Homework 8 - CS348 Spring 2019

Description - In this assignment, you will analyze the statistical properties of causal graphical models, and infer the effects of interventions using observational data.

Getting Started - You should complete the assignment using your own installation of Python 3 and the packages numpy, pandas, scipy, matplotlib, and seaborn. Download the assignment from Moodle and unzip the file. This will create a directory with this file, 'HW08.ipynb'.

Deliverables - The assignment has a single deliverable: this jupyter notebook file saved as a pdf. Please answer all coding and writing questions in the body of this file. Once all of the answers are complete, download the file by navigating the following menus: File -> Download as -> PDF via LaTeX. Submit the downloaded pdf file on gradescope. Alternatively, you can save the file as a pdf via the following: File -> Print Preview -> Print as pdf.

Data Sets - In this assignment, you will find one dataset in the `data` folder, `synthetic.csv`.

Academic Honesty Statement - Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. Sharing your solutions with other students is considered cheating. Posting your code to public repositories such as GitHub is also considered cheating. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

This academic honesty statement does not restrict you from reading official documentation or using other web resources for understanding the syntax of python, related data science libraries, or properties of distributions.

In [1]:
```
# Do not import any other libraries other than those listed here.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as skl
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import log_loss
```

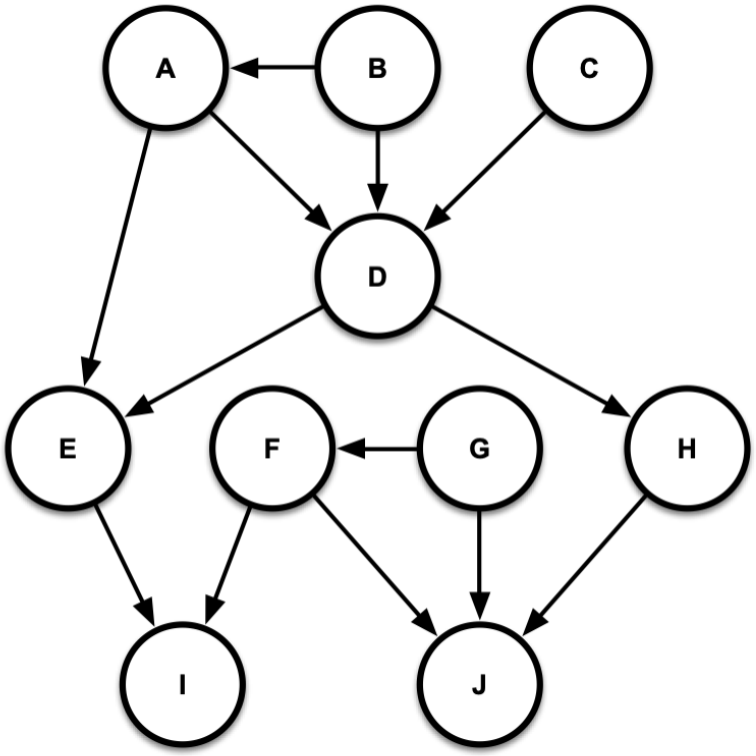
Problem 1 - Causal Graphical Models

In this problem you will analyze the statistical implications of various causal graphical model structures. You will also implement and use binary independence tests to differentiate between several candidate causal models for a synthetic dataset.

Part 1 (15 points)

Determine whether each of the following independence statements are implied by the graphical model below. For each independence statement that is not satisfied, describe one of the d-connecting path in the directed acyclic graph.

$A \perp J$
$A \perp C$
$B \perp F$
$B \perp F|J$
$A \perp C|H$

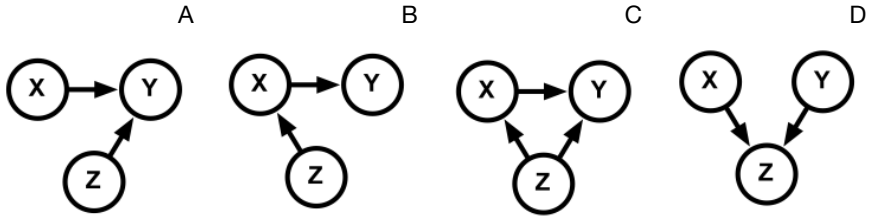


Part 1 Written Response

1. A is not conditionally independent of J since A-D-H-J is a d-connecting path.
2. A is conditionally independent of C.
3. B is conditionally independent of F.
4. B is not conditionally independent of F given J, since B-D-H-J-F is a d-connecting path.
5. A is not conditionally independent of C given H, since A-D-C is a d-connecting path.

Part 2 (20 points)

Your colleague has gathered a set of observational data from a data generating process over three variable X, Y, and Z. They know that the data generating process can be described using one of the following four causal graphical models, but they aren't sure which one. What is the maximum number of independence tests you would need to run in order to uniquely determine which causal graphical model describes the data generating process? Describe this sequence of independence tests, as well as their implications.



Hint: You may find it helpful to list the independence facts implied by each of the four causal graphical models.

Part 2 Written Response

The maximum number of independence tests is 3.

1. X independent of Z.
2. X independent of Y.
3. Z conditionally independent of Y given X.

Part 3 (15 points)

Write a function `marginal_independence(data, var_1, var_2)`, which takes as input a pandas dataframe, `data`, and two strings describing the variable names of the two variables that are being tested for conditional independence, `var_1` and `var_2`. This function should return `True` if `var_1` and `var_2` are independent and `False` otherwise. Use your function on `independent_data` and `dependent_data` generated below.

Note 1: Recall that two variables X and Y are marginally independent if P(Y) = P(Y|X) for all possible values of X. Rather than computing the typical statistics for hypothesis testing, for this assignment you can instead assume that the two variables are independent if |P(Y) - P(Y|X)| < 0.01.

Note 2: You can assume that `data` only contains binary random variables.

In [2]:
```
def independent_dgp():
    X = np.random.binomial(1, 0.5, 100000)
    Y = np.random.binomial(1, 0.5, 100000)
    return X, Y

def dependent_dgp():
    X = np.random.binomial(1, 0.5, 100000)
    Y = np.random.binomial(1, 0.2 * X + 0.4, 100000)
    return X, Y

independent_data = pd.DataFrame(np.array(independent_dgp()).T, columns=[
"X", "Y"])
dependent_data = pd.DataFrame(np.array(dependent_dgp()).T, columns=["X",
"Y"])
```

In [3]:
```
# Part 3 Solution

# --- write code here ---

def marginal_independence(data, var_1, var_2):
    size = len(data[var_2])
    cy1 = 0
    cy0 = 0
    for i in data[var_2]:
        if i == 1:
            cy1 += 1
        else:
            cy0 += 1
    py1 = cy1/size
    py0 = cy0/size

    cx1 = 0
    cx0 = 0
    for j in data[var_1]:
        if j == 1:
            cx1 += 1
        else:
            cx0 += 1
    px1 = cx1/size
    px0 = cx0/size

    cy0x0 = 0
    cy0x1 = 0
    cy1x0 = 0
    cy1x1 = 0
    for x,y in zip(data[var_1], data[var_2]):
        if x == 0 and y == 0:
            cy0x0 += 1
        elif x == 1 and y == 0:
            cy0x1 += 1
        elif x == 0 and y == 1:
            cy1x0 += 1
        else:
            cy1x1 += 1
    py0x0 = (cy0x0/size)/px0
    py0x1 = (cy0x1/size)/px1
    py1x0 = (cy1x0/size)/px0
    py1x1 = (cy1x1/size)/px1

    if abs(py1-py1x0)<0.01 and abs(py1-py1x1)<0.01 and abs(py0-py0x0)<0.
01 and abs(py0-py0x1)<0.01:
        return True
    return False

# This should return True
print(marginal_independence(independent_data, "X", "Y"))

# This should return False
print(marginal_independence(dependent_data, "X", "Y"))
```

```
True
False
```

Part 4 (15 points)

Write a function `conditional_independence(data, var_1, var_2, cond_var)`, which takes as input a pandas dataframe, `data`, two strings describing the variable names of the two variables that are being tested for conditional independence, `var_1` and `var_2`, and one string describing the conditioning variable, `cond_var`. This function should return `True` if `var_1` and `var_2` are conditionally independent given `cond_var` and `False` otherwise. Use your function `conditionally_independent_data` and `conditionally_dependent_data` generated below.

Note 1: Recall that two variables X and Y are conditionally independent given Z if P(Y|Z) = P(Y|X,Z) for all possible values of X and Z. Rather than computing the typical statistics for hypothesis testing, for this assignment you can instead assume that the two variables are independent if |P(Y|Z) - P(Y|X,Z)| < 0.01.

Note 2: You can assume that `data` only contains binary random variables.

Hint: While it's not necessary to complete this assignment, you may find it helpful to use your implementation of `marginal_independence` in your implementation of `conditional_independence`.

In [4]:
```
def conditionally_independent_dgp():
    # X and Z are independence given Y.
    X = np.random.binomial(1, 0.5, 100000)
    Y = np.random.binomial(1, 0.2 * X + 0.4, 100000)
    Z = np.random.binomial(1, 0.4 * Y + 0.3, 100000)
    return X, Y, Z

def conditionally_dependent_dgp():
    # X and Z are dependence given Y.
    X = np.random.binomial(1, 0.5, 100000)
    Y = np.random.binomial(1, 0.2 * X + 0.4, 100000)
    Z = np.random.binomial(1, 0.2 * Y + 0.3 + X * 0.2, 100000)
    return X, Y, Z

conditionally_independent_data = pd.DataFrame(np.array(conditionally_ind
ependent_dgp()).T, columns=["X", "Y", "Z"])
conditionally_dependent_data = pd.DataFrame(np.array(conditionally_depen
dent_dgp()).T, columns=["X", "Y", "Z"])
```

In [5]:
```
# Part 4 Solution

# --- write code here ---

def conditional_independence(data, var_1, var_2, cond_var):
    size = len(data[var_1])
    cy1z0, cy1z1, cy0z0, cy0z1 = 0, 0, 0, 0
    cz1, cz0 = 0, 0

    cy1x0z0, cy1x1z0, cy1x0z1, cy1x1z1 = 0,0,0,0
    cy0x0z0, cy0x1z0, cy0x0z1, cy0x1z1 = 0,0,0,0
    cx0z0, cx0z1, cx1z0, cx1z1 = 0, 0, 0, 0
    for x, y, z in zip(data[var_1], data[var_2], data[cond_var]):
        if x == 1 and y == 1 and z == 0:
            cy1z0 += 1
            cz0 += 1
            cy1x1z0 += 1
            cx1z0 += 1
        elif x == 0 and y == 1 and z == 1:
            cy1z1 += 1
            cz1 += 1
            cy1x0z1 += 1
            cx0z1 += 1
        elif x == 0 and y == 1 and z == 0:
            cy1z0 += 1
            cz0 += 1
            cy1x0z0 += 1
            cx0z0 += 1
        elif x == 1 and y == 1 and z == 1:
            cy1z1 += 1
            cz1 += 1
            cy1x1z1 += 1
            cx1z1 += 1
        elif x == 0 and y == 0 and z == 0:
            cy0z0 += 1
            cz0 += 1
            cy0x0z0 += 1
            cx0z0 += 1
        elif x == 0 and y == 0 and z == 1:
            cy0z1 += 1
            cz1 += 1
            cy0x0z1 += 1
            cx0z1 += 1
        elif x == 1 and y == 0 and z == 0:
            cy0z0 += 1
            cz0 += 1
            cy0x1z0 += 1
            cx1z0 += 1
        else:
            cy0z1 += 1
            cz1 += 1
            cy0x1z1 += 1
            cx1z1 += 1
    py1z0 = (cy1z0/size)/(cz0/size)
    py1z1 = (cy1z1/size)/(cz1/size)
    py0z0 = (cy0z0/size)/(cz0/size)
```

```
    py0z1 = (cy0z1/size)/(cz1/size)

    py1x0z0 = (cy1x0z0/size)/(cx0z0/size)
    py1x1z0 = (cy1x1z0/size)/(cx1z0/size)
    py1x0z1 = (cy1x0z1/size)/(cx0z1/size)
    py1x1z1 = (cy1x1z1/size)/(cx1z1/size)

    py0x0z0 = (cy0x0z0/size)/(cx0z0/size)
    py0x1z0 = (cy0x1z0/size)/(cx1z0/size)
    py0x0z1 = (cy0x0z1/size)/(cx0z1/size)
    py0x1z1 = (cy0x1z1/size)/(cx1z1/size)

    if abs(py1z0-py1x0z0)<0.01 and abs(py1z0-py1x1z0)<0.01 and abs(py1z1
-py1x0z1)<0.01 and abs(py1z1-py1x1z1)<0.01 and abs(py0z0-py0x0z0)<0.01 a
nd abs(py0z0-py0x1z0)<0.01 and abs(py0z1-py0x0z1)<0.01 and abs(py0z1-py0
x1z1)<0.01:
            return True
    return False
# This should return True
print(conditional_independence(conditionally_independent_data, "X", "Z",
"Y"))

# This should return False
print(conditional_independence(conditionally_dependent_data, "X", "Z",
"Y"))
```

```
True
False
```

Part 5 (10 points)

Using your implementations of `marginal_independence` and `conditional_independence` and your answer from part 2, determine which of the four causal graphical model structures corresponds to the data generating process used to produce the data in the file labeled "synthetic.csv" in the "data" folder.

Note: Your solution should show the results of all of the indepedence tests you describe in part 2. Make sure that you specify which model is the correct one in your written response.

In [6]:
```
# Part 5 Solution

data = pd.read_csv('data/synthetic.csv')

# --- write code here ---
if not marginal_independence(data, "X", "Z"):
    print('X is not independent of Z')
if not marginal_independence(data, "X", "Y"):
    print('X is not independent of Y')
if not conditional_independence(data, "X", "Z", "Y"):
    print('Z is not conditionally independent of Y given X')
```

```
X is not independent of Z
X is not independent of Y
Z is not conditionally independent of Y given X
```

Part 5 Written Response

The correct model is model C.

Problem 2 - Effect Estimation

In this problem you will implement a propensity score reweighting to estimate the effect of a diamond's cut on its price.

In [12]:
```
data = sns.load_dataset('diamonds')
data = data[(data['cut'] == 'Ideal') | (data['cut'] == 'Premium')]

treatment = (data['cut'] == "Ideal") * 1
outcome = data['price'] > 1000
covariates = np.array(data.drop(['color', 'clarity', 'cut', 'price'], ax
is=1))
```

Part 1 (10 points)

For all values of `max_depth` between 1 and 15, train a random forest classifier to predict the value of the treatment variable in the diamonds dataset, the cut of the diamond. Your model should use scikit-learn's `RandomForestClassifier` with `n_estimators=20` and `random_state=0`. Among these models, select the model that achieves the highest 10-fold cross-validation score using the negative log loss score.

Compare the score of your selected model against the negative log loss using 3 baseline models.

(1) $P(cut = 1|covariates) = 1$, regardless of the value of the covariates.

(2) $P(cut = 1|covariates) = 0$, regardless of the value of the covariates.

(3) $P(cut = 1|covariates) = p$, regardless of the value of the covariates. p is the proportion of instances in the dataset where $cut = 1$.

In [20]:
```
# Part 1 Solution

# --- write code here ---
p1, p0 = treatment.value_counts(normalize=True)

base1 = (data['cut']=="Ideal")*1 + (data['cut']=="Premium")*1
base2 = (data['cut']=="Ideal")*0 + (data['cut']=="Premium")*0
base3 = (data['cut']=="Ideal")*p1 + (data['cut']=="Premium")*p1

maxi = 100
max_index = 0
for dep in range(1, 16):
    model = RandomForestClassifier(n_estimators=20, random_state=0, max_
depth=dep)
    result = cross_val_score(model, covariates, treatment, cv=10, scorin
g='neg_log_loss').mean()
    if result > -maxi:
        maxi = -result
        max_index = dep

model = RandomForestClassifier(n_estimators=20, random_state=0, max_dept
h=max_index)
result1 = skl.metrics.log_loss(treatment, base1)
result2 = skl.metrics.log_loss(treatment, base2)
result3 = skl.metrics.log_loss(treatment, base3)
print("Score of selected model: "+ str(maxi))
print("Comparing with baseline1: "+str(result1))
print("Comparing with baseline2: "+str(result2))
print("Comparing with baseline3: "+str(result3))
```

```
Score of selected model: 0.2758071359455251
Comparing with baseline1: 13.477881628384727
Comparing with baseline2: 21.061206781922927
Comparing with baseline3: 0.6688444330220407
```

Part 2 (15 points)

Using the model you selected in part 1, estimate the average treatment effect of a diamond's cut being Ideal (as opposed to Premium) on the probability that its price is greater than $1000. Compare this result to an estimate that assumes that the covariates are indepdendent of the treatment. Explain how and why these estimates differ in 3-4 sentences. Propose a hypothesis for why these two estimates differ in terms of the problem domain.

Note: The data has been transformed so that both treatment and outcome are binary variables. Inspect the data processing in the cell above part 1 for details.

In [21]:
```
# Part 2 Solution

# --- write code here ---
model = RandomForestClassifier(n_estimators=20, random_state=0, max_dept
h=max_index)


count = 0
ctc1, ctc0 = 0, 0
c1, c0 = 0, 0
for cut, price in zip(treatment, outcome):
    if cut == 1 and price == True:
        ctc1 += 1
        c1 += 1
    elif cut == 0 and price == True:
        ctc0 += 1
        c0 += 1
    elif cut == 1 and price == False:
        c1 += 1
    elif cut == 0 and price == False:
        c0 += 1
ptc1 = (ctc1/len(outcome))/(c1/len(outcome))
ptc0 = (ctc0/len(outcome))/(c0/len(outcome))
print("Average treatment effect: "+str(ptc1-ptc0))
```

```
Average treatment effect: -0.0855716517139149
```

Part 2 Written Response

Type your written response here