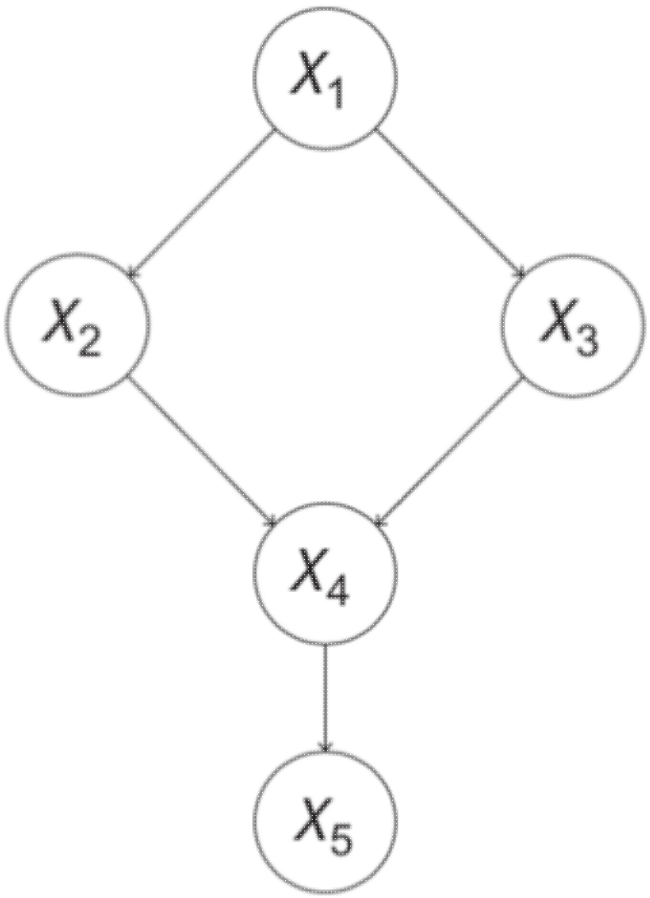
In [2]:

```
import numpy as np
import pandas as pd
from scipy.special import expit
from sklearn.linear_model import LogisticRegression
from keras.layers import Dense, Input
from keras.models import Model
```

In [3]:

```
N = 100000
inv_logit = expit
x1 = np.random.binomial(1, p=0.5, size=N) #Generating sunny/or not sunny days
x2 = np.random.binomial(1, p=inv_logit(-3.*x1)) #Sprinkler on/off depend on wether it's su
x3 = np.random.binomial(1, p=inv_logit(3.*x1)) #Rainy yes/no depend on wether it's sunny o
x4 = np.bitwise_or(x2, x3) #Sidewalk is wet if it's raining or if sprinklers are on
x5 = np.random.binomial(1, p=inv_logit(3.*x4)) #Slippery or not depends on wether or not s
X = pd.DataFrame({'$x_1$': x1, '$x_2$': x2, '$x_3$': x3, '$x_4$': x4, '$x_5$': x5})
```

The causal graph of the data generated above is the following:



Correlation matrix of variables in the system. Notice the negative relationship between x1 and x2, as well as between x2 and x3. This indicate that when the weather is good, the sprinkler is turned on and when it rains the sprinkler is turned off.

In [4]:

```
X.corr()
```

Out[4]:

	x_1	x_2	x_3	x_4	x_5
x_1	1.000000	-0.508249	0.499949	0.282120	0.140874
x_2	-0.508249	1.000000	-0.254979	0.255050	0.130049
x_3	0.499949	-0.254979	1.000000	0.678809	0.337129
x_4	0.282120	0.255050	0.678809	1.000000	0.501251
x_5	0.140874	0.130049	0.337129	0.501251	1.000000

The following shows the naive conditional expectation values for wether the sidewalk is wet given that the sprinkler is on.

In [5]:

```
X.groupby('$x_2$').mean()[['$x_5$']]
```

Out[5]:

	x_5
x_2	
0	0.860620
1	0.953285

The expected value of a variable X, denoted E(X), is found by multiplying each possible value of the variable by the probability that the variable will take that value, then summing the products:

$$ \begin{equation*} E(X) = \sum_xxP(X=x) \end{equation*} $$

Expected value with conditionals: $$ \begin{equation*} E(Y|X=x) = \sum_yyP(Y=y|X=x) \end{equation*} $$ **Note: Lose the probability distribution equation below, you do not show it in your code. Use conditional expected value above and Baye's rule (eq 1.8) in conjunction with your dataset to further detail expected value calculation.

The expected value is found by multiplying each possible value of a variable by the probability that the variable will take that value. Therefore to compute $E(Y|X=x)$, we would be required to first know the probability of Y taking on the value y conditional on the variable X taking on the value x. That is:

$$ P(Y=y|X=x) $$

Or more specifically, in our case:

$$ P(X_5=x_5|X_2=x_2) $$

Based on the causal graph above, this computation cannot be performed without taking into account the other variables that are also causes for X_5 Therefore; using the multiplication rule, the probability distribution of X_5 conditional on the observed value of X_2 is expressed as:

$$ \begin{equation*} P(X_1, X_2, X_3, X_4, X_5|X_2) =
P(X_5|X_4)P(X_4|X_2,X_3)P(X_3|X_1)P(X_2|X_1)P(X_1) \end{equation*} $$

Computing the expression above would be a hassle despite only having 5 variables. To illustrate this, let's take only the first term of the multiplication: $P(X_5|X_4)$. This term can be calculated as:

let's look at $X_5=1$ $$ P(X_5=1|X_4) = P(X_5=1|X_4=1)P(X_4=1) + P(X_5=1|X_4=0)P(X_4=0) $$

This would be repeated for the 4 remaining terms in the multiplication; from $P(X_4|X_3, X_2)$ to $P(X_1)$.

The neat thing about having such a well defined Structural Causal Model is that we do not need to do all these calculaitons. Since we know the variables and the functions that relate them we can simply look at the subset of our dataset conditioned on X_2. That is to only look at the samples in the population that have a particular value of X_2

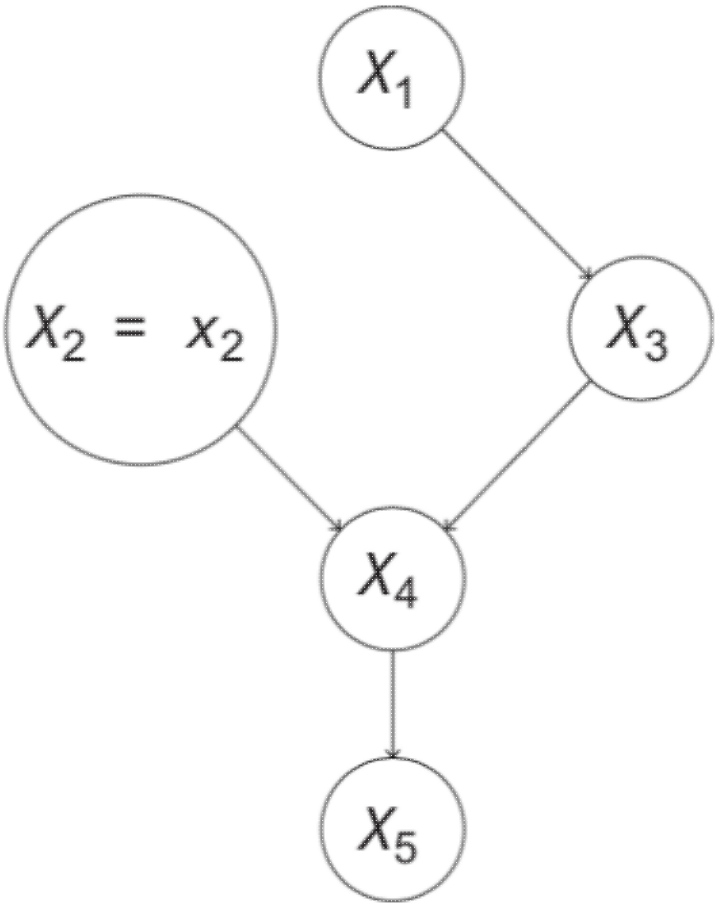
We wish to know the effect that sprinkler (on/off) has on how slippery the sidewalk is. This is the difference in the expected value of X_5 given the covariate X_2: $$ \begin{equation*} \sigma_{naive} = E[X_5 | X_2 = 1] - E[X_5 | X_2 = 0]\\ \sigma_{naive} = 0.95 - 0.86\\ \sigma_{naive} = 0.09 \end{equation*} $$ The sidewalk is 9 percentage points more likely to be slippery given that the sprinkler was on.

However this result does not show the true effect of X_2 on X_5, because of X_2's dependence on X_1 in this dataset. So we create the following dataset:

In [6]:
```
x1 = np.random.binomial(1, p=0.5, size=N)

x2_0 = np.random.binomial(1, p=0, size=N)
x2_1 = np.random.binomial(1, p=1, size=N)

x3 = np.random.binomial(1, p=inv_logit(3.*x1))

x4_0 = np.bitwise_or(x2_0, x3)
x4_1 = np.bitwise_or(x2_1, x3)

x5_0 = np.random.binomial(1, p=inv_logit(3.*x4_0))
x5_1 = np.random.binomial(1, p=inv_logit(3.*x4_1))

Xn = pd.DataFrame({'$x2_0$': x2_0, '$x2_1$': x2_1, '$x5_0$': x5_0, '$x5_1$': x5_1})
Xn['$x5_1$'].mean() - Xn['$x5_0$'].mean()
```

Out[6]:
```
0.12287000000000003
```

The dataset above is interesting because it elicits the power and usefulness of interventions. In this dataset the value of X_2 is independent from X_1. The causal link between X_1 and X_2 is broken, therefore the probability distributions on descendent variables of X_1 have also changed, as they no longer depend on X_1 (These descendent variables include X_2, X_4 and X_5). This dataset in effect depicts a different system from the first one where we simply observed X_2. The causal graph of the new system is the following:



Here the intervention alters the dependecies in the graph. The joint distribution after this intervention is the following: $$ \begin{equation*} P(X_5=x_5|do(X_2=x_2)) = \sum_{x_1}P(X_5=x_5|X_2=x_2,X_1=X_1)P(X_1=x_1) \end{equation*} $$ This adjustment formula computes the association between X_5 and X_2 for each value x_1 of X_1, then averages over those values. This is an adjustment for X_1; because X_1 is the only parent of X_2, which is also a cause of X_5. Notice that the expression above is an application of the more general Robins G-Formula: $$ \begin{equation*} P(X_j|do(X_i = x_i)) = \sum_Z P(X_j|X_i, Z)P(Z) \end{equation*} $$

In the result above we now get that the sidewalk in 12 percentage points more likely to be slippery given that the sprinkler was on. Let's now attempt to predict the effect of X_2 (sprinkler on/off) on X_5 (sidewalk slippery/or not). We can use a regression model such that: $$ \begin{equation} X_5 = \beta_0 + \beta_1X_2 \end{equation} $$ In this model we are doing the regression of X_5: (sidewalk slippery/or not), against the covariate X_2: (sprinkler on/off).

In [7]:

```
# build our model, predicting $x_5$ using $x_2$
model = LogisticRegression()
model = model.fit(X[['$x_2$']], X['$x_5$'])

# what would have happened if $x_2$ was always 0:
X0 = X.copy()
X0['$x_2$'] = 0 #This procedure is equivalent to d-sperating X2 from X1 as shown in the gr
y_pred_0 = model.predict_proba(X0[['$x_2$']])
```

```
# what would have happened if $x_2$ was always 1:
X1 = X.copy()
X1['$x_2$'] = 1
y_pred_1 = model.predict_proba(X1[['$x_2$']])

# now, let's check the difference in probabilities
y_pred_1[:, 1].mean() - y_pred_0[:,1].mean()
```

Out[7]:

```
0.09260476253319183
```

Result above is the same 0.09 difference as σ_{naive}. We wish to build a model that predicts an effect that is closer to the true effect. In order to this, we must take into consieration that X_2 depends on X_1: (Sunny day/not sunny day) for its value. X_1 is therefore a cause of X_2. By performing a regression on both covariates we can account for more on the variation in X_5. The new regression model therefore is the following: $$ \begin{equation} X_5 = \beta_0 + \beta_1X_2 + \beta_2X_1 \end{equation} $$

In [17]:

```
model = LogisticRegression()
model = model.fit(X[['$x_2$', '$x_1$']], X['$x_5$'])

# what would have happened if $x_2$ was always 0:
X0 = X.copy()
X0['$x_2$'] = 0 #This procedure is equivalent to d-sperating X2 from X1 as shown in the gr
y_pred_0 = model.predict_proba(X0[['$x_2$', '$x_1$']])

# what would have happened if $x_2$ was always 1:
X1 = X.copy()
X1['$x_2$'] = 1

# now, let's check the difference in probabilities
y_pred_1 = model.predict_proba(X1[['$x_2$', '$x_1$']])
y_pred_1[:, 1].mean() - y_pred_0[:,1].mean() #Calculating the average causal effect
```

Out[17]:

```
0.14194231684053804
```

Regression on both X_1 and X_2 gives an estimate difference that is closer to the true effect of X_2. In fact, it gives us the Average Causal Effect (ACE) which is expressed as: $$ \begin{equation} ACE = P(X_{5_{1}}|do(X_2=1)) - P(X_{5_{0}}|do(X_2=0)) \end{equation} $$ But our regression over-estimated the effect. Let's see if we can do better with a model that is more general than linear regression. We will use a deep feedforward architecture.

In [4]:

```
dense_size = 128
input_features = 2

x_in = Input(shape=(input_features,))
h1 = Dense(dense_size, activation='relu')(x_in)
h2 = Dense(dense_size, activation='relu')(h1)
h3 = Dense(dense_size, activation='relu')(h2)
y_out = Dense(1, activation='sigmoid')(h3)

model = Model(input=x_in, output=y_out)
model.compile(loss='binary_crossentropy', optimizer='adam')
model.fit(X[['$x_1$', '$x_2$']].values, X['$x_5$'])

Epoch 1/1
```

```
100000/100000 [==============================] - 4s 44us/step - loss: 0.3237
```

Out[4]:

```
<keras.callbacks.callbacks.History at 0x7f51140cc2b0>
```

Let's now perform the same prediction procedure as before

In [5]:

```
X_zero = X.copy()
X_zero['$x_2$'] = 0
x5_pred_0 = model.predict(X_zero[['$x_1$', '$x_2$']].values)

X_one = X.copy()
X_one['$x_2$'] = 1
x5_pred_1 = model.predict(X_one[['$x_1$', '$x_2$']].values)

x5_pred_1.mean() - x5_pred_0.mean()
```

Out[5]:

```
0.11946136
```

We see that this result is different from the one given by the logistic regression model. We can now compute how much less likely is the sidewalk to be slippery if we turn off the sprinkler.

In [6]:

```
X['$x_5$'].mean() - x5_pred_0.mean()
```

Out[6]:

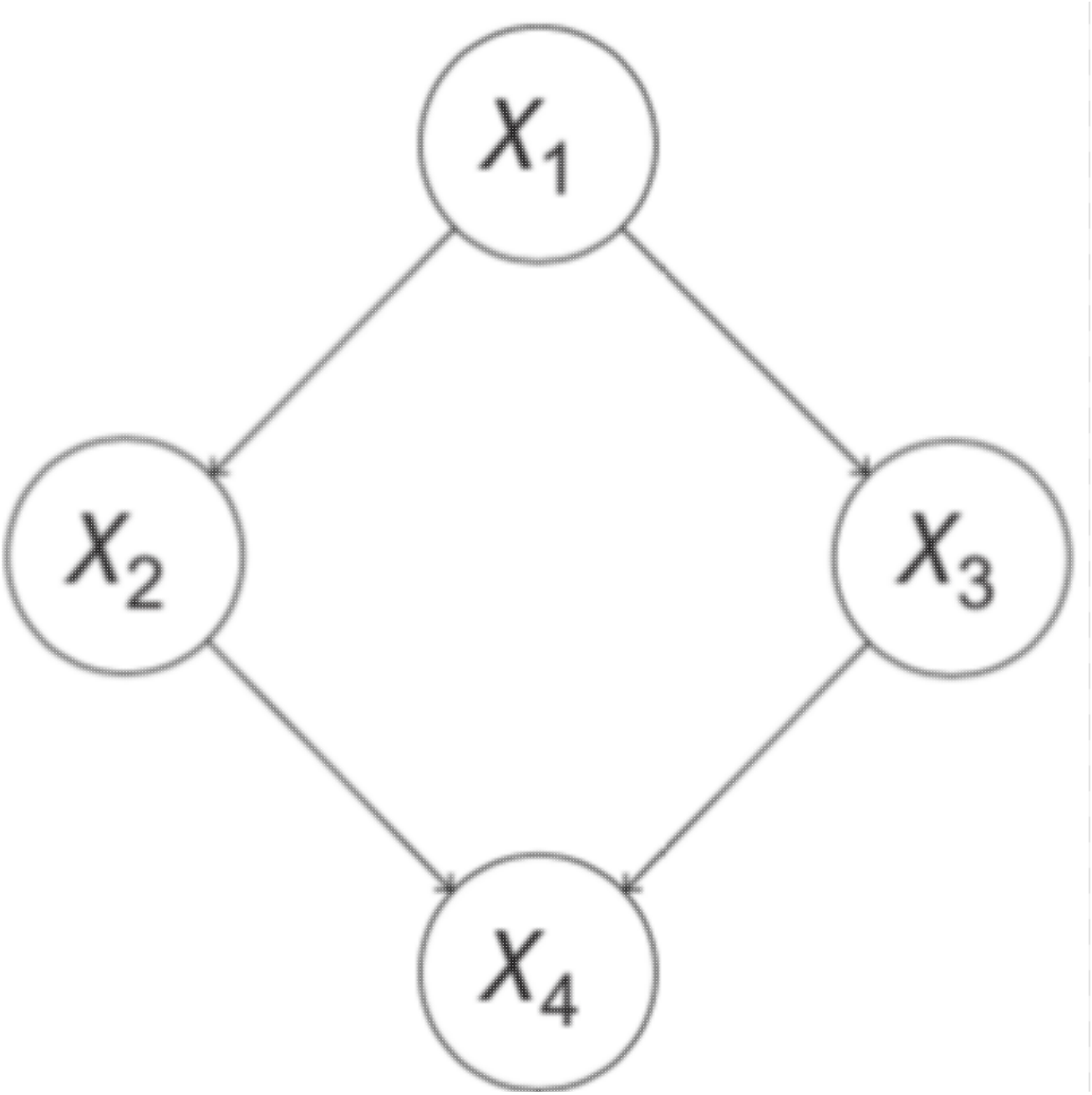
```
0.06620157037734986
```

It will be 7 percent less likely that the sidewalk is slippery if a policy is passed to keep sprinklers off.

Counterfactuals

Consider that we wish to know wether or not the sidewalk would be wet (X_4) on a particular day, had we shut off the sprinkler on that day. Such an operation is called a *counterfactual*. It let's us answer questions of the type "What would Y be in situation U had X been x". Where U are external considerations that affect the value Y and uniquely define the situation. One way to think about the difference between a counterfactual and an intervention is that the counterfactual answers questions at the individual level while intervention is only defined for population level analysis.

We wish to know the value of X_4 in a particular situation, if $X_2 = 0$ in that situation. Our SCM is the following:



This SCM however, is not the complete picture because it does not show the exogeous variables that cause X_4. We need these variables to find the set U of individual factors that uniquely define the situation at hand. In order to find U, we need to know the relationship between all the variables in the SCM. This can be done by estimating the linear equations that relate the variables through a regression. Specifically we wish to know the coefficients and intercepts for: $$ \begin{equation} X_1 = U_{X_1} \\ X_2 = U_{X_2} +aX_1 \\ X_3 = U_{X_3} + bX_1 \\ X_4 = U_{X_4} + cX_2 + dX_3 \end{equation} $$

In [29]:

```
model_x4 = LogisticRegression().fit(X[['$x_2$', '$x_3$']], X['$x_4$'])
model_x3 = LogisticRegression().fit(X[['$x_1$']], X['$x_3$'])
model_x2 = LogisticRegression().fit(X[['$x_1$']], X['$x_2$'])
```

In [69]:

```
score4 = model_x4.score(X[['$x_2$', '$x_3$']], X['$x_4$']) #Coefficient of determination o
```

```
score3 = model_x3.score(X[['$x_1$']], X['$x_3$'])
score2 = model_x2.score(X[['$x_1$']], X['$x_2$'])

print(f"R^2 for X4 eq: {score4}")
print(f"R^2 for X3 eq: {score3}")
print(f"R^2 for X2 eq: {score2}")
```

```
R^2 for X4 eq: 1.0
R^2 for X3 eq: 0.72777
R^2 for X2 eq: 0.727
```

In [36]:

```
Ux2 = model_x2.intercept_[0]
a = model_x2.coef_[0][0]

Ux3 = model_x3.intercept_[0]
b = model_x3.coef_[0][0]

Ux4 = model_x4.intercept_[0]
c = model_x4.coef_[0][0]
d = model_x4.coef_[0][1]

print(f" Ux2 = {Ux2}, a = {a}")
print(f" Ux3 = {Ux3}, b = {b}")

print(f" Ux4 = {Ux4}, c = {c}, d = {d}")
```

```
Ux2 = 0.003765389626754025, a = -2.997864127119259
 Ux3 = 0.019942438130488172, b = 2.9230582044638864
 Ux4 = -6.272693828941369, c = 13.13176894896531, d = 14.560563273568548
```

With the U factors and and coefficients determined, let's now pick a specific situation to perform our counterfactual on. With our newly learned SEM let's predict the default value of X_4 when $X_2 = 0$, that is when the sprinkler is off. We'll assume that this not a sunny day:

In [64]:

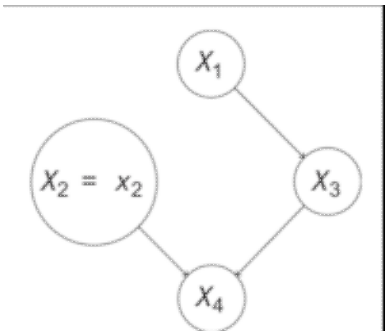
```
x1 = 0; #Not Sunny day
x2 = model_x2.predict(np.array([[x1]]))[0] #
print(f"Sprinkler status: {x2}")
x3 =  model_x3.predict(np.array([[x1]]))[0] #
x4 = model_x4.predict(np.array([[x2, x3]]))[0] #
print(f"Wet sidewalk status: {x4}")
```

```
Sprinkler status: 1
Slippery sidewalk status: 1
```

In this particular instance the side walk is wet when it's not a sunny day and when the sprinkler is on. This can be accounted for by one of the exogeneous variables. For example maybe in this particular instance someone spilled some water on the sidewalk. We have now completed step one of computing the counterfactual which is to use the evidence to determine the values of U. Now that the situation is set, we want to know what would have happened if we had intervened and turned off the sprinklers on that day, we express this as: $$ \begin{equation} X_{4_{x_2}}(U) = x_4 \end{equation} $$

Which reads, X_4 would be x_4 had X_2, been x_2.

We now intervene to create the following modified SCM:



In [65]:

```
x1 = 0; #Not sunny day
#x2 = model_x2.predict(np.array([[x1]]))[0] #
x2 = 0 #instead of predicting x2 using the learned model from the dataset, we intervene an
print(f"Sprinkler status: {x2}")
x3 =  model_x3.predict(np.array([[x1]]))[0] #
x4 = model_x4.predict(np.array([[x2, x3]]))[0] #
print(f"Wet sidewalk status: {x4}")

Sprinkler status: 0
Wet sidewalk status: 1
```

In the code above instead of predicting x2 using the learned model from the dataset, we intervene and set it's value to zero. We see that on this day, had we intervened and turned off the sprinkler the sidewalk would still be wet. This saves us an unecessary trip outside :).