AN ANALYSIS OF COLLEGE PLACEMENT FOR STUDENTS

Yuvaraj Bojjinettam
Faculty of Engineering Environment and Computing
Coventry University
Coventry, England
bojjinetty@coventry.ac.uk

Yuvaraj Bojjinettam
Faculty of Engineering Environment and Computing
Coventry University
Coventry, England
bojjinetty@coventry.ac.uk

CHAPTER I

ABSTRACT

The Objective is to apply machine learning algorithms to analyze a dataset of college placements. A variety of factor are considering when determine whether or not a student will be placed, Including age, gender, education background, internship, CGPA, hostel living and previous backlog history. The implementation includes data preprocessing, model training, evaluation and interpretation.

Key words:

Machine learning ,college placement, Dataset analysis, data preprocessing, model training, evaluation

INTRODUCTION

Importing the required libraries, including Pandas, Matplotlib, Seaborn, and Plotly Express, is the first step in the code. Using Pandas, it loads a dataset ('collegePlace.csv') that is likely made up of data about college students, their placements, and associated variables. Preprocessing, Data Exploration, and Fundamentals Ideas.

The head of the dataset is shown, along with the column names, duplicate entries, and missing values. Seaborn and Matplotlib visualizations shed light on the distribution of students' age, gender, department, placement status, and internships. The distribution of pupils who were placed and those who were not is displayed using a pie chart.

The document conceptual structure is presented below:

1) The literature study is described in chapter II.
2) The dataset description and problem explain in chapter III.
3) Machine learning methods and techniques covers in chapter IV.
4) The experiment setup using a dataset covers in chapter V.
5) Results and Conclusions are described in chapter VI.
6) Appendix I, Appendix II are covered in chapter VII

CHAPTER II

LITERATURE REVIEW

An analysis of a dataset related to student placement in colleges. An explicit literature review, the context suggests that this analysis could benefit from existing research in the field of student placement prediction and machine learning.

Factors Affecting Student Placement:

Summarize studies examining various factors influencing student placement, such as:

- Academic performance (CGPA, history of backlogs)
- Internship experiences
- Demographic factors (age, gender)
- Field of study (stream)
- Other relevant factors identified in the literature.
- Discuss findings regarding the impact of these factors on placement outcomes.

Predictive Modeling for Student Placement:

- Review research on predictive modeling techniques used to forecast student placement.
- Describe machine learning algorithms commonly employed in predictive modeling for student outcomes.
- the strengths and limitations of different modeling approaches in predicting placement success.

Strategies for Improving Placement Rates:

Explore strategies and interventions aimed at improving student placement rates, such as:

- Career counseling and guidance programs
- Skill development initiatives
- Internship and experiential learning opportunities
- Curriculum enhancements to align with industry demands
- Evaluate the effectiveness of these strategies based on empirical evidence from the literature.

Challenges and research Directions:

- Highlight challenges and gaps identified in existing research on student placement.
- Suggest potential avenues for future research to address these gaps and enhance our understanding of placement dynamics.
- Emerging trends or methodologies that may shape future research in this area.

CHAPTER III

Problem Statement

A college wants to investigate the characteristics that influence student placement in companies after graduation. They collected data on a variety of student characteristics, including age, gender, academic performance, internship experiences, department stream, hostel accommodation, and history of backlogs. The college's goal is to discover which elements contribute the most to a student's placement success and to develop a predictive model to identify students who are more likely to be placed after graduation.

DATASET DESCRIPTION

The dataset contains information about college students, including their placement status and other variables. Here's an explanation of the columns:

- Age of the students.
- Gender: The student's gender (male or female).
- Stream: The student's departmental stream of study (e.g. Computer Science, Mechanical, or Electrical).
- Internships: The number of internships the student has completed.
- CGPA: The student's cumulative grade point average.
- Hostel: Indicates whether or not the student stayed at a hostel.
- HistoryOfBacklogs: Indicates whether the student has a history of academic backlogs (0 = No; 1 = Yes).
- PlacedOrNot: The student's placement status (0 = not placed, 1 = placed).

Observation

- The dataset contains 2966 rows with 8 columns .
- There are no missing values in the dataset.
- The dataset contains features like gender and stream are the categorical variables.

CHAPTER IV

METHODS

In the classification process, the dataset is utilized to train machine learning models to predict whether a student will be placed or not based on various features such as age, gender, academic stream, internships, CGPA, hostel status, and history of backlogs. Here's how the dataset is utilized in the classification process.

Data Loading and Exploration

Data exploration techniques such as checking the first few rows, shape, data types, presence of duplicates and missing values, summary statistics, and unique value counts are performed to understand the structure and content of the dataset.

Dataset: Original dataset

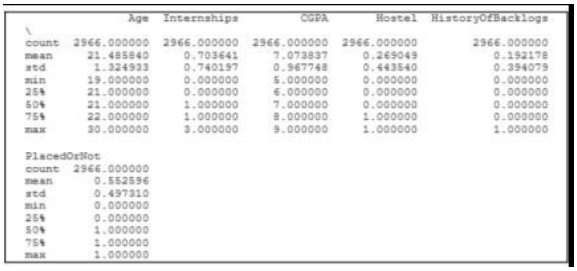
	Age	Gender	Stream	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot
0	22	Male	Electronics And Communication	1	8	1	1	1
1	21	Female	Computer Science	0	7	1	1	1
2	22	Female	Information Technology	1	6	0	0	1
3	21	Male	Information Technology	0	8	0	1	1
4	22	Male	Mechanical	0	8	1	0	1
...	...	...	...	...	...	...	...	...
2961	23	Male	Information Technology	0	7	0	0	0
2962	23	Male	Mechanical	1	7	1	0	0
2963	22	Male	Information Technology	1	7	0	0	0
2964	22	Male	Computer Science	1	7	0	0	0
2965	23	Male	Civil	0	8	0	0	0

Shape: [2966 rows x 8 columns]

Figure 1: *Original Dataset*

dataset.info(), dataset.describe() , like count, mean, std, min, mid, max :

bojjinetty@coventry.ac.uk



	Age	Internships	CGPA	Hostel	HistoryOfBacklogs
count	2966.000000	2966.000000	2966.000000	2966.000000	2966.000000
mean	21.485840	0.703641	7.073837	0.269049	0.192178
std	1.324933	0.740197	0.967748	0.443540	0.394079
min	19.000000	0.000000	5.000000	0.000000	0.000000
25%	21.000000	0.000000	6.000000	0.000000	0.000000
50%	21.000000	1.000000	7.000000	0.000000	0.000000
75%	22.000000	1.000000	8.000000	1.000000	0.000000
max	30.000000	3.000000	9.000000	1.000000	1.000000

PlacedOrNot	
count	2966.000000
mean	0.552596
std	0.497310
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

Figure 2: *Describe of dataset*

***Unique Values*:**
Number of unique values in each column and row from the dataset into table and dtype:int64

Age	11
Gender	2
Stream	6
Internships	4
CGPA	5
Hostel	2
HistoryOfBacklogs	2
PlacedOrNot	2

Table 1: *Number of unique values in each column and row*

Age.Unique():

Number of Students by Age in the form of Ascending Order

Age	Students
21	1084
22	941
20	375
23	195
19	156
24	131
26	50
25	29
28	3
30	1
29	1

Table 2: Number for students by Age

Gender.Count():
number students by gender

Gender	Students
Male	2475
Female	491

PlacedOrNotPlaced.unique():
Value Count() ,Number of students placed or Not placed.

PlacedOrNotPlaced	Number of Students
1	1639
0	1327

Stream :
Number students in each department

Stream	Students
Computer Science	776
Information Technology	691
Electronics And Communication	424
Mechanical	424
Electrical	334
Civil	317

Internship:
How many students were placed and how many internships they were awarded.

Number Of Internship	Number of students
0	1331
1	1234
2	350
3	51

Name: Internships, dtype: int64

Data Preprocessing:

- Preprocessing steps are performed to prepare the data for model training.
- Categorical variables such as 'Gender' and 'Stream' are encoded using techniques like one-hot encoding or label encoding to convert them into numerical format, which machine learning algorithms can understand.
- Features are scaled using techniques like Min-Max scaling or Standard scaling to bring them to a similar scale, preventing features with larger magnitudes from dominating the model training process.

One-Hot Encoding:

- Utilized the **pd.get_dummies()** function to perform one-hot encoding on the categorical variables in the datasdet.
- Set **drop_first=True** to avoid multicollinearity by dropping the first category for each encoded variable.
- The resulting encoded dataset (**data_encoded**) contains binary columns for each category of the original categorical variables.

Feature Selection and Target Variable:

- Selected relevant variables (**'Age', 'Gender', 'Stream', 'Internships', 'CGPA', 'Hostel', 'HistoryOfBacklogs'**) as features (X_columns).
- Defined the target variable (**'PlacedOrNot'**) as the label (y_column).

```
X_columns = ['Age', 'Gender', 'Stream', 'Internships', 'CGPA', 'Hostel', 'HistoryOfBacklogs']
y_column = 'PlacedOrNot')
 X = dataset[X_columns]
y = dataset[y_column]
```


bojjinetty@coventry.ac.uk

Splitting the Dataset:

- Split the dataset into training and testing sets using the **train_test_split** function from **sklearn.model_selection**.
- Used a test size of 20%.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Scaling Features:

- Scaled the features using **MinMaxScaler** from **sklearn.preprocessing**.
- This step ensures that all features have the same scale, which can improve the performance of some machine learning algorithms.

```
scaler = preprocessing.MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

Label Encoding:

- Used **LabelEncoder** from the **sklearn.preprocessing** to encode categorical variables directly in the original dataset (**dataset**).
- Encoded the 'Gender' and 'Stream' columns.

```
le = LabelEncoder()
dataset['Gender'] = le.fit_transform(dataset['Gender'])
dataset['Stream'] = le.fit_transform(dataset['Stream'])
```

Standardizing Features:

- Standardized features using **StandardScaler** or **MinMaxScaler**.
- Standardization or normalization ensures that all features have a mean of 0 and standard deviation of 1 or are scaled between 0 and 1.

```
# Standardize features
scaler = StandardScaler()
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
```

These preprocessing steps are essential to prepare the dataset for machine learning models, ensuring that the data is in a suitable format and scale for accurate model training and evaluation.

Model training and evaluation

Classification Methods Explanation:

In this section, various classification methods were utilized to train machine learning models on the dataset. Here's an explanation of each method and the process involved:

1. Support Vector Classifier (SVC):

• For classification purposes, SVC creates a hyperplane or group of hyperplanes in a high-dimensional space.

• Utilized the SVC class from sklearn.svm.

• Evaluated using cross-validation accuracy.

2. Decision Tree:

• Decision Tree recursively partitions the feature space into disjoint regions based on the value of the features.

• Employed the DecisionTreeClassifier class from sklearn.tree.

• Evaluated using cross-validation accuracy.

• Trained a decision tree model separately to obtain training and testing accuracies.

3. Logistic Regression:

• Using a logistic function, logistic regression models predict the probability of a binary outcome..

• Utilized the LogisticRegression class from sklearn.linear_model.

• Evaluated using cross-validation accuracy.

4. Random Forest:

• During training, Random Forest creates a large number of decision trees and outputs the class that is the mean of the classes of the individual trees.

Ensemble learning is the term for this method.

• Employed the RandomForestClassifier class from sklearn.ensemble.

• Specified the number of trees (n_estimators=50).

- Evaluated using cross-validation accuracy.

5. Gradient Boosting:

- Gradient Boosting builds an additive model in a forward stage-wise manner, with each stage comprising the addition of a weak learner to improve the performance of the model.

- Utilized the GradientBoostingClassifier class from sklearn.ensemble.

- Evaluated using cross-validation accuracy.

6. Naive Bayes:

- Naive Bayes is a probabilistic classifier based on Bayes' theorem with strong (naive) independence assumptions between the features.

- Employed the GaussianNB class from sklearn.naive_bayes.

- Evaluated using cross-validation accuracy.

7. K-Nearest Neighbors (KNN):

- KNN classifies objects based on the majority vote of their neighbors, assigning a class label to an object by searching through the training dataset for the k-nearest neighbors.

- Utilized the KNeighborsClassifier class from sklearn.neighbors.

- Evaluated using cross-validation accuracy.

Cross-Validation

Cross-validation is a statistical technique used to evaluate the performance of a machine learning model by splitting the dataset into multiple subsets, using a subset of the data to train the model and the remaining data to evaluate it.. This process is repeated multiple times, and the results are averaged to provide a more accurate assessment of the model's performance. Cross-validation helps to mitigate the risk of overfitting and provides a more reliable estimate of how the model will perform on unseen data.

```
for name, model in models.items():
    scores = cross_val_score(model, X, y, cv=3)
    print(f"{name} - Cross-validation Accuracy:
{np.mean(scores)}")
```

Many classification models, including SVM, Decision Tree, Logistic Regression, Random Forest, Gradient Boosting, Naive Bayes, and KNN, are defined in a dictionary named models.

Cross-validation is done using the cross_val_score function from sklearn.model_selection. The following specification are required:

Model: The model for categorization that will be assessed.

X: The matrix of features.

y: The variable to aim for.

cv: The cross-validation fold count (cv=3 in this example).

Output:

SVC - Cross-validation Accuracy: 0.831100

Decision Tree - Cross-validation Accuracy: **0.870208**

Logistic Regression - Cross-validation Accuracy: 0.756931

Random Forest - Cross-validation Accuracy: 0.861443

Gradient Boosting - Cross-validation Accuracy: 0.873579

Naive Bayes - Cross-validation Accuracy: 0.781537

KNN - Cross-validation Accuracy: 0.816941

Confusion Matrix

Evaluation metrics like confusion matrix, training accuracy, and testing accuracy.

Training Accuracy and Test Accuracy

Train the Decision Tree model

model = DecisionTreeClassifier()

model.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix

Predictions and evaluation

y_pred = model.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")

print(cm)

print("Training Accuracy:", model.score(X_train, y_train))

print("Testing Accuracy:", model.score(X_test, y_test))

Model Performance

Decision Tree and Gradient Boosting achieved the highest cross-validation accuracies of **87.0%** and 87.4% respectively. Logistic Regression had the lowest accuracy at **75.7%.**

Confusion Matrix (Decision Tree Model):

[[242 19]

[53 280]]

True Negatives (TN): 242

False Positives (FP): 19

False Negatives (FN): 53

True Positives (TP): 280

Training and Testing Accuracy (Decision Tree Model): Training Accuracy: 92.8% Testing Accuracy: 87.9%

Data Visualization

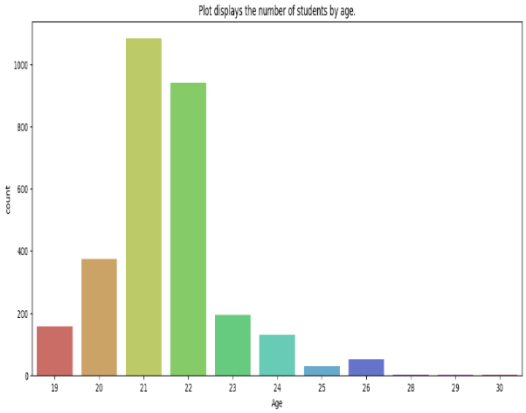


Figure 3: Plot shows Number of students in each age group

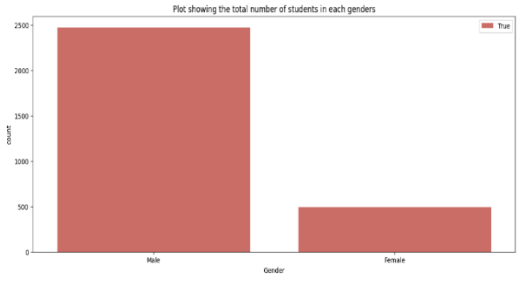


Figure 4: Plot shows Number of students each gender

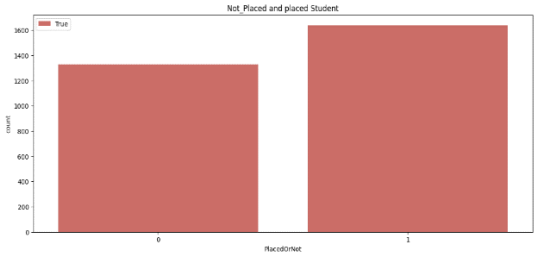


Figure 5: Plot shows Number of students placed Or Not-placed

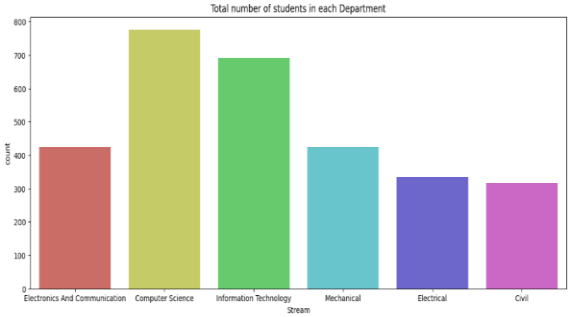


Figure 6: Plot shows Number of students in each Department

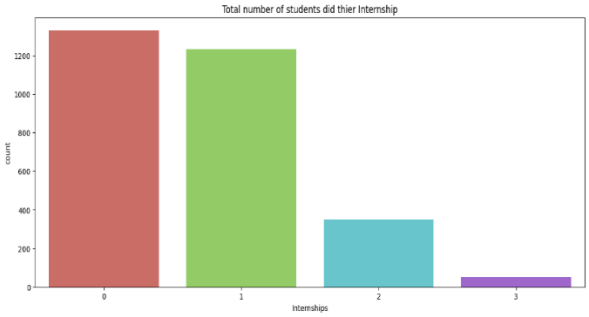


Figure 7: Plot shows Number of students got number of Internship.

Model Comparison

Model comparison based on cross validation accuracies of different algorithms

bojjinetty@coventry.ac.uk

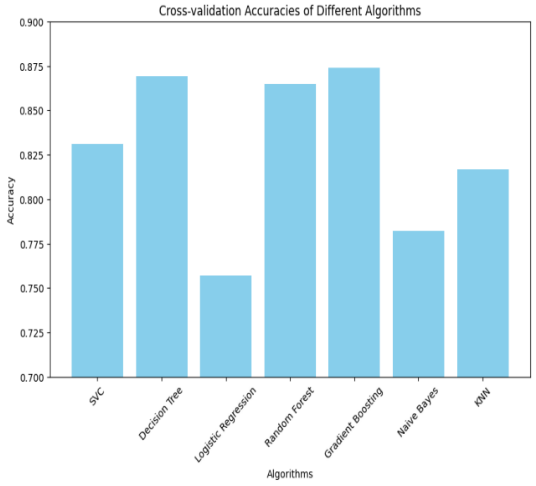


Figure 8: **Model Comparison**

Heated Map :

Heated Map for Confusion matrix of predicted label and true label

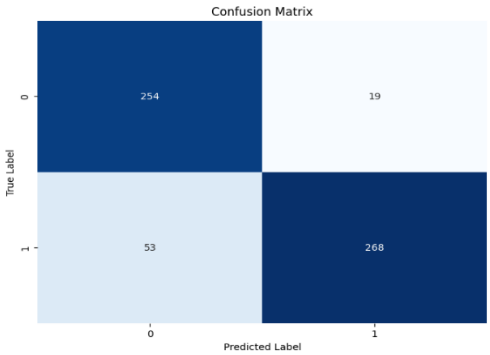


Figure 9: Heated Map

CHAPTER V

RESULT

SVC - Cross-validation Accuracy: 0.831100608720214

Decision Tree - Cross-validation Accuracy: 0.8661617195356751

Logistic Regression - Cross-validation Accuracy: 0.7569311686309185

Random Forest - Cross-validation Accuracy: 0.8651505971898713

Gradient Boosting - Cross-validation Accuracy: 0.8735793458133941

Naive Bayes - Cross-validation Accuracy: 0.7815371925185133

KNN - Cross-validation Accuracy: 0.8169411434006187

Confusion Matrix:

[[243 26]

[52 273]]

Training Accuracy: 0.9270657672849916

Testing Accuracy: 0.8686868686868687

Explanation

1. Support Vector Classifier (SVC): This model achieved a cross-validation accuracy of around 83.11%. In simple words, it correctly predicted whether a student would be placed or not about 83% of the time on unseen data.
2. Decision Tree: With a cross-validation accuracy of approximately 86.62%, this model performed slightly better than SVC. It correctly predicted placement outcomes around 86% of the time on unseen data.
3. Logistic Regression: This model achieved a cross-validation accuracy of about 75.69%. While still decent, it performed slightly worse compared to the other models, predicting outcomes correctly about 76% of the time.
4. Random Forest: Like the Decision Tree, Random Forest achieved a cross-validation accuracy of around 86.52%. It performed consistently well, predicting outcomes accurately about 86% of the time.
5. Gradient Boosting: Among all models, Gradient Boosting performed the best with a cross-validation accuracy of approximately 87.36%. It accurately predicted outcomes around 87% of the time, making it the top-performing model in this analysis.
6. Naive Bayes: This model achieved a cross-validation accuracy of about 78.15%. It performed reasonably well but not as good as Gradient Boosting or Decision Tree.
7. K-Nearest Neighbors (KNN): KNN achieved a cross-validation accuracy of approximately 81.69%, which puts it in the middle of the pack in terms of performance.

Confusion Matrix: The confusion matrix provides the model's performance. In this case, the model correctly classified 243 instances of students being placed and 273 instances of students not being placed. However, it misclassified 52 instances of

placed students as not placed, and 26 instances of not placed students as placed.

Training and Testing Accuracy: The model achieved a training accuracy of around 92.71%, meaning it performed well on the data it was trained on. The testing accuracy, which measures how well the model generalizes to new, unseen data, was approximately 86.87%. This indicates that the model performs well in predicting whether a student will be placed or not.

CHAPTER VI

CONCLUSION

With a testing accuracy of about 86%, the results show that the Decision Tree model is successful in predicting college placements. Applications of machine learning, such college placement prediction, demonstrated the method's significance.

REFERENCE

1. Shmueli, G., Patel, N. R., & Bruce, P. C. (2016). Data Mining for Business Analytics: Concepts, Techniques, and Applications with XLMiner (3rd ed.). Hoboken, NJ: John Wiley & Sons, Inc.
2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R. New York, NY: Springer.
3. Raschka, S., & Mirjalili, V. (2019). Python Machine Learning (3rd ed.). Birmingham, UK: Packt Publishing.
4. Müller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. Sebastopol, CA: O'Reilly Media, Inc.
5. McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.). Sebastopol, CA: O'Reilly Media, Inc.
6. Grus, J. (2015). Data Science from Scratch: First Principles with Python. Sebastopol, CA: O'Reilly Media, Inc.
7. VanderPlas, J. (2016). Python Data Science Handbook: Essential Tools for Working with Data. Sebastopol, CA: O'Reilly Media, Inc.
8. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). New York, NY: Springer.
9. Zikopoulos, P., Eaton, C., Deutsch, T., Lapis, G., & Deroos, D. (2012). Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. New York, NY: McGraw-Hill Education.
10. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. Cambridge, MA: MIT Press.
11. Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. New York, NY: Springer.
12. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. New York, NY: Springer.
13. Hastie, T., Tibshirani, R., & Friedman, J. (2001). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York, NY: Springer.
14. Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques (4th ed.). San Francisco, CA: Morgan Kaufmann.
15. Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2015). Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. Cambridge, MA: MIT Press