Lab_7_Notebook

November 23, 2021

[1]:
```
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)
```

[2]:
```
!conda install -y graphviz python-graphviz    #(Already downloaded dont need to␣
 ↪do again)
```

```
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
    openssl-1.1.1d             |       h516909a_0         2.1 MB  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-0.13.2     |             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:        23.4 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-0.13.2-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
  openssl                              1.1.1c-h516909a_0 -->
1.1.1d-h516909a_0



Downloading and Extracting Packages
xorg-xextproto-7.3.0 | 27 KB     | ##################################### | 100%
fontconfig-2.13.1    | 340 KB    | ##################################### | 100%
libuuid-2.32.1       | 26 KB     | ##################################### | 100%
xorg-kbproto-1.0.7   | 26 KB     | ##################################### | 100%
glib-2.58.3          | 3.3 MB    | ##################################### | 100%
xorg-libxt-1.1.5     | 367 KB    | ##################################### | 100%
libtool-2.4.6        | 512 KB    | ##################################### | 100%
```

```
graphviz-2.40.1       | 6.4 MB    | ##################################### | 100%
xorg-libxpm-3.5.12    | 63 KB     | ##################################### | 100%
expat-2.2.5           | 191 KB    | ##################################### | 100%
pixman-0.38.0         | 594 KB    | ##################################### | 100%
certifi-2019.9.11     | 147 KB    | ##################################### | 100%
python-graphviz-0.13  | 18 KB     | ##################################### | 100%
openssl-1.1.1d        | 2.1 MB    | ##################################### | 100%
xorg-libice-1.0.10    | 57 KB     | ##################################### | 100%
pcre-8.43             | 257 KB    | ##################################### | 100%
gettext-0.19.8.1      | 3.6 MB    | ##################################### | 100%
cairo-1.16.0          | 1.5 MB    | ##################################### | 100%
fribidi-1.0.5         | 112 KB    | ##################################### | 100%
graphite2-1.3.13      | 109 KB    | ##################################### | 100%
xorg-xproto-7.0.31    | 72 KB     | ##################################### | 100%
xorg-libsm-1.2.3      | 25 KB     | ##################################### | 100%
xorg-libxau-1.0.9     | 13 KB     | ##################################### | 100%
xorg-libxrender-0.9.  | 31 KB     | ##################################### | 100%
ca-certificates-2019  | 144 KB    | ##################################### | 100%
xorg-libx11-1.6.9     | 918 KB    | ##################################### | 100%
xorg-libxext-1.3.4    | 51 KB     | ##################################### | 100%
pthread-stubs-0.4     | 5 KB      | ##################################### | 100%
xorg-renderproto-0.1  | 8 KB      | ##################################### | 100%
libxcb-1.13           | 396 KB    | ##################################### | 100%
xorg-libxdmcp-1.1.3   | 18 KB     | ##################################### | 100%
harfbuzz-2.4.0        | 1.5 MB    | ##################################### | 100%
pango-1.42.4          | 517 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

[3]:
```
%matplotlib notebook
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.colors as cm
import numpy as np

dta_clean = dta.drop(["ApproachCount", "FemaleCount", "MaleCount", "Timestamp",␣
↪"pdtimes",
                      ␣
↪"CNN1","CNN2","CNN3","CNN4","CNN5","CNN6","CNN7","CNN8","CNN9","CNN11","CNN12"],␣
↪axis = 1)

y_continuous = dta_clean.pop("PersonCount")

y = (y_continuous > 30).astype(int)
print(y)
```

```
X = dta_clean

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

```
0       0
1       0
10      0
100     0
1000    1
       ..
995     1
996     1
997     1
998     1
999     1
Name: PersonCount, Length: 2880, dtype: int64
```

[4]:
```
colors = ["red","blue"]
fig = plt.figure(figsize=(8,8))
plt.scatter(X["ImgBright"].values,
            X["CNN10"].values,
            c=y,
            label = "High Foot Traffic",
            s=4,
            cmap = cm.ListedColormap(colors))
plt.grid(True)
plt.legend()
plt.ylabel("Values From Convolutional Neural Network (Feature 10)")
plt.xlabel("Image Brightness")
```

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

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

[4]:
```
Text(0.5, 0, 'Image Brightness')
```

[5]:
```
from sklearn.neighbors import KNeighborsClassifier
NNeighbors = KNeighborsClassifier(n_neighbors=10).fit(X_scaled,y)

xx1, xx2 = np.meshgrid(np.arange(0,30,5), np.arange(0, 100, 10))
XX = np.stack([xx1.ravel(), xx2.ravel()], axis=1)
XX_grid_scaled = scalingModel.transform(XX)

Z = NNeighbors.predict_proba(XX_grid_scaled)[:, 1]
```

```
Z = Z.reshape(len(xx1),len(xx2[0]))

colors = ["red","blue"]

fig = plt.figure(figsize=(8,8))
plt.contourf(xx1, xx2, Z, cmap=plt.cm.RdBu, alpha=.8)
plt.scatter(X["ImgBright"].values, X["CNN10"].values, c=y, label = "High Foot␣
 ↪Traffic",s=4,cmap = cm.ListedColormap(colors))
plt.xlim(0,25)
plt.ylim(0,170)
plt.grid(True)
plt.legend()
plt.ylabel("Values From Convolutional Neural Network (Feature 10)")
plt.xlabel("Image Brightness")


#Why the fuck is it onl blue up until a vule of 90 for CNN instead of the whole␣
 ↪way?
```

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

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

[5]:
```
Text(0.5, 0, 'Image Brightness')
```

[6]:
```
def viz_classifier(model):
    xx1, xx2 = np.meshgrid(np.arange(0,30,5), np.arange(0, 100, 10))
    XX = np.stack([xx1.ravel(), xx2.ravel()], axis=1)
    global scalingModel
    XX_grid_scaled = scalingModel.transform(XX)

    Z = model.predict_proba(XX_grid_scaled)[:, 1]

    Z = Z.reshape(len(xx1),len(xx2[0]))

    colors = ["red","blue"]

    fig = plt.figure(figsize=(8,8))
    plt.contourf(xx1, xx2, Z, cmap=plt.cm.RdBu, alpha=.8)
    plt.scatter(X["ImgBright"].values, X["CNN10"].values, c=y, label = "High␣
 ↪Foot Traffic",s=4,cmap = cm.ListedColormap(colors))
    plt.xlim(0,25)
    plt.ylim(0,170)
    plt.grid(True)
    plt.legend()
    plt.ylabel("Values From Convolutional Neural Network (Feature 10)")
    plt.xlabel("Image Brightness")
```

```
    plt.title(str(model))

viz_classifier(NNeighbors)
```

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

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

[7]:
```
from sklearn.svm import SVC
linear_svm = SVC(kernel = "linear", C=0.025, probability=True).fit(X_scaled,y)
viz_classifier(linear_svm)
```

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

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

[8]:
```
from sklearn.svm import SVC
radial_svm = SVC(gamma=2, C=1, probability=True).fit(X_scaled,y)
viz_classifier(radial_svm)
```

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

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

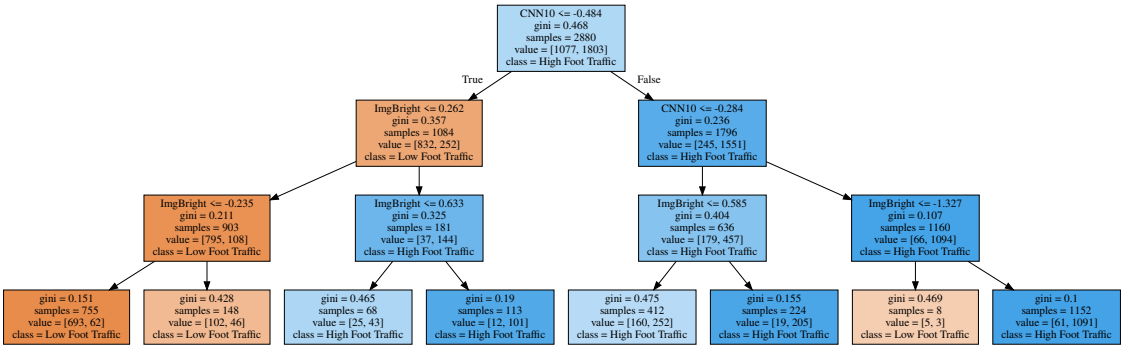
[9]:
```
from sklearn.tree import DecisionTreeClassifier
dTree = DecisionTreeClassifier(random_state = 1693, max_depth = 3).
 ↪fit(X_scaled, y)
viz_classifier(dTree)

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

graph = Source(sklearn.tree.export_graphviz(dTree, out_file=None,␣
 ↪feature_names=X.columns,
                                            class_names=["Low Foot␣
 ↪Traffic","High Foot Traffic"], filled = True))
display(SVG(graph.pipe(format='svg')))
```

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

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



[24]:
```
from sklearn.ensemble import RandomForestClassifier
rForest = RandomForestClassifier(n_estimators=1000, random_state = 1693,␣
 ↪max_depth = 3).fit(X_scaled, y)
viz_classifier(rForest)
```

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

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

[25]:
```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(alpha=1, max_iter=1000).fit(X_scaled,y)
viz_classifier(mlp)
```

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

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

[26]:
```
from sklearn.metrics import accuracy_score
import ipywidgets
from ipywidgets import interact, interact_manual

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.colors as cm
import numpy as np

dta_clean = dta.drop(["ApproachCount", "FemaleCount", "MaleCount", "Timestamp",␣
 ↪"pdtimes",
                      ␣
 ↪"CNN1","CNN2","CNN3","CNN4","CNN5","CNN6","CNN7","CNN8","CNN9","CNN11","CNN12"
                      ], axis = 1)

y_continuous = dta_clean.pop("PersonCount")
```

```
y = (y_continuous > 30).astype(int)

X = dta_clean

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

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier

@interact_manual(
selection = ipywidgets.Dropdown(options=
                                ["Nearest Neighbor",
                                "Radial SVM",
                                "Decision Tree",
                                "Neural Network",
                                "Random Forest"],
                                value = "Nearest Neighbor",description =␣
↪"Model"))

def viz_classifier(selection):
    if (selection == "Nearest Neighbor"):
        model = KNeighborsClassifier(n_neighbors=10).fit(X_scaled,y)
    if (selection == "Radial SVM"):
        model = SVC(gamma=2, C=1, probability=True).fit(X_scaled,y)
    if (selection == "Decision Tree"):
        model = DecisionTreeClassifier(random_state = 1693, max_depth = 3).
↪fit(X_scaled, y)
    if (selection == "Neural Network"):
        model = MLPClassifier(alpha=1, max_iter=1000).fit(X_scaled,y)
    if (selection == "Random Forest"):
        model = RandomForestClassifier(n_estimators=1000, random_state = 1693,␣
↪max_depth = 3).fit(X_scaled, y)

    xx1, xx2 = np.meshgrid(np.arange(0,30,5), np.arange(0, 100, 10))

    #stack our dummy data for the grid
    XX = np.stack([xx1.ravel(), xx2.ravel()], axis=1)
    global scalingModel
    XX_grid_scaled = scalingModel.transform(XX)

    Z = model.predict_proba(XX_grid_scaled)[:, 1]
    Z = Z.reshape(len(xx1),len(xx2[0]))
```

```
    colors = ["red","blue"]
    fig = plt.figure(figsize=(8,8))
    plt.contourf(xx1, xx2, Z, cmap=plt.cm.RdBu, alpha=.8)
    plt.scatter(X["ImgBright"].values, X["CNN10"].values, c=y, label = "High␣
 ↪Foot Traffic",s=4,cmap = cm.ListedColormap(colors))
    plt.xlim(0,25)
    plt.ylim(0,170)
    plt.grid(True)
    plt.legend()
    plt.ylabel("Values From Convolutional Neural Network (Feature 10)")
    plt.xlabel("Image Brightness")
    plt.title(str(selection) + "Accuracy: " + str(accuracy_score(y, model.
 ↪predict(X_scaled))))
```

```
interactive(children=(Dropdown(description='Model', options=('Nearest Neighbor',␣
 ↪'Radial SVM', 'Decision Tree'…
```

[17]:
```
%matplotlib notebook

import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator
import time
requests.packages.urllib3.disable_warnings()

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

pred_col_names = ["Timestamp", "trulyBusy", "estimateBusy",␣
 ↪"estimatedProbabilityBusy"]
pred_df = pd.DataFrame(columns=pred_col_names)

fig, ax = plt.subplots()
fig.set_size_inches(11,8)
fig.show()
fig.canvas.draw()
legend_draw=0
first_run=0
end_time = pd.Timestamp("9/1/2019 23:59")
time_step_min = 30
ax.set_yticks([0.25,0.75])
ax.set_yticklabels(["", "Observed Busy"])
ax2 = ax.twinx()
ax2._sharex = ax

timestamp = 0

while (cur_time < end_time):
```

```
   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")

   timestamp = timestamp + time_step_min

   if (first_run==0):
       dta["Minute"] = dta["Timestamp"].dt.minute + dta["Timestamp"].dt.hour *␣
↪60
       timestamp = dta["Minute"].iloc[0].astype(float)
       first_run = 1

   trulyBusy = (dta["PersonCount"].iloc[0] > 30).astype(int)

   dta_clean = dta.drop(["ApproachCount", "FemaleCount", "MaleCount",␣
↪"Timestamp", "pdtimes",
                      ␣
↪"CNN1","CNN2","CNN3","CNN4","CNN5","CNN6","CNN7","CNN8","CNN9","CNN11","CNN12"
                       ], axis = 1)

   scaled_X = scalingModel.transform(dta_clean[["CNN10", "ImgBright"]].values.
↪reshape(1,-1))

   estimateBusy = NNeighbors.predict(scaled_X)[0]
   prob = NNeighbors.predict_proba(scaled_X)[0]

   pred_df.loc[len(pred_df)] = [timestamp,trulyBusy, estimateBusy, prob[1]]

   plt.bar(pred_df["Timestamp"], height=pred_df["trulyBusy"].values,␣
↪width=30,color="blue")


   plt.xticks(np.arange(len(pred_df)), pred_df["Timestamp"])

   if(len(pred_df)<25):
       ax.xaxis.set_major_locator(MultipleLocator(30))
   if(len(pred_df)>25):
       ax.xaxis.set_major_locator(MultipleLocator(120))
   if(len(pred_df)>100):
       ax.xaxis.set_major_locator(MultipleLocator(360))
```

```
    ax.yaxis.set_major_locator(MultipleLocator(1))
    ax.grid(color="w",linestyle="-",zorder=1.0)

    ax2.plot(pred_df["Timestamp"], pred_df["estimatedProbabilityBusy"],␣
↪color="red")
    fig.canvas.draw()

    cur_time = cur_time + pd.Timedelta(minutes=time_step_min)
```

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

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

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: