtype: float64
 M6     -1.227697
```

```
T6    -1.195150
W6    -0.476038
TH6    0.067626
F6     0.344653
dtype: float64
```

[132]:

```
print("value for sigma -3 and 3")
print("for -3 ,           for 3")
for i in week_6_imputed.columns:
  sigmam_3 = (week_6_imputed[i].mean() - 3 * week_6_imputed[i].std())
  sigmap_3 =  (week_6_imputed[i].mean() + 3 * week_6_imputed[i].std())
  print("For day ",i)
  print(sigmam_3,sigmap_3)
```

```
value for sigma -3 and 3
for -3 ,           for 3
For day  M6
-1.5590241151102835 8.589024115110284
For day  T6
-0.29402498399668797 8.724024983996689
For day  W6
0.9463062200425099 8.73569377995749
For day  TH6
1.615947948278019 9.824052051721981
For day  F6
2.1318820959219327 11.078117904078068
```

[133]:

```
print("value for sigma -2 and 2")
print("for -2 ,           for 2")
for i in week_6_imputed.columns:
  sigmam_3 = (week_6_imputed[i].mean() - 2 * week_6_imputed[i].std())
  sigmap_3 =  (week_6_imputed[i].mean() + 2 * week_6_imputed[i].std())
  print("For day ",i)
  print(sigmam_3,sigmap_3)
```

```
value for sigma -2 and 2
for -2 ,           for 2
For day  M6
0.13231725659314453 6.897682743406856
For day  T6
1.208983344002208 7.221016655997792
For day  W6
2.24453748002834 7.43746251997166
For day  TH6
2.9839652988520124 8.456034701147987
For day  F6
3.6229213972812886 9.587078602718712
```

[134]:
```
print("value for sigma -1 and 1")
print("for -1 ,           for 1")
for i in week_6_imputed.columns:
  sigmam_3 = (week_6_imputed[i].mean() - 1 * week_6_imputed[i].std())
  sigmap_3 =  (week_6_imputed[i].mean() + 1 * week_6_imputed[i].std())
  print("For day ",i)
  print(sigmam_3,sigmap_3)
```

```
value for sigma -1 and 1
for -1 ,           for 1
For day  M6
1.8236586282965723 5.206341371703428
For day  T6
2.711991672001104 5.718008327998896
For day  W6
3.54276874001417 6.139231259985831
For day  TH6
4.351982649426006 7.088017350573994
For day  F6
5.1139606986406445 8.096039301359356
```

[144]:
```
dt = data_imputed.transpose()
```

[145]:
```
dt.to_csv("/content/drive/MyDrive/dowell/week 6/week_6transpose.csv")
```

[204]:
```
data = pd.read_csv("/content/drive/MyDrive/dowell/week 6/week_6transpose.csv",␣
 ↪header =1)
data =data.rename(columns = {"Student":"Days"})
print(data.shape)
data.tail()
```

```
(30, 1001)
```

[204]:
```
   Days  1  2  3  4  5  6  7  8  …  992  993  994  995  996  997  998  999
1000
25   M6  4  4  5  2  4  4  4  5  …    4    6    6    6    6    4    6    3
4
26   T6  5  5  5  3  5  5  5  6  …    4    6    6    6    6    4    6    3
5
27   W6  6  6  6  4  6  6  6  6  …    5    6    6    6    6    4    6    3
4
28  TH6  7  7  7  5  7  7  7  7  …    6    7    6    6    7    4    6    3
6
29   F6  8  8  8  6  8  8  8  8  …    7    8    6    8    7    5    7    3
6

[5 rows x 1001 columns]
```

[166]:

```
cols = data.columns[1:1001]
```

[183]:

```
dt = data.drop("Days",axis = 1)
print(dt.shape)
dt.head(1)
```

```
(30, 1000)
```

[183]:

```
   1  2  3  4  5  6  7  8  9  …  992  993  994  995  996  997  998  999  1000
0  4  1  3  1  2  5  5  3  5  …    2    3    1    4    2    5    1    1     2

[1 rows x 1000 columns]
```

[190]:

```
dt.index = data["Days"]
data_v = dt.values

#creating the train and validation set
train = data_v[:25]
valid = data_v[25:]
```

[191]:

```
valid.shape
```

[191]:

```
(5, 1000)
```

[192]:

```
train.shape
```

[192]:

```
(25, 1000)
```

[193]:

```
#fit the model
from statsmodels.tsa.vector_ar.var_model import VAR
import warnings
warnings.filterwarnings('ignore')

model = VAR(endog=train )
model_fit = model.fit(trend='nc')
```

[194]:

```
# make prediction on validation
prediction = model_fit.forecast(model_fit.y, steps=len(valid))
```

[195]:

```
prediction.shape
```

[195]:

```
(5, 1000)
```

[196]:

```
#converting predictions to dataframe
pred = pd.DataFrame(index=range(0,len(prediction)),columns=[cols])
for j in range(0,1000):
    for i in range(0, len(prediction)):
        pred.iloc[i][j] = prediction[i][j]
```

[197]:

```
pred.head()
```

[197]:

```
         1        2        3        4  …      997      998      999     1000
0  7.29956  7.68198  7.43337  7.44546  …  7.72453  5.89051  5.61251  4.41478
1  6.32574  6.87177  6.46078  6.46348  …  7.73061  5.63818  4.80548  3.91023
2  4.72537  5.62561  4.97731  4.70347  …  6.15996  5.31395  3.91822  3.22569
3   4.9592  5.75397  5.32909  4.70426  …   5.5217  4.83564  3.73972  3.22113
4  5.01583  5.92547  5.64755  4.74985  …  5.07441  4.73304  3.97753   3.0659

[5 rows x 1000 columns]
```

[198]:

```
#make final predictions
model = VAR(endog=data_v)
model_fit = model.fit()
yhat = model_fit.forecast(model_fit.y, steps=5)
print(yhat)
```

```
[[ 8.1026941   8.18344637  8.48201782 …  7.22215415  3.9832547
   6.29714064]
 [ 7.37188558  8.43971268  7.93103154 …  7.92024671  3.71573283
   6.49169632]
 [ 8.79796847  8.72638213  8.85289911 …  8.28210717  3.91294785
   7.08710869]
 [ 8.58195018  9.29076713  8.61577209 …  8.83798843  3.74828038
   7.45972641]
 [ 9.94668081 10.0915098   9.90502513 …  9.51566193  3.98674742
   8.19905255]]
```

[205]:

```
predicted_data = pd.DataFrame(yhat)
```

[212]:

```
##transpose the predicted Data
t_p = predicted_data.transpose()

print(t_p.shape)
```

```
(1000, 5)
```

[213]:

```
##converting the datatype int float
for i in t_p.columns:
  t_p[i] = t_p[i].astype(int)  # converting the datatype into float

t_p = t_p.rename(columns = {0:"M7",1:"T7",2:"W7",3:"TH7",4:"F7"})

td = t_p[["M7","T7","W7","TH7","F7"]]
td.head()
```

[213]:

```
   M7  T7  W7  TH7  F7
0   8   7   8    8   9
1   8   8   8    9  10
2   8   7   8    8   9
3   5   5   6    6   7
4   8   8   8    8   9
```

[216]:

```
td.to_csv("/content/drive/MyDrive/dowell/week 6/week7.csv")
```

[214]:

```
week_1_7 = data_imputed.join(td)
week_1_7.head()
```

[214]:

```
   Student  M1  T1  W1  TH1  F1  M2  T2  …  W6  TH6  F6  M7  T7  W7  TH7  F7
0        1   4   5   5    6   7   1   2  …   6    7   8   8   7   8    8   9
1        2   1   2   3    4   5   5   5  …   6    7   8   8   8   8    9  10
2        3   3   4   5    6   7   2   3  …   6    7   8   8   7   8    8   9
3        4   1   2   3    4   5   1   2  …   4    5   6   5   5   6    6   7
4        5   2   3   4    5   6   5   5  …   6    7   8   8   8   8    8   9

[5 rows x 36 columns]
```

[215]:

```
dz = week_1_7
dz.to_csv("/content/drive/MyDrive/dowell/week 6/week1_7.csv")
```

[223]:

```
data_p = week_1_7.values
days = ["Mon","Tues","Wed","Thus","Fri"]
print("Model accuracy of week 7 data with previous weeks data : ")
week_6_w7 = []
for i in range(5):
  X =data_p[:,i+26]
  y =  data_p[:,i+31]
  acc = accuracy(y, X)
  week_6_w7.append(acc)
  print(days[i], acc*100)
```

```
Model accuracy of week 7 data with previous weeks data :
Mon 5.0
Tues 6.9
Wed 5.7
Thus 26.1
Fri 24.0
```

[224]:

```
data_p = week_1_7.values
days = ["Mon","Tues","Wed","Thus","Fri"]
print("Model accuracy of week 7 data with previous weeks data : ")
week_6_w5 = []
for i in range(5):
  X =data_p[:,i+21]
  y =  data_p[:,i+26]
  acc = accuracy(y, X)
  week_6_w5.append(acc)
  print(days[i], acc*100)
```

```
Model accuracy of week 7 data with previous weeks data :
Mon 17.0
Tues 17.9
Wed 16.2
Thus 29.099999999999998
Fri 16.1
```

[226]:

```
per = (mean(week_6_w7)/mean(week_6_w5))*100

print("The accuracy percenntage of our imputed value is :",per)
```

```
The accuracy percenntage of our imputed value is : 70.30114226375909
```

[230]:

```
# MSE
from sklearn.metrics import mean_squared_error
from numpy import asarray as arr
mse = mean_squared_error(valid, yhat)
print(mse)
```

```
5.650149360659398
```

[231]:

```
td.skew()
```

[231]:

```
M7    -0.810745
T7    -0.686738
W7    -0.758557
TH7   -0.727454
F7    -0.807024
dtype: float64
```

[233]:

```
td.kurtosis()
```

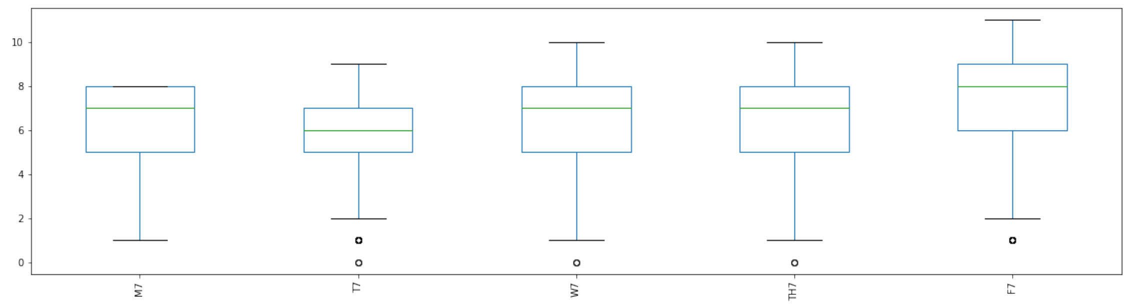
[233]:

```
M7     0.018426
T7    -0.093722
W7    -0.009720
TH7   -0.125536
F7     0.043175
dtype: float64
```

[234]:

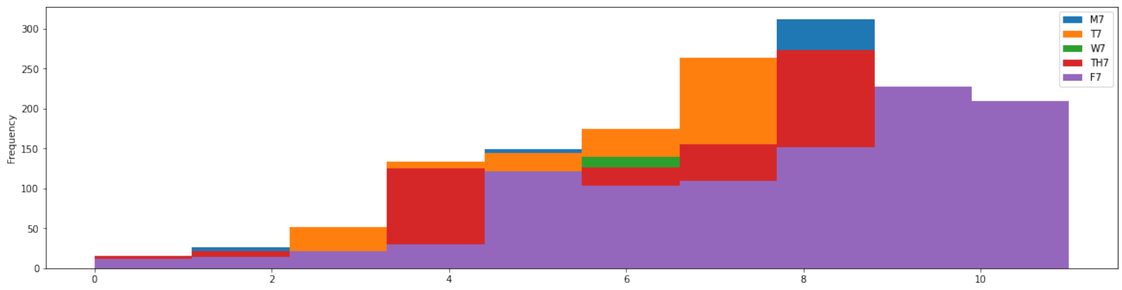
```
td.plot.box(figsize=(20,5))

plt.xticks(rotation='vertical')
```

```
[234]: (array([1, 2, 3, 4, 5]), <a list of 5 Text major ticklabel objects>)
```

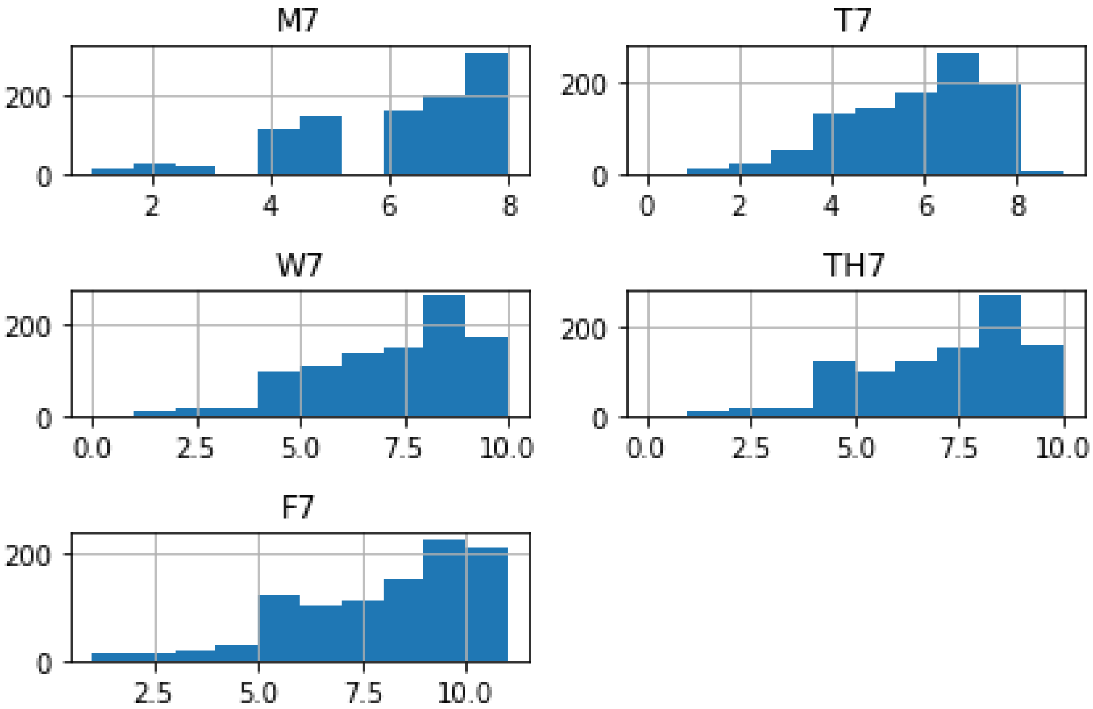


```
[237]: td.plot.hist(figsize=(20,5))

       plt.xticks(rotation='horizontal')
```

```
[237]: (array([-2.,  0.,  2.,  4.,  6.,  8., 10., 12.]),
         <a list of 8 Text major ticklabel objects>)
```



```
[239]: td.hist()

       plt.tight_layout()
```


M7
200
0
2
4
6
8
T7
200
0
0
2
4
6
8
W7
200
0
0.0
2.5
5.0
7.5
10.0
TH7
200
0
0.0
2.5
5.0
7.5
10.0
F7
200
0
2.5
5.0
7.5
10.0