2020/21

Student Name: Xuhui Gong

Student ID: 201447569

Project Title: COMP390
Data-Driven π-π Co-crystal Prediction Model with Positive and Unlabeled Data

Supervisor: Dr. Vitaliy Kurlin
Katerina Vriza

DEPARTMENT OF COMPUTER SCIENCE

The University of Liverpool, Liverpool L69 3BX

THE UNIVERSITY
of LIVERPOOL

COMP390 HONOURS YEAR PROJECT

Data-Driven π-π Co-crystal Prediction Model with Positive and Unlabeled Data

May 15, 2021

Table of Contents

Abstract

With the application of machine learning techniques in materials science increasing every year, many new materials are being discovered based on data-driven methods. The motivation for this study is to address the prediction challenges posed by the current imbalance of co-crystal data and the unreliability of negative data. For this problem, we aim to construct a binary classification model that can predict whether two different PAH molecules can be synthesised into new $\pi-\pi$ co-crystals. In this paper, we used a two-step PU learning algorithm based on only positive and unlabelled data, which can effectively circumvent the problems of data imbalance and unreliability. Comparing PU learning with the isolation forest algorithm based on one-class classification, the PU learning algorithm can obtain better performance, and it has a stronger ability to find potential positive samples.

Keywords: Machine Learning, Material Science, Co-crystal Prediction, Imbalanced Learning, Classification, Positive and Unlabeled Learning, Sampling Methods.

Acknowledgements

I would like to give heartful thanks to Katerina who provided helpful comments and suggestions. Special thanks also go to Dr. Vitaliy Kurlin, Dr. Olga Anosova, Dr. Ming Liu and Porf Simon Maskell whose opinions and information have helped me very much throughout the production of this study. I would also like to express my gratitude to my family for their moral support and warm encouragements.

Acronyms

CSD	Cambridge Structural Database
PU	Positive and Unlabeled Learning
RF	Random Forest
IF	Isolation Forest
PAHs	Polycyclic Aromatic Hydrocarbons
SMOTE	Synthetic Minority Over-sampling Technique
SCAR	Selected Completely at Random
DBSCAN	Density-based Spatial Clustering of Applications with Noise

List of Figures

List of Tables

Chapter 1

Introduction

Since the 1950s, computational science has been called the third paradigm of materials science, based on analytical methods such as thermodynamics and statistical mechanics. In the 21^{st} century, the introduction of machine learning has revolutionised the design of materials with the increase in computer computing power and the widespread interest in large amounts of data, making big data-driven science the fourth paradigm in materials science [1]. For example, the Cambridge Structural Database (CSD), the world's repository of small-molecule organic and metal-organic crystal structures, has been continuously updated with new data since its inception in 1965 [2], and the number of new materials discovered using data-driven methods has increased significantly in recent years.

1.1 Background

As a sub-field of materials science, co-crystals play an essential role in many fields, for example, in the pharmaceutical industry, where co-crystal can modulate the physical properties of an active pharmaceutical ingredient, including stability, solubility and dissolution [3, 4]. During a pandemic, the National Health Service in the UK recommends that COVID-19 self-isolated patients be given paracetamol as a priority when dealing with high temperatures. However, the

mechanical properties of paracetamol in pure form are impoverished, which means that it is not easy to make tablets of pure paracetamol in industrial. However, when co-crystalising with other molecules such as theophylline, it is possible to achieve an easily tabulated form [5].

In general, co-crystal is defined as a solid crystalline material consisting of two or more molecules in the same crystal lattice [6]. In this work, we focus on designing a specific type of co-crystals composed of two different polycyclic aromatic hydrocarbons (PAHs) connected with π-π stacking interactions. The PAHs as a common co-crystal molecular building block for generating π-π bonds composed of a π-electron conjugated system. As the simplest PAHs, the structure of naphthalene is shown in Figure 1.1, which shows that the π-orbital of naphthalene is very close to overlapping.

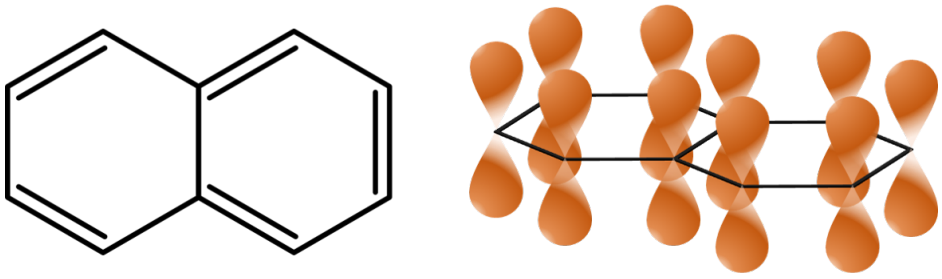


Figure 1.1: The two-dimensional structure of naphthalene, which is the simplest PAHs structure is shown on the left-hand side of the diagram. The right side shows that the ten π molecular orbitals of naphthalene are very close to each other and can overlap.

The overlapping of orbitals means that the electrons will be spread over a larger area, giving some stability to the whole system. Suppose there are multiple π-electron-rich molecules on a lattice, and some typical arrangement exists that facilitates π-π interactions between different molecules. In that case, the electronic properties are beneficial, which means that some co-crystals are structurally stable and have a higher chance of synthesising them in the laboratory. For example, as shown in Figure 1.2, the 1:1 combined co-crystal of Coran-

nulene and C60 [7], where a π-π interaction between Corannulene and C60 exists.

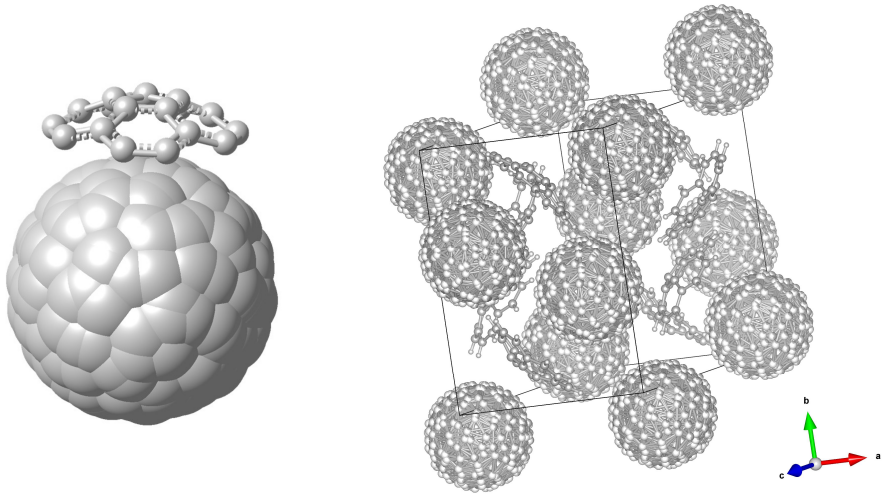


Figure 1.2: Co-crystal Structure of Corannulene and C60 with π-π Interactions.

1.2 Motivation

In the past, new materials were often discovered using a trial-and-error approach that was depended on personal experience. In general, co-crystal synthesis experiments are pretty time-consuming. Given the consumption of experimental development, traditional materials discovery methods could barely accommodate the large-scale demand for new materials [8]. Using machine learning models, we can effectively predict and discover the materials we need. Specifically, for the co-crystal prediction task, we use some combination of molecular characteristics as input to a machine learning model that can predict the co-crystals we need, for example, π-π co-crystals.

The motivation for this study was was to reduce the experimental screening time and effectively increase the success rate. We would like to propose a model that can have a high probability of predicting molecular pairs that could form stable crystal structures.

1.3 Definition of Objectives

As shown in Figure 1.3, the aim of this study is to design a data driven workflow such that, given two different molecules we could predict whether their combination can form a new co-crystal or not. Given the complexity of the co-crystallization problem, we are focusing on a smaller co-crystal subset which involves the interaction between a certain type of molecules, namely polyaromatic hydrocarbons (PAHs) forming $\pi - \pi$ co-crystals. These materials have recently gained both academic and industrial interest due to the unique electronic properties that might have [9–11].

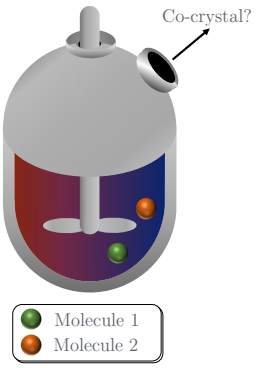


Figure 1.3: Objectives of Co-crystal Prediction

In contrast to the traditional trial-and-error approach, we hope to achieve the following objectives:

- Reducing the number of experiments
- Improving the success rate of of experimental screening
- Getting insights about the important factors that can drive to co-crystallization through the interpretation of our model

1.4 Current Challenges in Material Science

The ultimate goal of data-driven methods in materials science is to predict reliable materials. However, there are still many challenged to be faced:

(1) **The first challenge is the unbalanced nature of the data set**. In the past, failed experiments as negative data were often discarded, which led to a significant imbalance between positive and negative samples [12].

As shown in Figure 1.4, we have simulated the case of data imbalance, and the imbalance between the positive data C_1 and the negative data C_0 is 9:1. For the majority class C_1, we assume that C_1 follows a Gaussian distribution with variance 4 and mean 0; for the minority class C_0, it follows a one-dimensional Gaussian distribution with variance 1 and mean 1.5, where the distribution of data points is indicated in blue and yellow on the horizontal axis for the minority and majority classes. After accounting for the imbalance of data points, we use the solid line to show the probability density for each class and the dashed line to show the probability density after accounting for the proportion of different classes. According to Bayes rule, we can write the following inequality:

$$\mathbb{P}(C_1 \mid x) = \frac{\mathbb{P}(x \mid C_1)\mathbb{P}(C_1)}{\mathbb{P}(x)} > \frac{\mathbb{P}(x \mid C_0)\mathbb{P}(C_0)}{\mathbb{P}(x)} = \mathbb{P}(C_0 \mid x) \tag{1.1}$$

From Equation (1.1) it can be seen that in this case the probability of a C_1 occurrence given the Features x larger than the probability of a C_0 occurrence given the Features x. When the model predicts, data imbalance can lead to a situation where one category is always more likely than the others. For machine learning problems, a common underlying assumption is that the sample size of the data is balanced. Suppose the sample size of the dataset is extremely un-

balanced. In that case, this usually leads to the failure of conventional machine learning algorithms because most of algorithms assume a balanced class distribution [13].

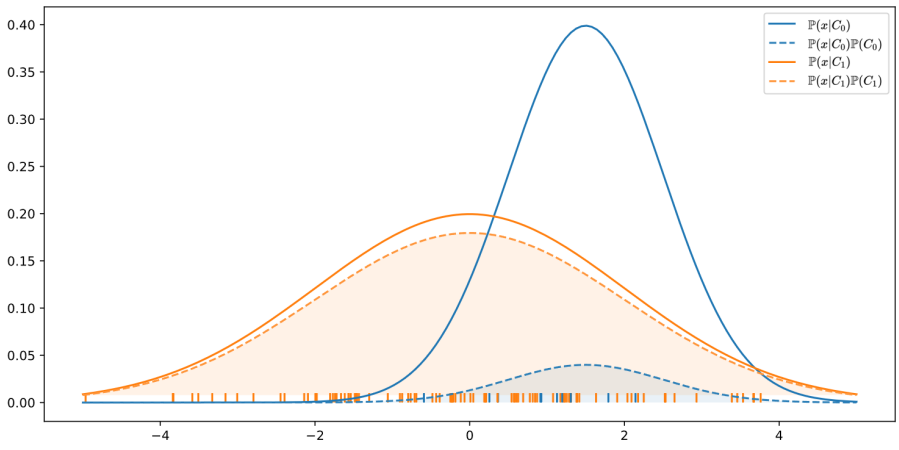


Figure 1.4: Example of data distribution for an unbalanced data set

In this research, we searched the Cambridge Structural Database (CSD) [2] for 1806 positive samples in the presence of π-π co-crystals, while negative data were scarce. Ultimately, the lab provided 11 negative samples with synthetic failure and a data imbalance of approximately 186:1.

(2) **Negative data are not always reliable**. Previously assigned negative samples may be mislabelled due to experimental conditions and limitations of the techniques used, leading to unreliable negative data selection and model training bias. In crystallisation experiments, the result of the experiment often depends on the experimenter's method of manipulation, the solubility of the material, the temperature of synthesis and some other conditions [14]. For example, in 2013, Luo et al. labelled pyrazine-2-carboxamide oxalic acid as negative. However, in 2016, Kulla et al. found that this combination (UZODUK) could form a new co-crystal [15–17].

1.5 Outline of Model Construction

In this section, we will reveal the basic process of co-crystal prediction. As shown in Figure 1.5, we first need to extract and clean our current data and select an appropriate machine learning algorithm for classification. In addition, we need to evaluate and hyperparameterize the algorithm to find the best model. Finally, we need to filter the predicted results experimentally. The results of the experiments will update our current database to form a closed loop to improve the accuracy of the model.

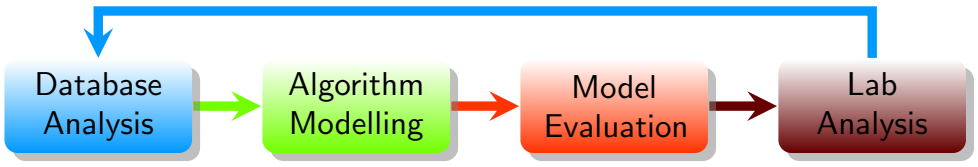


Figure 1.5: Basic Process of Co-crystal Prediction

1.5.1 Data Preparation

A prerequisite for a well-performing machine learning model is the quality of the dataset. As shown in Figure 1.6, we used the Cambridge Structural Database (Set *A*) [18], and we selected co-crystals in the CSD that had PAHs with π-π stacking as a positive dataset (Set *P*).

For the failed crystal synthesis data provided by the laboratory, we treated it as a negative sample (Set *N*).

In addition, unlabelled dataset needs to be generated. The purpose of generating unlabelled data is for a limited search of the chemical space of potential structures. ZINC 15 is a comprehensive chemical molecule database [19], in

this study, as shown in the flowchart in Figure 1.6, we generated unlabelled data (Set U) by using a selection of PAH molecules (Set S) from the ZINC15 database (Set T), where S is a proper subset of T, and U is a Cartesian product of S is defined in the formula (1.2), where U denotes the set of ordered pairs (s,s) with $s \in S$ and it represents possible combinations of co-crystal molecules:

$$U == \{(s_1, s_2) \in S \times S \mid s_1 \neq s_2\}) \tag{1.2}$$

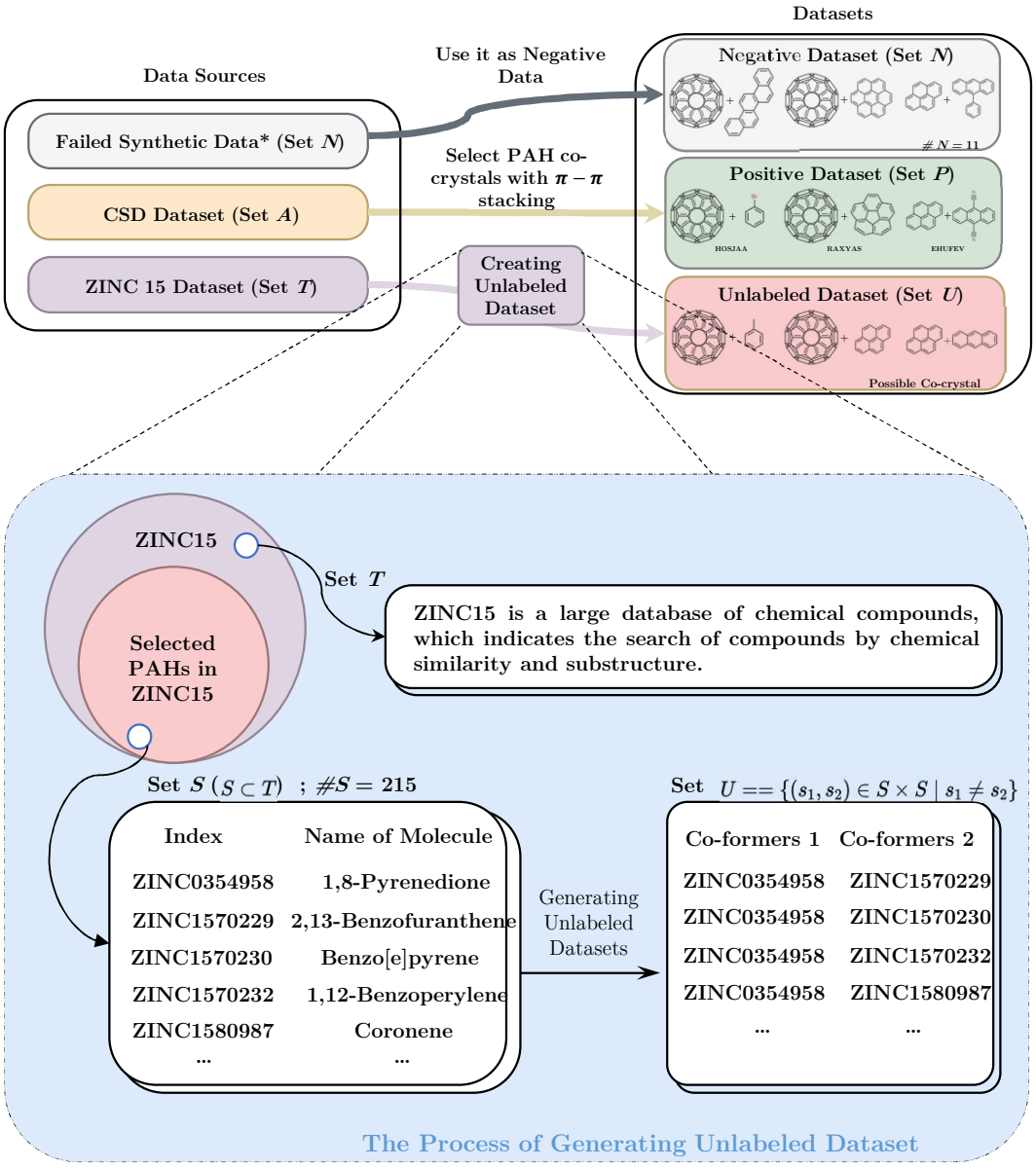


Figure 1.6: Data Preparation for Co-crystal Prediction

1.5.2 Feature Representation

Since the input to machine learning is a feature vector, molecules need to be transformed in a way to be understandable from the computer. Mordred is a freely available library for calculating molecular descriptors based on the string representation of the molecule (SMILES). [20]. In Figure 1.7, a co-crystal formed by the combination of two different molecules is represented by feature vectors divided into two co-former, each representing the molecules that form the co-crystal. For example, for the positive dataset, the blue rectangle corresponding to co-former 1 represents one of the molecules that make up the co-crystal. The small blue piece represents the feature vector of that molecule. Once we have completed the feature representation, we also need to do some data cleaning (in *Chapter 4.4*).

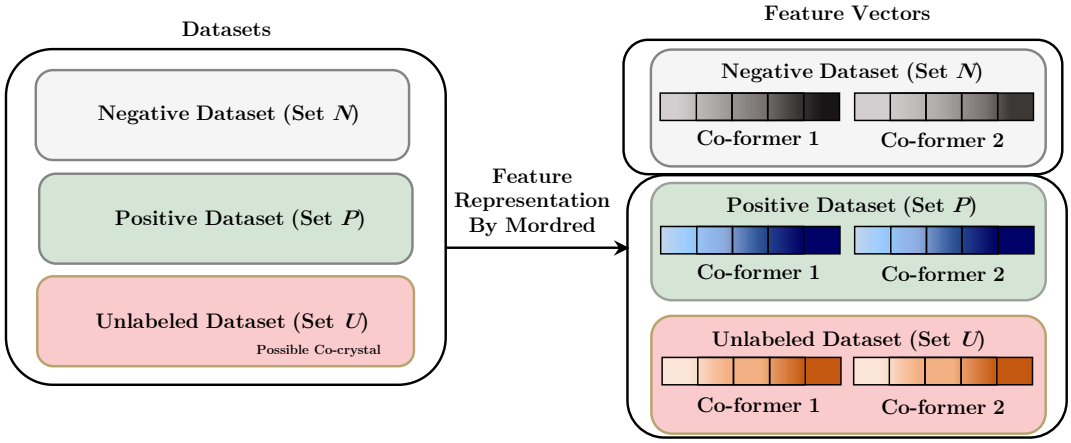


Figure 1.7: Feature Representation

1.5.3 Machine Learning Models

When the data has been prepared, we use three different workflows to achieve co-crystal prediction:

(1) Positive and Unlabeled Learning (PU Learning): After feature engineering, we used the positive and unlabelled learning algorithm (PU Learning) to implement the co-crystal prediction. The basic idea of PU learning for solving the data imbalance problem is shown in Figure 1.8.

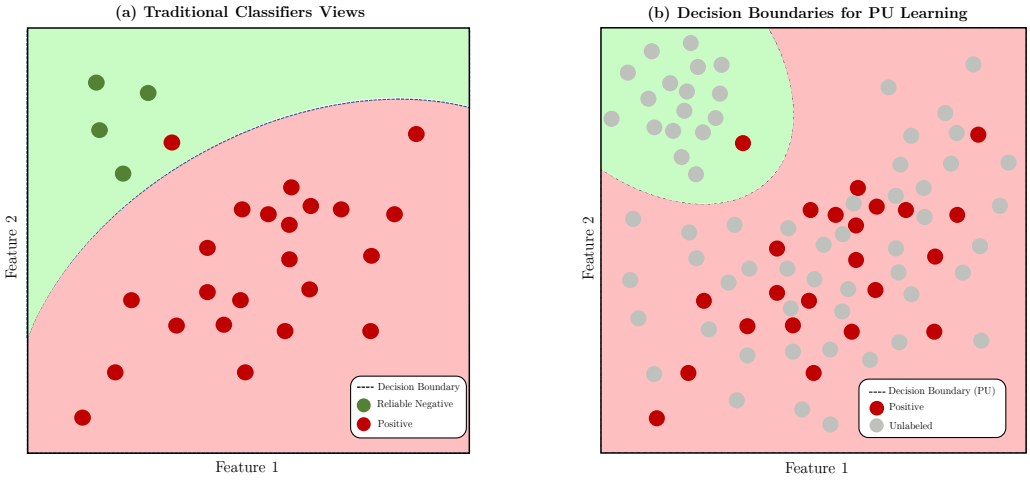


Figure 1.8: (a) Traditional binary imbalanced classification assuming the existence of reliable negative and positive samples. (b) Decision bounds after using PU learning algorithms.

Assuming we have reliable negative and positive samples, the decision boundary obtained by the traditional imbalance algorithm is shown in Figure 1.8 (a). As shown by the dashed line in Figure 1.8 (b), when we consider the distribution of unlabelled samples, the decision boundary is significantly different from that produced by the traditional algorithm. In this case, the decision boundary of the positive learning and unlabelled learning (PU Learning) algorithms have more reasonable decision bounds. However, the example in Figure 1.8 (a) is derived based on reliable negative samples, and the traditional binary classification algorithm is flawed when we cannot determine the reliability of the

negative samples. The PU learning algorithm can effectively handle unreliable data, such as the Spy algorithm, can achieve classification by finding reliable negative class samples in unlabeled data [21].

(2) Data Pre-processing Approaches: As shown in Figure 1.9, the model uses the DBSCAN [22] to remove outliers from the positive dataset (Set *P*). Then the number difference between positive and negative was balanced by the random under-sampling method and Synthetic Minority Over-sampling Technique (SMOTE) [23]. The model is constructed using a traditional binary classifier. In this paper, we use a random forest as the fundamental model. Finally, the unlabelled data were fed into the model for co-crystal prediction.

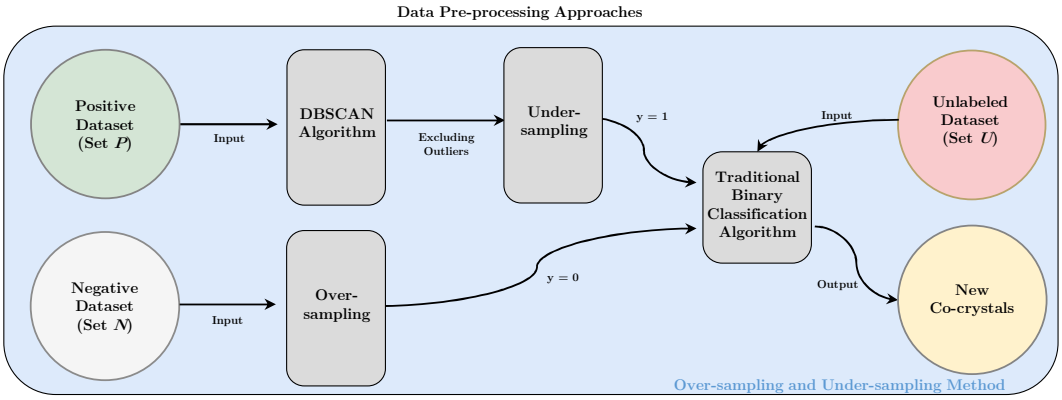


Figure 1.9: Data Pre-processing Approaches

At the beginning of the study, we did not fully understand the reliability mechanism of the negative class samples. In fact, the method was not valid because the artificially constructed negative samples were based on unreliable and biased negative samples.

However, in the future, if we have more unbiased negative samples, and if we can ensure higher reliability of negative samples through more repeated chemical experiments, this method is feasible.

(3) One Class Classification Based on the Isolation Forest:

As shown in Figure 1.10, one class classification algorithm only considers positive samples, which effectively prevents the problem of unbalanced and unreliable negative data. In this study, we use Isolation forest [24] as one class classification algorithms, since the core classification algorithms of our existing workflows use ensemble methods.

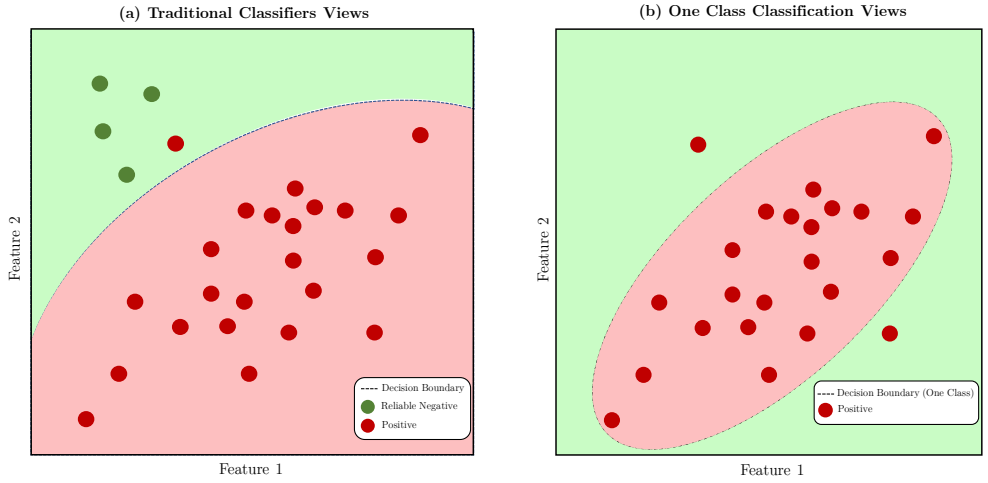


Figure 1.10: One Class Classification

Chapter 2

Literature Review

A major problem with the use of data-driven routes for co-crystal prediction tasks is the lack of negative examples, meaning that combinations of molecules that have been experimentally verified for not forming co-crystals are barely recorded. Several groups have now developed methods to deal with this problem and accelerate co-crystal discovery. One thing that most of these methods have in common is that they are all focusing on generating some negative data to enable the training of binary classifiers.

Wang et al. used the entire CSD co-crystal dataset and randomly generated artificial negative values (Negative values were constructed based on the Tanimoto similarity coefficient) to train several binary classifiers based on different binary classification algorithms. In addition, the effect of different chemical pair representations (molecular fingerprints) on different binary classification algorithms was also explored in the study [17].

Similarly, Wicker et al. created an in house co-crystal dataset of 680 molecular pairs consisting of both successful and unsuccessful experiments to train a binary support vector machine classifier achieving good performance both in the test set and an external validation set. In order to cope with the severely unbalanced data, the team selected 20 target molecules for the synthesis of co-crystals, which were manually screened to eliminate combinations of uncertain

results and to make the negative data more reliable [25].

Devogelaer et al. extracted the whole dataset of existing co-crystals in CSD (approximately 7000 structures) and then artificially generated the same amount of negative examples based on low scoring molecular pairs from a link prediction algorithm. The method is based on network science and assigns a statistical probability to the existence of a co-crystal of two co-formers. Using these balanced classes with both positives and negatives, they implemented neural network workflows to train the binary classifier [26].

Some studies directly use one classification algorithms to achieve co-crystal prediction. The scarcity of negative samples was avoided by Vriza et al. using multiple one-class classification algorithms, which used 1722 positive molecular combinations from the CSD database and generated an unlabelled dataset indicating possible molecular combinations using the ZINC15 database [27]. The final neural network-based model was selected based on cross-validation evaluation. This model led to the discovery of two new types of co-crystals: pyrene-6H-benzo[*c*]chromen-6-one and pyrene-9,10-dicyanoanthracene [27].

In addition, positive and unlabeled learning methods have been applied in several areas, for example, in text classification, where the Spy algorithm was used by Liu et al. to obtain reliable negative samples [28]. In the field of drug discovery, Lan et al. used PU learning to predict drug-target interactions [29]. In the field of materials, Nathan et al. used PU bagging algorithm to predict the most likely candidates for the successful synthesis of theoretically proposed 2D materials [30].

Although PU learning has been used in the field of materials prediction, there is a lack of research and literature in the field of co-crystal prediction. We would like to use a similar approach for this task.

In the present work, three workflows were developed for dealing with the co-crystal prediction problem. The first workflow is focused on developing a PU learning classifier that is training only on the positives and unlabelled data. In contrast, the second workflow involves generating artificial outliers based on some unsuccessful experiments between PAHs and the training of a binary classifier. The third workflow uses positive data to construct one class classifier.

Chapter 3

Theoretical Foundations

3.1 Label Mechanism of PU Learning

The PU data is defined as (x, y, s), where x represents the feature vector, y represents the actual class of the sample, and s represents whether the sample has been labelled by us [31].

The significance of the introduction of s stems from the practical problems we face in life, for example, in co-crystal synthesis experiments, where the domain expert is very sure about the positive co-crystal sample that can be verified repeatedly. In the case of negative samples, because of the unreliability of negative samples, the domain expert needs to eliminate all possibilities in order to confirm, which is unrealistic and requires a lot of time and effort to validate negative samples. To address this issue, we require that in sample acquisition experiments, we label positive samples that can be confirmed as $s = 1$ (this means that we expect samples at $s = 1$ to be unbiased samples of $y = 1$), while for uncertain results, we label them as $s = 0$ (there are both positive and negative data in the sample marked with $s = 0$.)

As shown in Table 3.1, we labeled positive data in the CSD as $s = 1$ and for molecular combinations in ZINC15, we labeled them as $s = 0$.

Table 3.1: Example of Label Mechanism of PU Learning

Num	**Source**	**Molecule 1**			**Molecule 2**			***y***	***s***
		ABC[1]	***...***	***Vabc***[2]	***ABC***	***...***	***Vabc***		
1	CSD	81.49		1329.52	57.05		7.87	1	1
2	ZINC15	14.78		207.91	15.23		179.87	/	0
3	ZINC15	14.78		207.91	16.64		211.82	/	0

3.2 Definition of PU Learning

Once we understand the labeling mechanism of PU learning, we give the following definition:

Given a set of instances of a positive class P and a set of unlabeled instances U which containing instances of class P and negative class N, the goal is to build a binary classifier that classifies the test set T into positive and negative classes, where T can be U [32].

3.3 PU and Binary Classification Conversions

We illustrate the core idea of PU learning by establishing a link between traditional binary classification and PU learning. First, traditional binary classification is supervised learning, and the classifier is built by using the feature vector $\boldsymbol{x}$ and the label $\boldsymbol{y}$ of the sample, we recognise that for all feature vector $\boldsymbol{x}$ obeying the conditional probability $\mathbf{Pr}(x \mid y \in \{0,1\})$, where $\mathbf{Pr}(y = 1)$ is the positive

[1] atom-bond connectivity index
[2] ABC van der waals volume

prior probability, for PU learning problem we do not know the proportion of positive samples in our overall distribution, however we can estimate the category prior probability by constructing a non-traditional classifier [31]:

$$\begin{aligned} &\boldsymbol{x} \sim \mathbf{Pr}(x \mid y \in \{0,1\}) \\ &\Leftrightarrow \boldsymbol{x} \sim \mathbf{Pr}(y=1)\,\mathbf{Pr}(x \mid y=1) + \mathbf{Pr}(y=0)\,\mathbf{Pr}(x \mid y=0) \end{aligned} \tag{3.1}$$

Based on the definition of PU learning and labelling mechanism, we can get:

$$\mathbf{Pr}(y=1 \mid s=1) = \frac{\mathbf{Pr}(y=1, s=1)}{\mathbf{Pr}(s=1)} = \frac{\mathbf{Pr}(s=1)}{\mathbf{Pr}(s=1)} = 1 \tag{3.2}$$

However, the problem is that we need to use a labelling mechanism to describe the process of selecting from the set of hidden positives in the unlabelled set, we define propensity score function [31]:

$$e(x) = \mathbf{Pr}(s=1 \mid x, y=1) \tag{3.3}$$

Based on the concept of labeling and the formula (3.1) for the probability density distribution $\mathbf{Pr}(x \mid y=1)$ for positive samples in traditional binary classification, similarly, we define the probability density distribution for labeled samples, it is important to note that the labelled samples are part of the full positive samples, and the relationship between the labelled samples and the full positive samples depends on the way we sample the labels [31]:

$$\mathbf{Pr}(x \mid s=1) \subset \mathbf{Pr}(x \mid y=1) \tag{3.4}$$

Based on the definition of PU learning, we transform the above equation into the following form, where $\mathbf{Pr}(s=1 \mid x, y=1)$ is the propensity score function, and $\mathbf{Pr}(s=1 \mid y=1)$ is the proportion of the label set to the overall positive sample, and $\mathbf{Pr}(x \mid y=1)$ is the probability distribution function of the positive sample [31]:

$$\begin{aligned} &\mathbf{Pr}(x \mid s=1) = \mathbf{Pr}(x \mid s=1, y=1) \\ &= \tfrac{\mathbf{Pr}(s=1|x,y=1)\mathbf{Pr}(x|y=1)}{\mathbf{Pr}(s=1|y=1)} \\ &= \tfrac{\mathbf{Pr}(s=1|x,y=1)}{\mathbf{Pr}(s=1|y=1)}\mathbf{Pr}(x \mid y=1) \\ &= \tfrac{e(x)}{\mathbf{Pr}(s=1|y=1)}\mathbf{Pr}(x \mid y=1) \end{aligned} \tag{3.5}$$

To simplify the model, we assume that the positive and unlabeled sets are from the same dataset [3] [31], which means that all samples of the molecular combinations obey a uniform distribution:

$$\begin{aligned} &\boldsymbol{x} \sim \mathbf{Pr}(y=1)\,\mathbf{Pr}(x \mid y=1) + \mathbf{Pr}(y=0)\,\mathbf{Pr}(x \mid y=0) \\ &\Leftrightarrow \mathbf{Pr}(y=1)\,\mathbf{Pr}(x \mid s=1)\,\mathbf{Pr}(s=1 \mid y=1) \\ &\quad +(1-\mathbf{Pr}(y=1))\mathbf{Pr}(x \mid s=0)\,\mathbf{Pr}(s=1 \mid y=1)) \end{aligned} \tag{3.6}$$

In the above equation, we learn that the following relationship exists between the a class prior $\mathbf{Pr}(y=1)$ and the label frequency $\mathbf{Pr}(s=1 \mid y=1)$, where $\mathbf{Pr}(s=1)$ represents the proportion of labelled samples to the total positive:

$$\mathbf{Pr}(s=1 \mid y=1) = \frac{\mathbf{Pr}(s=1, y=1)}{\mathbf{Pr}(y=1)} = \frac{\mathbf{Pr}(s=1)}{\mathbf{Pr}(y=1)} \tag{3.7}$$

[3] We simplified the model by selecting PAHs molecules.

In addition to this, we need to focus on the distribution of the overall positives for our selected labelled positives. To simplify the model, as shown in Figure 3.1, we assume that our labelled samples are selected completely at random (SCAR) [31], and under the current assumptions, we achieve unbiased[4] sampling for the labelled samples. In this case, our propensity score function $e(x)$ can be considered as:

$$e(x) = \mathbf{Pr}(s = 1 \mid x, y = 1) = \mathbf{Pr}(s = 1 \mid y = 1) \tag{3.8}$$

And since we have unbiased sampling positive for labelled samples and i.i.d, the relation in equation 3.4 can be defined as:

$$\mathbf{Pr}(x \mid s = 1) = \mathbf{Pr}(x \mid y = 1) \tag{3.9}$$

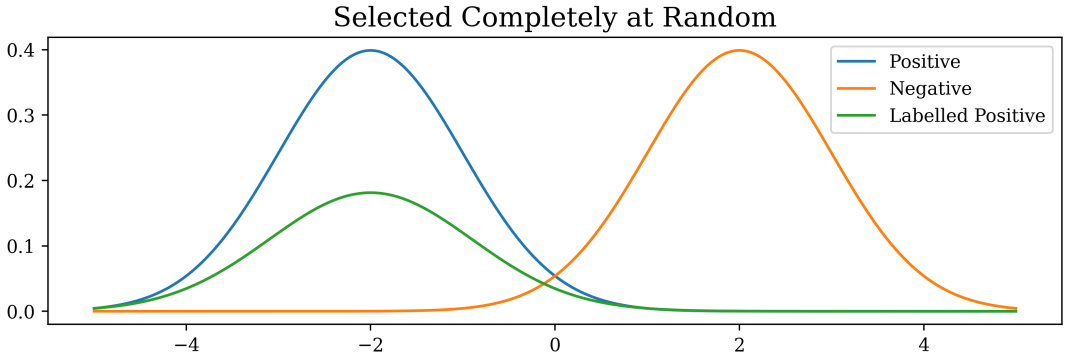


Figure 3.1: Selected completely at random assumption

In the following we build a link between traditional binary classification and

[4]This hypothesis holds when we have enough positive data in the CSD database.

PU learning by reconstructing the propensity score function under SCAR [31]:

$$\begin{aligned}
&\mathbf{Pr}(s=1 \mid y=1) = \mathbf{Pr}(s=1 \mid x, y=1) \\
&= \frac{\mathbf{Pr}(s=1,x,y=1)}{\mathbf{Pr}(x,y=1)} \\
&= \frac{\mathbf{Pr}(s=1,x)}{\mathbf{Pr}(x,y=1)} \\
&= \frac{\mathbf{Pr}(s=1,x)}{\mathbf{Pr}(x)} \frac{\mathbf{Pr}(x)}{\mathbf{Pr}(y=1,x)} \\
&= \frac{\mathbf{Pr}(s=1|x)}{\mathbf{Pr}(y=1|x)} \\
&\Rightarrow \mathbf{Pr}(s=1 \mid x) = \mathbf{Pr}(s=1 \mid y=1)\,\mathbf{Pr}(y=1 \mid x)
\end{aligned} \tag{3.10}$$

We know that for the traditional binary classification problem the objective is $\mathbf{Pr}(y=1 \mid x)$, while the objective of PU learning is $\mathbf{Pr}(s=1 \mid x)$, according to the above equation we can convert the PU learning problem to the traditional binary classification problem under SCAR conditions, where we only need to know $\mathbf{Pr}(s=1 \mid y=1)$, i.e. the proportion of the label set to the overall positive sample. In fact, in the discussion of the results in Chapter 5, we found that we can treat unlabeled data as 0 and all labeled samples as 1, and convert the PU learning classifier into a traditional binary classifier by lowering the threshold of the traditional binary classification (the essence of this threshold is $\mathbf{Pr}(s=1 \mid y=1)$).

Chapter 4

Materials and Methods

4.1 Extraction of Unlabelled Datasets

There are several ways to search the materials databases and construct a dataset of interest. In that work, we searched two databases for extracting PAHs. Initally, we extracted the currently known PAHs co-crystals from the CSD and then used the ZINC15 database to extract single PAHs that can be combined for forming possible molecular pairs. The Tanimoto similarity was implemented for scanning both databases, searching for molecules similar to a starting set of representative PAHs. The Tanimoto similarity uses the ratio of the intersecting set to the union set as the measure of similarity [33]. Represented as a mathematical equation, where N_a and N_b represents the number of attributes in each object (a, b) and N_{ab} is the number of attributes in common:

$$T(a,b) = \frac{N_{ab}}{N_a + N_b - N_{ab}} \tag{4.1}$$

By the above method, we finally selected 1861 positive samples from the CSD database (Positive samples consisted of two different molecular pairs) and 214 candidate molecules from ZINC15. According to equation (4.1), we need to combine these 214 candidate molecules to represent possible co-crystal combina-

tions, and we finally obtained a possible co-crystal combination of size 22577.

4.2 Failed Synthesis Experiments as Negative

In workflow 2, we wanted to use the negative class data to construct artificial negative class samples. We used a combination of molecules that scored high in the Vriza group's one-class classification model but failed to synthesise (See in Table 4.1) [27]. Due to the time and labour costs of chemical experiments, it is not possible to experiment with every possible combination, which is why negative samples are extremely limited. In addition, as shown in the table below, the use of data generation methods to generate artificial data for a limited number of negative class samples is likely to be strongly biased because the vast majority of negative class samples use C60 or Pyrene, and if random data is generated in high dimensional space and a binary classification algorithm is used, the algorithm is likely to be biased against the above molecules, i.e. to assume that new co-crystals are unlikely to be synthesised as long as a combination of C60 or Pyrene molecules is present. In fact, from a chemical point of view, combinations of molecules with C60 or Pyrene present have a higher probability of synthesizing new co-crystals, as both molecules provide multiple sites for π-π bond formation. In addition, the model obtained using the deep class learning algorithm also showed that Pyrene was the molecule with the highest frequency and score of predictions in the model [27]. The above message suggests that unless we have more diverse data, using oversampling methods to generate artificial negative class data may not result in a well-performing model.

Table 4.1: Failed Synthesis Experiments Set

Molecule 1	Molecule 2	Label
C60	Picene	0
C60	Fluoroanthene	0
C60	1,9-dicyanoanthracene	0
C60	9,10-bis(phenylethyl)anthracene	0
C60	Coronene	0
Pyrene	Triphenylene	0
Pyrene	1,2,3,4-tetrahydrophenanthrene4-one	0
Pyrene	1-vinyl-naphthalene	0
Pyrene	9-phenylanthracene	0
Phenanthracene	Triphenylene	0

4.3 Co-crystals Representation

The first and necessary step before any machine learning analysis of materials science data is to represent the material under consideration in a way that the computer can understand. In the process shown in Figure 4.1, we first use the Simplified Molecular Input Line Input System (SMILES) to represent different molecules. A SMILES string is a text-based representation of a molecule to describe its structure [34]. When we are given a SMILES string, molecular descriptors can be calculated. A *molecular descriptor* is defined as ”the final result of a logic and mathematical procedure, which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardised experiment” [35]. The calculation of molecular descriptors is based on mathematical and statistical methods to explain quantitative patterns of variation between the activity or physicochemical characteristics of a compound and its molecular structure [36–38]. In this study, we used Mordred to convert SMILES calculations into molecular descriptors because of the good performance and convenience of Mordred as free, open-source software [20]. To represent a co-crystal composed of two different molecules, as shown in Figure 4.1, we represent the co-crystal using a vector connection of

Mordred molecular descriptors [20], with each block in the vector representing a different molecular feature. To prevent bias in the machine learning model, two pairs of molecular descriptors describing the same set of co-crystals were swapped and recorded separately in the database [1].

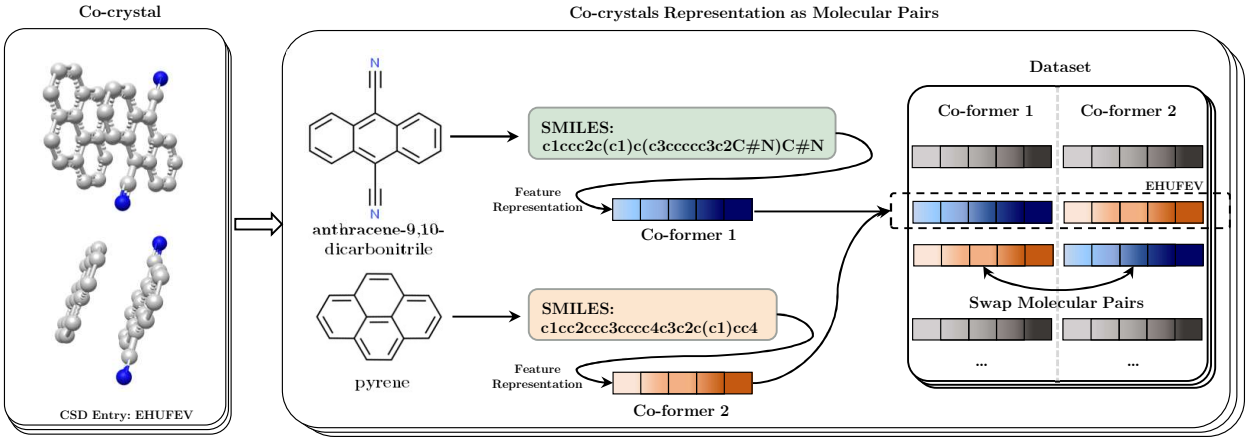


Figure 4.1: The diagram shows the basic process for representing a co-crystal consisting of two different molecules as features. In this example, we retrieve in the CSD database the co-crystal with the label 'EHUFEV', which consists of two different molecules, anthracene-9,10-dicarbonitrile and pyrene molecules. We used SMILES to represent these two molecules, and then used the Mordred molecular descriptor calculator to represent the co-crystal as two co-formers, where each small rectangle represents a feature. The two sets of eigenvalues were joined at the beginning and end to represent 'EHUFEV'. In addition, to ensure that the machine learning model was not biased, we swapped the order of the two sets of feature values and joined them again first and last.

4.4 Feature Engineering

In general, data and features determine the upper limit of machine learning, and models and algorithms simply approximate that limit. The current co-crystal corresponds to a 2×1613 molecular descriptor, and for traditional binary classification algorithms, the time complexity in the current dimension makes it necessary to reduce the dimensionality of the data. In this study, we performed Pearson correlation tests on the co-formers of the positive samples, where we

[1] Similar to symmetrical images in image processing.

removed features with correlations greater than 0.8 and used variance filtering for features with variances less than 0.4. In addition, we linked the molecules into pairs and removed features with correlations below 0.4 using Pearson and Spearman correlations, as we considered those with Pearson and Spearman correlations both above 0.4 and p-values below 10^{-3} to be significant and unbiased features [27, 39]. In this study, 40 features[2] were selected to represent the co-crystals (See Appendix A).

Because of the different magnitudes of the parameters, we use min-max normalization [40] to reduce the range of values to the interval [0,1], where x_i is an original value and x_i' is the normalized value:

$$x_i' = \frac{x_i - x_{min}}{x_{\max} - x_{\min}} \tag{4.2}$$

[2]Since we used two molecules to represent the co-crystals, we used 20 features for a single co-former.

4.5 Machine Learning Models

4.5.1 Two-step PU Learning Model

Overview

Based on the divisibility and smoothing assumptions, all positive samples are similar to labelled samples ($s = 1$) and different from negative samples [31]. In the above assumptions, we use a two-step PU learning algorithm, which first identifies reliable negative data using the Spy technique and classifies them using a traditional supervised algorithm.

As shown in Figure 4.2, we used a two-step technique in PU learning:

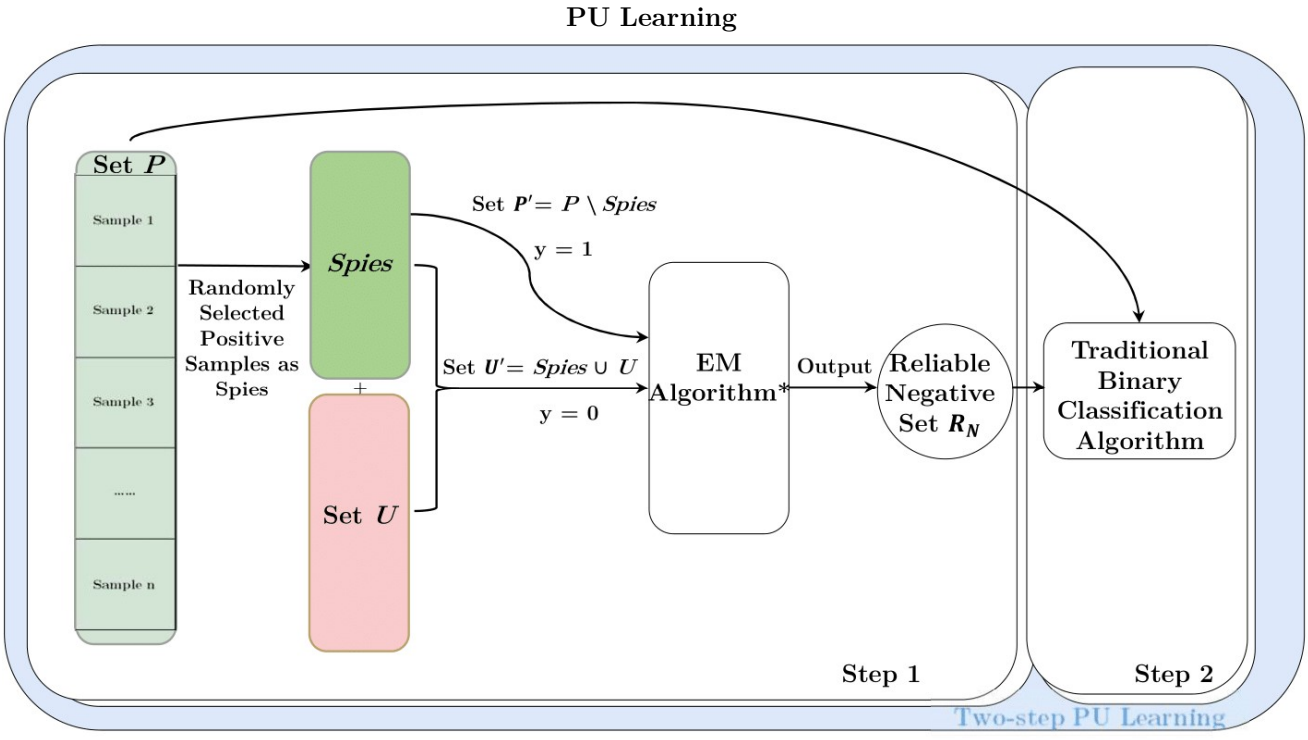


Figure 4.2: Two-step PU Learning

Step 1: First, we used the Spy algorithm to obtain reliable negative samples:

Select a random number of samples among the positive samples as *Spies* and then merge the set of these *Spies* with the unlabelled samples, we refer to the merged unlabelled data as U':

$$U' = Spies \cup U \tag{4.3}$$

Then we select the difference between the positive samples and the spies as the new positive sample input (P') to the EM algorithm[3], and U' as the negative sample input:

$$P' = P \setminus Spies \tag{4.4}$$

By iterating through the EM algorithm, we arrive at a reliable negative sample.

Step 2: Once we have a reliable negative sample, we can use other traditional binary classification algorithms to classify the reliable negative and positive samples for the task.

[3]According to the article *Partially supervised classification of text documents*, authors use the EM algorithm for text documents classification, however, the spy algorithm can be used in any module where the Sklearn framework can provide prediction probabilities.

Spy Algorithm

The PU learning algorithm is highly robust, which is advantageous when the data set is severely unbalanced or when there is unreliable data. The use of this method will make possible the discovery of new co-crystals.

As shown in Algorithm 1, we first enter the set of positive dataset P, the set of unlabelled data U, select the proportion of spies ratio s to be inserted into the unlabelled set and determine the spies tolerance θ. In the first step, the set of reliable negative RN is set as the empty set, and then a randomly selected positive sample of $s\%$ is used as a set $Spies$. Then we combine the unlabeled set U with the set $Spies$ as set U' and remove the set $Spies$ from the positive set P as P'. Subsequently, we train a classifier c, with P' as label 1 and U' as label 0 into the classifier. Finally, all elements of the unlabeled set U are iterated over, and if $\mathbf{Pr}(1 \mid u_i) \leq \theta$, the sample is added to the reliable negative sample set RN. In the second step, we construct the model M using the traditional binary classification algorithm with reliable negative RN and positive set P.

Algorithm 1: Two-step PU learning based on Spy

Input: Positive Set P, Unlabeled Set U, Spies Ratio s, Spies Tolerance θ

Output: Model M, Reliable Negative Set RN

```
Set RN = ∅ ;                                              // Step 1
Randomly select s samples from Set P as Spies;
Set U' = U ∪ Spies and Set P' = P \ Spies;
Treat P' as positive and U' as negative;
Train a classifier c with P' and U';
for u_i ∈ U do
    if Pr(1 | u_i) ≤ θ then
        RN = RN ∪ u_i;
Output RN;
Train a classifier M with P and RN ;                      // Step 2
```

We use Random Forests as a classifier for two-step PU learning, which is noise-resistant and can run on large datasets, and which provides an assessment of feature importance, facilitating the interpretability of our model. Random Forests belongs to the bagging (bootstrap aggregation) method in ensemble learning. It can be composed of multiple decision trees [41]. The structure of the Random Forests is shown in Figure 4.3, the middle part of the diagram represents multiple decision trees. Decision trees can be divided into three parts: Root, Node, and Leaf. For a single decision tree, we use the Gini index as a criterion for feature selection, where k denotes the k_{th} classification category and p_k denotes the probability that the sample belongs to the k_{th} category. In general, the larger the Gini index, the greater the uncertainty; the smaller the Gini index, the smaller the uncertainty, which implies a better classification using the split point used for the feature [41]:

$$\text{Gini}(p) = \sum_{k=1}^{K} p_k (1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2 \tag{4.5}$$

When we parallel connect multiple decision trees together, multiple decision trees are voted on and the final result is chosen. However, we also need to have diversity across the trees in order to generalise the results. We use Bootstrap samples before the training data is passed into each tree, which means that for each tree T_k, M_k training samples are randomly drawn from the training set with a putback as the training set for that tree (the putback sampling prevents biased sampling) [42]. In addition, for each node of the decision tree, some features are randomly selected[4], which allows for diversity between each decision tree.

[4]In order to clearly represent the difference in the construction of each decision tree, we have marked the difference in the input features for each decision tree above the decision tree, however, the actual algorithm represents that each node selects some features at random.

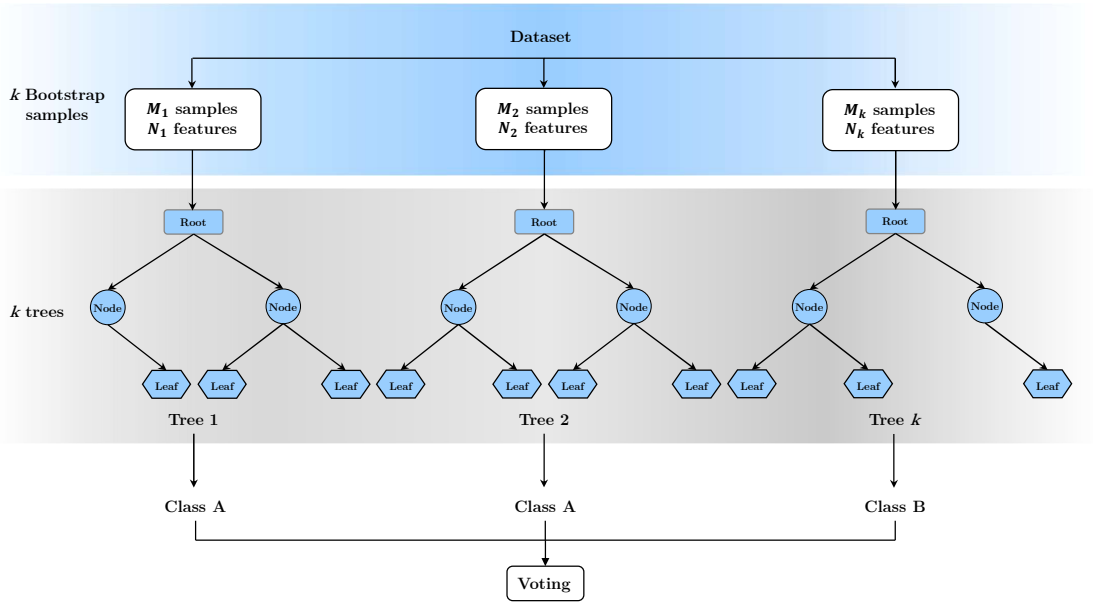


Figure 4.3: The figure represents the random forest model, and the light blue blocks in the figure indicate that each decision tree requires Bootstrap to obtain training data M_k, and that each node randomly selects some features N_k. When the data is input, all decision trees are constructed in parallel. Each decision tree produces a prediction when the data needs to be predicted, and these results are voted on to produce the final decision.

4.5.2 Data Pre-processing Model

For workflow 2, we first used the Density-based spatial clustering of applications with noise (DBSCAN) algorithm to remove outliers from the positive dataset.

Definition 4.5.1 (Neighbourhood)**.** It starts with any point p in the dataset and checks for neighbouring points within a given radius ε:

$$N_\varepsilon(p) = \{q \in D \mid \text{dist}(p,q) \leq \varepsilon\} \tag{4.6}$$

Definition 4.5.2 (Core object)**.** If there are points number greater than the threshold *MinPts* within a radius of ε, centered at p, point p is called a core object which is defined in formula (4.7):

$$|N_\varepsilon(p)| \geq MinPts \tag{4.7}$$

Definition 4.5.3 (Directly reachable)**.** The point p is the core object and the point p' is in core object's $\varepsilon - neighbourhood$, then called p' is directly reachable from p, we mark as:

$$p \mapsto p' \tag{4.8}$$

Definition 4.5.4 (Reachable)**.** If there exists a chain of core objects with $p_0 \mapsto p_1$, $p_1 \mapsto p_2$, ... , $p_{n-1} \mapsto p_n$, then point p_n is reachable from p_0:

$$p_0 \to p_n \tag{4.9}$$

Definition 4.5.5 (Noise)**.** If the point p is not reachable from any point, the point is marked as noise:

$$\emptyset \to p \tag{4.10}$$

As shown in Figure 4.4, we set *MinPts* to three, by definition, the red points are directly reachable by each other (the direction is bidirectional), point B does not constitute a core object as it cannot reach the threshold *MinPts*, however $A \to B$. So we have red and yellow points as a cluster; for blue point, it is not reachable from any point ($\emptyset \to p$), so we treat it as noise.

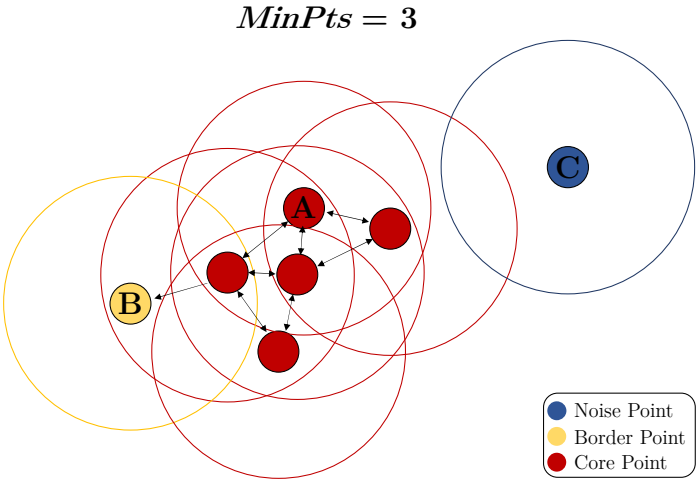


Figure 4.4: Example of DBSCAN based on a two-dimensional space.

However, the choice of the parameters Eps and $MinPts$ for DBSCAN is relatively difficult. In this research, we use the DMDBSCAN algorithm to find the appropriate parameters, the idea is to calculate the distance of all data points to their k_{th} nearest neighbour, with k-dist as the variable, and we can see a sharp change in the number of points matching the current distance [43]. As shown in Figure 4.5, we take the elbow point as the best Eps.

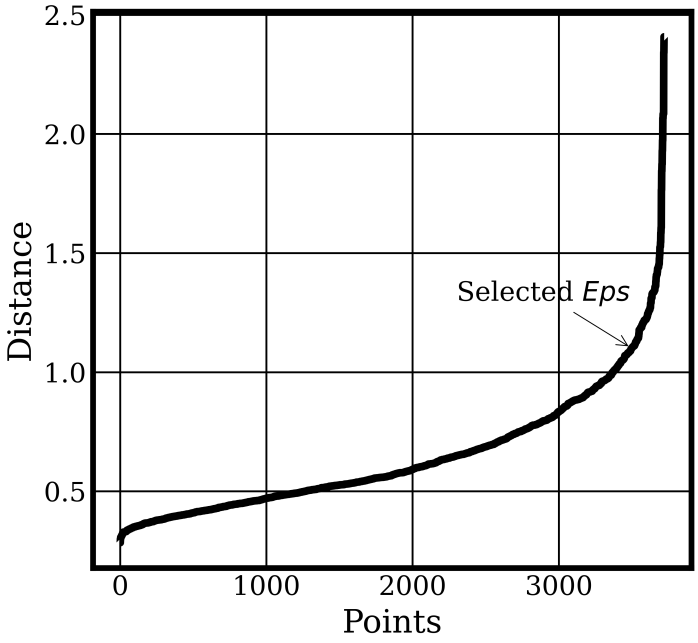


Figure 4.5: Finding the optimal value of Eps using the DMDBSCAN algorithm

After removing the outliers from the positive samples, we used a random under-sampling technique to reduce the number of positive and used Synthetic Minority Oversampling Technique (SMOTE) to generate artificial negative data [23]. The SMOTE can generate artificial data randomly by observing samples from certain minority classes [23]. The principle of the algorithm is shown in Figure 4.6. For each sample in the minority class, the L2 distance is used to calculate the distance to all samples of the minority class, and the sampling rate n is determined, then several samples are randomly selected from their neighbours and constructed according to the formula (4.11):

$$x_{new} = x_i + rand(0,1) \cdot |\hat{x}_i - x_i| \tag{4.11}$$

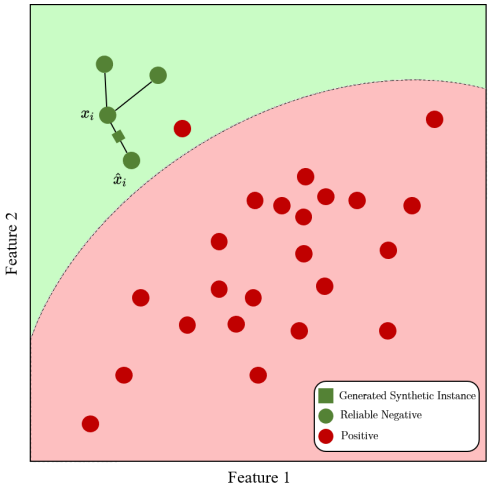


Figure 4.6: Synthetic Minority Oversampling Technique can generate artificial data randomly based on Euclidean distance

After outlier removal, over-sampling and under-sampling of the data, we use the Random Forest algorithm as the final classifier.

4.5.3 One Class Classification based on Isolation Forest

In workflow 3, we used an Isolation forest algorithm that was trained using only positive samples. We use a simple example to describe the Isolation forest algorithm, as shown in Figure 4.7, we perform a random cut on a two-dimensional plane, and pick a dimension of the current plane, then select a random cut point within the data range, the cut point divides the plane into two parts, and we continue this step recursively until we isolate the points on the plane. Figure 4.7(a) shows that a dense point X_i requires many cut lines to isolate it, whereas the point X_0 in Figure 4.7(b) is an outlier and can be seen to require only two partition lines to isolate it. In addition, since the choice of partition points is random, we need to use the ensemble method to make the results converge.

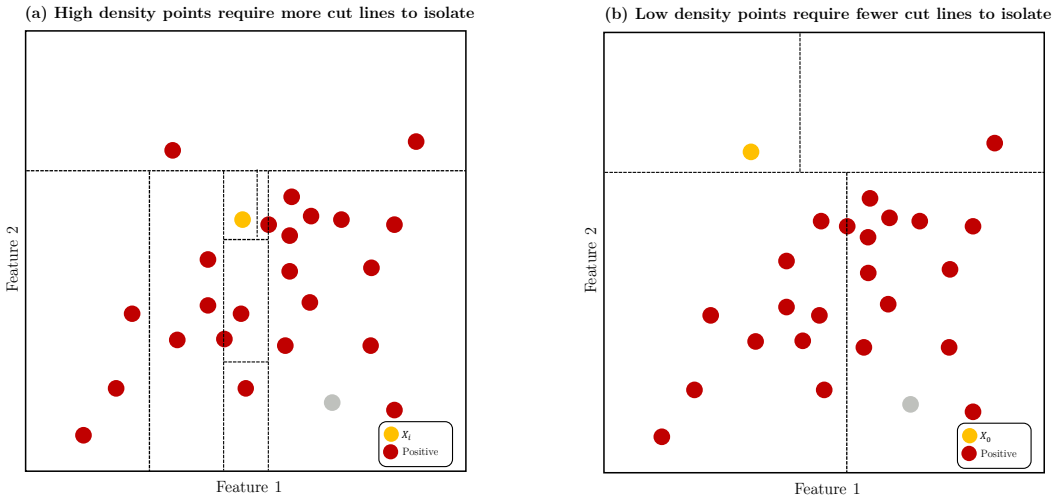


Figure 4.7: Isolation Forest Example

4.6 Hyperparameter Tuning and Model Evaluation

4.6.1 Model Evaluation Challenges

Hyperparameter tuning for positive and unlabelled data is difficult because the problem is completely different from traditional binary classification, where accuracy, precision, recall can be evaluated with a grid search hyperparameter to optimise the algorithm. However, because of the lack of reliable negative, we cannot know which of the unlabelled samples are positive. As shown in Table 4.2 we use the confusion matrix to describe the current dilemma.

Table 4.2: Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	*TP*	*FN*
Actual Negative	*FP*	*TN*

We usually use F_1 scores as a way of assessing traditional learning which is defined in equation 4.12, where the precision $p = \frac{TP}{TP+FP} = \mathbf{Pr}(y = 1 \mid \hat{y} = 1)$ and recall $r = \frac{TP}{TP+FN} = \mathbf{Pr}(\hat{y} = 1 \mid y = 1)$:

$$F_1(\hat{\mathbf{y}}) = \frac{2pr}{p+r} \tag{4.12}$$

According to the definition of the confusion matrix in Table 4.2, we can estimate the recall. However, we cannot calculate the precision rate because there are unlabelled samples that can be successfully synthesised and we cannot calculate the exact number of FP. In fact, we do not have the actual negative class, so the FP and TN in the second row of Table 4.2 are not known and cannot be calculated, so we cannot calculate the F_1 value.

We use the following evaluation criteria to approximate the F_1 score [44]:

$$F_1 = \frac{2pr}{p+r} \approx F = \frac{pr}{\mathbf{Pr}(\mathbf{y}=1)} = \frac{r^2}{\mathbf{Pr}(\hat{\mathbf{y}}=1)} \tag{4.13}$$

This method focuses on the numerator part of the F_1 score. If both precision and recall are high, then the F_1 score is also high. If one of precision and recall is low, then the F_1 score is low. Using this method, we can approximate the performance of the model [44].

Subsequently, we select the best model from the iteratively generated models according to the F-value enhancement strategy. When ΔF is larger than 1, the model performance is improving and we choose the model with the last improvement [45]:

$$\Delta F = \frac{F_i}{F_{i-1}} \tag{4.14}$$

However, the current F_1 approximation method suffers from the inability to compare data in the current study[5], and we are still using recall to evaluate the model. Once we resolve this issue in the future, the difficulties of the current evaluation will be effectively resolved.

[5]We used the approximation method from the pyod library to effectively evaluate one class classification algorithms. However, when we extended this method to PU learning, the scores of one class algorithms were not comparable to the scores of PU learning.

4.6.2 PU Learning and Isolation Forest Model

(1) Hidden Positive

In this study, we did not only evaluate the recall of the model, we further compare the ability of the different algorithm to detect potential positive samples. As shown in Figure 4.8, we trained three different types of classifiers using all positive samples with label 1 and unlabeled data with label 0 in the training set, and hiding 95% of the positive samples[6] (Setting the labels of some of the positive samples to 0).

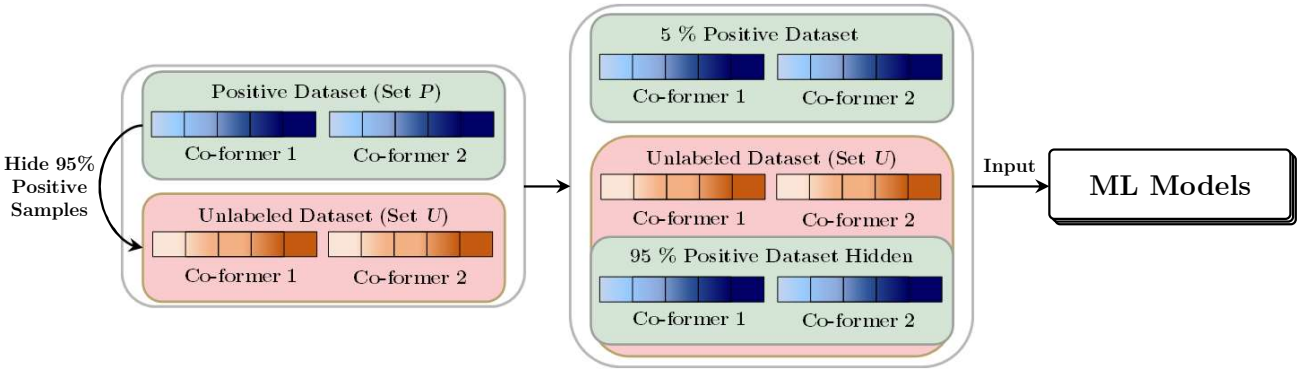


Figure 4.8: Hidden Positive

The three different classifiers are as follows:

1. Naive PU Learning (Using Random Forest classifier directly);

2. One Class Classification algorithm: Isolation Forest[7];

3. PU learning algorithm (Using the Spy algorithm as the first step[8] of the

[6]We do not know the real number of potentially synthesisable co-crystal combinations in the unlabelled data. It is feasible to assess the model's ability to uncover data by hiding positive samples to simulate the process by which we find synthesisable co-crystal combinations. However, this method has the drawback that some classifiers may find a large number of potentially synthesisable co-crystals instead of the original positive samples that we hide, resulting in a lower performance evaluation of the method. We compensate for this by assessing the distribution of prediction confidence of the classifiers and compared to the similarity of the remaining 5% of positive.

[7]Set contamination to 5%

[8]Set Spy Tolerance to 5%

two-step method and using Random Forest algorithm as the second step).

After the training is completed, let the model predict all unlabelled data (including hidden positive samples), we select a certain range of unlabelled data with high confidence of samples and compare between different models to find the percentage of hidden positive is found by each model.

(2) Additional Validation Set

In addition, as shown in Table 4.3, we provided the model with an additional validation set in which we used 11 failed co-crystal synthesis combinations and 5 successful co-crystal synthesis combinations not included in the positive training set for prediction.

Table 4.3: External Experimental Validation Set

Molecule 1	Molecule 2	Label
C60	Picene	0
C60	Fluoroanthene	0
C60	1,9-Dicyanoanthracene	0
C60	9,10-Bis(phenylethyl)anthracene	0
C60	Coronene	0
Pyrene	Triphenylene	0
Pyrene	1,2,3,4-Tetrahydrophenanthrene4-one	0
Pyrene	1-Vinyl-naphthalene	0
Pyrene	9-Phenylanthracene	0
Phenanthracene	Triphenylene	0
C60	Bromobenzene	1
C60	1,2-Dichlorobenzene	1
C60	Biphenyl	1
C60	9,10-Diphenylanthracene	1
C60	Corannulene	1

4.6.3 Data Pre-processing Model

In the future, this method could be feasible if we obtain more reliable negative samples for repeatable experiments:

We oversample the training set in each 5-fold cross-validation iteration to prevent the oversampled samples from being placed in both the training and validation sets to obtain spurious scores.

Then we used the Matthew correlation coefficient (MCC) to evaluate the quality of the model, it is often seen as a balancing measure that can be used even if the size of these classes is different. Matthew correlation coefficient is essentially a correlation coefficient between observed and predicted binary classifications; it returns values between -1 and $+1$. The coefficient $+1$ is a perfect prediction, 0 is no better than a random prediction, and -1 is a complete inconsistency between prediction and observation [46]. After testing, we found that MCC is very sensitive to FPR, which is helpful for us to use cross validation evaluation model. The definition of Matthew correlation coefficient is shown below:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4.15}$$

Chapter 5

Results and Discussion

5.1 Evaluation Result

5.1.1 Evaluation of Recall

As shown in Figure 5.1, we compared the Isolation forest-based one-class classification algorithm, the spy-based two-step PU learning and the Naive PU learning algorithm (directly treating positive data as 1 and unlabelled data as 0), respectively. We realise that the standard deviation of the Isolation forest algorithm fluctuates widely in the validation set, indicating that the learning process of the model is unstable in the validation set. However, the Isolation forest achieves a recall of 0.95 after inputting the complete training set. For the two-step PU learning algorithm based on the spy algorithm, it can be seen that the model achieves a suitable fitting curve and the cross-validation curve is more stable during the training process. The model finally achieves a recall of 0.94. For the Naive PU learning algorithm, the training process is stable but has a recall of 0.91. However, high recall is only one aspect, and with the limitations of the evaluation method, we still want to explore different models using other methods.

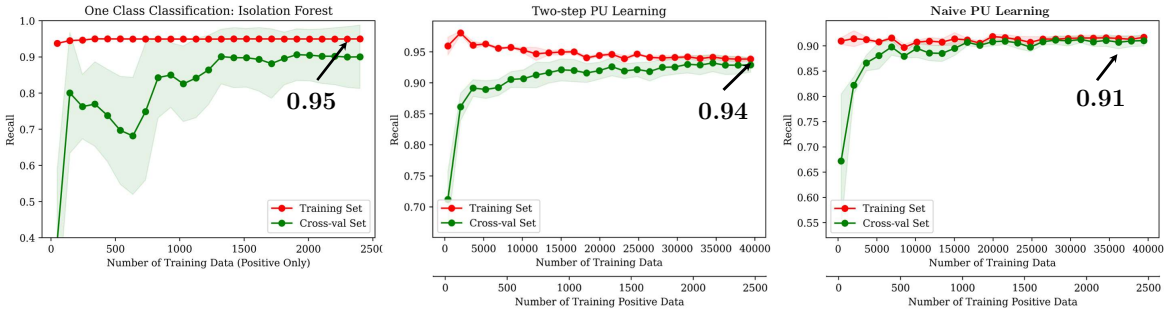


Figure 5.1: Learning curves for the three algorithms

5.2 Prediction Probability Distribution

In the previous section, we evaluated the recall of three different models. However, if all points are predicted to be inner, then the recall of an algorithm could be 100%. For this reason, we also need to examine and compare the discriminative ability of these algorithms by visualising the scores they assign to our unlabelled data.

As shown in Figure 5.2, we compare the predicted probability distributions of the three different algorithms for labelled and unlabeled data. By definition, we know that there are many positive and negative data in the unlabelled data. We take the decision threshold of the Isolation forest algorithm to 0.85, under which the algorithm can separate the outliers from the inner points in the unlabelled data. However, choosing a decision threshold of 0.85 means that we treat nearly half of the positive data as noise. We believe that the Isolation forest algorithm is less capable of separating in this case.

When we considered two-step PU learning, we chose 0.45 as the decision threshold for two-step PU learning. We found that the vast majority of unlabeled and labelled samples were well separated. Furthermore, the two-step PU learning algorithm yields about 20 data with a prediction probability of 0.9. We believe

that the distribution of these data points is very similar to that of the positive data.

For the Naive PU learning, we found that although the model could effectively separate the labelled data from the unlabelled data, the overlap obtained by the model was extremely limited, so we believe that the model obtained inferior results to the two-step PU method.

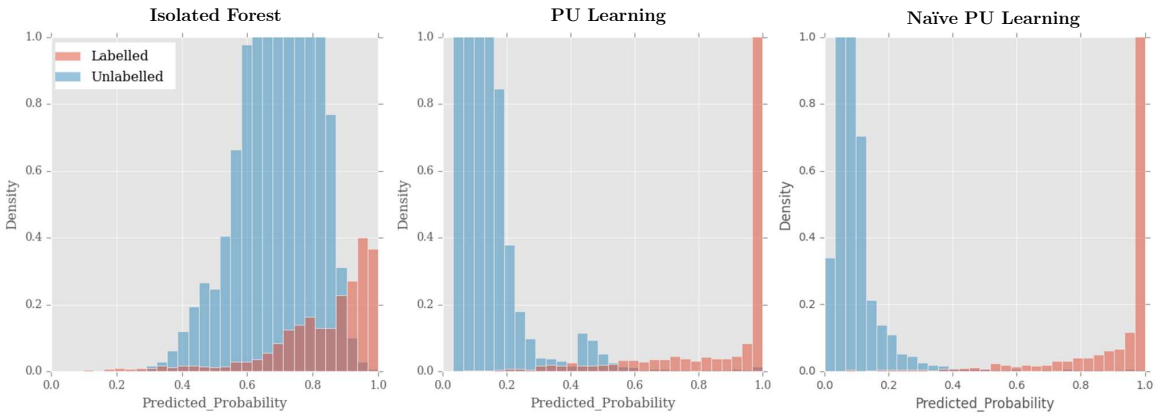


Figure 5.2: Predicted Probability Distribution Comparison

In the next section, we want to explore models to find positives hidden in unlabelled samples.

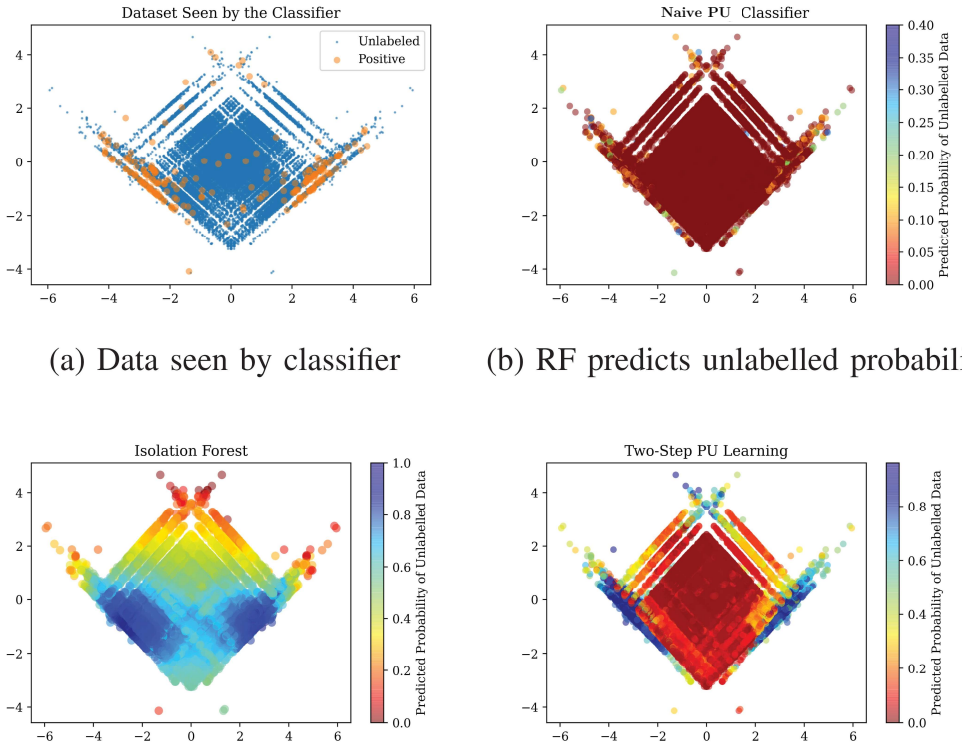
5.2.1 Hidden Positive

In this test, as shown in Figure 5.2(a)[1], we have hidden 95% of the positive samples. Figure 5.2(b) shows the prediction probabilities of the naive PU classifier based on Random Forest, and it can be seen that while the algorithm can mark a very small number of hidden positives with a relative high prediction probability, the majority of the hidden positives are in the range of 0 to 0.1. We recognise that although we can obtain a small number of hidden positives

[1]Figure is processed using PCA to reduce dimensionality.

by changing the decision threshold[2], the majority of hidden positives are difficult to distinguish from the negative category because the difference in prediction probability is small. In fact, based on the Separability and Smoothness assumptions, the hidden positives do differ from the negative samples, which may account for the small number of hidden positives found even with naive PU classifiers. Figure 5.2(c) shows the predicted probability distribution of the one class algorithm Isolation Forest. Since with the one class algorithm, we only need to input positive data when training the model, and the algorithm determines whether the unlabeled data are outliers or inliers. It can be seen that the Isolation Forest algorithm labels as positive a large amount of unlabelled data with a similar distribution to the positive training data. Figure 5.2(d) shows the probability distribution of unlabeled data predicted using the PU learning algorithm. There are many similarities between this algorithm and Isolation Forest in predicting probability distributions, however, the Isolation Forest algorithm labels more unlabelled data as positive.

[2]The evaluation in Figure 5.3 does not require a decision threshold, as we vary the decision threshold disguised by selecting a different number of samples with the highest predicted probability.



(a) Data seen by classifier

(b) RF predicts unlabelled probability

(c) Isolation Forest predicts unlabelled probability

(d) PU predicts unlabelled probability

Figure 5.3: Comparison of the predicted probability distributions of different algorithms after hiding some of the positive data

As shown in Figure 5.3[3], first, the x-axis represents the number of unlabeled samples (including hidden positive) that we selected as having the number of top prediction probability of the classifier. The y-axis represents the percentage of our hidden positives found in these unlabeled samples. For example, if we select 1000 unlabeled samples with top prediction probability, for naive PU classifier, nearly 80% of the unlabeled data are hidden positives.

After hiding 95% of the positive samples, the top 750 predictions of the above three classifiers were selected, and all classifiers had a good ability to find the hidden positive samples. However, when we continued to select the top 1000

[3]Please note that the parameters setting of each three algorithms are mentioned in Chapter 4

predictions, the naive PU classifier had difficulty distinguishing between potential positive samples and other unlabeled samples due to the large amount of data distributed in the prediction probability range 0 to 0.1 and the lack of differentiation in the probability scores of these data. Both the Spy-based PU learning algorithm and the Isolation Forest have the ability to find hidden positives. When we continued to increase the search range to 3000 samples, the Spy-based PU learning algorithm retrieved about 92% of the hidden positive samples, while Isolation Forest found about 60% of the hidden positive samples, and the naive PU classification algorithm based on Random Forest reduced to about 25%. We believe that the ability of the one class algorithm Isolation Forest and the PU learning algorithm to find hidden positive samples is considerable.

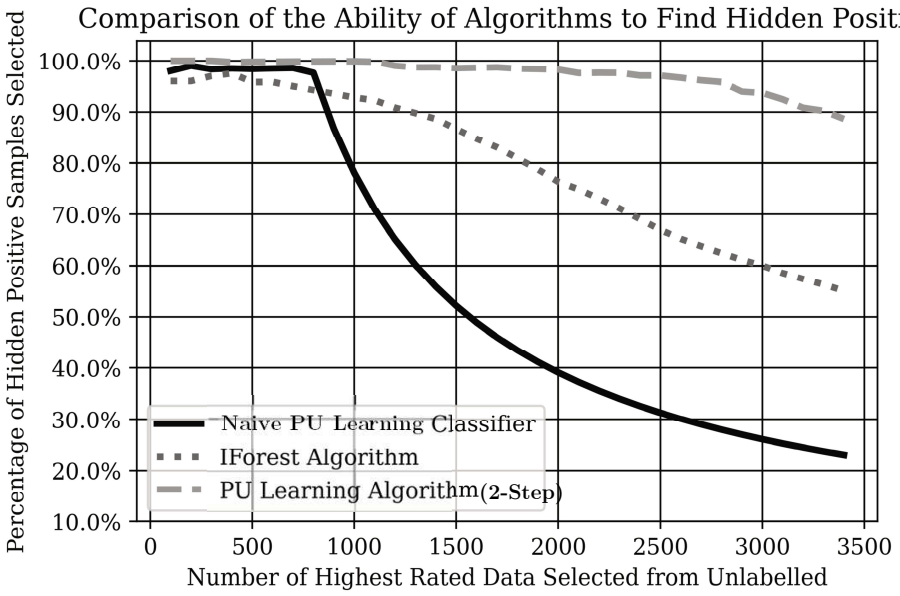


Figure 5.4: Comparison of the Ability of Algorithms to Find Hidden Positive: Comparing three different algorithms for selecting different numbers of predicted probability rankings found the proportion of positive samples in unlabelled data that were previously hidden.

5.2.2 External Experimental Validation

Figure 5.4 shows the prediction probabilities of the additional validation sets we tested, where the experimentally observed successful co-crystal molecule pairs are represented by green diamonds and the failed synthesis experiments are represented by red triangles. It can be seen that there are more successful experiments with higher prediction probabilities than failed synthesis experiments in the model constructed by the PU learning algorithm [4]. We recognise that the model is sensitive to combinations of molecules containing C60 and has a relatively high prediction probability, which is reasonable from a chemical point of view, as C60 can provide multiple $\pi - \pi$ sites and the probability of synthesising co-crystals of this molecule with other molecules is relatively high.

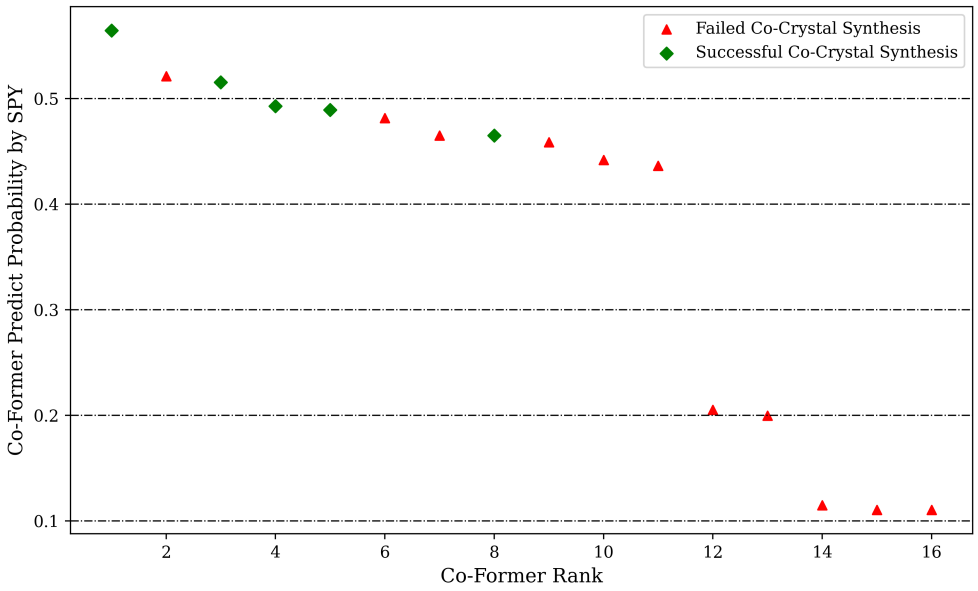


Figure 5.5: External Experimental Validation Prediction Probability Ranking

[4]We also tested the isolation forest model, which has a prediction probability greater than 90% for all samples in this experiment and cannot distinguish between successful and unsuccessful experiments. The problem arises because the data for the failed experiments were all derived from the highly scored molecular pairs in the One Class research [27].

5.3 Most Popular Molecules in the High Scoring Pair

In chemical terms, the higher the frequency of occurrence of the molecule in a prediction, the higher the probability that the molecule will be part of a new co-crystal. As shown in Table 5.1, the spy-based two-step PU learning algorithm was used as the end-use prediction model. After comparing the workflow with one class study when applied the new data and new molecular descriptors [27], we initially found that the top 3 predictions are the same.

Table 5.1: Top 5 Most Popular Molecules in the High Scoring Pair in PU Learning

Rank	ZINC15 Index	Molecule
1	ZINC000085548520	C60
2	ZINC000000967534	Toluol
3	ZINC000000968282	O-Xylol
4	ZINC000197382366	Circumcircumpyrene
5	ZINC000087515592	Circumovalene

5.4 Model Interpretation

As shown in Figure 5.6, since we use random forests in the two-step PU learning algorithm, we evaluate the importance of the features in the second step of the random forest (See *Appendix A* for the full meaning of the feature names). The implementation of this method can be simply understood as we randomly add noise to some of the features. If some of the features added to the noise do not have an excessive impact on the overall prediction results, then the features are considered unimportant. On the contrary, if some features with added noise have a significant impact on the overall prediction result, they are vital.

We can summarize their overall meaning as following:

(1) MIC1, MIC2, MIC3, IC3, IC4, IC5, ATS3s, ATS5s, GGI2, piPC8, piPC10, TIC1 and TIC3 related to the substructure of the molecules, the neighboring atoms and the connections between them. In combination with the diagram, we find that the substructure of the molecule plays a dominant role in the prediction of the model;

(2) GATS7c and Si relates to electronic structure of the molecule.

(3) AATS5v, AATS6v and BCUTv-1l features are volume-related;

(4) ATS1m relates to mass (molecular weight);

In this research, we observe the same trends as in the feature importance as one class classification paper [27], the substructure of the molecule seems to play the most important role, with the electronic characteristics, shape and molecular weight coming next. Although, these descriptors are not directly related to some understandable molecular features.

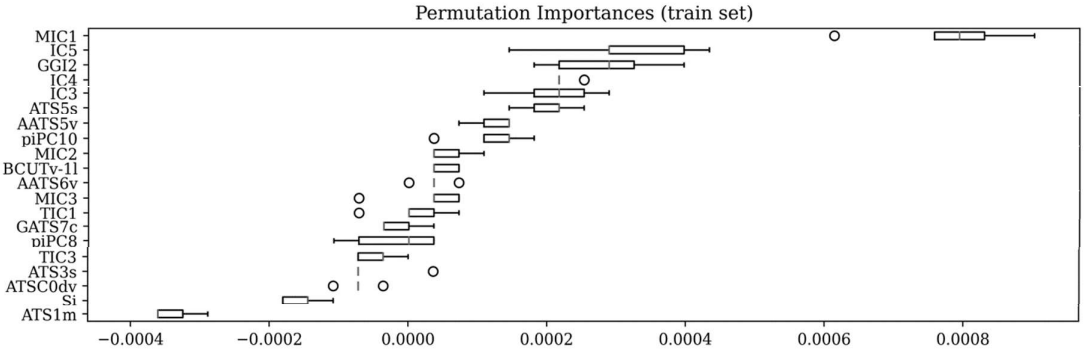


Figure 5.6: Feature importance of PU learning algorithms using random forests

Chapter 6

Conclusion

6.1 Conclusion

We aimed to predict whether new $\pi-\pi$ co-crystals could be synthesised from two different PAH molecules in this study. However, the current study faces potential challenges due to the significant imbalance in the data set and the unreliable nature of the negative samples.

To address the current problem, we used three different workflows to achieve this goal. First, we attempted to balance the data set using a data pre-processing approach that oversampled the negative data and undersampled the positive data. However, this approach could not achieve our initial goal until more reliable negative samples were available for repeatable experiments without bias sampling. We then proposed a PU learning approach that effectively uses positive and unlabelled data to avoid the problem of unreliable data. Due to the lack of real negative data, we used recall as an evaluation criterion and compared it to a third workflow of one class classification algorithms based on Isolation forests, in addition to trying to evaluate the ability of the model to find potential positive samples. In the end, we selected the PU learning algorithm as our final prediction model. Using the feature importance description in PU learning, we conclude that the key factor determining the synthesis of $\pi-\pi$ co-crystals

is the substructure of the molecule.

6.2 Future Work

In the future, the research can be enhanced in the following directions:

- Extending the $\pi - \pi$ co-crystal prediction to the prediction of the entire co-crystal dataset
- To propose better methods for model evaluation
- To perform chemical experiments on the predictions to improve the accuracy of the model

Chapter 7

BCS Project Criteria and Self-Reflection

7.1 Practical and Analytical Skills

The ability to apply the practical and analytical skills acquired during the degree programme:

The project is interdisciplinary and combines elements of data analysis and chemistry, some of which correspond to this research in the degree programme:

1) Data analysis skills: the research involves co-crystal data collection, cleaning and standardisation, and requires an understanding of the imbalances and unreliability of current co-crystal predictions. This competency corresponds to the data science component of the degree programme;

2) Programming competency: the project requires programming competency in several data sciences related programming languages. Although there is no Python language study in the degree programme, this is facilitated by the Java programming course in the degree programme, as modern programming languages are very close to each other;

3) Machine learning: The research partly addressed the challenges posed by the current data and successfully applied methods such as PU learning algorithms, one class classification algorithms, putting into practice the content of the degree course;

4) Problem collection and solving skills: The research proposed three different workflows. The methods used at the beginning of the project failed for a while, but thanks to the problem collection skills acquired during the university, by reading the literatures.

5) Communication skills: the project was cross-disciplinary and required communication with chemistry stakeholders, and it was through effective long-term communication that we were able to understand the problems and potential solutions for the negative samples;

6) Project organisation and planning skills: the study could not have been carried out without organised planning skills. The time schedule in the project plan, although slightly altered by the content of the current study, was generally in line with the plan, which corresponds to the project management aspect of the degree program.

7.2 Innovation and Creativity

Although PU learning-based methods have been used for many years, this study presents the first application of PU learning algorithms to the field of co-crystal prediction and, at some level, has the potential to motivate more researchers to focus on current problems and solutions. Furthermore, based on the existing two-step PU learning algorithm, we are considering modifying our approach to obtaining reliable negative samples and further extending our current dataset to

include more crystal data. In summary, the results of our current study are exciting as we seek to demonstrate that we can make a difference even when the data is inherently flawed.

7.3 Problem Solving and Evaluation

Synthesise information, ideas and practices to deliver a high quality solution and evaluate that solution:

As mentioned in the introductory chapter, after we faced the challenges posed by data deficiencies, we addressed the issues of data imbalance and unreliable negative data using PU learning algorithms and evaluated the solution in sections 4 and 5. Although we are currently only able to evaluate by the recall, in the future, we will be able to effectively evaluate all models by the F_1 approximation mentioned in the text. We are also looking forward to carrying out chemical experiments with the current results to refine further and enhance the current model.

7.4 Realistic Needs

Currently, our research is limited to $\pi - \pi$ co-crystals, and we hope to gradually extend the PU learning algorithm to the full CSD dataset, which will help us to discover more co-crystals with practical applications in the field of pharmacology and materials.

7.5 Self-management

The actual timeline for the project is shown in Figure 7.1. It has changed from the initial project plan, mainly because the data pre-processing methods we mentioned in the project plan were not feasible after learning that the negative data was unreliable. So we spent a week looking for other viable methods and eventually we changed our plan to move to the PU learning method. In addition, regular meetings with the project supervisor helped to check the current progress, which was extremely beneficial for the smooth running of the project. Overall, we have largely met and exceeded our original objectives in terms of time management.

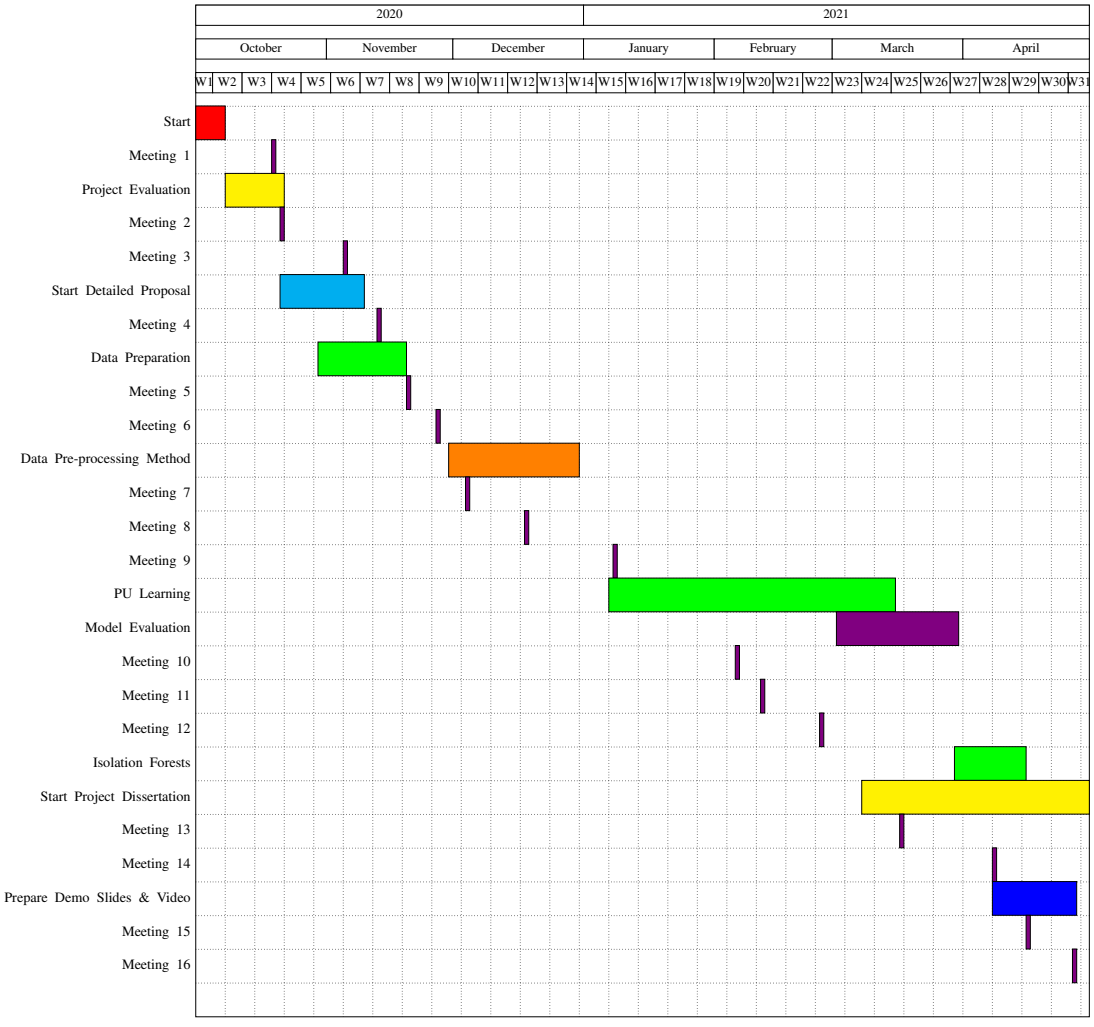


Figure 7.1: The Actual Timetable for the Project

7.6 Self-evaluation

In this section, we will reflect critically on the project as appropriate. Firstly, we need to reflect that in the initial planning of the project, we did not note the key elements of the unreliability of the negative dataset, which led to a significant amount of time being consumed in the early stages of the project. This problem was mainly due to a lack of understanding of some key concepts in the cross-discipline and should be eliminated in the future. However, it is normal for any project to be influenced in some way that causes a change in direction and it is worth concluding that we did not receive such a negative influence that the project was severely delayed, but rather we actively sought a solution to the problem. In terms of model evaluation, we are still not doing a good job. Although we have found an alternative to the F_1 evaluation method, the new method has not been implemented perfectly due to certain factors. In order to reduce the current embarrassment of judging models by recall alone, we evaluated the ability of the model to find hidden positives. Although this method is somewhat comparable, it still does not fully address the underlying problem. However, in the future, we will continue this research. We hope to find a more suitable evaluation method to compare the performance of the models and perform hyperparameter tuning.

References

[1] Claudia Draxl and Matthias Scheffler. Big data-driven materials science and its fair data infrastructure. *Handbook of Materials Modeling: Methods: Theory and Modeling*, pages 49–73, 2020.

[2] Colin R Groom, Ian J Bruno, Matthew P Lightfoot, and Suzanna C Ward. The cambridge structural database. *Acta Crystallographica Section B: Structural Science, Crystal Engineering and Materials*, 72(2):171–179, 2016.

[3] Magali B Hickey, Matthew L Peterson, Lisa A Scoppettuolo, Sherry L Morrisette, Anna Vetter, Hector Guzmán, Julius F Remenar, Zhong Zhang, Mark D Tawa, Sean Haley, et al. Performance comparison of a co-crystal of carbamazepine with marketed product. *European journal of pharmaceutics and biopharmaceutics*, 67(1):112–119, 2007.

[4] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[5] Shyam Karki, Tomislav Friščić, Laszlo Fabian, Peter R Laity, Graeme M Day, and William Jones. Improving mechanical properties of crystalline solids by cocrystal formation: new compressible forms of paracetamol. *Advanced materials*, 21(38-39):3905–3909, 2009.

[6] Srinivasulu Aitipamula, Rahul Banerjee, Arvind K Bansal, Kumar Biradha, Miranda L Cheney, Angshuman Roy Choudhury, Gautam R Desiraju, Amol G Dikundwar, Ritesh Dubey, Nagakiran Duggirala, et al. Poly-

morphs, salts, and cocrystals: what's in a name? *Crystal growth & design*, 12(5):2147–2152, 2012.

[7] Louise N Dawe, Tayel A AlHujran, Huu-Anh Tran, Jason I Mercer, Edward A Jackson, Lawrence T Scott, and Paris E Georghiou. Corannulene and its penta-tert-butyl derivative co-crystallize 1: 1 with pristine c 60-fullerene. *Chemical Communications*, 48(45):5563–5565, 2012.

[8] Jiazhen Cai, Xuan Chu, Kun Xu, Hongbo Li, and Jing Wei. Machine learning-driven new material discovery. *Nanoscale Advances*, 2(8):3115–3130, 2020.

[9] Weigang Zhu, Renhui Zheng, Xiaolong Fu, Hongbing Fu, Qiang Shi, Yonggang Zhen, Huanli Dong, and Wenping Hu. Revealing the charge-transfer interactions in self-assembled organic cocrystals: two-dimensional photonic applications. *Angewandte Chemie*, 127(23):6889–6893, 2015.

[10] Sang Kyu Park, Shinto Varghese, Jong H Kim, Seong-Jun Yoon, Oh Kyu Kwon, Byeong-Kwan An, Johannes Gierschner, and Soo Young Park. Tailor-made highly luminescent and ambipolar transporting organic mixed stacked charge-transfer crystals: an isometric donor–acceptor approach. *Journal of the American Chemical Society*, 135(12):4757–4764, 2013.

[11] Yinjuan Huang, Zongrui Wang, Zhong Chen, and Qichun Zhang. Organic cocrystals: Beyond electrical conductivities and field-effect transistors (fets). *Angewandte Chemie International Edition*, 58(29):9696–9711, 2019.

[12] Jonathan Schmidt, Mário RG Marques, Silvana Botti, and Miguel AL Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36, 2019.

[13] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[14] Amber L Thompson. Chemical crystallography: when are 'bad data''good data'? *Crystallography Reviews*, 25(1):3–53, 2019.

[15] Yang-Hui Luo and Bai-Wang Sun. Pharmaceutical co-crystals of pyrazinecarboxamide (pza) with various carboxylic acids: crystallography, hirshfeld surfaces, and dissolution study. *Crystal growth & design*, 13(5):2098–2106, 2013.

[16] Hannes Kulla, Sebastian Greiser, Sigrid Benemann, Klaus Rademann, and Franziska Emmerling. In situ investigation of a self-accelerated cocrystal formation by grinding pyrazinamide with oxalic acid. *Molecules*, 21(7):917, 2016.

[17] Dingyan Wang, Zeen Yang, Bingqing Zhu, Xuefeng Mei, and Xiaomin Luo. Machine-learning-guided cocrystal prediction based on large data base. *Crystal Growth & Design*, 20(10):6610–6621, 2020.

[18] Colin R Groom, Ian J Bruno, Matthew P Lightfoot, and Suzanna C Ward. The cambridge structural database. *Acta Crystallographica Section B: Structural Science, Crystal Engineering and Materials*, 72(2):171–179, 2016.

[19] Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

[20] Hirotomo Moriwaki, Yu-Shi Tian, Norihito Kawashita, and Tatsuya Takagi. Mordred: a molecular descriptor calculator. *Journal of cheminformatics*, 10(1):4, 2018.

[21] Bing Liu, Wee Sun Lee, Philip S Yu, and Xiaoli Li. Partially supervised classification of text documents. In *ICML*, volume 2, pages 387–394, 2002.

[22] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[23] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[24] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[25] Jerome GP Wicker, Lorraine M Crowley, Oliver Robshaw, Edmund J Little, Stephen P Stokes, Richard I Cooper, and Simon E Lawrence. Will they co-crystallize? *CrystEngComm*, 19(36):5336–5340, 2017.

[26] Jan-Joris Devogelaer, Hugo Meekes, Paul Tinnemans, Elias Vlieg, and Rene De Gelder. Co-crystal prediction by artificial neural networks. *Angewandte Chemie*, 132(48):21895–21902, 2020.

[27] Aikaterini Vriza, Angelos B Canaj, Rebecca Vismara, Laurence J Kershaw Cook, Troy D Manning, Michael W Gaultois, Peter A Wood, Vitaliy Kurlin, Neil Berry, Matthew S Dyer, et al. One class classification as a practical approach for accelerating π–π co-crystal discovery. *Chemical Science*, 12(5):1702–1719, 2021.

[28] Bing Liu, Wee Sun Lee, Philip S Yu, and Xiaoli Li. Partially supervised classification of text documents. In *ICML*, volume 2, pages 387–394, 2002.

[29] Wei Lan, Jianxin Wang, Min Li, Jin Liu, Yaohang Li, Fang-Xiang Wu, and Yi Pan. Predicting drug–target interaction using positive-unlabeled learning. *Neurocomputing*, 206:50–57, 2016.

[30] Nathan C Frey, Jin Wang, Gabriel Ivan Vega Bellido, Babak Anasori, Yury Gogotsi, and Vivek B Shenoy. Prediction of synthesis of 2d metal carbides and nitrides (mxenes) and their precursors with positive and unlabeled machine learning. *ACS nano*, 13(3):3031–3041, 2019.

[31] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020.

[32] Bing Liu. Pu learning - learning from positive and unlabeled examples.

[33] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[34] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[35] Roberto Todeschini and Viviana Consonni. *Molecular descriptors for chemoinformatics: volume I: alphabetical listing/volume II: appendices, references*, volume 41. John Wiley & Sons, 2009.

[36] Ren Wei and Kong Dexin. Dingliang gouxiao guanxi yanjiuzhong fenzimiaoshufu de xiangguanxing [relevance of molecular descriptors in quantitative conformational relationship studies]. *Computers and Applied Chemistry*, 26(11):1455–1458, 2009.

[37] Rajarshi Guha and Egon Willighagen. A survey of quantitative descriptions of molecular structure. *Current topics in medicinal chemistry*, 12(18):1946–1956, 2012.

[38] Balakumar Chandrasekaran, Sara Nidal Abed, Omar Al-Attraqchi, Kaushik Kuche, and Rakesh K Tekade. Computer-aided prediction of pharmacokinetic (admet) properties. In *Dosage Form Design Parameters*, pages 731–755. Elsevier, 2018.

[39] László Fábián. Cambridge structural database analysis of molecular complementarity in cocrystals. *Crystal Growth and Design*, 9(3):1436–1443, 2009.

[40] S Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[41] Mariana Belgiu and Lucian Drăguţ. Random forest in remote sensing: A review of applications and future directions. *ISPRS journal of photogrammetry and remote sensing*, 114:24–31, 2016.

[42] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[43] Mohammed TH Elbatta and Wesam M Ashour. A dynamic method for discovering density varied clusters. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(1), 2013.

[44] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455, 2003.

[45] Xiao-Li Li, Bing Liu, and See Kiong Ng. Negative training data can be harmful to text classification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 218–228, 2010.

[46] S. Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PLOS ONE*, 12:e0177678, 06 2017.

Appendix A

Features List

This section explains in detail all the features that we have used in this work, please note that the description of the feature is based on the *Mordred Descriptor Documentation* [20]: `https://mordred-descriptor.github.io/documentation/master/descriptors.html`

Table A.1: Features List

Descriptor	Description
MIC1	1-ordered modified information content
IC5	5-ordered neighborhood information content
GGI2	2-ordered raw topological charge
IC4	4-ordered neighborhood information content
IC3	3-ordered neighborhood information content
ATS5V	moreau-broto autocorrelation of lag 5 weighted by vdw volume
AATS5V	averaged moreau-broto autocorrelation of lag 5 weighted by vdw volume
piPC10	10-ordered pi-path count (log scale)
MIC2	2-ordered modified information content
BCUTv-1l	first lowest eigenvalue of Burden matrix weighted by vdw volume
AATS6v	averaged moreau-broto autocorrelation of lag 6 weighted by vdw volume
MIC3	3-ordered modified information content
TIC1	1-ordered neighborhood total information content
GATS7c	geary coefficient of lag 7 weighted by gasteiger charge
piPC8	8-ordered pi-path count (log scale)
TIC3	3-ordered neighborhood total information content
ATS3s	moreau-broto autocorrelation of lag 3 weighted by intrinsic state
ATSC0dv	centered moreau-broto autocorrelation of lag 0 weighted by valence electrons
Si	sum of constitutional weighted by ionization potential
ATS1m	moreau-broto autocorrelation of lag 1 weighted by mass

Appendix B

Source Code

This section shows part of the code we used to construct the PU study, for the full code please visit Github repository: `https://github.com/Charlie059/Co-Crystal-Prediction-Master`

```
# -*- coding: utf-8 -*-
"""PU_Learning.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1ViRxU5ElzyDkSGczWpaIXjVza3zRrYY9

# 1. Importing the libraries and datasets
"""

# Import the basic libraries
from sklearn import datasets, metrics
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from scipy.spatial.distance import squareform
from matplotlib import cm
import itertools
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from matplotlib.colors import ListedColormap, LinearSegmentedColormap

from google.colab import drive
drive.mount('/content/drive')

"""## 1.1 Import the Postive Data"""

# Import the dataset which contains all the molecules that constitute the first coformer of the
↪  extracted from CSD database and their dragon descriptors calculated after filtering with
↪  Pipeline Pilot

dataset1=pd.read_csv('/content/drive/My Drive/Colab
↪  Notebooks/cocrystal_design/Dataset/dataset1.csv')
dataset2=pd.read_csv('/content/drive/My Drive/Colab
↪  Notebooks/cocrystal_design/Dataset/dataset2.csv')
print(dataset1.shape)
print(dataset2.shape)

# Clean from combinations that one molecules has all nan descriptors Nov 26
drop_id =
↪  np.concatenate([dataset1.id[dataset1.ABC=='n'].values,dataset2.id[dataset2.ABC=='n'].values])
len(drop_id)
```

```
dataset1 = dataset1[~dataset1.id.isin(drop_id)]
dataset2 = dataset2[~dataset2.id.isin(drop_id)]
print(dataset1.shape)
print(dataset2.shape)

dataset1.head(2)

"""## 1.2 Import the Negative Data (Do not need now)"""

# Import the dataset which contains all the molecules that constitute the first coformer of the
↪  extracted from CSD database and their dragon descriptors calculated after filtering with
↪  Pipeline Pilot
negatives1 = pd.read_csv('/content/drive/My Drive/Colab
↪  Notebooks/cocrystal_design/Dataset/negatives/negative1.csv')
negatives2 = pd.read_csv('/content/drive/My Drive/Colab
↪  Notebooks/cocrystal_design/Dataset/negatives/negative2.csv')
print(negatives1.shape)
print(negatives2.shape)

# Set the columns of negative 1 and 2
numerical_cols = dataset1.columns[:]
negatives1.columns = numerical_cols
negatives2.columns = numerical_cols
negatives1.head(2)

negatives1

"""## 1.3 Import the Unlabelled Data - ZINC15"""

# Unlabelled data

# Generate the unknown dataset (unlabelled), this might take some minutes

# Read the Zinc dataset of purcasable molecules with their dragon descriptors
unlabeled = pd.read_csv('/content/drive/MyDrive/Colab
↪  Notebooks/cocrystal_design/Dataset/zinc15_dataset.csv')
val = unlabeled['Identifier'].values
length = len(val)

pairs = [[val[i],val[j]] for i in range(length) for j in range(length) if i!=j ]

# Remove the duplicate structures
no_dups = []
for pair in pairs:
  if not any(all(i in p for i in pair) for p in no_dups):
    no_dups.append(pair)

pairs = pd.DataFrame(no_dups)
keys = unlabeled['Identifier'].values
values = unlabeled.iloc[:, 2:].values
d = {key:value for key, value in zip(keys, values)}
print(len(d['ZINC000000354958']))

mol1_data= list()
for mol1 in pairs[0]:
    mol1_data.append(d[mol1])

mol1_data = pd.DataFrame(mol1_data, columns = unlabeled.iloc[:, 2:].columns.values)

mol2_data= list()

for mol2 in pairs[1]:
    mol2_data.append(d[mol2])
mol2_data = pd.DataFrame(mol2_data, columns= unlabeled.iloc[:, 2:].columns.values)

mol1_data

"""# 2. Feature selection
I use both Postive and Unlabelled for Fearture Selection
"""
```

```
# The two datasets are concateanted for identifing the highly correlated descriptors and remove
↪  them
df1_w=dataset1.iloc[:,2:]
df2_w=dataset2.iloc[:,2:]


#Unlabelled doesn't need to use iloc[:,2:] because we have done this before
df_unlabelled1_w = mol1_data
df_unlabelled2_w = mol2_data

data_postive = pd.concat([df1_w, df2_w])
data_unlabelled = pd.concat([df_unlabelled1_w, df_unlabelled2_w])

# data = pd.concat([data_postive, data_unlabelled])
data = data_postive

data_ = data.drop_duplicates(keep='first')

# Drop the highly linearly correlated features among the datasets
# Create correlation matrix
corr_matrix1 = data_.corr().abs()

# Select upper triangle of correlation matrix
upper1 = corr_matrix1.where(np.triu(np.ones(corr_matrix1.shape), k=1).astype(np.bool))

# Find index of feature columns with Pearson correlation greater than 0.92
to_drop1 = [column for column in upper1.columns if any(upper1[column] > 0.7)]

# Drop the descriptos will low variance, below 0.4
drop = data_.std()[data_.std() < 0.4].index.values
to_drop2= [x for x in drop if x not in to_drop1]
drop_final= to_drop1 + to_drop2
drop_final = drop

# Remove the selected features from the datasets
df1=df1_w.drop(columns=drop_final)
df1=df1.fillna(0)
df1 = df1.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)


df2=df2_w.drop(columns=drop_final)
df2=df2.fillna(0)
df2 = df2.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)

df_unlabelled1_w=df_unlabelled1_w.drop(columns=drop_final)
df_unlabelled1_w=df_unlabelled1_w.fillna(0)
df_unlabelled1_w = df_unlabelled1_w.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)

df_unlabelled2_w=df_unlabelled2_w.drop(columns=drop_final)
df_unlabelled2_w=df_unlabelled2_w.fillna(0)
df_unlabelled2_w = df_unlabelled2_w.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)

df2

print(df_unlabelled1_w.shape)
print(df_unlabelled2_w.shape)

# # Calculate the Spearman correlations of the datasets
# df1 = pd.concat([df1, df_unlabelled1_w])# concat the first table with postive and negative
# df2 = pd.concat([df2, df_unlabelled2_w])# concat the second table with postive and negative

df1

cor = df1.corrwith(df2, axis=0, drop=False, method='spearman').abs()
corr=cor.sort_values(ascending=False)

corr

# Construct a vector w which is used to keep only the descriptors that are correlated higher than
↪  0.30 using Spearman correlation
```

```
# In this vector, 1 is on the positions of the descriptors that have correlation coeficcient >
↪  0.1,  and 0 otherwise
w = np.array(cor)
np.nan_to_num(w,0)
w[w<0.4] =0
w[w=='NaN']=0
w[w>=0.4] =1

# Multiply the two datasets with the vector w, such that the descriptors with lower correlation
↪  will become zero and removed
df1 = df1*w
df2 = df2*w

df1_1 = df1.loc[:, (df1 != 0).any(axis=0)]
df2_2 = df2.loc[:, (df2 != 0).any(axis=0)]

df1_1

# Calculate the Pearson correlations of the datasets
cort= df1_1.corrwith(df2_2, axis=0, drop=False, method='pearson').abs()
corrt=cort.sort_values(ascending=False)

# Keep only descriptors with Pearson correlations > 0.4
w = np.array(cort)
np.nan_to_num(w,0)
w[w<0.4]=0
w[w>=0.4]=1
df1_1 = df1_1*w
df2_2 = df2_2*w
df1_1 = df1_1.loc[:, (df1_1 != 0).any(axis=0)]
df2_2 = df2_2.loc[:, (df2_2 != 0).any(axis=0)]

# Print out the p-value
#\hline
# & ABCGG & Graovac-Ghorbani atom-bond connectivity\\
from scipy.stats.stats import spearmanr
from scipy.stats.stats import pearsonr

for i in df1_1.columns.values:
  print("\hline")
  print("%s"% i + " " + 'Correlation:%2f' %  pearsonr(df1_1[i].values, df2_2[i].values)[0] + "  "
  ↪  + "p-value:%2f" %  pearsonr(df1_1[i].values, df2_2[i].values)[1])

#Save to a list the important features and then from both the positives and negatives keep only
↪  these features

selected_features = df1_1.columns
selected_features

selected_features.shape

"""# 3. Normalization and Concat_bidirectional of Dataset

## 3.1  Normalization with positives and Unlabelled
"""

#get the postive data and fillna to 0
df1=dataset1.iloc[:,2:]
df1 = df1.fillna(0)
df2=dataset2.iloc[:,2:]
df2 = df2.fillna(0)
df1 = df1.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)
df2 = df2.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)

#select feature
df1 = df1[selected_features]
df2 = df2[selected_features]

print(df1.shape)
print(df2.shape)
```

```
# Replace errors with 0 on the unlabelled dataset
final_1 = pd.concat([pairs[0],mol1_data],axis=1)
final_1 = final_1.fillna(0)
final_2 = pd.concat([pairs[1],mol2_data],axis=1)
final_2 = final_2.fillna(0)

final_11 = final_1.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)
final_22 = final_2.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)

#select feature
final_11 = final_11[selected_features]
final_22 = final_22[selected_features]

print(final_11.shape)
print(final_22.shape)

# Define a dictionary where the comb identifier will be combined to the feature vector
comb=[]
for i in range(1,final_11.shape[0]+1):
  comb.append('comb%s' % i)

# Construct the labelled dataset by contatenating the feature set of each coformer in both orders
# We have a vector of 50 dimensions now
def concat_bidirectional(dataset11, dataset22):

  return pd.concat([pd.concat([dataset1['id'], dataset11, dataset22], axis=1),
  ↪  pd.concat([dataset1['id'].apply(lambda x: f"{x}_"),dataset22, dataset11], axis=1) ])

def unlabelled_concat_bidirectional(dataset11, dataset22):
  return pd.concat([pd.concat([final_11['comb'], dataset11, dataset22], axis=1),
  ↪  pd.concat([final_11['comb'].apply(lambda x: f"{x}_"),dataset22, dataset11], axis=1) ])

# Standarize the positive data

X_scaler = MinMaxScaler()

df_concat = pd.concat([df1, df2])
df_concat_unlabelled = pd.concat([final_11, final_22])

df_concat = pd.concat([df_concat, df_concat_unlabelled])

df_concat = df_concat.drop_duplicates(keep='first')
numerical_cols = df_concat.columns[:]
df_scaled = pd.DataFrame(X_scaler.fit(df_concat), columns=numerical_cols, index=df_concat.index)

numerical_cols = df2.columns[:]
df1_scaled =  pd.DataFrame(X_scaler.transform(df1[numerical_cols]), columns=numerical_cols,
↪  index=df1.index)
df2_scaled = pd.DataFrame(X_scaler.transform(df2[numerical_cols]), columns=numerical_cols,
↪  index=df2.index)

#2/1 edit
df2_scaled_ = df2_scaled.add_suffix('_')


# Standarize the unlabeled data

final_11 = pd.concat([pd.DataFrame(comb, columns=['comb']), pd.DataFrame(final_11.values,
↪  columns=final_11.columns)],axis=1)
#skip the comb
unlabeled_1_scaled= final_11.iloc[:,1:]
final_1_scaled = pd.DataFrame(X_scaler.transform(unlabeled_1_scaled),columns=numerical_cols,
↪  index=final_11.index)

final_22 = pd.concat([pd.DataFrame(comb, columns=['comb']), pd.DataFrame(final_22.values,
↪  columns=final_22.columns)],axis=1)
#skip the comb
unlabeled_2_scaled= final_22.iloc[:,1:]
final_2_scaled = pd.DataFrame(X_scaler.transform(unlabeled_2_scaled),columns=numerical_cols,
↪  index=final_11.index)
```

```
#2/1 edit
final_2_scaled_ = final_2_scaled.add_suffix('_')


# Final bidirectional concatenated dataset, after feature selection and scaling
positive_data = concat_bidirectional(df1_scaled,df2_scaled)
positive_data_ = concat_bidirectional(df1_scaled,df2_scaled_)
unlabelled_data = unlabelled_concat_bidirectional(final_1_scaled,final_2_scaled)
unlabelled_data_ = unlabelled_concat_bidirectional(final_1_scaled,final_2_scaled_)

# Assing the label = 1 for all the positive data
positive_data['label']=1
positive_data_['label']=1

# Assing the label = 0 for all the unlabelled data
unlabelled_data['label']=0
unlabelled_data_['label']=0

# Reindex
positive_data = positive_data.reset_index(drop=True)
unlabelled_data = unlabelled_data.reset_index(drop=True)

positive_data_ = positive_data_.reset_index(drop=True)
unlabelled_data_ = unlabelled_data_.reset_index(drop=True)

positive_data_

unlabelled_data_

"""## 3.2 Orginal ZINC"""

# Define a dictionary where the comb identifier will be combined to the feature vector
comb=[]
for i in range(1,final_11.shape[0]+1):
  comb.append('comb%s' % i)
# Replace errors with 0 on the unlabelled dataset
final_1_org = pd.concat([pairs[0],mol1_data],axis=1)
final_1_org = final_1_org.fillna(0)
final_2_org = pd.concat([pairs[1],mol2_data],axis=1)
final_2_org = final_2_org.fillna(0)

final_11_org = pd.concat([pd.DataFrame(comb, columns=['comb']), pd.DataFrame(final_1_org.values,
↪  columns=final_1_org.columns)],axis=1)
final_22_org = pd.concat([pd.DataFrame(comb, columns=['comb']), pd.DataFrame(final_2_org.values,
↪  columns=final_2_org.columns)],axis=1)

a = pd.concat([final_11_org['comb'], final_1_org, final_2_org], axis=1)
b = pd.concat([final_11_org['comb'].apply(lambda x: f"{x}_"),final_2_org, final_1_org], axis=1)

a = a[['comb',0,1]]
b = b[['comb',1,0]]

a = a.rename(columns={0:'mol1',1:'mol2'})

b = b.rename(columns={1:'mol1',0:'mol2'})

unlabelled_org = pd.concat([a,b])

unlabelled_org = unlabelled_org.reset_index(drop=True)
unlabelled_org

unlabelled_org_w = pd.merge(unlabelled_org,unlabelled_data_,on='comb')
unlabelled_org_w = unlabelled_org_w.drop('label', axis=1)
unlabelled_org_w

"""## 3.3 Remove duplicates from the unlabelled dataset"""

columns = unlabelled_data.columns
columns = columns.delete(0)
```

```
columns = columns.delete(-1)
columns

#drop id and label
positive_data_w = positive_data_.drop('id', axis=1)
positive_data_w = positive_data_w.drop('label', axis=1)

unlabelled_data_w = unlabelled_data_.drop('comb',axis=1)
unlabelled_data_w = unlabelled_data_w.drop('label', axis=1)

merge_df = unlabelled_org_w.merge(positive_data_ , how='left', indicator=True)
merge_df = merge_df[merge_df['_merge'] == 'both']
unlabelled_dul = merge_df.drop('_merge', axis=1)
# unlabelled_dul.columns = columns
unlabelled_dul = unlabelled_dul.drop_duplicates(keep='first')
unlabelled_dul = unlabelled_dul.iloc[:-10,:-1]


unlabelled_dul = unlabelled_dul[unlabelled_dul.index % 2 == 0]
cols = list(unlabelled_dul.columns)
cols = [cols[-1]] + cols[:-1]
unlabelled_dul = unlabelled_dul[cols]

# these unlablled are duplicated in the postive
unlabelled_dul = pd.read_csv("/content/drive/MyDrive/Colab
↪  Notebooks/cocrystal_design/data/unlabelled_dul.csv")

unlabelled_dul

"""## 3.4 Normalization with negatives"""

#get the negative data and fillna to 0

neg1= negatives1.iloc[:,2:]
neg1 = neg1.fillna(0)
neg2= negatives2.iloc[:,2:]
neg2 = neg2.fillna(0)
neg1 = neg1.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)
neg2 = neg2.apply(lambda x: pd.to_numeric(x, errors='coerce')).fillna(0)


#select feature
neg1 = neg1[selected_features]
neg2 = neg2[selected_features]

print(neg1.shape)
print(neg2.shape)

#Scale the negatives based on the potitives
negative_1_scaled = pd.DataFrame(X_scaler.transform(neg1), columns=selected_features)
negative_2_scaled = pd.DataFrame(X_scaler.transform(neg2), columns=selected_features)
neg=pd.concat([negative_1_scaled, negative_2_scaled], axis =1)

def neg_concat_bidirectional(dataset11, dataset22):
  return pd.concat([pd.concat([negatives1['id'], dataset11, dataset22], axis=1),
  ↪  pd.concat([negatives1['id'].apply(lambda x: f"{x}_"),dataset22, dataset11], axis=1) ])

negatives_data = neg_concat_bidirectional(negative_1_scaled,negative_2_scaled)

# Assing the label = 0 for all the negative data
negatives_data['label'] = 0
negatives_data

"""# 4. PU Learning _ based on the SPY"""

positive_data_w = positive_data.drop('id', axis=1)
unlabelled_data_w = unlabelled_data.drop('comb',axis=1)

# merge the postive and unlabelled
X = pd.concat([positive_data_w,unlabelled_data_w])
```

```
X = X.reset_index(drop=True)
y = X['label']

# remove labels
# Print the data we inputed ### CHECKPOINT1
X = X.drop('label',axis = 1)
X

X.to_csv("/content/drive/MyDrive/Colab Notebooks/cocrystal_design/data/X_to_PU_Performance.csv")
y.to_csv("/content/drive/MyDrive/Colab Notebooks/cocrystal_design/data/y_to_PU_Performance.csv")

#import some necessary packages
import sys
sys.path.append('/content/drive/MyDrive/Colab Notebooks/cocrystal_design/utils/')
from pu_learning import spies
from baggingPU import BaggingClassifierPU

"""## 4.1 UMAP"""

import umap
reducer = umap.UMAP(random_state=42,n_components=3)
embedding = reducer.fit_transform(X)

import plotly.graph_objects as go
def plot_2d(component1, component2):

    fig = go.Figure(data=go.Scatter(
        x = component1,
        y = component2,
        mode='markers',
        marker=dict(
            size=10,
            color=y, #set color equal to a variable
            showscale=False,
            line_width=1
        )
    ))
    fig.update_layout(margin=dict( l=10,r=10,b=10,t=10),width=1000,height=600)
    fig.layout.template = 'plotly_dark'

    fig.show()

plot_2d(reducer.embedding_[:, 0],reducer.embedding_[:, 1])

"""## 4.2 PCA"""

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
df_scaled_PCA=pca.fit_transform(X)

df_scaled_PCA= pd.DataFrame(df_scaled_PCA)
df_scaled_PCA.rename(columns={0:'PCA1',1:"PCA2"},inplace=True)
# Check the new contents of the set
print('%d positive out of %d total' % (sum(y), len(y)))

# Plot the data set, as the models will see it
plt.scatter(
    df_scaled_PCA[y==0].iloc[:,0], df_scaled_PCA[y==0].iloc[:,1],
    c='k', marker='.', linewidth=1, s=5, alpha=0.5,
    label='Unlabeled'
)
plt.scatter(
    df_scaled_PCA[y==1].iloc[:,0], df_scaled_PCA[y==1].iloc[:,1],
    c='b', marker='o', linewidth=0, s=20, alpha=0.5,
    label='Positive'
)
plt.legend()
plt.title('Data set (as seen by the classifiers)')
plt.show()
```

```
from sklearn.ensemble import RandomForestClassifier
model = spies(RandomForestClassifier(n_estimators=200,max_depth =  12, oob_score = True,
↪  random_state=13,class_weight = {0:1,1:10}, n_jobs = -1),
↪  RandomForestClassifier(n_estimators=200,max_depth =  12, oob_score = True,
↪  random_state=15,class_weight = {0:1,1:10}, n_jobs = -1))
model.fit(X, y)
y_predict = model.predict(X)
y_proba = model.predict_proba(X)

y_proba

# Visualize this approach's results
plt.scatter(
    df_scaled_PCA[y==0].iloc[:,0], df_scaled_PCA[y==0].iloc[:,1],
    c = y_proba[y==0], linewidth = 0, s = 10, alpha = 0.5,
    cmap = 'jet_r'
)
plt.colorbar(label='Scores given to unlabeled points')
plt.title(r'Using ${\tt Spies}$')
plt.show()

import plotly.graph_objects as go

fig = go.Figure()
fig.add_trace(go.Histogram(
    x=y_proba[y==0],
    xbins=dict( # bins used for histogram
        start=0,
        end=1,
        size=0.2
    )
))
fig.show()

"""## 4.4 Predict"""

X_unlabelled = X[y==0]
X_unlabelled = X_unlabelled.reset_index(drop=True)

X_unlabelled

pd.set_option('max_columns', None)

y_proba[y==0] >= 0.5

top_zinc_index = X_unlabelled[y_proba[y==0] >= 0.5].index
top_zinc_index

res_Spy = unlabelled_org.iloc[top_zinc_index,:]
# res_Spy.reset_index(drop=True, inplace=True)
confidence = pd.DataFrame(y_proba[y==0][top_zinc_index])
confidence.index = top_zinc_index
res_Spy = pd.concat([res_Spy,confidence],axis = 1)
res_Spy = res_Spy.rename(columns={0: 'confidence'})

invaild_index = res_Spy.index[res_Spy.index<=22791]
invaild_index += 22791
invaild_index

invaild_index = invaild_index.intersection(res_Spy.index)
res_Spy = res_Spy.drop(invaild_index,axis=0)


#remove some comb which CSD have
res_Spy=res_Spy[~res_Spy['comb'].isin(['comb20579', 'comb22010',
↪  'comb22195','comb22384','comb22787','comb20579_', 'comb22010_',
↪  'comb22195_','comb22384_','comb22787_'])]
res_Spy = res_Spy.sort_values('confidence',ascending=False)
res_Spy
```

```
res_Spy.to_csv("/content/drive/MyDrive/Colab Notebooks/cocrystal_design/data/result_spy.csv")

mol_occur = res_Spy.iloc[:,1:3].apply(pd.value_counts)
mol_occur = mol_occur.fillna(0)
mol_occur['mol_occur_times'] = mol_occur['mol1'] + mol_occur['mol2']
mol_occur = mol_occur.sort_values('mol_occur_times',ascending=False)
mol_occur

"""# 5. PU Learning - BaggingClassifierPU (Soon to be abandoned)"""

from sklearn.tree import DecisionTreeClassifier

bc = BaggingClassifierPU(
    DecisionTreeClassifier(),
    n_estimators = 4000,  # 1000 trees as usual
    max_samples = sum(y), # Balance the positives and unlabeled in each bag
    n_jobs = -1           # Use all cores
)
bc.fit(X, y)
oob_decision_function = bc.oob_decision_function_[:,1]

# Visualize this approach's results
plt.scatter(
    df_scaled_PCA[y==0].iloc[:,0], df_scaled_PCA[y==0].iloc[:,1],
    c = oob_decision_function[y==0], linewidth = 0, s = 10, alpha = 0.5,
    cmap = 'jet_r'
)
plt.colorbar(label='Scores given to unlabeled points')
plt.title(r'Using ${\tt BaggingClassifierPU}$')
plt.show()

fig, ax = plt.subplots(figsize=(16, 10))
ax.hist(oob_decision_function[y==0], density=False)

for rect in ax.patches:
    height = rect.get_height()
    ax.annotate(f'{int(height)}', xy=(rect.get_x()+rect.get_width()/2, height),
                xytext=(0, 5), textcoords='offset points', ha='center', va='bottom')

top_zinc_index = X_unlabelled[oob_decision_function[y==0] >= 0.7].index

res_bagging = unlabelled_org.iloc[top_zinc_index,:]

confidence = pd.DataFrame(oob_decision_function[y==0][top_zinc_index])
confidence.index = top_zinc_index
res_bagging = pd.concat([res_bagging,confidence],axis = 1)
res_bagging = res_bagging.rename(columns={0: 'confidence'})

invaild_index = res_bagging.index[res_bagging.index<=22791]
invaild_index += 22791
invaild_index = invaild_index.intersection(res_bagging.index)
res_bagging = res_bagging.drop(invaild_index,axis=0)


#remove some comb which CSD have
res_bagging=res_bagging[~res_bagging['comb'].isin(['comb20579', 'comb22010',
↪  'comb22195','comb22384','comb22787','comb20579_', 'comb22010_',
↪  'comb22195_','comb22384_','comb22787_'])]
res_bagging = res_bagging.sort_values('confidence',ascending=False)
res_bagging

unlabelled_org.iloc[top_zinc_index,1:].apply(pd.value_counts)

mol_occur = res_bagging.iloc[:,1:3].apply(pd.value_counts)
mol_occur = mol_occur.fillna(0)
mol_occur['mol_occur_times'] = mol_occur['mol1'] + mol_occur['mol2']
mol_occur = mol_occur.sort_values('mol_occur_times',ascending=False)
mol_occur

"""# 6. Plot as a real case validation
```

```
Model Testing Set
"""

negatives_data_copy = negatives_data.iloc[0:11,:]
negatives_data_copy = negatives_data_copy.iloc[:,:-1]
negatives_data_copy = negatives_data_copy.iloc[:,1:]
negatives_data_copy['label']= 0

unlabelled_dul_copy = unlabelled_data
# unlabelled_dul_copy = unlabelled_dul_copy.drop(unlabelled_dul_copy.columns[[0, 1, 2, 3]],
↪  axis=1)
idx = [20578,22009,22194,22383,22786]
unlabelled_dul_copy = unlabelled_dul_copy.iloc[idx,:]
unlabelled_dul_copy = unlabelled_dul_copy.drop('comb',axis=1)
unlabelled_dul_copy['label']= 1
unlabelled_dul_copy.columns = negatives_data_copy.columns
unlabelled_dul_copy

frames = [negatives_data_copy, unlabelled_dul_copy]
neg_unlablled_dul_df = pd.concat(frames)
neg_unlablled_dul_df.reset_index(drop=True, inplace=True)
#Predict Input
neg_unlablled_dul_df

input_y = neg_unlablled_dul_df.iloc[:,-1]
input_X = neg_unlablled_dul_df.iloc[:,:-1]

"""Model Training Set"""

unlabelled_data_copy = unlabelled_data
unlabelled_data_copy = unlabelled_data_copy[~unlabelled_data_copy['comb'].isin(['comb20579',
↪  'comb22010', 'comb22195','comb22384','comb22787','comb20579_', 'comb22010_',
↪  'comb22195_','comb22384_','comb22787_'])]
unlabelled_data_copy

positive_data_copy = positive_data
positive_data_copy = positive_data_copy[~positive_data_copy['id'].isin(['HOSJAA', 'UWOBAL',
↪  'MULFOO','HULLAB','RAXYAS','HOSJAA_', 'UWOBAL_', 'MULFOO_','HULLAB_','RAXYAS_'])]
positive_data_copy

positive_data_w_copy = positive_data_copy.drop('id', axis=1)
unlabelled_data_w_copy = unlabelled_data_copy.drop('comb',axis=1)

# merge the postive and unlabelled
X_copy = pd.concat([positive_data_w_copy,unlabelled_data_w_copy])
X_copy = X_copy.reset_index(drop=True)
y_copy = X_copy['label']

# remove the labels
X_copy = X_copy.drop('label',axis = 1)
X_copy

np.random.seed(42)
# Shuffle dataset
permut = np.random.permutation(len(y_copy))
X_copy = X_copy.reindex(permut)
y_copy = y_copy.reindex(permut)

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.mixture import GaussianMixture
from sklearn.tree import DecisionTreeClassifier
# from pyod.models.ocsvm import OCSVM
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from xgboost import XGBClassifier
import sys
sys.path.append('/content/drive/MyDrive/Colab Notebooks/cocrystal_design/utils/')
```

```
from pu_learning import spies
from baggingPU import BaggingClassifierPU
#model_copy =
↪  spies(GaussianMixture(n_components=6,random_state=0),RandomForestClassifier(n_estimators=200,max_depth
↪  =  6, oob_score = True,  class_weight = {0:1,1:6}, n_jobs = -1)) # increse the second model to
↪  280 could increse the middle prediction rate
#model_copy = spies(MLPClassifier(alpha=0.0001,
↪  max_iter=20),RandomForestClassifier(n_estimators=100,max_depth =  7, oob_score = True,
↪  class_weight = {0:1,1:7}, n_jobs = -1))
#model_copy = spies(MLPClassifier(alpha=0.001, max_iter=100,verbose =
↪  True),MLPClassifier(alpha=0.0001, max_iter=10000,learning_rate = "adaptive",verbose =
↪  True,tol=1e-6))
#model_copy = spies(GaussianNB(),MLPClassifier(hidden_layer_sizes=(100, 200, 200 ,200,
↪  100),alpha=0.0001, max_iter=1000,learning_rate = "adaptive",verbose = True,tol=1e-5)) #
↪  increse the second model to 280 could increse the middle prediction rate
#model_copy =
↪  spies(GaussianMixture(n_components=3,random_state=0),RandomForestClassifier(n_estimators=200,max_depth
↪  =  7, oob_score = True,  class_weight = {0:1,1:1}, n_jobs = -1))
model_copy = spies(RandomForestClassifier(n_estimators=200,max_depth =  None, oob_score = True,
↪  class_weight = 'balanced', n_jobs =
↪  -1,random_state=0),RandomForestClassifier(n_estimators=200,max_depth =  5, oob_score = True,
↪  class_weight = 'balanced_subsample', n_jobs = -1,random_state=0))
model_copy.fit(X_copy, y_copy,spie_rate=0.1, spie_tolerance=0.1)

y_predict_neg_unlablled_dul_df_SPY = model_copy.predict(input_X)
y_proba_neg_unlablled_dul_df_SPY = model_copy.predict_proba(input_X)

unique, counts = np.unique(model_copy.predict(positive_data_w_copy.drop('label',axis = 1)),
↪  return_counts=True)
dict(zip(unique, counts))

rank_idx = np.argsort(-1*y_proba_neg_unlablled_dul_df_SPY)
rank_idx

y_proba_neg_unlablled_dul_df_SPY[::-1].sort()
y_proba_neg_unlablled_dul_df_SPY

rank_idx

input_y = input_y.reindex(rank_idx)
input_X = input_X.reindex(rank_idx)

import numpy as np
from matplotlib import pyplot as plt
plt.style.use('grayscale')
# plt.grid(b=None)
plt.rcParams['font.family'] = 'serif'
plt.rcParams['font.serif'] = ['Times New Roman'] + plt.rcParams['font.serif']
group = input_y
cdict = {1: 'green', 0: 'red'}
mdict = {1: "D", 0: "^" }
scatter_x = np.arange(1, 17, 1)

fig, ax = plt.subplots()
for g in np.unique(group):
    if(g == 0):
      label_name = "Failed Co-Crystal Synthesis"
    else:
      label_name = "Successful Co-Crystal Synthesis"
    ix = np.where(group == g)
    ax.scatter(scatter_x[ix], y_proba_neg_unlablled_dul_df_SPY[ix], c = cdict[g], label =
    ↪  label_name, s = 30, marker=mdict[g])
ax.legend()
plt.rcParams["figure.figsize"] = (10,6)
plt.grid(linestyle='-.',axis="y")
plt.xlabel('Co-Former Rank', fontsize=12)
plt.ylabel('Co-Former Predict Probability by SPY', fontsize=12)
plt.savefig(fname='Co-Former_rank.png',dpi=400, bbox_inches = 'tight')
plt.show()
```

```
# Notice that 4,6,8,9,10 is all contains C60

"""## 6.1 Model Preidct with new parameters"""

y_predict = model_copy.predict(X)
y_proba = model_copy.predict_proba(X)

X_unlabelled = X[y==0]
X_unlabelled = X_unlabelled.reset_index(drop=True)

top_zinc_index = X_unlabelled[y_proba[y==0] >= 0.5].index
top_zinc_index

res_Spy = unlabelled_org.iloc[top_zinc_index,:]
# res_Spy.reset_index(drop=True, inplace=True)
confidence = pd.DataFrame(y_proba[y==0][top_zinc_index])
confidence.index = top_zinc_index
res_Spy = pd.concat([res_Spy,confidence],axis = 1)
res_Spy = res_Spy.rename(columns={0: 'confidence'})

invaild_index = res_Spy.index[res_Spy.index<=22791]
invaild_index += 22791
invaild_index

#Print the molecular combination with the highest predicted probability ### CHECKPOINT2
invaild_index = invaild_index.intersection(res_Spy.index)
res_Spy = res_Spy.drop(invaild_index,axis=0)


#remove some comb which CSD have
res_Spy=res_Spy[~res_Spy['comb'].isin(['comb20579', 'comb22010',
↪  'comb22195','comb22384','comb22787','comb20579_', 'comb22010_',
↪  'comb22195_','comb22384_','comb22787_'])]
res_Spy = res_Spy.sort_values('confidence',ascending=False)
res_Spy

# Print the order of the most frequently occurring molecules
mol_occur = res_Spy.iloc[:,1:3].apply(pd.value_counts)
mol_occur = mol_occur.fillna(0)
mol_occur['mol_occur_times'] = mol_occur['mol1'] + mol_occur['mol2']
mol_occur = mol_occur.sort_values('mol_occur_times',ascending=False)
mol_occur

import seaborn as sns
from sklearn.metrics import roc_curve, confusion_matrix, auc



#model predicts probabilities of positive class
p = model_copy.predict_proba(X_copy)



#FIGURE
plt.figure(figsize=[15,4])


#2 -- Distributions of Predicted Probabilities of both classes
df = pd.DataFrame({'probPos':p, 'target': y_copy})
plt.hist(df[df.target==1].probPos, log = True, bins=25,alpha=.5, color='green',  label='Pos')
plt.hist(df[df.target==0].probPos, log = True, bins=25,alpha=.5, color='red', label='Neg')
plt.axvline(.5, color='blue', linestyle='--', label='Boundary')
plt.xlim([0,1])
plt.title('Distributions of Predictions', size=15)
plt.xlabel('Positive Probability (predicted)', size=13)
plt.ylabel('Samples (normalized scale)', size=13)
plt.legend(loc="upper right")
df
```

Appendix C

Full Table of Most Popular Molecules in the High Scoring Pair

ZINC 15 Index Search:
`http://zinc15.docking.org/substances/`

Table C.1: Most Popular Molecules in the High Scoring Pair in PU Learning

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000085548520	1.0	207.0	208.0
ZINC000000967534	17.0	35.0	52.0
ZINC000000968282	11.0	25.0	36.0
ZINC000197382366	9.0	1.0	10.0
ZINC000087515592	6.0	0.0	6.0
ZINC000000896527	1.0	5.0	6.0
ZINC000087515584	6.0	0.0	6.0
ZINC000059029793	5.0	0.0	5.0
ZINC000001845807	1.0	4.0	5.0
ZINC000100074226	4.0	0.0	4.0
ZINC000001687816	3.0	0.0	3.0
ZINC000001674476	1.0	2.0	3.0
ZINC000064622643	3.0	0.0	3.0
ZINC000001717025	3.0	0.0	3.0
ZINC000169747359	3.0	0.0	3.0
ZINC000057677596	3.0	0.0	3.0
ZINC000001706229	3.0	0.0	3.0
ZINC000002510896	3.0	0.0	3.0
ZINC000038147440	3.0	0.0	3.0
ZINC000064624846	3.0	0.0	3.0
ZINC000001580750	1.0	2.0	3.0
ZINC000004521424	1.0	2.0	3.0
ZINC000001530975	3.0	0.0	3.0

Table C.1 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000004769055	2.0	1.0	3.0
ZINC000001646307	1.0	2.0	3.0
ZINC000008034701	3.0	0.0	3.0
ZINC000006920393	0.0	3.0	3.0
ZINC000002167088	1.0	2.0	3.0
ZINC000100074301	3.0	0.0	3.0
ZINC000064624950	3.0	0.0	3.0
ZINC000001656970	1.0	2.0	3.0
ZINC000064624955	3.0	0.0	3.0
ZINC000064624956	3.0	0.0	3.0
ZINC000064624958	3.0	0.0	3.0
ZINC000064624959	3.0	0.0	3.0
ZINC000100074293	3.0	0.0	3.0
ZINC000064858311	3.0	0.0	3.0
ZINC000001668172	1.0	2.0	3.0
ZINC000100074278	3.0	0.0	3.0
ZINC000000388506	0.0	3.0	3.0
ZINC000070667148	2.0	1.0	3.0
ZINC000001654295	1.0	1.0	2.0
ZINC000004916677	2.0	0.0	2.0
ZINC000001615334	1.0	1.0	2.0
ZINC000002568170	1.0	1.0	2.0
ZINC000001665359	2.0	0.0	2.0
ZINC000002558831	2.0	0.0	2.0
ZINC000002810011	1.0	1.0	2.0
ZINC000000262398	1.0	1.0	2.0
ZINC000005705162	2.0	0.0	2.0
ZINC000002143278	2.0	0.0	2.0
ZINC000001620997	2.0	0.0	2.0
ZINC000012371961	1.0	1.0	2.0
ZINC000064622647	2.0	0.0	2.0
ZINC000000087024	2.0	0.0	2.0
ZINC000000388498	1.0	1.0	2.0
ZINC000000164418	1.0	1.0	2.0
ZINC000001562134	1.0	1.0	2.0
ZINC000000967759	1.0	1.0	2.0
ZINC000001849773	1.0	0.0	1.0
ZINC000002558787	1.0	0.0	1.0
ZINC000002558786	1.0	0.0	1.0
ZINC000002530747	1.0	0.0	1.0

Table C.1 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000001848575	1.0	0.0	1.0
ZINC000002539285	1.0	0.0	1.0
ZINC000001867001	1.0	0.0	1.0
ZINC000002034680	1.0	0.0	1.0
ZINC000002530978	1.0	0.0	1.0
ZINC000002168369	1.0	0.0	1.0
ZINC000002034704	1.0	0.0	1.0
ZINC000002163914	1.0	0.0	1.0
ZINC000002521500	1.0	0.0	1.0
ZINC000002040927	1.0	0.0	1.0
ZINC000002041197	1.0	0.0	1.0
ZINC000002507978	1.0	0.0	1.0
ZINC000002242728	1.0	0.0	1.0
ZINC000002172036	1.0	0.0	1.0
ZINC000002168416	1.0	0.0	1.0
ZINC000002077483	1.0	0.0	1.0
ZINC000002143276	1.0	0.0	1.0
ZINC000002530746	1.0	0.0	1.0
ZINC000003876016	1.0	0.0	1.0
ZINC000002558792	1.0	0.0	1.0
ZINC000038788724	1.0	0.0	1.0
ZINC000006118956	1.0	0.0	1.0
ZINC000006920369	1.0	0.0	1.0
ZINC000008579950	1.0	0.0	1.0
ZINC000008585874	1.0	0.0	1.0
ZINC000008683005	1.0	0.0	1.0
ZINC000038350321	1.0	0.0	1.0
ZINC000055757710	1.0	0.0	1.0
ZINC000004521187	1.0	0.0	1.0
ZINC000064622642	1.0	0.0	1.0
ZINC000070647192	1.0	0.0	1.0
ZINC000070666008	1.0	0.0	1.0
ZINC000070667144	1.0	0.0	1.0
ZINC000077311539	1.0	0.0	1.0
ZINC000078239350	1.0	0.0	1.0
ZINC000005935219	1.0	0.0	1.0
ZINC000003898186	1.0	0.0	1.0
ZINC000002558793	1.0	0.0	1.0
ZINC000002580648	1.0	0.0	1.0
ZINC000002558795	1.0	0.0	1.0

Table C.1 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000002558813	1.0	0.0	1.0
ZINC000002558814	1.0	0.0	1.0
ZINC000002558815	1.0	0.0	1.0
ZINC000002558822	1.0	0.0	1.0
ZINC000002577904	1.0	0.0	1.0
ZINC000002584246	1.0	0.0	1.0
ZINC000001845648	1.0	0.0	1.0
ZINC000003848023	1.0	0.0	1.0
ZINC000003860823	1.0	0.0	1.0
ZINC000003861058	1.0	0.0	1.0
ZINC000003876017	1.0	0.0	1.0
ZINC000003876018	1.0	0.0	1.0
ZINC000003876019	1.0	0.0	1.0
ZINC000003876020	1.0	0.0	1.0
ZINC000000027374	1.0	0.0	1.0
ZINC000001765793	1.0	0.0	1.0
ZINC000001562132	1.0	0.0	1.0
ZINC000001037160	1.0	0.0	1.0
ZINC000001078794	1.0	0.0	1.0
ZINC000001092275	1.0	0.0	1.0
ZINC000001482032	1.0	0.0	1.0
ZINC000001504642	1.0	0.0	1.0
ZINC000001505075	1.0	0.0	1.0
ZINC000001510094	1.0	0.0	1.0
ZINC000001530818	1.0	0.0	1.0
ZINC000001562131	1.0	0.0	1.0
ZINC000001570209	1.0	0.0	1.0
ZINC000000967827	1.0	0.0	1.0
ZINC000001570217	1.0	0.0	1.0
ZINC000001570219	1.0	0.0	1.0
ZINC000001570220	1.0	0.0	1.0
ZINC000001570221	1.0	0.0	1.0
ZINC000001570222	1.0	0.0	1.0
ZINC000001570229	1.0	0.0	1.0
ZINC000001570230	1.0	0.0	1.0
ZINC000001570231	1.0	0.0	1.0
ZINC000001570232	1.0	0.0	1.0
ZINC000001000251	1.0	0.0	1.0
ZINC000000967819	1.0	0.0	1.0
ZINC000001760831	1.0	0.0	1.0

Table C.1 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000000388139	1.0	0.0	1.0
ZINC000000066081	1.0	0.0	1.0
ZINC000000154638	1.0	0.0	1.0
ZINC000000157489	1.0	0.0	1.0
ZINC000000164198	1.0	0.0	1.0
ZINC000000165307	1.0	0.0	1.0
ZINC000000184654	1.0	0.0	1.0
ZINC000000262400	1.0	0.0	1.0
ZINC000000275176	1.0	0.0	1.0
ZINC000000354958	1.0	0.0	1.0
ZINC000000388499	1.0	0.0	1.0
ZINC000000967522	1.0	0.0	1.0
ZINC000000389555	1.0	0.0	1.0
ZINC000000394916	1.0	0.0	1.0
ZINC000000401218	1.0	0.0	1.0
ZINC000000404458	1.0	0.0	1.0
ZINC000000897131	1.0	0.0	1.0
ZINC000000900729	1.0	0.0	1.0
ZINC000000967335	1.0	0.0	1.0
ZINC000000967339	1.0	0.0	1.0
ZINC000000967412	1.0	0.0	1.0
ZINC000001574602	1.0	0.0	1.0
ZINC000001580747	1.0	0.0	1.0
ZINC000001580749	1.0	0.0	1.0
ZINC000001693315	1.0	0.0	1.0
ZINC000001669603	1.0	0.0	1.0
ZINC000001680348	1.0	0.0	1.0
ZINC000001685107	1.0	0.0	1.0
ZINC000001688068	1.0	0.0	1.0
ZINC000001689598	1.0	0.0	1.0
ZINC000001689804	1.0	0.0	1.0
ZINC000001689808	1.0	0.0	1.0
ZINC000001690839	1.0	0.0	1.0
ZINC000001693310	1.0	0.0	1.0
ZINC000001693568	1.0	0.0	1.0
ZINC000001580987	1.0	0.0	1.0
ZINC000001694147	1.0	0.0	1.0
ZINC000000038933	1.0	0.0	1.0
ZINC000001699878	1.0	0.0	1.0
ZINC000001700069	1.0	0.0	1.0

Table C.1 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000001718758	1.0	0.0	1.0
ZINC000001722248	1.0	0.0	1.0
ZINC000001725142	1.0	0.0	1.0
ZINC000001726969	1.0	0.0	1.0
ZINC000001758808	1.0	0.0	1.0
ZINC000001667772	1.0	0.0	1.0
ZINC000001666853	1.0	0.0	1.0
ZINC000001666852	1.0	0.0	1.0
ZINC000001661476	1.0	0.0	1.0
ZINC000001581013	1.0	0.0	1.0
ZINC000001581017	1.0	0.0	1.0
ZINC000001586329	1.0	0.0	1.0
ZINC000001589655	1.0	0.0	1.0
ZINC000001590020	1.0	0.0	1.0
ZINC000001591814	1.0	0.0	1.0
ZINC000001591842	1.0	0.0	1.0
ZINC000001598876	1.0	0.0	1.0
ZINC000001599927	1.0	0.0	1.0
ZINC000001599943	1.0	0.0	1.0
ZINC000001600920	1.0	0.0	1.0
ZINC000001601908	1.0	0.0	1.0
ZINC000001601909	1.0	0.0	1.0
ZINC000001601910	1.0	0.0	1.0
ZINC000001601911	1.0	0.0	1.0
ZINC000001601915	1.0	0.0	1.0
ZINC000001615335	0.0	1.0	1.0
ZINC000001632866	1.0	0.0	1.0
ZINC000001661474	1.0	0.0	1.0
ZINC000001697899	1.0	0.0	1.0

Table C.2: Most Popular Molecules in the High Scoring Pair in IF

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000000967534	35.0	98.0	133.0
ZINC000000896527	43.0	65.0	108.0
ZINC000000968282	30.0	59.0	89.0
ZINC000000900729	46.0	42.0	88.0
ZINC000001845807	40.0	42.0	82.0
ZINC000000967759	34.0	45.0	79.0
ZINC000001699878	27.0	34.0	61.0
ZINC000000388506	39.0	17.0	56.0
ZINC000000897131	28.0	15.0	43.0
ZINC000000164418	20.0	9.0	29.0
ZINC000000388499	22.0	6.0	28.0
ZINC000000388498	23.0	5.0	28.0
ZINC000006118956	19.0	0.0	19.0
ZINC000004521424	0.0	15.0	15.0
ZINC000002510896	10.0	5.0	15.0
ZINC000001717025	12.0	3.0	15.0
ZINC000001615335	2.0	13.0	15.0
ZINC000001646307	2.0	13.0	15.0
ZINC000100074293	11.0	3.0	14.0
ZINC000057677596	12.0	2.0	14.0
ZINC000038147440	12.0	2.0	14.0
ZINC000008034701	12.0	2.0	14.0
ZINC000064622647	8.0	5.0	13.0
ZINC000002167088	0.0	13.0	13.0
ZINC000006920393	0.0	13.0	13.0
ZINC000001580750	0.0	13.0	13.0
ZINC000064624846	9.0	4.0	13.0
ZINC000059029793	12.0	1.0	13.0
ZINC000004769055	9.0	4.0	13.0
ZINC000064858311	10.0	3.0	13.0
ZINC000001674476	2.0	11.0	13.0
ZINC000100074278	10.0	3.0	13.0
ZINC000001656970	0.0	12.0	12.0
ZINC000001687816	6.0	6.0	12.0
ZINC000001706229	7.0	5.0	12.0
ZINC000169747359	5.0	7.0	12.0
ZINC000002143278	9.0	3.0	12.0
ZINC000070667148	9.0	3.0	12.0
ZINC000100074301	9.0	3.0	12.0

Table C.2 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000001562134	2.0	10.0	12.0
ZINC000064624955	9.0	3.0	12.0
ZINC000100074226	7.0	3.0	10.0
ZINC000001581013	6.0	3.0	9.0
ZINC000001668172	0.0	9.0	9.0
ZINC000002810011	0.0	9.0	9.0
ZINC000002558795	7.0	2.0	9.0
ZINC000003876020	6.0	3.0	9.0
ZINC000001562132	7.0	2.0	9.0
ZINC000004521187	9.0	0.0	9.0
ZINC000070667144	7.0	2.0	9.0
ZINC000002558787	6.0	3.0	9.0
ZINC000001570231	6.0	3.0	9.0
ZINC000002168416	7.0	2.0	9.0
ZINC000002168369	7.0	2.0	9.0
ZINC000001580749	7.0	2.0	9.0
ZINC000001590020	6.0	3.0	9.0
ZINC000001562131	8.0	1.0	9.0
ZINC000003876016	5.0	3.0	8.0
ZINC000002558786	6.0	2.0	8.0
ZINC000070666008	3.0	5.0	8.0
ZINC000002242728	4.0	4.0	8.0
ZINC000001665359	4.0	4.0	8.0
ZINC000064624956	7.0	1.0	8.0
ZINC000001589655	4.0	4.0	8.0
ZINC000003876017	4.0	4.0	8.0
ZINC000001654295	0.0	8.0	8.0
ZINC000001598876	6.0	2.0	8.0
ZINC000001615334	0.0	8.0	8.0
ZINC000001580747	6.0	2.0	8.0
ZINC000064622643	6.0	2.0	8.0
ZINC000005705162	2.0	5.0	7.0
ZINC000001530975	4.0	3.0	7.0
ZINC000002558792	5.0	2.0	7.0
ZINC000064624958	7.0	0.0	7.0
ZINC000003848023	2.0	4.0	6.0
ZINC000064624950	6.0	0.0	6.0
ZINC000002521500	1.0	5.0	6.0
ZINC000012371961	0.0	6.0	6.0
ZINC000002558793	1.0	5.0	6.0

Table C.2 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000038788724	2.0	4.0	6.0
ZINC000000038933	3.0	3.0	6.0
ZINC000001693568	1.0	5.0	6.0
ZINC000000275176	1.0	5.0	6.0
ZINC000001570230	1.0	5.0	6.0
ZINC000001570232	1.0	5.0	6.0
ZINC000001581017	2.0	4.0	6.0
ZINC000001600920	3.0	3.0	6.0
ZINC000001601910	3.0	3.0	6.0
ZINC000001530818	3.0	3.0	6.0
ZINC000001570222	1.0	5.0	6.0
ZINC000002077483	3.0	3.0	6.0
ZINC000001632866	1.0	4.0	5.0
ZINC000001661476	2.0	3.0	5.0
ZINC000002530746	1.0	4.0	5.0
ZINC000001591814	5.0	0.0	5.0
ZINC000001722248	3.0	2.0	5.0
ZINC000001599943	3.0	2.0	5.0
ZINC000002558813	3.0	2.0	5.0
ZINC000002507978	1.0	4.0	5.0
ZINC000064624959	4.0	0.0	4.0
ZINC000002530747	3.0	1.0	4.0
ZINC000001601911	1.0	2.0	3.0
ZINC000001601915	3.0	0.0	3.0
ZINC000002558814	3.0	0.0	3.0
ZINC000005935219	1.0	2.0	3.0
ZINC000002558831	1.0	2.0	3.0
ZINC000002143276	3.0	0.0	3.0
ZINC000002539285	1.0	1.0	2.0
ZINC000064622642	1.0	1.0	2.0
ZINC000001580987	1.0	1.0	2.0
ZINC000002041197	1.0	1.0	2.0
ZINC000087515584	2.0	0.0	2.0
ZINC000001601908	1.0	1.0	2.0
ZINC000001601909	1.0	1.0	2.0
ZINC000002040927	1.0	1.0	2.0
ZINC000003876019	1.0	1.0	2.0
ZINC000001661474	1.0	1.0	2.0
ZINC000000262398	1.0	1.0	2.0
ZINC000003876018	1.0	1.0	2.0

Table C.2 continued from previous page

ZINC Index	Molecule 1	Molecule 2	Molecule Occur Times
ZINC000002558815	1.0	1.0	2.0
ZINC000001693315	1.0	1.0	2.0
ZINC000000087024	0.0	2.0	2.0
ZINC000002568170	0.0	2.0	2.0
ZINC000001570217	1.0	0.0	1.0
ZINC000070647192	0.0	1.0	1.0
ZINC000087515592	1.0	0.0	1.0
ZINC000000967827	0.0	1.0	1.0
ZINC000001570220	1.0	0.0	1.0
ZINC000001505075	1.0	0.0	1.0
ZINC000000157489	0.0	1.0	1.0
ZINC000001570219	1.0	0.0	1.0
ZINC000001078794	0.0	1.0	1.0
ZINC000002163914	0.0	1.0	1.0
ZINC000001570221	1.0	0.0	1.0
ZINC000001570229	0.0	1.0	1.0
ZINC000001599927	0.0	1.0	1.0
ZINC000003898186	0.0	1.0	1.0
ZINC000001688068	0.0	1.0	1.0
ZINC000002584246	0.0	1.0	1.0
ZINC000002580648	0.0	1.0	1.0
ZINC000002558822	0.0	1.0	1.0
ZINC000001697899	0.0	1.0	1.0
ZINC000001758808	0.0	1.0	1.0
ZINC000001760831	0.0	1.0	1.0
ZINC000001765793	0.0	1.0	1.0
ZINC000002530978	0.0	1.0	1.0
ZINC000002172036	0.0	1.0	1.0
ZINC000001867001	0.0	1.0	1.0

Appendix D

Predicted Molecule Pairs

Please refer to the GitHub repository address:

1) PU Learning:
`https://github.com/Charlie059/Co-Crystal-Prediction-Master/blob/main/data/res_Spy.csv`

2) Isolation Forest:
`https://github.com/Charlie059/Co-Crystal-Prediction-Master/blob/main/data/res_IF.csv`

Appendix E

Ethical Statements

All data used in this study, including CSD, ZINC15 and other data, were free of any personal data and all use of the data was approved.