SUPERVISED MACHINE LEARNING: USE CLASSIFICATION ALGORITHMS TO ACCURATELY IDENTIFY CUSTOMERS

PhD. Nguyen Hung Cuong[1], PhD. Tran The Tuan[2]

[1,2]University of Transport Technology

[1]E-mail: cuongnh@utt.edu.vn, [2] E-mail: tuantt83@utt.edu.vn

Abstract

Determining the exact customer is the most important factor affecting business operations of the enterprise, helping businesses increase selling ability, reduce costs and increase competitiveness. This article focuses on the classification algorithms used in Supervised Machine Learning to identify customers through a specific data set. Through the article, readers will be easier to access knowledge about Machine Learning, algorithms, as well as programming symbols in Python programming language from which can be applied in practice.

Keywords: Artificial intelligence, Supervised Machine Learning, classification algorithms, customers

1. INTRODUCTION

With the strong development of technology and different marketing channels, businesses are facing a huge challenge when dealing with a huge amount of customer data. Big companies like Amazon, Facebook, Google ... are the world leaders in the application of Artificial Intelligence (AI) into the business. Thanks to the best advantage of the AI system, it is the ability to self-study and continuous improvement through the process of testing and evaluating results, and helps the system become smarter and more accurate and help businesses operate more efficiently. This helps companies to use AI to attract customers with highly personalized marketing programs, and to properly impact each target customer group.

However, the main research and application in Vietnam today is the information technology engineers, and little known and applied from those who research and do business, because AI requires high the technical level. In order to make it easier for readers to acquire knowledge and AI applications in business, this article will cover the use of classification algorithms in Supervised Machine Learning to accurately identify customers. The article uses Python programming language and programming libraries: Numpy, Pandas, Matplotlib, Scikit-learn to handle an actual Marketing data set, which helps readers follow and apply to solve syour research and business issues.

2. THEORETICAL BASIS AND THE DATASET

2.1. Theoretical basis

Machine learning is a computer science discipline in which algorithms or algorithms are designed to learn from collected data. It often aims to predict the results from input data that cannot be seen (the amount of data is so large that users often cannot see it). (Shai Shalev-Shwartz & Shai Ben-David, 2014)

Supervised Machine Learning is a technique of machine learning to build a function from training data. Training data consists of pairs of input objects (Vector form), and desired outputs. The output of a function can be a continuous value (called regression), or it is possible to predict a classification label for an input object (called classification). (Shai Shalev-Shwartz & Shai Ben-David, 2014)

The content of the article focuses on research and application of algorithms to classify applications in economics, including:

Algorithms	Content
Logistic regression	Is a linear regression extension used for classification tasks. The output variable is binary, not continuous.
Decision tree	The model of classification or regression can be deeply understood, dividing data attribute values into branches at decision nodes until a final decision is made.

Naive Bayes	The Bayes method is a classification method using the Bayes theorem. The theorem updates previous knowledge of an event with an independent probability of each feature that may affect the event.
Support vector machine	Support Vector Machine, or SVM, is often used for classification tasks. SVM algorithm finds a super-flat dividing optimal layer. It is best used with a non-linear solver.
Random forest	The algorithm is built on the decision tree to improve drastically accuracy. Random forests create many simple decision trees and use the 'majority vote' method to decide which labels will return. For the classification task, the final prediction will be the one with the most votes; while for regression tasks, the average prediction of all trees is the final prediction.
AdaBoost	Techniques of classification or regression use a multitude of models to make decisions but consider them based on their accuracy in predicting results.
Gradient-boosting trees	Gradient-boosting trees is a modern classification / regression technique. It is focusing on the bugs of previous trees and trying to fix it.

Nguồn: https://dnmtechs.com/vi/ung-dung-cua-thuat-toan-may-hoc-machine-learning/

2.2. The Dataset

The article uses the Bank Marketing data set from: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing#. The data set is related to direct marketing campaigns (phone calls) of a Portuguese banking organization. The classification goal is to predict: if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

 # bank client data:

1 - age (numeric)

2 - job: type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')

3 - marital: marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high school', 'illiterate', 'professional course', 'university degree', 'unknown')

5 - default: has credit in default? (categorical: 'no', 'yes', 'unknown')

6 - housing: has housing loan? (categorical: 'no', 'yes', 'unknown')

7 - loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

 # related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular', 'telephone')

9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10 - day_of_week: last contact day of the week (categorical: 'mon', 'tue', 'wed', 'thu', 'fri')

11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

 # other attributes:

12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means that client was not previously contacted)

14 - previous: number of contacts performed before this campaign and for this client (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')

 # social and economic context attributes

16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)

17 - cons.price.idx: consumer price index - monthly indicator (numeric)

18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)

19 - euribor3m: euribor 3 months rate - daily indicator (numeric)

20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

21 – y: has the client subscribed a term deposit? (binary: 'yes', 'no')

3. USE CLASSIFICATION ALGORITHMS IN SUPERVISED MACHINE LEARNING

Pre-installed Python programming language and programming libraries: Numpy, Pandas, Matplotlib, Scikit-learn and Jupyter Notebook to be able to perform customer segmentation steps.

1. Import Libraries and The Dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv("bank-additional-full.csv", sep = ';')
data.head()
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

5 rows × 21 columns

In [3]:
```
#Converting biến phụ thuộc y từ categorical sang dummy
y = pd.get_dummies(data['y'], columns = ['y'], drop_first = True)
y_dummy = y
```

2. Pre-processing Data

2.1. Processing data related to customers

In [4]:
```
# Lấy trường thông tin khách hàng
client = data.iloc[: , 0:7]
client.head()
```

Out[4]:

	age	job	marital	education	default	housing	loan
0	56	housemaid	married	basic.4y	no	no	no
1	57	services	married	high.school	unknown	no	no
2	37	services	married	high.school	no	yes	no
3	40	admin.	married	basic.6y	no	no	no
4	56	services	married	high.school	no	no	yes

In [5]:
```
# Label encoding
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
client['job']      = labelencoder_X.fit_transform(client['job'])
client['marital']  = labelencoder_X.fit_transform(client['marital'])
client['education']= labelencoder_X.fit_transform(client['education'])
client['default']  = labelencoder_X.fit_transform(client['default'])
client['housing']  = labelencoder_X.fit_transform(client['housing'])
client['loan']     = labelencoder_X.fit_transform(client['loan'])
```

```
In [6]: # function to creat group of ages
        def age(dataframe):
            dataframe.loc[dataframe['age'] <= 30, 'age'] = 1
            dataframe.loc[(dataframe['age'] > 30) & (dataframe['age'] <= 45), 'age'] = 2
            dataframe.loc[(dataframe['age'] > 45) & (dataframe['age'] <= 60), 'age'] = 3
            dataframe.loc[(dataframe['age'] > 60) & (dataframe['age'] <= 100),'age'] = 4
            return dataframe

In [7]: age(client);

In [8]: client.head()
```

Out[8]:

	age	job	marital	education	default	housing	loan
0	3	3	1	0	0	0	0
1	3	7	1	3	1	0	0
2	2	7	1	3	0	2	0
3	2	0	1	1	0	0	0
4	3	7	1	3	0	0	2

2.2. Processing data related to Marketing campaigns

```
In [9]: # Data of Marketing campaigns
        marketing = data.iloc[: , 7:11]
        marketing.head()
```

Out[9]:

	contact	month	day_of_week	duration
0	telephone	may	mon	261
1	telephone	may	mon	149
2	telephone	may	mon	226
3	telephone	may	mon	151
4	telephone	may	mon	307

```
In [10]: # Label encoding
         marketing['contact']     = labelencoder_X.fit_transform(marketing['contact'])
         marketing['month']       = labelencoder_X.fit_transform(marketing['month'])
         marketing['day_of_week'] = labelencoder_X.fit_transform(marketing['day_of_week'])

In [11]: def duration(data):
             data.loc[data['duration'] <= 90, 'duration'] = 1
             data.loc[(data['duration'] > 90)  & (data['duration'] <= 180), 'duration']   = 2
             data.loc[(data['duration'] > 180) & (data['duration'] <= 300), 'duration']   = 3
             data.loc[(data['duration'] > 300) & (data['duration'] <= 480), 'duration']   = 4
             data.loc[data['duration']  > 480, 'duration'] = 5
             return data

In [12]: duration(marketing);
         marketing.head()
```

Out[12]:

	contact	month	day_of_week	duration
0	1	6	1	3
1	1	6	1	2
2	1	6	1	3
3	1	6	1	2
4	1	6	1	4

2.3. Processing data related to socio-economic

```
In [13]: se = data.loc[: , ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']]
         se.head()
```

Out[13]:

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
0	1.1	93.994	-36.4	4.857	5191.0
1	1.1	93.994	-36.4	4.857	5191.0
2	1.1	93.994	-36.4	4.857	5191.0
3	1.1	93.994	-36.4	4.857	5191.0
4	1.1	93.994	-36.4	4.857	5191.0

2.4. Processing other data fields

```
In [14]: others = data.loc[: , ['campaign', 'pdays','previous', 'poutcome']]
         others.head()
```

Out[14]:

	campaign	pdays	previous	poutcome
0	1	999	0	nonexistent
1	1	999	0	nonexistent
2	1	999	0	nonexistent
3	1	999	0	nonexistent
4	1	999	0	nonexistent

```
In [15]: # Label encoding
         others['poutcome']     = labelencoder_X.fit_transform(others['poutcome'])
```

```
In [16]: others.head()
```

Out[16]:

	campaign	pdays	previous	poutcome
0	1	999	0	1
1	1	999	0	1
2	1	999	0	1
3	1	999	0	1
4	1	999	0	1

```
In [17]: # Merge data after processing
         data_bank = pd.concat([client, marketing, se, others], axis = 1)
         data_bank = data_bank[['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
                                'contact', 'month', 'day_of_week', 'duration', 'emp.var.rate', 'cons.price.idx',
                                'cons.conf.idx', 'euribor3m', 'nr.employed', 'campaign', 'pdays', 'previous', 'poutcome']]
         data_bank.shape

Out[17]: (41188, 20)
```

2.5. Balancing data

```
In [18]: # Divide data into 2 parts according to Marketing results
         df = pd.concat([data_bank, y], axis = 1)
         df_majority = df[df['yes'] == 0]
         df_minority = df[df['yes'] == 1]
```

```
In [19]: df_majority.shape, df_minority.shape

Out[19]: ((36548, 21), (4640, 21))
```

```
In [20]: # Balancing
         from sklearn.utils import resample
         df_minority_upsampled = resample(df_minority,
                                          replace=True,
                                          n_samples= int(4640*5),
                                          random_state=142)
         df_minority_upsampled.shape

Out[20]: (23200, 21)
```

```
In [21]: df_final = pd.concat([df_majority, df_minority_upsampled])
         y_final = df_final['yes']
         df_final.head()
```

Out[21]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	campaign	pdays	previous	poutcome	yes
0	3	3	1	0	0	0	0	1	6	1	...	1.1	93.994	-36.4	4.857	5191.0	1	999	0	1	0
1	3	7	1	3	1	0	0	1	6	1	...	1.1	93.994	-36.4	4.857	5191.0	1	999	0	1	0
2	2	7	1	3	0	2	0	1	6	1	...	1.1	93.994	-36.4	4.857	5191.0	1	999	0	1	0
3	2	0	1	1	0	0	0	1	6	1	...	1.1	93.994	-36.4	4.857	5191.0	1	999	0	1	0
4	3	7	1	3	0	0	2	1	6	1	...	1.1	93.994	-36.4	4.857	5191.0	1	999	0	1	0

2.6 Splitting the dataset

In [22]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
```

In [23]:
```
X_train, X_test, y_train, y_test = train_test_split(df_final.drop('yes', axis = 1), y_final, test_size = 0.2, random_state = 123)
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

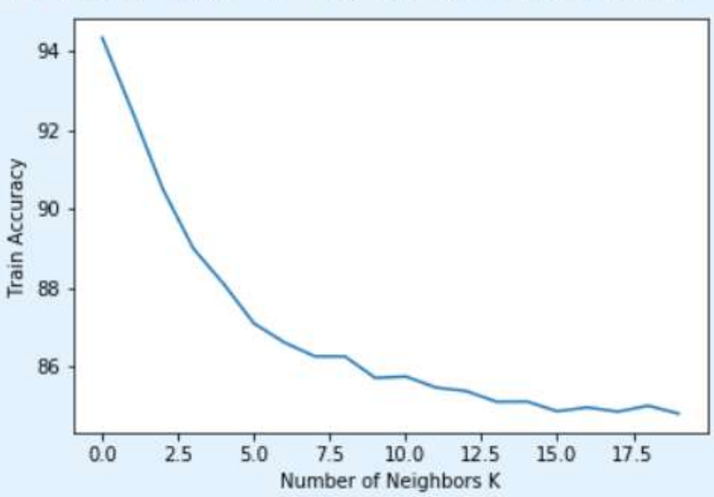
2.7. Standardize the scaler

In [24]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

3.1. KNN classifier

[26]:
```
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier

#Neighbors
neighbors = np.arange(0,20)

#Create empty list that will hold cv scores
cv_scores = []

#Perform 10-fold cross validation on training set for odd values of k:
for k in neighbors:
    k_value = k+1
    knn = KNeighborsClassifier(n_neighbors = k_value, weights='uniform', p=2, metric='euclidean')
    kfold = model_selection.KFold(n_splits=10, random_state=123)
    scores = model_selection.cross_val_score(knn, X_train, y_train, cv=kfold, scoring='accuracy')
    cv_scores.append(scores.mean()*100)
    print("k=%d %f" % (k_value, scores.mean()*100))

optimal_k = neighbors[cv_scores.index(max(cv_scores))]
print ("The optimal number of neighbors is %d with %f" % (optimal_k+1, cv_scores[optimal_k]))

plt.plot(neighbors, cv_scores)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Train Accuracy')
plt.show()
```

```
The optimal number of neighbors is 1 with 94.324040
```



```
# Chọn n_neighbors=1 do có Train Accuracy lớn nhất
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
knnpred = knn.predict(X_test)
print(accuracy_score(y_test, knnpred)*100)
```

```
94.82008368200837
```

3.2. Logistic regression

[28]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
logpred = logmodel.predict(X_test)

print(accuracy_score(y_test, logpred)*100)
```

```
85.2050209205021
```

3.3. Linear classifiers: Support Vector Machines

[29]:
```
# 1 kernel: sigmoid
from sklearn.svm import SVC
for this_gamma in [.01, 1.0, 10.0]:
    svc= SVC(kernel = 'sigmoid', gamma= this_gamma)
    svc.fit(X_train, y_train)
    svcpred = svc.predict(X_test)
    print(accuracy_score(y_test, svcpred)*100)
```

```
82.46861924686193
68.418410041841
69.81589958158996
```

[30]:
```
# 2 kernel: Radial Basis Function
from sklearn.svm import SVC
for this_gamma in [.01, .03, .06, .09, .1, 1.0, 10.0]:
    svc= SVC(kernel = 'rbf', gamma= this_gamma)
    svc.fit(X_train, y_train)
    svcpred = svc.predict(X_test)
    print(this_gamma, accuracy_score(y_test, svcpred)*100)
```

```
0.01 86.61924686192468
0.03 87.6234309623431
0.06 88.4602510460251
0.09 89.05439330543933
0.1 89.19665271966527
1.0 96.74476987447699
10.0 98.97071129707113
```

[31]:
```
# Gamma = 10
svc_rbf_10= SVC(kernel = 'rbf', gamma= 10)
svc_rbf_10.fit(X_train, y_train)
svcpred_rbf_10 = svc_rbf_10.predict(X_test)
print(accuracy_score(y_test, svcpred_rbf_10)*100)
print('SVC Confusion Matrix\n', confusion_matrix(y_test, svcpred_rbf_10))
```

```
10.0 98.97071129707113
SVC Confusion Matrix
 [[7278   46]
 [   77 4549]]
```

```
[32]: # 3 kernel: Linear
      from sklearn.svm import SVC
      svc_l= SVC(kernel = 'linear', gamma= 1)
      svc_l.fit(X_train, y_train)
      svcpred_l = svc_l.predict(X_test)
      print(accuracy_score(y_test, svcpred_l)*100)

      85.5397489539749
```

3.4. Decision Tree Classifier

```
[33]: from sklearn.tree import DecisionTreeClassifier
      dtree = DecisionTreeClassifier(criterion='gini') #criterion = entopy, gini
      dtree.fit(X_train, y_train)
      dtreepred = dtree.predict(X_test)
      print(accuracy_score(y_test, dtreepred)*100)

      95.10460251046025
```

3.5. Random Forest Classifier

```
[34]: # n=200
      from sklearn.ensemble import RandomForestClassifier
      rfc = RandomForestClassifier(n_estimators = 200)#criterion = entopy,gini
      rfc.fit(X_train, y_train)
      rfcpred = rfc.predict(X_test)
      print(accuracy_score(y_test, rfcpred)*100)

      95.38912133891213
```

```
[35]: # n=1000
      from sklearn.ensemble import RandomForestClassifier
      rfc = RandomForestClassifier(n_estimators = 1000)#criterion = entopy,gini
      rfc.fit(X_train, y_train)
      rfcpred = rfc.predict(X_test)
      print(accuracy_score(y_test, rfcpred)*100)

      95.33891213389121
```

3.6. Naive bayes classifier

```
[36]: from sklearn.naive_bayes import GaussianNB
      gaussiannb= GaussianNB()
      gaussiannb.fit(X_train, y_train)
      gaussiannbpred = gaussiannb.predict(X_test)
      probs = gaussiannb.predict(X_test)
      print(accuracy_score(y_test, gaussiannbpred)*100)

      76.41004184100419
```

3.7. Gradient boosting classifier

```
[37]: from sklearn.ensemble import GradientBoostingClassifier
      gbc = GradientBoostingClassifier()
      gbc.fit(X_train, y_train)
      gbcpred = gbc.predict(X_test)
      print(accuracy_score(y_test, gbcpred)*100)

      88.0
```

3.8. XGBoost Classifier

```
[38]: from xgboost import XGBClassifier
      xgb = XGBClassifier()
      xgb.fit(X_train, y_train)
      xgbprd = xgb.predict(X_test)
      print(accuracy_score(y_test, xgbprd)*100)

      88.00836820083681
```

4. DISCUSSION

Using Test data set for SVC algorithm with gamma = 10, accuracy_score is the largest (98.97) mean accurately forecast results in classification 98.97% identified customer.

The Classification report shows that: The exact customer forecast rate is 99%, and do not customer ≥ 98%.

```
from sklearn.metrics import classification_report
print('RFC Reports\n',classification_report(y_test, svcpred_rbf_10))
print('RFC Confusion Matrix\n', confusion_matrix(y_test, rfcpred))

RFC Reports
              precision    recall  f1-score   support

          0       0.99      0.99      0.99      7324
          1       0.99      0.98      0.99      4626

avg / total       0.99      0.99      0.99     11950

RFC Confusion Matrix
 [[6796  528]
 [  29 4597]]
```

The classification results have two types of errors:

Harmful errors, which means that the customer DOES NOT SUBSCRIBE but the classification results suggest that this customer has done. This type of errors makes losing customer of the bank. The classification results show that wrong 29/6825 cases reached 99.575% accuracy.

Harmless errors, which means that customers SUBSCRIBE, but classification results suggest that this customer has not done. Although wrongly aware, this error does not lose bank customers. The classification results show that the wrong 528/5056 cases reached 89.56% accuracy but this mistake did not harm the bank.

5. CONCLUSION

The industrial revolution 4.0 is going strong and changing every aspect of socio-economic life. Any company that knows how to apply advanced technology will gain huge advantages in the market. With the outstanding advantages of processing and analyzing large data, artificial intelligence brings great advantages for businesses to apply in their business operations, helping businesses minimize risk, increase competitiveness. The article introduced an intuitive way of classification algorithms in Supervised Machine Learning through actual data set to accurately identify customers. Thereby, readers can easily follow and apply to handle specific problems in their research or actual business.

REFERENCES

1. Alex Smola & S.V.N. Vishwanathan, 2008. *INTRODUCTION TO MACHINE LEARNING.* Cambridge University Press.
2. Edouard Duchesnayt & Tommy Löfsted, 2018. *Statistics and Machine Learning in Python - Release 0.2.* ftp://ftp.cea.fr.
3. Farhan Zaidi, 2018. *Hands-on Scikit-learn for Machine Learning [Video]*. Packt
4. Guido van Rossum & et al., 2018. *The Python Language Reference - Release 3.7.1.* Python Software Foundation.
5. NumPy community, 2018. *NumPy Reference - Release 1.15.4.*SciPy.org.
6. Shai Shalev-Shwartz & Shai Ben-David, 2014. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press.
7. Scikit-learn developers, 2018. *scikit-learn user guide - Release 0.20.1.* scikit-learn.org.
8. Wes McKinney & et al., 2018. *Pandas: powerful Python data analysis toolkit - Release 0.23.4.* PyData Development Team.