Lab_6_Notebook

November 23, 2021

[1]:
```
#!conda install -y anaconda graphviz
!conda install -y graphviz python-graphviz    #(Already downloaded dont need to␣
↪do again)
#!pip install graphviz
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.7.10
  latest version: 4.7.12

Please update conda by running

    $ conda update -n base conda



## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - graphviz
    - python-graphviz


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2019.9.11  |       hecc5488_0         144 KB  conda-forge
    cairo-1.16.0               |    hfb77d84_1002         1.5 MB  conda-forge
    certifi-2019.9.11          |           py37_0         147 KB  conda-forge
    expat-2.2.5                |    he1b5a44_1004         191 KB  conda-forge
    fontconfig-2.13.1          |    h86ecdb6_1001         340 KB  conda-forge
    fribidi-1.0.5              |    h516909a_1002         112 KB  conda-forge
```

```
    gettext-0.19.8.1           |  hc5be6a0_1002         3.6 MB  conda-forge
    glib-2.58.3                |  h6f030ca_1002         3.3 MB  conda-forge
    graphite2-1.3.13           |  hf484d3e_1000         109 KB  conda-forge
    graphviz-2.40.1            |     h0f2764d_1         6.4 MB  conda-forge
    harfbuzz-2.4.0             |     h9f30f68_3         1.5 MB  conda-forge
    libtool-2.4.6              |  h14c3975_1002         512 KB  conda-forge
    libuuid-2.32.1             |  h14c3975_1000          26 KB  conda-forge
    libxcb-1.13                |  h14c3975_1002         396 KB  conda-forge
    pango-1.42.4               |     ha030887_1         517 KB  conda-forge
    pcre-8.43                  |     he1b5a44_0         257 KB  conda-forge
    pixman-0.38.0              |  h516909a_1003         594 KB  conda-forge
    pthread-stubs-0.4          |  h14c3975_1001           5 KB  conda-forge
    python-graphviz-0.13       |           py_0          18 KB  conda-forge
    xorg-kbproto-1.0.7         |  h14c3975_1002          26 KB  conda-forge
    xorg-libice-1.0.10         |     h516909a_0          57 KB  conda-forge
    xorg-libsm-1.2.3           |  h84519dc_1000          25 KB  conda-forge
    xorg-libx11-1.6.9          |     h516909a_0         918 KB  conda-forge
    xorg-libxau-1.0.9          |     h14c3975_0          13 KB  conda-forge
    xorg-libxdmcp-1.1.3        |     h516909a_0          18 KB  conda-forge
    xorg-libxext-1.3.4         |     h516909a_0          51 KB  conda-forge
    xorg-libxpm-3.5.12         |  h516909a_1002          63 KB  conda-forge
    xorg-libxrender-0.9.10     |  h516909a_1002          31 KB  conda-forge
    xorg-libxt-1.1.5           |  h516909a_1003         367 KB  conda-forge
    xorg-renderproto-0.11.1    |  h14c3975_1002           8 KB  conda-forge
    xorg-xextproto-7.3.0       |  h14c3975_1002          27 KB  conda-forge
    xorg-xproto-7.0.31         |  h14c3975_1007          72 KB  conda-forge
    ------------------------------------------------------------
                                           Total:        21.3 MB

The following NEW packages will be INSTALLED:

  cairo              conda-forge/linux-64::cairo-1.16.0-hfb77d84_1002
  expat              conda-forge/linux-64::expat-2.2.5-he1b5a44_1004
  fontconfig         conda-forge/linux-64::fontconfig-2.13.1-h86ecdb6_1001
  fribidi            conda-forge/linux-64::fribidi-1.0.5-h516909a_1002
  gettext            conda-forge/linux-64::gettext-0.19.8.1-hc5be6a0_1002
  glib               conda-forge/linux-64::glib-2.58.3-h6f030ca_1002
  graphite2          conda-forge/linux-64::graphite2-1.3.13-hf484d3e_1000
  graphviz           conda-forge/linux-64::graphviz-2.40.1-h0f2764d_1
  harfbuzz           conda-forge/linux-64::harfbuzz-2.4.0-h9f30f68_3
  libtool            conda-forge/linux-64::libtool-2.4.6-h14c3975_1002
  libuuid            conda-forge/linux-64::libuuid-2.32.1-h14c3975_1000
  libxcb             conda-forge/linux-64::libxcb-1.13-h14c3975_1002
  pango              conda-forge/linux-64::pango-1.42.4-ha030887_1
  pcre               conda-forge/linux-64::pcre-8.43-he1b5a44_0
  pixman             conda-forge/linux-64::pixman-0.38.0-h516909a_1003
  pthread-stubs      conda-forge/linux-64::pthread-stubs-0.4-h14c3975_1001
  python-graphviz    conda-forge/noarch::python-graphviz-0.13-py_0
```

```
  xorg-kbproto       conda-forge/linux-64::xorg-kbproto-1.0.7-h14c3975_1002
  xorg-libice        conda-forge/linux-64::xorg-libice-1.0.10-h516909a_0
  xorg-libsm         conda-forge/linux-64::xorg-libsm-1.2.3-h84519dc_1000
  xorg-libx11        conda-forge/linux-64::xorg-libx11-1.6.9-h516909a_0
  xorg-libxau        conda-forge/linux-64::xorg-libxau-1.0.9-h14c3975_0
  xorg-libxdmcp      conda-forge/linux-64::xorg-libxdmcp-1.1.3-h516909a_0
  xorg-libxext       conda-forge/linux-64::xorg-libxext-1.3.4-h516909a_0
  xorg-libxpm        conda-forge/linux-64::xorg-libxpm-3.5.12-h516909a_1002
  xorg-libxrender    conda-forge/linux-64::xorg-libxrender-0.9.10-h516909a_1002
  xorg-libxt         conda-forge/linux-64::xorg-libxt-1.1.5-h516909a_1003
  xorg-renderproto   conda-forge/linux-64::xorg-renderproto-0.11.1-h14c3975_1002
  xorg-xextproto     conda-forge/linux-64::xorg-xextproto-7.3.0-h14c3975_1002
  xorg-xproto        conda-forge/linux-64::xorg-xproto-7.0.31-h14c3975_1007

The following packages will be UPDATED:

  ca-certificates                  2019.6.16-hecc5488_0 -->
2019.9.11-hecc5488_0
  certifi                              2019.6.16-py37_1 --> 2019.9.11-py37_0



Downloading and Extracting Packages
cairo-1.16.0         | 1.5 MB    | ##################################### | 100%
xorg-libx11-1.6.9    | 918 KB    | ##################################### | 100%
fribidi-1.0.5        | 112 KB    | ##################################### | 100%
xorg-libxau-1.0.9    | 13 KB     | ##################################### | 100%
xorg-libxext-1.3.4   | 51 KB     | ##################################### | 100%
libtool-2.4.6        | 512 KB    | ##################################### | 100%
gettext-0.19.8.1     | 3.6 MB    | ##################################### | 100%
harfbuzz-2.4.0       | 1.5 MB    | ##################################### | 100%
xorg-libxrender-0.9. | 31 KB     | ##################################### | 100%
fontconfig-2.13.1    | 340 KB    | ##################################### | 100%
xorg-libxt-1.1.5     | 367 KB    | ##################################### | 100%
xorg-xproto-7.0.31   | 72 KB     | ##################################### | 100%
xorg-renderproto-0.1 | 8 KB      | ##################################### | 100%
graphite2-1.3.13     | 109 KB    | ##################################### | 100%
expat-2.2.5          | 191 KB    | ##################################### | 100%
xorg-libxdmcp-1.1.3  | 18 KB     | ##################################### | 100%
certifi-2019.9.11    | 147 KB    | ##################################### | 100%
xorg-libxpm-3.5.12   | 63 KB     | ##################################### | 100%
xorg-libice-1.0.10   | 57 KB     | ##################################### | 100%
ca-certificates-2019 | 144 KB    | ##################################### | 100%
pixman-0.38.0        | 594 KB    | ##################################### | 100%
xorg-libsm-1.2.3     | 25 KB     | ##################################### | 100%
libxcb-1.13          | 396 KB    | ##################################### | 100%
libuuid-2.32.1       | 26 KB     | ##################################### | 100%
xorg-xextproto-7.3.0 | 27 KB     | ##################################### | 100%
```

```
graphviz-2.40.1       | 6.4 MB    | ##################################### | 100%
xorg-kbproto-1.0.7    | 26 KB     | ##################################### | 100%
glib-2.58.3           | 3.3 MB    | ##################################### | 100%
pcre-8.43             | 257 KB    | ##################################### | 100%
pango-1.42.4          | 517 KB    | ##################################### | 100%
pthread-stubs-0.4     | 5 KB      | ##################################### | 100%
python-graphviz-0.13 | 18 KB     | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

[2]:

```
import requests
import pandas as pd
r = requests.post('https://cdsw00.geo.sciclone.wm.edu/api/altus-ds-1/models/
↪call-model',
                  data = '{"accessKey":"m149rzguxkf56i4pnqsulvkmfx43zu5t",␣
↪"request":{"timestamp_start":"9/1/2019 0:00", "timestamp_end": "9/2/2019 23:
↪59"}}',
                  headers = {"Content-Type" : "application/json", "host":"cdsw.
↪geo.sciclone.wm.edu"}, verify=False)
dta = pd.read_json(r.json()["response"], orient="index")
```

```
/opt/conda/lib/python3.7/site-packages/urllib3/connectionpool.py:851:
InsecureRequestWarning: Unverified HTTPS request is being made. Adding
certificate verification is strongly advised. See:
https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
```

[3]:

```
dta
```

[3]:

```
      ApproachCount      CNN1      CNN10      CNN11     CNN12      CNN2  \
0                 0  0.144993   0.000000   0.000000  0.502654  0.580529
1                 0  0.448995   0.000000   0.000000  0.212248  0.251403
10                0  0.203227   0.222424   0.350840  0.979021  0.695390
100               0  0.942832   0.440697   0.725458  0.402949  0.218727
1000              2  0.891993   3.468451  17.282234  0.346783  0.548471
…               …       …         …          …         …         …
995               5  0.442996   7.835986   0.630775  0.627370  0.657769
996              14  0.519954  13.497248   2.973148  0.893685  0.549669
997               7  0.082457   3.362551  10.371099  0.058254  0.021871
998               2  0.166488   6.831756   6.516534  0.157694  0.491937
999              15  0.480170  11.793698  11.415221  0.961353  0.862517

          CNN3      CNN4      CNN5       CNN6      CNN7      CNN8      CNN9  \
0     0.058905  0.162328  0.660968   0.137908  0.618765  0.749660  0.731326
1     0.294498  0.376598  0.387818   0.386584  0.411739  0.926632  0.376281
10    0.544011  0.777266  0.431196   0.523430  0.493386  0.161756  0.496499
```

```
100     0.455083  0.697918  0.747933   1.241087  0.429827  0.425316  0.952403
1000    9.687640  0.481784  0.256119  10.724550  0.609483  0.199922  0.599720
…            …         …         …          …         …         …         …
995     5.538755  0.841678  0.570293   7.482780  0.338142  0.947615  0.982364
996    13.422923  0.737093  0.969022  10.066145  0.496396  0.710909  0.138578
997     9.204641  0.174609  0.440498   0.807013  0.801403  0.414806  0.430312
998     2.667281  0.067576  0.741823   4.886558  0.044934  0.006720  0.033820
999     7.360820  0.653160  0.984205   1.737072  0.199673  0.459773  0.050901

      FemaleCount  ImgBright  MaleCount  PersonCount           Timestamp  \
0               0   0.459320          0            0 2019-09-01 00:00:00
1               0   0.676425          0            0 2019-09-01 00:01:00
10              1   0.819431          1            2 2019-09-01 00:10:00
100             1   2.011970          1            2 2019-09-01 01:40:00
1000           24  11.197290         21           45 2019-09-01 16:40:00
…             …         …         …           …                   …
995            25   9.182725         22           47 2019-09-01 16:35:00
996            60  14.602948         44          104 2019-09-01 16:36:00
997            27  12.189318         21           48 2019-09-01 16:37:00
998            23  10.805675         22           45 2019-09-01 16:38:00
999            61   9.713843         58          119 2019-09-01 16:39:00

            pdtimes
0     1567296000000
1     1567296060000
10    1567296600000
100   1567302000000
1000  1567356000000
…               …
995   1567355700000
996   1567355760000
997   1567355820000
998   1567355880000
999   1567355940000

[2880 rows x 19 columns]
```

[4]:

```
%matplotlib inline

import seaborn as sns
dta.head()
```

[4]:

```
      ApproachCount      CNN1     CNN10      CNN11     CNN12      CNN2  \
0                 0  0.144993  0.000000   0.000000  0.502654  0.580529
1                 0  0.448995  0.000000   0.000000  0.212248  0.251403
10                0  0.203227  0.222424   0.350840  0.979021  0.695390
100               0  0.942832  0.440697   0.725458  0.402949  0.218727
```

```
1000               2  0.891993  3.468451  17.282234  0.346783  0.548471

          CNN3      CNN4      CNN5       CNN6      CNN7      CNN8      CNN9  \
0     0.058905  0.162328  0.660968   0.137908  0.618765  0.749660  0.731326
1     0.294498  0.376598  0.387818   0.386584  0.411739  0.926632  0.376281
10    0.544011  0.777266  0.431196   0.523430  0.493386  0.161756  0.496499
100   0.455083  0.697918  0.747933   1.241087  0.429827  0.425316  0.952403
1000  9.687640  0.481784  0.256119  10.724550  0.609483  0.199922  0.599720

      FemaleCount  ImgBright  MaleCount  PersonCount           Timestamp  \
0               0   0.459320          0            0 2019-09-01 00:00:00
1               0   0.676425          0            0 2019-09-01 00:01:00
10              1   0.819431          1            2 2019-09-01 00:10:00
100             1   2.011970          1            2 2019-09-01 01:40:00
1000           24  11.197290         21           45 2019-09-01 16:40:00

            pdtimes
0     1567296000000
1     1567296060000
10    1567296600000
100   1567302000000
1000  1567356000000
```

[5]:
```python
dta["Minute"] = dta["Timestamp"].dt.minute + dta["Timestamp"].dt.hour * 60
```

[6]:
```python
corr = dta.corr()
```

[7]:
```python
corr
```

[7]:
```
               ApproachCount      CNN1     CNN10     CNN11     CNN12  \
ApproachCount       1.000000 -0.008915  0.536363  0.541136  0.164596
CNN1               -0.008915  1.000000  0.010655  0.002688 -0.023752
CNN10               0.536363  0.010655  1.000000  0.504219  0.019835
CNN11               0.541136  0.002688  0.504219  1.000000  0.017520
CNN12               0.164596 -0.023752  0.019835  0.017520  1.000000
CNN2                0.006263  0.016893  0.013150  0.021466 -0.020529
CNN3                0.335500  0.009419  0.143755  0.126643 -0.001697
CNN4                0.029295  0.003465  0.032278  0.035860 -0.021399
CNN5               -0.004317 -0.009890 -0.022914 -0.024750  0.000608
CNN6                0.202306  0.000425  0.118759  0.138976 -0.021691
CNN7                0.003509 -0.004339  0.016574 -0.005331 -0.017581
CNN8               -0.003205 -0.011219  0.018547 -0.002281 -0.029429
CNN9               -0.118398 -0.013066  0.010284  0.006159 -0.008414
FemaleCount         0.898190 -0.007060  0.594811  0.582496  0.172478
ImgBright           0.322294 -0.019496  0.196240  0.207290 -0.021435
MaleCount           0.897994 -0.006565  0.602745  0.587624  0.169627
PersonCount         0.901177 -0.006850  0.600605  0.586920  0.171720
```

pdtimes	0.157462	0.003265	0.135678	0.108493	0.017141
Minute	0.311213	-0.001348	0.264170	0.248289	0.030708

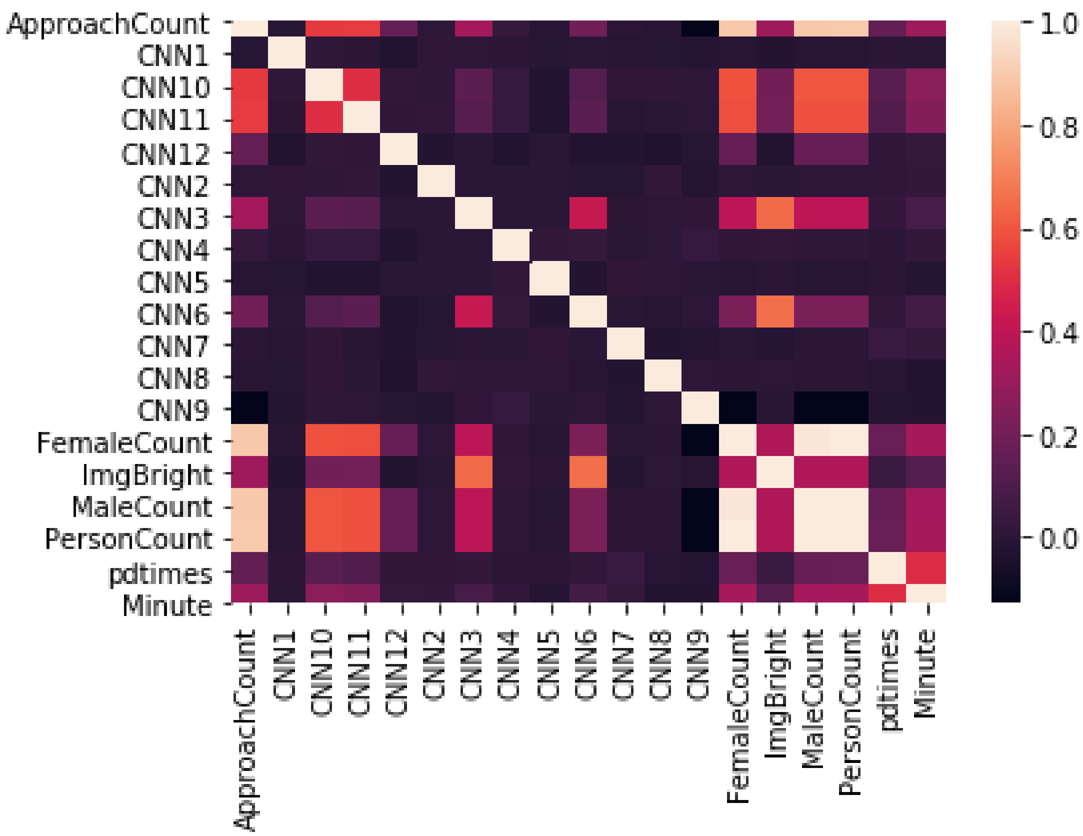
	CNN2	CNN3	CNN4	CNN5	CNN6	CNN7	\
ApproachCount	0.006263	0.335500	0.029295	-0.004317	0.202306	0.003509	
CNN1	0.016893	0.009419	0.003465	-0.009890	0.000425	-0.004339	
CNN10	0.013150	0.143755	0.032278	-0.022914	0.118759	0.016574	
CNN11	0.021466	0.126643	0.035860	-0.024750	0.138976	-0.005331	
CNN12	-0.020529	-0.001697	-0.021399	0.000608	-0.021691	-0.017581	
CNN2	1.000000	-0.002995	-0.002156	-0.000174	-0.010571	-0.010386	
CNN3	-0.002995	1.000000	-0.000067	-0.002712	0.425370	0.002918	
CNN4	-0.002156	-0.000067	1.000000	0.022286	0.029939	-0.001936	
CNN5	-0.000174	-0.002712	0.022286	1.000000	-0.019023	0.015897	
CNN6	-0.010571	0.425370	0.029939	-0.019023	1.000000	-0.001250	
CNN7	-0.010386	0.002918	-0.001936	0.015897	-0.001250	1.000000	
CNN8	0.021175	0.008013	0.007358	0.010213	-0.006595	-0.017387	
CNN9	-0.014637	0.017650	0.033821	-0.000528	0.006978	-0.011995	
FemaleCount	0.007577	0.396368	0.015008	-0.004900	0.225801	0.001215	
ImgBright	-0.002676	0.650763	0.019300	0.003563	0.657819	-0.016134	
MaleCount	0.008798	0.395211	0.012496	-0.008467	0.225924	0.003836	
PersonCount	0.008181	0.397179	0.013871	-0.006605	0.226633	0.002459	
pdtimes	0.016524	0.026377	0.002088	0.005258	0.023150	0.046265	
Minute	0.026530	0.084066	0.024143	-0.013424	0.072015	0.029897	

	CNN8	CNN9	FemaleCount	ImgBright	MaleCount	\
ApproachCount	-0.003205	-0.118398	0.898190	0.322294	0.897994	
CNN1	-0.011219	-0.013066	-0.007060	-0.019496	-0.006565	
CNN10	0.018547	0.010284	0.594811	0.196240	0.602745	
CNN11	-0.002281	0.006159	0.582496	0.207290	0.587624	
CNN12	-0.029429	-0.008414	0.172478	-0.021435	0.169627	
CNN2	0.021175	-0.014637	0.007577	-0.002676	0.008798	
CNN3	0.008013	0.017650	0.396368	0.650763	0.395211	
CNN4	0.007358	0.033821	0.015008	0.019300	0.012496	
CNN5	0.010213	-0.000528	-0.004900	0.003563	-0.008467	
CNN6	-0.006595	0.006978	0.225801	0.657819	0.225924	
CNN7	-0.017387	-0.011995	0.001215	-0.016134	0.003836	
CNN8	1.000000	0.008437	0.002774	0.007615	0.001626	
CNN9	0.008437	1.000000	-0.125456	-0.006127	-0.126101	
FemaleCount	0.002774	-0.125456	1.000000	0.362408	0.986311	
ImgBright	0.007615	-0.006127	0.362408	1.000000	0.360920	
MaleCount	0.001626	-0.126101	0.986311	0.360920	1.000000	
PersonCount	0.002240	-0.126191	0.996951	0.362947	0.996171	
pdtimes	-0.003461	-0.015641	0.173407	0.046790	0.172089	
Minute	-0.024497	-0.018007	0.335983	0.113465	0.334170	

	PersonCount	pdtimes	Minute
ApproachCount	0.901177	0.157462	0.311213

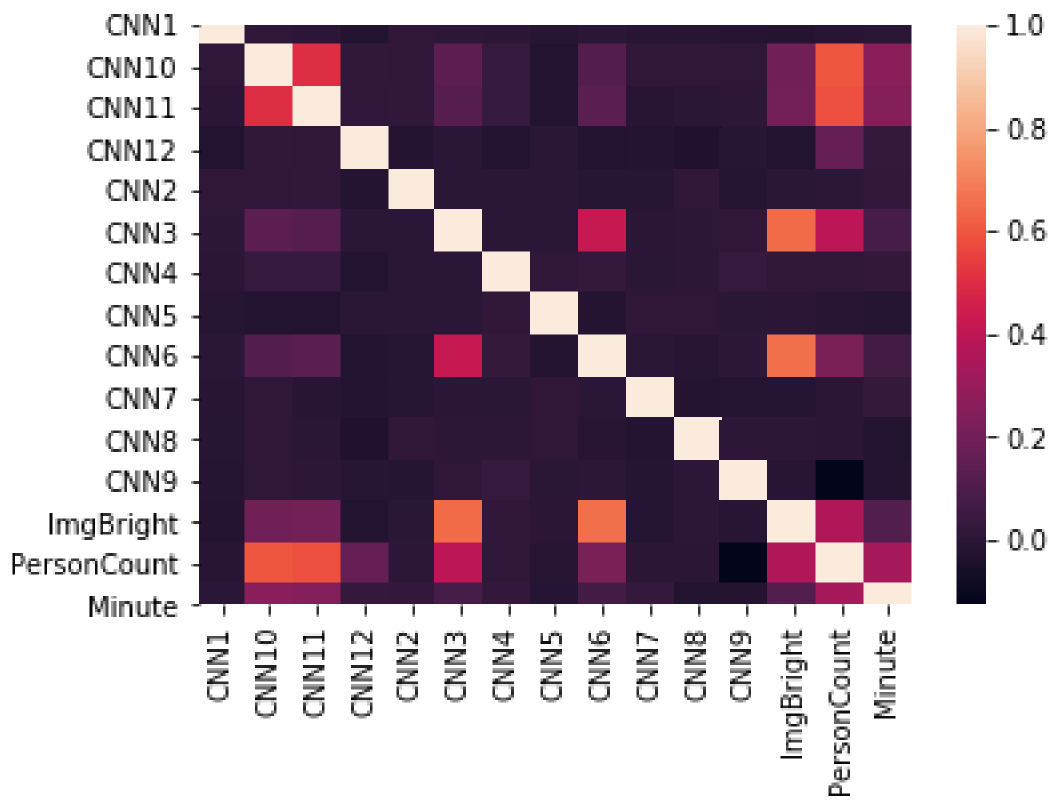
```
CNN1           -0.006850  0.003265 -0.001348
CNN10           0.600605  0.135678  0.264170
CNN11           0.586920  0.108493  0.248289
CNN12           0.171720  0.017141  0.030708
CNN2            0.008181  0.016524  0.026530
CNN3            0.397179  0.026377  0.084066
CNN4            0.013871  0.002088  0.024143
CNN5           -0.006605  0.005258 -0.013424
CNN6            0.226633  0.023150  0.072015
CNN7            0.002459  0.046265  0.029897
CNN8            0.002240 -0.003461 -0.024497
CNN9           -0.126191 -0.015641 -0.018007
FemaleCount     0.996951  0.173407  0.335983
ImgBright       0.362947  0.046790  0.113465
MaleCount       0.996171  0.172089  0.334170
PersonCount     1.000000  0.173378  0.336277
pdtimes         0.173378  1.000000  0.500000
Minute          0.336277  0.500000  1.000000
```

[8]:
```
ax = sns.heatmap(corr)
```



[9]:
```
dta_clean = dta.drop(["ApproachCount", "FemaleCount", "MaleCount", "Timestamp",␣
 ↪"pdtimes"], axis = 1)
```

[10]:
```
ax = sns.heatmap(dta_clean.corr())
```



[11]:
```
dta_clean = dta.drop(["ApproachCount", "FemaleCount", "MaleCount", "Timestamp",␣
 ↪"pdtimes"], axis = 1)
y = dta_clean.pop('PersonCount')
X = dta_clean

from sklearn import preprocessing
scalingModel = preprocessing.StandardScaler().fit(X)
X_scaled = scalingModel.transform(X)

print('Features for Observation 996 before Scaling;')
print(X.loc[996])

print('Features for Observation 996 after Scaling;')
print(X_scaled[996])

X_example = scalingModel.transform(X.iloc[996].values.reshape(1,-1))
```

```
Features for Observation 996 before Scaling;
CNN1          0.519954
CNN10        13.497248
CNN11         2.973148
CNN12         0.893685
CNN2          0.549669
CNN3         13.422923
CNN4          0.737093
CNN5          0.969022
CNN6         10.066145
CNN7          0.496396
CNN8          0.710909
CNN9          0.138578
ImgBright    14.602948
Minute      996.000000
Name: 996, dtype: float64
Features for Observation 996 after Scaling;
[-1.63632507 -0.05960616 -0.23232957  1.04583623  1.18990495 -0.96740363
  1.21820739  1.23817318 -0.49863179 -0.16374712  0.94260499 -0.40193354
 -0.10481988 -0.63628826]
```

[12]:
```
#Lasso Regression
from sklearn import linear_model
lasso_model = linear_model.Lasso(random_state = 1693)
lasso_model.fit(X_scaled, y)
print(list(zip(lasso_model.coef_, X.columns)))
lasso_model.predict(X_example)
```

```
[(-0.189539861536648, 'CNN1'), (36.41550346851474, 'CNN10'),
(34.780365892097414, 'CNN11'), (15.340684073447816, 'CNN12'), (-0.0, 'CNN2'),
(28.18960914801363, 'CNN3'), (-0.0, 'CNN4'), (0.21744841868071443, 'CNN5'),
(0.0, 'CNN6'), (-0.0, 'CNN7'), (0.0, 'CNN8'), (-12.981008564797, 'CNN9'),
(3.272856700355233, 'ImgBright'), (12.662131089977548, 'Minute')]
```

[12]: array([57.35519287])

[13]:
```
#OMP example (Orthganal Matching Pursuit)
from sklearn.linear_model import OrthogonalMatchingPursuit
orth_MatchingPursuit_model = OrthogonalMatchingPursuit().fit(X_scaled, y)

print(list(zip(orth_MatchingPursuit_model.coef_, X.columns)))
orth_MatchingPursuit_model.predict(X_example)
```

```
[(0.0, 'CNN1'), (63.15600513208173, 'CNN10'), (0.0, 'CNN11'), (0.0, 'CNN12'),
(0.0, 'CNN2'), (0.0, 'CNN3'), (0.0, 'CNN4'), (0.0, 'CNN5'), (0.0, 'CNN6'), (0.0,
'CNN7'), (0.0, 'CNN8'), (0.0, 'CNN9'), (0.0, 'ImgBright'), (0.0, 'Minute')]
```

[13]: array([77.67162426])

[14]:
```
from sklearn.linear_model import LinearRegression
linReg = LinearRegression().fit(X_scaled, y)
print(list(zip(linReg.coef_, X.columns)))
linReg.predict(X_example)
```

```
[(-1.1522592280111343, 'CNN1'), (36.87421541515054, 'CNN10'),
(35.287370245610866, 'CNN11'), (16.246763908124862, 'CNN12'),
(-0.5118646695773528, 'CNN2'), (28.831686654488955, 'CNN3'),
(-0.5769583160921092, 'CNN4'), (1.2371313713620147, 'CNN5'),
(-0.9529069089414348, 'CNN6'), (-0.48923972286775896, 'CNN7'),
(0.2666646460880795, 'CNN8'), (-13.976149596728849, 'CNN9'), (4.200620320250248,
'ImgBright'), (13.339764181593866, 'Minute')]
```

[14]:
```
array([59.74064565])
```

[15]:
```
#Ridge Regression
from sklearn.linear_model import Ridge
ridgeReg = Ridge(random_state = 1693)
ridgeReg.fit(X_scaled, y)
print(list(zip(ridgeReg.coef_, X.columns)))
ridgeReg.predict(X_example)
```

```
[(-1.1514493072521887, 'CNN1'), (36.86555448157319, 'CNN10'),
(35.27899516099241, 'CNN11'), (16.24177972451253, 'CNN12'),
(-0.5114199804904045, 'CNN2'), (28.815977681127258, 'CNN3'),
(-0.5767094224591198, 'CNN4'), (1.2362497398102554, 'CNN5'),
(-0.9510557257981658, 'CNN6'), (-0.4887458096643911, 'CNN7'),
(0.26658964202896324, 'CNN8'), (-13.970855824062067, 'CNN9'),
(4.211561288554989, 'ImgBright'), (13.33964885074893, 'Minute')]
```

[15]:
```
array([59.74723156])
```

[16]:
```
#Support Vector Regression
from sklearn.svm import LinearSVR
SVR =LinearSVR(random_state = 1693, tol = 1e-5)
SVR.fit(X_scaled, y)
print(list(zip(SVR.coef_, X.columns)))
SVR.predict(X_example)
```

```
[(-0.233023339133183, 'CNN1'), (38.91492947402267, 'CNN10'),
(36.692799730436775, 'CNN11'), (5.48025642172064, 'CNN12'),
(0.10243954096250954, 'CNN2'), (12.050489154776733, 'CNN3'),
(0.22979794735771228, 'CNN4'), (0.21752747933775068, 'CNN5'),
(0.8895455318502276, 'CNN6'), (-0.04735344949898237, 'CNN7'),
(0.0026317050417289265, 'CNN8'), (-4.215991565608434, 'CNN9'),
(1.8891145863173688, 'ImgBright'), (8.401950748417267, 'Minute')]
```

[16]:
```
array([44.92462493])
```

[17]:
```python
from sklearn.linear_model import HuberRegressor
huber = HuberRegressor().fit(X_scaled, y)
print(list(zip(huber.coef_, X.columns)))
huber.predict(X_example)
```

```
[(-0.057493808472410 48, 'CNN1'), (40.61626931733897, 'CNN10'),
(36.95607634190696, 'CNN11'), (6.66230550415619, 'CNN12'),
(-0.15338179813720612, 'CNN2'), (13.330974647811217, 'CNN3'),
(0.23253227195530846, 'CNN4'), (0.4391268056515255, 'CNN5'),
(0.2530020283791508, 'CNN6'), (-0.4046962444764348, 'CNN7'),
(0.11828899983422402, 'CNN8'), (-5.788294928053872, 'CNN9'), (2.012499396194212,
'ImgBright'), (8.795685226990777, 'Minute')]
```

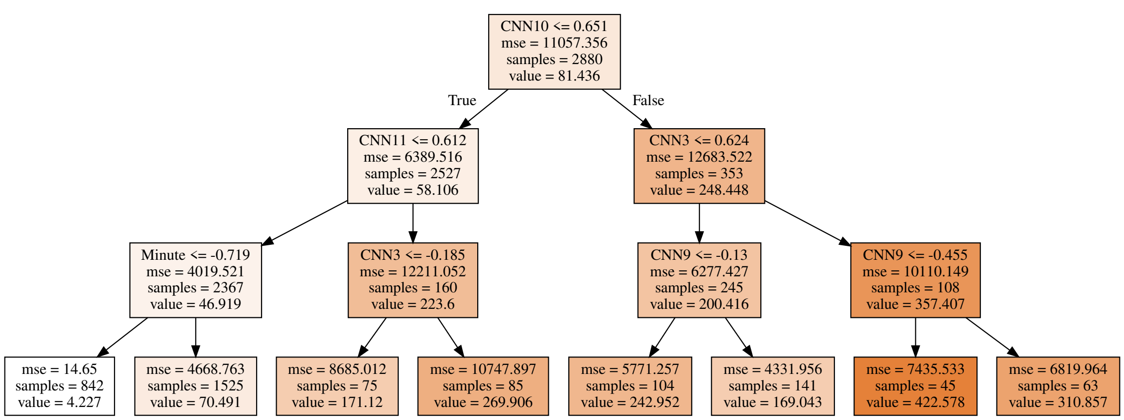
[17]: array([47.14174987])

[18]:
```python
from sklearn.tree import DecisionTreeRegressor
treeRegressor = DecisionTreeRegressor(random_state = 1693, max_depth = 3).
 ↪fit(X_scaled, y)

from IPython.display import SVG
from graphviz import Source
from IPython.display import display
import sklearn.tree

graph = Source(sklearn.tree.export_graphviz(treeRegressor, out_file=None,␣
 ↪feature_names=X.columns, class_names=['o','1','2'], filled = True))
display(SVG(graph.pipe(format='svg')))

treeRegressor.predict(X_example)
```



[18]: array([70.49114754])

[19]:
```
from sklearn.neighbors import RadiusNeighborsRegressor
radNeighbors = RadiusNeighborsRegressor(radius=2.5)
radNeighbors.fit(X_scaled, y)
print(radNeighbors.predict(X_example))
```

```
[56.8]
```

[20]:
```
from sklearn.neural_network import MLPRegressor
mlp = MLPRegressor(hidden_layer_sizes=(10,3), random_state = 1693,␣
 ↪max_iter=2000)
mlp.fit(X_scaled, y)
mlp.predict(X_example)
```

[20]:
```
array([33.29770806])
```

[21]:
```
from sklearn import linear_model
lars = linear_model.Lars(n_nonzero_coefs=1)
lars.fit(X_scaled, y)
print(list(zip(lars.coef_, X.columns)))
print(lars.predict(X_example))
```

```
[(0.0, 'CNN1'), (2.902523304341768, 'CNN10'), (0.0, 'CNN11'), (0.0, 'CNN12'),
(0.0, 'CNN2'), (0.0, 'CNN3'), (0.0, 'CNN4'), (0.0, 'CNN5'), (0.0, 'CNN6'), (0.0,
'CNN7'), (0.0, 'CNN8'), (0.0, 'CNN9'), (0.0, 'ImgBright'), (0.0, 'Minute')]
[81.26310285]
```

[22]:
```
from sklearn.linear_model import ElasticNet
elastic = ElasticNet(random_state = 1693)
elastic.fit(X_scaled, y)
print(list(zip(elastic.coef_, X.columns)))
print(elastic.predict(X_example))
```

```
[(-0.2853309134297988, 'CNN1'), (27.690147499460025, 'CNN10'),
(26.599642890934685, 'CNN11'), (10.89703505512573, 'CNN12'), (-0.0, 'CNN2'),
(17.878500451505843, 'CNN3'), (-0.0, 'CNN4'), (0.19664133438168338, 'CNN5'),
(1.722678378083506, 'CNN6'), (-0.0, 'CNN7'), (0.0, 'CNN8'), (-8.79405118535832,
'CNN9'), (8.514861005660839, 'ImgBright'), (11.905138663200436, 'Minute')]
[62.62489736]
```

[23]:
```
from sklearn.linear_model import PassiveAggressiveRegressor
passiveAgressiveModeling = PassiveAggressiveRegressor(max_iter=100,␣
 ↪random_state = 1693, tol = 1e-3)
passiveAgressiveModeling.fit(X_scaled, y)
print(list(zip(passiveAgressiveModeling.coef_, X.columns)))
print(passiveAgressiveModeling.predict(X_example))
```

```
[(-0.8700309071379682, 'CNN1'), (41.59353336321102, 'CNN10'),
(37.526148272007894, 'CNN11'), (11.324888887871031, 'CNN12'),
```

```
(1.4051497376187427, 'CNN2'), (21.02102276803901, 'CNN3'), (0.6920569156878602,
'CNN4'), (5.195550115345707, 'CNN5'), (-0.33855319239378523, 'CNN6'),
(-2.2819075415210124, 'CNN7'), (-1.9602796432244964, 'CNN8'),
(-8.402783520934781, 'CNN9'), (4.611025998374079, 'ImgBright'),
(6.086884625284848, 'Minute')]
[60.5661349]
```

[24]:

```
from sklearn.linear_model import RANSACRegressor
RANSAC = RANSACRegressor(random_state = 1693).fit(X_scaled, y)
print(list(zip(RANSAC.estimator_.coef_, X.columns)))
print(RANSAC.predict(X_example))
```

```
[(0.4724297871375373, 'CNN1'), (26.496173187622084, 'CNN10'),
(81.04743429570088, 'CNN11'), (1.095492178635407, 'CNN12'),
(-1.8361259791099087, 'CNN2'), (12.78252029413983, 'CNN3'),
(-0.5190513190208552, 'CNN4'), (2.446956135101898, 'CNN5'), (14.499852792017869,
'CNN6'), (-0.17808587903457962, 'CNN7'), (3.387008802043278, 'CNN8'),
(-5.072118817825963, 'CNN9'), (-10.463589393317577, 'ImgBright'),
(9.184762017489238, 'Minute')]
[37.41269555]
```

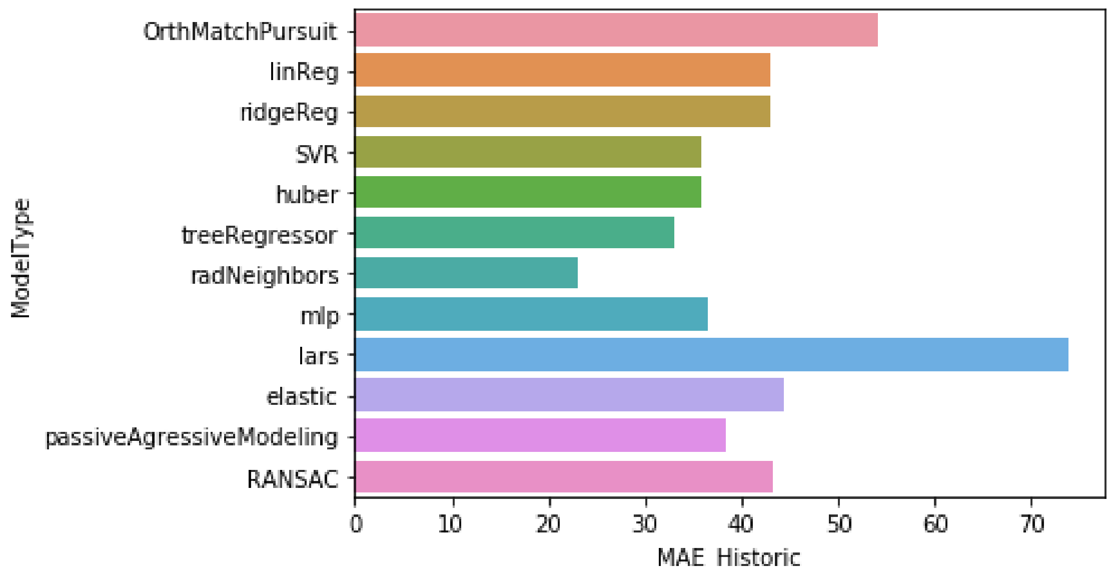
[25]:

```
col_names = ["ModelType", "MAE_Historic"]
accuracy_df = pd.DataFrame(columns = col_names)

from sklearn.metrics import mean_absolute_error
mae_lasso = mean_absolute_error(y, lasso_model.predict(X_scaled))
accuracy_df.loc[len(accuracy_df)] = ["OrthMatchPursuit", mean_absolute_error(y,␣
↪orth_MatchingPursuit_model.predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["linReg", mean_absolute_error(y, linReg.
↪predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["ridgeReg", mean_absolute_error(y,␣
↪ridgeReg.predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["SVR", mean_absolute_error(y, SVR.
↪predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["huber", mean_absolute_error(y, huber.
↪predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["treeRegressor", mean_absolute_error(y,␣
↪treeRegressor.predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["radNeighbors", mean_absolute_error(y,␣
↪radNeighbors.predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["mlp", mean_absolute_error(y, mlp.
↪predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["lars", mean_absolute_error(y, lars.
↪predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["elastic", mean_absolute_error(y, elastic.
↪predict(X_scaled))]
```

```
accuracy_df.loc[len(accuracy_df)] = ["passiveAgressiveModeling",␣
↪mean_absolute_error(y, passiveAgressiveModeling.predict(X_scaled))]
accuracy_df.loc[len(accuracy_df)] = ["RANSAC", mean_absolute_error(y, RANSAC.
↪predict(X_scaled))]

ax = sns.barplot(x="MAE_Historic",y="ModelType", data=accuracy_df)
```



[26]:
```
accuracy_df
```

[26]:

	ModelType	MAE_Historic
0	OrthMatchPursuit	54.035134
1	linReg	42.987470
2	ridgeReg	42.985787
3	SVR	35.803645
4	huber	35.835367
5	treeRegressor	33.065817
6	radNeighbors	23.142778
7	mlp	36.610336
8	lars	73.899527
9	elastic	44.394564
10	passiveAgressiveModeling	38.308017
11	RANSAC	43.123758

[27]:
```
%matplotlib notebook
import matplotlib.pyplot as plt
import time
requests.packages.urllib3.disable_warnings()
```

```
cur_time = pd.Timestamp("9/1/2019 0:00")

pred_col_names = ["Timestamp", "TruePersonCount", "EstimatedPersonCount", "MAE"]
pred_df = pd.DataFrame(columns=pred_col_names)

fig = plt.gcf()
fig.set_size_inches(8,6)
fig.show
fig.canvas.draw()
legend_draw=0

while (cur_time < pd.Timestamp("9/3/2019 23:59")):
    r = requests.post('https://cdsw00.geo.sciclone.wm.edu/api/altus-ds-1/models/
 ↪call-model',
                      data = '{"accessKey":"m149rzguxkf56i4pnqsulvkmfx43zu5t",␣
 ↪"request":{"timestamp_start":"' + str(cur_time)+ '", "timestamp_end":"' +␣
 ↪str(cur_time)+ '"}}',
                      headers = {"Content-Type" : "application/json", "host":"cdsw.
 ↪geo.sciclone.wm.edu"}, verify=False)
    dta = pd.read_json(r.json()["response"], orient="index")
    dta["Minute"] = dta["Timestamp"].dt.minute + dta["Timestamp"].dt.hour * 60

    timestamp= dta["Timestamp"].iloc[0]
    truepersoncount = dta["PersonCount"].iloc[0]
    dta_clean = dta.drop(["ApproachCount", "FemaleCount", "MaleCount",␣
 ↪"Timestamp", "pdtimes", "PersonCount"], axis = 1)

    scaled_X = scalingModel.transform(dta_clean.values.reshape(1,-1))

    estimated_person_count = radNeighbors.predict(scaled_X)[0]

    cur_time = cur_time + pd.Timedelta(minutes=240)

    MAE = abs(truepersoncount - estimated_person_count)

    pred_df.loc[len(pred_df)] = [timestamp, truepersoncount,␣
 ↪estimated_person_count, MAE]

    plt.plot(pred_df["Timestamp"].values, pred_df["TruePersonCount"].values,␣
 ↪"g^", label = "True Number of Persons")
    plt.plot(pred_df["Timestamp"].values, pred_df["EstimatedPersonCount"].
 ↪values, "b^", label = "Estimated Number of Persons")
    plt.plot(pred_df["Timestamp"].values, pred_df["MAE"].values, "r-.", label =␣
 ↪"Mean Absolute Error")
```

```
    if (legend_draw==0):
        plt.legend()
        legend_draw=1


    plt.xticks(rotation=90)

    fig.canvas.draw()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
/opt/conda/lib/python3.7/site-
packages/pandas/plotting/_matplotlib/converter.py:103: FutureWarning: Using an
implicitly registered datetime converter for a matplotlib plotting method. The
converter was registered by pandas on import. Future versions of pandas will
require you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[ ]:

```
```

[28]:

```
#Question 6
from sklearn import linear_model
lars = linear_model.Lars(n_nonzero_coefs=3)
lars.fit(X_scaled, y)
print(list(zip(lars.coef_, X.columns)))
print(lars.predict(X_example))
```

```
[(0.0, 'CNN1'), (27.806421058057335, 'CNN10'), (25.387772197202903, 'CNN11'),
(0.0, 'CNN12'), (0.0, 'CNN2'), (14.019215256660464, 'CNN3'), (0.0, 'CNN4'),
(0.0, 'CNN5'), (0.0, 'CNN6'), (0.0, 'CNN7'), (0.0, 'CNN8'), (0.0, 'CNN9'), (0.0,
'ImgBright'), (0.0, 'Minute')]
[60.31810724]
```

[29]:

```
#Question 8
X_example2 = scalingModel.transform(X.iloc[900].values.reshape(1,-1))
from sklearn.neighbors import RadiusNeighborsRegressor
radNeighbors = RadiusNeighborsRegressor(radius=2.5)
radNeighbors.fit(X_scaled[900], y)
print(radNeighbors.predict(X_example2))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-29-67968e157bab> in <module>
      3 from sklearn.neighbors import RadiusNeighborsRegressor
      4 radNeighbors = RadiusNeighborsRegressor(radius=2.5)
----> 5 radNeighbors.fit(X_scaled[900], y)
      6 print(radNeighbors.predict(X_example2))

/opt/conda/lib/python3.7/site-packages/sklearn/neighbors/base.py in fit(self, X, 
 ↪y)
    870         """
    871         if not isinstance(X, (KDTree, BallTree)):
--> 872             X, y = check_X_y(X, y, "csr", multi_output=True)
    873         self._y = y
    874         return self._fit(X)

/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
 ↪check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy,␣
 ↪force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples,␣
 ↪ensure_min_features, y_numeric, warn_on_dtype, estimator)
    717                         ensure_min_features=ensure_min_features,
    718                         warn_on_dtype=warn_on_dtype,
--> 719                         estimator=estimator)

    720     if multi_output:
    721         y = check_array(y, 'csr', force_all_finite=True, ensure_2d=False,

/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
 ↪check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,␣
 ↪force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
 ↪ensure_min_features, warn_on_dtype, estimator)
    519                     "Reshape your data either using array.reshape(-1, 1) 
 ↪if "
    520                     "your data has a single feature or array.reshape(1,␣
 ↪-1) "
--> 521                     "if it contains a single sample.".format(array))

    522
    523         # in the future np.flexible dtypes will be handled like object␣
 ↪dtypes

ValueError: Expected 2D array, got 1D array instead:
array=[ 1.43764998e+00 -4.98251770e-01 -5.33182698e-01 -4.38563662e-01
 -1.21248385e+00  1.68407155e-01  1.24696670e+00  2.43555565e-01
  5.39313054e-04 -9.64017898e-01  3.34814888e-01  4.16762607e-01
 -4.31253317e-01 -8.45577786e-01].
Reshape your data either using array.reshape(-1, 1) if your data has a single␣
 ↪feature or array.reshape(1, -1) if it contains a single sample.
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```