WhatIf? – Counterfactual Generation for Causal Inference and Model Evaluation

Praharsh Nanavati
Department of Data Science and Engineering
Indian Institute of Science Education and Research, Bhopal
praharsh19@iiserb.ac.in

***Abstract*—Counterfactual thinking is nowadays becoming a new topic of interest for many data scientists worldwide as it has significant contributions in the field of causality and interpretable machine learning.**

Counterfactuals deal with ”What if?” questions described by the following question: Given that the model's output for input x is y, what would be the output if input x is changed to x^*?

The very idea of counterfactuals lies on data instances which never took place but to estimate what would have happened if another decision path had been chosen.

Counterfactual generation is hence an interesting problem which can be tackled in many ways according to the problem. A counterfactual explanation describes a causal situation in the form: “If X had not occurred, Y would not have occurred”. An explanation of a prediction describes the smallest change to the feature values that changes the prediction to a predefined output.

Through this project, we explore counterfactual generation using DiCE-ML [1], which is a library that helps us understand an ML model by generating counterfactual data points that lead to the desired model output.

After generating counterfactual data, to check the consistency of the ML model and the quality of the counterfactuals, we build causal Directed Acyclic Graphs(DAGs) for both the factual and counterfactual data using CausalNex [2] which is built on a recent paper titled 'DAGs with NOTEARS' [3] which converts the combinatorial nature of structure learning of DAGs into a continuous optimization technique to compare the causal inferences made.

To generate counterfactuals, we have used the Palmer Archipelago Penguin classification dataset [4], which contains various features of a penguin's body-type along with it's habitat and surroundings.

***Index Terms*—Counterfactuals, Causality, Interpretability**

I. Introduction

Counterfactual thinking is a concept in psychology that involves the human tendency to create possible alternatives to life events that have already occurred; something that is contrary to what actually happened. Counterfactual thinking is, as it states: ”counter to the facts”.

The author of the book Interpretable ML [5] states that, ”In interpretable machine learning, counterfactual explanations can be used to explain predictions of individual instances”. The “event” is the predicted outcome of an instance, the “causes” are the particular feature values of this instance that were input to the model and “caused” a certain prediction. Displayed as a graph, the relationship between the inputs and the prediction is very simple: The feature values cause the prediction.

Counterfactuals are human-friendly explanations, because they are contrastive to the current instance and because they are selective, meaning they usually focus on a small number of feature changes.

Exploring “what-if” scenarios is an important way to inspect a machine learning (ML) model. The DiCE library helps you to understand an ML model by generating “what-if” data points that lead to the desired model output. Hence, generating counterfactuals using DiCE-ML is easy, convenient and robust.

CausalNex is a Python library that uses Bayesian Networks to combine machine learning and domain expertise for causal reasoning. It can be used to uncover structural relationships in data, learn complex distributions, and observe the effect of potential interventions. This library implements methods included in [3]. This paper mainly talks about learning structures of DAGs as a smooth, continuous optimization function by applying acyclicity constraints instead of a combinatorial problem. This is efficient as looking at the problem of structure learning combinatorially, adding a single node to a DAG can increase the number of possibilities super-exponentially.

To conclude, if a new model is being deployed that might affect what ground truth values are observed, a counterfactual evaluation policy must be thought of prior to the first deployment in production. This will enable the machine learning practitioner to monitor the model as well as have an unbiased data distribution for retraining. We check this by creating causal graphs. The exact procedure is mentioned in the upcoming section.

II. Methodology

A. Requirements

We import all relevant libraries that are used in the implementation of the project. This includes the following:

- Pandas for database management.
- Numpy for vital numerical computations.
- Plotly, Matplotlib and Seaborn for data visualization.
- Sci-kit-Learn for basic machine learning modules including classification.
- DiCE-ML for generating counterfactual data.
- CausalNex for creating causal DAGs from observational and counterfactual data.

B. Loading and Describing Data

We load the data obtained from [4] into a dataframe and describe the data using the $describe()$ and $info()$ functions. Now, using the $isnull().sum()$ function, we check for all nonzero attributes and using the $SimpleImputer$ from the sklearn library, nullify all of them. This gives us cleansed data. Now, the data is set for preprocessing.

C. Data Processing and Visualization

Firstly, we create individual dataframes for all species of penguins for better visualization and easy access of data. Followed by this, we also look at the Species-Based Gender Count for all penguins. Now, we can create plots of different species with different attributes. This was accomplished with substantial help from the plotly and pandas module for penguin data [6].

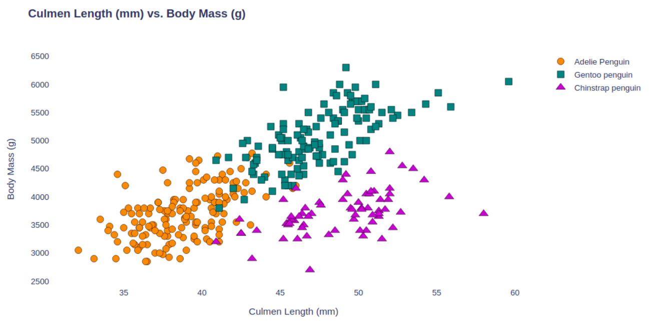

Fig. 1. Plots for Penguin Data for Culmen Length v Body Mass.

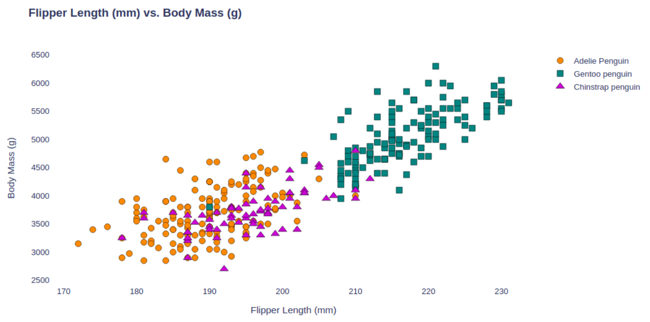

Fig. 2. Plots for Penguin Data for Flipper Length v Body Mass.

To get an idea of outliers, we also have a violin plot for all three Species of penguins. Now the task is to have a model classify these penguins based on other attributes. This will be explored in the next subsection.

D. Data Manipulation for a Classifier Model

Now, since the data is not numeric in nature, we need to cleanse and numerise the data. Firstly, we drop the 'studyName','Individual ID','Region','Date Egg', 'Comments','Delta 15 N (o/oo)','Delta 13 C (o/oo)', 'Stage', 'colour' and 'Sample Number' columns as we want our model

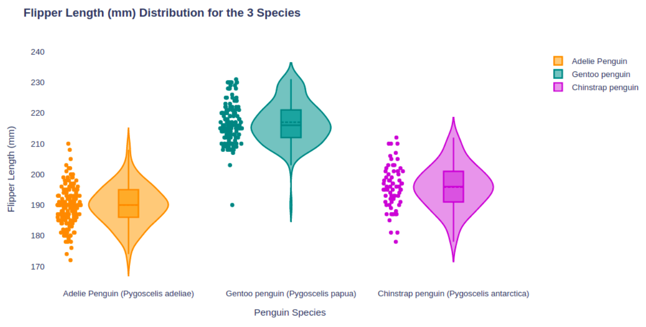

Fig. 3. Violin Plot for all Species.

to be set that way. The features of our interest also need to be converted into binary/ternary numeric variables. We tweak them as follows:

1) Species: Adelie: 0, Chinstrap: 1, Gentoo: 2.
2) Island: Torgersen: 0, Biscoe: 1, Dream: 2.
3) Clutch Completion: Yes: 1, No: 0.
4) Sex: FEMALE: 0, MALE: 1.

The other relevant features: Culmen length, Culmen depth, Flipper length and Body Mass are already numerical in nature. We also drop all rows with NaN instances in them. Now, our dataframe is ready to be used for Modelling.

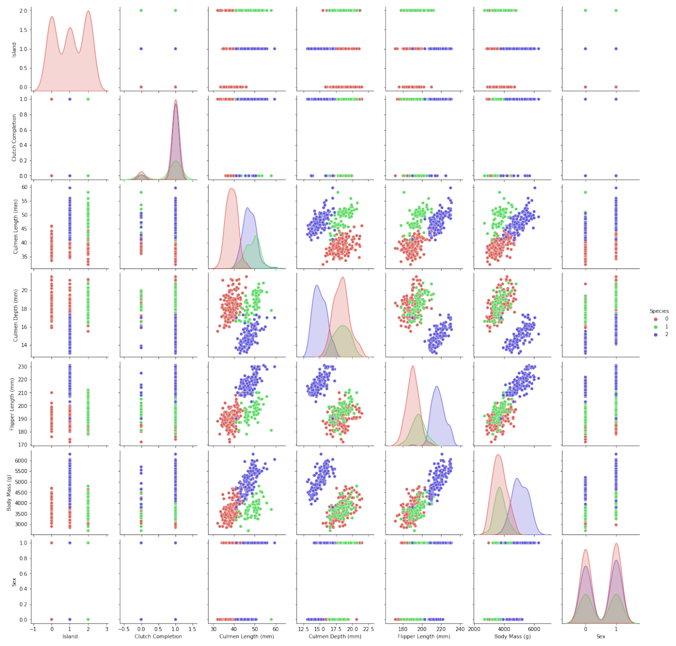

Fig. 4. Seaborn pairplot for numerised factual data.

E. Model

Using inbuilt modules of sklearn, we form a classifier model for our data with Species of the penguins as the target variable and the other features as continuous features. After this, we apply the regular train-test split in our data in the 80-20

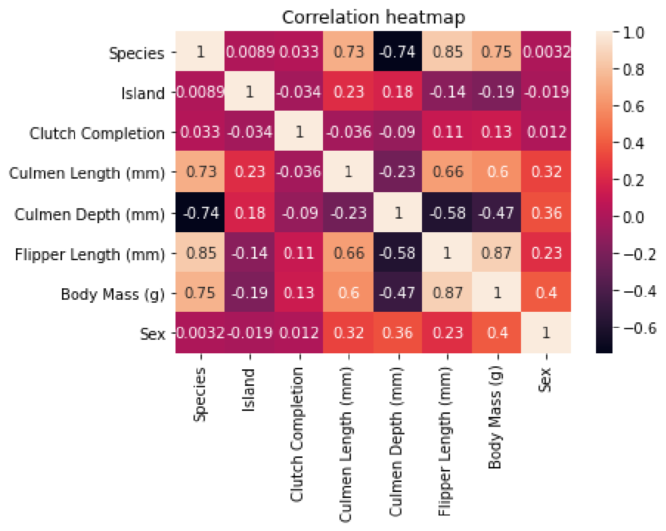

Fig. 5. Heatmap of the factual data pairplot.

ratio. After transforming data using the $StandardScaler$ and $OneHotEncoder$, we train our classification model.

F. DiCE-ML Implementation

Firstly, we specify the data for DiCE where we take the original dataframe and define the target variable and the continuous features. Then we describe our classifier model to DiCE. Then we invoke the explainer of the DiCE library. Now, using $generate\_counterfactuals()$, we get our counterfactuals based on the instances provided through x_train. We create one counterfactual for each instance in x_train and change the desired class accordingly. This would have been simpler if the target variable was binary in nature. However, once the counterfactuals are generated, we store them in a dataframe.
Note: Counterfactuals can be generated in much more efficient ways, with more meaning and for targeted and precise tasks.

G. Counterfactual Data Visualization

Based on the dataframe obtained in the previous subsection, we can try and visualize the counterfactual data. By doing so, we can check if the correlations are similar to the factual data. Also, this will vary for different sets of counterfactual data. In our case, we generated one counterfactual from every factual instance. This is not the best way to generate counterfactuals for causal inference or model evaluation. However, there exists enough research on generating good counterfactuals [7] [8] which we have not included in this project.

H. Causal Inference of Factual and Counterfactual Data

Here we use the CausalNex library to find the causal dependencies within our datasets – both factual and counterfactual. We use the $from\_pandas$ function to define our structural model for our dataframes. After modelling, we get our causal graphs. This process is explained in greater detail in the next section.

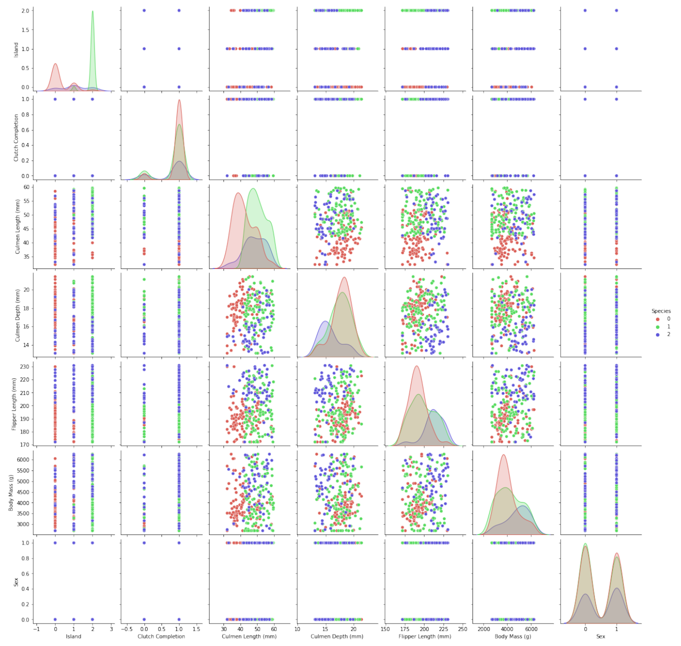

Fig. 6. Seaborn pairplot for numerised counterfactual data.

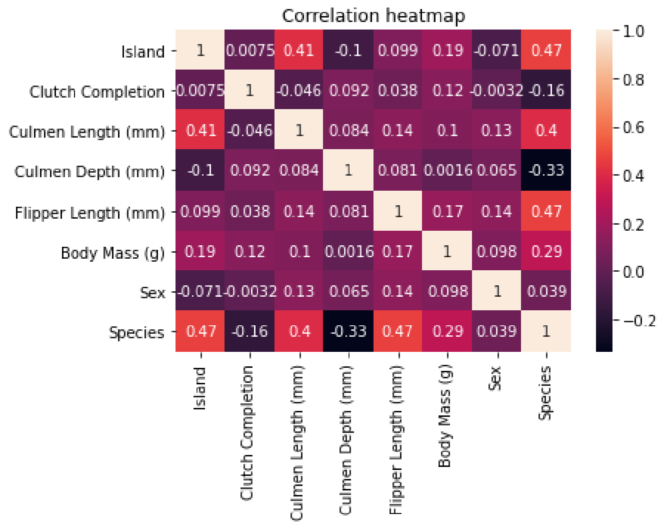

Fig. 7. Heatmap of the counterfactual data pairplot.

III. Results

Using CausalNex, we learn the structure of the causal dependencies in the factual and counterfactual datasets. Firstly, we use the $from\_pandas$ function to define our structural model for our dataframes. Then, we arbitrarily define a threshold for causal edges in the graph so that trivial instances can be removed. After this, we plot the causal graphs for both factuals and counterfactuals.

The output is a resultant Directed Acyclic Graph (DAG). For each set of counterfactuals, the similarity of the causal graphs varies. If the graphs are similar then; Either the model is consistent and the counterfactuals generated are good or

The model is inconsistent and the quality of counterfactuals generated is not good. If the graphs are dissimilar, the counterfactuals generated are good and the model is inconsistent or the counterfactuals generated are inaccurate and the model is consistent.

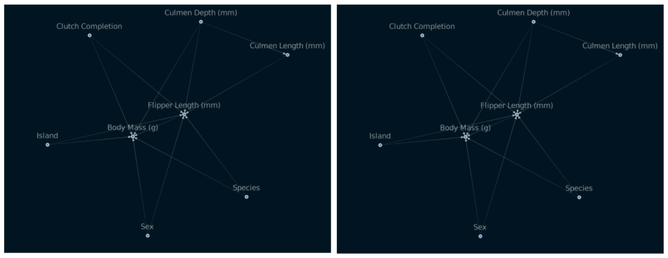

Fig. 8. Case 1: Where the Causal Graphs for Factual and Counterfactual data are identical.

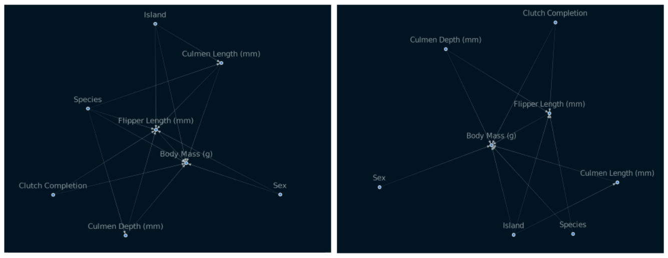

Fig. 9. Case 2: Where the Causal Graphs for Factual and Counterfactual data are different.

This graph was conostructed using a smooth optimization function module as discussed by the authors of 'DAG-Notears' [3].

IV. Conclusions and Future Work

A counterfactual explanation describes a causal situation in the form: "If X had not occurred, Y would not have occurred". For example: "If I hadn't taken a sip of this hot coffee, I wouldn't have burned my tongue". Event Y is that I burned my tongue; cause X is that I had a hot coffee. Thinking in counterfactuals requires imagining a hypothetical reality that contradicts the observed facts (for example, a world in which I have not drunk the hot coffee), hence the name "counterfactual". The ability to think in counterfactuals makes us humans so smart compared to other animals [5].

A counterfactual explanation of a prediction describes the smallest change to the feature values that changes the prediction to a predefined output.

Counterfactuals are human-friendly explanations, because they are contrastive to the current instance and because they are selective, meaning they usually focus on a small number of feature changes.

Through this project, we explore basic counterfactual generation using the DiCE-ML library for multi-class classification on the Penguin Species dataset and check for the causal dependencies in factual and counterfactual data.

In the future, we should be able to evaluate the quality of the counterfactuals generated for better model transparency and improvised causal inferences.

There are a few advantages of Counterfactual generations for explaining models:

- The interpretation of counterfactual explanations is very clear.
- The counterfactual method does not require access to the data or the model.
- The method works also with systems that do not use machine learning.
- The counterfactual explanation method is relatively easy to implement.

However, it is important to note that for each instance we will usually find multiple counterfactual explanations. This is inconvenient and it is also a practical challenge.

On the whole, the notion of counterfactual thinking has changed the outlook of interpretable machine learning but there is still a lot to be done.

References

[1] "Github - Interpretml/Dice: Generate Diverse Counterfactual Explanations For Any Machine Learning Model.". Github, 2021, https://github.com/interpretml/DiCE.

[2] "Welcome To Causalnex'S API Docs And Tutorials! — Causalnex 0.11.0 Documentation". Causalnex.Readthedocs.Io, 2021, https://causalnex.readthedocs.io.

[3] Zheng, X., Aragam, B., Ravikumar, P., & Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. arXiv preprint arXiv:1803.01422.

[4] "Palmer Archipelago (Antarctica) Penguin Data". Kaggle.Com, 2021, https://www.kaggle.com/parulpandey/palmer-archipelago-antarctica-penguin-data.

[5] Molnar, Christoph. "9.3 Counterfactual Explanations — Interpretable Machine Learning". Christophm.Github.Io, 2021, https://christophm.github.io/interpretable-ml-book/counterfactual.html.

[6] "Plotly & Pandas For The Palmer Penguins". Medium, 2021, https://towardsdatascience.com/plotly-pandas-for-the-palmer-penguins-f5cdab3c16c8.

[7] Keane, Mark T., and Barry Smyth. "Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI)." International Conference on Case-Based Reasoning. Springer, Cham, 2020.

[8] Smyth, Barry, and Mark T. Keane. "A Few Good Counterfactuals: Generating Interpretable, Plausible and Diverse Counterfactual Explanations." arXiv preprint arXiv:2101.09056 (2021).