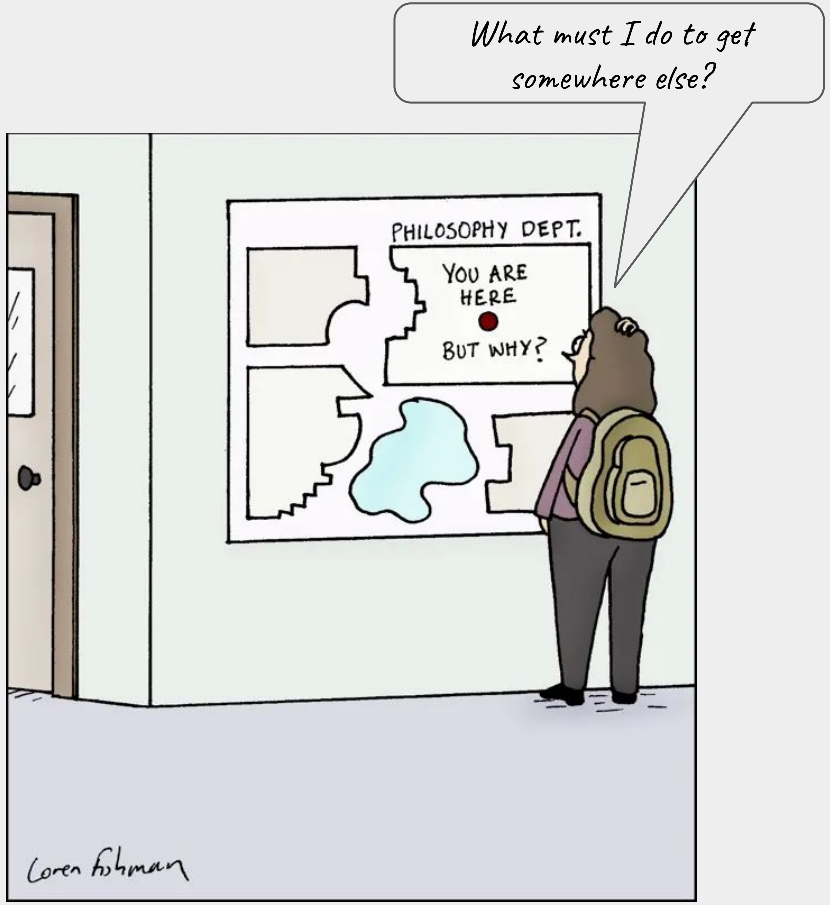
Explainable AI - Lecture 2

Counterfactuals

norwegian
open ai lab

NTNU

Counterfactual explanations



Explanation problem

We have a black-box model making an automated decision affecting end-users, for example:

- Loan approvals for bank customers: based on income, history, wealth, age, ... determine whether the customer will get a loan. Classification.

- Insurance price for car owners: based on accident history, car value, age, ... determine the insurance price for a given customer. Regression.

Explanation problem

We have a black-box model making an automated decision affecting end-users, for example:

- Loan approvals for bank customers: based on income, history, wealth, age, … determine whether the customer will get a loan. Classification.

- Insurance price for car owners: based on accident history, car value, age, … determine the insurance price for a given customer. Regression.

Today's scenario type:

1) An end-user asks for an outcome.
2) The model provides a prediction which is used as a decision.
3) The end-user wants to know **what it would take to change the outcome**

Explanation problem

A counterfactual explanation could state

> "If Tom had a higher income, then the loan would be approved"

or

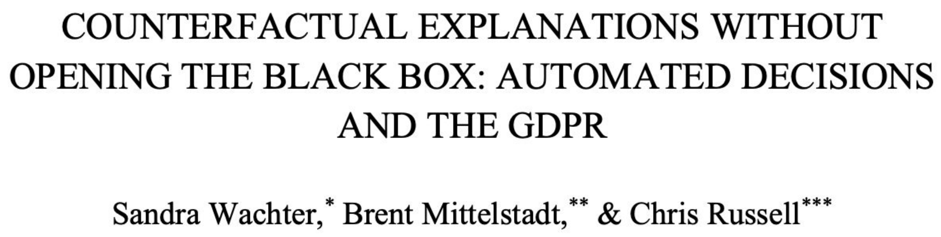
> "If Sara had less car accidents per year, her car insurance would be 1000 rand cheaper"

The question is "*what will it take to change the prediction from the* ***actual*** *value to the* ***desired*** *value?*"

Legal requirement for explanations

The EU General Data Protection Regulation (GDPR), states that individuals have the **right to receive explanations for and contest** automated decisions.

Largely in response to this, the following paper was written (one of today's additional resources):


COUNTERFACTUAL EXPLANATIONS WITHOUT OPENING THE BLACK BOX: AUTOMATED DECISIONS AND THE GDPR

Sandra Wachter,[*] Brent Mittelstadt,[**] & Chris Russell[***]

This is not a very technical paper, focusing heavily on legal requirements.

Legal requirement for explanations

The EU General Data Protection Regulation (GDPR), states that individuals have the **right to receive explanations for and contest** automated decisions.

Wachter et al (2017): COUNTERFACTUAL EXPLANATIONS WITHOUT OPENING THE BLACK BOX: AUTOMATED DECISIONS AND THE GDPR

This is not a very technical paper, focusing heavily on legal requirements.

Wachter et al use the following as an example of a counterfactual explanation:

"You were denied a loan because your annual income was £30,000.
If your income had been £45,000, you would have been offered a loan." ← *this is the counterfactual part*

Counterfactual explanations for legal requirements

Counterfactual explanations are useful to [1]:

1. Help the individual understand why a decision was reached;
2. Provide grounds to contest the decision if the outcome is undesired;
3. Understand what needs to change to receive a desired result in the future.

This **enhances the autonomy** of people subjected to automated decision and **help people recognize** when they should contest decisions [1].

Do you agree with the three points?

[1] The hidden assumptions behind counterfactual explanations and principal reasons, Barocas et al (2020)

Counterfactual explanations

In general:

A counterfactual explanation takes the form:

If input feature x_n had the value m_{cf} instead of m , the prediction $\hat{y}$ would change to $\hat{y}_{cf}$

Counterfactuals try to answer the question:

How could the input features change to get the desired prediction?

$\Rightarrow$If A, then desired outcome $\hat{y}_{cf}$

Counterfactual explanations

Where do we place this in the taxonomy?

Post-hoc?

Model-agnostic or model-specific?

Local or global?

Post-hoc methods	Model-agnostic	Model-specific
Local		
Global		

Counterfactual explanations

Where do we place this in the taxonomy?

Post-hoc: *We got a model and want to explain its behaviour.*

Model-agnostic or model-specific?

Local or global?

Post-hoc methods	Model-agnostic	Model-specific
Local		
Global		

Counterfactual explanations

Where do we place this in the taxonomy?

Post-hoc: *We got a model and want to explain its behaviour.*

Model-agnostic: *We care only about how to change the input to get the desired output.*

Local or global?

Post-hoc methods	Model-agnostic	Model-specific
Local		
Global		

Counterfactual explanations

Where do we place this in the taxonomy?

Post-hoc: *We got a model and want to explain its behaviour.*

Model-agnostic: *We care only about how to change the input to get the desired output.*

Local: *We care about one prediction, i.e. one data instance.*

Post-hoc methods	Model-agnostic	Model-specific
Local		
Global		

Counterfactual explanations

Where do we place this in the taxonomy?

Post-hoc: *We got a model and want to explain its behaviour.*

Model-agnostic: *We care only about how to change the input to get the desired output.*

Local: *We care about one prediction, i.e. one data instance.*

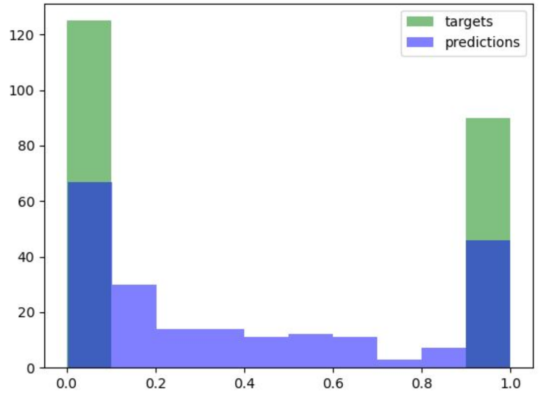
Post-hoc methods	Model-agnostic	Model-specific
Local	*Counterfactual explanations*	
Global		

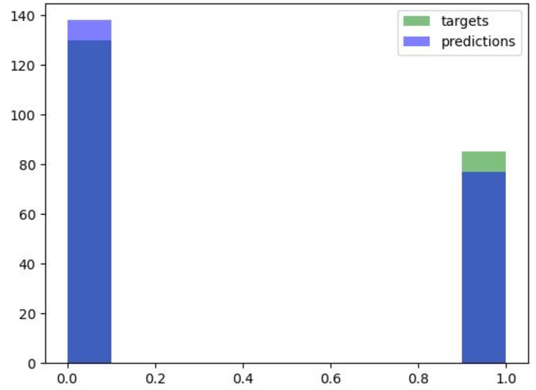
Let’s get some data and a model…

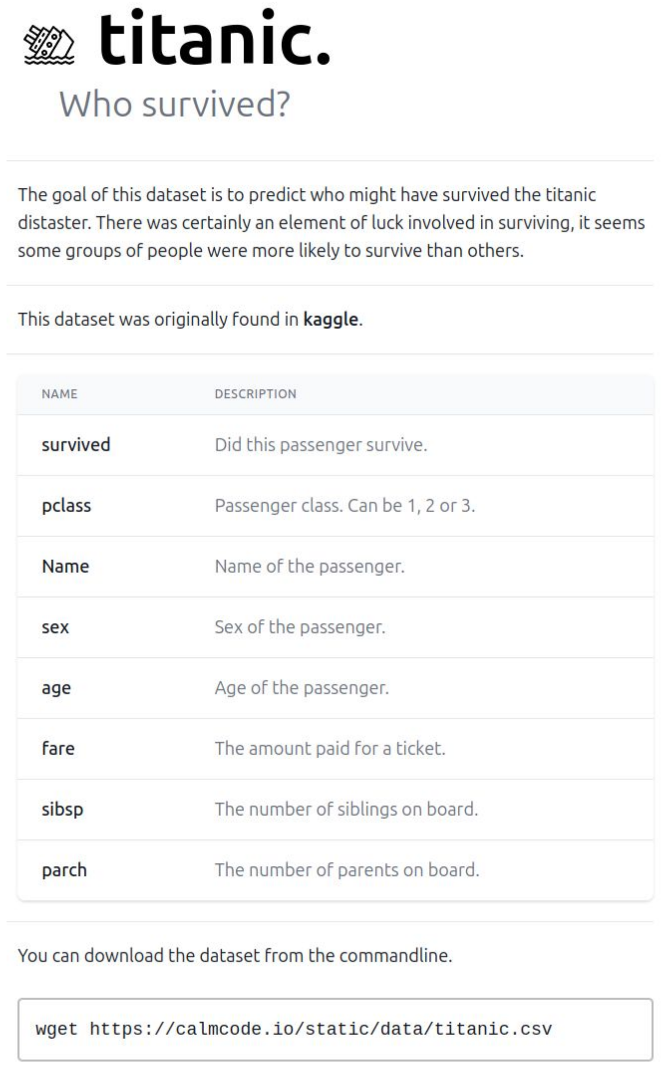
We’ll use the Titanic dataset as example

Features: Passenger class, Sex and Age

Model: XGBoost Classifier







titanic.

Who survived?

The goal of this dataset is to predict who might have survived the titanic distaster. There was certainly an element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

This dataset was originally found in **kaggle**.

NAME	DESCRIPTION
survived	Did this passenger survive.
pclass	Passenger class. Can be 1, 2 or 3.
Name	Name of the passenger.
sex	Sex of the passenger.
age	Age of the passenger.
fare	The amount paid for a ticket.
sibsp	The number of siblings on board.
parch	The number of parents on board.

You can download the dataset from the commandline.

```
wget https://calmcode.io/static/data/titanic.csv
```

```python
import pandas as pd
import numpy as np
import random

import matplotlib.pyplot as plt
import xgboost

from sklearn import metrics
from sklearn.model_selection import train_test_split
```

Data

First, get the .csv file containing the titanic data from https://calmcode.io/static/data/titanic.csv

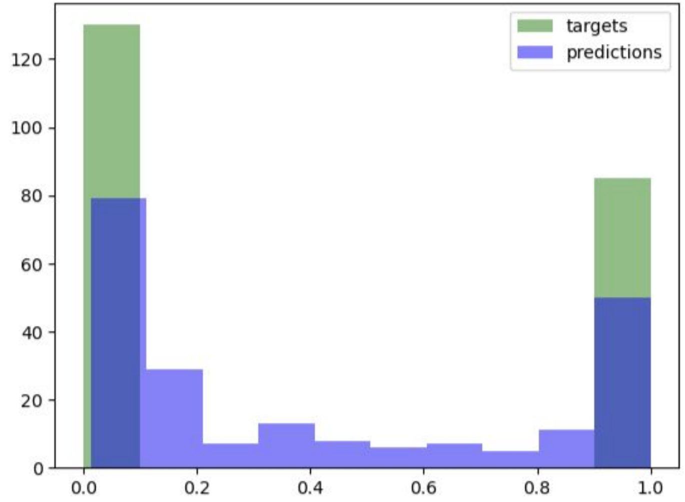
```python
df = pd.read_csv('../titanic_data.csv')
df = df[["pclass","sex","age","survived"]]
df.dropna(inplace=True)

features = df.columns

# Column sex is type string. Encode it to numerical
mapping = {"female": 0, "male": 1}
df["sex"] = df["sex"].map(mapping)

# Train test split
X_data = df[['pclass', 'sex', 'age']].values
y_data = df['survived'].values
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size=0.3)
```

Model

```python
def accuracy(true_values, predictions):
    return np.mean(true_values == predictions)
```

```python
model = xgboost.XGBClassifier()
model.fit(X_train, y_train)

# Predict and evaluate model
predictions = model.predict(X_test)
print("Accuracy: ", accuracy(y_test, predictions))
print("Precision: ", metrics.precision_score(y_test, predictions))
print("Sensitivity / recall: ", metrics.recall_score(y_test, predictions))

probs  = model.predict_proba(X_test)[:,1]

plt.hist(y_test, color="green", alpha=0.5, label="targets")
plt.hist(probs, color="blue", alpha=0.5, label="predictions")
plt.legend()
plt.show()
```

```
Accuracy:  0.827906976744186
Precision:  0.7926829268292683
Sensitivity / recall:  0.7647058823529411
```



Counterfactuals on the titanic

Features: `Passenger class, sex` and `age`.

Prediction: Chance of survival

    {0,1} from .predict or [0,1] from predict_proba

Let's make a passenger (data instance)

Paul: travelled in second class, is male and 25 years old.

Counterfactuals on the titanic

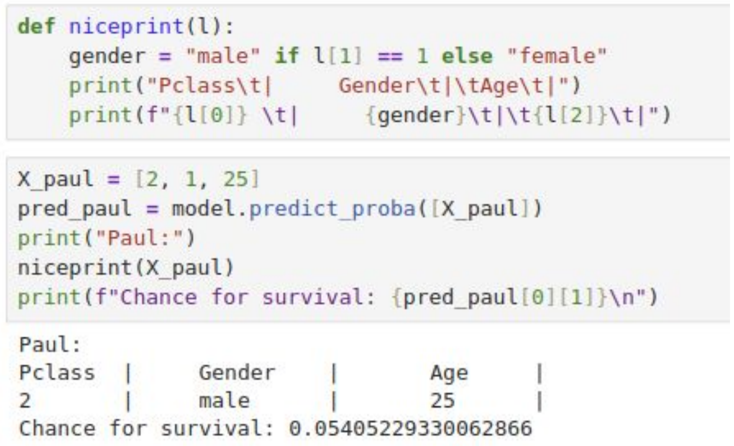
Features: Passenger class, sex and age.

Prediction: Change of survival

`{0,1} from .predict or [0,1] from predict_proba`

Let's make a passenger (data instance)

Paul: travelled in second class, is male and 25 years old.



```
def niceprint(l):
    gender = "male" if l[1] == 1 else "female"
    print("Pclass\t|     Gender\t|\tAge\t|")
    print(f"{l[0]} \t|     {gender}\t|\t{l[2]}\t|")
```

```
X_paul = [2, 1, 25]
pred_paul = model.predict_proba([X_paul])
print("Paul:")
niceprint(X_paul)
print(f"Chance for survival: {pred_paul[0][1]}\n")
```

```
Paul:
Pclass  |     Gender    |       Age     |
2       |     male      |       25      |
Chance for survival: 0.05405229330062866
```

Counterfactuals on the titanic

Features: Passenger class, sex and age.

Prediction: Change of survival

```
    {0,1} from .predict or [0,1] from predict_proba
```

Let's make a passenger (data instance)

Paul: travelled in second class, is male and 25 years old.
Model predicts ~0.05..., meaning a tiny chance of survival.

Counterfactual question: How do Paul's input features have to change in order for him to survive?

Counterfactuals on the titanic

Paul: travelled in second class, is male and 25 years old. [2, 1, 25]
$\hat{y}$ = 0.05...

Given a model *f* and our individual Paul *x*,
We want to know what it takes for Paul to survive. What do we have to do to get the counterfactual Paul?

1.

Counterfactuals on the titanic

Paul: travelled in second class, is male and 25 years old. [2, 1, 25]
$\hat{y} = 0.05...$

Given a model *f* and our individual Paul *x,*
We want to know what it takes for Paul to survive. What do we have to do to get the counterfactual Paul?

1. Pick a desired, different predicted value $\hat{y}_{cf}$

Counterfactuals on the titanic

Paul: travelled in second class, is male and 25 years old. [2, 1, 25]
ŷ = 0.05...

Given a model *f* and our individual Paul *x,*
We want to know what it takes for Paul to survive. What do we have to do to get the counterfactual Paul?

1. Pick a desired, different predicted value $\hat{y}_{cf}$
2. Try every combination of feature values in the training data, keeping the ones that yield $\hat{y}_{cf}$

Counterfactuals on the titanic

Paul: travelled in second class, is male and 25 years old. [2, 1, 25]
$\hat{y} = 0.05...$

Given a model f and individual x:

1. Pick a desired, different predicted value $\hat{y}_{cf}$
2. Try every combination of feature values in the training data, keeping the ones that yield $\hat{y}_{cf}$

We can try the following counterfactual Pauls:

$x_{cf} = [1,1,25]$ First class, male, 25 years old

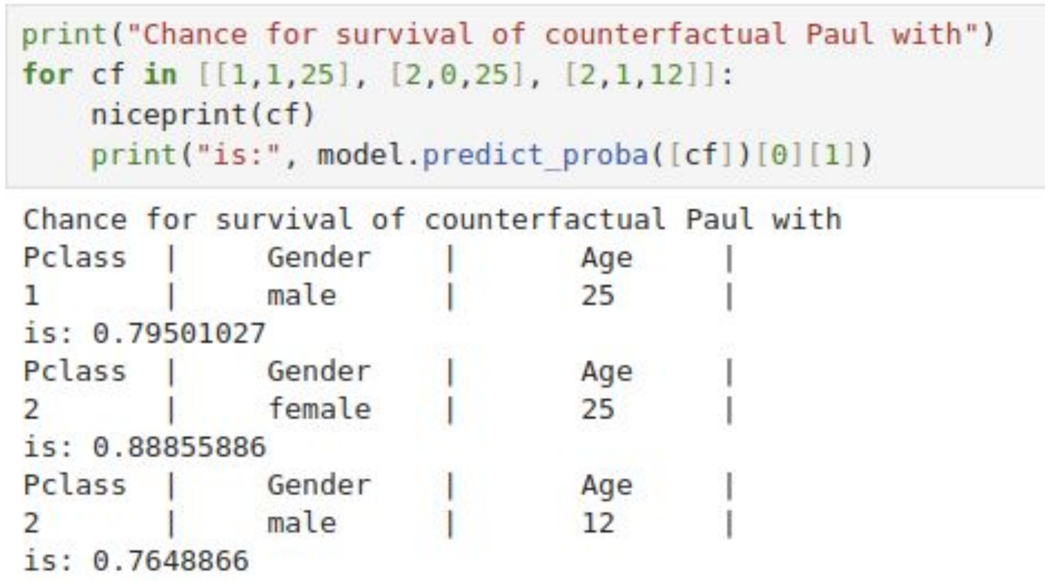
$x_{cf} = [2,0,25]$ Second class, female, 25 years old

$x_{cf} = [2,1,12]$ Second class, male, 12 years old

Counterfactuals on the titanic

Paul: travelled in second class, is male and 25 years old. [2, 1, 25]
$\hat{y} = 0.05...$

The predicted survival probabilities for our counterfactual Pauls:



```python
print("Chance for survival of counterfactual Paul with")
for cf in [[1,1,25], [2,0,25], [2,1,12]]:
    niceprint(cf)
    print("is:", model.predict_proba([cf])[0][1])
```

```
Chance for survival of counterfactual Paul with
Pclass  |       Gender      |         Age       |
1       |       male        |         25        |
is: 0.79501027
Pclass  |       Gender      |         Age       |
2       |       female      |         25        |
is: 0.88855886
Pclass  |       Gender      |         Age       |
2       |       male        |         12        |
is: 0.7648866
```

Counterfactuals on the titanic

Paul: travelled in second class, is male and 25 years old. [2, 1, 25]
$\hat{y} = 0.05...$

Counterfactual Pauls:

$x_{cf} = [1,1,25]$ First class, male, 25 years old $\Rightarrow \hat{y} = 0.80$

$x_{cf} = [2,0,25]$ Second class, female, 25 years old $\Rightarrow \hat{y} = 0.89$

$x_{cf} = [2,1,12]$ Second class, male, 12 years old $\Rightarrow \hat{y} = 0.76$

All of these Pauls survive. Nice. Or?

norwegian open ai lab NTNU

Counterfactuals on the titanic

Which of these do you think is best?

Why?

How would you define a good counterfactual?

	Actual instance	CF1	CF2	CF3
Pclass	2	1	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	**0.80**	**0.89**	**0.76**

Counterfactuals on the titanic

How do we define a good counterfactual?

The feature changes should be *actionable*

Which of these changes are actionable?

	Actual instance	CF1	CF2	CF3
Pclass	2	1	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	**0.80**	0.89	0.76

Counterfactuals on the titanic

How do we define a good counterfactual?

The feature changes should be *actionable*

The passenger class is actionable.

	Actual instance	CF1	CF2	CF3
Pclass	2	**1**	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	**0.80**	0.89	0.76

Counterfactuals on the titanic

How do we define a good counterfactual?

The feature changes should be *close*

Which of these changes are close?

	Actual instance	CF1	CF2	CF3
Pclass	2	1	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	**0.80**	**0.89**	**0.76**

Counterfactuals on the titanic

How do we define a good counterfactual?

The feature changes should be *close*

Changing by one unit can be close, if the feature is continuous. If it is discrete, it depends on the feature!

	Actual instance	CF1	CF2	CF3
Pclass	2	**1**	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	**0.80**	**0.89**	**0.76**

Counterfactuals on the titanic

How do we define a good counterfactual?

The prediction should be as *desired, meaning as close to $\hat{y}_{cf}$ as possible*

Which of these changes are desired?

	Actual instance	CF1	CF2	CF3
Pclass	2	1	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	0.80	0.89	0.76

norwegian open ai lab NTNU

Counterfactuals on the titanic

How do we define a good counterfactual?

The prediction should be as *desired, meaning as close to $\hat{y}_{cf}$ as possible*

We want Paul to survive, so an outcome well above 0.5. All of these outcomes are what we want.

	Actual instance	CF1	CF2	CF3
Pclass	2	1	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	**0.80**	**0.89**	**0.76**

Counterfactuals on the titanic

How do we define a good counterfactual?

The prediction should be *sparse, meaning change as few features as possible*

Which of these changes are sparse?

	Actual instance	CF1	CF2	CF3
Pclass	2	1	2	2
Sex	1	1	0	1
Age	25	25	25	12
Outcome	0.05	**0.80**	**0.89**	**0.76**

norwegian open ai lab NTNU

Counterfactuals on the titanic

How do we define a good counterfactual?

The prediction should be *sparse, meaning change as few features as possible*

All of the changes are sparse, changing only one feature.

	Actual instance	CF1	CF2	CF3
Pclass	2	**1**	2	2
Sex	1	1	**0**	1
Age	25	25	25	**12**
Outcome	0.05	**0.80**	**0.89**	**0.76**

Criteria for good counterfactuals

The feature changes should be *actionable*

What features should not be part of the explanation? Which can be?

The feature changes should be *close*

The counterfactual prediction should be as *desired*

The feature changes should be *sparse*

Criteria for good counterfactuals

The feature changes should be *actionable*

Features like sex or age should not be part of the explanation

The feature changes should be *close*

What does this mean for the counterfactual feature values?

The counterfactual prediction should be as *desired*

The feature changes should be *sparse*

One measure of closeness is the euclidean distance $d(x, x_{cf}) = \sum \sqrt{(x^2 - x_{cf}^{\,2})}$

Criteria for good counterfactuals

The feature changes should be *actionable*

Features like sex or age should not be part of the explanation

The feature changes should be *close*

The new feature values should be as close as possible to the old values

The counterfactual prediction should be as *desired*

What does this require of the counterfactual prediction?

The feature changes should be *sparse*

One measure of closeness is the euclidean distance $d(x, x_{cf}) = \sum \sqrt{(x^2 - x_{cf}^2)}$

Criteria for good counterfactuals

The feature changes should be *actionable*

Features like sex or age should not be part of the explanation

The feature changes should be *close*

The new feature values should be as close as possible to the old values

The counterfactual prediction should be as *desired*

The new prediction should be as close as possible to the desired outcome

The feature changes should be *sparse*

What does this require of the counterfactual feature vector?

One measure of closeness is the euclidean distance $d(x, x_{cf}) = \sum \sqrt{(x^2 - x_{cf}^2)}$

Criteria for good counterfactuals

The feature changes should be *actionable*

Features like sex or age should not be part of the explanation

The feature changes should be *close*

The new feature values should be as close as possible to the old values

The counterfactual prediction should be as *desired*

The new prediction should be as close as possible to the desired outcome

The feature changes should be *sparse*

Suggesting to change fewer features is better than suggesting to change many

One measure of closeness is the euclidean distance $d(x, x_{cf}) = \sum \sqrt{(x^2 - x_{cf}^{\,2})}$

Finding counterfactuals, Wachter et al (2017)

Changing features by hand is obviously not an ideal approach.

Instead, we can formulate an *optimisation problem*, as suggested by Wachter et al in the 2017 paper.

Finding counterfactuals, Wachter et al (2017)

Changing features by hand is obviously not an ideal approach.

Instead, we can formulate an *optimisation problem, as suggested by Wachter et al in the 2017 paper.*

For a model f, datapoint x, prediction $\hat{y}$, and desired outcome $\hat{y}_{cf}$, find the counterfactual datapoint x_{cf} that minimises the loss

$$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$$

What do the terms in this loss function represent?

Finding counterfactuals, Wachter et al (2017)

Changing features by hand is obviously not an ideal approach.

Instead, we can formulate an *optimisation problem*

For a model f, datapoint x, prediction $\hat{y}$, and desired outcome $\hat{y}_{cf}$, find the counterfactual datapoint x_{cf} that minimises the loss

$$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$$

First term: Response proximity (how close the outcome is to the desired outcome)

Second term: Feature proximity (how close the counterfactual feature values are to the original)

λ: tradeoff between response and feature proximity

Finding counterfactuals, Wachter et al (2017)

You should remember this loss function :)

$$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$$

What is not ideal about this loss function?

What are four the requirements we discussed for a good counterfactual?

Finding counterfactuals, Wachter et al (2017)

You should remember this expression :)

$$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$$

How does this expression incorporate the requirements for a good counterfactual?

Desirable:

Close:

Actionable:

Sparse:

Finding counterfactuals, Wachter et al (2017)

You should remember this expression :)

$$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$$

How does this expression incorporate the requirements for a good counterfactual?

Desirable: the first term makes sure the new prediction is close to the desired prediction

Close:

Actionable:

Sparse:

Finding counterfactuals, Wachter et al (2017)

You should remember this expression :)

$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$

How does this expression incorporate the requirements for a good counterfactual?

Desirable: the first term makes sure the new prediction is close to the desired prediction

Close: the second term makes sure the new feature values are close to the original feature values

Actionable:

Sparse:

Finding counterfactuals, Wachter et al (2017)

You should remember this expression :)

$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$

How does this expression incorporate the requirements for a good counterfactual?

Desirable: the first term makes sure the new prediction is close to the desired prediction

Close: the second term makes sure the new feature values are close to the original feature values

Actionable: ... *there is no information in this loss function about whether feature changes are possible, and unrealistic feature combinations are not penalised.*

Sparse:

Finding counterfactuals, Wachter et al (2017)

You should remember this expression :)

$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$

How does this expression incorporate the requirements for a good counterfactual?

Desirable: the first term makes sure the new prediction is close to the desired prediction

Close: the second term makes sure the new feature values are close to the original feature values

Actionable: ... *there is no information in this loss function about whether feature changes are possible, and unrealistic feature combinations are not penalised.*

Sparse: ... *changing 5 features by 1 can yield the same distance as changing one feature by 5*

Finding counterfactuals, Dandl et al (2020)

Counterfactual explanations were extended in the 2020 paper *Multi-Objective Counterfactual Explanations* by Dandl et al (additional resource 2/2):

Multi-Objective Counterfactual Explanations

Susanne Dandl(✉), Christoph Molnar, Martin Binder, and Bernd Bischl

Department of Statistics, LMU Munich, Ludwigstr. 33, 80539 Munich, Germany
susanne.dandl@stat.uni-muenchen.de

Here, the loss function is changed to a four-objective function

$$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = o_1(f(x_{cf}), \hat{y}_{cf}), o_2(x, x_{cf}), o_3(x, x_{cf}), o_4(x, \mathbf{X}^{\text{obs}})$$

Finding counterfactuals, Dandl et al (2020)

Multi-Objective Counterfactual Explanations, Dandl et al (2020):

Here, the loss function is changed to a four-objective function

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = \text{o}_1(f(x_{cf}), \hat{y}_{cf}), \text{o}_2(x, x_{cf}), \text{o}_3(x, x_{cf}), \text{o}_4(x, \mathbf{X}^{\text{obs}})$

What do the terms o_1-o_4 represent?

Finding counterfactuals, Dandl et al (2020)

Multi-Objective Counterfactual Explanations, Dandl et al (2020):

Here, the loss function is changed to a four-objective function

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = \text{o}_1(f(x_{cf}), \hat{y}_{cf}), \text{o}_2(x, x_{cf}), \text{o}_3(x, x_{cf}), \text{o}_4(x, \mathbf{X}^{\text{obs}})$

o_1 takes care of **desirable**: the new prediction $f(x_{cf})$ should be close to the desired outcome $\hat{y}_{cf}$

norwegian open ai lab NTNU

Finding counterfactuals, Dandl et al (2020)

Multi-Objective Counterfactual Explanations, Dandl et al (2020):

Here, the loss function is changed to a four-objective function

$$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = \text{o}_1(f(x_{cf}), \hat{y}_{cf}), \text{o}_2(x, x_{cf}), \text{o}_3(x, x_{cf}), \text{o}_4(x, \mathbf{X}^{\text{obs}})$$

o_1 takes care of **desirable**: the new prediction $f(x_{cf})$ should be close to the desired outcome $\hat{y}_{cf}$

o_2 takes care of **closeness**: the counterfactual x_{cf} should be as close as possible to the original x

Finding counterfactuals, Dandl et al (2020)

Multi-Objective Counterfactual Explanations, Dandl et al (2020):

Here, the loss function is changed to a four-objective function

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = \text{o}_1(f(x_{cf}), \hat{y}_{cf}), \text{o}_2(x, x_{cf}), \text{o}_3(x, x_{cf}), \text{o}_4(x, \mathbf{X}^{\text{obs}})$

o_1 takes care of **desirable**: the new prediction $f(x_{cf})$ should be close to the desired outcome $\hat{y}_{cf}$

o_2 takes care of **closeness**: the counterfactual x_{cf} should be as close as possible to the original x

o_3 takes care of **sparseness**: the counterfactual x_{cf} should involve few feature changes in x

Finding counterfactuals, Dandl et al (2020)

Multi-Objective Counterfactual Explanations, Dandl et al (2020):

Here, the loss function is changed to a four-objective function

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = \text{o}_1(f(x_{cf}), \hat{y}_{cf}), \text{o}_2(x, x_{cf}), \text{o}_3(x, x_{cf}), \text{o}_4(x, \mathbf{X}^{\text{obs}})$

o_1 takes care of **desirable**: the new prediction $f(x_{cf})$ should be close to the desired outcome $\hat{y}_{cf}$

o_2 takes care of **closeness**: the counterfactual x_{cf} should be as close as possible to the original x

o_3 takes care of **sparseness**: the counterfactual x_{cf} should involve few feature changes in x

o_4 takes care of **actionability**: the counterfactual x_{cf} should be **likely** compared to actual data points $\mathbf{X}^{\text{obs}}$

Finding counterfactuals, Dandl et al (2020)

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = o_1(f(x_{cf}), \hat{y}_{cf}), o_2(x, x_{cf}), o_3(x, x_{cf}), o_4(x, \mathbf{X}^{\text{obs}})$

desirable: the new prediction $f(x_{cf})$ should be close to the desired outcome $\hat{y}_{cf}$.

One possibility is the L_1 norm:

$$o_1(\hat{f}(\mathbf{x}'), y') = \begin{cases} 0 & \text{if } \hat{f}(\mathbf{x}') \in y' \\ \inf_{y' \in y'} |\hat{f}(\mathbf{x}') - y'| & \text{else} \end{cases}$$

This is just the distance between the new prediction and the desired outcome.

If the two are the same, $o_1 = 0$
If they are far apart, o_1 is large

Finding counterfactuals, Dandl et al (2020)

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = o_1(f(x_{cf}), \hat{y}_{cf}), o_2(x, x_{cf}), o_3(x, x_{cf}), o_4(x, \mathbf{X}^{\text{obs}})$

closeness: the counterfactual x_{cf} should be as close as possible to the original x.

Euclidean distance is nice, but doesn't handle categorical features. The Gower distance does this:

$$o_2(\mathbf{x}, \mathbf{x}') = \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x'_j)$$

p denotes the feature indices, so the distance is normalised, and

$$\delta_G(x_j, x'_j) = \begin{cases} \frac{1}{\widehat{R}_j} |x_j - x'_j| & \text{if } x_j \text{ numerical} \\ \mathbb{I}_{x_j \neq x'_j} & \text{if } x_j \text{ categorical} \end{cases}$$

For numerical features, this is the vector norm, normalised by the range of feature j's values, R_j .

For categorical features, the distance is 0 for equal features, and 1 otherwise.

Finding counterfactuals, Dandl et al (2020)

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = o_1(f(x_{cf}), \hat{y}_{cf}), o_2(x, x_{cf}), o_3(x, x_{cf}), o_4(x, \mathbf{X}^{\text{obs}})$

sparseness: the counterfactual x_{cf} should involve few feature changes in x.

We can measure this using the L_0 norm

$$o_3(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_0 = \sum_{j=1}^{p} I_{x'_j \neq x_j}$$

That is, counting all instances for which a feature value is different in x_{cf} compared to x.

Finding counterfactuals, Dandl et al (2020)

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = o_1(f(x_{cf}), \hat{y}_{cf}), o_2(x, x_{cf}), o_3(x, x_{cf}), o_4(x, \mathbf{X}^{\text{obs}})$

actionability: this is represented as the counterfactual x_{cf} being **likely** compared to actual data points $\mathbf{X}^{\text{obs}}$

We can approximate the likelihood of a data point x_{cf} using observed data.

One possibility is using the Gower distance between x_{cf} and the nearest data point $x^{[1]} \in \mathbf{X}^{\text{obs}}$

$$o_4(\mathbf{x}', \mathbf{X}^{\text{obs}}) = \frac{1}{p}\sum_{j=1}^{p} \delta_G(x'_j, x_j^{[1]})$$

This involves comparing x_{cf} to every data point in $\mathbf{X}^{\text{obs}}$.

Finding counterfactuals, Dandl et al (2020)

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = o_1(f(x_{cf}), \hat{y}_{cf}), o_2(x, x_{cf}), o_3(x, x_{cf}), o_4(x, \mathbf{X}^{\text{obs}})$

Notice that this expression has no parameter like the λ from earlier:

$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$

What does that mean?

Finding counterfactuals, Dandl et al (2020)

$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = o_1(f(x_{cf}), \hat{y}_{cf}), o_2(x, x_{cf}), o_3(x, x_{cf}), o_4(x, \mathbf{X}^{\text{obs}})$

Notice that this expression has no parameter like the λ from earlier.

The four objectives are not combined into a single objective through a weighted sum, and there is no tradeoff between them. **All four terms are optimized simultaneously**.

This can be done using a variety of optimisation algorithms. We'll look at the one most commonly used.

The optimization problem

How to solve the 4-part optimization?

Remember, we want to find a solution that optimizes all four criteria:

o_1: the new prediction $f(x_{cf})$ should be **close** to the desired outcome $\hat{y}_{cf}$

o_2: the distance between the original datapoint x and the counterfactual x_{cf} should be **small**

o_3: the counterfactual x_{cf} should have **many** features unchanged compared to x_{cf}

o_4: the counterfactual x_{cf} should be as close as possible to at least one point in $\mathbf{X}^{\text{obs}}$

There is no weighting between these; they are optimised simultaneously.

How to solve the 4-part optimization?

The most widely used optimization algorithm for this is the Nondominated Sorting Genetic Algorithm (Deb et al. 2002), short **NSGA-II**.

In Python, this is available through the optuna package.

The goal of NSGA-II is to **find the Pareto front for the defined objectives** (o_1-o_4).

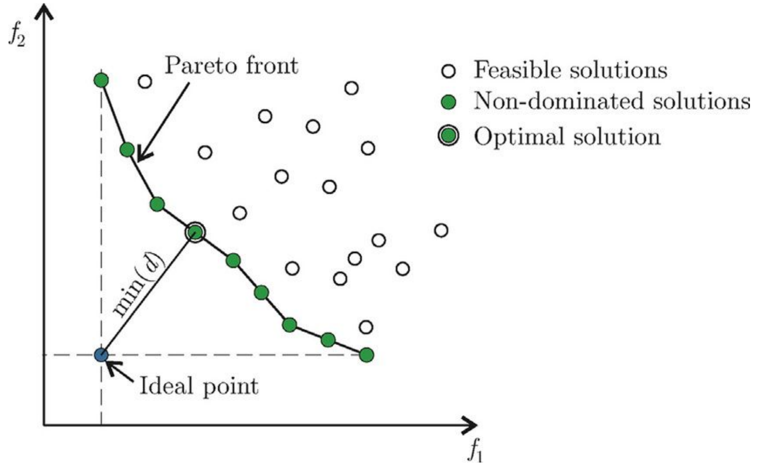
Pareto front

Intuitively:

If a solution is on the Pareto front, you can't make one objective better without making at least one other worse.

Any solution not on the Pareto front is strictly worse.
(that's the "nondominated" in NSGA)

Can you guess what the Pareto front is in our case?



Pareto front

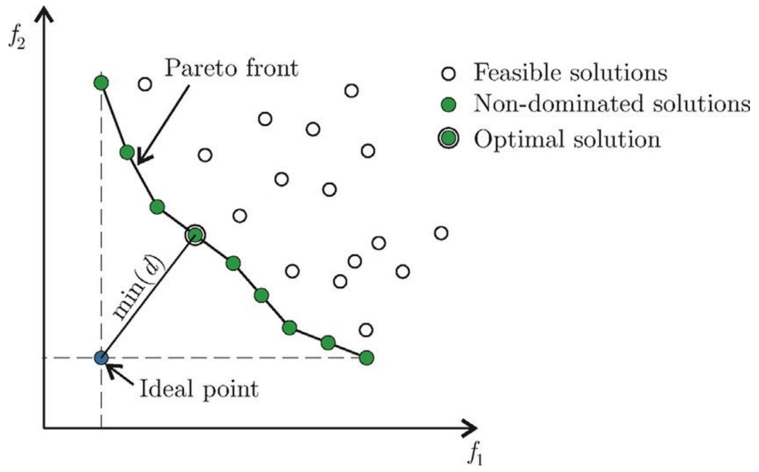
Intuitively:

If a solution is on the Pareto front, you can't make one objective better without making at least one other worse.

Any solution not on the Pareto front is strictly worse.

Can you guess what the Pareto front is in our case?

Hint: It is a list of points. What do these points represent?



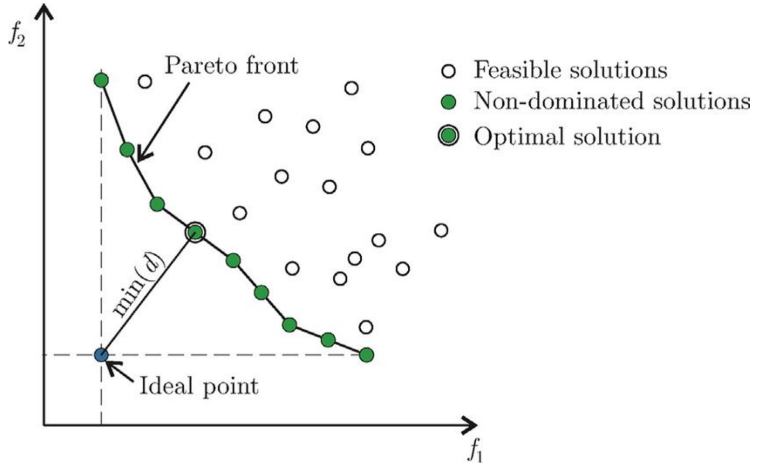
Pareto front

Intuitively:

If a solution is on the Pareto front, you can't make one objective better without making at least one other worse.

Any solution not on the Pareto front is strictly worse.

In our case, **the Pareto front is the list of all counterfactuals**.



Nondominated Sorting Genetic Algorithm

NSGA is a genetic algorithm, meaning that it generates a population of candidate solutions and evolves them toward better solutions.

In our case:

1. a population of candidates for x_{cf} is initialized by randomly changing some of x's features

Nondominated Sorting Genetic Algorithm

1. a population of candidates for x_{cf} is initialized by randomly changing some of x's features

2. each candidate is evaluated using our four objective functions

Nondominated Sorting Genetic Algorithm

1. a population of candidates for x_{cf} is initialized by randomly changing some of x's features
2. each candidate is evaluated using our four objective functions

3. some candidates are selected randomly, where fitter candidates are more likely to be selected

Nondominated Sorting Genetic Algorithm

1. a population of candidates for x_{cf} is initialized by randomly changing some of x's features
2. each candidate is evaluated using our four objective functions
3. some candidates are selected randomly, where fitter candidates are more likely to be selected

4. the candidates are pairwise recombined to produce children.
   Numerical feature values are averaged and categorical features are crossed over.
   Also, feature values are slightly perturbed to explore the whole feature space ← *mutation*

Nondominated Sorting Genetic Algorithm

1. a population of candidates for x_{cf} is initialized by randomly changing some x's features
2. each candidate is evaluated using our four objective functions
3. some candidates are selected randomly, where fitter candidates are more likely to be selected
4. the candidates are pairwise recombined to produce children. Numerical feature values are averaged and categorical features are crossed over. Feature values are slightly perturbed to explore the whole feature space (mutation)

5. the candidates (parents and children) are sorted using the nondominated sorting algorithm, and the most promising half is kept.

Nondominated Sorting Genetic Algorithm

1. a population of candidates for x_{cf} is initialized by randomly changing some x's features
2. each candidate is evaluated using our four objective functions
3. some candidates are selected randomly, where fitter candidates are more likely to be selected
4. the candidates are pairwise recombined to produce children. Numerical feature values are averaged and categorical features are crossed over. Feature values are slightly perturbed to explore the whole feature space (mutation)
5. the candidates (parents and children) are sorted using the nondominated sorting algorithm, and the most promising half is kept

6. repeat from step 2 until stopping criterion is met.

Nondominated Sorting Genetic Algorithm

1. a population of candidates for x_{cf} is initialized by randomly changing some x's features
2. each candidate is evaluated using our four objective functions
3. some candidates are selected randomly, where fitter candidates are more likely to be selected
4. the candidates are pairwise recombined to produce children. Numerical feature values are averaged and categorical features are crossed over. Feature values are slightly perturbed to explore the whole feature space (mutation)
5. the candidates (parents and children) are sorted using the nondominated sorting algorithm, and the most promising half is kept.
6. repeat selection, recombination, mutation etc from step 2 until stopping criterion is met.

The final population after stopping is our list of counterfactuals.

Homework :)

You get a model, a set of observations and a datapoint from me.

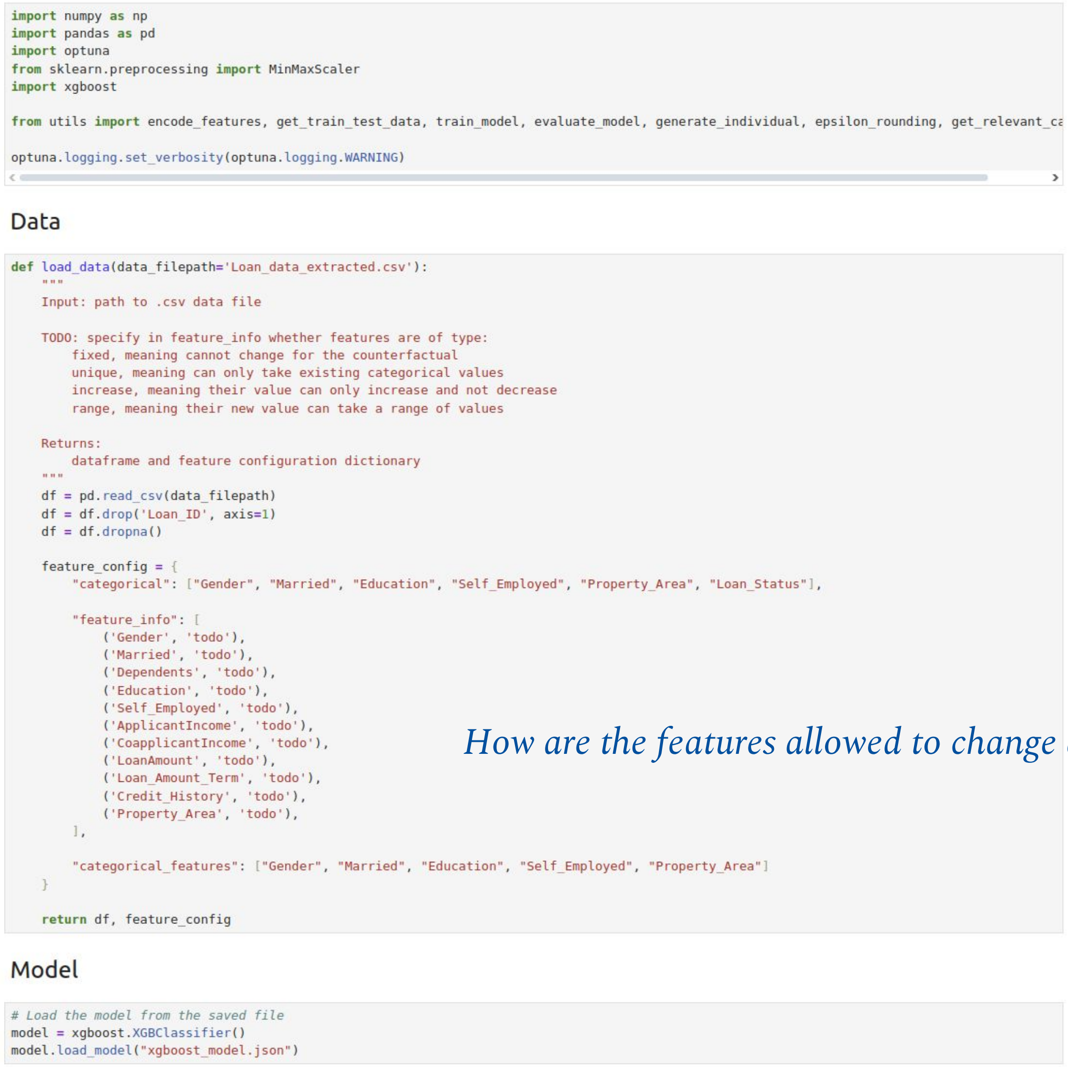
The task is is simple: **find a counterfactual that flips the model prediction**.

Do this by:

1) trial and error, probing the model as often as you want
2) implementing Wachter et al (2017) and solving the weighted 1-object optimisation function
3) **using the provided notebook, complete the required code and solve the 4-objective optimisation problem using optuna.** *This one is a bit challenging, so don't get frustrated if you struggle :)*

You can of course also:

- find libraries online that solve the problem for you
- bribe a fellow student to give you the solution
- do something I haven't thought of.



```python
import numpy as np
import pandas as pd
import optuna
from sklearn.preprocessing import MinMaxScaler
import xgboost

from utils import encode_features, get_train_test_data, train_model, evaluate_model, generate_individual, epsilon_rounding, get_relevant_ca

optuna.logging.set_verbosity(optuna.logging.WARNING)
```

Data

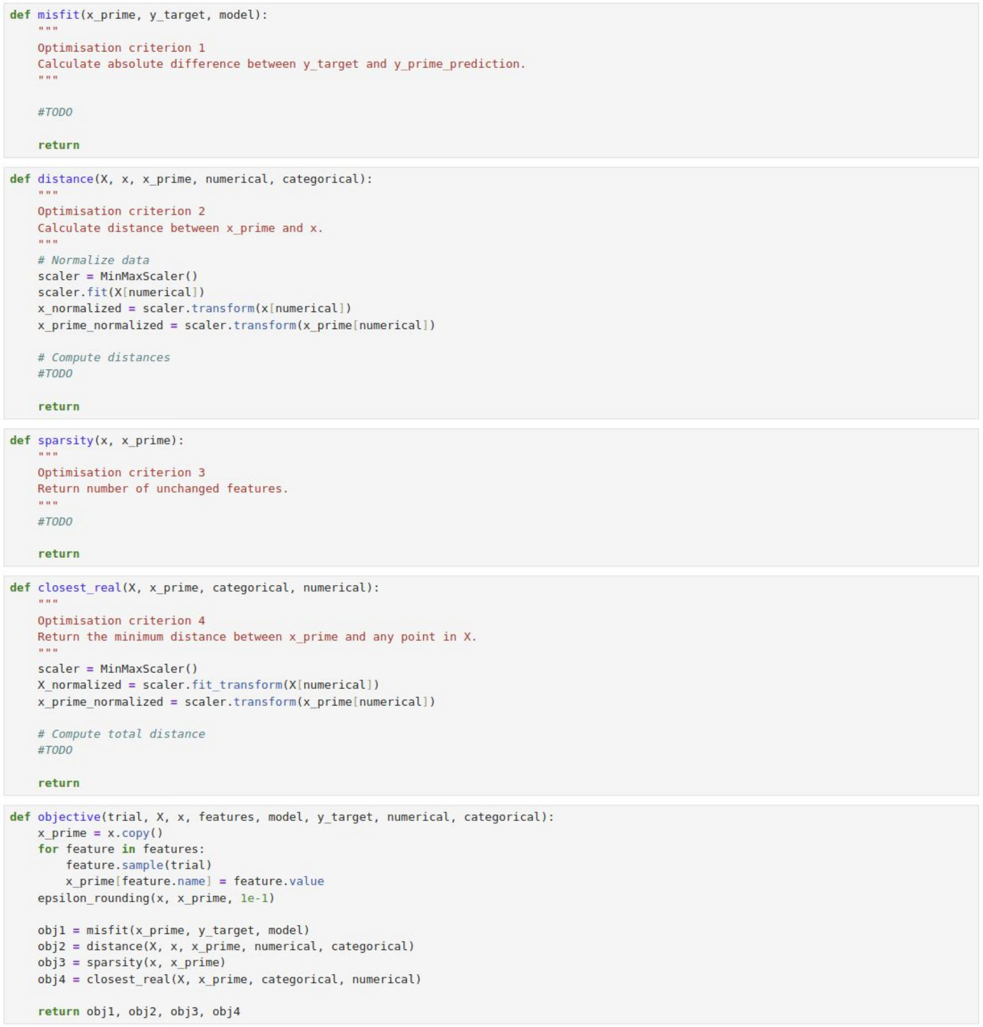
```python
def load_data(data_filepath='Loan_data_extracted.csv'):
    """
    Input: path to .csv data file

    TODO: specify in feature_info whether features are of type:
        fixed, meaning cannot change for the counterfactual
        unique, meaning can only take existing categorical values
        increase, meaning their value can only increase and not decrease
        range, meaning their new value can take a range of values

    Returns:
        dataframe and feature configuration dictionary
    """
    df = pd.read_csv(data_filepath)
    df = df.drop('Loan_ID', axis=1)
    df = df.dropna()

    feature_config = {
        "categorical": ["Gender", "Married", "Education", "Self_Employed", "Property_Area", "Loan_Status"],

        "feature_info": [
            ('Gender', 'todo'),
            ('Married', 'todo'),
            ('Dependents', 'todo'),
            ('Education', 'todo'),
            ('Self_Employed', 'todo'),
            ('ApplicantIncome', 'todo'),
            ('CoapplicantIncome', 'todo'),
            ('LoanAmount', 'todo'),
            ('Loan_Amount_Term', 'todo'),
            ('Credit_History', 'todo'),
            ('Property_Area', 'todo'),
        ],

        "categorical_features": ["Gender", "Married", "Education", "Self_Employed", "Property_Area"]
    }

    return df, feature_config
```

Model

```python
# Load the model from the saved file
model = xgboost.XGBClassifier()
model.load_model("xgboost_model.json")
```

How are the features allowed to change during the search?

TODO: Code for counterfactual search



```python
def misfit(x_prime, y_target, model):
    """
    Optimisation criterion 1
    Calculate absolute difference between y_target and y_prime_prediction.
    """

    #TODO

    return
```

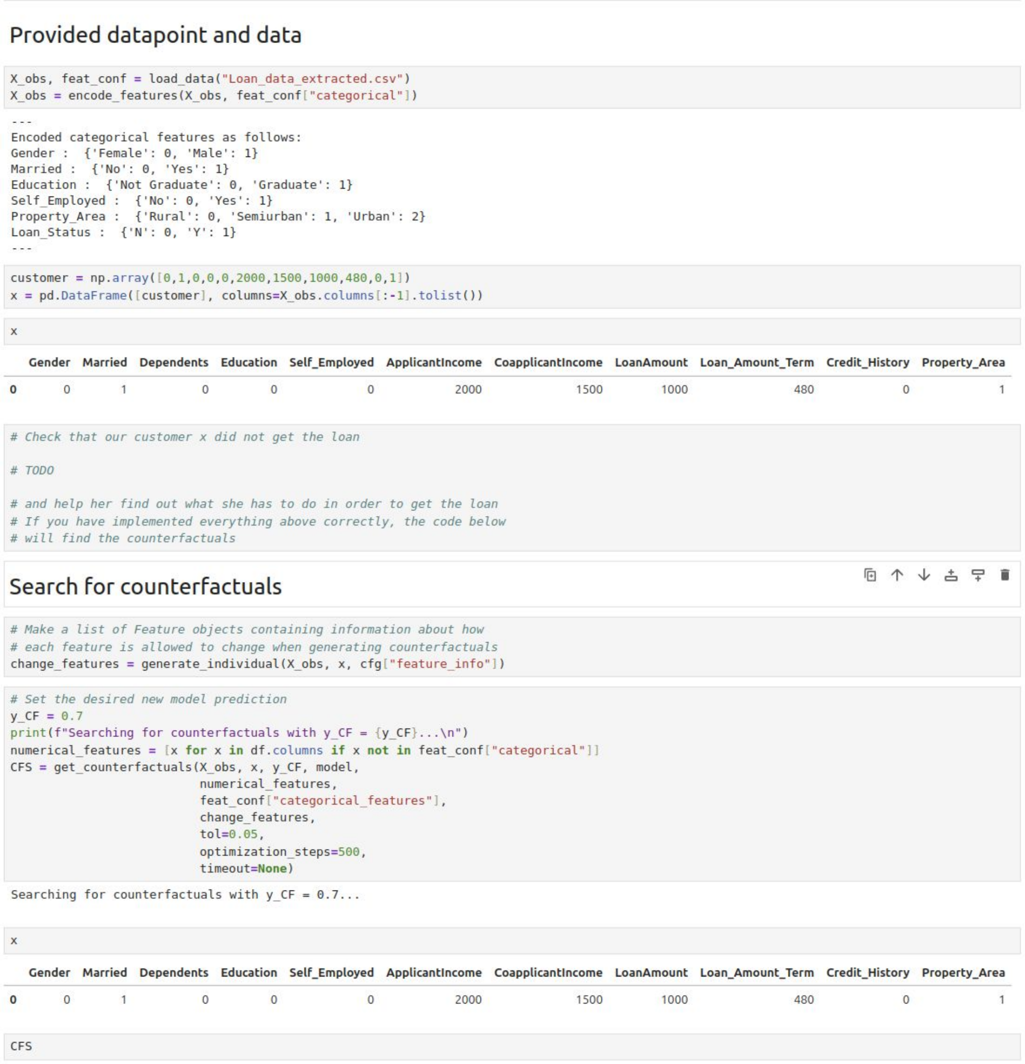
```python
def distance(X, x, x_prime, numerical, categorical):
    """
    Optimisation criterion 2
    Calculate distance between x_prime and x.
    """
    # Normalize data
    scaler = MinMaxScaler()
    scaler.fit(X[numerical])
    x_normalized = scaler.transform(x[numerical])
    x_prime_normalized = scaler.transform(x_prime[numerical])

    # Compute distances
    #TODO

    return
```

```python
def sparsity(x, x_prime):
    """
    Optimisation criterion 3
    Return number of unchanged features.
    """
    #TODO

    return
```

```python
def closest_real(X, x_prime, categorical, numerical):
    """
    Optimisation criterion 4
    Return the minimum distance between x_prime and any point in X.
    """
    scaler = MinMaxScaler()
    X_normalized = scaler.fit_transform(X[numerical])
    x_prime_normalized = scaler.transform(x_prime[numerical])

    # Compute total distance
    #TODO

    return
```

```python
def objective(trial, X, x, features, model, y_target, numerical, categorical):
    x_prime = x.copy()
    for feature in features:
        feature.sample(trial)
        x_prime[feature.name] = feature.value
    epsilon_rounding(x, x_prime, 1e-1)

    obj1 = misfit(x_prime, y_target, model)
    obj2 = distance(X, x, x_prime, numerical, categorical)
    obj3 = sparsity(x, x_prime)
    obj4 = closest_real(X, x_prime, categorical, numerical)

    return obj1, obj2, obj3, obj4
```

Write code for the four criteria we discussed



Provided datapoint and data

```
X_obs, feat_conf = load_data("Loan_data_extracted.csv")
X_obs = encode_features(X_obs, feat_conf["categorical"])
```

```
---
Encoded categorical features as follows:
Gender :  {'Female': 0, 'Male': 1}
Married :  {'No': 0, 'Yes': 1}
Education :  {'Not Graduate': 0, 'Graduate': 1}
Self_Employed :  {'No': 0, 'Yes': 1}
Property_Area :  {'Rural': 0, 'Semiurban': 1, 'Urban': 2}
Loan_Status :  {'N': 0, 'Y': 1}
---
```

```
customer = np.array([0,1,0,0,0,2000,1500,1000,480,0,1])
x = pd.DataFrame([customer], columns=X_obs.columns[:-1].tolist())
```

```
x
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	0	1	0	0	0	2000	1500	1000	480	0	1

```
# Check that our customer x did not get the loan

# TODO

# and help her find out what she has to do in order to get the loan
# If you have implemented everything above correctly, the code below
# will find the counterfactuals
```

Search for counterfactuals

```
# Make a list of Feature objects containing information about how
# each feature is allowed to change when generating counterfactuals
change_features = generate_individual(X_obs, x, cfg["feature_info"])
```

```
# Set the desired new model prediction
y_CF = 0.7
print(f"Searching for counterfactuals with y_CF = {y_CF}...\n")
numerical_features = [x for x in df.columns if x not in feat_conf["categorical"]]
CFS = get_counterfactuals(X_obs, x, y_CF, model,
                          numerical_features,
                          feat_conf["categorical_features"],
                          change_features,
                          tol=0.05,
                          optimization_steps=500,
                          timeout=None)
```

```
Searching for counterfactuals with y_CF = 0.7...
```

```
x
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	0	1	0	0	0	2000	1500	1000	480	0	1

```
CFS
```

For this data point, produce a list of counterfactuals.

Additional resources

Wachter et al (2017): COUNTERFACTUAL EXPLANATIONS WITHOUT OPENING THE BLACK BOX: AUTOMATED DECISIONS AND THE GDPR

Dandl et al (2020): Multi-Objective Counterfactual Explanations

Strengths and weaknesses

Strengths

What's good about counterfactuals as explanations?

Strengths

Counterfactuals…

- have a clear interpretation: people intuitively understand the counterfactual and its validity
- are fairly easy to implement: being an optimisation problem, we only need a loss function and an optimiser
- can be generated for any system that receives inputs and returns outputs - not limited to machine learning models, but can be used for any AI system
- are particularly suited for recourse, i.e. helping the end user change the decision
- are not an approximation; the new instance x_{cf} is guaranteed to produce y_{cf}.

Many XAI methods are approximations and open to interpretation. This is not the case for counterfactuals.

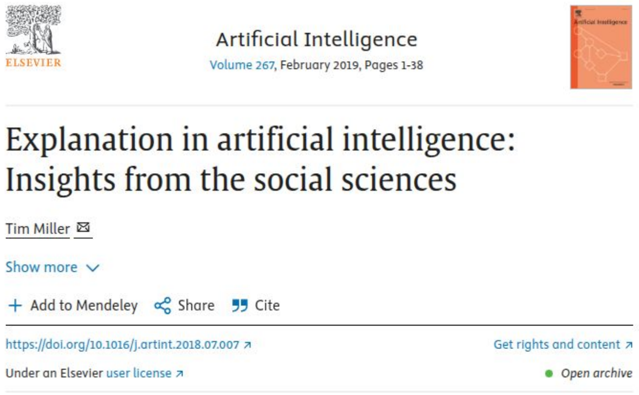
Strengths

We don't know if CFEs will be accepted by regulatory bodies.

One of the most influential papers in XAI, Miller (2019), states (and is widely cited for):

"Explanations are contrastive"

Explanations that answer **"Why event A rather than event B?"** are considered better in terms of understandability compared to explanations for "Why event A?"


ELSEVIER

Artificial Intelligence
Volume 267, February 2019, Pages 1-38

Explanation in artificial intelligence: Insights from the social sciences

Tim Miller

Show more

Add to Mendeley Share Cite

https://doi.org/10.1016/j.artint.2018.07.007

Get rights and content

Under an Elsevier user license

Open archive

Weaknesses

What's not so great about counterfactuals as explanations?

Weaknesses

Counterfactuals are not as useful for model and data insights: for an XAI researcher, a counterfactual holds very little information about the model.

Counterfactuals are not unique: For each x, there are many x_{cf}. This is the Rashomon effect for CFs.

When is the list of CFs complete? How many CFs should the end-user receive? Making a selection to provide to the user introduces a bias.

The causality perspective

A perspective from causality

Counterfactual explanations are known by a variety of terms: **contrastive, actionable recourse, algorithmic recourse, action guiding**, …

In the literature on causality, counterfactual has a different meaning. There, it's safer to refer to the CFEs we just learned about as **contrastive explanations**.

A perspective from causality

A counterfactual according to the causality literature is a statement on the form

> I have a **causal model**, input x and prediction y. What would happen to y if x changed to x' ?

The answer to this question is a valid counterfactual *although the prediction* y stays the same.

The counterfactual makes a statement about the consequence of a change.

Finally, the change from x to x' must follow a causal model.

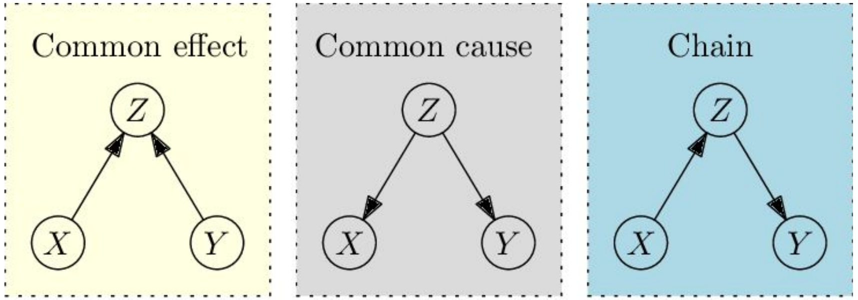
Causal models in two seconds...

Most counterfactual / contrastive XAI methods *would not be accepted* as causal explanations. In order to generate causal explanations, we would need either a causal model or at least interventional data.

A **structural causal model** (SCM) tells us about the causal relationships in the world that generated our data.

Interventional data are collected by intervening in the real world and observing the consequence.

These can be used to generate realistic data points x_{cf}.



A perspective from causality

A counterfactual according to the causality literature is a statement on the form

> I have a **causal model**, input x and prediction y. What would happen to y if x changed to x'?

The answer to this question is a valid counterfactual although the prediction y stays the same.

The counterfactual makes a statement about the consequence of a change.

A counterfactual x' is actionable / feasible only if it is reachable by action, starting from the original state

What we’re doing

We *could* have made a SCM for our data features and generated causal counterfactuals and contrastive explanations.

We will not be studying the causal perspective on counterfactuals further in this course. You only have to focus on the methods by Wachter et al (2017):

$$L(x, x_{cf}, \hat{y}_{cf}, \lambda) = \lambda(f(x_{cf}) - \hat{y}_{cf})^2 + d(x, x_{cf})$$

And Dandl et al (2020):

$$L(x, x_{cf}, \hat{y}_{cf}, \mathbf{X}^{\text{obs}}) = \text{o}_1(f(x_{cf}), \hat{y}_{cf}), \text{o}_2(x, x_{cf}), \text{o}_3(x, x_{cf}), \text{o}_4(x, \mathbf{X}^{\text{obs}})$$

Get going on generating counterfactuals, be patient and kind to yourself - this is your first XAI task :)

See you tomorrow!