Notes and Accords, Quantifying Perfume's Ephemerality

Ross Roberts
December 2021

Table of Contents

1 Problem Statement

A perfume is generally built with three layers of notes. These are elementary essential (or synthetic) oils with singular scents. The layers are divided between top notes, middle notes, and base notes. The top notes contain oils with lower molecular weight than oils from the lower layers. Because of this, top notes project off the skin more readily than middle and base notes. These are the first molecules to reach the nose and can be smelled a good distance away from the skin. These are also the first scents to evaporate from the skin entirely, leaving just the impression of the heavier notes. Top notes are usually bright, like citrus fruits and white florals. Base notes are heavy and make up the heart of the fragrance. They cling to the skin and must be sensed up close. Woody scents like cedar, vetiver, and the synthetic compounds Iso E Super, Ambroxan, and Cetalox are often used as base notes. Middle notes fall somewhere between the outer layers. Since the persistence of each note varies in time on the skin, the perfume can be conceptualized as an evolving, ephemeral piece of art, like a musical composition. [1] [2] [3]

Fragrance notes in varying concentrations combine to form accords. If carefully balanced, the accord emerges as an entirely new scent, with its underlying notes virtually indistinguishable from one another. For example, labdanum, benzoin, and Tolu Balsam can combine to form what's called an amber accord. Birch tar, styrax, castoreum, myrtle, and cade can combine to form a leather accord. The accord is a subjective experience. Different master perfumers may use different notes to create similar accords. Additionally, a perfume may contain a multitude of accords, each simultaneously competing with and complementing one another to create a unique, fragrant experience. [4]

This project will explore the relationship between a perfume's constituent notes and their emergent accords. Given a perfume's list of notes and concentrations, can the resulting accords be predicted? Can the target consumer (male, female, or both) be derived from the notes and accords? How does a perfume's makeup influence its popularity? For example, if Channel were to create a new perfume targeted to men, which notes/accords should be selected to maximize sales?

2 Methodology

First, data was sourced from an online perfume encyclopedia. Each perfume record contains information including component notes and accords along with a variety of user ratings. Mutual information scores were calculated between each pair of accords as well as between each note/accord pair and compared against counts of cooccurrence for each of the pairs. Principal component analysis was performed on both the note and accord data to reduce the perfume feature space into 2-dimensional visual representations. These representations were plotted against qualities including target gender and longevity in order to discover any notable correlations. The perfumes were then clustered by accords into six groups using hierarchical agglomerative clustering. The clusters serve as proxies to the six generally accepted styles of perfume and are used to demonstrate multiclass classification.

Four methods of multiclass classification were used to predict the style proxies from the perfume notes, and four methods of multilabel classification were used to predict perfume accords from perfume notes. For both groups of multiclass and multilabel classification methods, two models were inherently multiclass/multilabel: random forest and neural network. In the remaining cases, binary logistic regression models were extended to accommodate multiclass/multilabel classification using one vs. rest / one vs. one (multiclass), and multioutput

/ classifier chaining (multilabel) extension methods. Overviews of multiclass and multilabel classification are provided in sections 4 and 5, respectively. The models were trained on an 80% split of the data while classification performance was measured against the remaining 20% split using a handful of relevant performance metrics defined in section 6.

3 Data Preprocessing and Exploration

3.1 Data Source and Preprocessing

The data source contains 51,212 perfume records collected from fragrantica.com, an online perfume encyclopedia with an active user base [5]. Each perfume entry on the site allows users to vote for myriad fragrance qualities including component notes and accords, a general rating, longevity, and sillage, a subjective measure of a perfume's scent trail left behind in passing [6]. This work selects the perfume name, the target gender (male, female, or unisex), a rating from 1 to 5, the number of votes towards the rating, 5 columns containing the number of votes towards categories indicating longevity (poor, weak, moderate, long, very long), 4 columns containing the number of votes towards categories indicating sillage (soft, moderate, heavy, enormous), a list of notes including indicators for top, middle, and bottom, and a list of accords from each record. All textual data was converted to lowercase and stripped of non-alphanumeric characters to reduce unnecessary duplication in further processing.

Ratings were normalized from 0 to 1. Longevity was quantified by assigning each category a score from 0 to 4 (poor:0, weak:1, moderate:2, long:3, very long:4) and averaging the scores weighted by corresponding vote counts. The weighted averages were then normalized from 0 to 1. Sillage was quantified using a similar method. The lists of notes and accords were transformed into one-hot-encoded matrices with notes/accords as columns and perfumes as rows such that a value of 1 indicates that a note/accord is an element of a perfume. Notes and accords were dropped if fewer than 30 perfumes contained them, and perfumes were dropped if they did not contain any notes or accords.

The final data set was left with 48,246 unique perfumes, 531 unique notes, and 65 unique accords.

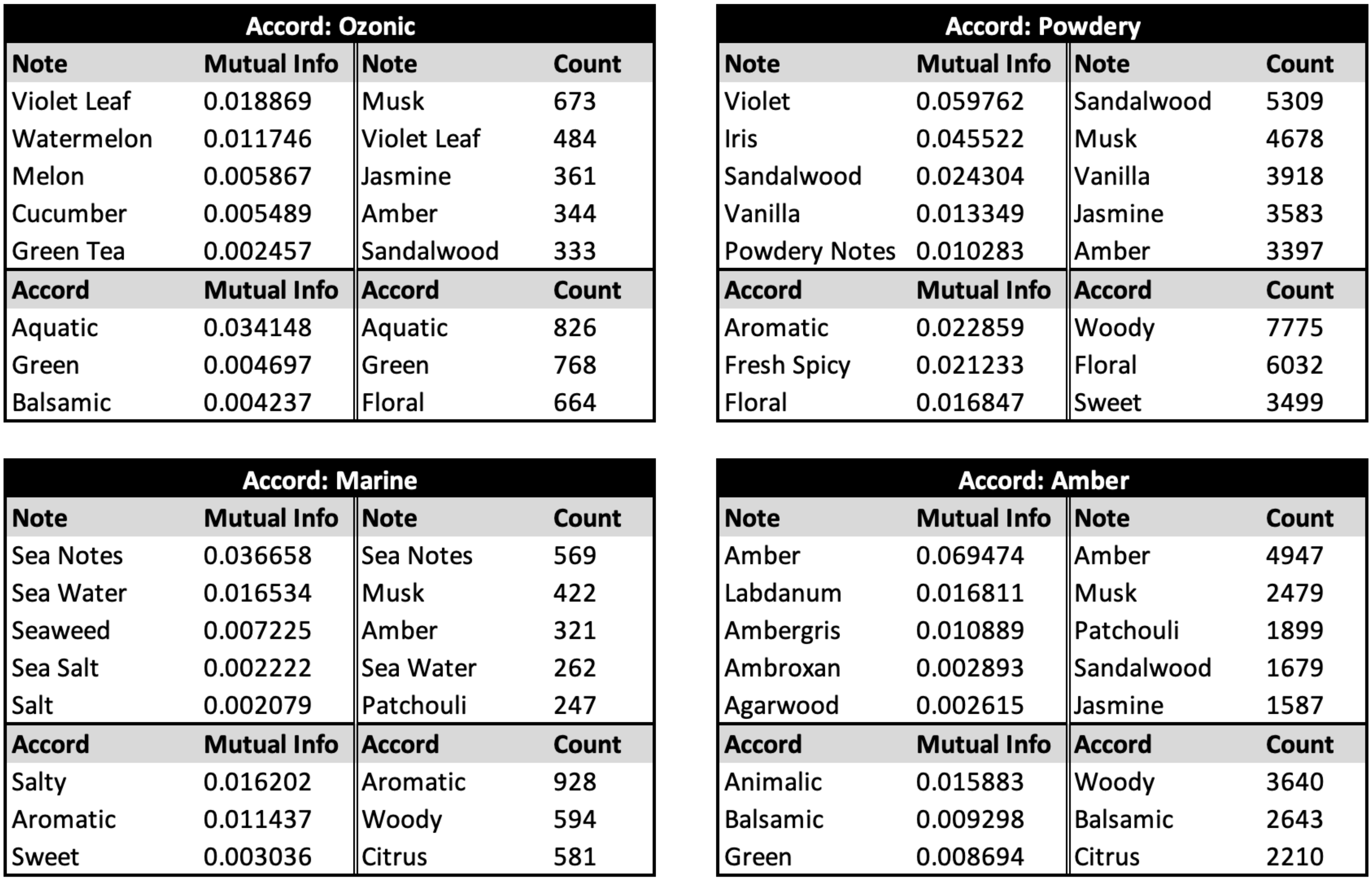
3.2 Data Exploration

Mutual Information

Mutual information was first calculated between each note/accord pair using the *mutual_info_score* function from the *sklearn.metrics* library. Similarly, mutual information was calculated between each pair of accords. Then, for each accord, notes and accords were ranked by mutual information score. The five notes and three accords with the highest mutual information score are presented for four sample accords in Table 1.

Additionally, the number of observed cooccurrences for each note/accord and accord/accord pair were counted. In other words, for accord i and accord j, how many perfumes contained both accords? Cooccurrences between notes and accords were ranked for each accord and the counts are displayed alongside the mutual information score in Table 1.

Table 1. Mutual information of perfume accords and notes:



Accord: Ozonic			
Note	**Mutual Info**	**Note**	**Count**
Violet Leaf	0.018869	Musk	673
Watermelon	0.011746	Violet Leaf	484
Melon	0.005867	Jasmine	361
Cucumber	0.005489	Amber	344
Green Tea	0.002457	Sandalwood	333
Accord	**Mutual Info**	**Accord**	**Count**
Aquatic	0.034148	Aquatic	826
Green	0.004697	Green	768
Balsamic	0.004237	Floral	664

Accord: Powdery			
Note	**Mutual Info**	**Note**	**Count**
Violet	0.059762	Sandalwood	5309
Iris	0.045522	Musk	4678
Sandalwood	0.024304	Vanilla	3918
Vanilla	0.013349	Jasmine	3583
Powdery Notes	0.010283	Amber	3397
Accord	**Mutual Info**	**Accord**	**Count**
Aromatic	0.022859	Woody	7775
Fresh Spicy	0.021233	Floral	6032
Floral	0.016847	Sweet	3499

Accord: Marine			
Note	**Mutual Info**	**Note**	**Count**
Sea Notes	0.036658	Sea Notes	569
Sea Water	0.016534	Musk	422
Seaweed	0.007225	Amber	321
Sea Salt	0.002222	Sea Water	262
Salt	0.002079	Patchouli	247
Accord	**Mutual Info**	**Accord**	**Count**
Salty	0.016202	Aromatic	928
Aromatic	0.011437	Woody	594
Sweet	0.003036	Citrus	581

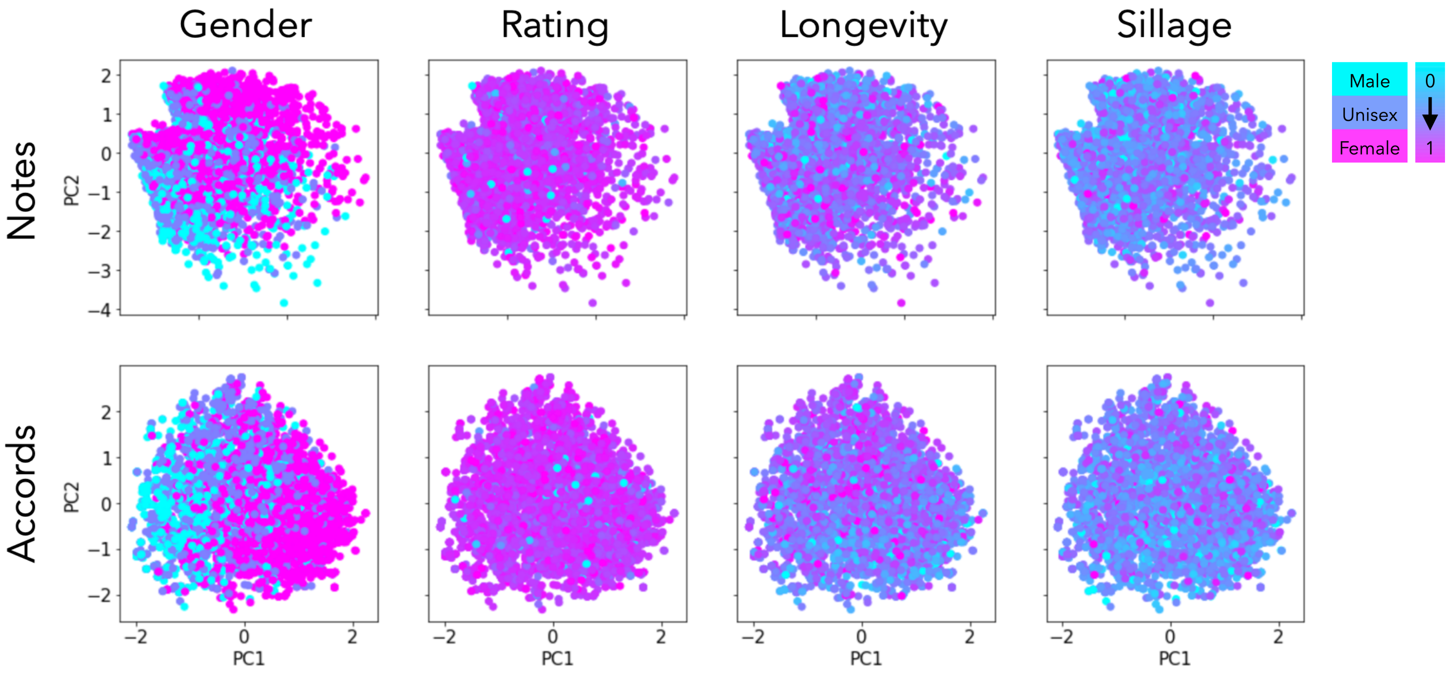
Accord: Amber			
Note	**Mutual Info**	**Note**	**Count**
Amber	0.069474	Amber	4947
Labdanum	0.016811	Musk	2479
Ambergris	0.010889	Patchouli	1899
Ambroxan	0.002893	Sandalwood	1679
Agarwood	0.002615	Jasmine	1587
Accord	**Mutual Info**	**Accord**	**Count**
Animalic	0.015883	Woody	3640
Balsamic	0.009298	Balsamic	2643
Green	0.008694	Citrus	2210

Notice that for each accord shown, the items ranked by mutual information differ from the items ranked by cooccurrence. The presence or (lack of presence) of items ranked by mutual information in a perfume can be thought of as good predictors for the accord, whereas the items ranked by count might not provide as much information. For example, most of the accords in Table 1 have Musk, Amber, Jasmine, and Sandalwood listed as some of the most frequently occurring notes. These notes are most likely common in perfumery as a whole and not necessarily indicators of a specific accord.

Dimensionality Reduction

Next, principal component analysis was used to construct 2-dimensional representations of the 531-dimensional note structure and 65-dimensional accord structure. 5,000 sampled perfumes were plotted using the top two principal components (PC1 and PC2) for both the note and accord representations. Note, perfumes were randomly sampled among those with non-zero vote counts since perfumes with no votes would not have meaningful ratings. Each representation is plotted against gender, rating, longevity, and sillage. The categorical variable gender is represented with magenta (female), purple (unisex), and cyan (male) plot points, whereas the continuous features span the color space from cyan (0) to magenta (1). The eight corresponding plots are shown in Figure 1.

Figure 1. PCA dimensionality reduction plots:



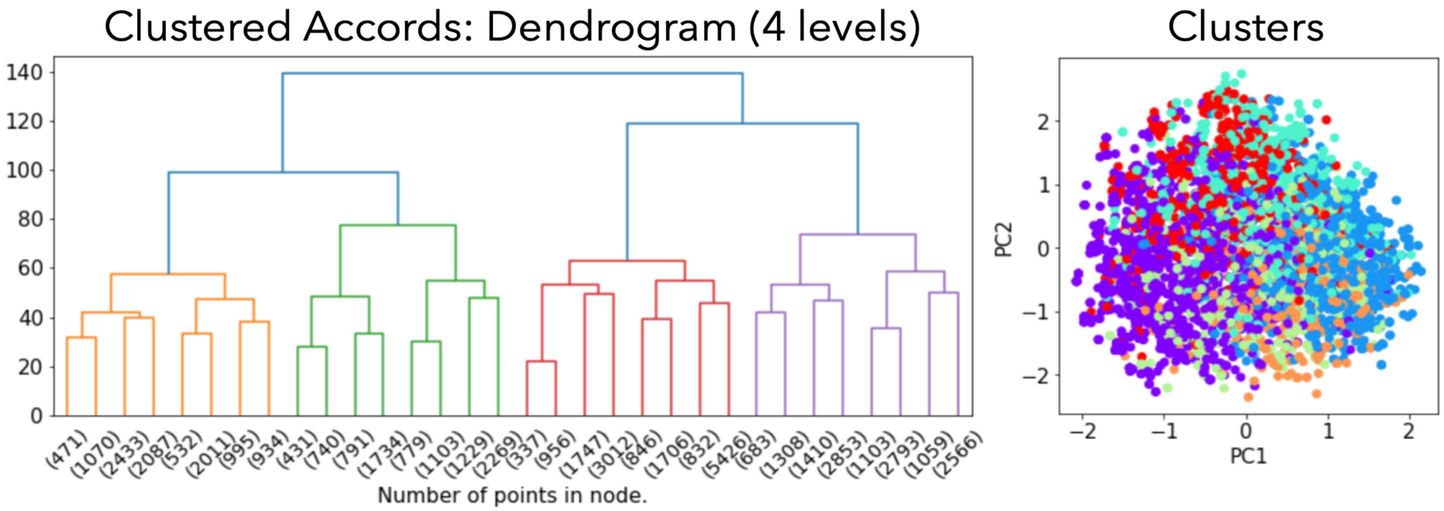
From visual inspection, gender appears to be relatively separated between male and female across the accord and note representations. This indicates that a perfume's set of notes and accords captures a fair amount of information regarding its target gender. Thus, gender might be a good candidate for prediction given notes and accords. The continuous features do not exhibit such a striking separation, though if those features are in fact relatively linearly separable, the separations might not be captured by the two principal directions shown.

Data Augmentation

This section of the data exploration shifts the focus towards the main goal of this project: predicting accords given a set of notes, a multilabel classification problem. As an intermediate step to differentiate between multiclass and multilabel classification models, an attempt will be made to predict fragrance styles from fragrance notes. The six generally accepted styles of perfume are floral, green, aquatic, citrus, fruity, and gourmand [7]. Unfortunately, the source data does not indicate style. As a workaround, the accord data was used to cluster perfumes into six groups. While these groups might not actually indicate style, the clusters will be treated as style proxies to be predicted from notes, a multiclass classification problem. Again, the goal for this part is to demonstrate the differences between multilabel and multiclass classification models, not to necessarily gain insights into the data. Under the assumption that there exists a statistical relationship between the note and accord data, the hope is that the clustering should be able to capture enough information about the accord data in order for a meaningful relationship to be modeled between the clusters and the note data.

The perfumes were hierarchically clustered by accord using the *AgglomerativeClustering* function from the *sklearn.cluster* library. The model was generated using Ward's minimum variance method. At a high level, the data begins with each point belonging to a separate cluster. At each step of the recursive algorithm, clusters are merged such that the within-cluster variance is minimized. This process repeats until the algorithm has grouped the data into the specified number of clusters [8]. The within-cluster variance was calculated using Euclidean distance between pairs of points. The hierarchy is displayed graphically as a dendrogram diagram and the six clusters are displayed with varying colors on the PCA representation of the cluster data in Figure 2.

Figure 2. Hierarchical clustering of perfume accords



4 Multiclass Classification

4.1 Overview

Binary classification methods attempt to classify data with the property that each data point belongs to exactly one of two classes. Multiclass classification methods generalize this approach to include data points belonging to exactly one of *many* potential classes. For example, let there be n observations of m features with L observed classes. Then each observation can be notated as $(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{y}_{\mathrm{i}})$ where $\boldsymbol{x}_{\mathrm{i}}$ is a $(1 \times \mathrm{m})$ feature vector and $\boldsymbol{y}_{\mathrm{i}}$ is a $(1 \times \mathrm{L})$ one-hot-encoded vector of the observed class such that $y_{\mathrm{ij}} = 1$ if $\boldsymbol{x}_{\mathrm{i}}$ belongs to class j and $y_{\mathrm{ij}} = 0$ otherwise for $\mathrm{j} = 1, \dots, \mathrm{L}$. Since each observation can only belong to one class, $\sum_{\mathrm{j}=1}^{\mathrm{L}} \mathrm{y}_{\mathrm{ij}} = 1$ for $\mathrm{i} = 1, \dots, \mathrm{n}$. The set of observations can be expressed in terms of matrices $\boldsymbol{X}_{\mathrm{n}\times\mathrm{m}}$ and $\boldsymbol{Y}_{\mathrm{n}\times\mathrm{L}}$. A multiclass classifier $\boldsymbol{\mathcal{C}}$ trained on $\boldsymbol{X}$ and $\boldsymbol{Y}$ is used to predict a $(1 \times \mathrm{L})$ vector of labels $\boldsymbol{z}_{\mathrm{i}}$ from a new set of features $\boldsymbol{x}_{\mathrm{i}}^{*}$. Row $\boldsymbol{z}_{\mathrm{i}}$ will contain one value of 1 (indicating class) and $\mathrm{L} - 1$ values of 0. In matrix form, this is notated $\boldsymbol{\mathcal{C}} : \boldsymbol{X}^{*} \rightarrow \boldsymbol{Z}$. [9]

Many machine learning models inherently handle multilabel classification. This work will focus on two inherently multiclass classification models: random forest and neural network. In random forest multiclass classification, each component decision tree is grown using methods comparable to those of binary decision trees, i.e., decision rules are based on the features $\boldsymbol{X}$, and splits are determined by some impurity measure of the potentially multiple classes in the terminal nodes (leaves). For example, Gini Impurity can be extended to the multiclass case by

$$\mathrm{G}(\boldsymbol{p}) = \sum_{\mathrm{j}=1}^{\mathrm{L}} \mathrm{p}_{\mathrm{j}} \sum_{\mathrm{k}\neq\mathrm{j}}^{\mathrm{L}} \mathrm{p}_{\mathrm{k}} = 1 - \sum_{\mathrm{j}=1}^{\mathrm{L}} (\mathrm{p}_{\mathrm{j}})^{2}$$

where $\boldsymbol{p} = [\mathrm{p}_{1,}\mathrm{p}_2, \ldots, \mathrm{p}_\mathrm{L}]$ is a vector of probabilities corresponding to a leaf such that $\mathrm{p_j}$ represents the percentage of data points in the leaf belonging to class j. Prediction of a new data point $\boldsymbol{x}_\mathrm{i}$ is taken as the most represented class in $\boldsymbol{x}_\mathrm{i}$'s resultant leaf. In neural network multiclass classification, the model is trained with an output layer containing L nodes, and the prediction of a new data point $\boldsymbol{x}_\mathrm{i}$ is taken as the class corresponding to the node with the highest activation. [10]

The random forest model was generated using scikit-learn's *RandomForestClassifier* with 100 estimators (decision trees) and default parameters. The neural network model was generated using scikit-learn's *MLPClassifier* with two hidden layers of 20 nodes and default parameters.

4.2 Binary Classification Extensions

Inherently binary classifiers can be extended to support multilabel classification in a variety of ways. Two popular methods were explored: One vs. Rest and One vs. One multiclass classification. These extensions were implemented using scikit-learn's *OneVsRestClassifier* and *OneVsOneClassifier* functions from the *sklearn.multiclass* library. The functions act as wrappers to pre-instantiated binary classification models. Scikit-learn's *LogisticRegression* classifier was extended in both cases. Besides setting the maximum number of iterations to 1000, the logistic regression models were initialized using default parameters.

One vs. Rest

The One vs. Rest strategy splits the multiclass model into L independent binary classification models, one for each class. Each model is trained using the full feature set $\boldsymbol{X}$ and a response vector $\boldsymbol{y}_{.\mathrm{j}}$ where $y_\mathrm{ij} = 1$ if $\boldsymbol{x}_\mathrm{i}$ belongs to class j and $y_\mathrm{ij} = 0$ otherwise. Since the previously notated expression of multiclass classification used one-hot-encoding for the class response variable, this is equivalent to separately training binary classification models on $\boldsymbol{X}$ with each of the columns of $\boldsymbol{Y}$. Classifier $\boldsymbol{\mathcal{C}}_\mathrm{j}$ trained on feature matrix $\boldsymbol{X}$ and response vector $\boldsymbol{y}_{.\mathrm{j}}$ will be notated as $\boldsymbol{\mathcal{C}}_\mathrm{j}(\boldsymbol{X}; \boldsymbol{y}_{.\mathrm{j}})$ for $\mathrm{j} = 1, \ldots, \mathrm{L}$. For a new data point $\boldsymbol{x}_\mathrm{i}^*$, the prediction vector $\boldsymbol{z}_\mathrm{i}$ is constructed using the prediction *probabilities* of each of the binary classifiers' predictions. Specifically, the predicted class will be the class whose binary classifier had the highest prediction probability. Formally, $\mathrm{z_{ij}} \ = \ 1$ if $\mathrm{j} = \underset{\mathrm{j^*} \in \{1,\ldots,\mathrm{L}\}}{\mathrm{argmax}}\{\ \mathbb{P}(\boldsymbol{\mathcal{C}}_1 : \boldsymbol{x}_\mathrm{i}^* \to 1), \mathbb{P}(\boldsymbol{\mathcal{C}}_2 : \boldsymbol{x}_\mathrm{i}^* \to 1), \ldots, \mathbb{P}(\boldsymbol{\mathcal{C}}_\mathrm{L} : \boldsymbol{x}_\mathrm{i}^* \to 1)\}$ and $\mathrm{z_{ij}} = 0$ otherwise. [9]

Because this method uses prediction probabilities, the base binary classifier must output a probability with its classification. Logistic regression was chosen for this reason. Also, since L separate binary classifiers must be trained, this method becomes computationally inefficient with a large number of classes, or with inherently inefficient binary classifiers. [9]

One vs. One

Speaking of computational inefficiency, the One vs. One strategy splits the multiclass model into $\mathrm{k} = (\mathrm{L}^2 \text{-} \mathrm{L})/2$ independent binary classification models, this time, one for each pair of classes. The goal for each classifier is to then select a class 'winner' (or tie) from each pair of classes for each data point. The set of classifiers can be represented by $\boldsymbol{\mathcal{C}}_{\{\mathrm{a,b}\}}(\boldsymbol{X}; \xi[\boldsymbol{y}_{.\mathrm{a}}, \boldsymbol{y}_{.\mathrm{b}}])$ for all pairs $\{\mathrm{a}, \mathrm{b}\} \in \{1, \ldots, \mathrm{L}\}$. Note, the function ξ is designed to convert two binary column vectors into one vector fit for binary classification. The function is defined to operate on pairwise elements of the two vectors such that $\xi_{\mathrm{i}}[\mathrm{y_{ia}}, \mathrm{y_{ib}}] = 1$ when $\mathrm{y_{ia}} = 1$, $\xi_{\mathrm{i}}[\mathrm{y_{ia}}, \mathrm{y_{ib}}] = \text{-}1$ when $\mathrm{y_{ib}} = 1$, and $\xi_{\mathrm{i}}[\mathrm{y_{ia}}, \mathrm{y_{ib}}] = 0$ when $\mathrm{y_{ia}} = \mathrm{y_{ib}} = 0$ [11]. Since each data point belongs to a single class, $\mathrm{y_{ia}}$and $\mathrm{y_{ib}}$will never both equal 1. Now, instead of modeling $\{0,1\}$ class labels, each binary classification is modeled on $\{\text{-}1,0,1\}$ labels to account for instances where the data point belongs to neither class a nor b. For a new data point $\boldsymbol{x}_{\mathrm{i}}^{*}$, the prediction vector $\boldsymbol{z}_{\mathrm{i}}$ is constructed using the counts of class winners across each pairwise comparison of the binary classifiers' predictions. Specifically, the predicted class will be the class that had the highest number of 'wins' across the binary classifiers. Formally, $\mathrm{z_{ij}} = 1$ if $\mathrm{j} = \max_{\mathrm{j^*}}\{\operatorname{count}_{\mathrm{j^*} \in \{\mathrm{a,b}\}}(\boldsymbol{\mathcal{C}}_{\{\mathrm{a,b}\}} : \boldsymbol{x}_{\mathrm{i}}^{*} \rightarrow \{\text{-}1 \text{ OR } 1\})\}$ and $\mathrm{z_{ij}} = 0$ otherwise. [9]

This method is significantly more computationally intensive than One vs. Rest, and with a large number of classes, the event of a tie between two classes becomes increasingly likely.

5 Multilabel Classification

5.1 Overview

Multilabel classification is a method of classification that attempts to classify data where individual data points might belong to multiple classes. Multilabel classification generalizes multiclass classification in this regard. There are multiple classes (potential labels) in both methods; however, multiclass classifiers assign a single label to each data point, whereas multilabel classifiers assign a subset of the potential labels to each point. For example, let there be n observations of m features with L observed classes (labels). Then each observation can be notated as $(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{y}_{\mathrm{i}})$ where $\boldsymbol{x}_{\mathrm{i}}$ is a $(1 \times \mathrm{m})$ feature vector and $\boldsymbol{y}_{\mathrm{i}}$ is a $(1 \times \mathrm{L})$ one-hot-encoded vector of observed classes such that $y_{\mathrm{ij}} = 1$ if $\boldsymbol{x}_{\mathrm{i}}$ belongs to class j or $y_{\mathrm{ij}} = 0$ if $\boldsymbol{x}_{\mathrm{i}}$ does not belong to class j for $\mathrm{j} = 1, \dots, \mathrm{L}$. Note that a single observation can belong to multiple classes, i.e., each observation can have multiple labels. The set of observations can be expressed in terms of matrices $\boldsymbol{X}_{\mathrm{n \times m}}$ and $\boldsymbol{Y}_{\mathrm{n \times L}}$. A multilabel classifier $\boldsymbol{\mathcal{C}}$ trained on $\boldsymbol{X}$ and $\boldsymbol{Y}$ is used to predict a $(1 \times \mathrm{L})$ vector of labels $\boldsymbol{z}_{\mathrm{i}}$ from a new set of features $\boldsymbol{x}_{\mathrm{i}}^{*}$. In matrix form, this is notated $\boldsymbol{\mathcal{C}} : \boldsymbol{X}^{*} \rightarrow \boldsymbol{Z}$.

Many machine learning models inherently handle multilabel classification. This work will continue to focus on the two models used in section **4**, random forest and neural network. Here, they will be used to predict perfume accords, the labels, from perfume notes, the features.

5.2 Binary Classification Extensions

Like multiclass classification, inherently binary classifiers can be extended to support multilabel classification in many ways. Again, two popular methods were explored: multioutput classification and classifier chains. These extensions were implemented using scikit-learn's *MultiOutputClassifier* and *ClassifierChain* functions from the *sklearn.multioutput* library. Scikit-learn's *LogisticRegression* classifier was extended in both cases. Besides setting the maximum number of iterations to 1000, the logistic regression models were initialized using default parameters.

Multioutput

Multioutput classification is the simplest of the two extensions. In this extension, the training response $\boldsymbol{Y}$ is split into columns $\boldsymbol{y}_{\cdot\mathrm{j}}$ for $\mathrm{j} = 1, \dots, \mathrm{L}$. Then, L binary classification models are independently trained on $\boldsymbol{X}$, one for each of the $\boldsymbol{y}_{.\mathrm{j}}$'s. Classifier $\boldsymbol{\mathcal{C}}_{\mathrm{j}}$ trained on feature matrix $\boldsymbol{X}$ and response vector $\boldsymbol{y}_{.\mathrm{j}}$ will be notated as $\boldsymbol{\mathcal{C}}_{\mathrm{j}}(\boldsymbol{X}; \boldsymbol{y}_{.\mathrm{j}})$ for $\mathrm{j} = 1,\ldots,\mathrm{L}$. This essentially fits a separate binary classifier on the full feature set for each of the potential labels. For a new data point $\boldsymbol{x}_{\mathrm{i}}^{*}$, the prediction vector $\boldsymbol{z}_{\mathrm{i}}$ is constructed from each of the classifiers' predictions: $\mathrm{z_{ij}} = 1$ if $\boldsymbol{\mathcal{C}}_{\mathrm{j}} : \boldsymbol{x}_{\mathrm{i}}^{*} \rightarrow 1$ or $\mathrm{z_{ij}} = 0$ if $\boldsymbol{\mathcal{C}}_{\mathrm{j}} : \boldsymbol{x}_{\mathrm{i}}^{*} \rightarrow 0$ for $\mathrm{j} = 1,\ldots,\mathrm{L}$ [12].

This method is relatively easy to implement, but it does not take mutual information between classes into account. Informally, consider a scenario where an observation is 70% likely to be labeled A and 80% likely to be labeled B. For whatever reason, no response has ever been observed with both labels A and B. Without taking mutual information between label distributions into account, a multilabel model might label the observation both A and B, whereas a more informed model would have labeled the observation B and not A.

Classifier Chains

The second extension explored attempts to incorporate mutual information between label distributions into the classification. Like in the multioutput method, the training response $\boldsymbol{Y}$ is split into columns $\boldsymbol{y}_{.\mathrm{j}}$ for $\mathrm{j} = 1,\ldots,\mathrm{L}$. However, instead of L binary classification models trained independently, they are trained sequentially with each successive model incorporating the previous models' response data in its input feature data. More precisely, $\boldsymbol{\mathcal{C}}_{1}(\boldsymbol{X}; \boldsymbol{y}_{.1}) \Rightarrow \boldsymbol{\mathcal{C}}_{2}([\boldsymbol{X}, \boldsymbol{y}_{.1}]; \boldsymbol{y}_{.2}) \Rightarrow \boldsymbol{\mathcal{C}}_{3}([\boldsymbol{X}, \boldsymbol{y}_{.1}, \boldsymbol{y}_{.2}]; \boldsymbol{y}_{.3}) \Rightarrow \ldots \Rightarrow \boldsymbol{\mathcal{C}}_{\mathrm{L}}([\boldsymbol{X}, \boldsymbol{y}_{.1}, \boldsymbol{y}_{.2}, \ldots, \boldsymbol{y}_{.\mathrm{L}\text{-}1}]; \boldsymbol{y}_{.\mathrm{L}})$. In practice, the columns $\boldsymbol{y}_{.\mathrm{j}}$ are appended to $\boldsymbol{X}$ horizontally to create successive $(\mathrm{n} \times \mathrm{m} + \mathrm{j})$ input matrices for $\mathrm{j} = 1,\ldots,\mathrm{L}\text{-}1$. Now, for prediction, the response data is not available, so instead of incorporating the previous model's response data, the previous model's prediction $\mathrm{z_{ij\text{-}1}}$ is appended instead as a proxy. Therefore, given a new data point $\boldsymbol{x}_{\mathrm{i}}^{*}$, $\boldsymbol{z}_{\mathrm{i}}$ is constructed as follows: $\mathrm{z_{ij}} = 1$ if $\boldsymbol{\mathcal{C}}_{\mathrm{j}} : [\boldsymbol{x}_{\mathrm{i}}^{*}, \mathrm{z_{i1}}, \ldots, \mathrm{z_{ij\text{-}1}}] \rightarrow 1$ or $\mathrm{z_{ij}} = 0$ if $\boldsymbol{\mathcal{C}}_{\mathrm{j}} : [\boldsymbol{x}_{\mathrm{i}}^{*}, \mathrm{z_{i1}}, \ldots, \mathrm{z_{ij\text{-}1}}] \rightarrow 0$ for $\mathrm{j} = 1,\ldots,\mathrm{L}$ [12].

While this method incorporates *some* mutual information between label distributions, it is not complete. The first label is predicted blindly without any information from the other labels, whereas the last label is predicted with information from all the previous labels. Thus, the order of the chain can directly impact model performance. One approach to overcome this limitation is to ensemble many classifier chains each trained with random orderings. For the perfume accord chain classification, 100 classifier chains were trained using random orderings of the 65 possible accords. For the final prediction, $\mathrm{z_{ij}} = 1$ was accepted as a label if more than half of the ensembled models also predicted $\mathrm{z_{ij}} = 1$.

6 Performance Metrics

Multilabel Metrics

Since the output of a multilabel classification over n new observations will be a matrix $\boldsymbol{Z}_{\mathrm{n}\times\mathrm{L}}$, typical performance measures for binary classification models like accuracy, precision, recall, and F1 score will need to be extended to account for the extra dimension. First, a measure of 0/1 loss is defined:

$$\mathbf{0/1\ Loss} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathbb{I}(\boldsymbol{y}_{i.} \neq \boldsymbol{z}_{i.})$$

This is a row by row (example by example) comparison of the actual response to the predicted response that reports the percentage of examples (data points) that do not match the true response exactly. The indicator function $\mathbb{I}$ contributes a 1 to the sum if the entire row of the true response $\boldsymbol{y}_{i.}$ does not match the entire row of the predicted response $\boldsymbol{z}_{i.}$ for $\mathrm{i} = 1, \dots, \mathrm{n}$. A lower 0/1 loss score indicates better model performance; however, it does not distinguish between models that get most of the labels correct (but not all) and models that get none of the labels correct for each example. The second loss function introduced overcomes this limitation by comparing element by element:

$$\textbf{Hamming Loss} = \frac{1}{\mathrm{nL}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\sum_{\mathrm{j}=1}^{\mathrm{L}}\mathbb{I}(\mathrm{y}_{\mathrm{ij}} \neq \mathrm{z}_{\mathrm{ij}})$$

The Hamming loss metric reports the total percentage of incorrect predictions for each potential label of each example. Again, a lower Hamming loss indicates better model performance, but it does not account for sparse response matrices. Since the perfume accord data is sparse, as most perfumes are each labeled a small subset of the potential accords, a model could predict all 0’s (no labels) and still score favorably with respect to the Hamming loss metric.

Accuracy, precision, recall, and F1 scores are extended by way of averaging. First, by taking an example-based approach, each score is calculated for each row (each predicted data point) and averaged over all the rows. The next two approaches are label-based: macro and micro averaging. Macro averaging is symmetric to example-based averaging. Each score is calculated for each column (each label) and averaged over all the columns. Micro averaging considers the entire matrix at once and counts each element as a true positive (TP), true negative (TN), false positive (FP), or a false negative (FN). The scores are then calculated as they would be in the case of a binary classification. The formulas for each of these metrics are displayed in Table 2 [13].

Table 2. Multilabel classification performance metrics

	Example Based	Macro Averaged	Micro Averaged
Accuracy	$\frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{j=1}^{L}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L}[y_{ij} \vee z_{ij}]}$	$\frac{1}{L}\sum_{j=1}^{L}\frac{\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{i=1}^{n}[y_{ij} \vee z_{ij}]}$	$\frac{\sum_{j=1}^{L}\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L}\sum_{i=1}^{n}[y_{ij} \vee z_{ij}]}$
Precision	$\frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{j=1}^{L}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L} y_{ij}}$	$\frac{1}{L}\sum_{j=1}^{L}\frac{\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{i=1}^{n} y_{ij}}$	$\frac{\sum_{j=1}^{L}\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L}\sum_{i=1}^{n} y_{ij}}$
Recall	$\frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{j=1}^{L}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L} z_{ij}}$	$\frac{1}{L}\sum_{j=1}^{L}\frac{\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{i=1}^{n} z_{ij}}$	$\frac{\sum_{j=1}^{L}\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L}\sum_{i=1}^{n} z_{ij}}$
F1	$\frac{2}{n}\sum_{i=1}^{n}\frac{\sum_{j=1}^{L}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L}[y_{ij} + z_{ij}]}$	$\frac{2}{L}\sum_{i=1}^{L}\frac{\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{i=1}^{n}[y_{ij} + z_{ij}]}$	$2\frac{\sum_{j=1}^{L}\sum_{i=1}^{n}[y_{ij} \wedge z_{ij}]}{\sum_{j=1}^{L}\sum_{i=1}^{n}[y_{ij} + z_{ij}]}$

In the formulas above, the binary operator $a \wedge b$ will only contribute a 1 to the sum when $a = b = 1$. This covers TP instances. The binary operator $a \vee b$ will contribute a 1 to the sum when $a = 1$ or $b = 1$ (inclusive). This covers TP + FP + FN instances. A sum over y_{ij} counts all predicted positives (TP + FP), whereas a sum over z_{ij} counts all true instances (TP + FN). A sum over $y_{ij} + z_{ij}$ is similar to $a \vee b$, except it double counts the TP instances. The F1 score is calculated as the harmonic mean of the precision and recall score.

Macro averaged metrics do not take class imbalance into account since each class's score is given equal weight in the average, whereas micro averaged scores are calculated granularly with equal weight given to each label. Therefore, for data sets with class imbalance, micro averaged scores can be a better indicator of model performance. Example-based metrics offer a more generalized view of single instance performance since the scores are averaged over each data point [13].

Multiclass Metrics

In multiclass classification, accuracy is defined to be $(1/\mathrm{n})\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathbb{I}(\boldsymbol{y}_{i.} = \boldsymbol{z}_{i.})$, i.e., the number of correct predictions divided by the number of predicted data points. Since each response is one-hot-encoded, this equates to the percentage of exact matches between rows of the predicted and true responses. In contrast to the multilabel performance metrics, taking example-based TP, FP, TN, and FN counts will not yield pertinent information since each example (row) only contains a single label. It can also be shown that the micro averaged label-based metrics all evaluate to a single value, the accuracy score. This is because a FP or FN for one example's label will necessarily cancel out a FN or FP (respectively) for a second label, again, since each example is only assigned one label. Macro averaging remains a viable metric by calculating scores for each class and dividing by the number of classes. However, as previously discussed, the macro averaged metrics do not account for class imbalance. This is rectified by weighting each class's metric by the percentage of data points belonging to each class before averaging. The weights are represented by $1/|\mathrm{C_j}|$ where $|\mathrm{C_j}|$ is the number of examples in class j in the formulas for the weighted metrics in Table 3 [14].

Table 3. Multiclass classification performance metrics

	Macro	Weighted
Precision	$\frac{1}{\mathrm{L}}\sum_{\mathrm{j}=1}^{\mathrm{L}}\frac{\mathrm{TP_j}}{\mathrm{TP_j}+\mathrm{FP_j}}$	$\frac{1}{\mathrm{L}}\sum_{\mathrm{j}=1}^{\mathrm{L}}\frac{1}{\lvert\mathrm{C_j}\rvert}\left(\frac{\mathrm{TP_j}}{\mathrm{TP_j}+\mathrm{FP_j}}\right)$
Recall	$\frac{1}{\mathrm{L}}\sum_{\mathrm{j}=1}^{\mathrm{L}}\frac{\mathrm{TP_j}}{\mathrm{TP_j}+\mathrm{FN_j}}$	$\frac{1}{\mathrm{L}}\sum_{\mathrm{j}=1}^{\mathrm{L}}\frac{1}{\lvert\mathrm{C_j}\rvert}\left(\frac{\mathrm{TP_j}}{\mathrm{TP_j}+\mathrm{FN_j}}\right)$
F1	$\sum_{\mathrm{j}=1}^{\mathrm{L}}\frac{2(\mathrm{Precision_j}\times\mathrm{Recall_j})}{\mathrm{Precision_j}+\mathrm{Recall_j}}$	$\sum_{\mathrm{j}=1}^{\mathrm{L}}\frac{1}{\lvert\mathrm{C_j}\rvert}\left(\frac{2(\mathrm{Precision_j}\times\mathrm{Recall_j})}{\mathrm{Precision_j}+\mathrm{Recall_j}}\right)$

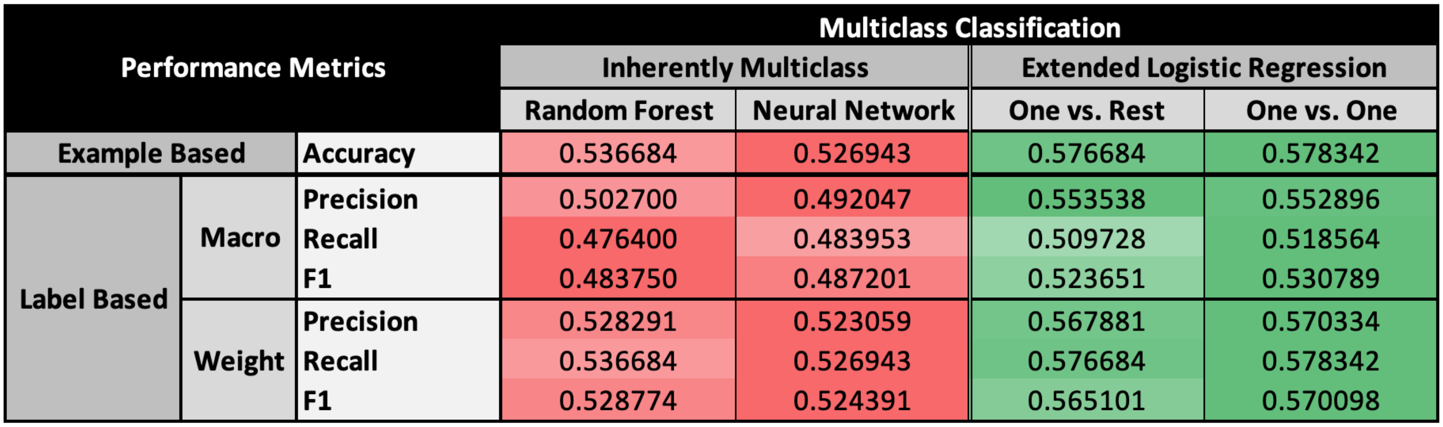
Note, in the formulas above, the TP, FP, and FN counts are indexed by class.

7 Evaluation and Final Results

Multiclass Classification Results

Each multiclass classification model's performance is shown in Table 4 across the previously defined metrics. For each metric, the model with the highest score (lowest in the case of the loss metrics) is displayed green, whereas the model with the lowest score (highest in the case of the loss metrics) is displayed red. The two models with middle performance are displayed shades of red to green depending on their spread between the lowest and highest score.

Table 4. Multiclass classification results



Performance Metrics			Multiclass Classification			
			Inherently Multiclass		Extended Logistic Regression	
			Random Forest	Neural Network	One vs. Rest	One vs. One
Example Based		Accuracy	0.536684	0.526943	0.576684	0.578342
Label Based	Macro	Precision	0.502700	0.492047	0.553538	0.552896
		Recall	0.476400	0.483953	0.509728	0.518564
		F1	0.483750	0.487201	0.523651	0.530789
	Weight	Precision	0.528291	0.523059	0.567881	0.570334
		Recall	0.536684	0.526943	0.576684	0.578342
		F1	0.528774	0.524391	0.565101	0.570098

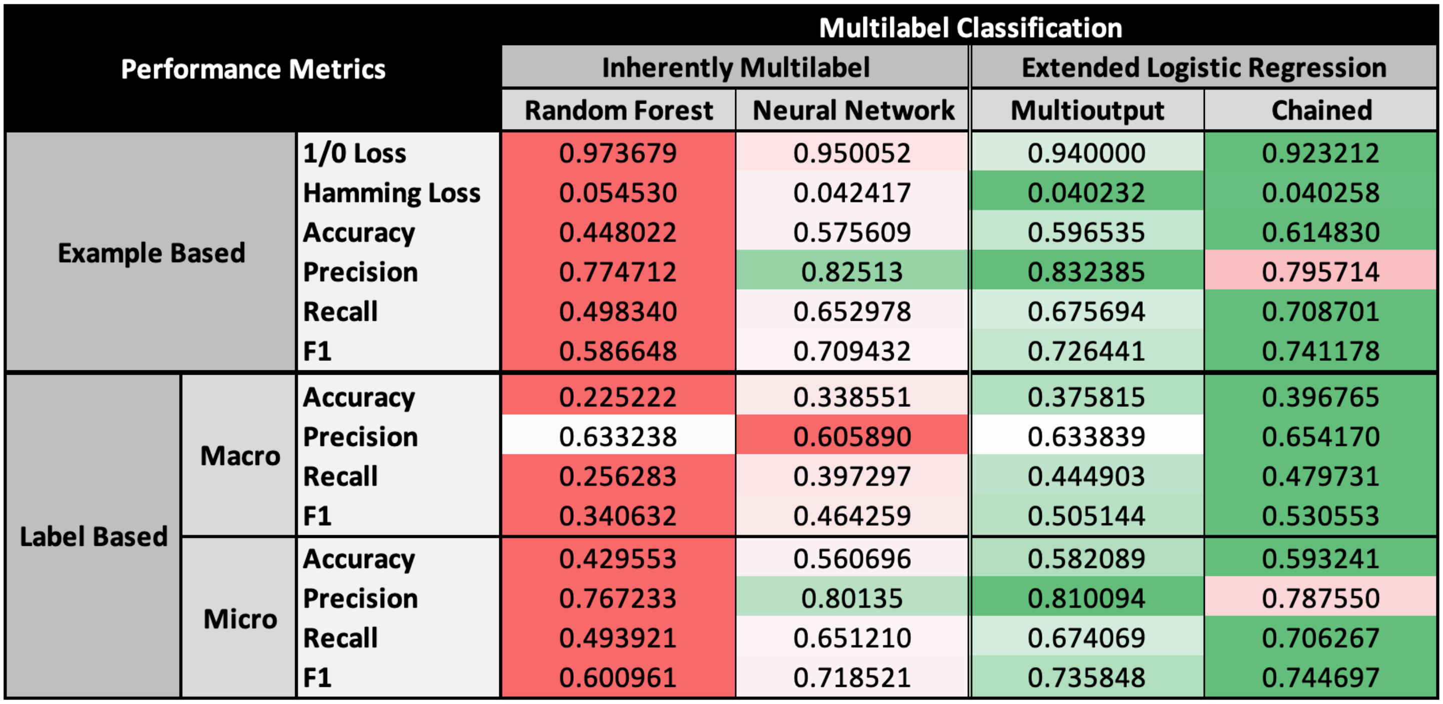
The extended logistic regression models clearly outperformed the inherently multiclass models with the One vs. One extension slightly outperforming the One vs. Rest extension. Within the inherently multiclass models, the neural network slightly outperformed the random forest in all metrics except macro averaged precision and recall. Given that the style proxy classes are imbalanced (0:14862, 1:10533, 2:7521, 3:5380, 4:3696, 5:6254), the weighted metrics are most likely more indicative of true model performance. Between the two extended logistic regression models, One vs. One slightly outperformed One vs. Rest on every metric, though the substantial increase in computational resources to perform a One vs. One extension might not be worth the slight increase in performance for production level models.

Multilabel Classification Results

Each multilabel classification model's performance is shown in Table 4 across the relevant metrics. Again, the extended logistic regression models outperformed the inherently multiclass models. Within the inherently multiclass models, the neural network clearly outperformed the random forest except in macro averaged precision. Within the extended logistic regression models, the ensembled classifier chain model outperformed the multioutput model except in example-based and micro averaged precision. For those two metrics, the ensembled chain performed worse than the neural network. This indicates that even though the ensembled chain predictions contained more correct predictions than the other models, there were also more instances of false positives predictions. Since the chained model was ensembled over 100 instances of classifier chains, this measure might be tuned by adjusting the threshold number of instances required for a prediction to be made. Similar to the case in the multiclass classification models, the ensembled chain model was by far the most computationally inefficient. Again, this information would need to be carefully considered for models in a production environment.

Notice that the 1/0 loss scores at around 95% are significantly higher than the Hamming loss scores at around 5%. This indicates that the models are not predicting the accords exactly from the notes, but on average, they are predicting at least most of the accords. Now, this is influenced by the 0 class instances due to the sparsity of the accord data, but at the very least it indicates that a non-trivial relationship can be modeled between the note and accord data. In both cases, all of the models performed significantly better than random guessing.

Table 4. Multiclass classification results



Performance Metrics			Multilabel Classification			
			Inherently Multilabel		Extended Logistic Regression	
			Random Forest	Neural Network	Multioutput	Chained
Example Based		1/0 Loss	0.973679	0.950052	0.940000	0.923212
		Hamming Loss	0.054530	0.042417	0.040232	0.040258
		Accuracy	0.448022	0.575609	0.596535	0.614830
		Precision	0.774712	0.82513	0.832385	0.795714
		Recall	0.498340	0.652978	0.675694	0.708701
		F1	0.586648	0.709432	0.726441	0.741178
Label Based	Macro	Accuracy	0.225222	0.338551	0.375815	0.396765
		Precision	0.633238	0.605890	0.633839	0.654170
		Recall	0.256283	0.397297	0.444903	0.479731
		F1	0.340632	0.464259	0.505144	0.530553
	Micro	Accuracy	0.429553	0.560696	0.582089	0.593241
		Precision	0.767233	0.80135	0.810094	0.787550
		Recall	0.493921	0.651210	0.674069	0.706267
		F1	0.600961	0.718521	0.735848	0.744697

8 Future Work

This work represents a first step towards understanding the relationship between the subjective experience of a perfume's accords and its objective note breakdown. The following questions indicate potential areas for future research:

- How accurately can the target gender be predicted from a perfume's notes/accords?
- Can note hierarchy (top, middle, base) be used to improve accord classification performance?
- To what degree can the notes/accords of a perfume influence its market potential?
- To what degree are fragrance styles influenced by a perfume's note breakdown?
- How have perfumes changed in composition over the years?
- Do certain design houses or countries favor specific combinations of fragrance notes?
- Do other neural network configurations lead to better performance in predicting accords?
- Can mutual information between response classes or some other measure be used to construct optimally ordered binary classification chains?
- Etc.

Works Cited

[1] FragranceX, "Fragrance Notes," [Online]. Available: https://www.fragrancex.com/blog/fragrance-notes/.

[2] Fragrantica, "Notes," [Online]. Available: https://www.fragrantica.com/notes/.

[3] M. Dunkel, U. Schmidt, S. Struck, L. Berger, et al., "SuperScent--A Database of Flavors and Scents," Nucleic Acids Res., 2009.

[4] Sylvaine-Delacourte, "The Accord," [Online]. Available: https://www.sylvaine-delacourte.com/en-us/guide/the-accord.

[5] sagikeren88, "Fragrances and Perfumes," *Kaggle*, [Online]. Available: https://www.kaggle.com/sagikeren88/fragrances-and-perfumes.

[6] Fragrantica, [Online]. Available: https://www.fragrantica.com.

[7] D. Schwartz, "The Fragrance Wheel Guide: From Traditional to Modern," *Dapper Confidential*, 2021. [Online]. Available: https://www.dapperconfidential.com/traditional-and-modern-fragrance-families/.

[8] J. H. Ward, "Hierarchical Grouping to Optimize an Objective Function," *Journal of the American Statistical Association,* 1963.

[9] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[10] A. Kumar, Multi-Class Classification, Carnegie Mellon University, 2015.

[11] P. Pornntiwa, E. Okafor, M. Groefsema, S. He, L. R. B. Schomaker and M. A. Wiering, "One-vs-one Classification for Deep Neural Networks," *Pattern Recognition,* vol. 108, 2020.

[12] D. Xu, Y. Shi, I. W. Tsang, Y.-S. Ong, C. Gong and X. Shen, A Survey on Multi-output Learning, arXiv:1901.00248v2, 2019.

[13] M. S. Sorower, A Literature Survey on Algorithms for Multi-label Learning, 2010.

[14] M. Grandini, E. Bagli and G. Visani, Metrics for Multi-class Classification: An Overview, arXiv:2008.05756, 2020.