Using Machine Learning to Create an e-Nose

The GitHub repository can be found here.

The models can be cloned from here:

- Model 1: Pinot & Rose.
- Model 2: Spirits
- Model 3: Spirits Tensorflow
- Model 4: Spirits Tensorflow

Introduction

This project is an attempt to create an electronic nose, or an e-Nose, to detect.

The applications of an e-Nose are broad and quite interesting: from providing diagnostic aid, such as analyzing fecal VOCs for IBS/IBD detection (Ahmed et al., 2013), inferring glucose levels from the breath ketone/acetone of diabetics (Ye et al., 2022), to facilitating quality assurance and product/brand fingerprinting where only trained experts and experienced aficionados can easily tell the difference between whiskies from their scents. (Zhang et al., 2022)

In systems of connected environments, they have been used to support sustainable harvesting and agricultural systems, such as identifying when a peach tree may be ready for harvesting! (Voss, Ricardo Antonio Ayub and Sergio Luiz Stevan, 2020)

Of course, the e-Nose architecture is not standard - it simply refers to using a chemical gas sensor to classify the nature of the substance at hand. Commercial sensors that are most sensitive towards and calibrated for a single chemical react to a wider range of chemicals, and e-Noses usually non-specific sensors with high cross-sensitivity. Metal oxide semiconductors (MOS) are widely used in air quality analysis and e-Nose functions due to their small size and ability to detect VOCs as well as CO, NO2 and NH3, however chemically sensitive conductive and non-conductive polymer films as well as quartz crystal microbalance systems have been used successfully and cater best towards VOCs and uses with

non-specific needs. (Machado et al., 2023; Yong Shin Kim et al., 2005; Julian et al., 2020)

Wilson (2013) describes the e-Nose system as such:

> A complete electronic-nose system typically consists of several integrated and/or interfaced components including a multisensor array (composed of several to many gas sensors with broad sensitivity and cross-reactivity or partially-overlapping selectivity), a data-processing and analysis unit such as an artificial neural network (ANN), software having digital pattern-recognition algorithms, and often aroma reference-library databases containing stored files with digital fingerprints of specific aroma reference (signature) patterns.

I took inspiration from Benjamin Cabe's nose and his classification of different alcohols. I originally wanted to make a model that classified coffee beans into their origin, like this very interesting project, but I didn't have coffee beans on hand (and the sensor I purchased for VOC classification and detection was stolen along with my backpack in Soho.)

Research Question

Imperceptible subtleties and patterns are picked up by machines all the time, but given a brief "sniff" exposure, can we train a model to classify "gourmet" fooditems that have a similar smell and only have minute differences in odor?

Device Overview

The E-Nose project is composed of the following primary components:

1. Gas Sensor: This is a MOS sensor, capturing six types of gases with PPM quantities. This allowed me to create a gaseous or odor fingerprint of each alcohol with the composition and variation (min, max, average, etc.) of the concentration of each gas.

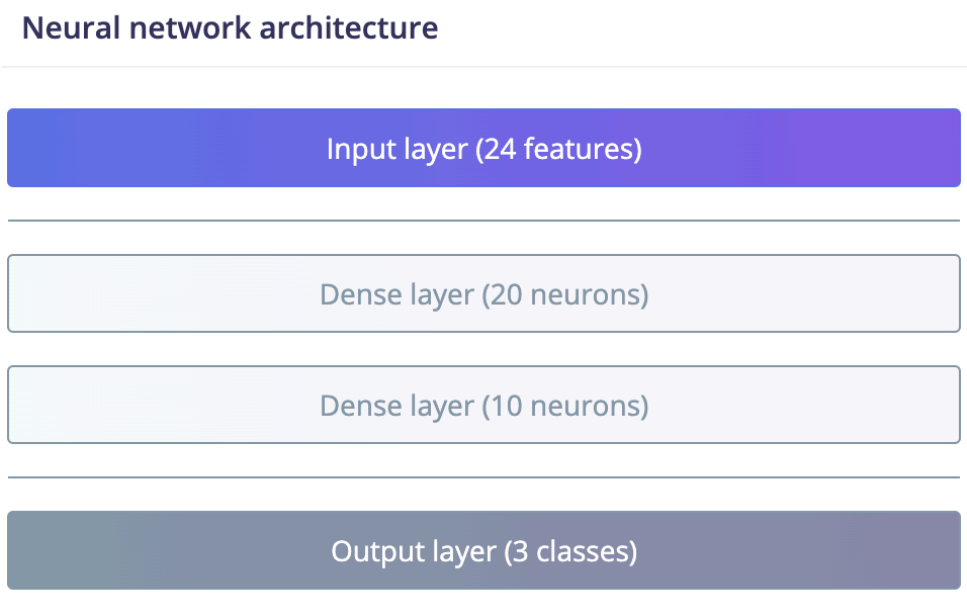
- Carbon Monoxide (CO): Range is 1 – 1000 ppm
- Nitrogen Dioxide (NO2): Range is 0.05 – 10 ppm
- Ethanol (C2H5OH): Range is 10 – 500 ppm
- Hydrogen (H2): Range is 1 – 1000 ppm
- Ammonia (NH3): Range is 1 – 500 ppm
- Methane (CH4): Range is above 1000 ppm

2. Processing Unit: An Arduino Nano 33 BLE collects the sensor data and publishes it to the serial monitor. A deep learning model is trained on this data and deployed via a TensorFlow Lite Arduino library for real-time classification of alcohol.
3. Visual Output: An LCD screen displays the classification / label of the data.
4. Enclosure: The enclosure allows airflow from the liquid, this has had several iterations. See `README.md` file in the repository for more information.

Application Overview

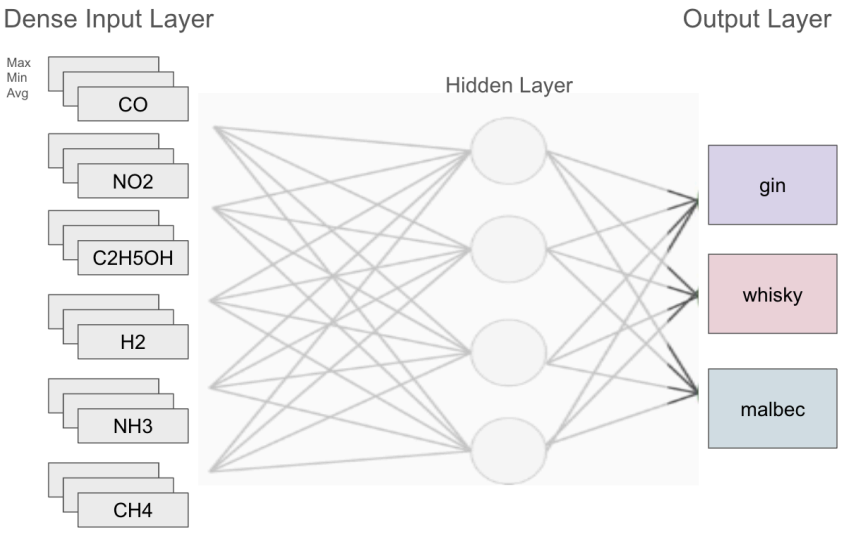
Models 1 & 2: Complex Neural Network

This model is more complex with multiple dense layers, and more controlled training parameters like batch size and determinism flags. This model is more robust but unnecessarily complex for my needs. For these models I did not use any DSP blocks because the standard Edge Impulse options ones were geared towards time-series data; I think this approach generally didn't work well because it was meant to be timeseries data.



Models 3 & 4: Softmax Regression and Multi-Layer Perceptron with final softmax

This developed from a simple neural network that has a single dense layer of inputs and uses softmax activation. It is essentially performing a logistic regression that works for multi-class classification and was suitable for my simple classification task. In model 3, I simply used regression. To enhance my model's capacity to capture complex patterns in the data, I then integrated two hidden layers with ReLu. Model Architecture diagram adapted from this Analytics Vidhya video.



Data

Data Collection

The training database uses the serial output readings of from my sensor. After initiating the sensor and collecting 25-30 mins of data per stimuli, I copied the serial monitor readings to a textfile. To ensure the data's robustness, I collected data directly from bottle, from a cup, and in different temperatures to ensure the model works in a wider range of conditions. In further experiments I used a wide mouthed jar as I felt this was most representative of use. I collected roughly equal amounts of each class to have a balanced dataset.

The variation in gas concentration allowed for improved feature selection. This serial output formatting facilitated conversion to csv:

```
[timestamp],[CO average],[CH4 average],[C2H5OH average],
[H2 average],[NH3 average],[NO2 average],[CO min],[CH4
min],[C2H5OH min],[H2 min],[NH3 min],[NO2 min],[CO max],
[CH4 max],[C2H5OH max],[H2 max],[NH3 max],[NO2 max]
```

Data Cleaning

After collection, the serial data needed to be reformatted into an csv to be readable by EdgeImpulse. This data could sometimes contain unnecessary readings like "warming up" or "0.0," which necessitated that I create a python cleaning script in codelabs.

Data Organization

I used the 60/20/20 split for training, validation, and testing data. Edge Impulse automatically did this for me or I specified it in my regression models. I stored my data in a google drive folder for easy access by colabs.

Model Iterations

Model 1: Wine Classifier Model

Test 1:

This experiment is intended to develop a workflow of collecting and reformatting data and gaining some experience with the proper parameters to train the model with. This model is meant to establish that the gas signatures can be used to classify the alcohols. This model uses:

- Rose Wine
- Pinot Grigio Wine,
- Air

Both alcohols are of the same percentage and have the same bottle. The bottles were shaken gently to promote vapor before data collection and capped until the first line of data is printed. The sensor sits on the small mouth of the bottle to limit exposure to air. Data was collected for a minimum of 10 minutes.

The features were mostly separated, and the model consistently performed very poorly on the rosé data. There was 4 minutes less data for the rose, so I believed the error could likely arise from insufficient data.

Test 2

This experiment was intended to confirm that the model can be deployed in live settings.

What I did differently?

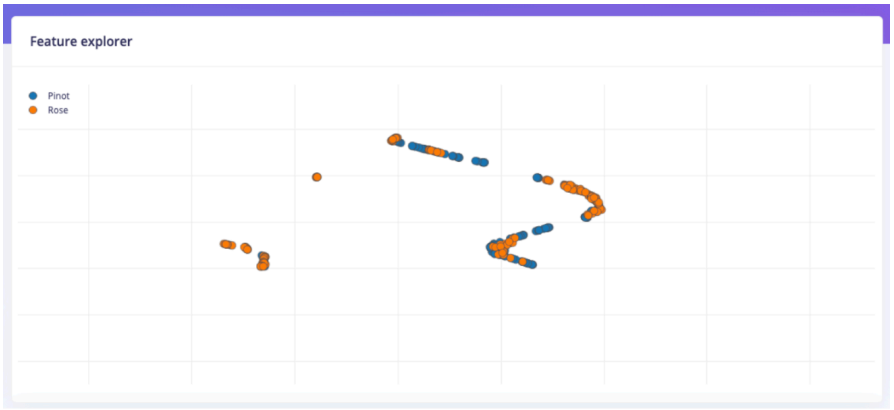
- Removed air as a control
- Increased training cycles
- Decreased learning time
- Added more data
- Modified the CSV wiizard to augment my data
- Only 12 input axes - removed 6 input axes (CH4 and NO2 data) thanks to Martin's advice!

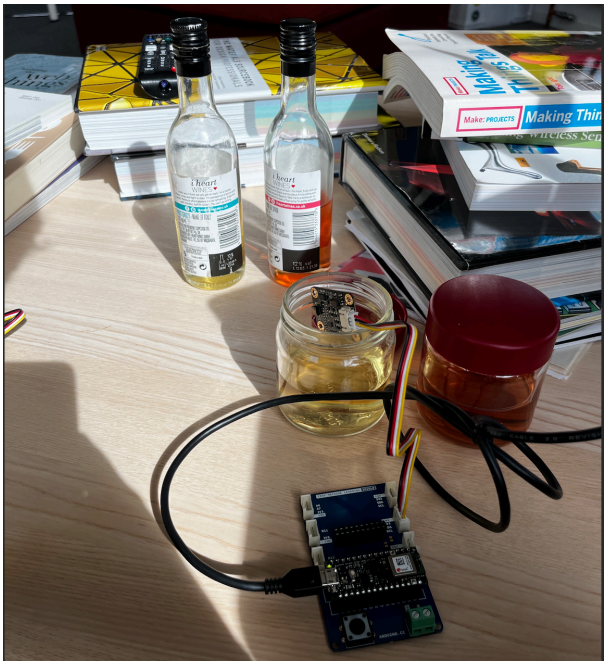
While multiple iterations on training settings, lowering the learning rate to .0001, increasing training cycles to 60, and lowering confidence threshold from 0.6 to 0.3 dramatically increased accuracy rates and lowered loss, this resulted in an overfit model - this was brought to my attention by my programme lead.

Test 3

What I did differently?

- Only 9 input axes - modifying features (CH4, NH3, C2H5O5)
- Provided more and better data (container has a wider mouth and is in partial to direct sun)





Test 4

What I did differently?

- Refined input axes, based on EI feature importance list

The outcome was worse than ever - 47.5% accuracy. I needed to pivot.

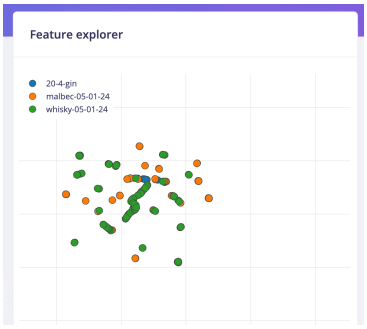
Model 2: Spirit Classifier Model (EI)

Test 1

What did I do differently?

- Different, more distinct alcohols
- Increased amount of data per class
- Updated the holder to be less precarious

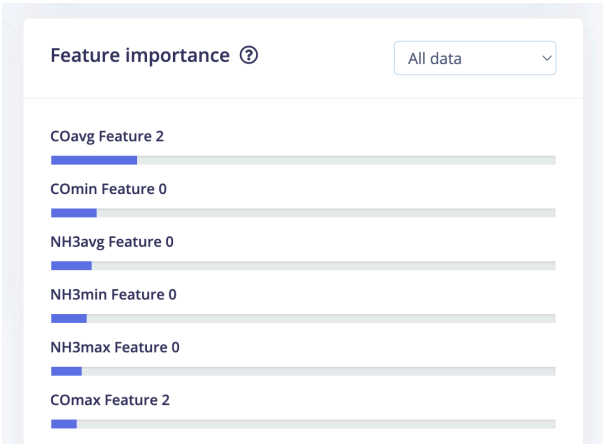
When using the feature explorer on my data, we can see that the higher percentage alcohols are poorly differentiated and all overlapping. So, as expected, this produced a highly inaccurate model.



Confusion matrix (validation set)

	20-4-GIN	MALBEC-05-01-24	WHISKY-05-01-24
20-4-GIN	**18.9%**	67.9%	13.2%
MALBEC-05-01-24	0%	**100%**	0%
WHISKY-05-01-24	17.8%	69.9%	**12.3%**
F1 SCORE	0.26	0.64	0.20

I wanted to figure out whether I could remove more axes to at least improve accuracy but the EdgeImpulse feature rankings were nondescript and unhelpful.



Upon Martin's suggestion, I decided to use some kind of regression analysis to determine which axes to remove.

Linear Regression

My goal was to use the type of beverage as a categorical dependent variable to see which features are most important in distinguising the three types of alcohol using the scikit-learn `linear_model` in google Colab. From there, my coefficient analysis showed me that my NO2 data had even

less influence on the alcohol type than my timestamps! (You can upload your files and access the script here)

Interpreting the Coefficients:

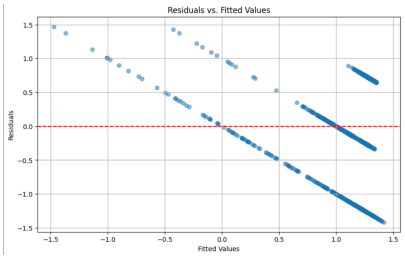
In this linear regression model, the coefficients indicate how a unit change in each feature is expected to influence the dependent variable (the type of beverage). Even just upon reviewing the data with the naked eye, it was obvious to remove CH4 and NO2 before this model. The more interesting element was the range in importance of H2 data and the relative unimportance of ethanol content.

- NH3max (0.02847629): An increase in the maximum recorded level of NH3 (ammonia) is associated with an increase in the label value.
- COavg (0.02533769): Higher average levels of CO (carbon monoxide) tend to increase the label value, indicating a possible distinguishing feature for different substances.
- NH3avg (-0.02169604): Higher average ammonia levels are associated with a decrease in the label value.
- COmax (-0.01397733): A higher maximum CO level tends to decrease the label value.
- NO2avg, NO2min, NO2max: These coefficients are extremely close to zero, suggesting that the levels of NO2 (nitrogen dioxide) have virtually no linear influence on the label in my model.

	Coefficient
NH3max	2.847629e-02
COavg	2.533769e-02
NH3avg	-2.169604e-02
NH3min	1.980907e-02
COmax	-1.397733e-02
C2H5OHmax	1.148369e-02
H2max	-1.118022e-02
C2H5OHavg	-5.219320e-03
COmin	-5.206166e-03
H2min	1.968620e-03
CH4min	-7.487104e-04
C2H5OHmin	-5.380484e-04
CH4avg	3.962340e-04
CH4max	-1.368577e-04
H2avg	5.848507e-05
timestamp	-5.273289e-05
NO2avg	1.040834e-17
NO2min	5.204170e-18
NO2max	0.000000e+00

Checking for linearity:

After reading more, I realized that using a linear regression may not be appropriate. The actual model should ideally be a "multinomial logistic regression model," as I was using more than two categories as my dependent variable. As such, I decided to check for non-linearity in my data with residual analysis to check for non-linear patterns. After fitting the model, predictions for the training data are made, and residuals (the differences between actual and predicted values) are calculated. From here, I used a Residuals vs. Fitted Values Plot to check if there are any apparent patterns in the residuals.



I think that I'm getting these three distinct lines because of the nature of my data and that using a logistic regression might better be able to find the probability of category membership.

Multinomial Logistic Regression

Multinomial logistic regression provides a more nuanced interpretation by showing how each predictor affects the log-odds of being in each specific class relative to a reference class (SPSS Analysis, 2024). I saw that a 70/30 data split was common for smaller datasets, like mine, and updated this split to use more testing data.

Interpreting the Coefficients:

Each row in the coefficient table corresponds to one class relative to the `Label 0` baseline, which I set as gin:

```
Coefficients from Multinomial Logistic Regression:
   timestamp     COavg    CH4avg  C2H5OHavg     H2avg    NH3avg  NO2avg  \
0  -0.000322 -0.001297 -0.007940  -0.004922 -0.016677  0.021463     0.0
1   0.000651  0.004650 -0.000282   0.006582  0.024992 -0.105235     0.0
2  -0.000329 -0.003353  0.008222  -0.001659 -0.008315  0.083771     0.0

       COmin    CH4min  C2H5OHmin     H2min    NH3min  NO2min     COmax  \
0 -0.025454  0.000687  -0.002574 -0.043496  0.017558     0.0  0.035959
1 -0.011464 -0.001075  -0.023938 -0.010214 -0.110952     0.0  0.002928
2  0.036918  0.000388   0.026512  0.053710  0.093394     0.0 -0.038887

      CH4max  C2H5OHmax     H2max    NH3max  NO2max
0  0.006437   0.018159  0.030694  0.028448     0.0
1  0.006193   0.008994  0.033346 -0.102723     0.0
2 -0.012630  -0.027153 -0.064040  0.074274     0.0
```

Class 0 (Gin, assuming it's the baseline):

- No coefficients since it's the baseline comparison for the other classes.

Class 1 (Whisky) vs. Class 0 (Gin):

- COavg (0.004650): Positive coefficient implies increasing CO average levels increase the odds of being whisky over gin.
- NH3avg (-0.105235): Strong negative relationship suggesting higher ammonia averages greatly reduce the odds of the sample being whisky over gin.

Class 2 (Malbec) vs. Class 0 (Gin):

- CH4avg (0.008222): Indicates an increase in methane average levels increases the likelihood of being malbec compared to gin.
- NH3avg (0.083771): Higher ammonia averages increase the odds of being malbec over gin, opposite of its effect on whisky.

Accuracy and Performance:

```
Confusion Matrix:
[[ 59  19  20]
 [  5 125  24]
 [  0  13 118]]

Accuracy Score:
Accuracy: 0.79
```

Test 2

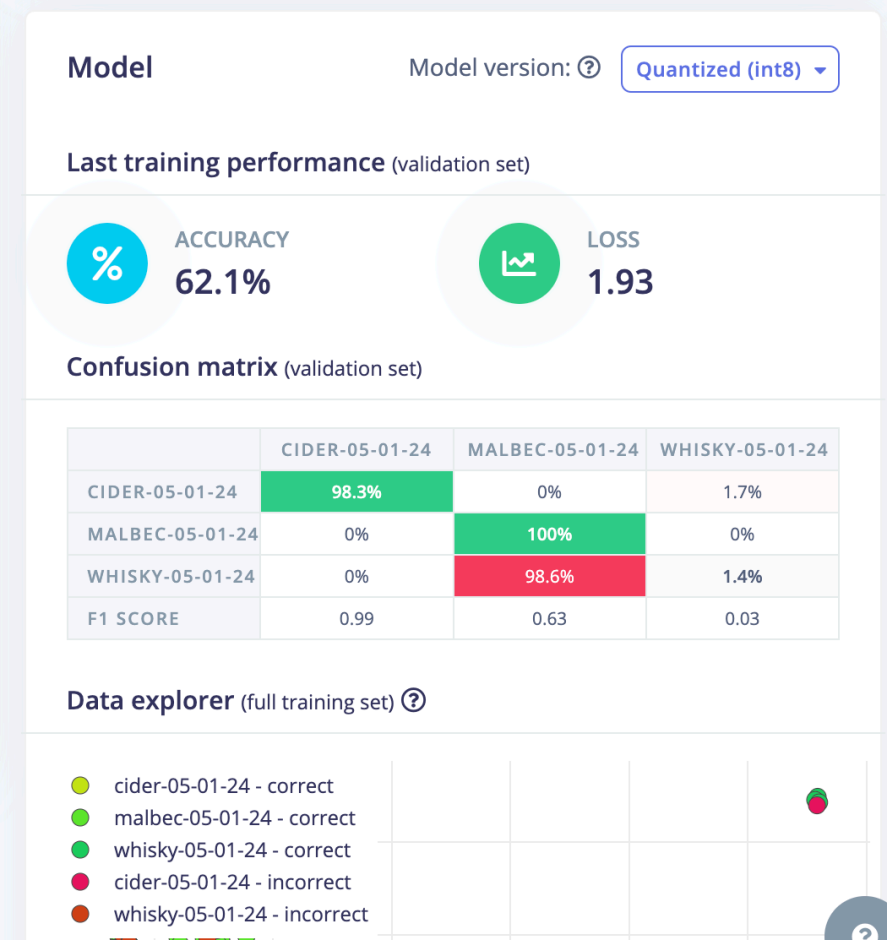
What did I do differently?

- Reduced features to 6 input axes - only used NH3(avg, max, min) and CO(avg, max, min)!

Test 3

What did I do differently?

- Maintained features at 6 input axes - only used NH3(avg, max, min) and CO(avg, max, min)!
- Added a new alcohol that looked like it had different enough readings to replace gin



While it did well on the cider, the confusion regarding whisky / malbec led me to abandon this approach.

Model 3: Spirit Classifier Model (TF lite)

I converted my logistic regression model to a Tensorflow model for potential deployment.

Model 4:

I added two deep layers to my regression model to increase the complexity.

Experiments

Model 1 - Wine

Experiments	DSP	Axes	Training Cycles	Learning Rate	Accuracy	Loss	
1	Raw Data (none)	18	30	0.0005	61.8%	13.72	
2	Raw Data (none)	12	60	0.0001	97.1%	1.16	***Overfit data***
3	Raw Data (none)	9	60	0.0002	66.1%	11.17	
4	Raw Data (none)	6	60	0.0002	47.5%	18.15	

Model 2 - Spirits EI

Experiments	DSP	Axes	Training Cycles	Learning Rate	Accuracy	Loss
1	Raw Data (none)	12	60	0.0002	47.5%	6.32
2	Raw Data (none)	6	60	0.0002	47.5%	3.12
3	Raw Data (none)	6	60	0.0002	62.1%	1.93

Model 3 - Spirit Softmax

Experiments	Model	Axes	Training Cycles	Learning Rate	Neurons	Accuracy	Loss
1	Logistic Regression	18	60	0.001 (adam)	3	69.1%	0.8425

Model 4 - Spirit MLP

Experiments	Model	Axes	Training Cycles	Learning Rate	Neurons	Accuracy	Loss
1	MLP + Logistic Regression	18	60	0.001 (adam)	10, 20	88.2%%	0.3289

Reflections and Observations

Initial tests showed promise but revealed issues like insufficient data and general confusion between certain classes, leading to poor performance. Subsequent tests improved by adjusting model parameters and training data volume, yet some models overfit despite higher accuracy in controlled conditions. A recurring theme was the challenge of discerning the root cause of various issues — whether they stemmed from sensor limitations, model constraints, or simply the need for more comprehensive data. This ambiguity often complicated the decision-making process regarding the direction of improvements and adjustments.

Pain Points 🤡

Sensors

It was difficult to determine if the poor differentiation between classes was due to the sensor's inability to capture subtle differences in gas concentrations due to the intrinsic limitations in sensitivity, specificity, and range.

Model Selection

The choice between different types of models (complex neural networks vs. simpler logistic regression) also presented challenges. Initially, it was not clear whether the complexity of the neural networks was necessary and appropriate, or if their depth and complexity were overfitting to noise in the data rather than capturing useful patterns. Simplifying the models helped in some respects but also led to doubts about whether the models were just obscuring errors now that I wasn't interacting with the model via a sleek user interface.

Wins 🙌

Incremental Complexity

This approach to problem solving was a success - starting with the simplest model that could possibly work and gradually increasing complexity between Models 3 and 4 allowed me to better understand the impact of the hidden layers. Moving from complex neural networks to simpler models like softmax regression and multilayer perceptrons also meant I selected approaches that were more appropriate for the nature of the classification task. The adaptations in models 3 and 4, which included adding hidden layers and adjusting network structures to better handle the data complexity, showed improved performance and were steps in the right direction for enhancing accuracy.

Data Collection Methodology

The methodology for data collection, especially the variety in sampling conditions (different containers and temperatures), was successful in gathering a diverse set of data points which enriched the dataset's robustness.

Data Cleaning Script

One of the standout successes of the E-Nose project was the significant growth in technical skills and confidence, particularly in areas that were initially intimidating. Prior to this project, Python was a daunting tool. The necessity to clean and preprocess the data for modeling led to the development of a Python script capable of transforming raw sensor outputs into structured CSV files. The successful creation and implementation of the data cleaning script and learning to using colabs felt like a significant milestone.

Broader Exposure to ML Concepts

This project served as a crash course in various machine learning concepts and techniques, from feature selection and model optimization to understanding different types of regression and their applications in classification tasks.

Future Directions

Moving forward, I would try to make the data forwarding function work or use a serial reader to move data from my serial monitor to a CSV file. I

would also try to collect data with a different sensor to see if this was the issue. Obviously, I'd increase my data volume to refine the models further. I'd also like to refine my enclosure - I bought a fan and hollowed out a medical model of a silicone nose to try and make a creepy handheld device, but the sensor simply could not pick up data through the nose without suction or external air flow.

Bibliography

1.Ahmed, I., Greenwood, R., Ben, Ratcliffe, N.M. and Probert, C.S. (2013). An Investigation of Fecal Volatile Organic Metabolites in Irritable Bowel Syndrome. PloS one, 8(3), pp.e58204–e58204. doi:https://doi.org/10.1371/journal.pone.0058204. 2. Cabé, B. (2021). How I Built a Connected Artificial Nose (and How You Can Too!). Benjamin Cabé. Available at: https://blog.benjamin-cabe.com/2021/08/03/how-i-built-a-connected-artificial-nose. 3. Hampson, M. (2022). This E-Nose Sniffs Out the Good Whiskey. IEEE Spectrum. https://spectrum.ieee.org/electronic-nose-whiskey. 4. Julian, T., Shidiq Nur Hidayat, Aditya Rianjanu, Agus Budi Dharmawan, Hutomo Suryo Wasisto and Kuwat Triyana (2020). Intelligent Mobile Electronic Nose System Comprising a Hybrid Polymer-Functionalized Quartz Crystal Microbalance Sensor Array. ACS omega, 5(45), pp.29492–29503. doi:https://doi.org/10.1021/acsomega.0c04433. 5. SPSS Analysis (2024). Multinomial Logistic Regression in SPSS - Explained with Example. Statistical Analysis Services For Academic Researches. Available at: https://spssanalysis.com/multinomial-logistic-regression-in-spss/. 6. Voss, J., Ricardo Antonio Ayub and Sergio Luiz Stevan (2020). E-nose Prototype to Monitoring the Growth and Maturation of Peaches in the Orchard. IEEE sensors journal, 20(20), pp.11741–11750. doi:https://doi.org/10.1109/jsen.2020.3000070. 7. Wilson, A.D. (2013). Diverse Applications of Electronic-Nose Technologies in Agriculture and Forestry. Sensors, 13(2), pp.2295–2348. doi:https://doi.org/10.3390/s130202295. 8. Ye, Z., Wang, J., Hua, H., Zhou, X. and Li, Q. (2022). Precise Detection and Quantitative Prediction of Blood Glucose Level With an Electronic Nose System. IEEE sensors journal, 22(13), pp.12452–12459. doi:https://doi.org/10.1109/jsen.2022.3178996. 9. Yong Shin Kim, Ha, S.-C., Yang, Y. and Youn Tae Kim (2005). Portable electronic nose system based on the carbon black–polymer composite sensor array. ResearchGate. https://www.researchgate.net/publication/223857478_Portable_electronic_nose_system_based_on_the_carbon_black-polymer_composite_sensor_array. 10. Zhang, W., Liu, T., Brown, A., Ueland, M., Forbes, S.L. and Su, S.W. (2022). The Use of Electronic Nose

for the Classification of Blended and Single Malt Scotch Whisky. IEEE sensors journal, 22(7), pp.7015–7021. doi:https://doi.org/10.1109/jsen.2022.3147185.

Declaration of Authorship

I, Elinor Oren, confirm that the work presented in this assessment is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

ELINOR OREN

ASSESSMENT DATE