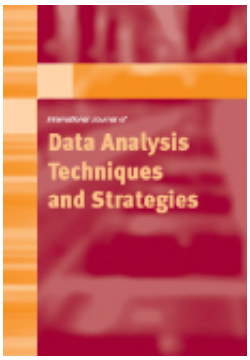


International Journal of Data Analysis Techniques and Strategies

ISSN online: 1755-8069 - ISSN print: 1755-8050
https://www.inderscience.com/ijdats

Machine learning made easy: a beginner's guide for causal inference and discovery methods using Python

Irfan Saleem, Ali Irfan

DOI: 10.1504/IJDATS.2025.10064732

Article History:

Received:	07 March 2024
Last revised:	24 May 2024
Accepted:	24 May 2024
Published online:	14 March 2025

Machine learning made easy: a beginner's guide for causal inference and discovery methods using Python

Irfan Saleem*

Faculty of Business,
Sohar University,
Sohar, PC 311, Sultanate of Oman
ORCID: https://orcid.org/0000-0002-1564-6053
Email: isaleem@su.edu.om
*Corresponding author

Ali Irfan

Department of Computer Science
IU International University of Applied Sciences,
53604, Bad Honnef, Germany
ORCID: https://orcid.org/0009-0004-6427-6687
Email: iali20125@gmail.com

Abstract: Machine learning is widely recognised and extensively used for data modelling and prediction across fields, including business and healthcare, to name a few of them, for informed decision-making. Numerous machine learning algorithms have been devised and deployed across multiple programming languages throughout the preceding decades for causal inference and discovery. This research, however, briefly introduces causal inference and discovery methods, accompanied by Python code for beginners. First, this study talks about machine learning in brief. Then, this study differentiates between causal discovery and causal inference. Thirdly, the study aims to describe popular machine-learning methods. Finally, this paper demonstrates the practical uses of these causal inference and discovery packages in Python. The study has recommended future research and implications for using machine learning methods.

Keywords: Python; machine learning; causal discovery (CD); causal inference (CI); linear regression; Peter-Clark (PC) algorithm; artificial intelligence.

Reference to this paper should be made as follows: Saleem, I. and Irfan, A. (2025) 'Machine learning made easy: a beginner's guide for causal inference and discovery methods using Python', *Int. J. Data Analysis Techniques and Strategies*, Vol. 17, No. 1, pp.36–53.

Biographical notes: Irfan Saleem is an Associate Professor at Sohar University in Oman and is currently pursuing an online Master's degree in Artificial Intelligence at IU International Applied Science University in Germany. He has extensive teaching experience in Gulf, Asian, and European institutes of higher education using the case study method. He publishes papers in reported journals indexed by the Chartered Association of Business Schools (ABS),

Scopus, the Social Sciences Citation Index (SSCI), and the Australian Business Deans Council (ABDC). His integrated research areas include artificial intelligence, machine learning, big data, family SMEs, digital transformation, digital model innovation, and digital entrepreneurship.

Ali Irfan is affiliated with the Department of Computer Science at IU International University of Applied Sciences, Germany as a student for an online degree. He is passionate about programming using Python and his is interested to research in the area of computer science along with his freelancing services to develop various IT related projects. He is also planning to enrol for undergraduate of Electrical and Computer Engineering at Sohar University of Oman.

1 Introduction

Since its inception, exploring the natural world has been a driving force behind human understanding and knowledge. Specifically, the investigation into the origins of natural events captivated several philosophers in ancient Greece (Nogueira et al., 2022). In contemporary times, the investigation of causality encompasses several fields of research to address inquiries related to the reasons behind phenomena. Over the last several years, examining causal connections has also become an essential component of the artificial intelligence (AI) field, including data scientists and machine learning specialists.

In statistics and, lately, in machine learning, thoroughly examining the causal mechanisms of those predicting various outcomes is of utmost importance for people, professionals, and organisations. Usually, data engineers and scientists use future event predictions by assuming the normality of the data using linear and non-linear regressions. For example, structural equation modelling (SEM) can be used to predict how well a company will do financially by adopting artificial intelligence. Business analysts can explore the direct and mediating causal acyclic relationship between cause and effect. For instance, a recent study tested the role of AI adoption for organisational resilience using SEM, assuming that the data is normally distributed (Saleem et al., 2023). However, it is worth noting that traditional linear regression (Scikit-Learn[1]) and SEM (SEMOPY[2]) have limitations in accurately estimating causality for the data without the symmetric and bell-shaped pattern of the Gaussian properties (Shimizu et al., 2006, 2011). This is because methods like SEM depend on the co-variance structure of the data and assume that the dataset has a Gaussian or normal distribution (Shimizu, 2014). Therefore, recently, a lot of new techniques have been put forward in the field of data science and artificial intelligence for using the non-Gaussian features of data to figure out how variables are related for better causal discovery because sometimes the data being used does not fulfil the normality assumptions of regression during testing and training of machines (Ikeuchi et al., 2023).

The study contributes in multiple ways. First, we have uniquely used machine learning and AI methods with Python codes. Second, we have used examples from the field of business to demonstrate the application of casual interference and Discovery to

make it more applicable to address further calls for research and its application in multiple studies (Wang et al., 2024a; Gupta and Sharma, 2023; Utkin and Zhuk, 2012). Last but not least this research study implies that the field of data science and AI is still in its infancy regarding business and other related fields. Therefore this study has recommended future research in the last section of this paper.

The present study encompasses five primary aims. So, in Section 1, this study briefly talks about machine learning and reviews causal discovery. Then, this study differentiates between causal discovery and causal inference. Thirdly, this study describes a list of popular machine-learning methods both for causal inference and discovery. Additionally, this study demonstrates the practical uses of these causal discovery packages in Python. Finally is the conclusion section along with future research recommendations.

2 Causal inference and discovery

Causality research may be broadly categorised into two primary branches: causal discovery (CD) and causal inference (CI). The CD method primarily focuses on acquiring causal information directly from observational data (Nogueira et al., 2022). At the same time, the CI seeks to estimate the effect resulting from a modification of a specific variable on an outcome of interest. In machine learning, causal inference should determine cause-and-effect relationships between variables, not just correlation, as we usually use beta values of independent variables in regression (Pearl, 1995). To predict stock performance, causal factors should be identified as specific interventions that affect outcomes (financial returns), not just statistical connections between market conditions and financial returns. Thus, we can say that causal inference can help to make smarter choices, take corrective action, and understand how independent variables could possibly affect complex systems at their core (Pearl, 2010). However, the goal of causal discovery in data science is to derive a causal structure or set of potential causal models from the data available for testing and training (Kalainathan et al., 2020). To put it another way, given a test dataset, data scientists should construct a causal model that adequately represents the potential reality.[3] Before proceeding further, let us clarify the difference between causal inference and discovery. See Table 1, which is compiled based on the latest study (Nogueira et al., 2022).

Table 1 Differences between causal inference and causal discovery

Comparison basis	*Causal inference* (*CI*)	*Causal discovery* (*CD*)
Definition	The CI is the process of concluding causal relationships based on observed data	The CD is the task of identifying the underlying causal structure among variables
Goal	CI's goal is to make valid and reliable statements about the causal effect of one variable on another	CD aims to uncover the causal relationships or dependencies between variables from observational data
Approach	CI relies on statistical methods, experimental design, and randomised controlled trials	It involves algorithms and statistical methods to identify patterns and dependencies in data that suggest causal relationships

Table 1 Differences between causal inference and causal discovery (continued)

Comparison basis	*Causal inference* (*CI*)	*Causal discovery* (*CD*)
Data requirements	CI often requires controlled experiments or interventions	CD can work with observational and test datasets, making it suitable for scenarios where experiments are not feasible
Example	In CI, an analyst may conduct a randomised controlled trial to assess digitalisation's causal effect on firm performance	In CD, the job of a data scientist is to provide a machine-learning model using an algorithm to determine causality between temperature and tea sales using test data, which can be learned over time, and discover seasonal trends in selling coffee

Source: Authors Notes from review of Nogueira et al. (2022)

3 Machine learning methods

Causality has emerged as a valuable instrument for addressing certain constraints in correlation-based machine learning systems. As a branch of artificial intelligence, machine learning should include techniques, methods, and models to learn and make predictions based on the test and training data to get the desired results (Hao and Ho, 2019). For example, the causal discovery model could help people make better decisions (Kalainathan et al., 2020; Glymour et al., 2019). According to another study, causal inference is the process of determining cause and effect that relies significantly on creating a reference model that solidifies the knowledge obtained (Zanga et al., 2022). In order to fulfil this requirement, causal discovery offers a range of techniques that can retrieve a graphical representation of the underlying process by utilising both gathered data and existing knowledge (Zanga et al., 2022). The causal discovery has diverse implications in healthcare, government, stock markets, and multinational companies for effective decision-making, avoiding future financial losses, and taking proactive measures (Hao and Ho, 2019).

3.1 Acyclic and cyclic causal discovery

Closed loops or cycles generate cyclic causality. Events or factors may directly influence each other or generate feedback loops where one event affects the initial events (Glymour et al., 2019). For instance, economic systems can exhibit cyclic causality. Take an example of consumer spending, which will increase manufacturing, which then boosts employment and income. Finally, with cyclic causality logic, the income growth of consumers can boost consumer spending, producing a feedback loop.

If there are no loops or cycles in the cause-and-effect chain, the connection or system is said to be acyclic (Glymour et al., 2019). Contrastingly, independent variables, factors, or causes only affect other events in one direction without creating closed loops. Consider a scenario. There are two major economic factors in a small town: 'demand for housing'

and 'housing prices'. A causal one-direction or acyclic link can be observed because as 'demand for housing' rises, so do "housing prices".

Over the last three decades, scholars have discussed essential causal discovery methods (Malinsky and Danks, 2018). Most are being coded in Python as packages lately (Glymour et al., 2019). These discovery methods range from score-based and constraint-based methods (conditional independence) to those based on a broader category like SEM, also called functional causality. First, a score-based method for finding causes is called a constraint-based method (Spirtes, 2010). This includes greedy equivalence search (GES). Then constraint-based approaches include PC (Peter and Clark)[4] and its extensions, fast causal inference (FCI).[5] Finally, Linear Non-Gaussian Acyclic Model (LiNGAM), Post Nonlinear (PNL), and Additive Noise Model (ANM) belong to a broader category of causal functionality like SEM. For modelling and analysing causal links, we can also use graph libraries in Python. One of the libraries frequently utilised for this purpose is NetworkX, which can perform analysis on both acyclic and cyclic graphs (Vasiliauskaite et al., 2022).

4 Machine learning methods of causal inference and causal discovery

4.1 Using regression for causal inference

This section can be helpful for beginners to first apply the causal inference in Python using simple regression in machine learning before we present the causal discovery methods. Causal inference (CI) is a step-by-step process for concluding causal relationships based on observed data (Shimizu, 2014; Utkin and Zhuk, 2012). Many packages in Python can be used to implement CI using linear regression. However, for this section, we have adopted scikit-learn.[6] The scikitlearn package provides multiple options to apply linear regression. For instance, linear regression can be used in more than a dozen ways in Python. This package includes Elastic Net, Huber Regressor, Passive Aggressive Regressor, and many more. There are many other regressions, but they are beyond the scope of this short review. For instance, Support Vector Regression, Decision Tree Regression, K-Nearest Neighbours Regression, and Neural Network Regression. This study has provided four sample codes available in Listings 1–4 for machine learning beginners about regressions, including linear, Lasso, polynomial, and Ridge. However, our key focus is to provide some coding examples of causal discovery methods in machine learning covered in subsequent sections. Please note that these examples are randomly generated for training and testing purposes. Table 2 summarises the characteristics and use cases for four basic types of regression to help beginners understand the potential application of simple regression in machine learning.

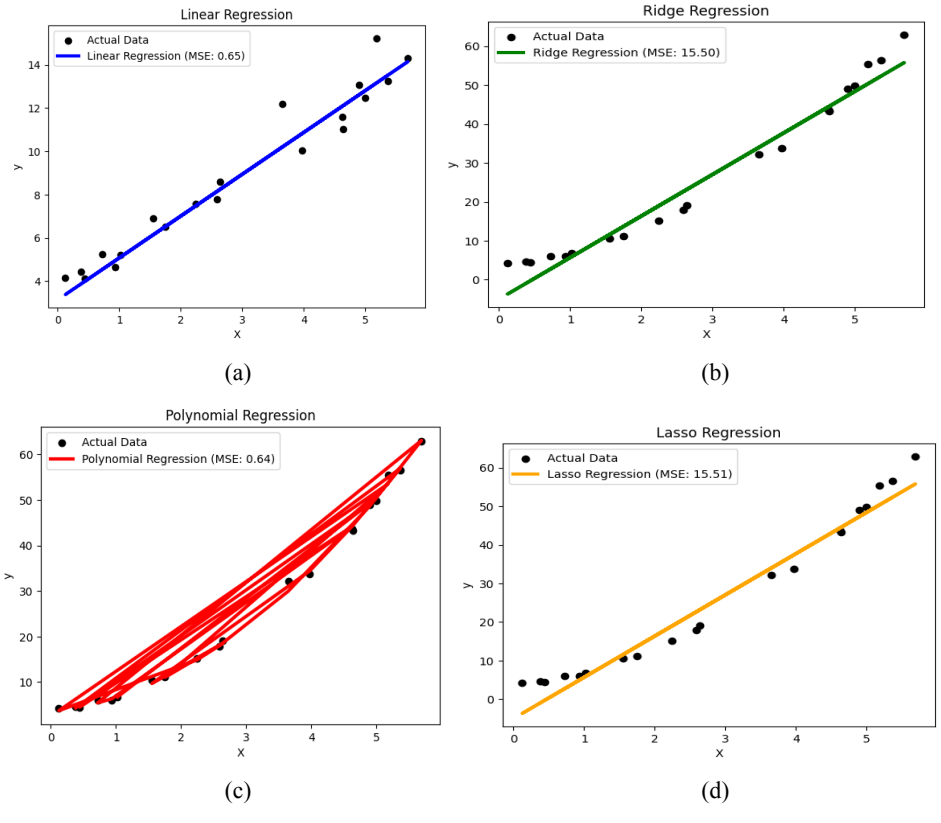
Moreover, Figure 1 shows the output of four types of regressions. The conventional method of causal delivery is based on constraints and scores (Malinsky and Danks, 2018; Glymour et al., 2019; Kalainathan et al., 2020). For instance, the PC algorithm, FCI, and GES (Zanga et al., 2022).

Table 2 Regression types, characteristics and use cases for causal inference

Regression type	*Characteristics*	*Use case in business*
Linear regression (LR)	The LR assumes a linear relationship between the features, and data scientists can easily interpret the results	LR can be used to predict sales based on advertising spending or estimate the price of a house based on its features
Polynomial regression (PR)	The PR extends linear regression by introducing polynomial features to capture non-linear relationships	PR can predict the number of defects in a manufacturing process based on certain factors
Ridge regression (RR)	The RR introduces regularisation to the LR by adding a penalty term to the cost function to prevent overfitting	PR can help forecast stock prices with multiple interrelated variables
Lasso regression	LaR is another form of regularised LR that uses L1 regularisation, leading to sparse feature selection	LaR can predict customer churn in a subscription-based service with various features

Source: Authors Notes

Figure 1 Output for four types of regressions: (a) simple linear; (b) ridge; (c) poly and (d) lasso (see online version for colours)



(a) (b)

(c) (d)

4.2 Peter-Clark algorithm

The Peter and Clark (PC) algorithm is one of the earliest algorithms that remains consistent when subjected to independent and identically distributed (IID) sampling, provided there are no latent confounders. There are four key assumptions for the PC algorithm (Spirtes et al., 2000).

1 The Faithfulness Assumption states that the statistically independent correlations found in the data accurately reflect the causal structure that exists. The Markov Assumption states that in a causal graph, every variable is independent of its non-descendants while considering its parents.

2 Algorithmic Markov Condition states that the method accurately detects conditional independence relationships that align with the Markov assumption.

3 The technique employs statistical tests for conditional independence that effectively ascertain the independence of variables when conditioned on another set of variables.

4 *Absence of hidden variables*: The algorithm operates under the assumption that there are no unseen or hidden variables that exert an influence on the observed variables.

4.2.1 Application of PC algorithm for causal discovery in Python

The Peter and Clark (PC) algorithm initiates with a graph that has all possible connections between nodes and thereafter executes a sequence of steps to eliminate edges, relying on the independent structure of the network. Ultimately, it endeavours to align a maximum number of edges. Referring to the first two CODE Listings 5 and 6, the study has used the code to implement the impact of employee training and technology adoption on the employees' task performance. Then, the task performance of employees results in organisational performance. The study used randomly generated data using Python code, which results in the following output by following the steps presented in the previous research (Glymour et al., 2019). Moreover, Figure 2 shows the production of the Python code.

4.3 Non-Gaussian models

4.3.1 Linear non-Gaussian acyclic model for causal discovery

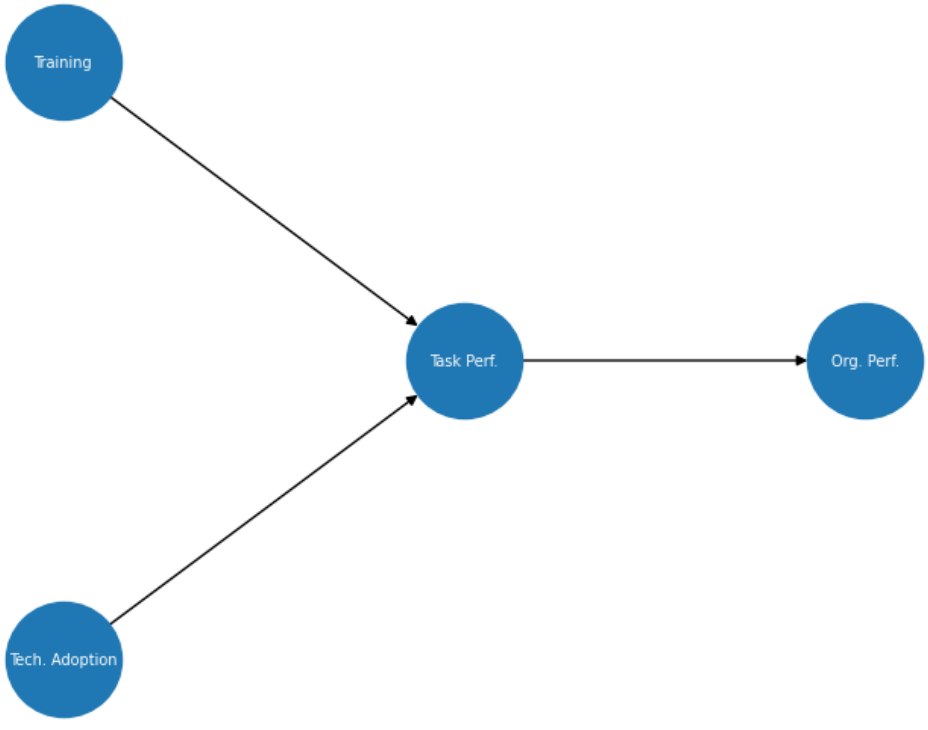
Non-Gaussian data has become more common in machine learning in recent years because the data distribution may hold more information than the covariance matrix can adequately show, and one of them is LiNGAM. The LiNGAM encompasses many machine-learning techniques (Shimizu et al., 2006). This feature renders it a practical toolkit for statisticians specialising in educational and behavioural research. LiNGAM stands for Linear Non-Gaussian Acyclic Model. It works by making certain assumptions about the underlying structure of the data:

1 The data is linear, in that changes in one variable are directly proportional to changes in another.

2 The data is non-Gaussian, in that it does not follow the normal/Gaussian distribution.

3 The data is acyclic, in that there are no cycles or loops in the causal relationships.

Figure 2 PC algorithm (see online version for colours)



4.3.2 Application of LiNGAM package for causal discovery in Python

Developers have implemented non-Gaussian causal discovery techniques using programming languages, including Python libraries, e.g., LiNGAM.[7] This section discusses the newly developed Python module designed for causal discovery called LiNGAM and its essential functions.

4.3.3 Installation of LiNGAM package

The data scientist needs to run the commands available in Listing 7 to install the LiNGAM package in Python on the terminal using VScode or Jupyter Notebook: This study has used official tutorials to introduce various models of LiNGAM.[8] This study used real-world and imaginary scenarios to understand acyclic causal discovery effectively using dummy data.

4.3.4 Application of DirectLiNGAM in Python

The DirectLiNGAM approach estimates a causal order of variables by gradually removing each independent component's influence from the data provided in the machine model (Shimizu et al., 2011). The number of steps required in the DirectLiNGAM to finish this process is proportional to the number of variables considered in the model. This study adopted a dataset in sociology from an older study (Shimizu et al., 2011). Refer to the following figure to see the application of DirectLiNGAM to estimate a

causal ordering of variables to predict Son's income (refer to Listing to apply the code in Python). The dataset had six observed variables, and matrix values were adopted from an older study (Shimizu et al., 2011). According to the survey in the study of Shimizu et al. (2011), a1 represents the father's profession level, a2 represents the son's income, a3 represents the father's education, a4 represents the son's occupation level, a5represents the son's education, and a6 represents the number of siblings. The sample selection was carried out according to the following criteria:

a individuals without experience in agriculture

b individuals between the ages of 35 and 44

c individuals of Caucasian ethnicity

d male individuals

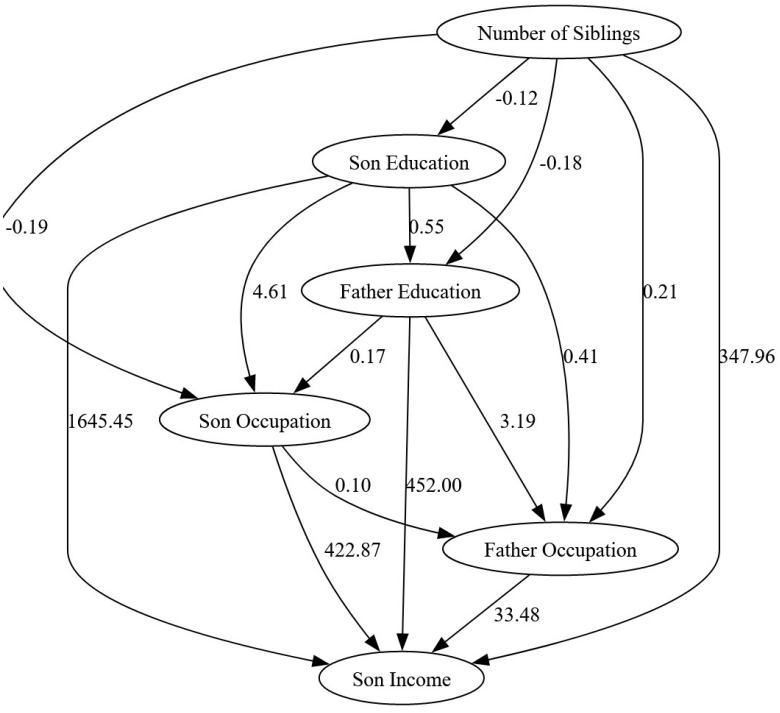
e individuals who were actively employed or seeking employment at the time of the survey

f individuals with complete data for all variables

g data collected between the years 1972 and 2006.

The size of the sample was 1380. Following is the output based on code (See Listing 8). Referring to Figure 3, all of the directed edges estimated by DirectLiNGAM were consistent with the domain knowledge, except for the directed edge from a5: son's education to a3: father's education.

Figure 3 LiNGAM



4.4 Other hybrid methods of causal discovery

4.4.1 Fast causal inference algorithm

The FCI method produces accurate results that are valid even when confounding factors are present (Glymour et al., 2019). As stated earlier, the PC algorithm operates under the assumption that there are no unobserved variables that could potentially influence both of the measured variables (Spirtes et al., 2000; Glymour et al., 2019). Similar to the PC algorithm, other versions of the FCI method exist. Mostly, FCI aims to enhance speed by sacrificing some amount of information. One such example is the recursive fast causal inference (RFCI) algorithm. The key library for testing the FCI algorithm and sample code are available online.[9]

4.4.2 The greedy equivalence search

The GES algorithm differs from PC and FCI. (Glymour et al., 2019) The GES initially uses an empty graph instead of a complete undirected graph (Chickering, 2002). The GES then selectively adds necessary edges and subsequently systematically removes unneeded edges. At each phase of the process, a choice is taken regarding whether to add a directed edge to the graph based on its potential to increase the fit, which is quantified by a quasi-Bayesian score like the Z-score, which is similar to a statistical hypothesis test. Subsequently, the obtained model is associated with the matching Markov equivalence class. Once the score can no longer be enhanced, the GES algorithm proceeds to evaluate each edge individually to determine if removing it will result in the greatest improvement in the score. This process continues until no more edges can be removed to enhance the score further. An additional variation of the GES is the greedy fast causal inference (GFCI) GFCI (Glymour et al., 2019). According to research, the GFCI is a composite of GES and FCI. The GFCI method has demonstrated superior accuracy in numerous simulations compared to the original FCI algorithm (Zanga et al., 2022). The key library for testing the GES and sample code are available online.[10]

5 Conclusion and future research

This paper used Python codes to present causal inference, discovery theory, and algorithms for beginners. There are four key contributions of this research. The first contribution is to introduce the difference between causal inference and discovery. The second contribution of this study is to present the literature review on the methods for causal inference and discovery, including Linear regression, PC, FCI, LiNGAM, and GES. The top journals, like the *Journal of Machine Learning*, were reviewed to ensure the quality of this study. Finally, this study provided listings of codes using popular Python packages.

Overall, this provides a comprehensive literature review, and GitHub links two hybrid types of causal discovery methods, i.e., GES and GFCI. Data scientists can look closer and develop application codes for these causal discovery machine learning methods. At the same time, machine learning scholars and artificial intelligence scholars can think

about more applications and extensions of packages like LiNGAM to address future calls for further research (Zheng et al., 2024; Ikeuchi et al., 2023). For instance, lately, two studies compared the healthcare systems of eight developed nations using Support Vector Regression of machine learning (ML) methods as the latest applications (Wang et al., 2024a, 2024b). Moreover, the bibliometric analysis of supervised deep learning for the field of psychology and related fields is recommended (Gupta and Sharma, 2023; Uddin and Chowdhury, 2024). Finally, Old classical methods of ML-like logic regression can be reviewed using Python and other ML languages for data science and artificial intelligence.

References

Chickering, D.M. (2002) 'Optimal structure identification with greedy search', *Journal of Machine Learning Research*, Vol. 3, November, pp.507–554.

Glymour, C., Zhang, K. and Spirtes, P. (2019) 'Review of causal discovery methods based on graphical models', *Frontiers in Genetics*, Vol. 10, p.524.

Gupta, G.K. and Sharma, D.K. (2023) 'Depression detection using semantic representation based semi-supervised deep learning', *International Journal of Data Analysis Techniques and Strategies*, Vol. 15, No. 3, pp.217–237.

Hao, J. and Ho, T.K. (2019) 'Machine learning made easy: a review of scikit-learn package in Python programming language', *Journal of Educational and Behavioral Statistics*, Vol. 44, No. 3, pp.348–361.

Ikeuchi, T., Ide, M., Zeng, Y., Maeda, T.N. and Shimizu, S. (2023) 'Python package for causal discovery based on LiNGAM', *Journal of Machine Learning Research*, Vol. 24, pp.1–8.

Kalainathan, D., Goudet, O. and Dutta, R. (2020) 'Causal discovery toolbox: uncovering causal relationships in Python', *Journal of Machine Learning Research*, Vol. 21, No. 1, pp.1406–1410.

Malinsky, D. and Danks, D. (2018) 'Causal discovery algorithms: a practical guide', *Philosophy Compass*, Vol. 13, No. 1, pp.1–11.

Nogueira, A.R., Pugnana, A., Ruggieri, S., Pedreschi, D. and Gama, J. (2022) 'Methods and tools for causal discovery and causal inference', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 12, No. 2, p.e1449.

Pearl, J. (1995) 'Causal diagrams for empirical research', *Biometrika*, Vol. 82, No. 4, pp.669–688.

Pearl, J. (2010) 'The foundations of causal inference', *Sociological Methodology*, Vol. 40, No. 1, pp.75–149.

Saleem, I., Hoque, S.M.S., Tashfeen, R. and Weller, M. (2023) 'The interplay of AI adoption, IoT edge, and adaptive resilience to explain digital innovation: evidence from German family-owned SMEs', *Journal of Theoretical and Applied Electronic Commerce Research*, Vol. 18, No. 3, pp.1419–1430.

Shimizu, S. (2014) 'LiNGAM: non-Gaussian methods for estimating causal structures', *Behaviormetrika*, Vol. 41, pp.65–98.

Shimizu, S., Hoyer, P.O., Hyvärinen, A., Kerminen, A. and Jordan, M. (2006) 'A linear non-Gaussian acyclic model for causal discovery', *Journal of Machine Learning Research*, Vol. 7, No. 10, pp.2003–2030.

Shimizu, S., Inazumi, T., Sogawa, Y., Hyvarinen, A., Kawahara, Y., Washio, T. and Hoyer, P. (2011) 'DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model', *Journal of Machine Learning Research*, Vol. 12, pp.1225–1248.

Spirtes, P. (2010) 'Introduction to causal inference', *Journal of Machine Learning Research*, Vol. 11, No. 5, pp.1643–1662.

Spirtes, P., Glymour, C.N. and Scheines, R. (2000) *Causation, Prediction, and Search*, MIT Press, Cambridge, Massachusetts London, England.

Uddin, M.S. and Chowdhury, M. (2024) 'A user friendly anger and anxiety dis-order prediction scheme using machine learning and a mobile application for mental healthcare', *International Journal of Data Analysis Techniques and Strategies*, Vol. 16, No. 1, pp.47–81.

Utkin, L.V. and Zhuk, Y.A. (2012) 'A machine learning algorithm for classification under extremely scarce information', *International Journal of Data Analysis Techniques and Strategies*, Vol. 4, No. 2, pp.115–133.

Vasiliauskaite, V., Evans, T.S. and Expert, P. (2022) 'Cycle analysis of directed acyclic graphs. *Physica A: Statistical Mechanics and Its Applications*, Vol. 596, pp.1–22.

Wang, J., Qin, Z., Hsu, J. and Zhou, B. (2024a) 'A fusion of machine learning algorithms and traditional statistical forecasting models for analyzing American healthcare expenditure', *Healthcare Analytics*, Vol. 5, p.100312.

Wang, J., Pei, Z.K., Wang, Y. and Qin, Z. (2024b) 'An investigation of income inequality through autoregressive integrated moving average and regression analysis', *Healthcare Analytics*, Vol. 5, June, p.100287.

Zanga, A., Ozkirimli, E. and Stella, F. (2022) 'A survey on causal discovery: theory and practice', *International Journal of Approximate Reasoning*, Vol. 151, pp.101–129.

Zheng, Y., Huang, B., Chen, W., Ramsey, J., Gong, M., Cai, R. and Zhang, K. (2024) *Causal-Learn: Causal Discovery in Python*, Vol. 25, No. 60, pp.1−8.

Notes

[1]A Python Package scikit-learn: https://scikit-learn.org/ (Accessed 25 March, 2024).

[2]A Python package for Structural Equation Modelling: https://semopy.com/ (Accessed 26 March, 2024).

[3]A Causal Discovery Toolbox: https://github.com/FenTechSolutions/CausalDiscoveryToolbox (Accessed 26 March, 2024).

[4]A Python implementation of the PC algorithm: https://github.com/Black-Swan-ICL/PyPCAlg (Accessed 27 March, 2024).

[5]Causal discovery algorithms and tools: https://github.com/IntelLabs/causality-lab (Accessed 28 March, 2024).

[6]A set of Python modules for machine learning and data mining: https://pypi.org/project/scikit-learn/ (Accessed 28 February, 2024).

[7]LiNGAM: A new method for estimating structural equation models: https://pypi.org/project/ LiNGAM (Accessed 26 February, 2024).

[8]LiNGAM Tutorials: https://LiNGAM.readthedocs.io/en/latest/ (Accessed 21 February, 2024).

[9]An index of algorithms for learning causality with data: https://github.com/rguo12/awesome-causality-algorithms (Accessed 2 April, 2024).

[10]Python implementation of the GES algorithm for causal discovery: https://github. com/juangamella/ges (Accessed 20 April, 2024).

Appendix A: Python codes listings for paper

Listing 1 Python code for simple linear regression using scikit-learn (see online version for colours)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Generate synthetic data
np.random.seed(42)
X = 6 * np.random.rand(100, 1)
y = 3 + 2 * X + np.random.randn(100, 1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Linear Regression
try:
    linear_model = LinearRegression()
    linear_model.fit(X_train, y_train)
    y_pred_linear = linear_model.predict(X_test)
    mse_linear = mean_squared_error(y_test, y_pred_linear)

    # Visualize the linear regression line
    plt.scatter(X_test, y_test, color='black', label='Actual Data')
    plt.plot(X_test, y_pred_linear, color='blue', linewidth=3,
    label=f'Linear Regression (MSE: {mse_linear:.2f})')

    # Plot settings
    plt.title('Linear Regression')
    plt.xlabel('X')
    plt.ylabel('y')
    plt.legend()
    plt.show()
except Exception as e:
    print(f"Linear Regression Error: {e}")
```

Listing 2 Python code for lasso regression using scikit-learn (see online version for colours)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error

# Generate synthetic data
np.random.seed(42)
X = 6 * np.random.rand(100, 1)
y = 3 + 2 * X + 1.5 * X**2 + np.random.randn(100, 1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Lasso Regression
try:
    lasso_model = Lasso(alpha=0.1)
    lasso_model.fit(X_train, y_train)
    y_pred_lasso = lasso_model.predict(X_test)
    mse_lasso = mean_squared_error(y_test, y_pred_lasso)

    # Visualize the lasso regression line
    plt.scatter(X_test, y_test, color='black', label='Actual Data')
    plt.plot(X_test, y_pred_lasso, color='orange', linewidth=3,
    label=f'Lasso Regression (MSE: {mse_lasso:.2f})')

    # Plot settings
    plt.title('Lasso Regression')
    plt.xlabel('X')
    plt.ylabel('y')
    plt.legend()
    plt.show()
except Exception as e:
    print(f"Lasso Regression Error: {e}")
```

Listing 3 Python code for ridge regression using scikit-learn (see online version for colours)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error

# Generate synthetic data
np.random.seed(42)
X = 6 * np.random.rand(100, 1)
y = 3 + 2 * X + 1.5 * X**2 + np.random.randn(100, 1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
=0.2, random_state=42)

# Ridge Regression
try:
    alpha = 1.0 # You can adjust the regularization strength
    ridge_model = Ridge(alpha=alpha)
    ridge_model.fit(X_train, y_train)
    y_pred_ridge = ridge_model.predict(X_test)
    mse_ridge = mean_squared_error(y_test, y_pred_ridge)

    # Visualize the ridge regression line
    plt.scatter(X_test, y_test, color='black', label='Actual Data')
    plt.plot(X_test, y_pred_ridge, color='green', linewidth=3,
    label=f'Ridge Regression (MSE: {mse_ridge:.2f})')

    # Plot settings
    plt.title('Ridge Regression')
    plt.xlabel('X')
    plt.ylabel('y')
    plt.legend()
    plt.show()
except Exception as e:
    print(f"Ridge Regression Error: {e}")
```

Listing 4 Python code for polynomial regression using scikit-learn (see online version for colours)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error

# Generate synthetic data
np.random.seed(42)
X = 6 * np.random.rand(100, 1)
y = 3 + 2 * X + 1.5 * X**2 + np.random.randn(100, 1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Polynomial Regression
try:
    degree = 2 # You can adjust the degree of the polynomial
    poly_model = make_pipeline(PolynomialFeatures(degree),
    LinearRegression())
    poly_model.fit(X_train, y_train)
    y_pred_poly = poly_model.predict(X_test)
    mse_poly = mean_squared_error(y_test, y_pred_poly)

    # Visualize the polynomial regression line
    plt.scatter(X_test, y_test, color='black', label='Actual Data')
    plt.plot(X_test, y_pred_poly, color='red', linewidth=3,
    label=f'Polynomial Regression (MSE: {mse_poly:.2f})')

    # Plot settings
    plt.title('Polynomial Regression')
    plt.xlabel('X')
    plt.ylabel('y')
    plt.legend()
    plt.show()
except Exception as e:
    print(f"Polynomial Regression Error: {e}")
```

Listing 5 Installation of packages to run PC algorithm (see online version for colours)

```
$ pip install numpy networkx ges gcastle==1.0.3rc3 matplotlib torch
# Python 3.7.9 or earlier is required to use the Torch package and run
the PC algorithm code
```

Listing 6 Python code for PC algorithm (see online version for colours)

```
import os
os.environ['CASTLE_BACKEND'] = 'pytorch'

from collections import OrderedDict
import warnings
import numpy as np
import networkx as nx
import ges
from castle.common import GraphDAG
from castle.metrics import MetricsDAG
from castle.datasets import IIDSimulation, DAG
from castle.algorithms import PC, ICALiNGAM, GOLEM
import matplotlib.pyplot as plt

warnings.simplefilter('ignore')
def get_n_undirected(g):
        total = 0
    for i in range(g.shape[0]):
        for j in range(g.shape[0]):
            if (g[i, j] == 1) and (g[i, j] == g[j, i]):
                total += .5
    return total

# Let's implement this PC structure using an example from the field of
business using dummy data
training = np.random.randn(1000)
tech_adoption = np.random.randn(1000)
task_perf = training + tech_adoption + .1 * np.random.randn(1000)
org_perf = .7 * task_perf + .1 * np.random.randn(1000)
pc = PC()
pc.learn(pc_dataset)

# Get learned graph
learned_graph = nx.DiGraph(pc.causal_matrix)

# Relabel the nodes
MAPPING = {k: v for k, v in zip(range(4), ['Training', 'Tech.
Adoption', 'Task Perf.', 'Org. Perf.'])}
learned_graph = nx.relabel_nodes(learned_graph, MAPPING, copy=True)

# Position nodes on graph pos = {
    "Training" : np.array([0, 1]),
    "Tech. Adoption" : np.array([0, -1]),
    "Task Perf." : np.array([1, 0]),
    "Org. Perf." : np.array([2, 0])
}

# Plot the PC Node graph
nx.draw(
    learned_graph,
    pos=pos,
    with_labels=True,
    node_size=3000,
    font_size=7,
    font_color='white'
)
```

Listing 7 Installation of LiNGAM package (see online version for colours)

```
$ pip install lingam
$ pip install numpy
$ pip install pandas
```

Listing 8 Python code for DirectLiNGAM (see online version for colours)

```
import numpy as np
import pandas as pd
import graphviz
import lingam
from lingam.utils import make_dot

# Adjacency matriX
m = np.array([
      [0.0, 0.0, 3.19, 0.10, 0.41, 0.21],
      [33.48, 0.0, 452.0, 422.87, 1645.45, 347.96],
      [0.0, 0.0, 0.0, 0.0, 0.55, -0.18],
      [0.0, 0.0, 0.17, 0.0, 4.61, -0.19],
      [0.0, 0.0,0.0, 0.0, 0.0, -0.12],
      [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
])
l1 = ['Father Occupation','Son Income','Father Education','Son
Occupation','Son Education','Number of Siblings']
model = lingam.DirectLiNGAM()
dot = make_dot(m, labels=l1)

# Save png
dot.format = 'png'
dot.render('dag')
dot
```