Total points for this HW: 10

Make sure that you run all your codes and that all results are printed.

Please note: Copying and pasting other people's work is absolutely prohibited. Any such cases will be reported to CUSP's education team and severely punished. Discussion is encouraged, and feel free to exchange ideas with your classmates, but please write your own code and do your own work.

In [1]:
```
from IPython.display import Image
import pandas as pd
import numpy as np
```

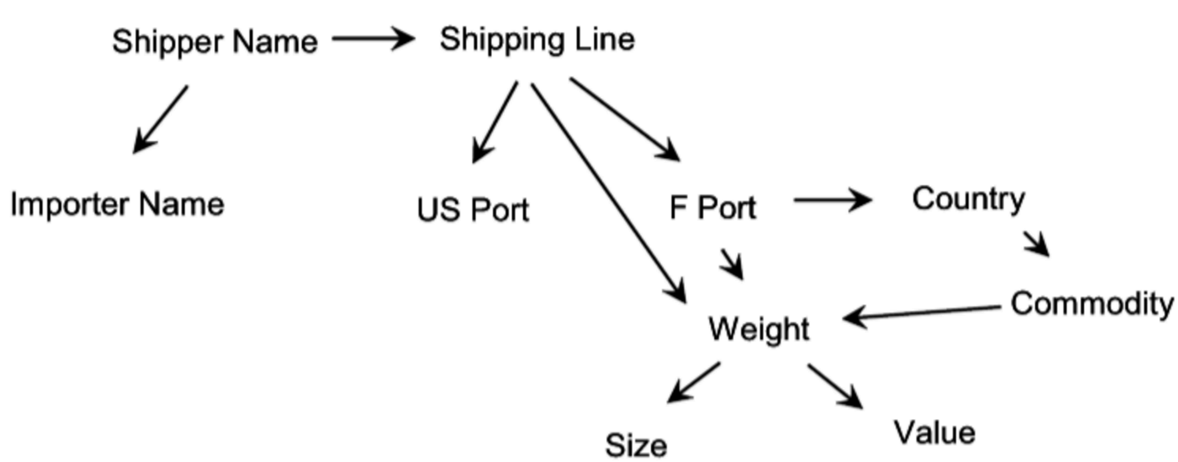
Question 1 (15%)

This task is to be done with manual calculations rather than using Python.

Given the following learned Bayesian network structure explaining the relationships between variables in container shipping data:

In [2]:
```
Image('HW3Q1.png')
```

Out[2]:



1) Which of the following conditional independence relationships hold? Choose "Independent" or "Dependent" for each (6%):

CI (Shipper Name, Value | F Port)? Dependent

CI (Shipper Name, Value | Shipping Line)? Dependent

CI (Foreign Port, Commodity | Country)? Independent

CI (Foreign Port, Commodity | County, Weight)? Independent

2) Now consider a smaller dataset with only four discrete attributes (Shipping Line, US Port, Foreign Port, Weight), and the following conditional probability distributions:

Shipping Line: CSCO (80%), ASCO (20%)

Foreign Port | Shipping Line = CSCO: Yokohama (40%), Vancouver (60%)

Foreign Port | Shipping Line = ASCO: Vancouver (100%)

US Port | Shipping Line = ASCO: Seattle (100%)

US Port | Shipping Line = CSCO: Seattle (20%), Los Angeles (80%)

Weight | Shipping Line = CSCO, Foreign Port = Vancouver: Light (30%), Medium (20%), Heavy (50%)

Weight | Shipping Line = CSCO, Foreign Port = Yokohama: Light (10%), Medium (60%), Heavy (30%)

Weight | Shipping Line = ASCO, Foreign Port = Vancouver: Light (15%), Medium (15%), Heavy (70%)

Which of the following packages is most anomalous?

a) A light package shipped from Vancouver to Seattle by ASCO

b) A medium package shipped from Vancouver to Seattle by CSCO

c) A heavy package shipped from Yokohama to Los Angeles by CSCO

To answer this question, compute the likelihood of each package given the Bayesian Network (lowest likelihood = most anomalous). You must show your calculations to receive credit. (9%)

(Your answers here, including all calculations)

a) A light package shipped from Vancouver to Seattle by ASCO

0.2 *1* 1 * 0.15 = 0.03

b) A medium package shipped from Vancouver to Seattle by CSCO

0.8 *0.6* 0.2 * 0.2 = 0.0192

c) A heavy package shipped from Yokohama to Los Angeles by CSCO

0.8 *0.4* 0.8 * 0.3 = 0.0768

So, b) is most anomalous

Question 2. Bayesian Network Learning (35%)

In this question, we use dataset: "HW3Q2.csv" for Bayesian Network Learning.

In [3]:
```
from sklearn.model_selection import train_test_split
data2=pd.read_csv("HW3Q2.csv")
train,test=train_test_split(data2,random_state=9,test_size=0.4)
data2.head()
```

Out[3]:

	A	B	C	D	E	F
0	1	0	0	0	1	1
1	0	0	0	1	0	1
2	1	1	0	1	1	1
3	1	0	0	1	1	1
4	2	0	1	0	1	1

a) Use the training data to select the best structure you want to use for Bayesian Network Learning. Please use Hill Climbing with BIC score metric. (10%)

b) Use the Bayesian Estimator to estimate the CPDs for your model and visualize the network with CPDs. (15%)

c) Use the model to predict "A" for the testing dataset. Report the out-of-sample prediction accuracy. (10%)

In [4]:
```
train
```

Out[4]:

	A	B	C	D	E	F
9962	2	1	1	0	1	0
2014	0	0	0	1	1	0
7683	0	0	0	1	0	0
2359	0	0	0	0	0	0
6906	1	0	0	1	1	0
...	...	...	...	...	...	...
6200	1	0	1	1	1	1
501	0	0	0	1	1	1
6782	1	1	1	1	1	1
4444	0	0	0	1	1	0
8574	2	1	0	1	1	1

6000 rows × 6 columns

a) Use the training data to select the best structure you want to use for Bayesian Network Learning. Please use Hill Climbing with BIC score metric. (10%)

In [5]:
```
import pandas as pd
import numpy as np
from pgmpy.estimators import K2Score, BicScore, BDeuScore
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import HillClimbSearch

# create some random data with dependencies

data = train
hc = HillClimbSearch(data)
best_model = hc.estimate(scoring_method='BicScore')
print(best_model.edges())
```

```
  0%|          | 0/1000000 [00:00<?, ?it/s]
[('B', 'A'), ('D', 'A'), ('E', 'A')]
```

b) Use the Bayesian Estimator to estimate the CPDs for your model and visualize the network with CPDs. (15%)

In [6]:
```
import numpy as np
import pandas as pd
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import BayesianEstimator
```

In [7]:
```
model = BayesianNetwork([('B', 'A'), ('D', 'A'), ('E', 'A')])
model.fit(data, estimator=BayesianEstimator, prior_type='K2')
for cpd in model.get_cpds():
    print("CPD of {variable}:".format(variable=cpd.variable))
    print(cpd)
```

```
CPD of B:
+------+--------+
| B(0) | 0.5015 |
+------+--------+
| B(1) | 0.4985 |
+------+--------+
CPD of A:
+------+----------------------+-----+----------------------+
| B    | B(0)                 | ... | B(1)                 |
+------+----------------------+-----+----------------------+
| D    | D(0)                 | ... | D(1)                 |
+------+----------------------+-----+----------------------+
| E    | E(0)                 | ... | E(1)                 |
+------+----------------------+-----+----------------------+
| A(0) | 0.4965893587994543   | ... | 0.001349527665317139 |
+------+----------------------+-----+----------------------+
| A(1) | 0.5006821282401092   | ... | 0.5033738191632928   |
+------+----------------------+-----+----------------------+
| A(2) | 0.001364256480218281 | ... | 0.4939271255060729   |
+------+----------------------+-----+----------------------+
| A(3) | 0.001364256480218281 | ... | 0.001349527665317139 |
+------+----------------------+-----+----------------------+
CPD of D:
+------+----------+
| D(0) | 0.493336 |
+------+----------+
| D(1) | 0.506664 |
+------+----------+
CPD of E:
+------+--------+
| E(0) | 0.5015 |
+------+--------+
| E(1) | 0.4985 |
+------+--------+
```

c) Use the model to predict "A" for the testing dataset. Report the out-of-sample prediction accuracy. (10%)

In [8]:
```python
hc = HillClimbSearch(train)
best_model = BayesianNetwork(hc.estimate(scoring_method='BicScore').edges()) #
```

```
  0%|          | 0/1000000 [00:00<?, ?it/s]
```

In [9]:
```python
best_model.fit(train, estimator=BayesianEstimator, prior_type='K2')
for cpd in best_model.get_cpds():
    print("CPD of {variable}:".format(variable=cpd.variable))
    print(cpd)
for thenode in best_model.nodes():
    print(best_model.local_independencies(thenode))
```

```
CPD of B:
+------+--------+
| B(0) | 0.5015 |
+------+--------+
| B(1) | 0.4985 |
+------+--------+
CPD of A:
+------+----------------------+-----+----------------------+
| B    | B(0)                 | ... | B(1)                 |
+------+----------------------+-----+----------------------+
| D    | D(0)                 | ... | D(1)                 |
+------+----------------------+-----+----------------------+
| E    | E(0)                 | ... | E(1)                 |
+------+----------------------+-----+----------------------+
| A(0) | 0.4965893587994543   | ... | 0.001349527665317139 |
+------+----------------------+-----+----------------------+
| A(1) | 0.5006821282401092   | ... | 0.5033738191632928   |
+------+----------------------+-----+----------------------+
| A(2) | 0.001364256480218281 | ... | 0.4939271255060729   |
+------+----------------------+-----+----------------------+
| A(3) | 0.001364256480218281 | ... | 0.001349527665317139 |
+------+----------------------+-----+----------------------+
CPD of D:
+------+----------+
| D(0) | 0.493336 |
+------+----------+
| D(1) | 0.506664 |
+------+----------+
CPD of E:
+------+--------+
| E(0) | 0.5015 |
+------+--------+
| E(1) | 0.4985 |
+------+--------+
(B ⊥ E, D)

(D ⊥ B, E)
(E ⊥ B, D)
```

In [10]:

```python
train_x = train.loc[:, ['B', 'D', 'E']]
predicted_train = best_model.predict(train_x)
print("In sample:",(train.loc[:,'A'].reset_index(drop=True)==predicted_train['A

test_x = test.loc[:, ['B', 'D', 'E']]
predicted_test = best_model.predict(test_x)
print("Out of sample:",(test.loc[:,'A'].reset_index(drop=True)==predicted_test
```

```
  0%|          | 0/8 [00:00<?, ?it/s]
In sample: 0.5766666666666667
  0%|          | 0/8 [00:00<?, ?it/s]
Out of sample: 0.5645
```

Question 3. Clustering: Spatial and Temporal Distributions of Chicago Crimes (50%)

In this question you will use k-means and Gaussian mixture clustering in sklearn and hierarchical clustering in scipy to answer the question, “Do different types of crime display different trends over space and time?" The dataset "HW3Q3_1.csv" consists of data for 119 different types of crime, each of which occurred at least 100 times in Chicago during the year 2016. For each crime type, we have various features representing the spatial and temporal distribution of crime, including:

- The proportion of all crimes of that type that occurred on each day of the week (day_Sun, day_Mon, ..., day_Sat).
- The proportion of all crimes of that type that occurred on each hour of the day (hour_0 = midnight to 12:59am, hour_1 = 1am to 1:59am, ..., hour_23 = 11pm to 11:59pm).
- The proportion of all crime of that type that occurred in each of the 77 community areas of Chicago (community_area_1 ... community_area_77).

We also have, for each crime type, its categorization by the FBI:

- Category = "P1V" corresponds to Part 1 Violent Crime, i.e., serious violent crimes
- Category = "P1P" corresponds to Part 1 Property Crime, i.e., serious property crimes
- Category = "P2" corresponds to Part 2 (less serious) crimes.

To answer parts a through f, you should cluster the 119 crime types using k-means into k = 3 clusters using only the hour of day (hour_0..hour_23) attributes.

a) Copy each cluster's mean values for hour_0...hour_23 into a DataFrame and create a line graph to visualize these values by cluster. (5%)

b) Describe the three different hour-of-day trends represented by these three clusters (5%).

c) Do you notice any consistent trends about which crime types are assigned to which cluster? Note that by a "crime type", we are referring to specific crimes such as "narcotics" or "assault", not the FBI categories. (5%)

d) Do the three clusters have different day-of-week trends? Again, visualize the trends for each cluster by creating a line graph and discuss any notable differences. (5%)

e) Do the three clusters affect different types of communities/neighborhoods? To answer this question, you could first compute the proportions of "cluster 1", "cluster 2", and "cluster 3" crimes for each community area, and identify particular community areas with disproportionate amounts of a given cluster. You can then use the provided file (HW3Q3_2.csv), to determine whether these community areas have any notable common characteristics (poverty, overcrowding, etc.). You may also wish to consult the Chicago Community Areas map at https://en.wikipedia.org/wiki/Community_areas_in_Chicago. (5%)

f) How well do the three groups formed by clustering hour-of-day trends correspond to the FBI's division between P1V, P1P, and P2 crimes? (5%)

g) For part g, you will use the same dataset to compare the clusters produced by several different methods. But this time you should cluster using only the *day-of-week* (not hour-of-day) attributes (day_Sun..day_Sat). Please perform four different clusterings using (i) k-means, (ii) Gaussian mixture models, (iii) Bottom-up hierarchical clustering with "single link" distance metric, and (iv) Bottom-up hierarchical clustering with "complete link" distance metric. In each case, you should choose the number of clusters using the silhouette method (or another established method of your choice- please specify). For each clustering, report the number of clusters formed and the number of elements in each cluster. You should also identify any notable similarities or differences between the clusterings. (20%)

In [11]:
```
data3=pd.read_csv("HW3Q3_1.csv")
data3.head()
```

Out[11]:

	crime_type	Category	day_Sun	day_Mon	day_Tue	day_Wed	day_Thu	day_Fri	day_S
0	ARSON: BY FIRE	P1P	0.138810	0.135977	0.155807	0.121813	0.130312	0.147309	0.1699
1	ASSAULT: AGG PO HANDS NO/MIN INJURY	P2	0.151852	0.118519	0.162963	0.122222	0.129630	0.129630	0.1851
2	ASSAULT: AGGRAVATED: HANDGUN	P1V	0.149912	0.139405	0.141506	0.131349	0.136953	0.133100	0.1677
3	ASSAULT: AGGRAVATED: OTHER DANG WEAPON	P1V	0.125000	0.139000	0.148000	0.153000	0.133000	0.142000	0.1600
4	ASSAULT: AGGRAVATED: OTHER FIREARM	P1V	0.156863	0.107843	0.166667	0.117647	0.147059	0.117647	0.1862

5 rows × 110 columns

In [12]:
```
data4=pd.read_csv("HW3Q3_2.csv")
data4.head()
```

Out[12]:

	Community Area Number	COMMUNITY AREA NAME	centroid_x	centroid_y	PERCENT OF HOUSING CROWDED	PERCENT HOUSEHOLDS BELOW POVERTY	PERCEN AGED 16 UNEMPLOYE
0	1	Rogers Park	1164399.219	1947666.815	7.7	23.6	8
1	2	West Ridge	1158307.200	1943243.722	7.8	17.2	8
2	3	Uptown	1168228.082	1930980.022	3.8	24.0	8
3	4	Lincoln Square	1159618.804	1933105.743	3.4	10.9	8
4	5	North Center	1161104.228	1924056.010	0.3	7.5	5

In [13]:

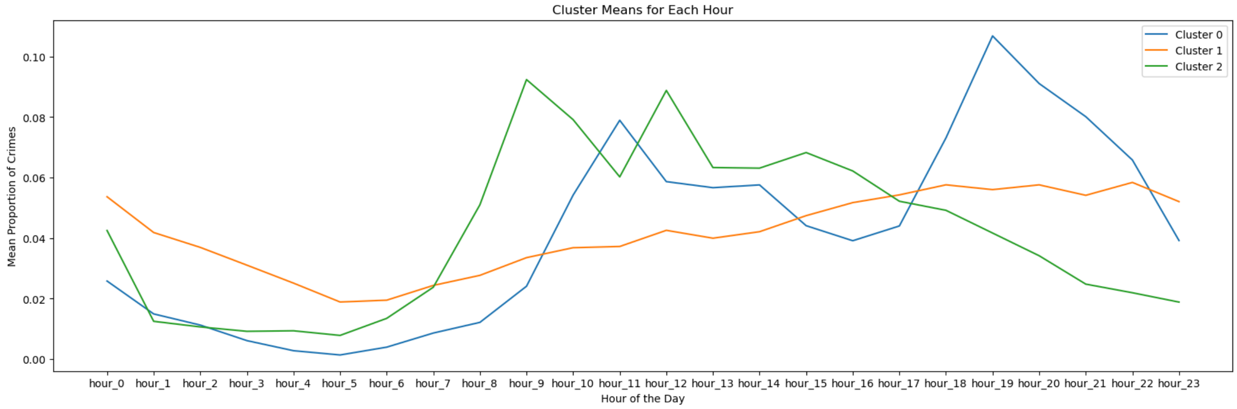
```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

a) Copy each cluster's mean values for hour_0...hour_23 into a DataFrame and create a line graph to visualize these values by cluster. (5%)

In [14]:

```
# Forgot the syntax of do the for loop to pick out the targeted columns, so I a
columns_to_cluster = [col for col in data3 if col.startswith('hour_')]

# Perform k-means clustering
kmeans = KMeans(n_clusters=3, random_state=0).fit(data3[columns_to_cluster])


data3['cluster'] = kmeans.labels_


cluster_means_df = pd.DataFrame()
for column in columns_to_cluster:
    cluster_means = data3.groupby('cluster')[column].mean()
    cluster_means_df[column] = cluster_means


cluster_means_df = cluster_means_df.reset_index()

plt.figure(figsize=(20, 6))
for i in range(cluster_means_df.shape[0]):
    plt.plot(cluster_means_df.columns[1:], cluster_means_df.iloc[i, 1:], label=

plt.title('Cluster Means for Each Hour')
plt.xlabel('Hour of the Day')
plt.ylabel('Mean Proportion of Crimes')
plt.legend()
plt.show()
```

```
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



b) Describe the three different hour-of-day trends represented by these three clusters (5%).

The line segment for Cluster 0 shows that the crime rate peaks around 11 AM and 7 PM each day. After midnight, the crime rate slowly decreases, but it rises sharply after 5 AM.

The line segment for Cluster 1 tends to be flat and regular, with little fluctuation in the crime rate throughout the day. Starting from midnight, the crime rate gradually decreases, and from 5 AM, it begins to climb gradually until just before midnight.

The line segment for Cluster 2 suggest that the crime rate reaches the peak at the 8am in the morning and 12pm. The crime rate dropping down since 3pm afternoon.

c) Do you notice any consistent trends about which crime types are assigned to which cluster? Note that by a "crime type", we are referring to specific crimes such as "narcotics" or "assault", not the FBI categories. (5%)

In [15]:
```
cluster_0 = data3.loc[data3['cluster']==0]
```

In [16]:
```
cluster_1 = data3.loc[data3['cluster']==1]
```

In [17]:
```
cluster_2 = data3.loc[data3['cluster']==2]
```

In [18]:
```
counts0 = cluster_0.groupby('crime_type')['Category'].count()
counts0
```

Out[18]:
```
crime_type
DECEPTIVE PRACTICE: COUNTERFEITING DOCUMENT                          1
GAMBLING: GAME/DICE                                                  1
INTERFERENCE WITH PUBLIC OFFICER: OBSTRUCTING IDENTIFICATION        1
NARCOTICS: MANU/DEL:CANNABIS 10GM OR LESS                            1
NARCOTICS: MANU/DEL:CANNABIS OVER 10 GMS                             1
NARCOTICS: MANU/DELIVER: HEROIN (WHITE)                              1
NARCOTICS: MANU/DELIVER:CRACK                                        1
NARCOTICS: POSS: CANNABIS 30GMS OR LESS                              1
NARCOTICS: POSS: CANNABIS MORE THAN 30GMS                            1
NARCOTICS: POSS: CRACK                                               1
NARCOTICS: POSS: HEROIN(WHITE)                                       1
NARCOTICS: POSS: PCP                                                 1
NARCOTICS: POSS: SYNTHETIC DRUGS                                     1
NARCOTICS: POSSESSION OF DRUG EQUIPMENT                              1
NARCOTICS: SOLICIT NARCOTICS ON PUBLICWAY                            1
OTHER OFFENSE: GUN OFFENDER: ANNUAL REGISTRATION                     1
OTHER OFFENSE: GUN OFFENDER: DUTY TO REGISTER                        1
OTHER OFFENSE: LICENSE VIOLATION                                     1
OTHER OFFENSE: PAROLE VIOLATION                                      1
OTHER OFFENSE: SEX OFFENDER: FAIL TO REGISTER                        1
PROSTITUTION: SOLICIT ON PUBLIC WAY                                  1
PUBLIC PEACE VIOLATION: RECKLESS CONDUCT                             1
WEAPONS VIOLATION: UNLAWFUL POSS OF HANDGUN                          1
Name: Category, dtype: int64
```

cluster_0: mostly Narcotics and other offense, little amount of prostitution, public peace violation and weapons violation.

In [19]:
```python
counts1 = cluster_1.groupby('crime_type')['Category'].count()
counts1
```

Out[19]:
```
crime_type
ARSON: BY FIRE                                      1
ASSAULT: AGG PO HANDS NO/MIN INJURY                 1
ASSAULT: AGGRAVATED: HANDGUN                        1
ASSAULT: AGGRAVATED: OTHER DANG WEAPON              1
ASSAULT: AGGRAVATED: OTHER FIREARM                  1
                                                   ..
THEFT: POCKET-PICKING                               1
THEFT: PURSE-SNATCHING                              1
WEAPONS VIOLATION: RECKLESS FIREARM DISCHARGE       1
WEAPONS VIOLATION: UNLAWFUL USE HANDGUN             1
WEAPONS VIOLATION: UNLAWFUL USE OTHER DANG WEAPON   1
Name: Category, Length: 69, dtype: int64
```

cluster_1: mostly Assault , Theft and weapons violation

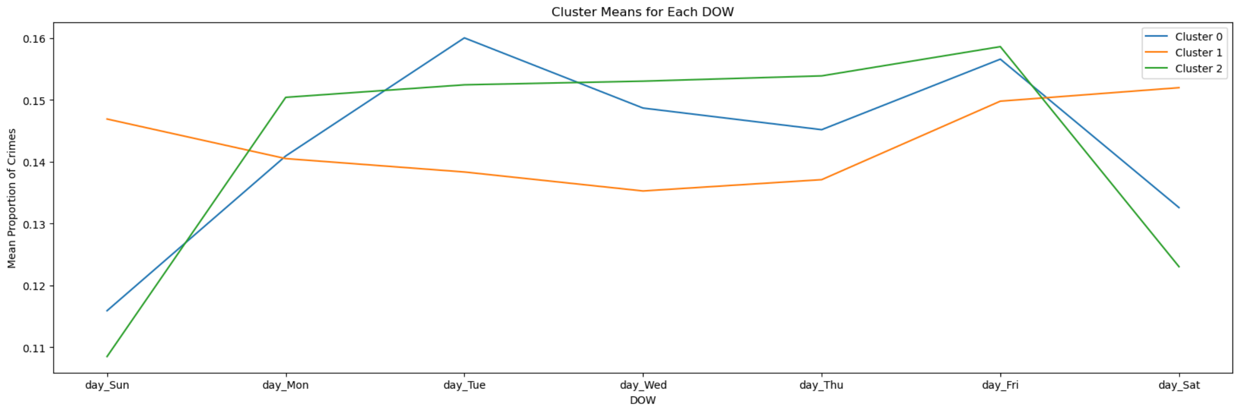
In [20]:
```python
counts2 = cluster_2.groupby('crime_type')['Category'].count()
counts2
```

Out[20]:
```
crime_type
ASSAULT: PRO EMP HANDS NO/MIN INJURY                           1
BATTERY: PRO EMP HANDS NO/MIN INJURY                           1
BURGLARY: UNLAWFUL ENTRY                                       1
DECEPTIVE PRACTICE: ATTEMPT - FINANCIAL IDENTITY THEFT         1
DECEPTIVE PRACTICE: BOGUS CHECK                                1
DECEPTIVE PRACTICE: COUNTERFEIT CHECK                          1
DECEPTIVE PRACTICE: CREDIT CARD FRAUD                          1
DECEPTIVE PRACTICE: FINANCIAL IDENTITY THEFT $300 AND UNDER    1
DECEPTIVE PRACTICE: FINANCIAL IDENTITY THEFT OVER $ 300        1
DECEPTIVE PRACTICE: FORGERY                                    1
DECEPTIVE PRACTICE: FRAUD OR CONFIDENCE GAME                   1
DECEPTIVE PRACTICE: ILLEGAL USE CASH CARD                      1
NARCOTICS: FOUND SUSPECT NARCOTICS                             1
OFFENSE INVOLVING CHILDREN: CHILD ABDUCTION                    1
OFFENSE INVOLVING CHILDREN: OTHER OFFENSE                      1
OTHER OFFENSE: ANIMAL ABUSE/NEGLECT                            1
OTHER OFFENSE: HARASSMENT BY ELECTRONIC MEANS                  1
OTHER OFFENSE: HARASSMENT BY TELEPHONE                         1
OTHER OFFENSE: OTHER CRIME AGAINST PERSON                      1
OTHER OFFENSE: OTHER CRIME INVOLVING PROPERTY                  1
OTHER OFFENSE: SEX OFFENDER: FAIL REG NEW ADD                  1
OTHER OFFENSE: TELEPHONE THREAT                                1
PUBLIC PEACE VIOLATION: BOMB THREAT                            1
SEX OFFENSE: PUBLIC INDECENCY                                  1
THEFT: ATTEMPT THEFT                                           1
THEFT: FROM BUILDING                                           1
THEFT: RETAIL THEFT                                            1
Name: Category, dtype: int64
```

cluster_2: mostly deceptive practice and other offense.

d) Do the three clusters have different day-of-week trends? Again, visualize the trends for each cluster by creating a line graph and discuss any notable differences. (5%)

In [21]:
```python
#Copy and paste the coding from a) and change the picked column to the columns
DOW = [col for col in data3 if col.startswith('day_')]

#Use the cluster used in part a)
cluster_means_dow = pd.DataFrame()
for column in DOW:
    cluster_means = data3.groupby('cluster')[column].mean()
    cluster_means_dow[column] = cluster_means


cluster_means_dow = cluster_means_dow.reset_index()

plt.figure(figsize=(20, 6))
for i in range(cluster_means_dow.shape[0]):
    plt.plot(cluster_means_dow.columns[1:], cluster_means_dow.iloc[i, 1:], labe

plt.title('Cluster Means for Each DOW')
plt.xlabel('DOW')
plt.ylabel('Mean Proportion of Crimes')
plt.legend()
plt.show()
```



e) Do the three clusters affect different types of communities/neighborhoods? To answer this question, you could first compute the proportions of "cluster 1", "cluster 2", and "cluster 3" crimes for each community area, and identify particular community areas with disproportionate amounts of a given cluster. You can then use the provided file (HW3Q3_2.csv), to determine whether these community areas have any notable common characteristics (poverty, overcrowding, etc.). You may also wish to consult the Chicago Community Areas map at https://en.wikipedia.org/wiki/Community_areas_in_Chicago. (5%)¶

In [22]:
```
community_columns = [col for col in data3 if col.startswith('community_area_')]
community_df  = data3[community_columns]
community_df['cluster']= data3['cluster']
```

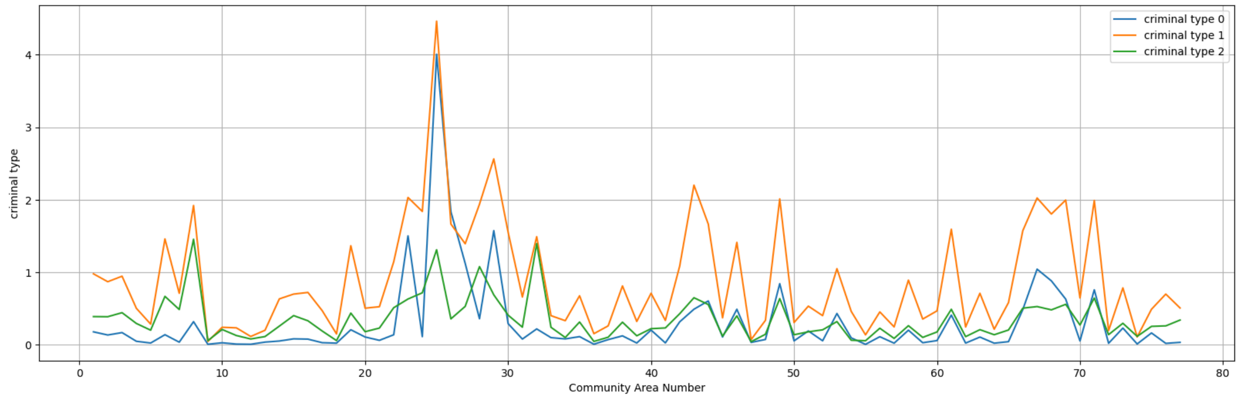
```
/var/folders/qp/9y56mfxx3zq2c_cjbvf9xg_w0000gn/T/ipykernel_71421/2408059473.p
y:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  community_df['cluster']= data3['cluster']
```

In [23]:
```
proportions = community_df.groupby(['cluster']).sum()
proportions.index = ['Alpha', 'Beta', 'Gamma']
proportions = proportions.T
```

In [24]:
```
data4['cluster_0'] = proportions['Alpha'].values
data4['cluster_1'] = proportions['Beta'].values
data4['cluster_2'] = proportions['Gamma'].values
```

In [25]:
```
plt.figure(figsize=(20, 6))
plt.plot(data4['Community Area Number'], data4['cluster_0'], label='criminal ty
plt.plot(data4['Community Area Number'], data4['cluster_1'], label='criminal ty
plt.plot(data4['Community Area Number'], data4['cluster_2'], label='criminal ty

plt.xlabel('Community Area Number')
plt.ylabel('criminal type')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



f) How well do the three groups formed by clustering hour-of-day trends correspond to the FBI's division between P1V, P1P, and P2 crimes? (5%)

In [26]:
```
cluster_0
```

Out[26]:

	crime_type	Category	day_Sun	day_Mon	day_Tue	day_Wed	day_Thu	day_
39	DECEPTIVE PRACTICE: COUNTERFEITING DOCUMENT	P2	0.136691	0.125899	0.131894	0.141487	0.134293	0.169
48	GAMBLING: GAME/DICE	P2	0.150000	0.138889	0.183333	0.100000	0.144444	0.177
50	INTERFERENCE WITH PUBLIC OFFICER: OBSTRUCTING ...	P2	0.161710	0.167286	0.139405	0.115242	0.137546	0.139
57	NARCOTICS: MANU/DEL:CANNABIS 10GM OR LESS	P2	0.107438	0.168044	0.140496	0.143251	0.154270	0.173
58	NARCOTICS: MANU/DEL:CANNABIS OVER 10 GMS	P2	0.106618	0.150735	0.183824	0.161765	0.113971	0.187
59	NARCOTICS: MANU/DELIVER: HEROIN (WHITE)	P2	0.133197	0.133197	0.176230	0.094262	0.145492	0.182
60	NARCOTICS: MANU/DELIVER:CRACK	P2	0.109489	0.167883	0.109489	0.145985	0.160584	0.197
62	NARCOTICS: POSS: CANNABIS 30GMS OR LESS	P2	0.126085	0.145738	0.156457	0.146503	0.152374	0.148
63	NARCOTICS: POSS: CANNABIS MORE THAN 30GMS	P2	0.097561	0.128049	0.143293	0.167683	0.167683	0.164
65	NARCOTICS: POSS: CRACK	P2	0.120347	0.131514	0.135236	0.156328	0.150124	0.151
66	NARCOTICS: POSS: HEROIN(WHITE)	P2	0.130026	0.167102	0.149869	0.137337	0.154047	0.130
67	NARCOTICS: POSS: PCP	P2	0.134615	0.134615	0.134615	0.115385	0.147436	0.173
68	NARCOTICS: POSS: SYNTHETIC DRUGS	P2	0.095000	0.115000	0.170000	0.190000	0.140000	0.155
69	NARCOTICS: POSSESSION OF DRUG EQUIPMENT	P2	0.095808	0.149701	0.173653	0.155689	0.149701	0.131
70	NARCOTICS: SOLICIT NARCOTICS ON PUBLICWAY	P2	0.134615	0.158654	0.139423	0.177885	0.149038	0.129
77	OTHER OFFENSE: GUN OFFENDER: ANNUAL REGISTRATION	P2	0.143885	0.172662	0.187050	0.129496	0.107914	0.129
78	OTHER OFFENSE: GUN OFFENDER: DUTY TO REGISTER	P2	0.048077	0.240385	0.192308	0.144231	0.115385	0.144
81	OTHER OFFENSE: LICENSE VIOLATION	P2	0.070664	0.072805	0.177730	0.207709	0.194861	0.156

	crime_type	Category	day_Sun	day_Mon	day_Tue	day_Wed	day_Thu	day_
86	OTHER OFFENSE: PAROLE VIOLATION	P2	0.119578	0.168816	0.162954	0.144197	0.109027	0.147
88	OTHER OFFENSE: SEX OFFENDER: FAIL TO REGISTER	P2	0.078652	0.112360	0.207865	0.162921	0.157303	0.174
92	PROSTITUTION: SOLICIT ON PUBLIC WAY	P2	0.088608	0.009845	0.182841	0.185654	0.196906	0.156
94	PUBLIC PEACE VIOLATION: RECKLESS CONDUCT	P2	0.135040	0.149162	0.152692	0.151809	0.131509	0.133
116	WEAPONS VIOLATION: UNLAWFUL POSS OF HANDGUN	P2	0.142334	0.133388	0.150468	0.145181	0.125661	0.149

23 rows × 111 columns

In [27]:
```
c_counts0 = len(cluster_0)/(cluster_0['Category'].value_counts())
c_counts0
```

Out[27]:
```
P2    1.0
Name: Category, dtype: float64
```

In [28]:
```
c_counts1 = len(cluster_1)/(cluster_1['Category'].value_counts())
c_counts1
```

Out[28]:
```
P2     2.029412
P1V    2.875000
P1P    6.272727
Name: Category, dtype: float64
```

In [29]:
```
c_counts2 = len(cluster_2)/(cluster_2['Category'].value_counts())
c_counts2
```

Out[29]:
```
P2     1.173913
P1P    6.750000
Name: Category, dtype: float64
```

g) For part g, you will use the same dataset to compare the clusters produced by several different methods. But this time you should cluster using only the *day-of-week* (not hour-of-day) attributes (day_Sun..day_Sat). Please perform four different clusterings using (i) k-means, (ii) Gaussian mixture models, (iii) Bottom-up hierarchical clustering with "single link" distance metric, and (iv) Bottom-up hierarchical clustering with "complete link" distance metric. In each case, you should choose the number of clusters using the silhouette method (or another established method of your choice- please specify). For each clustering, report the number of clusters formed and the number of elements in each cluster. You should also identify any notable similarities or differences between the clusterings. (20%)

In [30]:
```
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm
import numpy as np
data_km = data3.copy()
columns_to_cluster = [col for col in data_km if col.startswith('day_')]
```

```
Km = []

for n_clusters in range(2,11):
    km = KMeans(n_clusters=n_clusters, random_state=0)
    cluster_labels = km.fit_predict(data_km[columns_to_cluster])
    silhouette_avg = silhouette_score(data_km[columns_to_cluster], cluster_labe
    Km.append({'number of clusters': n_clusters, 'silhouette_score': silhouett

#match the number of cluster and score
Km = pd.DataFrame(Km)

best_cluster = Km.loc[Km['silhouette_score'].idxmax()]

# Re-fit
best_n_clusters = int(best_cluster['number of clusters'])
kmeans_cl = KMeans(n_clusters=best_n_clusters, random_state=0)
fit_kmeans = kmeans_cl.fit(data_km[columns_to_cluster])

data_km['Km_cluster'] = fit_kmeans.labels_

print(best_cluster)
print(data_km['Km_cluster'].value_counts())
```

```
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/fengcharles/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kme
ans.py:1412: FutureWarning: The default value of `n_init` will change from 10
to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
number of clusters    2.000000
silhouette_score      0.234967
Name: 0, dtype: float64
0    63
1    56
Name: Km_cluster, dtype: int64
```

In [31]:

```python
from sklearn.mixture import GaussianMixture

data_gmm = data3.copy()
columns_to_cluster = [col for col in data_gmm if col.startswith('day_')]
Gmm = []

for n_clusters in range(2,11):
    gmm = GaussianMixture(n_components=n_clusters, random_state=0)
    cluster_labels = gmm.fit_predict(data_gmm[columns_to_cluster])

    # Calculate the silhouette score
    silhouette_avg = silhouette_score(data_gmm[columns_to_cluster], cluster_lal
    Gmm.append({'number of clusters': n_clusters, 'silhouette_score': silhouet
```

```python
#match the number of cluster and score
Gmm = pd.DataFrame(Gmm)

best_cluster = Gmm.loc[Gmm['silhouette_score'].idxmax()]

# Re-fit
best_n_clusters = int(best_cluster['number of clusters'])
gmm_cl = GaussianMixture(n_components=best_n_clusters, random_state=0)
fit_gmm = gmm_cl.fit(data_gmm[columns_to_cluster])

data_gmm['cluster'] = fit_gmm.predict(data_gmm[columns_to_cluster])

print(best_cluster)
print(data_gmm['cluster'].value_counts())
```

```
number of clusters    2.000000
silhouette_score      0.256727
Name: 0, dtype: float64
1    89
0    30
Name: cluster, dtype: int64
```

In [32]:

```python
from sklearn.cluster import AgglomerativeClustering #library from chatgpt
from sklearn.metrics import silhouette_score
import pandas as pd

data_Agglom = data3.copy()
columns_to_cluster = [col for col in data_Agglom if col.startswith('day_')]
Agglom = []

for n_clusters in range(2, 11):
    agglom = AgglomerativeClustering(n_clusters=n_clusters, linkage='single')
    cluster_labels = agglom.fit_predict(data_Agglom[columns_to_cluster])
    silhouette_avg = silhouette_score(data_Agglom[columns_to_cluster], cluster_
    Agglom.append({'number of clusters': n_clusters, 'silhouette_score': silhou

# Convert list to DataFrame
Agglom = pd.DataFrame(Agglom)
best_cluster = Agglom.loc[Agglom['silhouette_score'].idxmax()]

# Re-fit
best_n_clusters = int(best_cluster['number of clusters'])
agglom_cl = AgglomerativeClustering(n_clusters=best_n_clusters, linkage='single

cluster_labels = agglom_cl.fit_predict(data_Agglom[columns_to_cluster])


data_Agglom['cluster'] = cluster_labels

print(best_cluster)
print(data_Agglom['cluster'].value_counts())
```

```
number of clusters    2.000000
silhouette_score      0.637765
Name: 0, dtype: float64
0    118
1      1
Name: cluster, dtype: int64
```

In [33]:
```
from sklearn.cluster import AgglomerativeClustering #library from chatgpt
from sklearn.metrics import silhouette_score
import pandas as pd

data_Agglom = data3.copy()
columns_to_cluster = [col for col in data_Agglom if col.startswith('day_')]
Agglom = []

for n_clusters in range(2, 11):
    agglom = AgglomerativeClustering(n_clusters=n_clusters, linkage='complete'
    cluster_labels = agglom.fit_predict(data_Agglom[columns_to_cluster])
    silhouette_avg = silhouette_score(data_Agglom[columns_to_cluster], cluster_
    Agglom.append({'number of clusters': n_clusters, 'silhouette_score': silhou

# Convert list to DataFrame
Agglom = pd.DataFrame(Agglom)
best_cluster = Agglom.loc[Agglom['silhouette_score'].idxmax()]

# Re-fit
best_n_clusters = int(best_cluster['number of clusters'])
agglom_cl = AgglomerativeClustering(n_clusters=best_n_clusters, linkage='comple

cluster_labels = agglom_cl.fit_predict(data_Agglom[columns_to_cluster])


data_Agglom['cluster'] = cluster_labels

print(best_cluster)
print(data_Agglom['cluster'].value_counts())
```

```
number of clusters    2.000000
silhouette_score      0.610065
Name: 0, dtype: float64
0    117
1      2
Name: cluster, dtype: int64
```

note:

For the clustering by kmeans, the cluster amount is relatively more average, each type of cluster have similar amount.

The gaussian matrix is similar to the kmeans, it keep the average amount for each cluster.

The single and complete bottom line cluster the data in significant apart amount, and the silhouette score for these two method is relatively higher than other two.

In [ ]: