Michael Lee

ID 009966260

July 28, 2024

Analyzing Audio Recordings of Baby Cries Using Machine Learning for Cry Baby Recognition

Part A: Letter of Transmittal

07/25/24

Jean Ashbury, Chief Executive Officer

Babies-R-Us

1 Geoffrey Way

Wayne, NJ 07470-2030

RE: Cry Baby Analyzer – Proposed Pilot Initiative

Dear Jean,

I hope this letter finds you well. As you are aware, Babies-R-Us (BRU) is seeing a decrease in overall market share due to shifts in consumer spending. Big retailers like Amazon, Temu, and Walmart continue to corner the baby market and BRU is struggling to maintain market share. We've had slow 1Q and 2Q sales and the remainder of the year is projected to be flat. To remain competitive in an everchanging, saturated market, I'm proposing that BRU create a state-of-the-art cry baby analyzer to differentiate itself from competitors and regain some of the market share.

We all know how difficult the first few months can be when having a new baby. Those first moments of life for a newborn are some of the most important and formidable for new parents, especially young adults. This period can be incredibly stressful. In order to help new parents navigate this uncharted life experience, we have the opportunity to provide a way for them to analyze why their new bundle of joy is crying and take steps to accurately address their baby's needs the first time.

This initiative has enormous upside for our organization. Once the cry baby analyzer model is generated, we have an intellectual tool that can be embedded in a number of products. We could use this in chips inside baby monitors, baby cameras, and baby monitoring apps. We also have the potential to license this intellectual property to other companies. All in all Jean, I think we can expect an ROI that's easily 10-20X in my humble estimation. If we do this right, this model will translate into brand loyalty. If we take the guesswork out of new parenting, they will trust us more with other baby products.

During my tenure as VP of Operations at Owlet, I oversaw a number of projects that focused on identifying baby comfort. These included temperature and humidity recognition as well as motion and cry detection. These relatively simple executions were incredibly profitable for Owlet and boosted sales to record levels. With further refinement and recent advances in machine learning, we have the potential to provide even more value and insight to our customers and get BRU back on track to having a great fiscal year.

I think we could get this project completely wrapped up in 3 months for less than $150k. For the pilot, we can get this done in a couple weeks for under $10,000. We may need to get a little creative for obtaining baby cry samples, but I have faith in our Marketing Dept. Let's get Legal looped in as soon as possible and see what type of incentives we could offer to parents in order for them to provide us with audio samples. For the pilot, I feel confident we have enough employees and friends of employees to source the first round of baby cry samples.

Let's circle back end of week on this. I'd love to hear your thoughts and get some feedback.

We'll talk soon,

[illegible]

Michael Lee

Chief Baby Analytics Officer

Part B: Project Proposal Plan

Project Summary

Babies-R-Us (BRU) has seen slow sales growth for 1Q and 2Q 2024. As more and more consumers turn to online shopping, senior management sees an opportunity to regain some of our market share by creating a state-of-the-art machine learning model that helps new parents understand why their baby is crying. This Cry Baby Analyzer (CBA) will be created and leveraged in numerous ways. From hardware integration within baby monitors and baby cameras to software licensing on mobile apps that focus on raising children, the CBA is expected to generate $1-2MM in new sales in 2025 alone, $100K of which is projected to be in monthly licensures of the model.

This entire project will be split into two phases:
Phase 1 – Pilot – To assess the viability of the ML model and ensure the results are generalizable.
Phase 2 – Production – Scaling up the ML model with more audio samples and deploying the model.

IMPORTANT: We will only be focusing on Phase 1 with this Project Proposal.

PROJECT DELIVERABLES

The CBA will be built using the Librosa library to convert 10-second audio files of a crying baby into Mel spectrograms (ie visual representations of the spectrum of frequencies of a signal as it varies with time) and Mel-frequency cepstral coefficients (MFCCs). These audio features will then be augmented with varying parameters including pitch shift, noise addition, volume change, time shifts, etc. to create a varied dataset that better represents real world data. These augmented audio files will then be fed into a multi-input neural network using the Tensorflow and Keras libraries to create a classification model that will predict the reason for a baby cry and/or the solution to the cry. Other notable libraries that will be used for data manipulation and visual representations include the following: sci-kit learn, numpy, matplotlib.pyplot, seaborn, and pickle. The pilot model will be available through Jupyter Notebook (ie a web application user interface) for test users to make single predictions off of new audio clips and a user guide will be made available.

By creating and integrating this model into hardware and software applications, BRU should see revenue growth of $1-2MM and recurring monthly billables of $50-100K. This pilot program is integral to the success of the overall project.

Data Summary

In order to ensure BRU creates a robust and scalable machine learning model, we will be starting with a pilot CBA using a small sample of available audio recordings of crying babies. This data will be generated by employees' children and friends / family of employees's children and collected in a private Google Drive repository. Audio files will be labeled in the following format "child_number.cry_occurrence cry_reason-cry_solution" prior to uploading. A child number corresponds to each child subject and will be kept in a table for reference. For example, "1.15 Gas_Pains-Belly_Hold.m4a" could be a file name, where '1' corresponds to a child named 'Brandon H.' and 15 represents the 15th sample for this child.

For the machine learning model to work well, each audio file needs to be trimmed down to 10 seconds. Data Analytics will manually review each audio file and split them into separate 10-second clips which capture different cry occurrences and the file name will be relabled with a set of preliminary categories for cry reason and cry solution. For multiple clips that come from one sample, Data Analytics will use the following format: "child_number.cry_occurrence.split_number cry_reason_A-cry_solution_B".
Ex. "1.15.3 Discomfort-Adjust.m4a"
After enough samples are collected, Data Analytics will perform a more thorough analysis of the file name samples to determine a list of comprehensive reasons and solutions, which capture all of the sample reasons and solutions.

After final labeling, Data Analytics will run a script to convert each audio file into a consistent wav format. From there data augmentation will occur. Each audio file will be augmented with varying parameters including pitch shift, noise addition, volume change, time shifts, etc. to create a varied dataset that better represents real world data. Data augmentation will increase the sample size 5X. The new format for augmented files is as follows: "child_number.cry_occurrence.split_number cry_reason_A-cry_solution_B_aug_aug_number".
Ex. "1.15.3 Discomfort-Adjust_aug_3.wav"

Now that the audio files are labeled correctly and data augmentation has occurred, audio files are ready for feature extraction. each audio file will be resized. Audio files will be converted into Mel spectrograms and Mel-frequency cepstral coefficients (MFCCs) for model visualization. To ensure consistent sized objects with scaled values that allow meaningful comparisons, each Mel Spectrogram and MFCC will be standardized and converted to a length of 431 time frames with 100 features extracted. This will create a batch of 3D array objects, stacked as a 4D array. Each batch of Mel Spectrograms will have a shape of 1115 (sample size) x 100 (Mel filter banks) x 431 (time frames) x 1 (single-channel audio). Each batch of MFCCs will have a shape of 1115 (sample size) x 100 (MFCC features) x 431 (time frames) x 1 (single-channel audio) with values on the same scale as the Mel Spectrograms. Each of these objects must have values on the same scale and be the same size for the model to work. Using the Pickle library, the scalers used for the Mel Spectrogram and MFCC objects are saved to allow for decoding of predictions after the model is trained.

Audio file names will also be extracted by reason and solution. Once the reason, solution, Mel Spectrogram features and MFCC features have been extracted from each audio file, reason labels and solution labels will be binarized with the MultiLabelBinarizer() function from sci-kit learn which allows each category to be properly encoded. Using the Pickle library again, both the reason encoder and solution encoder are saved to allow for decoding of predictions after the model is trained. All encoders, scalers, models, audio files, source code will be stored in the private Google Drive repository.

The data is now preprocessed and can be split into a test set and training set. Although the sample size will be small, this dataset will still give us enough variety to meet the needs of this pilot project. Data augmentation will increase the variety of our samples substantially and using multiple clips from a single cry incident should capture the subtle changes in baby cries. For the pilot, we just need to ensure that our model is better than guessing at predicting baby cries. Once we determine that it is, we can refine the model to include more features to analyze and a much bigger sample size. All of the data being provided will be voluntarily shared by employees and friends/family of employees so there are no legal or ethical concerns at this time.

Implementation

The CRISP-ML (Cross Industry Standard Process for Machine Learning) methodology will be used for this project. This methodology is an extension of the CRISP-DM methodology. It provides a structured approach for business development, implementation, and maintenance of this first pilot model. Here is an outline of the project's implementation using the CRISP-ML methodology:

- Business and Data Understanding
    - Create a preliminary category list of reasons and solutions for baby cries
    - Collect samples of baby cries and compare it to the category list for accuracy
    - Update and finalize category list based on new samples to ensure each sample is accounted for
    - Collect roughly equal samples for each reason / solution category
- Data Preparation
    - Organize, clean and preprocess audio files (eg standardization and data augmentation)
    - Extract relevant features (eg Mel Spectrogram and MFCCs)
    - Encode reason and solution labels
    - Split data into train and test sets
- Modeling
    - Select and train a multi-input neural network (CNN for Mel Spectrogram and FCNN for MFCCs)

    - Tune hyperparameters to optimize performance
- Evaluation
    - Validate model against test data (eg k-fold Cross Validation, Confusion Matrix)
- Deployment
    - Deploy model on Jupyter Labs for continued use and refinement
- Feedback and Refinement
    - Collect feedback on model performance and parameters
    - Collect additional audio samples and continue to tune hyperparameters as sample size increases

Timeline

Sprint	Start	End	Tasks
1	07/29/24	08/04/24	Technical proof of concept is presented to relevant stakeholders.
2	08/05/24	08/11/24	Project proposal submitted for review
3	08/12/24	08/18/24	Project sponsor approves charter.
4	08/19/24	08/25/24	Build out ML framework
5	08/26/24	09/01/24	Gather pilot data and build training / test sets for Cry Baby Analyzer
6	09/02/24	09/08/24	Complete initial training and validate results with project sponsor
7	09/09/15	09/15/24	Retrain / Tune models based on project sponsor feedback
8	09/16/15	09/22/24	Legal and Marketing Team approves Marketing Campaign
9	09/23/24	09/29/24	Launch Marketing Campaign (Cry Baby Sample Acquisition) Gather model data and build training / test sets for Cry Baby Analyzer
10	09/30/24	10/06/24	Gather model data and build training / test sets for Cry Baby Analyzer (cont.)
11	10/07/24	10/13/24	Complete retraining and validate results with project sponsor Final models are approved.
12	10/14/24	10/20/24	Build backend and APIs
13	10/21/24	10/27/24	Deploy model

Evaluation Plan

Continuous evaluation will be conducted throughout the entire project life cycle by the Quality Control and Data Analytics teams. During Business and Data Understanding, Data Analytics will verify the reason and solution categories and ensure 90% of audio samples fit neatly into one of the categories. If sample categorizing drops below 90%, new categories will be assessed and reverified.

During the Data Preparation phase, unit tests will be run to ensure the audio files are accurately being uploaded into the training and test sets as Mel Spectrograms and MFCCs, and corresponding reason and solution labels. Data will be pulled to check for any null data and ensure all audio files are appropriately preprocessed (100% of data will be verified).

During model creation and tuning, hyperparameters such as number of epochs, batch size, number of convolutional, pooling and dense layers, activation functions, dropout and normalization rates, will be tracked and analyzed to determine an optimal model setting. An accuracy rate of 30%+ (slightly higher than the base rate of guessing a category) is acceptable for this pilot project.

During the Evaluation phase, the model will be run with different training / test splits to find optimal data split. Additionally, k-fold cross validation will be conducted to better estimate the model's ability to generalize to unseen data. An accuracy rate of 30%+ (slightly higher than the base rate of guessing a category) is acceptable for this pilot project.

For the Deployment and Feedback & Refinement Phases, Quality Control will perform integration tests to ensure all components of the model are functioning correctly in the Jupyter Labs environment (100% integration is expected). Additionally, they will send out surveys to all participants who access and use the model via Jupyter Labs to capture any feedback and areas for improvement.

Resources and Costs

Resource	Description	Cost
Development Team – Lead ML Engineer	$150 / hour (Phase 1) 40 hours / per week / for 4 weeks (Phase 2) 40 hours / per week / for 8 weeks	 $24,000 $48,000
Marketing and Data Analytics Teams – Cry Sample Acquisition Campaign***	(Phase 2) Generate 10,000+ cry samples from 100+ babies***	$40,000

Puget Systems Multi GPU Workstation	(Phase 1) Computer to create and run machine learning models	$25,500
Quality Control / Testing	$50 / hour (Phase 1) 40 hours / per week / for 2 weeks (Phase 2) 40 hours / per week / for 4 weeks	 $4,000 $8,000
	Phase 1 Total	$53,500
	Phase 2 Total	$96,000
	Grand Total	$149,500

Part C: Application

Here is a list of all files submitted via WGU Capstone Task 2:

- Updated Capstone Topic Approval Form
- Capstone Task 2 Submission
- 'data_raw' folder – Contains all raw data prior to any splitting and categorizing. Included only for reference.
- 'data_categorized.split' folder – Contains all raw data that has been split and categorized. Used to convert audio to wav files.
- 'data_wav-converted' folder – Empty. Will be used to create augmented wav files.
- 'prediction' folder – Contains five audio samples which can be used for single predictions of the model
- 'cry_baby_analyzer.ipynb' – Jupyter Notebook which contains Machine Learning model and application

Part D: Post-implementation Report

Solution Summary

Babies-R-Us (BRU) has seen slow sales growth for 1Q and 2Q 2024. As more and more consumers turn to online shopping, senior management saw an opportunity to regain some of our market share by creating a state-of-the-art machine learning model that helps new parents understand why their baby is crying. The Cry Baby Analyzer (CBA) pilot project was created to determine a framework for deploying a fully functional model later this year.

The pilot CBA is a working proof of concept that illustrates it is possible to effectively predict why a baby is crying. This pilot allows BRU to move onto phase 2 of the project which will entail capturing a larger data set. Upon testing and tuning of the subsequent model, BRU will move onto implementation of the model which will give the company the flexibility to leverage the model in all types of hardware and software integration systems. This is expected to result in generation of $1-2MM in new sales in 2025 alone, $100K of which is projected to be in monthly licensures of the model.

Data Summary

In order to meet the needs of this Cry Baby Analyzer pilot project, 79 audio samples of baby cries were provided by a few employees and collected in a private Google Drive repository. A majority of the audio files (74 samples) came from one child of one employee who provided a plethora of different samples. Audio files used the following format "child_number.cry_occurrence cry_reason-cry_solution" prior to uploading. A child number corresponded to each child subject and was kept in a table for reference. For example, "1.15 Gas_Pains-Belly_Hold.m4a" could be a file name, where '1' corresponds to a child named 'Brandon H.' and 15 represents the 15th sample for this child.

Data Analytics manually reviewed each audio file and split them into separate 10-second clips which captured different cry occurrences. The file name was relabled with a set of preliminary categories for cry reason and cry solution. For multiple clips that came from one sample, Data Analytics used the following format: "child_number.cry_occurrence.split_number cry_reason_A-cry_solution_B".
Ex. "1.15.3 Discomfort-Adjust.m4a"

After 50 samples were collected, Data Analytics performed an analysis of the file name samples to determine a list of comprehensive reasons and solutions. The initial category list was the following:

Reasons – Hungry, Tired, Gas, Uncomfortable, Lonely

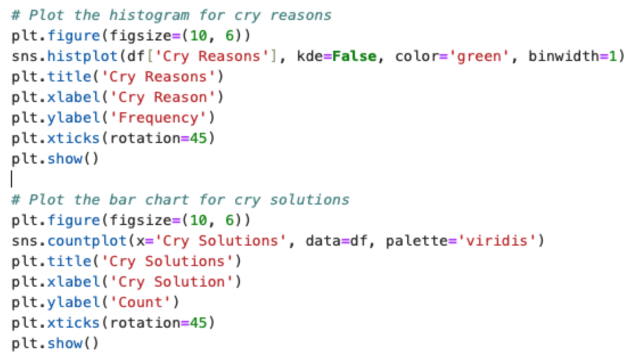
Solutions – Feed, Sleep, Adjust, Soothe, Attention

After Data Analytics validated the audio files against the dataset, however, they found a category acceptance rate of only 75% for the new samples, meaning the labels did not accurately describe all of the audio files. After all 79 samples were collected, Data Analytics created new categories to better match the audio samples. The following categories were used, with a 100% category acceptance rate:

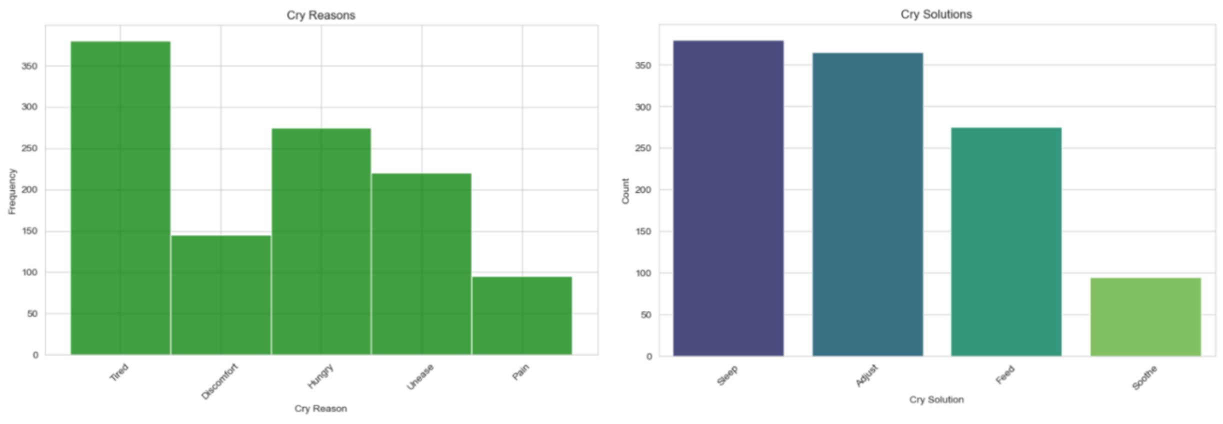
Reasons – Hungry, Tired, Unease, Discomfort, Pain

Solutions – Feed, Sleep, Adjust, Soothe

Once the categories were decided, Data Analytics continued to manually review each audio file and sliced and relabeled them into 10 second clips, creating a categorized dataset of 224 objects. After final labeling, Data Analytics converted each audio file into a consistent wav format. Audio files were then augmented with varying parameters including pitch shift, noise addition, volume change, time shifts, etc. to create a larger and more diverse dataset that better represents real world data. Each audio file produced five randomly augmented audio files, making the total dataset five times bigger (1,116 audio samples). Here is the distribution of reasons and solutions after audio augmentation:



```
# Plot the histogram for cry reasons
plt.figure(figsize=(10, 6))
sns.histplot(df['Cry Reasons'], kde=False, color='green', binwidth=1)
plt.title('Cry Reasons')
plt.xlabel('Cry Reason')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()

# Plot the bar chart for cry solutions
plt.figure(figsize=(10, 6))
sns.countplot(x='Cry Solutions', data=df, palette='viridis')
plt.title('Cry Solutions')
plt.xlabel('Cry Solution')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

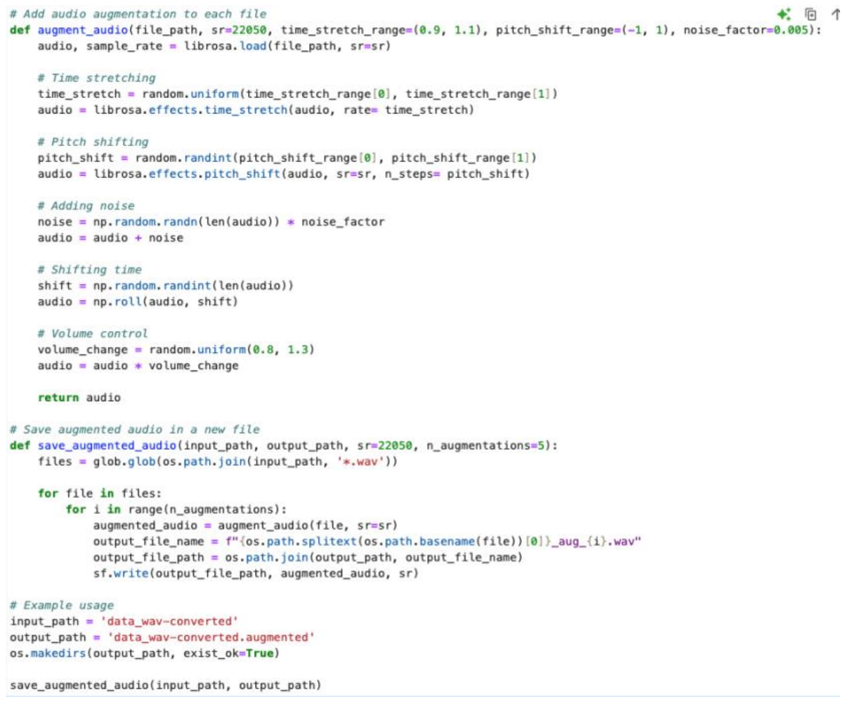


Cry Reasons and Solutions

The new format for augmented files was as follows:

“child_number.cry_occurrence.split_number cry_reason_A-cry_solution_B_aug_aug_number”.

Ex. “1.15.3 Discomfort-Adjust_aug_3.wav”

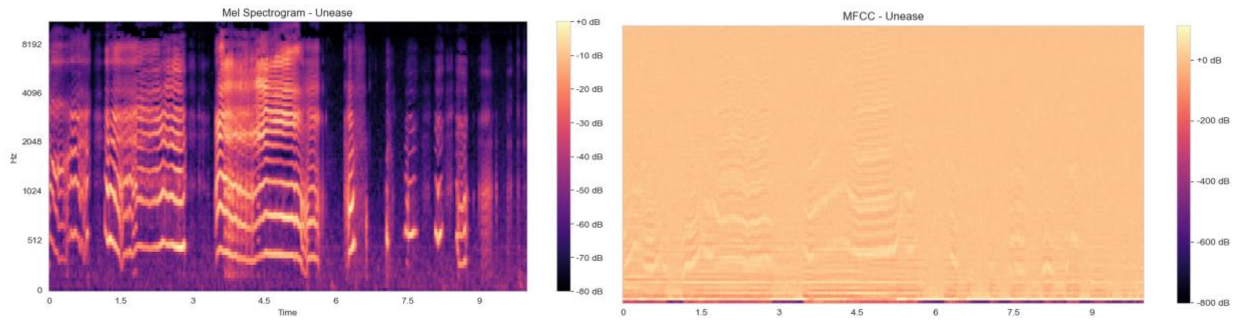


```
# Add audio augmentation to each file
def augment_audio(file_path, sr=22050, time_stretch_range=(0.9, 1.1), pitch_shift_range=(-1, 1), noise_factor=0.005):
    audio, sample_rate = librosa.load(file_path, sr=sr)

    # Time stretching
    time_stretch = random.uniform(time_stretch_range[0], time_stretch_range[1])
    audio = librosa.effects.time_stretch(audio, rate= time_stretch)

    # Pitch shifting
    pitch_shift = random.randint(pitch_shift_range[0], pitch_shift_range[1])
    audio = librosa.effects.pitch_shift(audio, sr=sr, n_steps= pitch_shift)

    # Adding noise
    noise = np.random.randn(len(audio)) * noise_factor
    audio = audio + noise

    # Shifting time
    shift = np.random.randint(len(audio))
    audio = np.roll(audio, shift)

    # Volume control
    volume_change = random.uniform(0.8, 1.3)
    audio = audio * volume_change

    return audio

# Save augmented audio in a new file
def save_augmented_audio(input_path, output_path, sr=22050, n_augmentations=5):
    files = glob.glob(os.path.join(input_path, '*.wav'))

    for file in files:
        for i in range(n_augmentations):
            augmented_audio = augment_audio(file, sr=sr)
            output_file_name = f"{os.path.splitext(os.path.basename(file))[0]}_aug_{i}.wav"
            output_file_path = os.path.join(output_path, output_file_name)
            sf.write(output_file_path, augmented_audio, sr)

# Example usage
input_path = 'data_wav-converted'
output_path = 'data_wav-converted.augmented'
os.makedirs(output_path, exist_ok=True)

save_augmented_audio(input_path, output_path)
```

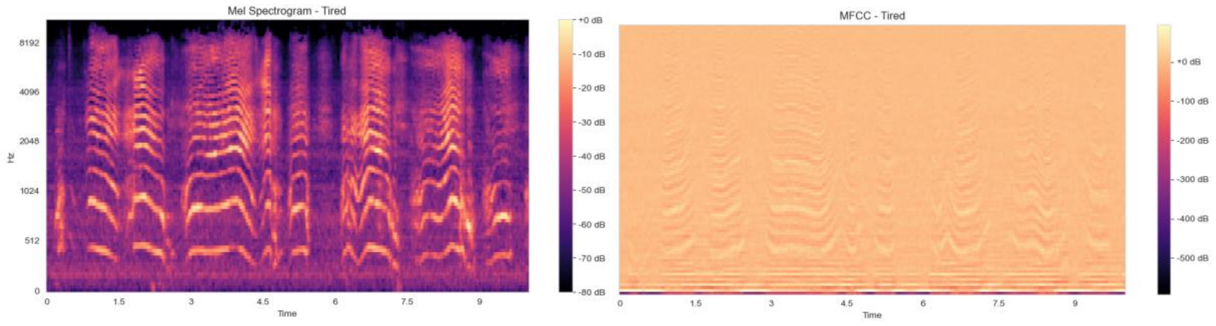
Audio Augmentation

After data augmentation, Data Analytics performed feature extraction on each audio file. Audio files were converted into Mel spectrograms and Mel-frequency cepstral coefficients (MFCCs) for model visualization. To ensure consistent-sized objects with scaled values that allowed for meaningful comparisons, each Mel Spectrogram and MFCC was standardized and converted to a length of 431 time frames with 100 features extracted. This created a batch of 3D array objects, stacked as a 4D array. Each batch of Mel Spectrograms had a shape of 1115 (sample size) x 100 (Mel filter banks) x 431 (time frames) x 1 (single-channel audio). Each batch of MFCCs had a shape of 1115 (sample size) x 100 (MFCC features) x 431 (time frames) x 1 (single-channel audio) with values on the same scale as the Mel Spectrograms. Using the Pickle library, the scalers used for the Mel Spectrogram and MFCC objects were saved to allow for decoding of predictions after the model was trained.

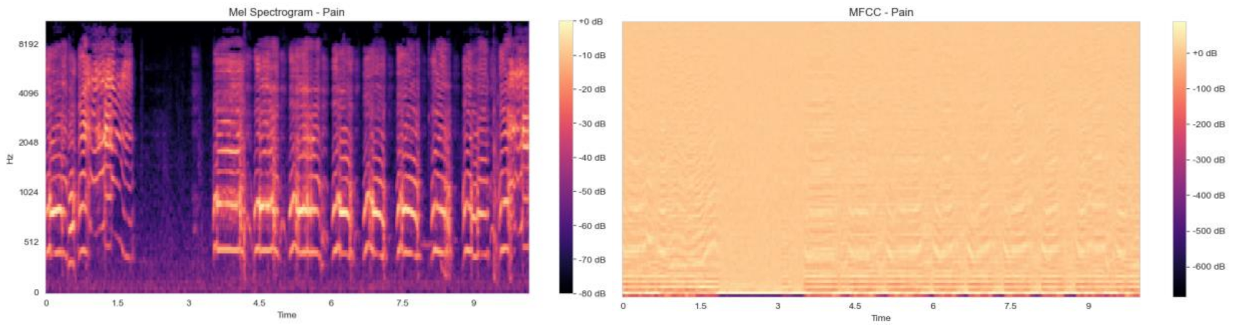
Below are examples of audio files converted into both a Mel Spectrogram and an MFCC Spectrogram for each of the five identified cry reason categories:



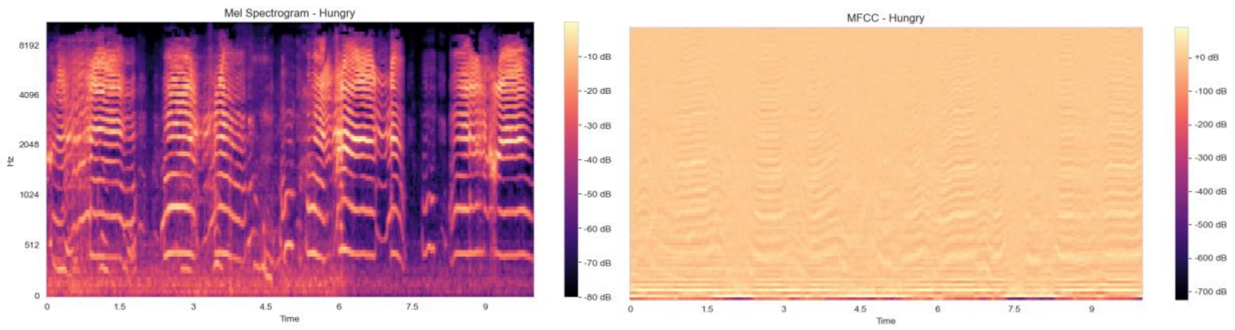
Unease



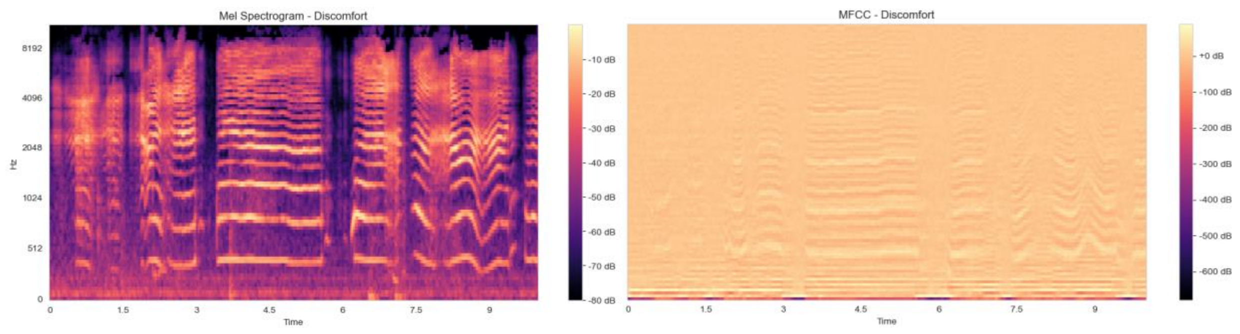
Tired



Pain



Hungry



Discomfort

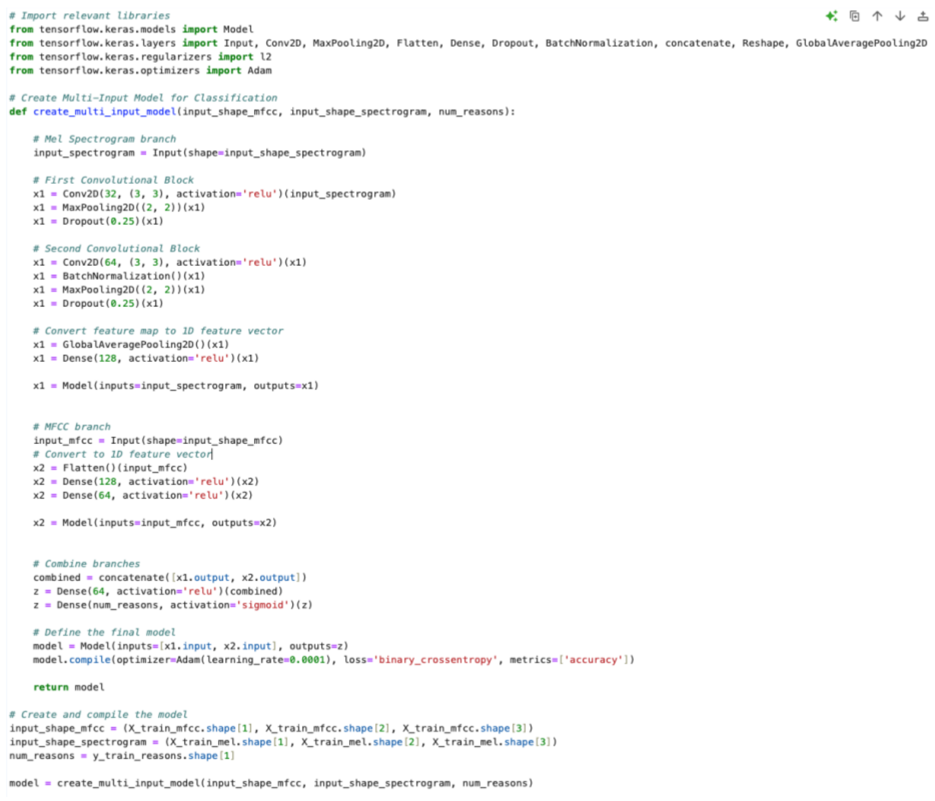
Lastly, audio file names were extracted by reason and solution and binarized with the MultiLabelBinarizer() function from sci-kit learn which allowed each category to be properly encoded. Using the Pickle library again, both the reason encoder and solution encoder were saved to allow for decoding of predictions after the model was trained. All encoders, scalers, models, audio files, and source code were stored in the private Google Drive repository.

Each audio file followed these preprocessing steps and were converted into multiple 2D array objects for reason and solution categories as well as 3D array objects with both Mel Spectrogram and MFCC Spectrogram features. The processed dataset was then split into a test set and training set and was ready for model training.

Machine Learning

A supervised classification model was generated in order to predict the reason why a baby is crying. The machine learning model is a multi-input neural network which leverages the distinct features of both the Mel Spectrogram and the MFCC Spectrogram. The model has two branches: one which uses a Convolutional Neural Network (CNN) to process the visual features of the Mel Spectrogram and a separate Fully Connected Neural Network (FCNN) to process the features extracted from the MFCC Spectrogram. These two branches are then combined and fed into two more dense layers and a final classification probability is reached. In order for this approach to work, the Mel Spectrogram and MFCC Spectrograms were converted into 4D array objects of the same size with standardized features.

Mel Spectrogram Shape: 1115 (sample size) x 100 (Mel filter banks) x 431 (time frames) x 1 (single-channel audio)
MFCC Object Shape: 1115 (sample size) x 100 (MFCC features) x 431 (time frames) x 1 (single-channel audio)



```python
# Import relevant libraries
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, concatenate, Reshape, GlobalAveragePooling2D
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam

# Create Multi-Input Model for Classification
def create_multi_input_model(input_shape_mfcc, input_shape_spectrogram, num_reasons):

    # Mel Spectrogram branch
    input_spectrogram = Input(shape=input_shape_spectrogram)

    # First Convolutional Block
    x1 = Conv2D(32, (3, 3), activation='relu')(input_spectrogram)
    x1 = MaxPooling2D((2, 2))(x1)
    x1 = Dropout(0.25)(x1)

    # Second Convolutional Block
    x1 = Conv2D(64, (3, 3), activation='relu')(x1)
    x1 = BatchNormalization()(x1)
    x1 = MaxPooling2D((2, 2))(x1)
    x1 = Dropout(0.25)(x1)

    # Convert feature map to 1D feature vector
    x1 = GlobalAveragePooling2D()(x1)
    x1 = Dense(128, activation='relu')(x1)

    x1 = Model(inputs=input_spectrogram, outputs=x1)


    # MFCC branch
    input_mfcc = Input(shape=input_shape_mfcc)
    # Convert to 1D feature vector
    x2 = Flatten()(input_mfcc)
    x2 = Dense(128, activation='relu')(x2)
    x2 = Dense(64, activation='relu')(x2)

    x2 = Model(inputs=input_mfcc, outputs=x2)


    # Combine branches
    combined = concatenate([x1.output, x2.output])
    z = Dense(64, activation='relu')(combined)
    z = Dense(num_reasons, activation='sigmoid')(z)

    # Define the final model
    model = Model(inputs=[x1.input, x2.input], outputs=z)
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

    return model

# Create and compile the model
input_shape_mfcc = (X_train_mfcc.shape[1], X_train_mfcc.shape[2], X_train_mfcc.shape[3])
input_shape_spectrogram = (X_train_mel.shape[1], X_train_mel.shape[2], X_train_mel.shape[3])
num_reasons = y_train_reasons.shape[1]

model = create_multi_input_model(input_shape_mfcc, input_shape_spectrogram, num_reasons)
```

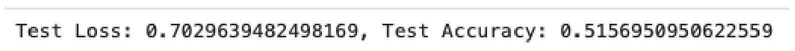
Creating a multi-input neural network

After using a model that tried predicting cry reasons based on reasons and/or solutions, it became evident that the model performed best when only looking at cry reasons. This model was created using the Tensorflow, Keras and Numpy libraries. Tensorflow and Keras libraries allowed for the seamless creation of convolutional, pooling and dense layers which are integral to neural network modeling. Numpy allowed the creation of same-sized arrays which is necessary for model inputs. For the Mel Spectrogram branch, two sets of convolutional and pooling layers (alternating) were used, followed by the flattened layer and one dense layer of 128 neurons. For the MFCC branch, two dense layers were used (128 and 64 neurons respectively). Relu activation was used for every layer except the final layer, which used sigmoid activation. Loss was calculated using binary_crossentropy.

Initially, a CNN model was created based only on the Mel Spectrograms using data augmentation on the Mel Spectrogram images, but the accuracy of the model was lacking seemingly from the small sample size and unnatural data augmentation (ie augmenting the image of the audio file instead of the actual audio file). I tried tuning all sorts of hyperparameters but the accuracy did not substantially change. Consequently, the decision was made to include the MFCC spectrogram to increase the number of individual features of each sample and reduce the errors associated with a small sample size. Additionally, audio file augmentation replaced Mel spectrogram image augmentation. By using a multi-input neural network, the model extracts the maximum data points from each audio file and can find the features which will provide the best predictions for baby cries.

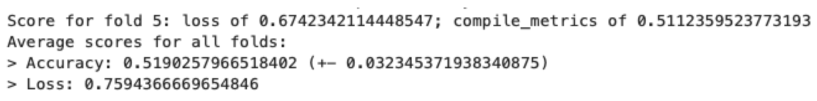
Validation

After finding a model with an acceptable accuracy rate (51.6%) that was greater than the target accuracy rate of 30%, K-Fold Cross Validation was applied to gauge the accuracy of the model to predict results with new data. After applying this validation, the overall model accuracy was found to be 51.9%. Although this accuracy is low, it is important to note that this is a pilot project and the sample size is relatively small. As more data is collected and Data Analytics further refines the sample categories, we anticipate the accuracy rate to increase substantially.



```
Test Loss: 0.7029639482498169, Test Accuracy: 0.5156950950622559
```

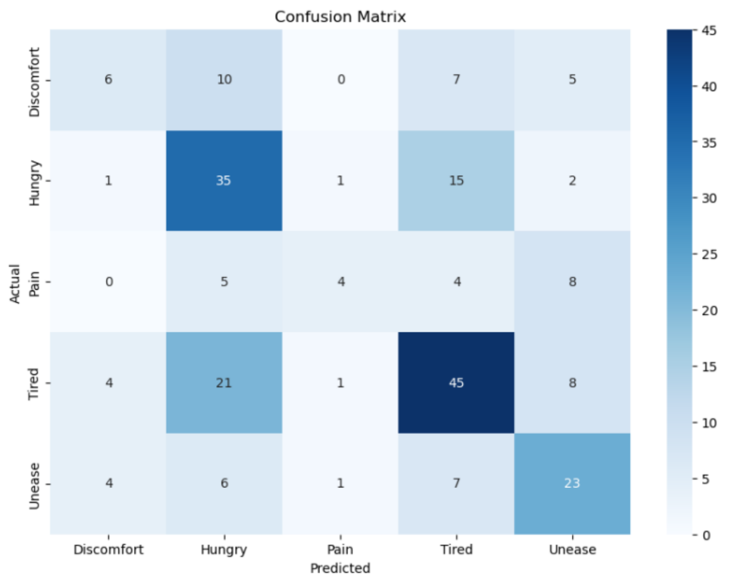
Model Accuracy after 10 Epochs



```
Score for fold 5: loss of 0.6742342114448547; compile_metrics of 0.5112359523773193
Average scores for all folds:
> Accuracy: 0.5190257966518402 (+- 0.032345371938340875)
> Loss: 0.7594366669654846
```

K-fold Cross Validation Results

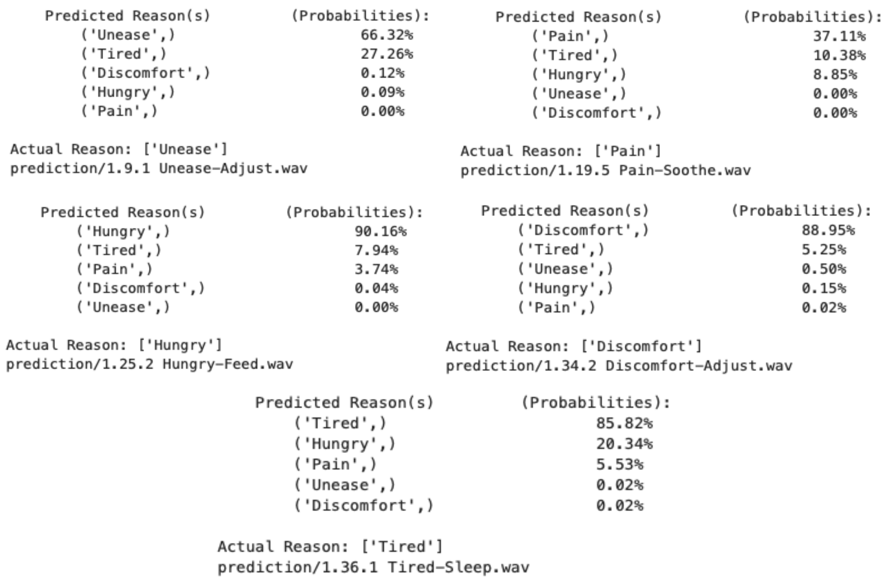
Below is a confusion matrix showing where the model is most getting confused by the labels:



Confusion Matrix

The confusion matrix indicates that the model is good at predicting Unease and consistently gets confused between Hungry and Tired labels. The model is worst at predicting Pain and Discomfort, which can most likely be attributed to the relatively small sample size. We will continue to monitor the confusion matrix as more samples are collected.

Provided below are the results of five single predictions of the model. These audio samples were withheld from the model training set and are being used to gauge the order and probability of the provided predictions. Even though the probability of the predictions are low with some audio files, overall the model is coming up with accurate predictions when compared to the audio label.



```
    Predicted Reason(s)         (Probabilities):
        ('Unease',)                     66.32%
        ('Tired',)                      27.26%
        ('Discomfort',)                 0.12%
        ('Hungry',)                     0.09%
        ('Pain',)                       0.00%

Actual Reason: ['Unease']
prediction/1.9.1 Unease-Adjust.wav
```

```
    Predicted Reason(s)         (Probabilities):
        ('Pain',)                       37.11%
        ('Tired',)                      10.38%
        ('Hungry',)                     8.85%
        ('Unease',)                     0.00%
        ('Discomfort',)                 0.00%

Actual Reason: ['Pain']
prediction/1.19.5 Pain-Soothe.wav
```

```
    Predicted Reason(s)         (Probabilities):
        ('Hungry',)                     90.16%
        ('Tired',)                      7.94%
        ('Pain',)                       3.74%
        ('Discomfort',)                 0.04%
        ('Unease',)                     0.00%

Actual Reason: ['Hungry']
prediction/1.25.2 Hungry-Feed.wav
```

```
    Predicted Reason(s)         (Probabilities):
        ('Discomfort',)                 88.95%
        ('Tired',)                      5.25%
        ('Unease',)                     0.50%
        ('Hungry',)                     0.15%
        ('Pain',)                       0.02%

Actual Reason: ['Discomfort']
prediction/1.34.2 Discomfort-Adjust.wav
```

```
    Predicted Reason(s)         (Probabilities):
        ('Tired',)                      85.82%
        ('Hungry',)                     20.34%
        ('Pain',)                       5.53%
        ('Unease',)                     0.02%
        ('Discomfort',)                 0.02%

Actual Reason: ['Tired']
prediction/1.36.1 Tired-Sleep.wav
```

Single Prediction Results

Visualizations

Visualisations have been included throughout the post-implementation report.

User Guide

Step-by-Step:

1. **Install Python 3.8 or higher**: You can download Python from python.org.
2. **Install Required Libraries using 'pip':**

   pip install numpy tensorflow librosa matplotlib scikit-learn pickle-mixin jupyterlab
3. **Install 'ffmpeg':**
   a. Download the ffmpeg executable from the FFmpeg website.
   b. Extract the downloaded zip file.
   c. Add the ffmpeg/bin directory to your system's PATH environment variable.
4. **Download the Project File (included in WGU Project Task 2 submission)**
5. **Launch Jupyter Lab** from the project directory (where the project file is saved)
6. In Jupyter Lab, navigate to the cry_baby_analyzer.ipynb notebook and open it.

7. Navigate to the Run Menu, select ‘Run All Cells’

   Note: This will take a few minutes to execute. This will create the augmented audio files in wav format, the multi-input neural network model, and train the model on the data.

8. To experiment with making a new prediction:
   a. In Jupyter Notebooks, scroll down to the bottom section called ‘Single Prediction’.
      i. Under ‘Choose Audio File to Predict’, change the audio file to get a new prediction.
   b. In the Project File that you downloaded, open the ‘prediction’ folder for a list of audio files to experiment with. Copy and paste a different audio file into the audio_file_path variable on Jupyter Notebooks.

      Important: When you copy and paste the new file name, make sure you only paste over the file name and not over the ‘prediction’ folder path

      (i.e. Only replace the underlined portion: Ex. audio_file_path = 'prediction/<u>2.1.3 Hungry-Feed.wav</u>')
   c. After uploading a new file in the audio_file_path variable, run the cell (Shift + Return)
   d. Scroll to the bottom of the cell to see the results

9. For a detailed review of the process of creating and training this model, expand each section in the notebook to review the code and comments

Reference Page

N/A