The Smell of Polymers: Prediction of Molecular Odor Characteristics by Deep Learning Models

Christos Koromilas, Stephan Possen, Mireia Fortuño Ledesma,
Lázár Dávid Barta, Mathias Verschueren

January 2024

Abstract

Polymer products have the possibility to emit a certain smell which can be harmful to the experience of the user when functioning as a container for food or drinks. Molecules present in these products can leach into consumables and alter their taste and smell as well. To prevent this from happening, companies need to test the polymers they use for production on smell and intensity. This process involves a professional panel, and can be expensive and time consuming. In this paper different approaches to automate the testing process are discussed. Using Deep Learning models, we try to predict if a polymer is odorless or not, as well as predicting the correct smells. Anomaly detection is used for predicting if a polymer is odorless or not, and different Multi-Label Classification models are used for predicting the correct smells. In the end, a combination of models using High-Level Data Fusion seems to provide the best results for predicting smells, while the Anomaly Detection looks to be very promising for predicting if a polymer is odorless.

1 Introduction

It is common for newly bought products made out of plastic to smell when unpacked for the first time. For example, an inflatable pool for your backyard or a water bottle for sports. In those cases, it is usually an expected occurrence, not harmful to the experience of the user. The same can not be said about products that are meant to be used for substance and fluids consumption. These products should not have a strong odor, and should definitely not alter the smell and/or taste of any substances they come in contact with. When producing these kind of products, it is important to keep in mind what kind of molecules are used. Different molecules, and more specifically the composition and structures of the functional groups within, lead to different types of smells and intensity, crucial aspects for the perception of an aromatic compounds [1]. As a result, it is important for companies to know the smells exhibited by the molecules they produce, and whether or not they are perceptible by the end consumer, in order to come up with new products that are acceptable.

Any company that produces polymers intended for medical use, or consumption purposes, is subjected to strict EU regulations concerning leachables, i.e. molecules leaching from the material into the body, water, or food under normal use conditions. The principles set out in Regulation (EC) No 1935/2004 [2] require that materials do not: (1) Release their constituents into food at levels harmful to human health, and (2) Change food composition, taste, and odor in an unacceptable way.

This is the case of Envalior, a leading global engineering materials solution provider. At Envalior, leachable experiments are performed in order to determine the identity and quantity of leachables. In case any leachable is detected, a test panel has to perform an odor and taste test. Unfortunately, it is still difficult to relate panel results to the molecular properties of a leechable. On top of this, the tests are also time

consuming, expensive, and they require expert knowledge. As it stands, there is no analytical way to determine if an identified leachable will be odorant or not.

The problem the company faces is closely related to an open issue in the field of olfactory science. Researchers have long known that the chemical structure of the molecules we inhale influences what we smell [3], but until recently, it has been impossible to predict the smell of a molecule from its chemical structure [4], while the most modern methods still achieve far from perfect results [5].

Researchers have resorted to using Machine Learning techniques to unravel the field of olfactory science. The latest AI model has achieved human-level skill in describing how certain chemicals will smell, closing a critical gap in the scientific understanding of olfaction [6]. It outperforms previously published models to the point that, replacing a trained human's responses with the model output, would improve overall panel description, though the model still can't match the level of a full panel of experts [5].

In order to contribute to this open issue and to help Envalior with the problem they face, the goal of the project is defined: to provide Envalior with Machine Learning models that can predict the presence of odor descriptors and describe which odors are present in leachables identified/quantified in extracts. This will allow the company to minimize time consuming and expensive analytics and test, as well as contribute to the key challenge in olfaction of mapping molecular structure to odor perception.

The **Research Questions** that the team wants to answer with the project are also defined:

- Does a Deep Learning model accurately distinguish odorant from non-odorant molecules?
- How effectively can a Deep Learning model assign correct odor descriptors to molecules?
- How can Data Fusion improve the performance of predictive models?

2 Background

2.1 Feature Extraction

A very important part of this project is the extraction of the features from the molecules, in order to make them machine readable. We will use the features to feed the molecules into our machine learning systems, and generate predictions for their odors. This section explores some of the previous work that has been done regarding this topic.

RDKit is a free and open-source cheminformatics toolkit designed to handle chemical information as well as aid in Machine Learning tasks. It provides a range of functionalities including the manipulation of chemical structures, computation of molecular descriptors, and generation of molecular fingerprints, all of which are crucial in the fields of drug discovery and computational chemistry. It uses the SMILES (Simplified Molecular Input Line Entry System) notation. With a Python interface, it is accessible for data analysis and can be integrated into various Machine Learning pipelines, making it a valuable tool for researchers and practitioners in related domains [7].

Molecular Fingerprints are binary encoded representations of molecules that show the indication or absence of particular substructures in the molecule. Fingerprints can have several formats depending on the molecules and the information we require. Fingerprinting is useful to compare molecules and to make them machine readable [8]. One of the most used fingerprinting technique is the *Morgan Fingerprint*, or also referred to as *Circular Fingerprint*. It converts chemical atom groups into a binary vector, using 2 defining parameters: length and radius. The term "length" specifies the dimension of the vector, while "radius" defines the scope of the atom groups. An increased radius is capable of representing larger atom groups. Within the vector, each bit corresponds to a distinct atom group. However, it does not indicate the number of each group present in the molecule [9].

Molecular Parameters or **Descriptors**

are also useful for the featurization of the molecules. There exists a method in the library *RDKit* to derive descriptors based on the SMILE of a molecule. These descriptors give a good insight to the molecule, and allow to retrieve features that describe it [10].

Graph Neural Networks are graph or structural representations of molecules. They are commonly used, since they have access to the characteristics of the molecule. This method is often used to predict on given materials' properties thanks to their capacity to learn their internal representations [11]. For our case, this method is potentially very useful, and is worth consideration. Using the python library **MolGraph** [12], which is a package specifically developed to create G-NN for molecular data, we can get the graph representation of molecules. In this representation:

- **Nodes** represent atoms, encompassing various atomic features.
- **Edges** represent bonds, detailing the bond features between atoms.

2.2 One-class Classification

In the context of Anomaly Detection, the use of Autoencoders, a special type of Neural Network, stands out in the field of One-Class Classification [13]. These networks are adept at learning compressed representations of data in an unsupervised manner, which makes them particularly valuable for dimensionality reduction and feature learning tasks. In industries like those at Envalior, where the focus is on identifying anomalies in odor profiles of products, Autoencoders prove to be highly effective. They operate by learning a normal data representation, and subsequently identifying deviations or anomalies based on reconstruction errors, i.e. the differences between original and reconstructed inputs. Although Autoencoders offer substantial benefits in Anomaly Detection, their performance hinges on appropriate data representation, the correct setting of reconstruction error thresholds, model complexity, and continuous model updates to cater to new or evolving data patterns. Their adaptability and effectiveness in capturing the nuanced differences in data makes them a crucial tool in maintaining the high standards of product quality and regulatory compliance.

2.3 Multi-Label Classification

This section delves into the different Machine Learning methods that can be used on the extracted features to predict the odorant properties of molecules. Though these methods are in essence different from feature extraction, they can not completely be studied independently, since different models can require different representations as input data.

Binary Relevance and **Classifier Chains** are two simple methods of building a Multi-Label Classification model. Both these methods adapt algorithms that are originally designed for Single-Label Classification to be used in Multi-Label scenarios. They train multiple of the same base binary classifier, one for each label. The difference between these methods is that Binary Relevance treats each label independently, training several independent binary classifiers where correlation between odors is discarded, while Classifier Chains consider the dependencies between labels, meaning that a classifier Ci uses the predictions of all the previous classifiers Cj , where $j < i$ [14].

Since the base classifier can be any binary classification model, like a Random Forest, they have the advantage of enabling inspection of feature importance, thereby enabling insight into what features contribute most to the classification, and giving a deeper understanding of the feature-odor relationships.

K. Saini and V. Ramanthan [15] have reported results comparable to state of the art Graph Neural Networks when applying them to the problem of odor prediction of molecules. From the results, Binary Relevance was deemed to achieve higher accuracy than Classifier Chains in this problem setting. Brian K. Lee et al. [5] also implemented Random Forest as a baseline method in comparison to a method based on Graph Neural Networks, which outperformed the former.

Another approach to predict the odor of molecules is the use of Deep Learning techniques in the form of **Neural Networks**. The idea is to design a Neural Network with different heads (i.e. output nodes) corresponding to the different odor labels present in the dataset. This allows to use it in the context of Multi-Label Classification problems. Liang Shang et al. [16] first used multiple forms of feature extraction: Molecular parameters, Molecular Fingerprints, Molecular Graph, Atomic Interactions. After feature extraction, they trained a multi-headed Neural Network to predict the odors. Finally, they used Convolutional Neural Networks to extract graph based features, resulting in the highest prediction accuracy of the Neural Network.
Brain K. Lee et al. [5] recently created a Principal Odor Map using a Message Passing Neural Network (MPNN), which is a type of **Graph NN**. Their achieved performance surpassed that of the median panel expert. The MPNN ends in two layers, the first represents the extracted features from the graph-based representation, while the final layer represents the odor descriptors. The odor map is created from the features of the former. Smell predictions are made by finding the coordinates in the feature space of a specific molecule, and projection them on the axes corresponding to the different odors. The norm of the resulting vector is also representative of the intensity of the smell of a molecule.

Other methods: AdaBoost, Support Vector Machine, Gradient Boosting [17] can also be used to predict smell from molecules, though this study uses a relatively small amount of data, so it does not have strong conclusions about their performance.

2.4 Data Fusion

Data Fusion can be used to produce more consistent, accurate, and useful information than that provided by any individual data source or model alone [18]. This technique is widely used in various fields such as image and signal processing, and data analytics. The integration process in Data Fusion enhances the reliability, range, and accuracy of information, making it a valuable tool in complex systems analysis. Some of the Data Fusion options available are [19]:

- **Low-Level Fusion (Data-Level Fusion):** This is the most basic form of fusion, where raw data from different sources is combined before any processing occurs. It is useful when all data sources are of the same type, and the fusion aims to enhance the raw data's quality or quantity.

- **Mid-Level Fusion (Feature-Level Fusion):** Here, features extracted from raw data are combined, as observed in Figure 1. It is more abstract than Low-Level fusion because it involves data that has already been processed. This method is beneficial when data from different sources is complementary, and it can enhance the model's learning.

- **High-Level Fusion (Decision-Level Fusion):** This involves merging the outcomes of several models, as shown in Figure 2. It is a more complex form of fusion, used when there are multiple models analyzing different kinds of data, and want to combine their findings to make a final decision or prediction.

Some known techniques for Decision Fusion are explained below:

Majority Voting is a simple and intuitive method, where each classifier in the group makes a prediction (vote) for each input sample. The final decision is determined by the majority of the votes. This method assumes equal importance and reliability for all classifiers, making it straightforward to implement. Majority voting is especially effective when the classifiers are diverse and individually accurate, as it can reduce the risk of an erroneous decision by relying on the collective wisdom of the group [20].

Bayesian Consensus is a probabilistic model that considers the uncertainty and reliability of individual classifiers or experts. It

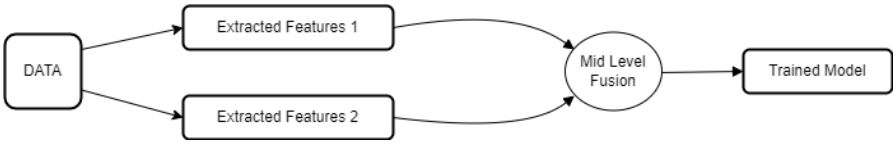


Figure 1: Mid-Level Fusion

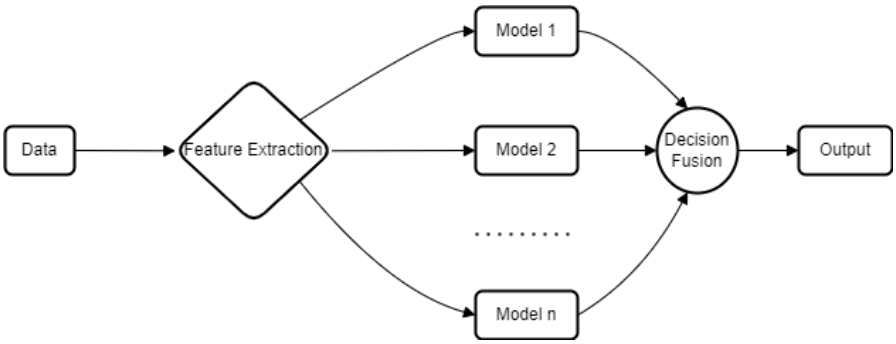


Figure 2: High-Level Fusion

updates the probability estimates of the outcomes based on Bayes' theorem. Each classifier contributes to the final decision based on its performance and the confidence of its predictions. This method is particularly useful when the data is incomplete or uncertain. The Bayesian consensus approach ensures that the final decision takes into account the reliability and the performance of each model, making it a powerful tool for Decision-Level fusion [20].

Weighted Voting, on the other hand, is a straightforward yet effective technique where each classifier is assigned a weight based on its importance or reliability. The final decision is made based on the weighted sum of the votes from each classifier. Unlike simple Majority Voting, Weighted Voting allows for the differentiation between more reliable classifiers and those that are less so. This method ensures that the contributions of individual classifiers to the final decision are proportional to their performance, which enhances the decision-making process, especially when classifiers have varying levels of accuracy and reliability [20].

All these methods offer robust frameworks for combining the insights from multiple models or classifiers. By leveraging these techniques, researchers can enhance the reliability and accuracy of their decision-making systems, particularly in complex scenarios involving diverse data sources and models.

2.5 Limitations and Open Issues

Despite the fact that extensive research has been done on predicting the descriptors of the odors of molecules, the main issue reaming is the dependency on the limited data at hand, and the computing capabilities available to the researchers.

3 Implementation

3.1 Implementation Approach

For the implementation of the project, there are 2 tasks at hand:

- Predict if molecules have an odor or not.
- Predict what odors a molecule has.

The data to solve this tasks is a public dataset that was given to the team. For *Task 1*, we create a new target variable that will be 1 when the label is "odorless", and 0 when any kind of smell is described. For *Task 2*, we use

the multi-label information. We also study the label distribution of this data, finding that it is highly imbalanced. Therefore, at the end, we provide label handing options that we believe can provide better results.

For both tasks, we first perform feature extraction from the molecules' SMILES representation, in order to obtain features to feed to the different models. Once we have obtained the features, we start building the models. For *Task 1*, we build One-Class Classification models based on Autoencoders. For *Task 2*, we build 7 different Multi-Label classification models. We also implement High-Level Data Fusion on the output of each task, so that we can compare if including these techniques in our modeling approach improves their performance.

Finally, we evaluate the models with the adequate metrics for each task, and discuss the results.

3.2 Data

The dataset provided to the group for the project originally comes from an AIcrowd challenge called "Learning to Smell" [21], where people were challenged to solve the same problem as the group in *Task 2*, predicting the odors a molecule has. The dataset obtained from the challenge is called "train.csv", and it contains 4316 rows (molecules) and 2 columns: SMILES, which is the SMILE encoding for each molecule, and SENTENCE, which is string of associated odors to that particular molecule, like "fruity, rose". In particular there are 109 different odors in the data, and the total number of different combinations of these is 2772.

We performed some exploratory data analysis of the smells in SENTENCE to see how they are distributed. We saw that some of the smells have almost no representation, for example the odor "odorless" has only 57 records in the data, out of 4316 (1.3% of the data). This complicates the prediction tasks, since there won't be sufficient representative data for some of the smells, and the more present smells, like "fruity", can overtake the predictions.

3.3 Feature Extraction and Data Preparation

In this study, we aimed to harness the rich information embedded in the chemical structures by extracting a comprehensive set of features. This was facilitated by the utilization of RDkit and Molgraph libraries, each offering specialized functionalities for chemical informatics and graph-based molecular data, respectively. Here is an overview of the feature extraction process:

1. **Chemical Descriptors:** Leveraging the capabilities of the RDKit library, we extracted a diverse set of chemical descriptors. These descriptors, such as molecular weights and sizes, provide fundamental insights into the chemical properties of the molecules.

2. **Circular Morgan Fingerprints:** To capture the essence of molecular structure, we computed Circular Morgan Fingerprints. These binary vectors effectively represent the presence or absence of specific chemical substructures, offering a compact yet informative representation of the molecules.

3. **Atoms and Bonds (Graph Neural Network Features):** Using the Molgraph library, we delved into the graph representation of molecules. This approach allowed us to harness the structural intricacies of molecules in a format conducive to Graph NN models.

3.3.1 Data Preprocessing and Integration

Post feature extraction, the dataset underwent meticulous preprocessing to ensure quality and consistency. For the Descriptors:

- Missing values were identified in the features and imputed using a KNN imputer.

- Features with constant values across the dataset were removed, as they do not contribute to model discrimination.

- The remaining numerical features were scaled to a range between 0 and 1.

Then we applied a Mid-Level Fusion, and the features from different sources (descriptors and fingerprints) were concatenated to form a comprehensive dataset, paving the way for robust model training.

The preprocessing pipeline was instrumental in refining the dataset, ensuring that the subsequent modeling phase was built on a foundation of clean, normalized, and integrated data.

3.4 One-class Classification

3.4.1 Experimentation with Autoencoders

Given the complex nature of our dataset, and the intricate challenge of Anomaly Detection within it, Autoencoders were identified as particularly suitable models due to their proficiency in capturing and reconstructing high-dimensional data. Our approach involved the construction and training of 3 distinct Autoencoder architectures, each tailored to a specific type of feature representation of the chemical compounds:

1. An Autoencoder dedicated to capturing the nuances of Chemical Descriptors, which are detailed characterizations of chemical properties.

2. An Autoencoder designed for Circular Morgan Fingerprints, capturing the essence of molecular structure in a binary form.

3. An Autoencoder for Fused Data, which combines Chemical Descriptors and Circular Morgan Fingerprints, to leverage a comprehensive representation of the compounds as Mid-Level Data Fusion.

To ensure that our models achieved the best possible performance, we embarked on a rigorous hyperparameter tuning. This process entailed the exploration and optimization of several key parameters:

- The depth of the network, or the number of hidden layers, which influences the model's capacity to learn complex patterns.

- The breadth of the network, or the number of neurons in each layer, which dictates the level of detail the model can capture.

- The learning rate, a crucial hyperparameter in the training process, affecting how quickly the model learns.

Our tuning efforts were driven by the goal of minimizing the Mean Squared Reconstruction Error, a common loss function for Autoencoders that measures the disparity between the input and its reconstruction. This loss function is particularly well-suited for training Autoencoders, as it encourages the learning of efficient data encodings. The minimization was performed using Keras-Tuner, an advanced hyperparameter tuning library that facilitates the automation of the optimization process.

The culmination of our Anomaly Detection system involved establishing a suitable threshold. This threshold is critical, as it decides whether a given molecular compound is considered as an anomaly (odorless), based on its reconstruction error. The procedure for setting this threshold included:

1. Refining the model by training it on the full dataset to leverage all available information.

2. Employing a validation set to calculate the reconstruction errors, and using ROC Curve analysis to ascertain the optimal threshold that balances the True Positive Rate and False Positive Rate, thereby maximizing the Area Under the Curve (AUC).

By meticulously tuning and validating our models, we ensured that they not only perform well on the training data, but also exhibit robust generalization to new unseen data, thereby serving as reliable tools for odorlessness prediction in chemical compounds.

3.5 Multi-label Classification

For *Task 2*, we try 3 different types of models for Multi-Label Classification: Binary Relevance and Classifier Chains, Normal Neural Networks, and Graph Neural Networks. For the first 2 techniques, we build 2 different models with different features, one for fingerprints data, and one for descriptors data. Additionally, for both NN models, we also optimize them with Keras tunner, obtaining 2 extra models. Therefore, we get 7 models at the end, and we perform High-Level Data Fusion on their results, to obtain a final prediction for *Task 2*.

It is important to mention that the team decided that each model will output the top 3 smells with highest probability, regardless of if these probabilities are high or not. We took this as inspiration from the Challenge Learning to Smell [21], since the cardinality of the data (mean number of labels per sample) is 3.

We divided the data for all models into 80% training, 10% validation, and 10% testing, to have all the models trained and evaluated on the same data.

3.5.1 Binary Relevance Models

One of the possibilities explored in the Background section for Multi-Label Classification was Binary Relevance and Classifier Chain models. In this section, we experiment with them.

The features we use to build the model are the Chemical Descriptors and the Circular Morgan Fingerprints, separately, because these features are very different. Because of this, we aim to build 2 separate models instead of 1 that combines all features. Therefore, we experiment with building a model for the Descriptors data, and another model for the Fingerprints.

We first experiment with the Binary Relevance method, using the class provided by the Python package skmultilearn [22]. Within the Binary Relevance, we try different base binary classifiers. In particular, we experimented with Random Forest Classifier [23], XGBoost Classifier [24], and SVC (Support Vector Classification) [25]. We also experiment with providing class weights to classifiers, to help with the imbalance of classes. We then run the same experiments, but with the method of Chain Classifiers, using a different class provided by skmultilearn [26]. It is important to notice that we perform all experiments for both sets of features, independently.

The combinations that achieves the best results, based on the tests we perform for the Fingerprints and for the Descriptors, are then considered as the final models.

3.5.2 Neural Network Models

To achieve odor prediction, we also implemented Neural Networks, using the TensorFlow's Keras API [27]. Trying to manually optimize the NN implementation, a number of different NN model configurations have been developed with different hyperparameters, as visible on Table 1. For each model, for the input layer, we took the descriptors or the fingerprints and for the output layer, we use the one hot encoded probability of each known odor to our model. Each model uses ReLu activation function for all layers, except for the output layer, where Sigmoid is used.

Model	Layer 1	Layer 2	Layer 3
Model 1	256	-	-
Model 2	128	-	-
Model 3	128	128	-
Model 4	256	128	-
Model 5	256	256	-
Model 6	256	128	64
Model 7	64	-	-
Model 8	64	64	-
Model 9	64	128	-

Table 1: Basic NN models

The different models were manually created to obtain a comparison between different hyperparamters, so we can study them and choose the model that performs the best. In the next part of this subsection, we will discuss the automation of this manual process.

3.5.3 Optimized NN Models

An attempt to enhance the base Neural Network models was also made. The following hyperparameters that are present in these models were optimized using KerasTuner [28]: Number of units in dense layers, and activation function. Both the model using the molecular descriptors and the model using the fingerprints have the same structure for optimization which can be seen in Table 2 below.

Hyperparameter	Optimizable value
Size input	32 - 512
Size layer 1	32 - 512
Size layer 2	32 - 512
Size layer 3	32 - 512
Activation input	Relu/TanH/Sigmoid
Activation layer 1	Relu/TanH/Sigmoid
Activation layer 2	Relu/TanH/Sigmoid
Activation layer 3	Relu/TanH/Sigmoid

Table 2: Optimization structure for NN models

Note that for the size of the layers, a step size of 32 was used, and that the final layer, with an output of 109 for all the different smells, always used the Sigmoid activation function.

3.5.4 Graph NN

For the implementation of a Graph NN, we made use of the MolGraph [12] packages in combination with Keras, and TensorFlow.

The graphs are created by both generating atom and bond embeddings for the atoms and bonds in the molecule, based on their properties. Some of the Atom features include: Atomic Symbol, hybridization, and Aromatic. Some of the Bond features are: BondType, and Conjugated. A list of all the features included can be found in the Appendix B.1.

The whole network consists of 2 Graph Convolutional Layers, followed by 2 Dense feed forwards layers, and a final output layer, as seen in Table 3.

We did some experiments with the number of layers and their activation functions. The final model composition we chose was the one that achieved the best results according to our tests. The optimizer we used was the Keras implementation for the Adam optimizer [29]. The learning rate was set to 0.007, as larger learning rate resulted in an unstable training process.

Layer	Activation function	Size
GCN	ReLU	64
GCN	ReLU	32
Dense	ReLU	128
Dense	ReLU	64
Output	Sigmoid	109

Table 3: Configuration of final GNN

3.6 High-Level Data Fusion

To further refine our approach, we employed Decision Fusion techniques. These techniques involve combining the predictions from different models, to improve the overall performance.

3.6.1 Task1

Specifically for *Task 1*, the predictions from the Descriptors model and the predictions of the Circular Morgan Fingerprint model were combined. The task is a One-Class Classification, so we adapted the Bayesian Decision Fusion and the Weighted Voting, that are based for Binary Classification, to work on our task with Autoencoders.

3.6.2 Task2

For *Task 2*, we combine the predictions from the 7 different Multi-Label Classification models we obtain: Binary Relevance for Descriptors, Binary Relevance for Fingerprints, Neural Network for Descriptors, Neural Network for Fingerprints, Optimized NN for Descriptors, Optimized NN for Fingerprints, and Graph NN. The technique chosen to combine these models' outputs is Majority Voting.

In particular to our implementation, each of the 7 models returns a prediction of the form [top1 OD, top2 OD, top3 OD]. Therefore, we assign a score of 3 to the 1st odor, a score of 2

to the 2nd odor, and a score of 3 to the 3rd odor. Then, the scores received by each smell, from each model, are summed to obtain a final ranking of odors with their corresponding scores, and the top 3 odors with highest accumulated scores are returned as the final prediction.

Since there could be cases where some odors are tied on a position, it is important to have some sort of ordering of what model should break the tie with its decision. We create this raking later on, based on the results we obtained from the evaluation of the individual models.

3.7 Evaluation

An important part of this project is the evaluation of the performance of the models. Since *Task 1* is a Binary Classification problem, **Accuracy**, **Precision**, **Recall**, **F1-score**, and **AUC** will give a good indication of the model's performance. However, for *Task 2*, these metrics might not be a good choice. For Multi-Label Classification problems, such as predicting the smells of polymers where it is possible to have multiple smells, using Accuracy as a metric can be way too strict.

Consider the example of comparing two bit vectors in Figure 3. Notice that the two vectors are close to being the same vector except for the 6th bit. Going by logic, this would be regarded as a pretty good prediction, but using Accuracy, this prediction would be considered as completely wrong, since it only considers it correct when both vectors are exactly the same. Therefore, metrics that are different from Accuracy, Precision, Recall, and F1-score should to be considered.

$$\hat{\mathbf{y}} = [0, 0, 0, 0, 1, 0, 0]$$

$$\mathbf{y} = [0, 0, 0, 0, 1, 1, 0]$$

Figure 3: Example of comparing two bit vectors

The following two metrics are considered to evaluate the Multi-label Classification models:

The first metric is the **Hamming Loss**, which looks at the individual bits when comparing two bit vectors (2 prediction vectors). After comparing all the individual bits, the total amount of wrongly classified bits is divided by the total length of the vector. Unfortunately, in our case, this will still not provide any real information about the model's ability to predict the correct smells. The reason for this is that there are 109 different smells, and on average, a polymer has 3 odors assigned to it. This means that if a model were to predict all zeros (no present odor), the Hamming Loss would still be low, and seemingly look like a good performance. Therefore, Hamming Loss can not be considered a good metric to use for evaluating the quality of our models either.

Predicted:	[Fruity, metallic, pungent]
True:	[Fruity, rose, herbal, sweet]

Figure 4: Example of comparing predicted and true smells

The second is the **Jaccard Score**. It evaluates the number of correct predictions with regard to the total number of predicted smells to be present, by dividing the size of the intersection between predicted and true smells by the union of predicted and true smells. Consider an example of comparing two sentences containing smells in Figure 4. Notice that the predicted labels and the true labels have one smell in common, and thus the value of the intersection is 1. The value of the union is 6, since in total there are six different smells predicted. The Jaccard Score will then give a value of $1/6 \approx 0.16$. This is a much better metric to evaluate the performance of the Multi-Label Classification models, since it does not take the negative predictions into account. Therefore, the Jaccard Score will be the main evaluation metric to consider in this project.

4 Results & Discussion

In this section, we show the results of the models from *Task 1* and *Task 2* individually, and with the use of Data Fusion techniques. We then discuss the findings to asses in answering the research questions.

4.1 Task 1 results

As mentinoned before in the One-Class Classification section, we tried both Binary and One-Class Classification. The results in Binary Classification were good but not representative for the number of samples that we had for each class. In comparison, with One-Class Classification models we didn't get so good results, but they were more representative for the results.

4.1.1 Binary and One-Class Classification

The results of the Binary and One-Class Classification models, as outlined in Tables A.1 and A.2, provide a comprehensive view of model performance across different feature sets and classification strategies. The performance metrics, including Accuracy, Precision, Recall, F1-Score, and AUC (Area Under the Curve), are critical in understanding the efficacy of the models in distinguishing between odorless and odorous molecules.

In the context of Binary Classification (Table A.1), the models demonstrated high accuracy, especially with the Binary Deep Learning (DL) approach across all feature sets. However, it is noticeable that the Precision, Recall, and F1-Score for the minority class (odorless molecules, denoted as '1') are significantly lower than those for the majority class. This discrepancy underscores the challenge of class imbalance in the dataset. While the models are proficient in identifying the majority class, their ability to detect the minority class is limited, as evidenced by the lower scores for the odorless molecules.

The Fused feature set combined with the Binary DL model showed promising results, achieving the highest F1-Score for the minority class. This indicates that the fusion of feature sets potentially enhances the model's capability to capture the distinctive characteristics of odorless molecules, thereby improving performance metrics for the minority class.

Moving to One-Class Classification (Table A.2), the models were specifically tasked with identifying the normal (odorant) class and flagging the anomalies (odorless). The One-Class SVM (OC-SVM) models exhibit a balanced accuracy around 0.50, indicating a challenge in effectively distinguishing between the two classes. In contrast, the Autoencoder models performed significantly better, with higher accuracy and AUC values. This suggests that the Autoencoder's capability to reconstruct input data and identify reconstruction errors as anomalies is a promising approach for this particular dataset.

The Autoencoder's performance is notably superior in the Descriptors feature set, achieving the highest AUC of 0.82. This could be attributed to the Descriptors providing a more nuanced representation of the molecular structures, which the Autoencoder effectively leverages to differentiate between normal and anomalous data points.

In summary, while Binary Classification models show high accuracy, their performance on the minority class needs attention. The One-Class Classification approach, particularly with Autoencoders, demonstrates potential in handling the imbalanced nature of the dataset. The fusion of feature sets and the optimization of Autoencoder architectures could further enhance model performance, a prospect that merits further investigation.

4.1.2 Performance of Autoencoders

The Autoencoder models, designed for anomaly detection in the context of odor classification, demonstrate the nuanced capability of unsupervised learning techniques in handling unbalanced datasets. Table A.3 reveals a consistent theme across the models: while accuracy remains relatively high, precision and recall for the odorless class (considered anomalies) are modest. The F1-Scores further reflect the challenge of achieving a balanced trade-off between precision and recall in such imbalanced

settings.

When examining the confusion matrices in Figure A.1, we observe a significant number of false negatives across all feature sets. This outcome is indicative of the models' tendency to favor the majority class. However, it's worth noting that the number of false positives is substantially lower, suggesting that when the models do predict an odorless molecule, they tend to do so with higher confidence.

The ROC AUC Curves in Figure A.2 further contextualize the models' performance. While the area under the curve (AUC) for Descriptors and Fused feature sets indicates a fair degree of separability, the relatively lower AUC for Fingerprints hints at the complexity of capturing the full essence of molecular structures through this representation alone.

The fusion techniques — Bayesian Decision and Weighted Voting — present intriguing outcomes. The Weighted Voting achieves a notable accuracy of 0.95; however, this comes at the expense of precision and recall for the minority class. In contrast, the Bayesian Decision approach leans towards better recall at a lower overall accuracy, emphasizing its utility in scenarios where missing an odorless molecule is significantly disadvantageous.

In summary, the results before hyperparameter optimization illuminate the inherent trade-offs when modeling imbalanced datasets. These models set a benchmark for subsequent optimization efforts. The subsequent fine-tuning through KerasTuner, as will be discussed, aims to refine these trade-offs to better serve the identification of odorless molecules in chemical compounds.

4.1.3 Hyperparameters optimization

In this A.3 the architectures of the optimized models are shown. The hyperparameter optimization process, as captured in Table A.4, has resulted in enhanced model performances, as evidenced by the improved metrics across the board. Accuracy levels remain exceptionally high, with notable gains in AUC for the Descriptors model, indicating a strong ability to differentiate between classes. It is essential, however, to consider these results in light of the inherent class imbalance which makes high accuracy somewhat expected when the model is biased towards the majority class.

The confusion matrices (Figures within A.4) reveal an important aspect of the model's performance post-optimization. While false negatives have been reduced, indicating an increased sensitivity to the minority class of odorless molecules, the persistent occurrence of false positives implies a tendency of the models to misclassify some odorant molecules as odorless.

Particularly noteworthy is the performance of the Bayesian Decision Fusion method, which exhibits a more balanced approach to classification than the Weighted Voting Fusion, as suggested by its confusion matrix and the corresponding ROC curve (Figures A.6a and A.7a). Despite a slight decrease in accuracy compared to the Weighted Voting Fusion method, the Bayesian approach demonstrates an increased capability to correctly identify odorless molecules, which is vital for practical applications where the detection of such molecules is critical.

Furthermore, the ROC curves (Figure A.5) for individual models and fusion techniques underscore the nuanced performance improvements. The ROC curve for the Descriptors model, in particular, highlights its superior specificity and sensitivity among the individual feature sets.

In summary, the optimization efforts have successfully fine-tuned the models, providing a better balance between sensitivity and specificity, especially in the context of an imbalanced dataset. While the accuracy remains high, the real progress is noted in the improved AUC and F1-Scores, which are more indicative of the models' practical performance. These results affirm the effectiveness of hyperparameter tuning and its crucial role in preparing models for real-world deployment, where the cost of misclassification can be significant.

4.1.4 Optimised Autoencoders with optimum thresholds

The refined thresholds as presented in Table A.5 exhibit a marked improvement in the detection capabilities of the models, particularly in their ability to identify the minority class of odorless molecules without a substantial increase in false positives. This balance is essential in practical scenarios where the cost of missing an odorless molecule could be high.

The confusion matrices (Figures within A.8) offer a visual confirmation of the models' performances. The Descriptors model, after threshold adjustment, has managed to maintain high true negative rates while improving the true positive rates, as evidenced by the lower number of false negatives. This demonstrates an improvement in the model's sensitivity without a significant trade-off in specificity.

The Bayesian Decision Fusion technique, now with its threshold optimized, shows a remarkable ability to classify the minority class correctly (Figure A.9a). Although this comes with an increased rate of false positives, the trade-off results in a higher Recall, which is crucial for applications where the detection of an odorless state is more critical than the occasional misclassification of an odorant one.

The ROC AUC Curves (Figure A.10) for the high-level fusion techniques further illustrate this point. The area under the curve (AUC) for the Bayesian Decision Fusion has seen an increase, which suggests that while the model is making more false positive errors, its ability to detect true positives (odorless molecules) has improved significantly.

In essence, the optimization and threshold adjustments have honed the models' ability to discern between odorant and odorless molecules effectively. It emphasizes the models' heightened sensitivity and reaffirms the significance of precise threshold tuning in the domain of anomaly detection, where the minority class is of special interest.

4.2 Task 2 results

We first discuss the individual results achieved by the 7 different models on their own, and then compare those results to the ones using Decision Fusion.

Model	Accuracy	Precision	Recall
Descriptors	0.01	0.29	0.34
Fingerprints	0.01	0.30	0.33

Table 4: Binary Relevance Results part 1

Model	F1-score	Jaccard	Hamming
Descriptors	0.29	0.21	0.04
Fingerprints	0.29	0.20	0.04

Table 5: Binary Relevance Results part 2

4.2.1 Results Binary Relevance

After performing all the previously mentioned experiments, trying Binary Relevance and Classifier Chain models with different base classifiers, we find that, surprisingly enough, the final models that achieved the best results according to the tests performed for each set were the same: the Binary Relevance method with the XGBoost base Classifier and using class weights. In particular the results can be observed in Tables 4 and 5.

We observe that the Jaccard Score for both models is 0.21 and 0.20, which are the highest out of all different experiments. We can also observe that some of the other metrics also obtain the highest results. It is particularly interesting to notice that Recall is 0.34 and 0.33, which, considering how restrictive Recall is about perfectly matching solutions, is quite high.

The full tables of the results of all experiments can be found in the Appendix in Table B.13, Table B.14, Table B.15, and Table B.16.

It is important to notice that the results of the same base classifiers between Binary Relevance and Classifier Chains were almost identical or pretty similar. But considering that the implementation of Classifier Chains is more complex, and that the training time is higher, the models using Binary Relevance were deemed as the best option.

4.2.2 Results Neural Network

As mentioned in an earlier section, the manually tunned Neural Network measured the performances of 9 different NN model configurations, with different number of layers and different hyperparameters (see Table 1). The results are visible on Table 6.

Model	Loss	Recall	Precision	Acc.
Model 1	0.030	0.179	0.397	0.167
Model 2	0.025	0.131	0.447	0.140
Model 3	0.029	0.154	0.406	0.161
Model 4	0.031	0.188	0.385	0.176
Model 5	0.039	0.197	0.377	0.153
Model 6	0.030	0.164	0.368	0.159
Model 7	0.023	0.127	0.447	0.142
Model 8	0.024	0.113	0.444	0.159
Model 9	0.027	0.154	0.415	0.195

Table 6: Performances of basic NN models, trained on the descriptors

Table 6 illustrates that the metrics across various models are comparable, indicating minimal variation in performance among the different models.

The loss used is the focal loss [30]. Focal loss is a type of loss function designed to address class imbalance by focusing more on hard-to-classify examples. It modifies the standard cross-entropy loss so that it down-weights the loss assigned to well-classified examples. This approach helps in enhancing model performance on imbalanced datasets by prioritizing learning from a minority class or more complex examples. Our models aim to minimize the loss. The values obtained are below 0.03, so the losses are satisfactory for us.

The observed Recall, of approximately 0.17, is relatively low. This is attributed to the model's probability-based output for the presence or absence of each odor (Figure 5), leading to results that are not strictly binary (1 or 0) as in the training dataset. We will explore potential strategies to mitigate this issue in a subsequent section.

Regarding Precision, the models demonstrate a rate of around 0.42. This indicates that approximately 42% of the odors predicted by the models are correct. Although a higher Precision is desirable, the limited volume of data available constrains the models' ability to yield more accurate predictions.

In terms of Categorical Accuracy, which assesses the model's performance in correctly classifying each instance into one of several categories, our model exhibits a notably low score of around 0.17. This low accuracy is partly due to the model's prediction mechanism, which outputs probabilities instead of binary values (Figure 5). Consequently, many instances with high probabilities (e.g., 0.99) are not classified as positive. We will discuss a potential resolution to enhance this aspect of the model's performance in the following paragraph.

In reviewing the previously discussed metrics, it was observed that certain values were sub-optimal for the desired performance of our Neural Network. As mentioned, this is primarily due to the model's probabilistic prediction approach. To address this, we use the proposed method earlier, where only the top 3 predictions, based on their probability values, are classified as positive, while the remaining are considered negative. These top 3 odors are then compared with the actual expected odors. This approach, however, is a compromise, as it does not account for scenarios where the number of actual odors differs from 3.

Because of this, we have developed a novel similarity metric to evaluate the congruence between the 3 predicted odors and the expected odors in the NN models. This metric operates as follows:

- It counts the number of odors that match between the predicted and expected sets.
- The metric calculates the proportion of these matches relative to the number of actual odors, up to 3.
- By basing the proportion on up to 3 odors, this metric mitigates some of the inaccuracies inherent in a model that rigidly predicts 3 odors.

This new metric offers a more nuanced assessment of the model's predictive accuracy in the context of our specific problem domain.

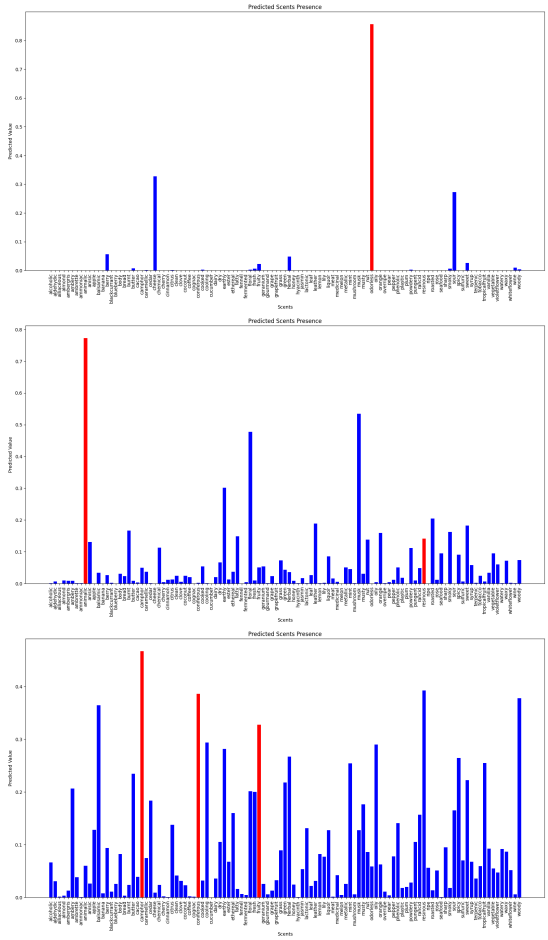


Figure 5: Examples of the probabilistic prediction for different odors. The red columns represent the true odors.

Model	Similarity
Model 1	0.3905
Model 2	0.3964
Model 3	0.3776
Model 4	0.3930
Model 5	0.3790
Model 6	0.3724
Model 7	0.3832
Model 8	0.3891
Model 9	0.3895

Table 7: Similarity between the real and predicted odors for each model, using the top 3 predictions.

See the similarity between the real and predicted odors for each model, using the top 3 predictions on Table 7. As observed, it gives a better view of the accuracy of each model, and allows a better comparison between them.

On Table 7, it is visible that the second model (see Table 1) has the best performance. This model will be used for the Decision Fusion of *Task 2*.

In addition to the bespoke similarity metric, we also calculated standard performance metrics, including Accuracy, Precision, Recall, F1-Score, Hamming Loss, and the Jaccard Score, specifically for the predictions involving the top 3 odors. As explained on the Evalaution section, we pay special attention to the Jaccard Score. The results of these computations are detailed in Table B.17, located in the Appendix of this document.

Table B.17 demonstrates an overall improvement in results compared to Table 6, attributed to the implementation of our novel approach.

The observed Accuracy in the data is notably low, a phenomenon we hypothesize to be a consequence of the limited and potentially substandard quality of the dataset. This hypothesis is supported by the fact that subsequent models did not exhibit a significant increase in Accuracy.

Regarding Precision, the values are comparable, but lower to those previously obtained. This consistency is likely due to our method of

Model	Jaccard Score
Model 1	0.2278
Model 2	0.2212
Model 3	0.2198
Model 4	0.2188
Model 5	0.2114
Model 6	0.2140
Model 7	0.2292
Model 8	0.2287
Model 9	0.2192

Table 8: Jaccard score of basic NN models, with top 3 prediction

focusing on the top 3 predicted odors, which may not always include the expected odors. In contrast, our earlier approach did not limit the number of predicted odors, allowing for a broader range of high-probability odors.

The Recall metric showed a marked improvement with the new methodology, now at approximately 0.31. This enhancement is due to the binary nature of odor predictions in the new approach, enabling the model to more accurately differentiate between the presence and absence of odors. While about 31% of the predicted odors are correct, there remains room for improvement, which could be achieved with higher-quality datasets.

The F1-Score, also at around 0.31, indicates a modest level of effectiveness in balancing Precision and Recall. This score suggests the model's limited Accuracy in correctly identifying True Positives, underlining the need for further refinement in accurately classifying positive cases while reducing False Positives and Negatives. Further research can be done to investigate alternative models to potentially improve these metrics.

Finally, the Hamming Loss, which is quite low at approximately 0.037, is satisfactory given the dataset's quality. This metric reflects the average frequency of incorrect label predictions across all classes and samples, suggesting a relatively low error rate in our model's label predictions.

Table 8 presents the Jaccard scores for each NN configuration, employing our revised prediction methodology. The Jaccard Score is a critical metric for our models, providing insights into the degree of correspondence between the actual and predicted odors. This score bears resemblance to our custom-developed similarity metric, but with a notable distinction: it does not incorporate the error correction tailored to the top 3 prediction issue. Consequently, the Jaccard Scores are comparatively lower for each NN configuration. The highest Jaccard Score achieved with our dataset is 0.2287, which, while not exceptionally high, reflects the limitations of the top 3 prediction and the data available for this analysis.

In addressing the inaccuracies stemming from the probabilistic nature of our NN model, another proposed solution involved the implementation of a threshold. This approach was initially considered but eventually not pursued due to the variability in confidence levels across different predictions. The initial plan was to set the threshold at 0.5. However, as illustrated in Figure 5, there are instances where the highest probability of an odor being present falls below 0.5, yet the odor is pertinent to our prediction. Lowering the threshold to accommodate such cases would inadvertently include irrelevant odors in scenarios where the average confidence level is high. This dilemma led to the decision against the implementation of a fixed threshold, highlighting the challenge in balancing sensitivity and specificity within the probabilistic framework of our model.

For comprehensive visualizations and comparative analyses of the predictions generated by each model, refer to the Appendix. Detailed figures, from Figure B.11 to Figure B.15, provide in-depth illustrations of these model outputs.

4.2.3 Results Otpimized NN

KerasTuner was used to try and optimize the number of neurons in each dense layer, and the choice of activation function in each layer. Normally, in every layer, the ReLu activation function would be used, as it counters the vanish-

Parameter	Value
Objective	val loss
Max epochs	50
Factor	3
Overwrite	true

Table 9: Parameters used for tuner

ing gradient problem the best, but we decided it was interesting to test if the use of different activation functions in this specific problem could lead to better results. In Table 9 the parameters and values used to initialize the tuner can be found. They were kept the same for both models (descriptors and fingerprints). Note that Validation **Loss** was used for the objective instead of Validation **Accuracy**, since it was already clear that Accuracy was not a good metric for this task, and therefore optimizing on it was not a good idea.

The results of the hyperparameter search for both models can be found in Table 10. The values resulting from the tuner were the combination that gave the lowest loss on the validation data. There is a chance that this does not generalize to the test data, and, therefore, even after optimization, it might still not be the best model. The low amount of data also makes it hard to create a model that is able to generalize well, since there is a high chance that there exists data in the test set that is not similar to the training data.

The values of the hyperparameters seems to be quite similar, but the model using the fingerprints as input uses less neurons in the first 3 layers than the model using the descriptors. A potential explanation could be that it is trying to reduce the amount of incoming information more to prevent over-fitting and generalize better, since the size of the input vector is larger than that of the descriptors. However, an explanation for the use of Sigmoid instead of TanH in the 3rd dense layer is hard to give.

After optimization, the models were created with the set of optimal hyperparameters and then trained. The results can be found in Tables 11 and 12.

The results show a similar performance for

	Hyperparameter	Value
Descriptors	Input units	352
	Dense 1	96
	Dense 2	448
	Dense 3	512
	Activation input	ReLu
	Activation dense 1	TanH
	Activation dense 2	ReLu
	Activation dense 3	TanH
Fingerprints	Input units	256
	Dense 1	64
	Dense 2	320
	Dense 3	512
	Activation input	ReLu
	Activation dense 1	TanH
	Activation dense 2	ReLu
	Activation dense 3	Sigmoid

Table 10: Hyperparameter values after optimization

	Accuracy	Precision	Recall
Descriptors	0.00	0.33	0.37
Fingerprints	0.00	0.31	0.36

Table 11: Performance metrics of optimized NN models part 1

	F1-score	H. Loss	Jaccard
Descriptors	0.33	0.04	0.22
Fingerprints	0.31	0.04	0.21

Table 12: Performance metrics of optimized NN models part 2

both models, with the model using the descriptors having a slightly higher Jaccard Score. Comparing these results with the results in Table 8 shows that optimizing did not give any performance increase. It is possible that the small amount of highly unbalanced data doesn't allow for any optimization. More in depth research for optimization could be done, but was not feasible in the project's time frame.

4.2.4 Results Graph NN

A variety of configurations of Graph Neural Networks were explored, with differing results. For the purpose of result analysis, only the results of the best performing model are included in this section, while the remaining results can be found in the Appendix B.1.

The resulting metrics of the output of the GNN can be found in Table 13. We can see the GNN performs worse in terms of Recall an Categorical accuracy, though it achieves higher Precision compared to the the results of the NN shown in a previous section.

Data	Recall	Precision	Accuracy
Train	0.0341	0.5515	0.1399
Val	0.0262	0.6071	0.1319
Test	0.0235	0.4615	0.1551

Table 13: Results of the output of the GNN

Next we can observe the resulting metrics for the top 3 odor labels output by the model, shown in Table 14. We can see that the results are quite similar to the other models.

F1-score	Hamming Loss	Jaccard
0.29	0.04	0.19

Table 14: Metrics for the top 3 labels output of the GNN model on the test set.

4.2.5 Results High-Level Data Fusion

Since there could be cases where some odors are tied on a position, it is important to have some sort of ordering of what model should break the tie with its decision. Based on the performance from the individual models, we have obtained this ordered list of models to consider for tie-breaking:

1. Optimized NN (Descriptors)
2. NN (Descriptors)
3. Optimized NN (Fingerprints)
4. Binary Relevance (Descriptors)
5. Binary Relevance (Fingerprints)
6. Graph NN
7. NN (Fingerprints)

Therefore, if a tie occurred, it would be broken by taking the predicted odor that came from the highest ranked model in this list.

Task 2 Model	Accuracy	Precision
BR (Descr)	0.01	0.29
BR (Fing)	0.01	0.30
NN (Descr)	0.00	0.32
NN (Fing)	0.01	0.29
Opt NN (Descr)	0.00	0.33
Opt NN (Fing)	0.00	0.31
Graph NN	0.00	0.29
Decision Fusion	0.01	0.36

Table 15: Results of individual models and decision fusion task 2 (a)

Recall	F1-score	Jaccard	Hamming Loss
0.34	0.29	0.21	0.04
0.33	0.29	0.20	0.04
0.36	0.32	0.22	0.04
0.32	0.28	0.19	0.04
0.37	0.33	0.23	0.04
0.36	0.31	0.21	0.04
0.33	0.29	0.19	0.04
0.41	0.36	0.26	0.03

Table 16: Results of individual models and decision fusion task 2 (b)

Tables 15 & 16 show the full results of the individual models for *Task 2*, comparing them with each other, as well as with the addition of the Decision Fusion technique explained earlier.

As shown, with the implementation of Majority Voting, the final model shows an improvement on all metrics compared to the individual predictions. Therefore, these results show that Decision Fusion in this case was a good idea.

5 Label Handling

A persistent obstacle in the field of Machine-Learning-based olfactometry (ML-GCO) is the excessive number of odor descriptors (ODs). The models can be trained on the different 109 smells present in the given data, but applying label handling can improve the performance of these models, by reducing the number of ODs.

We investigate 4 different label handling options that have been proposed in ML-GCO related papers, and discuss how, and if, to use them.

The first option is a clustering of odors proposed in [31], where 265 representative ODs are classified into 20 categories using a co-occurrence Bayesian embedding model and investigated with Hierarchical Clustering Analysis. The good thing about this option is that it reduces the number of target variable values from 109 to 20. The problem it introduces is that the clusters can be very generic, for example you can have "plastic" and "earthy" in the same cluster. Therefore, the results can be ambiguous and hard to interpret.

The second option is to use the same 55 odors used in [5]. Since the original data has 109 ODs, we have 2 strategies for the remaining odors not present in the 55 given in the paper. The first approach is to manually classify the remaining 54 ODs as one of the 55 from the paper. The problem with this approach is that it introduces a lot of human errors and biases. The second approach is to drop the 54 ODs. The downside is that the dataset is reduced to 84% of the original data.

The third option uses the same strategy as the previous one but using the smells from another paper [4], which only considers 20 ODs. The same problem of manually having to classify the remaining odors still occurs here, but now there are much more odors to classify, which introduced too many possible errors and biases, and the data reduction is also too much.

The final fourth option is one proposed by the team writing this paper, which is to use the top 30 ODs with higher representation and filter the data to only consider these odors, since the remaining 79 have almost no representation in the data. This results in a dataset with 90% of the original data, which is really good considering that there are only 30 possible values to predict instead of 109.

Additionally, the group also performed frequent item-set mining using the Apriori algorithm to see if we could find smells that always appear together in order to group odors into groups. We found that frequent item-set mining didn't result in anything meaningful, so this approach was discarded.

Unfortunately, due to time limitations and unforeseeable problems with code development and integration of the models, these label handling options were not tested on the final models for *Task 2*. Nevertheless, we included this section on the report so that any future person who continues this line of research can look into testing them, following the notes and instructions on the deliverable "LabelHandling.ipyng" file, where new target variables with the label handling options applied are created.

6 Conclusions

Regarding the Research Questions initially stated, the results of this project demonstrate that a Deep Learning model can in fact distinguish odorant from non-odorant molecules, although not to the fullest extent. Additioanlly, we have seen that several Machine Learning and Deep Learning models can assign correct odor descriptors to molecules, to some degree, but the limited available data, in combination with the high amount of labels, complicated both tasks. Finally, the results indicated that different models perform better/worse with respects to different metrics, and that Data Fusion, in particular High-Level Data Fusion, improves the performance of the models for both tasks.

In conclusion, we are content with the outcomes of our project. Although the results did not yield highly accurate odor predictions, the project successfully met its primary objectives: exploring, studying, and implementing various predictive modeling approaches.

This endeavor, being our inaugural project in the master's program, has significantly broadened our knowledge base, which will undoubtedly be beneficial in our future endeavors. Participating in this project marked the first time many of us independently applied AI models in a practical setting, providing a valuable learning experience. Despite encountering challenges and making mistakes, these experiences have been instrumental in our learning process, offering lessons that will be vital for future projects.

Finally, selecting this particular subject matter proved to be a wise decision. It allowed us to gain extensive insight into various predictive modeling techniques, and provided an excellent practical exercise in researching and applying these methods. The knowledge and experience gained through this project have been immensely rewarding.

7 Project Members Work Distribution

This is how the work has been distributed between the team members:

- **Christos Koromilas:** Data Preparation, Feature Extraction, Task 1 Model (One-Class Classification), Data Fusion Task 1.
- **Stephan Possen:** Evaluation Metrics, Optimized NN Models.
- **Mireia Fortuño Ledesma:** Binary Relevance and Classifier Chain Models, Label Handling Ideas, Data Fusion Task 2.
- **Lázár Dávid Barta:** Neural Networks Models, Custom Similarity Metric.
- **Mathias Verschueren:** Graph NN Model.

Apart from this, all the members have worked together on the report and presentations, and helped each other when needed.

References

[1] Michael Czerny et al. "The influence of molecular structure on odor qualities and odor detection thresholds of volatile alkylated phenols". In: *Chemical Senses* 36.6 (2011), pp. 539–553.

[2] European Commission. *General Legislation Food Safety*. URL: https://food.ec.europa.eu/safety/chemical-safety/food-contact-materials/legislation_en#:~:text=The%20principles%20set%20out%20in,odour%20in%20an%20unacceptable%20way.

[3] Wynne Parry. "AI Predicts What Chemicals Will Smell like to a Human". In: *ScientificAmerica* (2022).

[4] Andreas Keller et al. "Predicting human olfactory perception from chemical features of odor molecules". In: *Science* 355.6327 (2017), pp. 820–826. DOI: 10.1126/science.aal2014. eprint: https://www.science.org/doi/pdf/10.1126/science.aal2014. URL: https://www.science.org/doi/abs/10.1126/science.aal2014.

[5] Brian K. Lee et al. "A principal odor map unifies diverse tasks in olfactory perception". In: *Science* 381.6661 (2023), pp. 999–1006. DOI: 10.1126/science.ade4401. eprint: https://www.science.org/doi/pdf/10.1126/science.ade4401. URL: https://www.science.org/doi/abs/10.1126/science.ade4401.

[6] Karen Kreeger. "AI Cracks the Code on Odor Perception". In: *Neuro-Science News* (2023). URL: https://neurosciencenews.com/odor-perception-ai-23858/.

[7] URL: rdkit.org.

[8] *Molecular fingerprints and similarity searching*. URL: https://openbabel.org/docs/dev/Fingerprints/intro.html#:~:text=Molecular%20fingerprints%20are%20a%20way,particular%20substructures%20in%20the%20molecule..

[9] Shifa Zhong and Xiaohong Guan. "Count-Based Morgan Fingerprint: A More Efficient and Interpretable Molecular Representation in Developing Machine Learning-Based Predictive Regression Models for Water Contaminants' Activities and Properties". In: *Environmental Science & Technology* 0.0 (2023). PMID: 37406199, p. 10. DOI: 10.1021/acs.est.3c02198. eprint: https://doi.org/10.1021/acs.est.3c02198. URL: https://doi.org/10.1021/acs.est.3c02198.

[10] Greg Landrum. URL: https://greglandrum.github.io/rdkit-blog/posts/2022-12-23-descriptor-tutorial.html.

[11] Patrick Reiser et al. "Graph neural networks for materials science and chemistry". In: *Communications Materials* 3.1 (Nov. 2022), p. 93. ISSN: 2662-4443. DOI: 10.1038/s43246-022-00315-6. URL: https://doi.org/10.1038/s43246-022-00315-6.

[12] Alexander Kensert, Gert Desmet, and Deirdre Cabooter. "MolGraph: a Python package for the implementation of small molecular graphs and graph neural networks with TensorFlow and Keras". In: *arXiv preprint arXiv:2208.09944* (2022).

[13] Chong Zhou and Randy C Paffenroth. "Anomaly detection with robust deep autoencoders". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 665–674.

[14] Kartik Nooney. *Deep dive into multi-label classification..! (With detailed Case Study)*. URL: https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff.

[15] Kushagra Saini and Venkatnarayan Ramanathan. "Predicting odor from molecular structure: a multi-label classification approach". In: *Scientific Reports* 12.1 (Aug. 2022). DOI: 10.1038/s41598-022-18086-y. URL: https://doi.org/10.1038/s41598-022-18086-y.

[16] Liang Shang et al. "Artificial intelligence-based gas chromatography-olfactometry for sensory evaluation of key compounds in food ingredients". In: *bioRxiv* (2022). DOI: 10.1101/2022.04.20.488977. eprint: https://www.biorxiv.org/content/early/2022/06/06/2022.04.20.488977.full.pdf. URL: https://www.biorxiv.org/content/early/2022/06/06/2022.04.20.488977.

[17] Rinu Chacko et al. "Data based predictive models for odor perception". en. In: *Sci. Rep.* 10.1 (Oct. 2020), p. 17136.

[18] Marek Gagolewski. "Data fusion: theory, methods, and applications". In: *arXiv preprint arXiv:2208.01644* (2022).

[19] Marina Cocchi. "Chapter 1 - Introduction: Ways and Means to Deal With Data From Multiple Sources". In: *Data Fusion Methodology and Applications*. Ed. by Marina Cocchi. Vol. 31. Data Handling in Science and Technology. Elsevier, 2019, pp. 1–26. DOI: https://doi.org/10.1016/B978-0-444-63984-4.00001-6. URL: https://www.sciencedirect.com/science/article/pii/B9780444639844000016.

[20] D. Ballabio, R. Todeschini, and V. Consonni. "Chapter 5 - Recent Advances in High-Level Fusion Methods to Classify Multiple Analytical Chemical Data". In: *Data Fusion Methodology and Applications*. Ed. by Marina Cocchi. Vol. 31. Data Handling in Science and Technology. Elsevier, 2019, pp. 129–155. DOI: https://doi.org/10.1016/B978-0-444-63984-4.00005-3. URL: https:

//www.sciencedirect.com/science/article/pii/B9780444639844000053.

[21] URL: https://www.aicrowd.com/challenges/learning-to-smell.

[22] URL: http://scikit.ml/api/skmultilearn.problem_transform.br.html.

[23] URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[24] URL: https://xgboost.readthedocs.io/en/stable/python/python_api.html.

[25] URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[26] URL: http://scikit.ml/api/skmultilearn.problem_transform.cc.html.

[27] URL: https://www.tensorflow.org/guide/keras.

[28] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[29] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[30] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[31] Liang Shang et al. "Machine-Learning-Based Olfactometry: Odor Descriptor Clustering Analysis for Olfactory Perception Prediction of Odorant Molecules". In: *bioRxiv* (2022). DOI: 10.1101/2022.04.20.488973. eprint: https://www.biorxiv.org/content/early/2022/04/22/2022.04.20.488973.full.pdf. URL: https://www.biorxiv.org/content/early/2022/04/22/2022.04.20.488973.

Appendices

A Task 1

A.1 Binary and One-Class Classification

Feature Set	Model	Accuracy	Precision 0 / 1	Recall 0 / 1	F1-Score 0 / 1
Fingerprints	SVC	0.85	0.99 / 0.06	0.86 / 0.67	0.92 / 0.11
	Binary DL	0.97	0.99 / 0.27	0.98 / 0.33	0.98 / 0.3
Descriptors	SVC	0.90	1.00 / 0.10	0.90 / 0.75	0.95 / 0.17
	Binary DL	0.97	0.99 / 0.33	0.99 / 0.33	0.99 / 0.33
Fused	SVC	0.91	0.99 / 0.10	0.91 / 0.67	0.95 / 0.17
	Binary DL	0.98	0.99 / 0.75	1.00 / 0.33	0.99 / 0.46

Table A.1: Binary Classification (balanced weights)

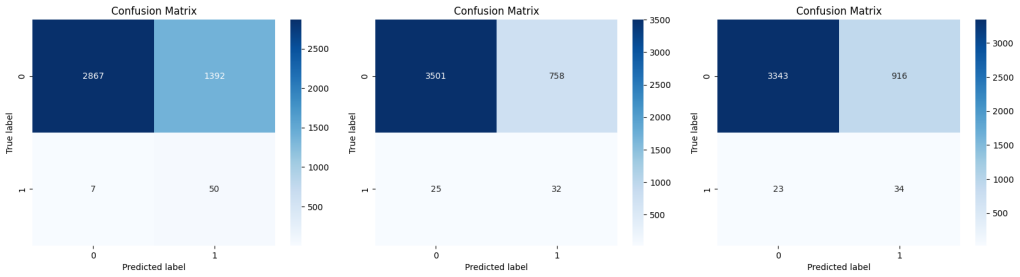
Feature	Model	Accuracy	Precision 0 / 1	Recall 0 / 1	F1-Score 0 / 1	AUC
Fingerprints	OC-SVM	0.50	0.99 / 0.01	0.50 / 0.51	0.67 / 0.03	0.47
	Autoencoder	0.86	0.99 / 0.04	0.86 / 0.44	0.92 / 0.08	0.75
Descriptors	OC-SVM	0.50	1.00 / 0.02	0.50 / 0.95	0.66 / 0.05	0.16
	Autoencoder	0.89	0.99 / 0.05	0.90 / 0.40	0.94 / 0.09	0.82
Fused	OC-SVM	0.50	0.99 / 0.01	0.50 / 0.56	0.66 / 0.03	0.42
	Autoencoder	0.86	0.99 / 0.04	0.86 / 0.47	0.92 / 0.08	0.78

Table A.2: One-Class Classification

A.2 Autoencoders

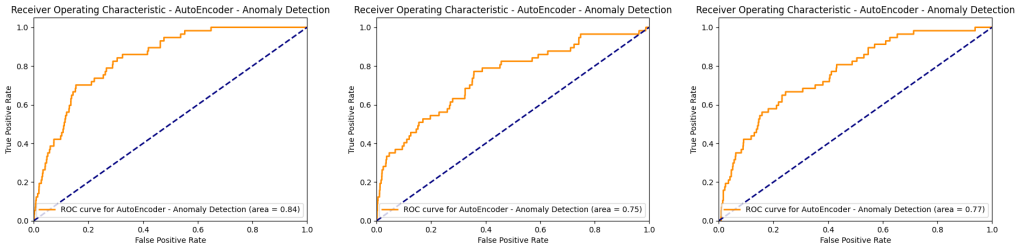
Model	Accuracy	Precision	Recall	F1-Score	AUC	Threshold
Descriptors	0.89	0.05	0.46	0.10	0.88	0.0033
Fingerprints	0.87	0.05	0.46	0.08	0.75	0.0067
Fused	0.86	0.04	0.46	0.08	0.77	0.0057
Bayesian Decision	0.80	0.04	0.61	0.08	0.7100	-
Weighted Voting	0.95	0.10	0.30	0.15	0.6303	-

Table A.3: Performance metrics for Autoencoder models and Fusion Techniques before Keras-Tuner



(a) Descriptors (b) Fingerprints (c) Fused

Figure A.1: Confussion Matrices



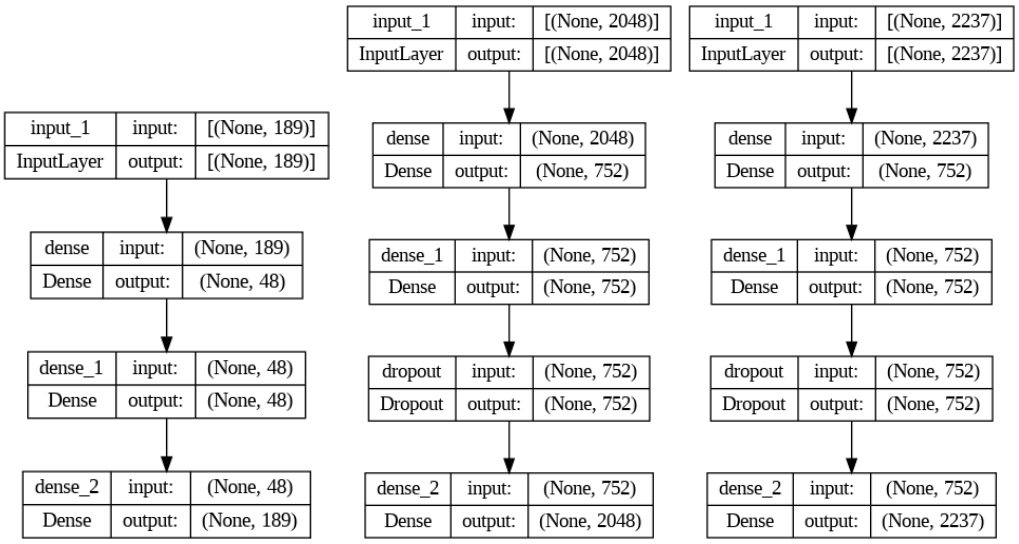
(a) Descriptors (b) Fingerprints (c) Fused

Figure A.2: ROC AUC Curves

A.3 Optimized Hyperparameters

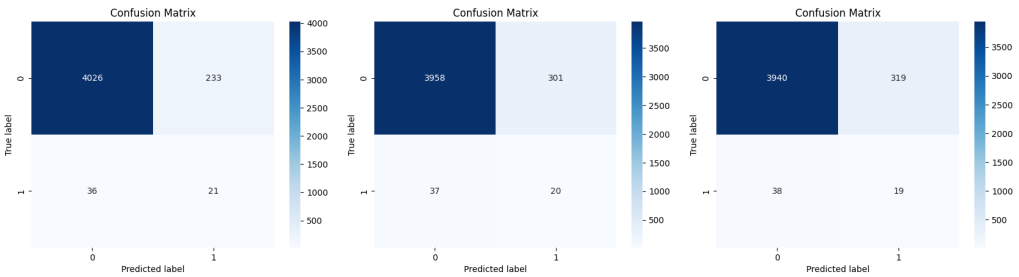
Model	Accuracy	Precision	Recall	F1-Score	AUC	Best Threshold
Descriptors	0.94	0.08	0.37	0.14	0.8463	0.001566
Fingerprints	0.92	0.06	0.35	0.11	0.7366	0.005378
Fused	0.92	0.06	0.33	0.10	0.7585	0.003726
Bayesian Decision	0.89	0.06	0.51	0.11	0.7024	-
Weighted Voting	0.97	0.12	0.21	0.15	0.5946	-

Table A.4: Performance metrics for the best models



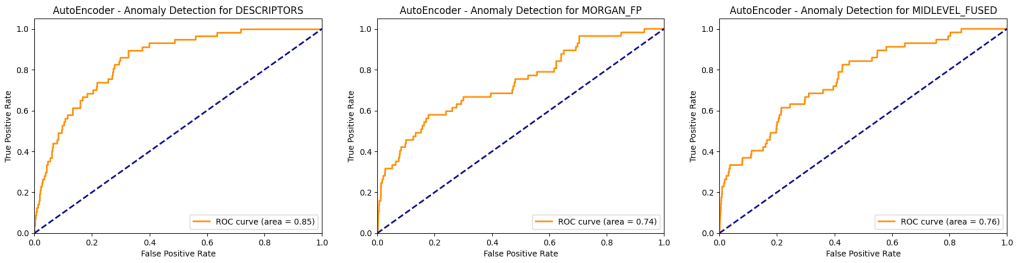
(a) Descriptors Optimized (b) Fingerprints Optimized (c) Fused data Optimized

Figure A.3: Multiple Optimized Architectures



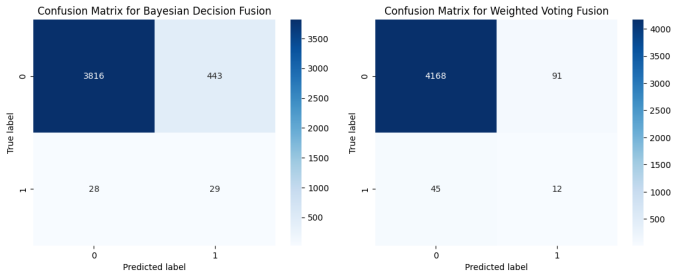
(a) Descriptors (b) Fingerprints (c) Fused

Figure A.4: Confussion Matrices



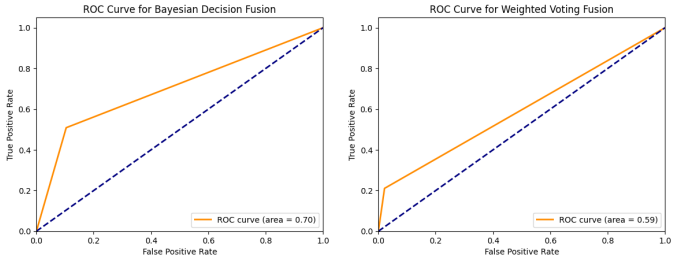
(a) Descriptors (b) Fingerprints (c) Fused

Figure A.5: ROC AUC Curves



(a) Bayesian Decision (b) Weighted Voting

Figure A.6: Confusion Matrices for High-Level Fusion Techniques



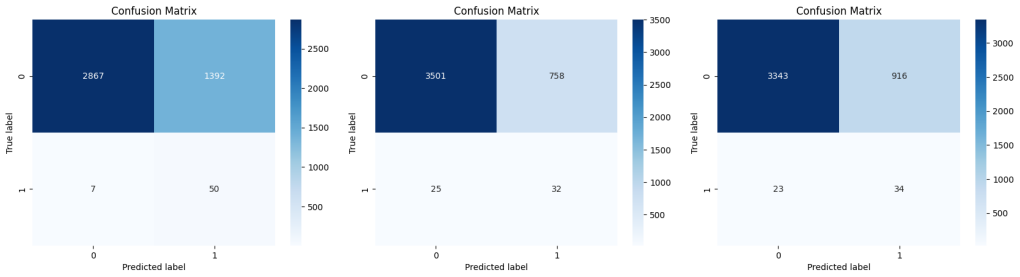
(a) Bayesian Decision (b) Weighted Voting

Figure A.7: ROC AUC Curves for High-Level Fusion Techniques

A.4 Best Threshold

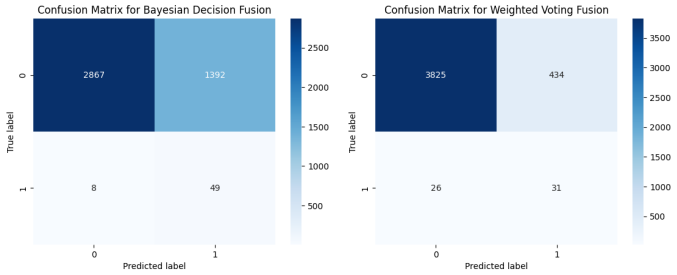
Model	Accuracy	Precision	Recall	F1-Score	AUC	Best Threshold
Descriptors	0.68	0.03	0.88	0.07	0.8463	0.000606
Fingerprints	0.82	0.04	0.56	0.08	0.7366	0.003660
Fused	0.78	0.04	0.60	0.07	0.7585	0.002140
Bayesian Decision	0.68	0.03	0.86	0.07	0.7664	-
Weighted Voting	0.89	0.07	0.54	0.12	0.7210	-

Table A.5: Performance metrics for the best models after threshold adjustment



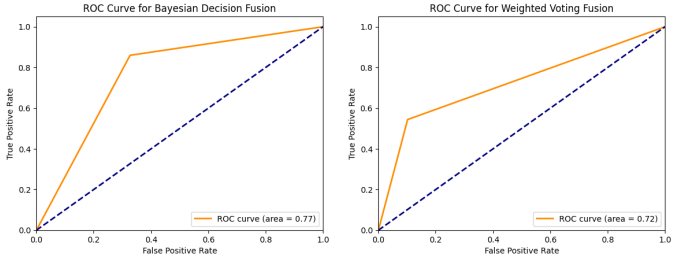
(a) Descriptors (b) Fingerprints (c) Fused

Figure A.8: Confussion Matrices



(a) Bayesian Decision (b) Weighted Voting

Figure A.9: Confusion Matrices for High-Level Fusion Techniques with the best threhold



(a) Bayesian Decision (b) Weighted Voting

Figure A.10: ROC AUC Curves for High-Level Fusion Techniques with the best threhold

B Task 2

B.1 GNN

Atom properties for graph embedding

- Symbol
- Hybridization

- CIPCode
- ChiralCenter
- FormalCharge
- TotalNumHs
- TotalValence
- NumRadicalElectrons
- Degree
- Aromatic
- Hetero
- HydrogenDonor
- HydrogenAcceptor
- RingSize
- Ring
- CrippenLogPContribution
- CrippenMolarRefractivityContribution
- TPSAContribution
- LabuteASAContribution
- GasteigerCharge

Bond properties for Graph embedding

- BondType
- Conjugated
- Rotatable
- Stereo

Layer Type	Size	Activation Function
GCN	128	ReLU
GCN	64	ReLU
Dense	128	ReLU
Dense	64	ReLU
Output	109	Sigmoid

Table B.6: Configuration 1: Layer Specifications

Layer Type	Size	Activation Function
GCN	64	ReLU
GCN	32	ReLU
Dense	128	ReLU
Dense	128	ReLU
Output	109	Sigmoid

Table B.7: Configuration 2: Layer Specifications

Layer Type	Size	Activation Function
GCN	32	ReLU
GCN	32	ReLU
Dense	128	ReLU
Dense	64	ReLU
Output	109	Sigmoid

Table B.8: Configuration 3: Layer Specifications

Layer Type	Size	Activation Function
GCN	128	ReLU
GCN	64	ReLU
GCN	32	ReLU
Dense	128	ReLU
Dense	64	ReLU
Output	109	Sigmoid

Table B.9: Configuration 4: Layer Specifications

Layer Type	Size	Activation Function
GCN	64	ReLU
GCN	32	ReLU
Dense	64	ReLU
Output	109	Sigmoid

Table B.10: Configuration 5: Layer Specifications

Layer Type	Size	Activation Function
GCN	80	ReLU
GCN	40	ReLU
Dense	128	ReLU
Dense	64	ReLU
Output	109	Sigmoid

Table B.11: Configuration 6: Layer Specifications

Data Set	**Train**				**Validation**			
Configuration	**loss**	**recall**	**precision**	**cat_accuracy**	**loss**	**recall**	**precision**	**cat_accuracy**
1	0.088	0.0524	0.5789	0.1529	0.0948	0.0436	0.5495	0.148
2	0.0869	0.0562	0.6179	0.1593	0.0952	0.0428	0.5355	0.1634
3	0.0896	0.0348	0.5823	0.1213	0.0955	0.0271	0.5536	0.1184
4	0.094	0.0144	0.5407	0.1207	0.0972	0.0131	0.5085	0.121
5	0.0882	0.0432	0.5431	0.1378	0.095	0.0393	0.4891	0.1532
6	0.0897	0.032	0.577	0.1336	0.0961	0.0214	0.5	0.1338

Table B.12: Table holding the metrics the different GNN model configurations achieved on our train and testing set. These tests were run to determine the configuration to use for our final model. Note that these tests were run before implementing the Jaccard and Hamming Loss. We made the assumption that a model achieving higher categorical Accuracy would also perform better on the other metrics.

B.2 Binary Relevance and Classifier Chain Results

Base Classifier	Accuracy	Precision	Recall	F1-Score	Jaccard	Hamming Loss
Random Forest	0.01	0.27	0.28	0.27	0.16	0.04
XGBoost	0.01	0.27	0.30	0.27	0.18	0.04
SVC	0.00	0.16	0.28	0.16	0.14	0.04
Random Forest w/ Class weights	0.01	0.29	0.30	0.29	0.19	0.04
XGBoost w/ Class weights	0.01	0.29	0.34	0.29	0.21	0.04
SVC w/ Class weights	0.00	0.21	0.30	0.21	0.18	0.04

Table B.13: Binary Relevance with Descriptors

Base Classifier	Accuracy	Precision	Recall	F1-Score	Jaccard	Hamming Loss
Random Forest	0.01	0.26	0.27	0.26	0.15	0.04
XGBoost	0.01	0.26	0.29	0.26	0.18	0.04
SVC	0.00	0.10	0.19	0.10	0.15	0.04
Random Forest w/ Class weights	0.01	0.28	0.30	0.28	0.18	0.04
XGBoost w/ Class weights	0.01	0.30	0.33	0.29	0.20	0.04
SVC w/ Class weights	0.00	0.17	0.25	0.17	0.18	0.04

Table B.14: Binary Relevance with Fingerprints

Base Classifier	Accuracy	Precision	Recall	F1-Score	Jaccard	Hamming Loss
Random Forest	0.00	0.22	0.24	0.22	0.15	0.04
XGBoost	0.00	0.25	0.28	0.25	0.17	0.04
SVC	0.00	0.10	0.22	0.10	0.10	0.04
Random Forest w/ Class weights	0.01	0.25	0.29	0.25	0.18	0.04
XGBoost w/ Class weights	0.01	0.28	0.32	0.28	0.19	0.04
SVC w/ Class weights	0.00	0.15	0.25	0.15	0.16	0.04

Table B.15: Classifier Chains with Descriptors

Base Classifier	Accuracy	Precision	Recall	F1-Score	Jaccard	Hamming Loss
Random Forest	0.00	0.22	0.24	0.22	0.13	0.04
XGBoost	0.00	0.23	0.27	0.23	0.15	0.04
SVC	0.00	0.09	0.18	0.09	0.13	0.04
Random Forest w/ Class weights	0.01	0.26	0.28	0.26	0.17	0.04
XGBoost w/ Class weights	0.01	0.28	0.31	0.28	0.19	0.04
SVC w/ Class weights	0.01	0.21	0.23	0.21	0.16	0.04

Table B.16: Classifier Chains with Fingerprints

B.3 Performances of basic NN models, with top 3 prediction, trained on the descriptors

Model	Accuracy	Precision	Recall	F1-score	Hamming Loss	Jaccard Score
Model 1	0.0124	0.3216	0.3564	0.3243	0.0369	0.2278
Model 2	0.0145	0.3113	0.3474	0.3149	0.0375	0.2212
Model 3	0.0103	0.3127	0.3472	0.3153	0.0374	0.2198
Model 4	0.0103	0.3092	0.3440	0.3120	0.0376	0.2188
Model 5	0.0062	0.3037	0.3412	0.3076	0.0379	0.2114
Model 6	0.0083	0.3065	0.3405	0.3092	0.0377	0.2140
Model 7	0.0124	0.3223	0.3600	0.3271	0.0368	0.2292
Model 8	0.0124	0.3223	0.3558	0.3246	0.0368	0.2287
Model 9	0.0083	0.3113	0.3464	0.3156	0.0375	0.2192

Table B.17: Performances of basic NN models, with top 3 prediction, trained on the descriptors

B.3.1 Predictions by different simple NN models

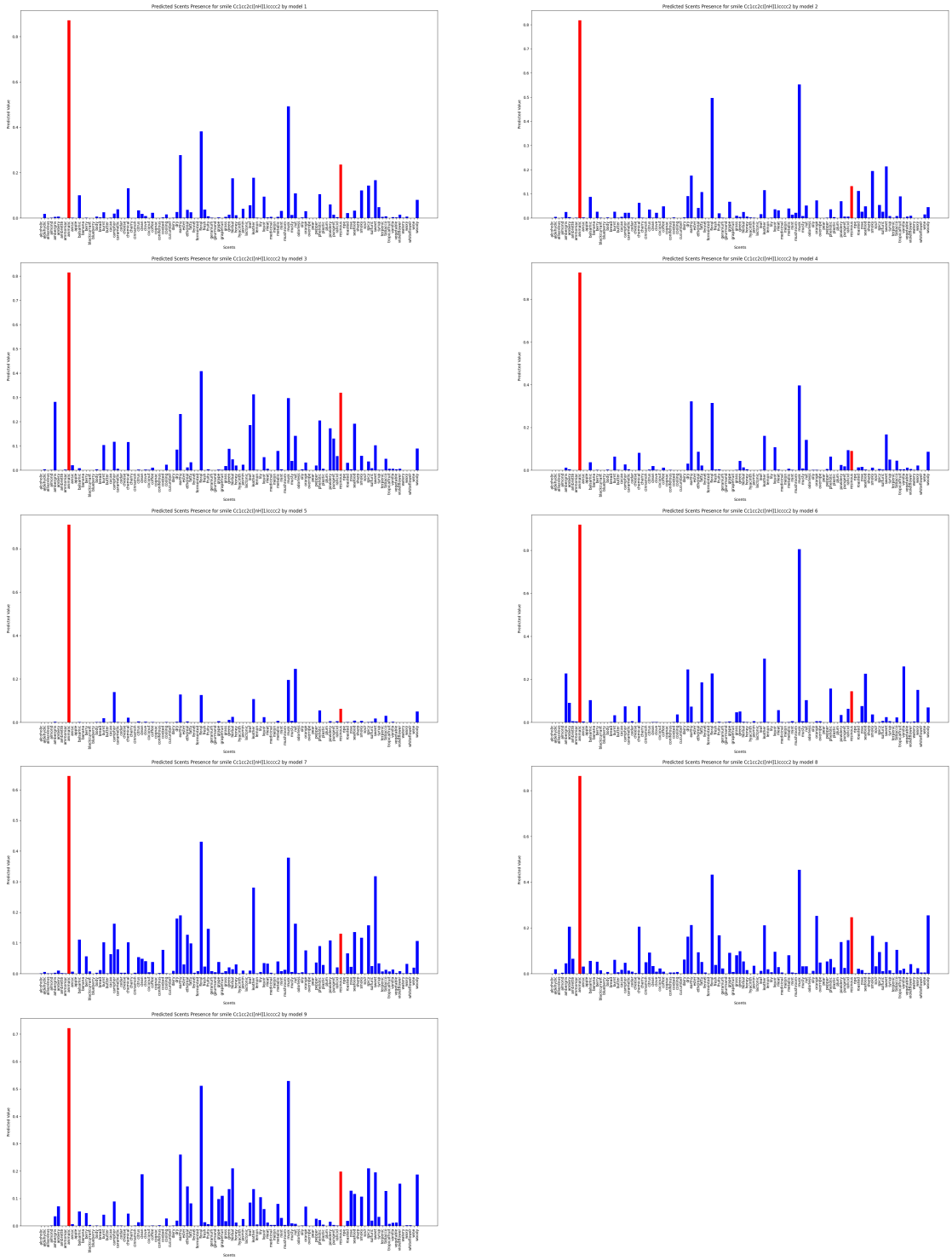


Figure B.11: Predictions of odors by each model for SMILES Cc1cc2c([nH]1)cccc2. The red bars represent the expected odors.

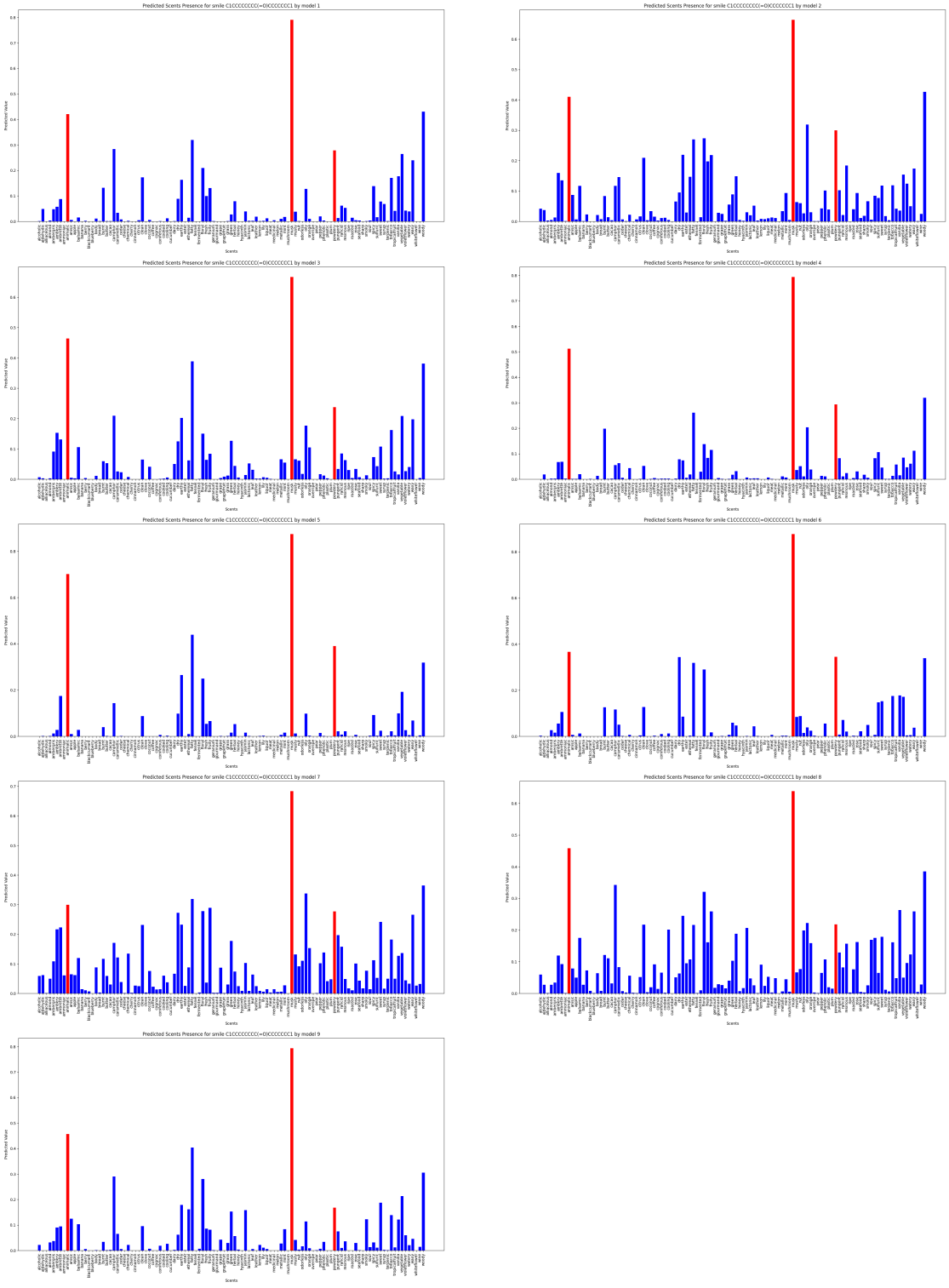


Figure B.12: Predictions of odors by each model for SMILES C1CCCCCCCC(=O)CCCCCCC1. The red bars represent the expected odors.

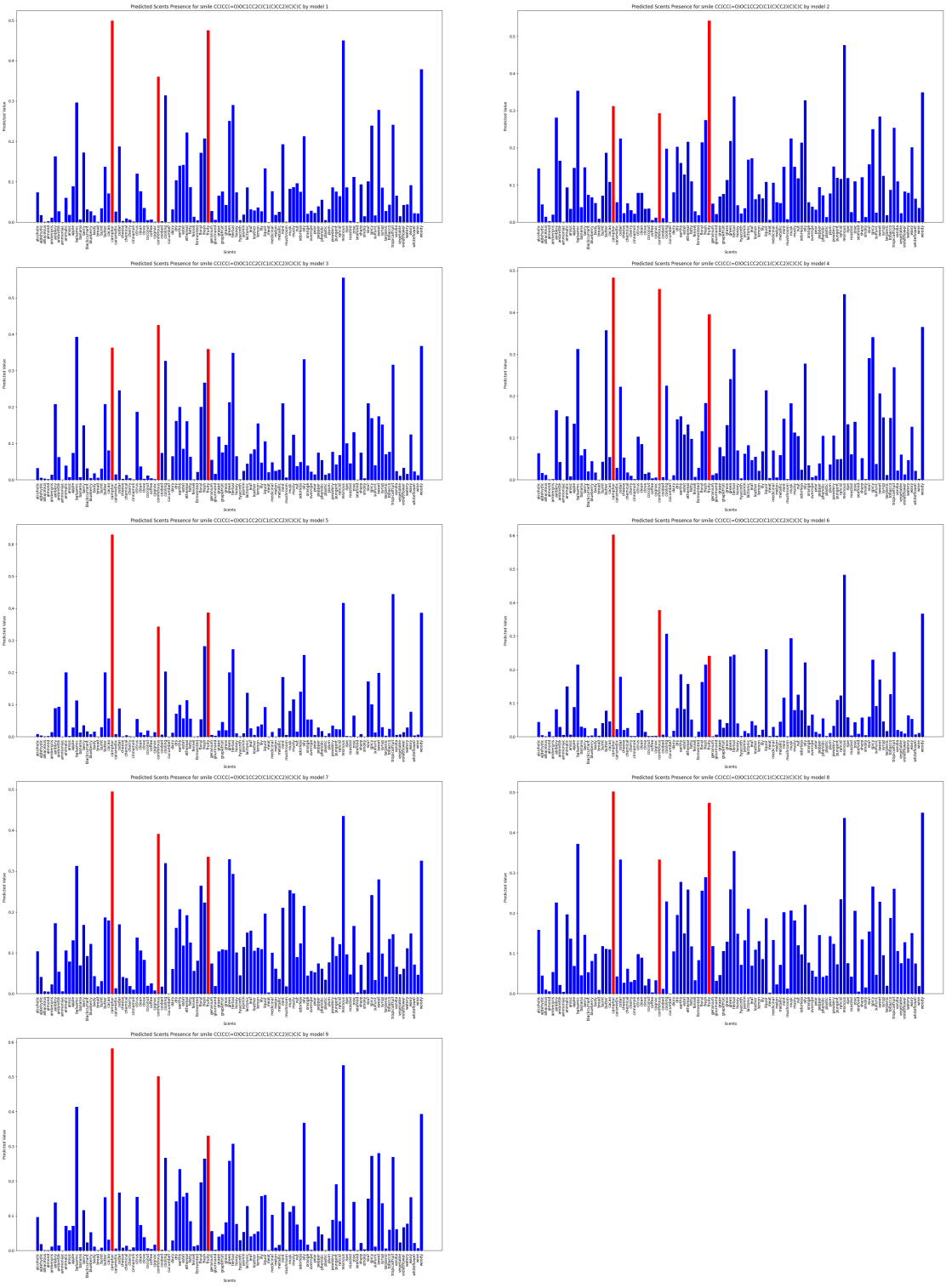


Figure B.13: Predictions of odors by each model for SMILES CC(CC(=O)OC1CC2C(C1(C)CC2)(C)C)C. The red bars represent the expected odors.

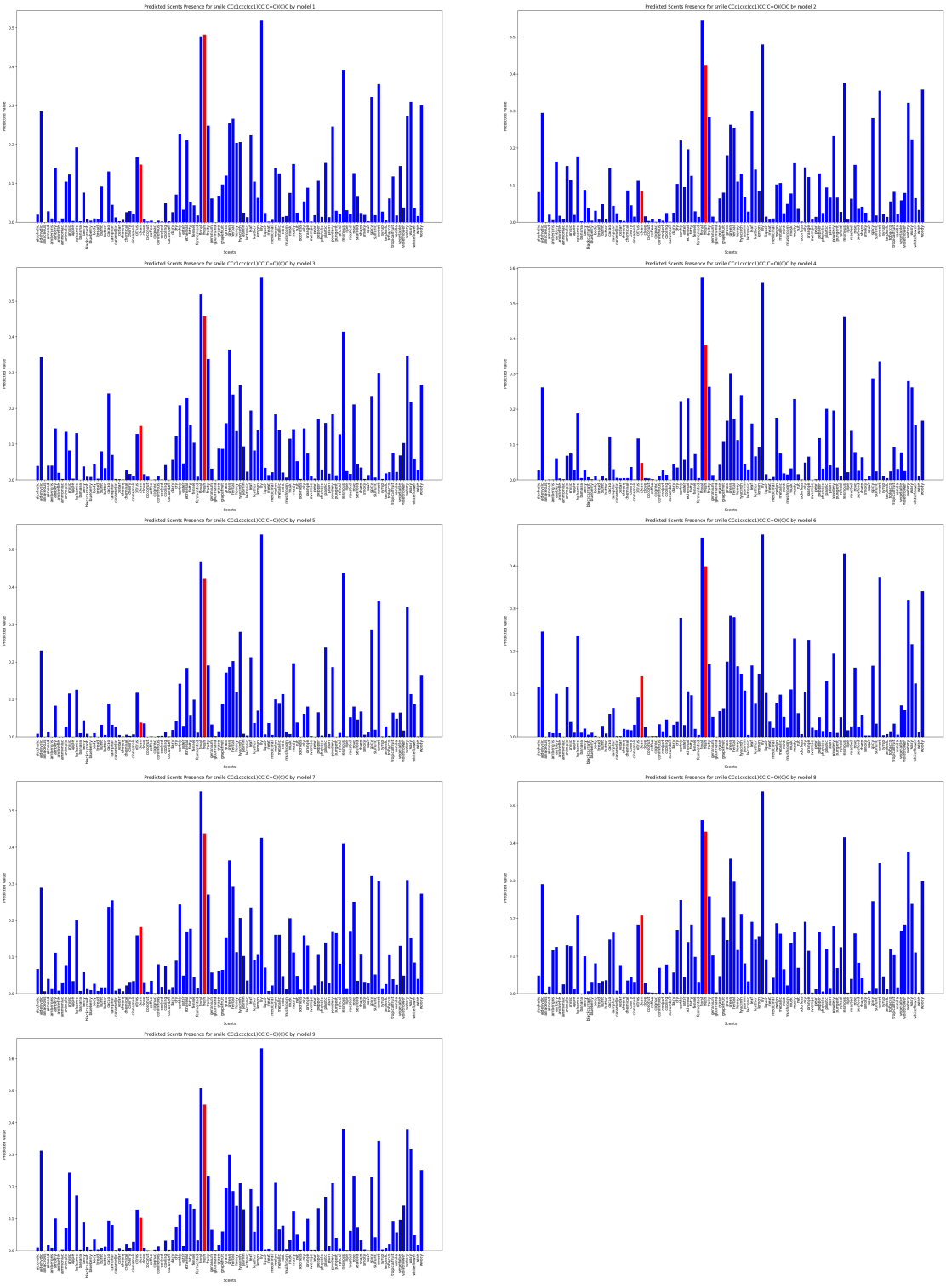


Figure B.14: Predictions of odors by each model for SMILES CCc1ccc(cc1)CC(C=O)(C)C. The red bars represent the expected odors.

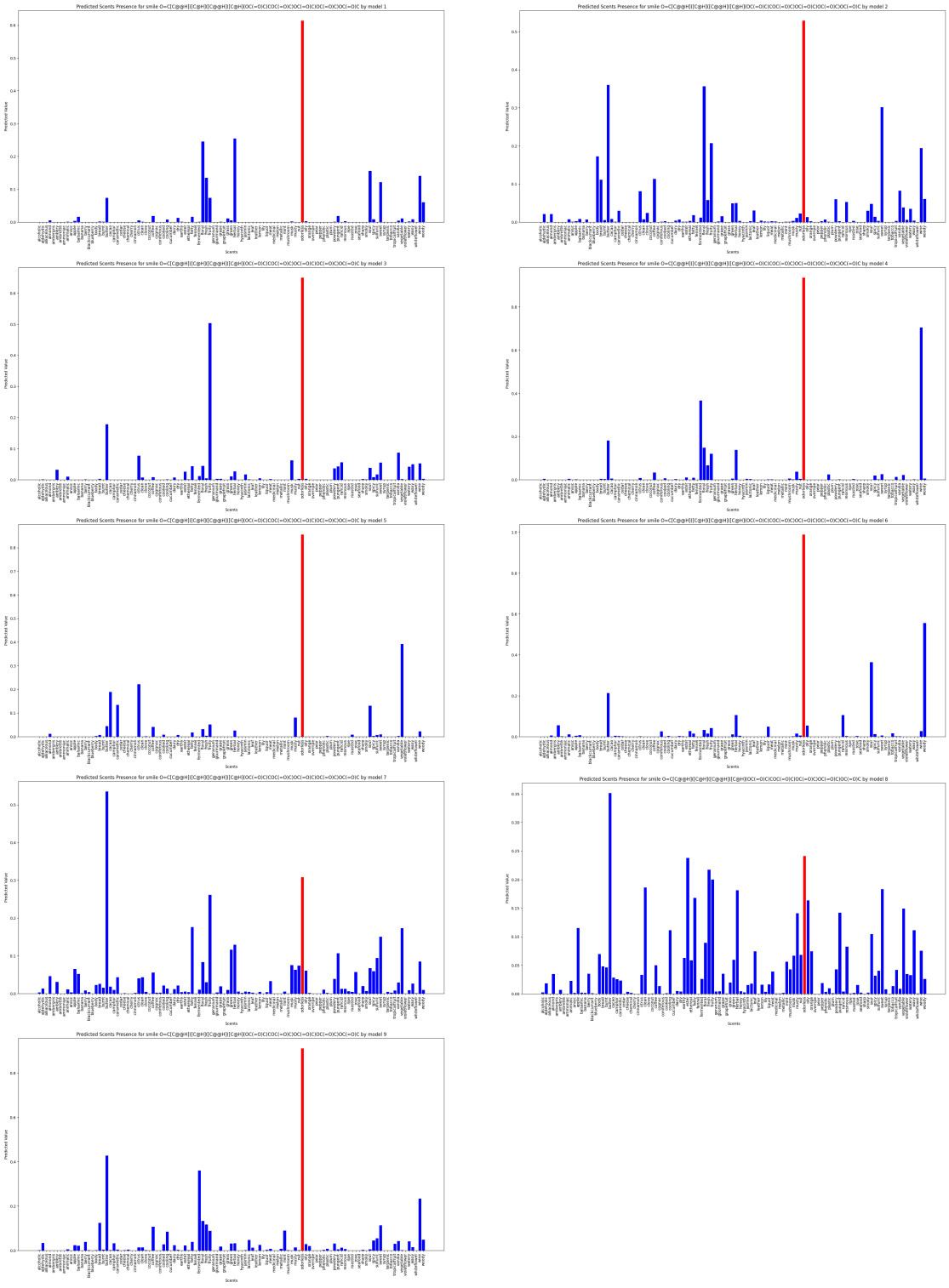


Figure B.15: Predictions of odors by each model for O=C[C@@H]([C@H]([C@@H]([C@H](OC(=O)C)COC(=O)C)OC(=O)C)OC(=O)C)OC(=O)C. The red bars represent the expected odors.

C Data

C.1 Missing values and KNN Imputer

Missing values in the Descriptors's columns were imputed using the KNN imputer due to the relatively small number of missing samples

Descriptor	**NaNs Count**
MaxPartialCharge	2
MinPartialCharge	2
MaxAbsPartialCharge	2
MinAbsPartialCharge	2
BCUT2D_MWHI	12
BCUT2D_MWLOW	12
BCUT2D_CHGHI	12
BCUT2D_CHGLO	12
BCUT2D_LOGPHI	12
BCUT2D_LOGPLOW	12
BCUT2D_MRHI	12
BCUT2D_MRLOW	12

Table C.18: NaNs per Descriptor Column and Imputation Strategy